

To the Graduate Council:

I am submitting herewith a thesis written by Zeqian Shen entitled “Database Similarity Search in Metric Spaces: Limitations and Opportunities”. I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

J. Douglas Birdwell

Major Professor

We have read this thesis
and recommend its acceptance:

Tsewei Wang

John Chiasson

Accepted for the Council:

Anne Mayhew

Vice Chancellor
and Dean of Graduate Studies

Database Similarity Search in Metric Spaces: Limitations and Opportunities

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Zeqian Shen
August 2004

To my parents, Ruinian Shen and Hui Ha.

To nina, my wife.

Acknowledgments

My foremost thank goes to my adviser Dr. J. Douglas Birdwell. Without him, this thesis would not have been possible. He brought up the idea of embedding data objects in metric spaces for similarity search. His patience and encouragement carried me on through difficult times, and for his insights and suggestions that helped to shape my research skills. His valuable feedback contributed greatly to this thesis.

I am grateful to Dr. Tsewei Wang, who is a great teacher and excellent at explaining mathematical concepts and theories. She explained SVD to me, which I will never forget in my life. Moreover, I thank her for taking care of me in last two and half years.

I would also like to thank Dr. John Chiasson for serving on my committee.

I thank all the students and staffs in LIT for their efforts and assistance and a special thank goes to Mr. Zhong Tang, who helped me a lot, since I came to LIT. I enjoyed all the vivid discussions we had on various topics.

Last but not least, I thank my wife and my parents for always being there when I needed them most, and for supporting me through all these years.

This work was partially supported by the U.S. Department of Justice, Federal Bureau of Investigation under contract $J-FBI-98-083$. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Government.

Abstract

Generic database similarity search is one of the most challenging problems in current database research. Generic data are not simply structured data with several keys of numeric or alphabetic types. Traditional search algorithms that only check specified fields and keys are not effective. Similarity searches find the objects that are similar to a target using a specified similarity criterion. Tree-structured indexing techniques based on metric spaces are widely used to solve this problem. Existing methods can be divided into two categories: approaches based upon Voronoi partitions and approaches based upon reference points. The later one is the focus of this research. The problem of database similarity search using reference points in metric spaces is formulated, and the key issues are addressed. This research focuses upon two broad sets of open problems: Analysis of the limitations of approaches to similarity search using metric spaces, and development of criteria that can be used to evaluate the opportunities for new design methods.

The performance limitations of similarity search based on metric spaces are analyzed and proved to be imposed by statistical characteristics of the data collection. A new concept, *range threshold*, is defined to evaluate the feasibility of tree-structured indexing techniques based upon reference points in metric spaces. A method to estimate the range threshold is provided, which makes it possible to check the feasibility of this approach for a data set prior to implementation.

The opportunities for different approaches are evaluated by criteria based on search efficiency and utility. Comparison of different Minkowski metrics and data extraction methods using PCA (principle component analysis) are presented. Search utilities are demonstrated by examples. Several issues related to index tree structure are addressed.

Experimental results show that a taller tree yields better performance. All these results indicate that the approaches based upon reference points in metric spaces are promising.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Literature Review	4
1.3	Contributions	6
2	Problem Formulation	7
2.1	Problem of Similarity Search	7
2.1.1	Data Extraction for Generic Data	7
2.1.2	Generic Data Similarity Search	8
2.1.3	Tree-Structured Indexing	8
2.2	Approaches Based on Metric Space	9
2.2.1	Metric Space	10
2.2.2	Similarity Search in Metric Spaces	14
2.2.3	Tree-Structured Indexing in Metric Space	15
2.3	Performance Evaluation and Experimental Setup	27
2.4	Research Problems and Overview of Results	29
2.4.1	Performance Limitations	29

2.4.2	Evaluation of Performance and Utility of Data Extraction Methods and Metrics	29
2.4.3	Issues Related to Tree Indexing Structures	30
2.5	Description of Remaining Chapters	30
3	Performance Limitations	31
3.1	Overview of Performance Limitations	31
3.2	Performance Evaluation and Pruning Rate	32
3.3	Limitations on Pruning Rate and Search Range	34
3.4	Experimental Results and Discussions	40
3.5	Summary	51
4	Evaluation of Performance and Utility of Data Extraction Methods and Metrics	52
4.1	Performance Evaluation of Data Extraction Methods and Metrics	52
4.1.1	Comparison Using Different Minkowski Metrics	54
4.1.2	Comparison Using Different Data Extraction Methods	57
4.2	Utility Evaluation of Data Extraction Methods and Metrics	61
4.3	Sample Results of Image Database Search	66
4.3.1	Search Results of Different Minkowski Metrics	66
4.3.2	Search Results of Different PCA	66
4.3.3	Result Set Size Versus Search Range	78
4.4	Summary	86
5	Issues Related to Tree Index Structures	87
5.1	Overview of Issues Related to Tree Index Structures	87

5.2	Maximum Leaf Size	87
5.3	Number of Branches	88
5.4	Tree Height	92
5.5	Summary	100
6	Conclusions	101
6.1	Overview	101
6.2	Current Results	102
6.3	Future Work	104
	Bibliography	105
	Vita	111

List of Tables

3.1	Sampled Distance Standard Deviation of Data Sets	41
3.2	Estimated Values and Experimental Results of Range Threshold	51
4.1	Comparing Different Metrics by the Value of ϕ_r , $r = 4$	57
4.2	Comparing Different PCA by $\phi(r)$, $r = 4$	60
4.3	Comparing Different Combinations of Minkowski Metrics and PCA Using Different Settings by $\phi(r)$, $r = 4$	60
4.4	Utility Scores for Different Minkowski Metrics	65
4.5	Utility Scores for Different Data Extraction Methods	65

List of Figures

2.1	Samples mapped into in metric space	13
2.2	Similarity Search in Metric Space	14
2.3	A 2-level Tree Indexing Structure in a 2D Metric Space	16
2.4	Processing of Search Query (q, r)	17
2.5	The Effect of Partition Strategy on Candidate Set Size	18
2.6	Voronoi Partition in a 2D Euclidean Space	19
2.7	Partition of a 2-level Indexing Tree	22
2.8	Two Partitioning Strategies: Equal Width and Equal Percentage	23
2.9	Triangle Inequality in Similarity Search	24
2.10	Search Method Based upon Reference Points	26
3.1	Pruning Rate in a 2D Metric Space	33
3.2	Pruning Operation on Each Node	35
3.3	Search criterion Applied on Distance Histogram	36
3.4	Comparison of Pruning Rates	36
3.5	Histogram of Distance Distribution (Color Histogram of Images)	42
3.6	Average Pruning Rate vs. Normalized Search Range (Color Histogram of Images)	43

3.7	$\frac{N_{res}}{N_{total}}$ & $\frac{N_{cand}}{N_{total}}$ vs. Normalized Search Range (Color Histogram of Images)	44
3.8	Histogram of Distance Distribution (Mitochondrial DNA Sequences)	45
3.9	Average Pruning Rate vs. Normalized Search Range (Mitochondrial DNA Sequences)	46
3.10	$\frac{N_{res}}{N_{total}}$ & $\frac{N_{cand}}{N_{total}}$ vs. Normalized Search Range (Mitochondrial DNA Sequences)	47
3.11	Histogram of Distance Distribution (Synthetic Vectors)	48
3.12	Average Pruning Rate vs. Normalized Search Range (Synthetic Vectors)	49
3.13	$\frac{N_{res}}{N_{total}}$ & $\frac{N_{cand}}{N_{total}}$ vs. Normalized Search Range (Synthetic Vectors)	50
4.1	Comparing the Performance of Two Data Extraction Methods	54
4.2	Comparing Different Minkowski Metrics by $\frac{N_{res}}{N_{cand}}$	56
4.3	Singular Values Obtained by Performing SVD on the Rowing Photo Data Set	58
4.4	Comparing Different PCA by $\frac{N_{res}}{N_{cand}}$	59
4.5	Examples Photos Belong to Different Types	62
4.6	Search Results of Methods Using Different Metrics (1)	63
4.7	Search Results of Methods Using Different Metrics (2)	64
4.8	Search Results of Methods Using Different Metrics (a1)	67
4.9	Search Results of Methods Using Different Metrics (a2)	68
4.10	Search Results of Methods Using Different Metrics (b1)	69
4.11	Search Results of Methods Using Different Metrics (b2)	70
4.12	Search Results of Methods Using Different PCA (a1)	71
4.13	Search Results of Methods Using Different PCA (a2)	72
4.14	Search Results of Methods Using Different PCA (b1)	73

4.15 Search Results of Methods Using Different PCA (b2)	74
4.16 Search Results of Methods Using Different PCA (c1)	75
4.17 Search Results of Methods Using Different PCA (c2)	76
4.18 Search Results of Methods Using Different PCA (c3)	77
4.19 Number of Search Results vs. Normalized Search Range (Different Minkowski Metrics)	79
4.20 Number of Search Results vs. Normalized Search Range (Different Data Extraction Methods)	80
4.21 Search Targets	81
4.22 Search Results and Candidate Set of Different Metrics and Search Ranges Using Search Target #1	82
4.23 Search Results and Candidate Set of Different Metrics and Search Ranges Using Search Target #2	83
4.24 Search Results and Candidate Set of Different Metrics and Search Ranges Using Search Target #3	84
4.25 Search Results and Candidate Set of Different Metrics and Search Ranges Using Search Target #4	85
5.1 Number of Metric Computations under Different Search Range (Color Histogram of Images)	89
5.2 Number of Metric Computations under Different Search Range (Mito- chondrial DNA Sequences)	90
5.3 Number of Metric Computations under Different Search Range (Rowing Photos)	91

5.4	Number of Metric Computations under Different Number of Branches (Color Histograms of Images)	93
5.5	Number of Metric Computations under Different Number of Branches (Mitochondrial DNA Sequence)	94
5.6	Number of Metric Computations under Different Number of Branches (Rowing Photos)	95
5.7	Number of Metric Computations under Different Tree Height (Color His- tograms of Images)	96
5.8	Number of Metric Computations under Different Tree Height (Mitochon- drial DNA Sequence)	97
5.9	Number of Metric Computations under Different Tree Height (Rowing Photo)	98

Chapter 1

Introduction

1.1 Motivation

Support for generic data and search or ranking by similarity are two of the main problems in current database research. With the continuing development of databases and information technologies, generic data are widely used. Generic data are not simply structured data with several keys of numeric or alphabetic types. Examples include free text, images, multimedia and genetic sequences. Traditional search algorithms that only check specified fields and keys are no longer effective. New data structures and algorithms that match the requirements of generic data are required. Moreover, in the traditional approaches, a domain expert analyzes the data and specifies keys and fields. Then, searches are restricted to the specification of value sets on these structured keys and fields. For generic data, expert knowledge of a specific attribute or field is usually unavailable. Therefore, a database for generic data should be able to search without knowledge from a domain expert.

There is also a need for similarity search capabilities in databases. Similarity search

strategies are very different from traditional exact search strategies. An exact search selects and returns the record, whose keys are exactly within specified sets. However, for recent content-based search strategies, it is rare to find an exact match to these sets, especially on those generic data types mentioned above. A similarity search strategy finds the objects that are similar to a target using a specified similarity criterion. In many cases, these strategies are more practical and useful. For example, in a forensic investigation, the suspects are identified by comparing DNA sequences. Individuals whose DNA are sufficiently similar to the DNA samples collected from a crime scene cannot be excluded and remain suspects. The solution of a similarity search problem requires the choice of a proper similarity measure and design of an algorithm based on the similarity relation between data objects. Similarity search has many variations. For example, the K-Nearest Neighbor search, which is widely used in all kinds of practical applications, aims to find k most similar objects to a target [15].

Another difficult challenge is the high dimensionality of data. The dimensionality of current data is beyond the range that a traditional database approach can handle. Data with 10 dimensions, or keys, was quite high in past years; in comparison, mitochondrial DNA sequences contain more than 600 bases (corresponding to keys or dimensions). A solution that overcomes the curse of dimensionality is required.

Here are several applications where high-dimensional generic database similarity search methods are useful.

1. Genetic Sequence Database

The database contains sequences of symbols. These sequences represent DNA, RNA, or proteins. There are at least two classes of applications. One attempts to identify, for example the basis of diseases by searching for particular subsequences

or structures in large sequences. This is also known as local sequence alignment, and a well known algorithm is BLAST [28]. Another application is human identification. Individuals can be uniquely identified by their genetic fingerprints, which are sets of measurements, or sampled regions, of the DNA in their genomes. This is widely used in forensic investigations [21].

2. Multimedia Database

Multimedia databases include databases for images, videos clips, audio files and similar data [23] [38]. Voice recognition, facial recognition and fingerprint matching are applications of these technologies. The traditional way to implement these applications is selection and extraction of a few key features from the data to construct structured data. This procedure relies heavily on the help of a domain expert. Improvements in generic database technologies might allow the data to be treated as generic data that could be searched directly.

3. Medical Database

Medical databases store medical data such as images of X-rays and MRI (magnetic resonance imaging) brain scans. Identifying the cause of symptoms by searching for similar past cases in a database could be helpful for diagnosis [22] [16].

4. Geological Database

A geological database stores all types of geological data, such as satellite images, maps, and terrain features such as water or roads. An ability to locate some places by searching similar samples from these data would be valuable for many applications.

5. Time Series

Financial databases and stock price databases are two of the most widely used applications of time series. The goal is to predict the future value by finding similar past patterns. Time series data are also prevalent in engineering and scientific applications [27].

All these applications listed above could be aided by a general solution for generic data similarity search.

1.2 Literature Review

An approach to solve the problem of similarity search based on a *metric space* [12] has been studied for many years, and a lot of algorithms have been designed and proposed. The similarity search approach based on a metric space was first introduced in [9]. The *triangle inequality* [12], the core of similarity search algorithms based on metric spaces, was first used in [34]. A large number of algorithms have been implemented based on these works. All these algorithms can be divided into two categories according to the way in which they partition the metric space. Some partition the space using reference points, while others achieve that based on Voronoi partitions [2].

In the approaches based on reference points, several points in a metric space are chosen as reference points. The distances between these points and all the remaining points are calculated. Then, the metric space is partitioned according to these distances. Yianilos [36] implemented vp-tree using this idea. The detailed description of this method can be found in Section 2.2.3. In order to reduce the number of branches traversed and improve the performance, a method called the excluded middle vantage points forest was introduced in [37]. In order to reduce the number of metric calculations

needed during search processing, Yates proposed the fq-tree [25] that uses the same reference point on all the nodes at the same level of the tree. Thus, only one metric calculation is needed when a query descends one level in the tree. However, fq-tree will be taller than vp-tree and might cause longer search processing time. Yates also recommended the fhq-tree that is an improved fq-tree with fixed height [25] [4]. A solution with more than one reference point on each node was also implemented [7]. The benefit of using multiple reference points on one node is a higher fan out below the node.

One of the critical problems of the vp-tree algorithm is that it is not dynamic. This shortcoming was removed in [1]. A second problem in this approach is how to choose the reference points. A general guideline for choosing reference points was provided in [30]. It concludes that the reference points should be away from cluster centers. Several reference-selecting algorithms were compared, and the variance of the density function of distances between references and samples was used as a criterion for evaluating the performance of reference point sets in [5]. A method called the HF-algorithm was proposed in [26]. A good review and the incremental selection method are provided in [3]. Another key issue is how to select partitions. Two popular partitions, equal width and equal percentage, were reviewed and compared in [15] and [10].

Voronoi partitions [13] [2] are another way to partition a metric space. The space is partitioned into Voronoi cells with respect to a set of split points. The Voronoi cell here is defined as the set of all points that lie closer to the split point than to any other split point in the space. The GH-Tree [32], one of the methods based on Voronoi partitions, uses two split points on each tree node. Therefore, the metric space is partitioned by a generalized hyperplane. The GNAT [8] uses more than two split points to partition

the space. M-tree, implemented by Paolo Ciaccia and described in [24], is a dynamic version of GNAT.

Most of the methods mentioned above have been used in practical applications. Vp-tree is one of most often used methods. It was implemented for sequence proximity search [29] [35] and content-based image indexing [11] [19]. One of the variations of vp-tree, mvp-tree, was successfully applied to a multimedia database [6]. In MoBios (Molecular Biological Information System) [33], mvp-tree was implemented and claimed to have better performance than M-tree. For methods based on Voronoi partitions, M-tree was used for both string approximate search [20] and web image retrieval [17].

1.3 Contributions

In this research, the problem of database similarity search is formulated. Approaches based upon metric spaces are introduced. A tree-structured index based upon reference points is used. During research on different search methods, experimental results show that performance limitations exist. Some performance measures and concepts are defined to help explore the causes of performance limitations of tree-structured indexing techniques based upon metric spaces. A set of criteria are designed to evaluate the opportunities of new methods. Different metrics and data extraction methods are evaluated and compared for search efficiency and utility. The opportunities provided by different tree indexing structures are also evaluated. A taller tree yields better performance for database similarity search based upon metric spaces, and a binary tree with maximum leaf size equal to one is recommended.

Chapter 2

Problem Formulation

2.1 Problem of Similarity Search

The generic database similarity search problem is formulated in this section. Tree-structured indexing techniques are chosen as the solution. The main steps, general requirements and goals of tree-structured indexing methods are presented.

2.1.1 Data Extraction for Generic Data

Data extraction is the first and also a very critical step of generic data similarity search. An optimal data extraction method can lead to a great improvement of search performance. For example, a good data extraction method can reduce the dimensionality of data, and avoid the curse of dimensionality. Therefore, a criterion that can be used to evaluate and compare different data extraction methods is required. This problem will be covered in Chapter 4 by in-depth analysis and experiments using an image database.

2.1.2 Generic Data Similarity Search

A similarity search strategy finds all data objects more similar to a target object than a specified threshold. The first problem is how to define and evaluate the similarity between two objects. There are many ways to measure similarity, such as dice coefficient [14], overlap measure [14] and sequence alignment [18], and these methods, in general, depend upon the data. Dissimilarity measures, such as Euclidean distance and edit distance [18], are often used instead of the similarity measures. Dissimilarity measures are used much more often than similarity measures in practical applications, and this research uses dissimilarity measures as well. A proper dissimilarity measure that preserves the desired characteristics of data can help to obtain satisfactory results. For example, sequence database usually choose edit distance as the dissimilarity measure [29], while the protein sequence database chose local alignment [28]. The most straightforward search algorithm calculates the dissimilarities between all the objects and the search target. The search results are the objects within the predefined dissimilarity range. However, the time complexity of this algorithm is $O(N)$, where N is the total number of objects. Linear time complexity is unacceptable, and an algorithm with sublinear complexity is needed.

2.1.3 Tree-Structured Indexing

Using a predefined indexing structure helps to reduce the search time, because only part of the objects will be scanned. The three main steps of these algorithms are:

1. Build an indexing structure.
2. When a search request is received, derive a candidate set from the indexing structure. The candidate set excludes the objects that can not satisfy the similarity

requirement.

3. Exhaustively search the candidate set, and find the resulting objects that satisfy the similarity requirement.

Step 2 can be performed quickly, whereas the time required to perform step 3 scales linearly with the size of the candidate set. It is therefore desirable to make the candidate set as small as feasible, while pursuing the efficiency of step 2. In order to perform step 2, an indexing structure can be built based on similarity relations between objects and their preservation in the index. Similar objects should be stored close to each other in the indexing structure, such that all the objects within a specified dissimilarity to a search target can be rapidly identified. The most effective indexing structure is a tree that indexes and stores data in a hierarchy according to similarity relations between data objects. The search time of a tree-structured index is the sum of the time of descent of the tree and the time required to exhaustively search the candidate set.

2.2 Approaches Based on Metric Space

Many researchers have proposed approaches based on a metric space (see definition in Section 2.2.1). All the samples are mapped to points in a metric space. Dissimilarities between database objects are evaluated quantitatively by the metric. Similarity searches are modeled by range queries in the metric space: Find objects within a certain metric value, or distance, from a specified target. The underlying concepts are:

1. The sample space is modeled as a metric space;
2. Samples are mapped to points in the metric space;
3. Dissimilarity between two objects is quantified by the metric;

4. A similarity search becomes a problem of finding all points within a certain distance from the target point in the metric space;
5. Indexing the data is performed by partitioning the points in the metric space.

The explanation of these relations and the mathematical formulation of this approach are presented below.

2.2.1 Metric Space

Let X be an arbitrary set. A function $d: X \times X \rightarrow \mathbb{R} \cup \{\infty\}$ is a *metric* on X if the following conditions are satisfied for all $x, y, z \in X$.

$$\text{Positiveness: } d(x, y) > 0 \text{ if } x \neq y, \text{ and } d(x, x) = 0 \quad (2.1)$$

$$\text{Symmetry: } d(x, y) = d(y, x) \quad (2.2)$$

$$\text{Triangle inequality: } d(x, z) \leq d(x, y) + d(y, z) \quad (2.3)$$

A *metric space* is a set with a metric. In a formal language, a metric space is a pair (X, d) , where d is a metric on X . Elements of X are called *points* of the metric spaces; $d(x, y)$ is referred to as the *distance* between points x and y [12].

In this research, the dissimilarity of mtDNA sequences are evaluated by *edit distance*. The edit distance between two strings is defined as the minimum number of edit operations; or insertions, deletions, and substitutions, needed to transform one string into another [18]. Given an alphabet A , then for any string $s_i \in A^*$ and element $\alpha \in A$, three types of edit operations are defined.

1. Deletion of the n^{th} element: $D_n(s_i)$
2. Insertion of an element α after position n : $I_n(s_i, \alpha)$

3. Substitution of an element at position n by another element α : $S_n(s_i, \alpha)$

An edit sequences $Q(s_i, s_j)$ is composed of a set of consecutive edit operations that, when applied to s_i , transform s_i into s_j . This sequence is in general not unique. The length of an edit sequence is defined as

$$\|Q(s_i, s_j)\| = \text{Number of edit operations in } Q(s_i, s_j). \quad (2.4)$$

The edit distance from s_i to s_j is then defined as the minimum length over all edit sequences Θ , that transform s_i into s_j

$$ed(s_i, s_j) = \min_{\Theta} \|Q(s_i, s_j)\|. \quad (2.5)$$

For example, given two strings $s_1 = agctgctcc$ and $s_2 = agcttccc$, the edit distance from s_1 to s_2 is 2, and the corresponding minimizing edit sequence is

$$Q(s_1, s_2) = S_5(I_6(s_1, t), g). \quad (2.6)$$

Edit distances satisfy the three conditions for metrics.

$$ed(s_i, s_j) > 0 \text{ if } s_i \neq s_j, \text{ and } ed(s_i, s_i) = 0. \quad (2.7)$$

Given the minimum edit sequence $Q(s_i, s_j)$, an edit sequence $Q(s_j, s_i)$ that transforms s_j into s_i can always be found following these steps:

1. Reverse the order of $Q(s_i, s_j)$;
2. Replace each edit operation with the compliment operation:

$$D_n(s_i) \rightarrow I_{n-1}(s_j, \alpha),$$

$$I_n(s_i, \alpha) \rightarrow D_{n+1}(s_j), \text{ and}$$

$$S_n(s_i, \alpha) \rightarrow S_n(s_j, \beta).$$

For example, for $s_1 = agctgctcc$ and $s_2 = agcttccc$, given the minimum edit sequence $Q(s_1, s_2) = S_5(I_6(s_1, t), g)$, $Q(s_2, s_1) = D_7(S_5(s_2, t))$. The length of edit sequence $Q(s_j, s_i)$ obtained by this method equals to the length of edit sequence $Q(s_i, s_j)$. By the definition of edit distance,

$$ed(s_j, s_i) \leq \|Q(s_j, s_i)\| = \|Q(s_i, s_j)\| = ed(s_i, s_j). \quad (2.8)$$

In the same manner, $ed(s_i, s_j) \leq ed(s_j, s_i)$ can be obtained. So the symmetry condition is satisfied.

$$ed(s_i, s_j) = ed(s_j, s_i) \quad (2.9)$$

Given two minimum edit sequences $Q(s_i, s_j)$ and $Q(s_j, s_k)$, an edit sequence $Q(s_i, s_k)$ that transform s_i into s_k can be obtained by connecting the two edit sequences, yielding $Q(Q(s_i, s_j), s_k)$. By the definition of edit distance,

$$ed(s_i, s_k) \leq \|Q(Q(s_i, s_j), s_k)\| \quad (2.10)$$

$$= \|Q(s_i, s_j)\| + \|Q(s_j, s_k)\| \quad (2.11)$$

$$\leq ed(s_i, s_j) + ed(s_j, s_k) \quad (2.12)$$

Therefore, the triangle inequality is satisfied. From Eq. 2.7, 2.9 and 2.12, edit distance is a metric. The algorithms computing edit distance usually use dynamic programming, which can be extremely time consuming.

The set of all possible data objects is modeled as a metric space, where sample objects are points in the space. Dissimilarities are evaluated by the distance. A small distance, $d(x, y)$, between two points x and y indicates small dissimilarity between the objects represented by x and y . In other words, the objects x and y are similar to each other. In order to illustrate this more clearly, take a set of 2-dimensional data as an example. The data set contains five objects u_1, u_2, u_3, u_4, u_5 . Each object has two attributions and is mapped into a point in a 2D Euclidean space, as shown in Figure 2.1. One metric in an Euclidean space is simply the Euclidean distance. Objects u_1 and u_2 are the most similar pair, and their corresponding points are close to each other in the 2D Euclidean space. u_1 is far away from u_3 in the space, and they are relatively more dissimilar than other pairs.

Note that there is some danger in a 2D Euclidean analogy, as a metric need not satisfy all the properties of Euclidean distance, which can be derived from a norm. For example, there need be no concept of angle, basis vector, or direction in a metric space. Although the diagrams in the document are by necessity two-dimensional, it is useful to keep in mind that many intuitive concepts from 2D Euclidean space may not hold in a metric space.

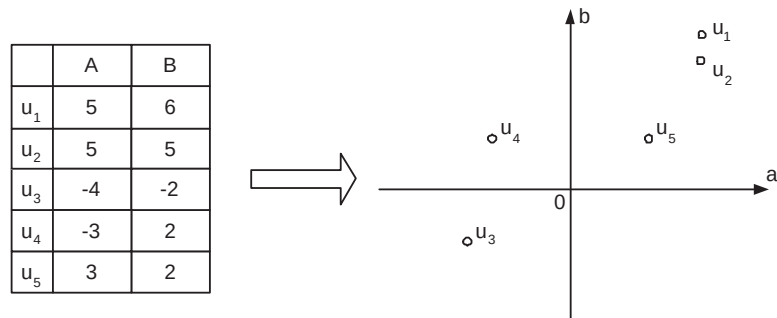


Figure 2.1: Samples mapped into in metric space

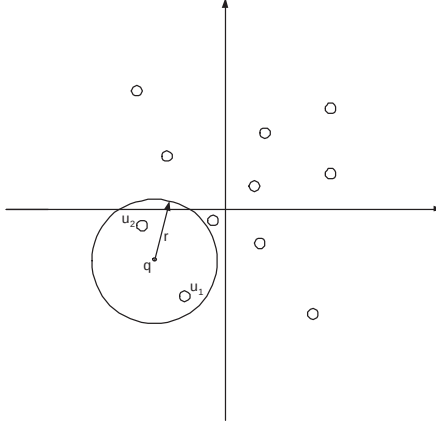


Figure 2.2: Similarity Search in Metric Space

2.2.2 Similarity Search in Metric Spaces

A requirement of approaches based on metric spaces is that the dissimilarity measurement used must be a metric. Many practical applications meet this requirement. For instance, in a forensic mitochondrial DNA (mtDNA) sequence database, a natural dissimilarity measurement is edit distance, which is a metric. If the dissimilarity measure is a metric, similarity search can be modeled as finding points within a certain distance from the target point in a metric space. A precise formulation of similarity search in a metric space follows.

Given query (q, r) on a data set in a metric space, U , where q is the search target and r is the search range; find all objects that are within distance r from the point q in the metric space, or the set $\{u_i \in U \mid d(q, u_i) \leq r\}$, which is called the *result set* of query (q, r) . Figure 2.2 illustrates similarity search in a 2D Euclidean space, in which the metric is the Euclidean distance. The circle in Figure 2.2 is centered at the search target q , and its radius equals to the search range r . Thus, the points within the circle are the results of search query (q, r) . In this example, the result set is $\{u_1, u_2\}$.

2.2.3 Tree-Structured Indexing in Metric Space

Search methods based on metric spaces can use tree-structured indexing techniques to achieve a sublinear time complexity. At each tree node, indexing divides the data set into several subsets based on similarity relations between objects. Indexing based on a metric space is equivalent to hierarchically partitioning the space into several subsets. An example that illustrates a 2-level tree in a 2D Euclidean space follows. As demonstrated in Figure 2.3(a) and Figure 2.3(b), the space is partitioned into three subsets P_1 , P_2 and P_3 at the first level of the tree. Each subset is assigned to a child node. At the second level, each of the three subsets is partitioned further into 3 subsets. For example, P_1 is partitioned into P_{11} , P_{12} and P_{13} . Figure 2.3(d) illustrates that P_{11} , P_{12} and P_{13} are assigned to three child nodes $node_4$, $node_5$ and $node_6$, respectively.

During a search processing, the query (q, r) propagates through the nodes that contain objects within the search criterion until the leaves are reached. In other words, the subsets that intersect with the circle that represents the search criterion in the metric space are kept for further pruning. The final candidate set is the union of all the subsets that survive, corresponding to the leaves of the tree. Then, all the objects in the candidate set will be searched exhaustively, and those within the circle are submitted as search results. Figure 2.4 illustrates how a query (q, r) is processed in the 2-level tree index structure presented above. In Figure 2.4(a), q is the search target, r is the search range and the circle represents the search criterion. At the first level of the tree, P_2 and P_3 intersect the circle. The query propagates through $node_2$ and $node_3$. At $node_2$, the search criterion intersects with one subset, P_{23} . At $node_3$, the search criterion intersects with two subsets, P_{31} and P_{32} . Therefore, the union of these three subsets is chosen as

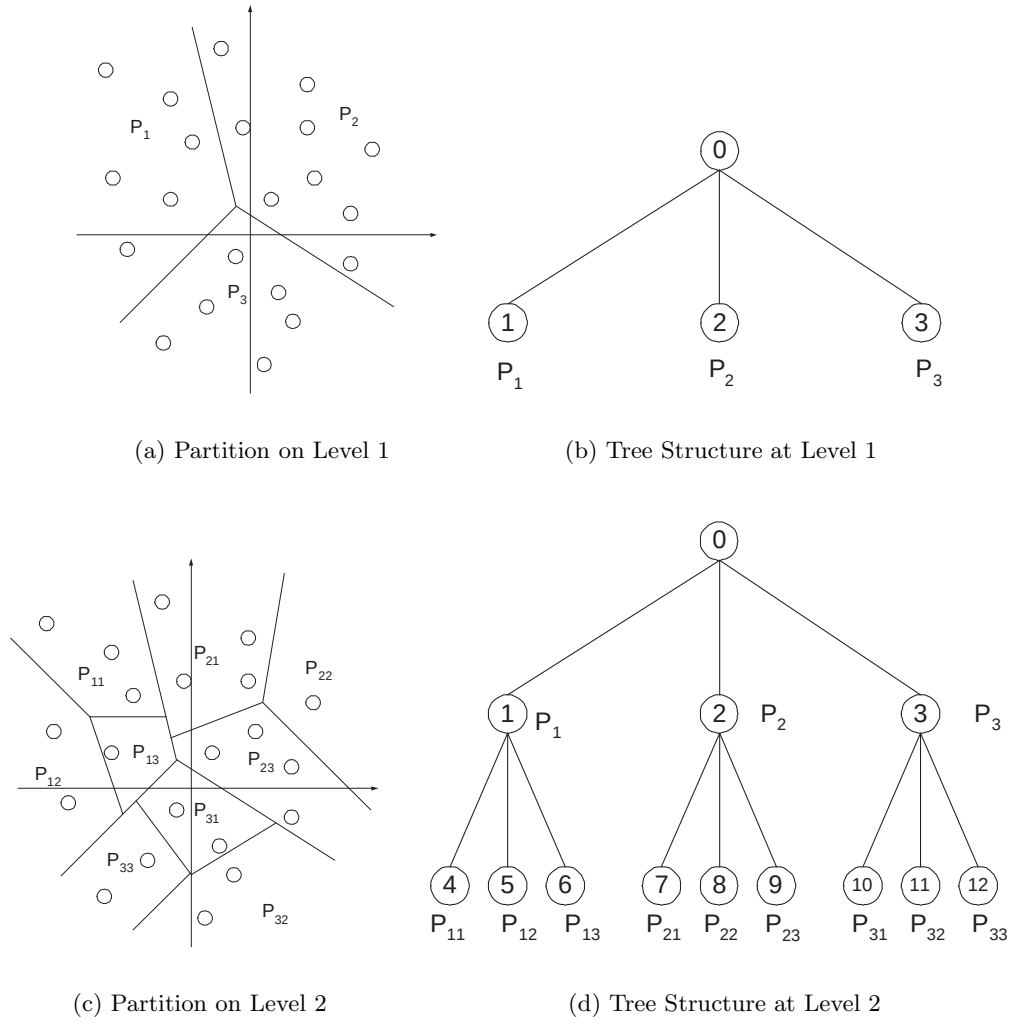


Figure 2.3: A 2-level Tree Indexing Structure in a 2D Metric Space

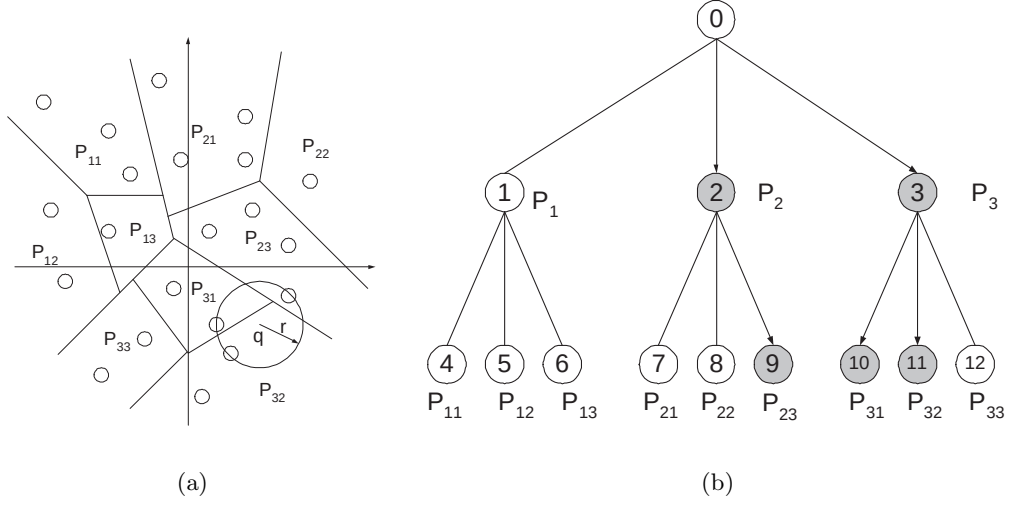


Figure 2.4: Processing of Search Query (q, r)

the candidate set, which will be searched exhaustively. The search time is

$$T = N_{cand} \times T_{metric} + T_{index}, \quad (2.13)$$

where N_{cand} is the number of objects in the candidate set, T_{metric} is the time required to perform one metric computation and T_{index} is the time required to use the index to identify the candidate set. The metric calculation is often so time consuming that T_{index} in Eq (2.13) can be neglected. Therefore, N_{cand} can be used as a performance measurement. The size of the candidate set depends heavily on the data indexing structure. In other words, the method used to partition the sample space affects the size of the candidate set. Figure 2.5 compares the candidate sets of two single level trees with different partitioning methods. In both trees, the space is divided into 3 subsets P_1 , P_2 and P_3 . In Figure 2.5(a), since P_1 and P_2 intersect the search criterion, these two partitions are chosen as the candidate set. In Figure 2.5(b), the candidate set is P_1 ,

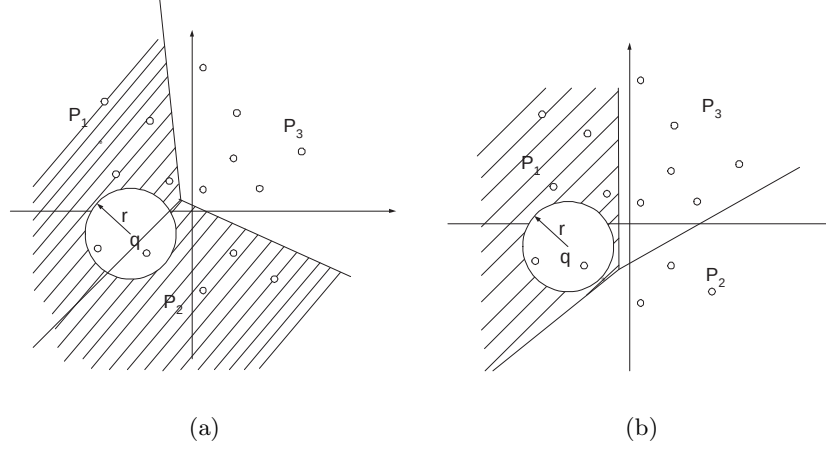


Figure 2.5: The Effect of Partition Strategy on Candidate Set Size

which contains fewer points than the previous candidate set. Thus, the search time will be shorter. This example shows that different partitions lead to different candidate set sizes. The goal is to find an optimal partition that minimizes the expected candidate set size N_{cand} over all queries while keeping the time required to resolve the candidate set small. This goal is in general not achievable, and one must content with suboptimal results.

Many approaches have been designed and implemented. Most can be divided into two categories: methods using reference points, and methods based on Voronoi partitions.

Indexing Based upon Voronoi Partitions

The approaches based on Voronoi partitions [13] divide the space into several Voronoi cells. Given a set of points U in a metric space and a subset R of U , the Voronoi cell of

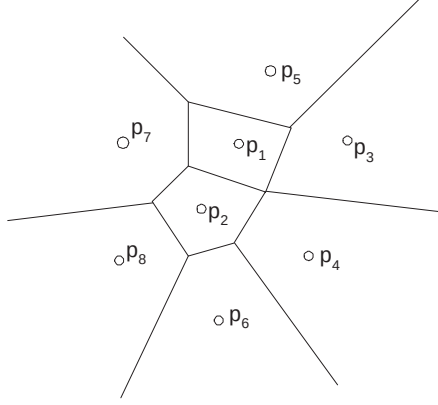


Figure 2.6: Voronoi Partition in a 2D Euclidean Space

a point $p_i \in R$ is defined as,

$$V(p_i) = \{u \in U | d(p_i, u) \leq d(p, u), \forall p \in R\}. \quad (2.14)$$

In words, $V(p_i)$ contains all the points in U that are at least as close to p_i as to any other point in R . Ties are broken arbitrarily, and can on occasion cause ambiguity in membership in search result sets. Figure 2.6 shows a Voronoi partition in a 2D Euclidean space.

In these approaches, on each node of the tree, several split points are chosen, making up the set R . The space is divided into Voronoi partitions based upon these split points. Voronoi partitions are not the focus of this research. For a comprehensive review and in-depth discussion, see [2].

Indexing Based upon Reference Points

Partitioning approaches using reference points choose several reference points in the space and assign one or more of them to each node of an indexing tree. The set of

objects associated with a node is divided into several subsets according to the distances between the objects and the reference points. Child nodes repeat this process with other reference points until leaves are reached. In this manner, the whole space is hierarchically partitioned into portions of annular regions.

Many methods have been designed to select reference points. However, an effective reference points selection method is still not available. In this research, reference points are chosen randomly. The original implementation in vp-tree [36] assigns a different reference point to each node. Yates proposed fq-tree [25], in which all the nodes at the same level use the same reference point. This strategy saves a lot of metric computations, when multiple nodes at the same level are visited. Yates thought the disadvantage was this strategy would lead to a taller tree and longer tree traversal time. However, for indexing trees based on metric spaces, the tree traversal time is no longer a critical part of the search time. Moreover, in this report, our experiments show that a taller tree will have better performance as presented in Chapter 5. Therefore, the method of fq-tree is chosen as the indexing structure in this research. The detailed mathematical description of the indexing tree structure is presented below.

Given the tree height h , h reference points $\{p_1, p_2 \dots p_h\}$ are chosen. A reference point p_i is assigned to the nodes at level i of the tree. At level i , the space is partitioned into several annular regions R_{ij} centered at the reference point p_i with respect to the distance. Given the set of data points U embedded in the metric space, the annular regions associated with reference point p_i are

$$R_{ij} = \{u_k \in U | d(u_k, p_i) \in [a_{ij}, a_{ij+1})\}, \quad (2.15)$$

where $\{a_{ij}\}_{j=0}^{n_i}$ is an increasing sequence with $a_{i0} = 0$. The indexing tree can be imagined as h levels of annular regions that overlap. The final partition consists of the intersections of all the annular regions. For example, one element of the final subsets found by descending along a path from root to leaf is,

$$P_i = R_{11} \cap R_{21} \cap R_{31} \dots R_{h1}. \quad (2.16)$$

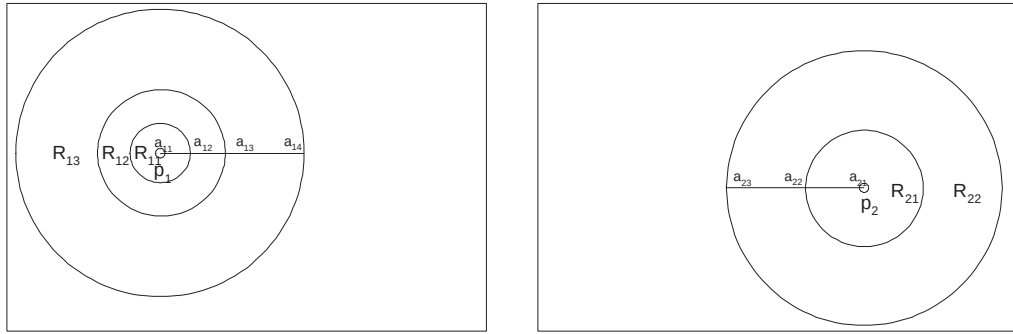
Figure 2.7 illustrates an example of the partition of a 2-level indexing tree. In Figure 2.7(a), the space at tree level 1 is partitioned into three annular regions R_{11} , R_{12} and R_{13} (with a fourth region implicitly extending from the largest boundary shown in the figure to infinity, which is generally empty). At tree level 2, the space is partitioned into two annular regions R_{21} and R_{22} . In Figure 2.7(c), the final partition of the 2-level indexing tree is produced by the intersections of these five annular regions. There are eight subsets in the final partition. Three of them are noted for example:

$$P_1 = R_{13} \cap R_{22} = \{u \in U | d(u, p_1) \in [a_{13}, a_{14}) \text{ and } d(u, p_2) \in [a_{22}, a_{23})\}; \quad (2.17)$$

$$P_2 = R_{13} \cap R_{21} = \{u \in U | d(u, p_1) \in [a_{13}, a_{14}) \text{ and } d(u, p_2) \in [a_{21}, a_{22})\}; \quad (2.18)$$

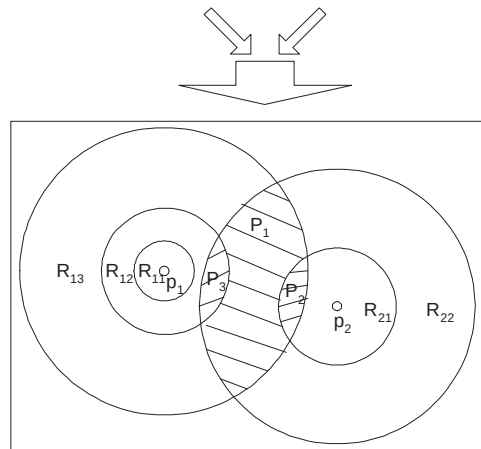
$$P_3 = R_{12} \cap R_{22} = \{u \in U | d(u, p_1) \in [a_{12}, a_{13}) \text{ and } d(u, p_2) \in [a_{22}, a_{23})\}. \quad (2.19)$$

Selection of the partition determines how the range of the metric evaluated over the set of objects and a reference point is divided into a set of $\{[a_{ij}, a_{ij+1})\}$. Equal width and equal percentage are two widely used partitioning strategies. Equal width means the metric is partitioned into intervals of equal size. The equal percentage method partitions the metric's range into intervals such that each interval contains the same number of objects. Figure 2.8 illustrates both partitioning strategies in a 2D Euclidean



(a) Partition at Level 1

(b) Partition at Level 2



(c) Final Partition

Figure 2.7: Partition of a 2-level Indexing Tree

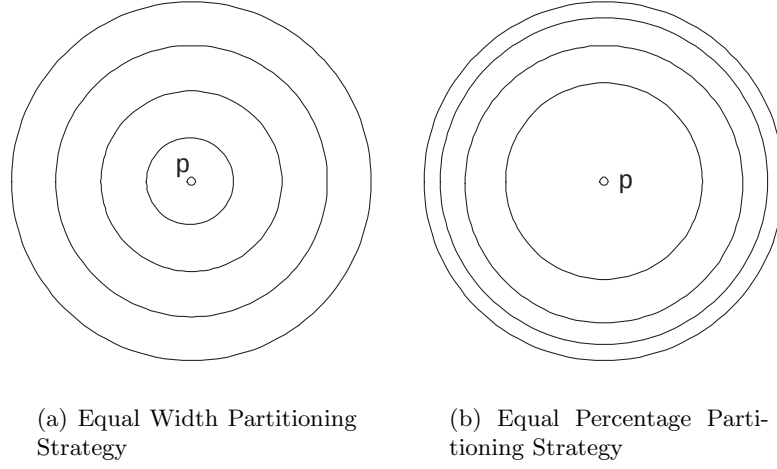


Figure 2.8: Two Partitioning Strategies: Equal Width and Equal Percentage

space. The main difference between these two strategy is that the equal percentage partitioning strategy guarantees that the index tree is balanced, while the strategy based on equal width does not. For traditional index trees, balancing is a critical issue for search performance. However, unbalancing is a key performance issue for indexing high dimensional metric spaces [10]. An unbalanced tree is usually taller than a balanced one. In Chapter 5, a conclusion is drawn that a taller tree yields better performance. Therefore, equal width partitioning is used in this research.

Search Method Using Indices Based upon Reference Points

The essential idea of search methods using indices based upon reference points is to use the triangle inequality in Eq (2.3) to rule out partitions, and therefore paths of descent in the index tree, that can not contain a solution. The search request propagates through the tree structured index, and a candidate set is generated. A result set, which is a subset of the candidate set, is obtained by exhaustively searching the candidate set. The candidate set of query (q, r) is found using the triangle inequality. In Figure 2.9,

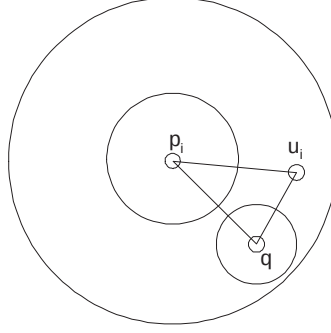


Figure 2.9: Triangle Inequality in Similarity Search

three points, a reference point p_j , the query target q and an object u_i are located in the metric space. By the triangle inequality, the value of the metrics, or distances, as represented in the figure by lines, are related by the inequalities.

$$d(q, u_i) \leq d(u_i, p_j) + d(q, p_j), \quad (2.20)$$

and

$$d(q, p_j) \leq d(u_i, p_j) + d(q, u_i) \implies d(q, p_j) - d(u_i, p_j) \leq d(q, u_i), \quad (2.21)$$

or

$$d(q, p_j) - d(u_i, p_j) \leq d(q, u_i) \leq d(q, p_j) + d(u_i, p_j). \quad (2.22)$$

If u_i belongs to the result set, it should satisfy the search criterion

$$d(q, u_i) \leq r, \quad (2.23)$$

or

$$d(q, p_j) - r \leq d(u_i, p_j) \leq d(q, p_j) + r. \quad (2.24)$$

Therefore, a necessary condition SC that must hold in order for the search criterion to be satisfied is,

$$SC = \bigcap_{j=1}^k \{u_i \in U | d(u_i, p_j) \in [d(q, p_j) - r, d(q, p_j) + r]\}. \quad (2.25)$$

The candidate set $Cand$ is the union of all the stored objects lying within partitions that intersect the search criterion SC ,

$$Cand = \bigcup_{i=1}^t \{P_i | P_i \cap SC \neq \phi\}, \quad (2.26)$$

where t is the total number of partitions. Once the search request has been restricted to the candidate set, the candidate set is scanned exhaustively to get the result set,

$$Res = \{u_i \in U | u_i \in Cand \text{ and } d(u_i, q) \leq r\}. \quad (2.27)$$

Figure 2.10 illustrates an example of processing a search query (q, r) on a two level index tree based upon reference points. In Figure 2.10(a), all three subsets intersect with the search criterion, and in Figure 2.10(b), two subsets intersect with the search criterion. The shaded area in Figure 2.10(c), which is the intersection of two dark areas in Figure 2.10(a) and 2.10(b), represents the candidate set. One component of the search time is assumed proportional to the size of the candidate set, due to linear search. A second component is due to traversal of the tree, and a third component is due to computation of the metric distance from the query to each reference point. This is summarized by

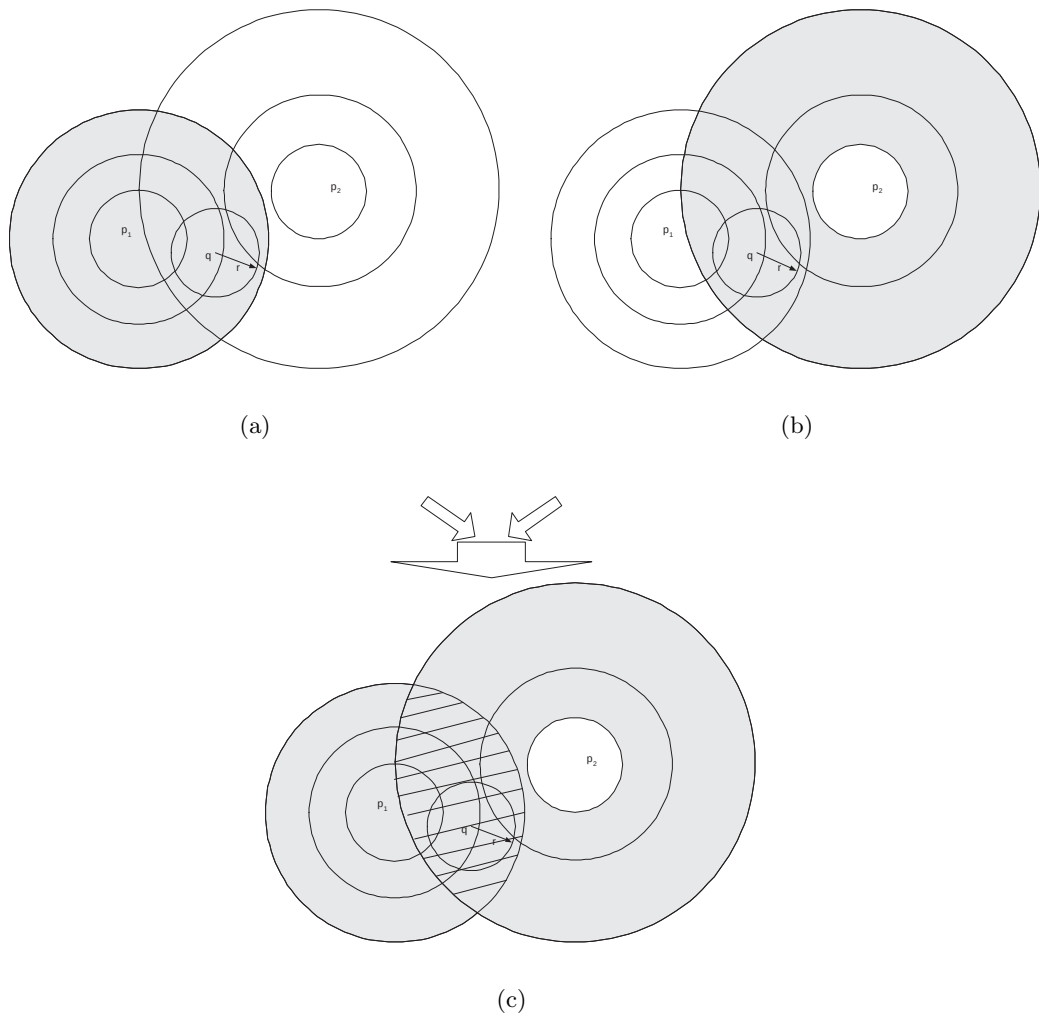


Figure 2.10: Search Method Based upon Reference Points

the equation

$$T = N_{ref} \times T_{metric} + N_{cand} \times T_{metric} + T_{tree} \quad (2.28)$$

$$= (N_{ref} + N_{cand}) \times T_{metric} + T_{tree}, \quad (2.29)$$

where N_{ref} is the number of reference points, N_{cand} is the number of objects in the candidate set, and T_{tree} is the tree traversal time. Let $N_{metric} = N_{ref} + N_{cand}$, which is the total number of metric computations. Since metric computations are usually so time consuming, T_{tree} can be neglected. Therefore, N_{metric} is used as one of the criterion to evaluate search performance in this report. Actually, in most situations, $N_{cand} \gg N_{ref}$, so the size of candidate set N_{cand} determines the value of N_{metric} . The goal is to minimize the value of N_{cand} . In other words, the goal is to make the candidate set as small as possible, subject to the constraint that it must contain all solutions to the query.

2.3 Performance Evaluation and Experimental Setup

The number of metric computations dominates the search query processing time. Therefore, N_{metric} is used as one criterion to evaluate the search performance of index tree in this report. In Chapter 3 and Chapter 4, the problems discussed are associated with different result sets. Therefore alternative performance evaluation criteria are used. See those chapters for details. Moreover, in order to compare the performance across different data sets, *normalized search ranges* are used. A normalized search range is the ratio between the search range and the average distance between all the stored objects,

or $\frac{r}{\mu}$, where the average distance μ is,

$$\mu = \frac{\sum_{i=1}^{N_{total}} \sum_{j=i+1}^{N_{total}} d(u_i, u_j)}{N_{total} \times (N_{total} - 1)/2}. \quad (2.30)$$

Four different data sets are used to evaluate performance of search methods. These are a database of mitochondrial DNA sequences, a database of color histograms of images, a set of 100-dimensional synthetic vectors, and a database of rowing photos:

- 5,736 different mitochondrial DNA sequences are extracted from the FBI's mtDNA Population Database [21]. Each sequence is composed of two regions (HV1 and HV2) of the whole mtDNA sequence and contains about 600 bases. Edit distance is used as a measure of dissimilarity;
- 10,000 32-dimensional vectors describing the color histograms of images extracted from the Corel image collection are used from the UCI KDD Database Repository located at <http://kdd.ics.uci.edu/databases/CorelFeatures/CorelFeatures.html>. Euclidean distance is used as a measure of dissimilarity;
- 10,000 100-dimensional synthetic vectors are created, using independently normally distributed random values with variance equaling to 0.2 and mean equaling to 0. Euclidean distance is also used;
- 1,092 512D feature vectors are extracted from the photos of a rowing team engaged in different activities and groupings taken over a two year period. The feature vectors are composed of the gray scale histogram and hue histogram derived from the original photo.

The indexing structure used in the experiments is a binary tree with the same reference point assigned to all the nodes at the same level.

2.4 Research Problems and Overview of Results

In this section, a brief overview of the key issues that are discussed in this report is presented, along with a summary of the achievements of this research effort.

2.4.1 Performance Limitations

The performance of similarity searches in metric spaces is not good enough for general use according to the experiments performed on the synthetic vectors and other data. Investigations and experiments are reported that were used to identify the causes of performance limitations of similarity search methods based upon metric spaces. The performance limitations are imposed by the statistical characteristics of a data set. The maximum feasible search range is limited by the variance of the distances between all the stored objects in the database. A method to estimate the maximum feasible search range is provided. Experimental results show that this estimation method works very well.

2.4.2 Evaluation of Performance and Utility of Data Extraction Methods and Metrics

Data extraction and metric selection are two very critical steps of database similarity search. They are conducted before constructing of the tree-structured index. A good data extraction method and proper metric can yield good search performance. A performance evaluation criterion based upon search efficiency is designed. Two examples

comparing different Minkowski metrics and PCA (principal component analysis) are presented to demonstrate the use of this criterion. A further discussion on search utility is provided, which illustrates some directions for future work.

2.4.3 Issues Related to Tree Indexing Structures

The opportunities of several issues related to tree indexing structures are evaluated. The effects of the number of branches descending to child nodes from each index node and the maximum number of objects allowed to be associated with each leaf are discussed in this report. The conclusions are that a binary tree index works best, and that leaf size should be minimized. Moreover, from the discussion, an improved tree index structure with fixed height is recommended.

2.5 Description of Remaining Chapters

This report is organized as follows: Chapter 3 explores the performance limitations of similarity search based upon metric spaces. In Chapter 4, performance of different metrics and data extraction methods is evaluated and compared based upon search efficiency and utility. Chapter 5 discusses some issues related to tree-structured indices based on metric spaces. A summary of this work and discussion of open research issues is provided in Chapter 6.

Chapter 3

Performance Limitations

3.1 Overview of Performance Limitations

The number of metric computations, N_{metric} , was introduced in Section 2.4 as a measure that can be used to evaluate search performance. The goal is to minimize the number of metric computations performed during search request processing. During research on different search methods, experiments show that when the search range is beyond a certain value, the value of N_{metric} will become so large that search performance will become extremely low. Since N_{cand} determines N_{metric} , the value of N_{cand} is critical. Results indicate that the value of N_{cand} becomes very close to N_{total} for large range specifications, which means almost all the objects are exhaustively searched. In this situation, no matter what the value of N_{res} is, the search method's performance is unacceptable. Even if most of the candidates are final search results, the candidate set is so big that it is not worth building a tree index structure before searching. The critical value of the query range required to obtain minimum acceptable search performance is named the *range threshold* in this report. Obviously, what users care about is whether

or not the range threshold is larger than search ranges of interest. Data sets whose range thresholds are smaller than the search ranges of interested are not likely to yield satisfactory performance using the tree-structured indexing techniques based on metric spaces. In this chapter, the causes of performance limitations are explored, and a method to estimate the range threshold is provided.

3.2 Performance Evaluation and Pruning Rate

Obviously, the performance limitations and the range threshold are affected by the characteristics of data. Comparisons between different data sets helps to explore the causes of performance limitations. Therefore, a method to compare the search performance of different data is required. The normal criterion N_{metric} is no longer effective, because different data have different values of N_{total} , which affect the value of N_{cand} , the critical part of N_{metric} . Thus, the ratio between N_{cand} and N_{total} is used as a performance evaluation criterion in this chapter. Moreover, different data sets yield different values of N_{res} , which also affect the value of N_{cand} . $\frac{N_{res}}{N_{total}}$ is used to help illustrate search performance. The normalized search range $r_{normalized}$ is used instead of the original search range.

In the sections below, a new concept, *pruning rate*, is used to assist in understanding the mathematical reasoning. The definition of pruning rate is

$$PR = \frac{\text{Number of Objects Being Pruned Away}}{\text{Number of Total Objects}}. \quad (3.1)$$

Actually, the objects being pruned away are the objects that are not included in the

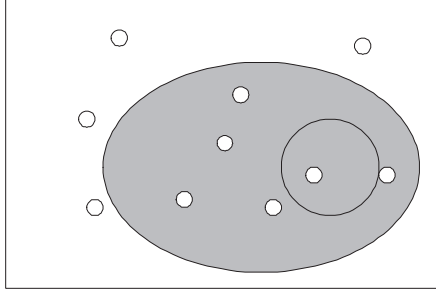


Figure 3.1: Pruning Rate in a 2D Metric Space

candidate set. Thus, the pruning rate can also be calculated by

$$PR = 1 - \frac{N_{cand}}{N_{total}}. \quad (3.2)$$

Figure 3.1 illustrates the concept of pruning rate in a 2D metric space. The shadow area in Figure 3.1 is the candidate set, and the circle indicates the search request. There are 6 objects in the candidate set, and 10 objects in the space. The pruning rate is

$$PR = 1 - 6/10 = 40\% \quad (3.3)$$

For a tree-structured index, the total pruning rate is

$$PR_{total} = \frac{N_{total} - N_{cand}}{N_{total}}, \quad (3.4)$$

or

$$N_{cand} = N_{total}(1 - PR_{total}). \quad (3.5)$$

Since the pruning rate on each node is more helpful for the performance analysis in this

chapter, average pruning rate of a tree is defined as below

$$PR_{avg} = PR_{total}^{1/N_{ref}} = \left(\frac{N_{total} - N_{cand}}{N_{total}}\right)^{1/N_{ref}}. \quad (3.6)$$

The ratio between size of a candidate set and total number of objects can be calculated by

$$\frac{N_{cand}}{N_{total}} = (1 - PR_{avg})^{N_{ref}}. \quad (3.7)$$

A larger PR_{avg} will lead to a smaller value of $\frac{N_{cand}}{N_{total}}$. Therefore, PR_{avg} can also be used to evaluate search performance.

3.3 Limitations on Pruning Rate and Search Range

Pruning rate is a function of search range, and decreases with increasing range. A large candidate set relative to the size of the data set means the average pruning rate is low. The range threshold is the value of a search range required to obtain a minimum acceptable pruning rate. In order to find the underlying relations between the average pruning rate and the search range, the essential concepts of the tree-structured indexing techniques based on metric spaces need to be addressed. The core of this indexing technique is derived from the triangle inequality, and bounds the distance metric between a candidate object u and a reference point p by functions of the reference point, query target, and query range r :

$$d(q, p) - r \leq d(u, p) \leq d(q, p) + r, \quad (3.8)$$

Only the points u that satisfy this inequality will be included in the candidate set, and other points will be pruned away. In Figure 3.2, the objects in shadow area are left out.

The pruning rate is

$$PR = \frac{\text{Number of Objects within the Shadow Area}}{\text{Number of Total Objects}}. \quad (3.9)$$

A histogram of the distances between the reference point p and all objects will help to demonstrate the situation. Figure 3.3 illustrates the search criterion for query (q, r) applied on a distance histogram. The worst case, when $d(p, q)$ is near the median value of distance, is illustrated. The shaded parts that are outside the range $[d(p, q) - r, d(p, q) + r]$ are pruned away. The area of the blank part under the curve is the size of the candidate set. Therefore, a larger shaded area implies a higher pruning rate and better performance. Figure 3.4 shows the same search criterion performed on two data sets with different distance distributions. The distance distribution in Figure 3.4(b) has a larger variance, and the shaded area is much bigger than that in Figure 3.4(a). The pruning rate depends heavily on the statistical characteristics of a data set, and a large distance variance will lead to a large pruning rate and better search performance.

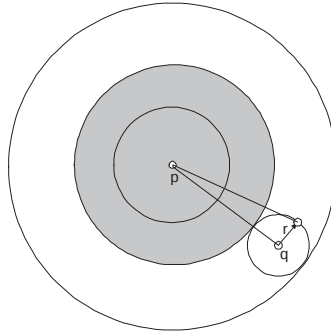


Figure 3.2: Pruning Operation on Each Node

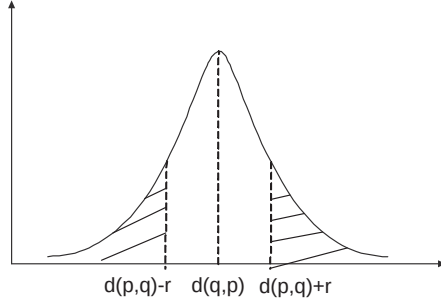


Figure 3.3: Search criterion Applied on Distance Histogram

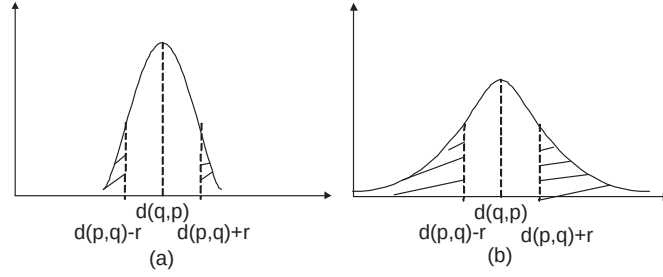


Figure 3.4: Comparison of Pruning Rates

On the other hand, if the minimum acceptable pruning rate is fixed, the data set with the larger distance variance may have larger range threshold. As mentioned, for any search target q_i , and using the same definitions of symbols as before, the pruning rate is

$$PR = P(|d(u, p) - d(q_i, p)| \geq r) \quad (3.10)$$

Let $d(q_i, p) = \mu + \alpha_i$, where μ is the mean of distances between p and all the points.

$$= P(|d(u, p) - \mu - \alpha_i| \geq r) \quad (3.11)$$

$$= P((d(u, p) - \mu - \alpha_i)^2 \geq r^2) \quad (3.12)$$

By the Markov Inequality,

$$\leq \frac{E((d(u, p) - \mu - \alpha_i)^2)}{r^2} \quad (3.13)$$

$$= \frac{E((d(u, p) - \mu)^2 - 2\alpha_i \times (d(u, p) - \mu) + \alpha_i^2)}{r^2} \quad (3.14)$$

$$= \frac{E((d(u, p) - \mu)^2) - 2\alpha_i \times E(d(u, p) - \mu) + \alpha_i^2}{r^2} \quad (3.15)$$

$$= \frac{\sigma^2 - 0 + \alpha_i^2}{r^2}, \quad (3.16)$$

where σ^2 is the variance of distances between p and all the points. The average pruning rate for all search target is

$$\overline{PR} \leq \frac{\sigma^2 + E(\alpha_i^2)}{r^2} \quad (3.17)$$

$$= \frac{\sigma^2 + E((d(q_i, p) - \mu)^2)}{r^2} \quad (3.18)$$

$$= \frac{\sigma^2 + \sigma^2}{r^2} \quad (3.19)$$

$$= \frac{2\sigma^2}{r^2} \quad (3.20)$$

Sample enough points from a data set, and pick 5% of sampled points as reference points. Calculate the variance of distances between each reference points and all the sampled points, and obtained the average value of variances. If this sampled variance is used in Eq 3.20, $\overline{PR}$ can be used to approximate the average pruning rate of a tree, PR_{avg} .

$$PR_{avg} \cong \overline{PR} \leq \frac{2\sigma^2}{r^2}, \quad (3.21)$$

Therefore, the average pruning rate PR_{avg} is determined by the ratio of the variance and square of the query range. If the query range r is fixed, when the variance σ^2 decreases, PR_{avg} will also decrease. From the Eq 3.21, the upper bound of the search

range is

$$r \leq \sqrt{\frac{2\sigma^2}{PR_{avg}}}. \quad (3.22)$$

From Eq 3.22, given the minimum acceptable pruning rate, the threshold range can be estimated. By Eq 3.21, the candidate set size for the range threshold is

$$N_{cand} = N_{total} \times (1 - PR_{avg})^{N_{ref}} \geq N_{total} \left(1 - \frac{2\sigma^2}{r^2}\right)^{N_{ref}}. \quad (3.23)$$

The search time is

$$T = (N_{cand} + N_{ref}) \times T_{metric} + T_{tree} \quad (3.24)$$

$$\geq (N_{total} \times (1 - \frac{2\sigma^2}{r^2})^{N_{ref}} + N_{ref}) \times T_{metric} + T_{tree}. \quad (3.25)$$

This provides a lower bound of the search time of a index structure that uses N_{ref} reference points and is built for a database that contains N_{total} objects. Given the database performance requirement of k queries/sec,

$$T \leq \frac{1}{k}. \quad (3.26)$$

By Eq 3.25,

$$\Rightarrow (N_{total} \times (1 - \frac{2\sigma^2}{r^2})^{N_{ref}} + N_{ref}) \times T_{metric} + T_{tree} \leq \frac{1}{k}. \quad (3.27)$$

Comparing to T_{metric} , T_{tree} is usually so small that it can be neglected.

$$\Rightarrow (N_{total} \times (1 - \frac{2\sigma^2}{r^2})^{N_{ref}} + N_{ref}) \times T_{metric} \leq \frac{1}{k}. \quad (3.28)$$

$$\Rightarrow N_{total} \times (1 - \frac{2\sigma^2}{r^2})^{N_{ref}} + N_{ref} \leq \frac{1}{k \times T_{metric}} \quad (3.29)$$

$$\Rightarrow 1 - \frac{2\sigma^2}{r^2} \leq (\frac{\frac{1}{k \times T_{metric}} - N_{ref}}{N_{total}})^{1/N_{ref}} \quad (3.30)$$

$$\Rightarrow r^2 \leq \frac{2\sigma^2}{1 - (\frac{\frac{1}{k \times T_{metric}} - N_{ref}}{N_{total}})^{1/N_{ref}}} \quad (3.31)$$

$$\Rightarrow r \leq \sqrt{\frac{2\sigma^2}{1 - (\frac{\frac{1}{k \times T_{metric}} - N_{ref}}{N_{total}})^{1/N_{ref}}}} \quad (3.32)$$

Let this upper bound be the range threshold r_{max} .

$$r_{max} = \sqrt{\frac{2\sigma^2}{1 - (\frac{\frac{1}{k \times T_{metric}} - N_{ref}}{N_{total}})^{1/N_{ref}}}}, \quad (3.33)$$

Using Eq 3.33, the range threshold r_{max} satisfying the performance requirement of k queries/sec can be estimated. This is a good method that can be used to test the feasibility of the tree-structured indexing technique based on metric spaces before applying it on a data set. The main steps are:

1. Calculate the distance variance of sampled points;
2. Calculate the maximum search range r_{max} by Eq 3.33;
3. If r_{max} is greater than the search range of interest, the algorithm is feasible; otherwise it is not.

3.4 Experimental Results and Discussions

A number of experiments are conducted to demonstrate the conclusions discussed above. Three datasets: image color histogram data, mtDNA sequence data, and synthetic vector data are used. Three graphs are used to summarize the findings for each dataset. The histogram of distance distributions is plotted to show the statistical characteristics of each data set. The average pruning rate, PR_{avg} , is given next, plotted as a function of normalized search range. The largest possible average pruning rate for any given search range is desired, consistent with retention in the candidate set of all search results. This is because $(1 - PR_{avg})^{N_{ref}}$ is the fraction of the database that must be searched exhaustively in linear time. The third graph presents two curves on a common axis. The higher curve plots the fraction of objects from the database that remain in the candidate set after pruning, as a function of normalized search range. This curve increases with increasing search range because the method is able to exclude fewer objects. The lower curve is the fraction of objects in the candidate set that satisfy the search criteria and are results. In general, indexing strategies that minimize the gap between these two curves are most desirable. If a probability density function of normalized search ranges likely to be encountered in the applications is known, the desirability of an indexing strategy can be quantified as the inverse, or negative, of the integral of the difference between the upper and lower curves, weighted by the probability density of range. Histograms computed from histories of queries can also be used. For the experiment of each data set, ten trees are built using the same settings. For each tree, 200 searches using different search targets are conducted for each search range. The average candidate set size and result set size of the 200 searches are calculated as the results for each tree. The variation of experimental results of ten trees at each search range are illustrated by a

bar in the figures. Figure 3.5, 3.8, and 3.11 are the histograms of distance distribution. Different average pruning rates under varying search ranges are illustrated in Figure 3.6, 3.9, and 3.12. Also, the values of two ratios $\frac{N_{cand}}{N_{total}}$ and $\frac{N_{res}}{N_{total}}$ at different search ranges are illustrated in Figure 3.7, 3.10, and 3.13.

Table 3.1 lists the sampled standard deviations of normalized distances of three data sets. MtDNA sequence data have the maximum distance standard deviation, and the synthetic vector data have the minimum distance standard deviation. Comparing the search performance of these two data sets, the performance using mtDNA sequence data is much better than that using synthetic vector data for the same normalized search range. The experiment using the synthetic vector data shows a case, in which the similarity search method is not feasible because of the extremely small distance variance. These results agree with conclusions derived by the analysis in Section 3.3, that a data set with a larger distance variance can achieve a better search performance for the same normalized search range.

In order to avoid those complicate parameters, the minimum average pruning rate is given to check the range threshold estimation algorithm. The minimum average pruning rate is set to be 5%. By Eq. (3.22), the estimated range thresholds are calculated. Both the estimated range thresholds and the thresholds derived from experimental results are listed in Table 3.2. The estimated values are greater than the experiment results, so the estimation method is optimistic in these cases, but it is reasonably consistent.

Table 3.1: Sampled Distance Standard Deviation of Data Sets

	Image Color Histograms	MtDNA Sequences	Synthetic Vectors
Sampled Standard Deviation	0.2594	0.3058	0.0264

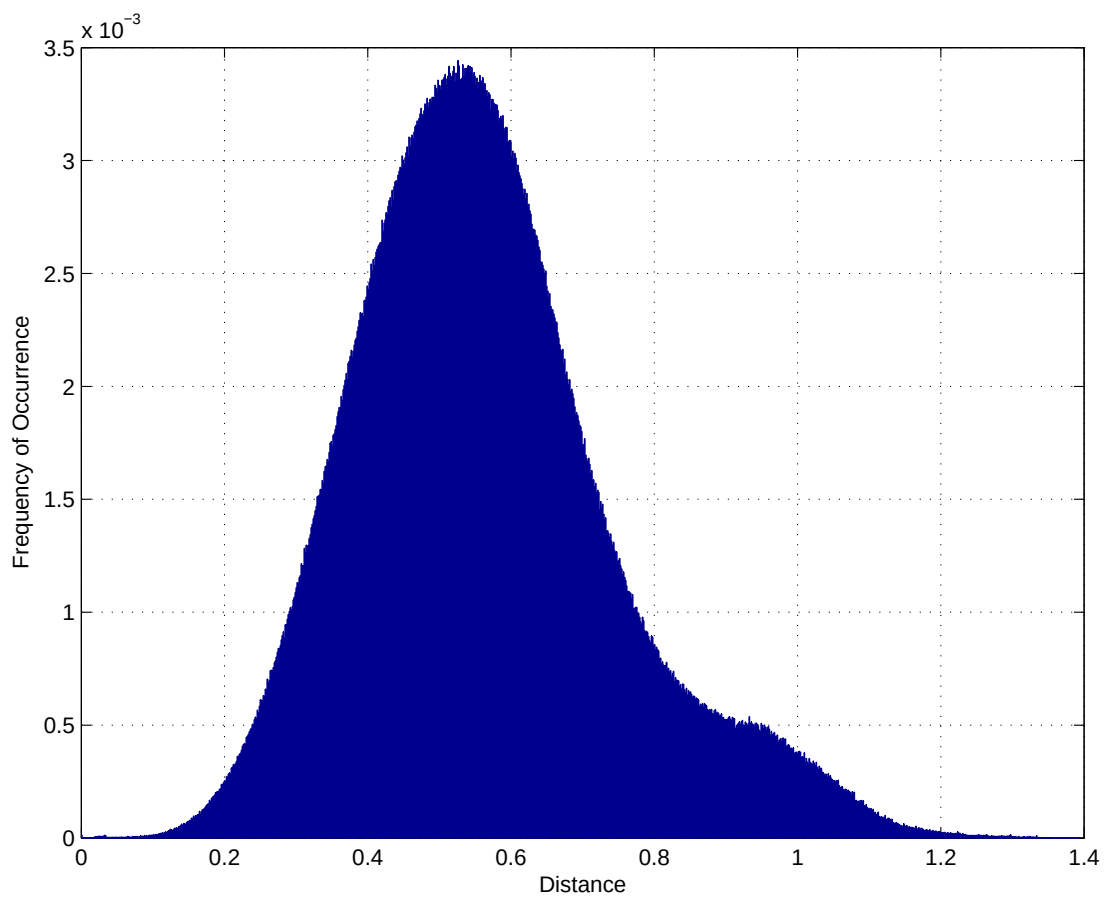


Figure 3.5: Histogram of Distance Distribution (Color Histogram of Images)

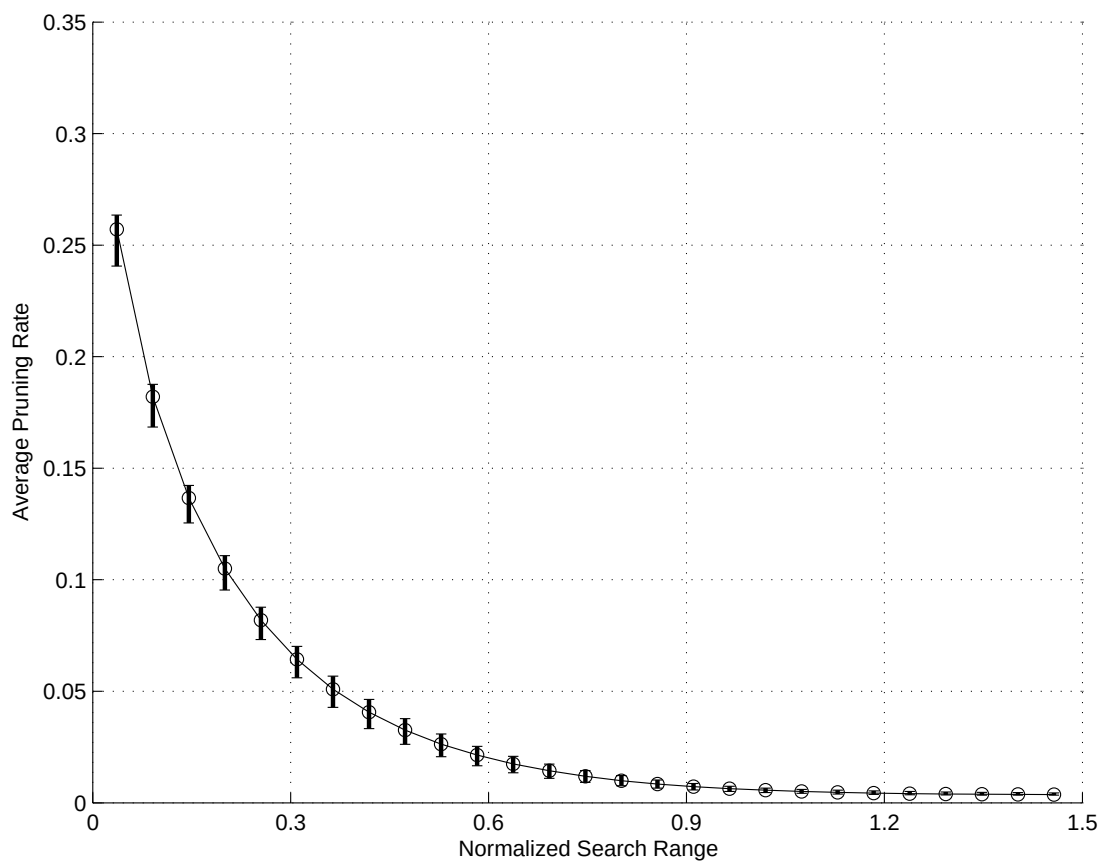


Figure 3.6: Average Pruning Rate vs. Normalized Search Range (Color Histogram of Images)

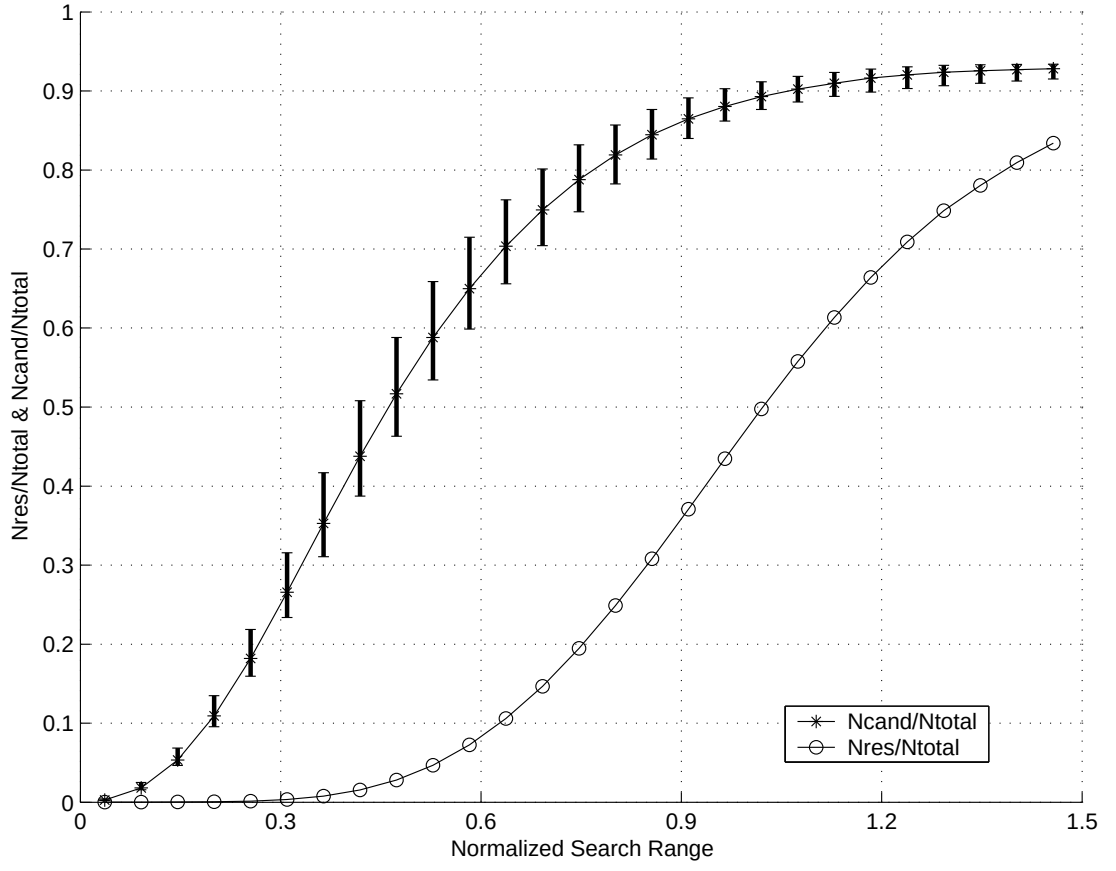


Figure 3.7: $\frac{N_{res}}{N_{total}}$ & $\frac{N_{cand}}{N_{total}}$ vs. Normalized Search Range (Color Histogram of Images)

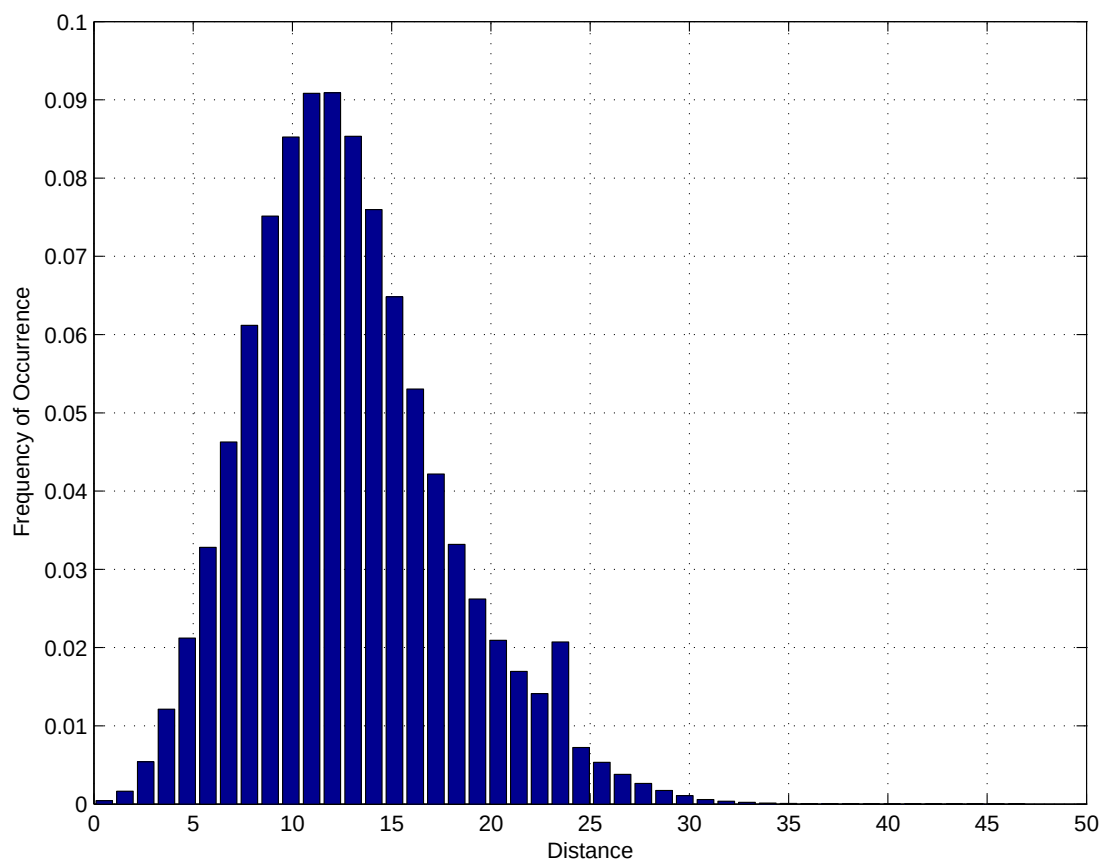


Figure 3.8: Histogram of Distance Distribution (Mitochondrial DNA Sequences)

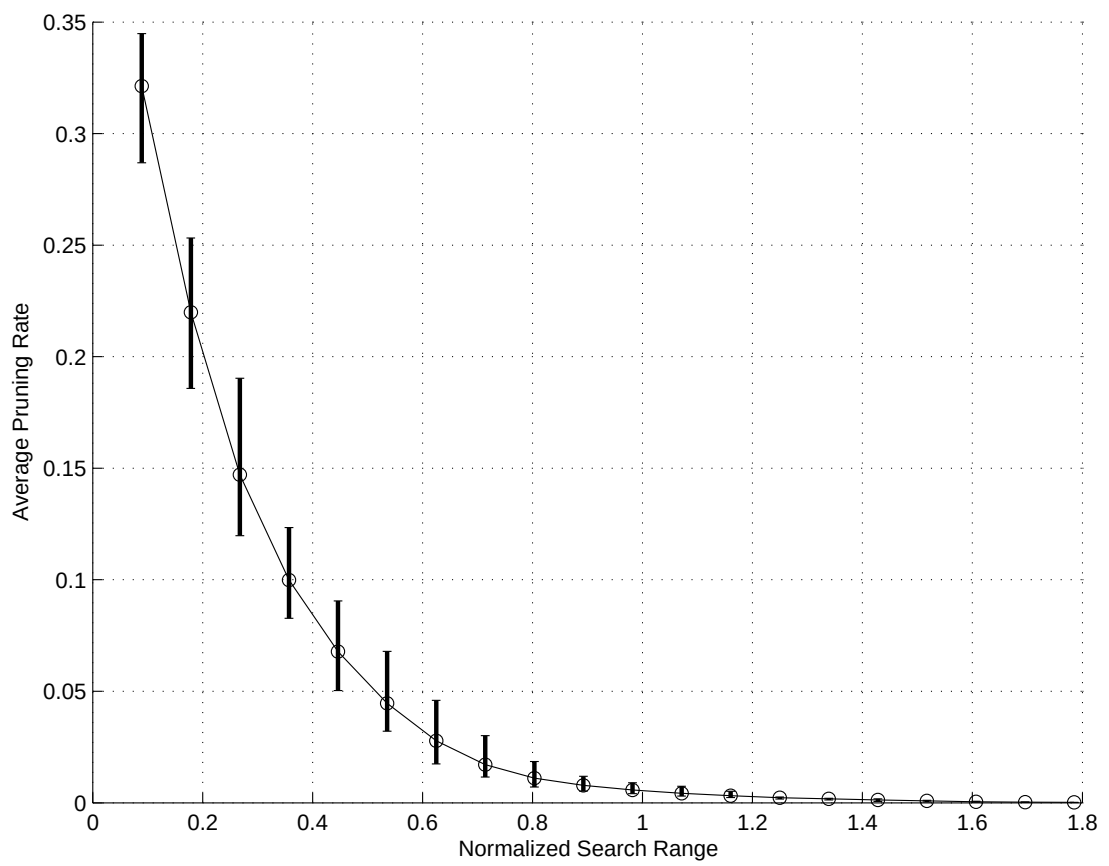


Figure 3.9: Average Pruning Rate vs. Normalized Search Range (Mitochondrial DNA Sequences)

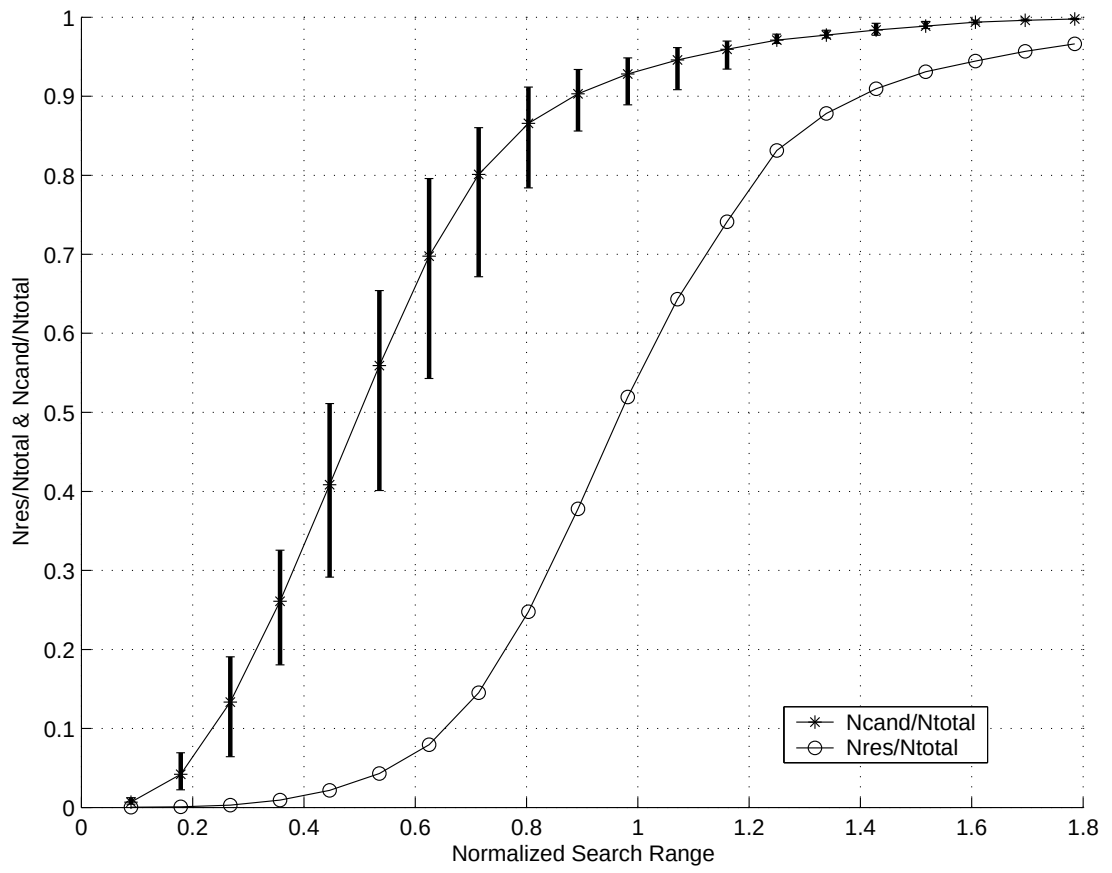


Figure 3.10: $\frac{N_{res}}{N_{total}}$ & $\frac{N_{cand}}{N_{total}}$ vs. Normalized Search Range (Mitochondrial DNA Sequences)

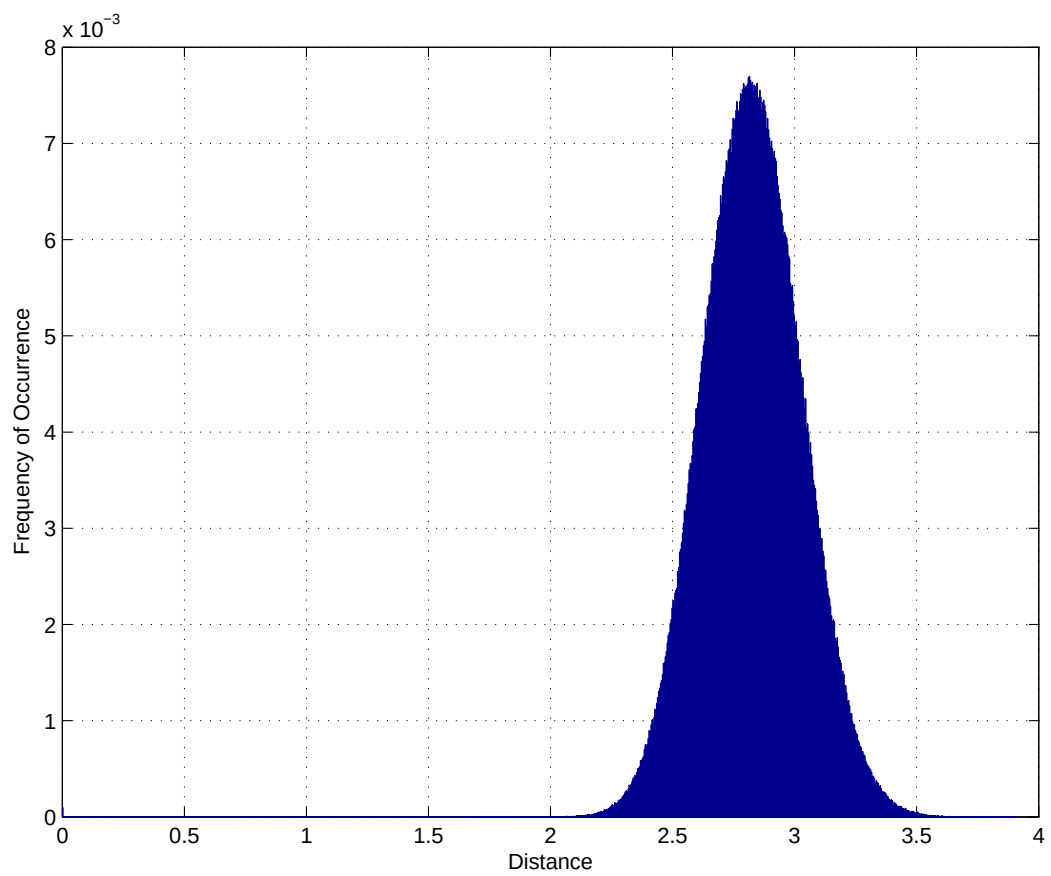


Figure 3.11: Histogram of Distance Distribution (Synthetic Vectors)

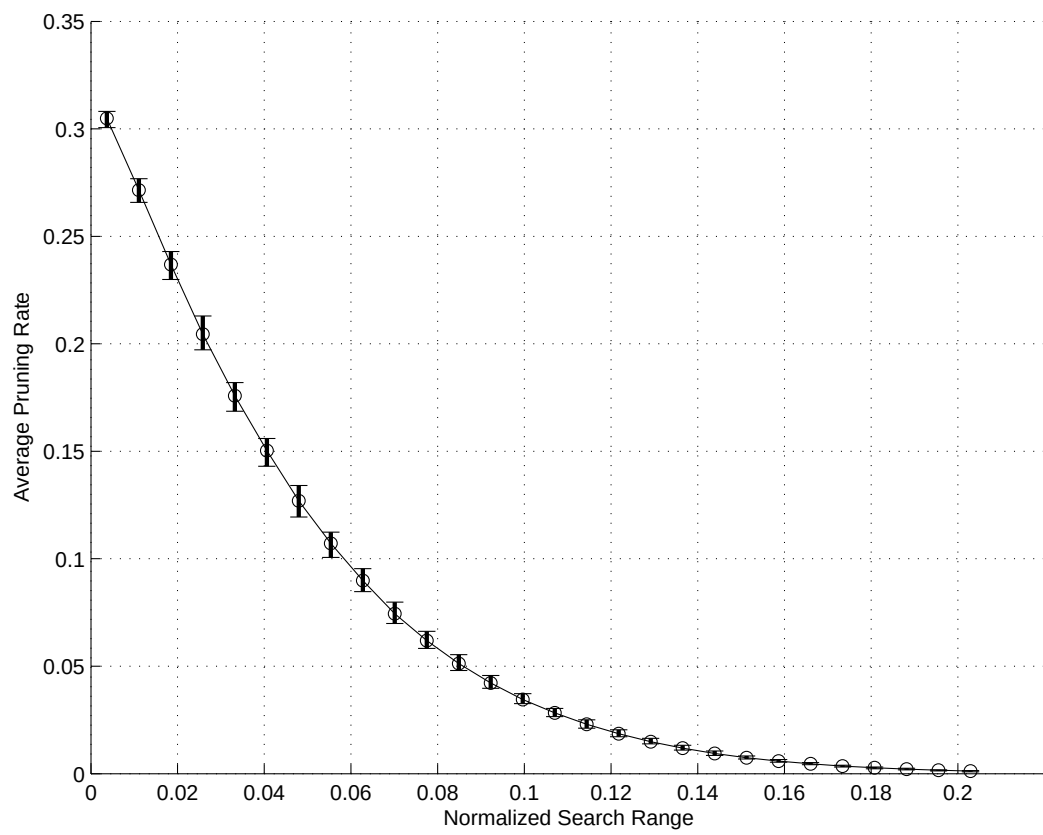


Figure 3.12: Average Pruning Rate vs. Normalized Search Range (Synthetic Vectors)

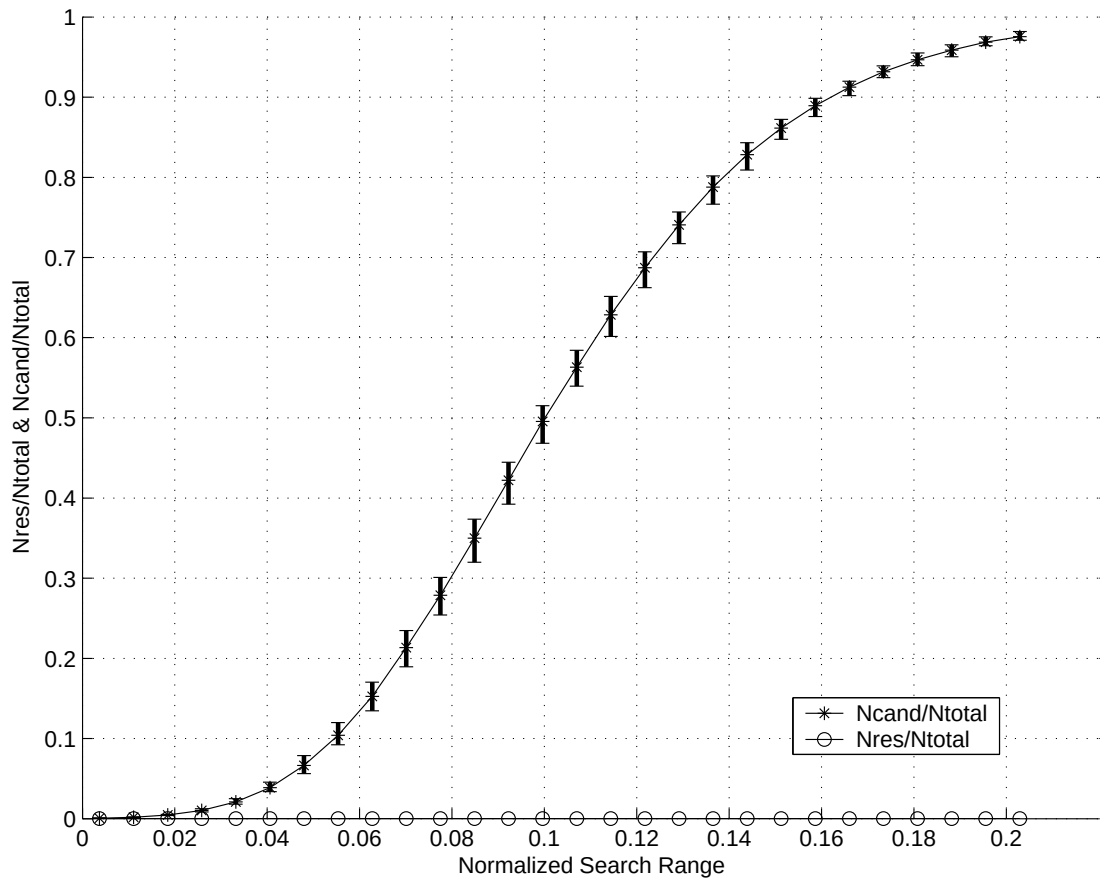


Figure 3.13: $\frac{N_{res}}{N_{total}}$ & $\frac{N_{cand}}{N_{total}}$ vs. Normalized Search Range (Synthetic Vectors)

Table 3.2: Estimated Values and Experimental Results of Range Threshold

Range Threshold	Image Color Histograms	mtDNA Sequences	Synthetic Vectors
Estimated Value	1.1602	1.3675	0.2793
Experimental Result	0.3642	0.4643	0.0813

The ratio of estimated value to experimentally derived value is roughly 3 – 4 in all these cases. For the experimental results, the mtDNA sequence data have the maximum range threshold, and the synthetic vector data have the smallest range threshold, which also supports the conclusion drawn in Section 3.3.

3.5 Summary

In this section, the performance limitations of similarity search approaches based on the triangle inequality have been explored. The conclusion that the pruning rate and the range threshold are affected by the statistical characteristics of the data set is drawn. The search methods based on metric spaces should perform well on a data set with a larger distance variance. Given a data set, a method to estimate the range threshold and check the feasibility of the search method is presented. Experiments are carried out on three different data sets. The experimental results support the conclusions mentioned above.

Chapter 4

Evaluation of Performance and Utility of Data Extraction Methods and Metrics

4.1 Performance Evaluation of Data Extraction Methods and Metrics

Metric selection and data extraction are two critical steps of generic database similarity search methods before constructing of a tree-structured index. A metric has to be selected to evaluate the degree of dissimilarity between objects. Data extraction methods transforms the data in a way that allows the subsequent search method to achieve optimal performance. One of the most widely used data extraction methods employs PCA (principal component analysis) to first reduce the dimensionality of the original data. One challenge of selecting metrics and data extraction methods is how to evaluate

and compare the performance of different approaches. Usually, the number of metric computations determined by the candidate set size is used as the criterion to evaluate search performance. However, this criterion only works for comparing different search methods that produce the same result set. In other words, the comparison of candidate set size is feasible when the search results are the same. Different metrics and data extraction methods will yield different result sets. Therefore, a new criterion that considers both the candidate set size and result set size is required. The ratio between N_{res} and N_{cand} is chosen to meet this requirement. A proper metric or good data extraction method should yield a large value of $\frac{N_{res}}{N_{cand}}$. In other words, N_{res} should be close to N_{cand} , which means few unnecessary metric computations are performed during the search. Actually, the value of $\frac{N_{res}}{N_{cand}}$ indicates the efficiency of a data extraction method. Moreover, the normalized search range $r_{normalized}$ is used instead of the original search range. Therefore, a figure that illustrates the values of $\frac{N_{res}}{N_{cand}}$ against different $r_{normalized}$ is used to evaluate the performance of different metrics and data extraction methods. In such a figure, the performance is indicated by the area under the curve of $\frac{N_{res}}{N_{cand}}$, and a larger area means a better performance with respect to search efficiency. Figure 4.1 is an example figure of two different methods a and b . The area under curve a is larger than that under curve b . Thus, the search performance of using data extraction method a is better than that using b . In order to make this criterion more suitable for practical applications, an improved performance evaluation method is also provided. Assume the search ranges are distributed exponentially, i.e.,

$$p(r_{normalized}) = e^{-\gamma r_{normalized}}. \quad (4.1)$$

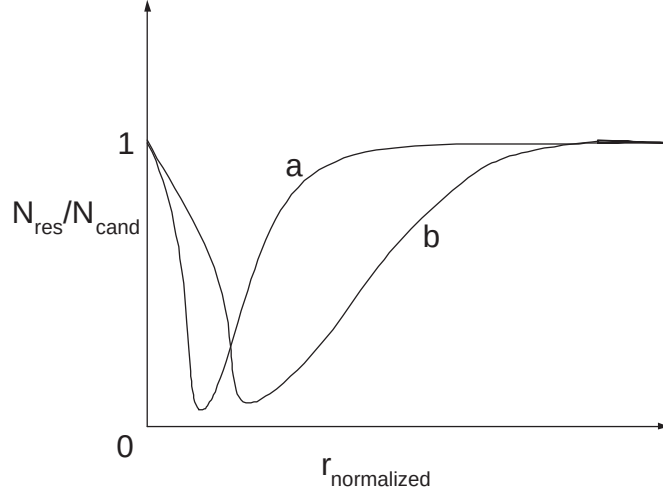


Figure 4.1: Comparing the Performance of Two Data Extraction Methods

The search performance for search ranges smaller than r can be evaluated by an integration,

$$\phi(r) = \int_0^r \frac{N_{res}}{N_{cand}} e^{-\gamma \hat{r}} d\hat{r}. \quad (4.2)$$

In the following sections, several experiments that use this criterion to compare the performance of different metrics and data extraction methods are presented. Selection of three Minkowski metrics and three data extraction methods using different settings are evaluated and compared. The rowing photo database described in Section 2.4.1 is used.

4.1.1 Comparison Using Different Minkowski Metrics

Metric selection is one of the most important problems of similarity search techniques based on metric spaces. A proper metric space is desired that can best preserve the characteristics of the data that are embedded in it. There are many available metrics, such as the Euclidean distance, Manhattan distance and Tangent distance. In this sec-

tion, the performance of three different Minkowski metrics are compared to demonstrate how the metric selection criterion is used. For $d - dimensional$ objects a and b , the Minkowski metric is

$$l_k(a, b) = \left(\sum_{i=1}^d |a_i - b_i|^k \right)^{1/k}; \quad (4.3)$$

Three Minkowski metrics are compared, i.e.,

- l_1 Manhattan Distance

$$d(a, b) = \sum_i |a_i - b_i|; \quad (4.4)$$

- l_2 Euclidean Distance

$$d(a, b) = \sqrt{\sum_i |a_i - b_i|^2}; \quad (4.5)$$

- l_∞

$$d(a, b) = \max_i |a_i - b_i|. \quad (4.6)$$

In Figure 4.2, the performance of the search methods using each of these three Minkowski metrics is illustrated. The area under the curve of l_∞ is the largest, and that under the curve of l_1 is the smallest. Therefore, the metric l_∞ yields the best performance. Assume the search range is exponentially distributed with the parameter $\gamma = 1$.

$$\phi(r) = \int_0^r \frac{N_{res}}{N_{total}} e^{-\hat{r}} d\hat{r}. \quad (4.7)$$

The values of $\phi(r)$ according to different metrics listed in Table 4.1 make the results more clear. The search method using l_∞ as the metric yields the largest value of $\phi(r)$ and the best search performance with respect to search efficiency. The table lists all the values of $\phi(r)$ for different combinations of Minkowski metrics and data extraction

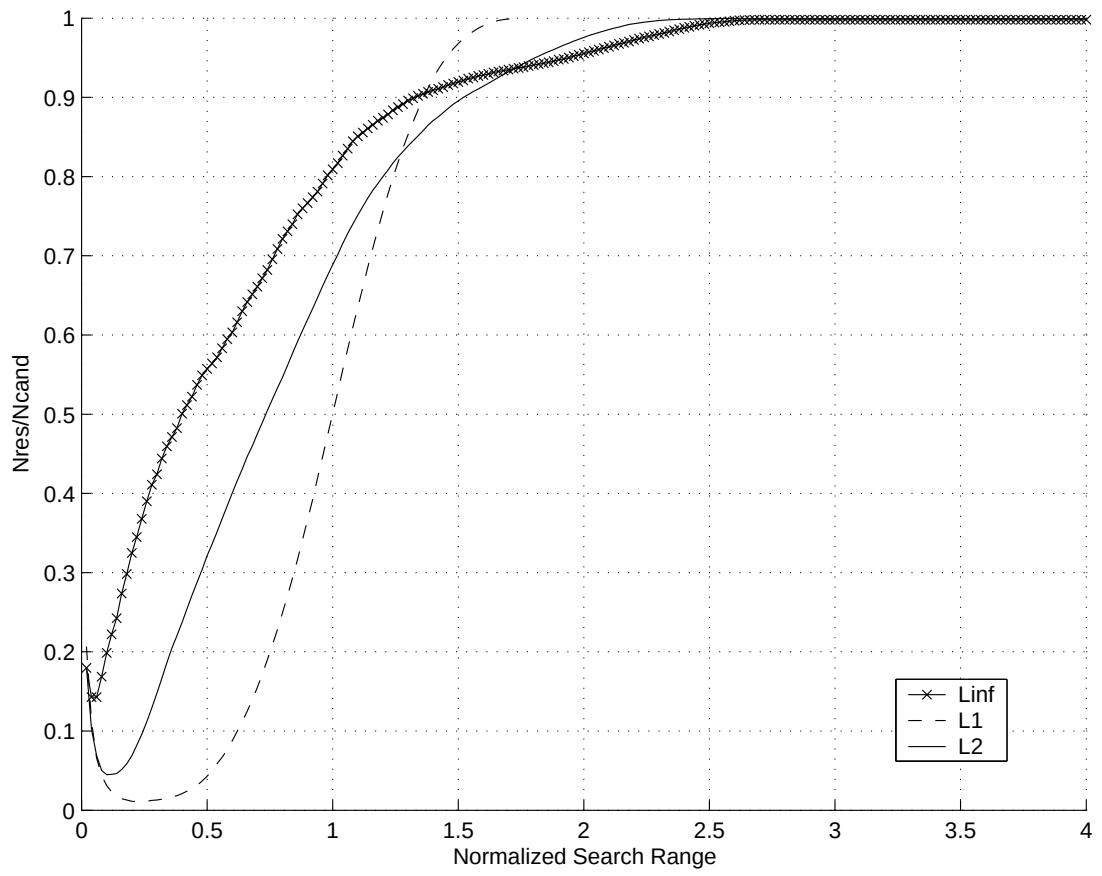


Figure 4.2: Comparing Different Minkowski Metrics by $\frac{N_{res}}{N_{cand}}$

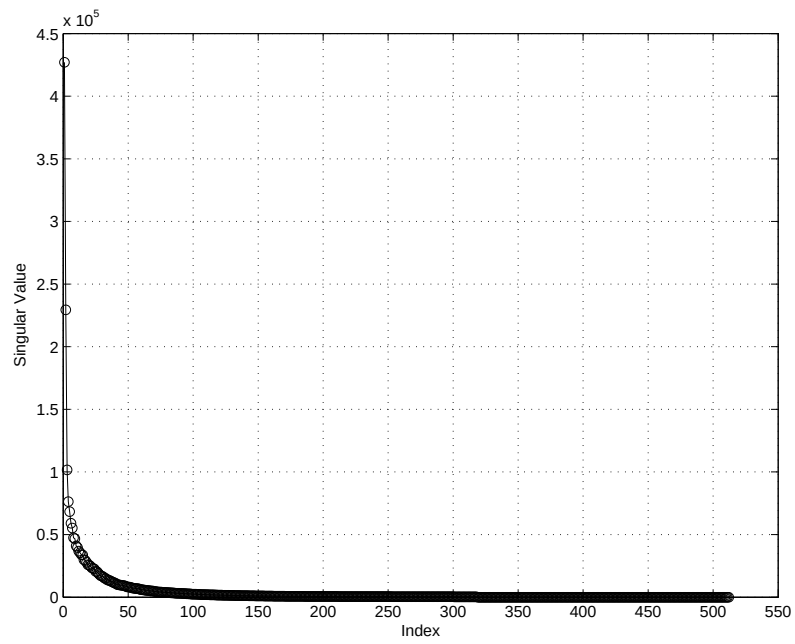
Table 4.1: Comparing Different Metrics by the Value of ϕ_r , $r = 4$

	l_1	l_2	l_∞
ϕ_r	0.3779	0.4852	0.6178

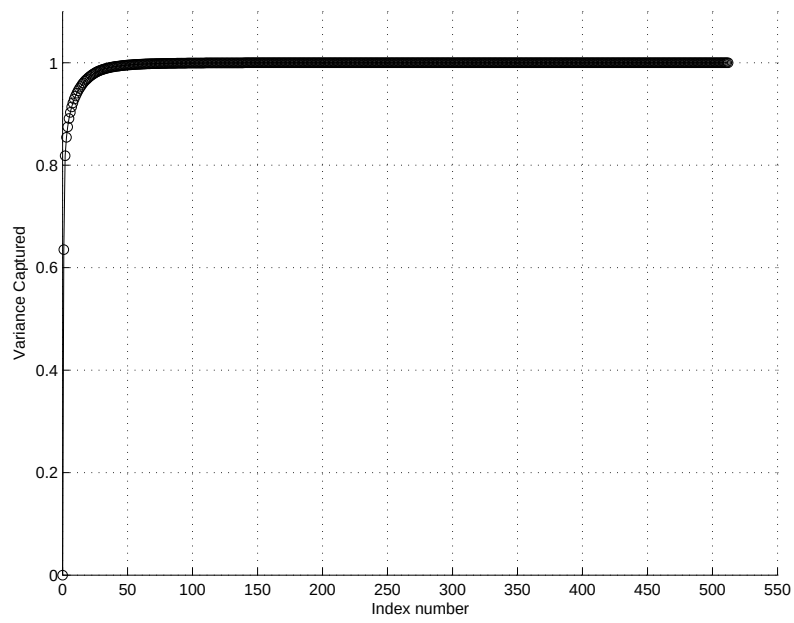
methods is presented in Section 4.1.2.

4.1.2 Comparison Using Different Data Extraction Methods

There are many data extraction methods that can be used to transform the data, such that a better search performance can be achieved. PCA (principal component analysis) is one of the methods that have been widely used to reduce high dimensionality without losing much of the original data content. The key of PCA is the number of principal components being kept to represent the original data. A set of experiments are conducted to compare the performance between PCA data extractions that keep different numbers of principal components. l_∞ is chosen as the metric based on the conclusion of Section 4.1.1. The data extracted from the rowing photo collection are first mean centered. Then, an SVD (singular value decomposition) is performed on the data. Figure 4.3(a) illustrates the singular values obtained from SVD, and Figure 4.3(b) is the scree plot of the singular values [31]. The first 2 principal components capture 81.85% of the total variation, and the first 6 principal components capture 90.32%. Therefore, three different settings are compared: no PCA, PCA2 (PCA keeping the first 2 principal components) and PCA6 (PCA keeping the first 6 principal components). Figure 4.4 illustrates the value of $\frac{N_{res}}{N_{cand}}$ for these three cases versus the normalized search range. The curve associated with the PCA that kept the first 2 principal components (PCA2) has the largest area under it. PCA2 achieves the best search performance, and the search method without first using PCA to reduce the dimensionality yields the worse



(a) Singular Values



(b) Scree Plot of Singular Values

Figure 4.3: Singular Values Obtained by Performing SVD on the Rowing Photo Data Set

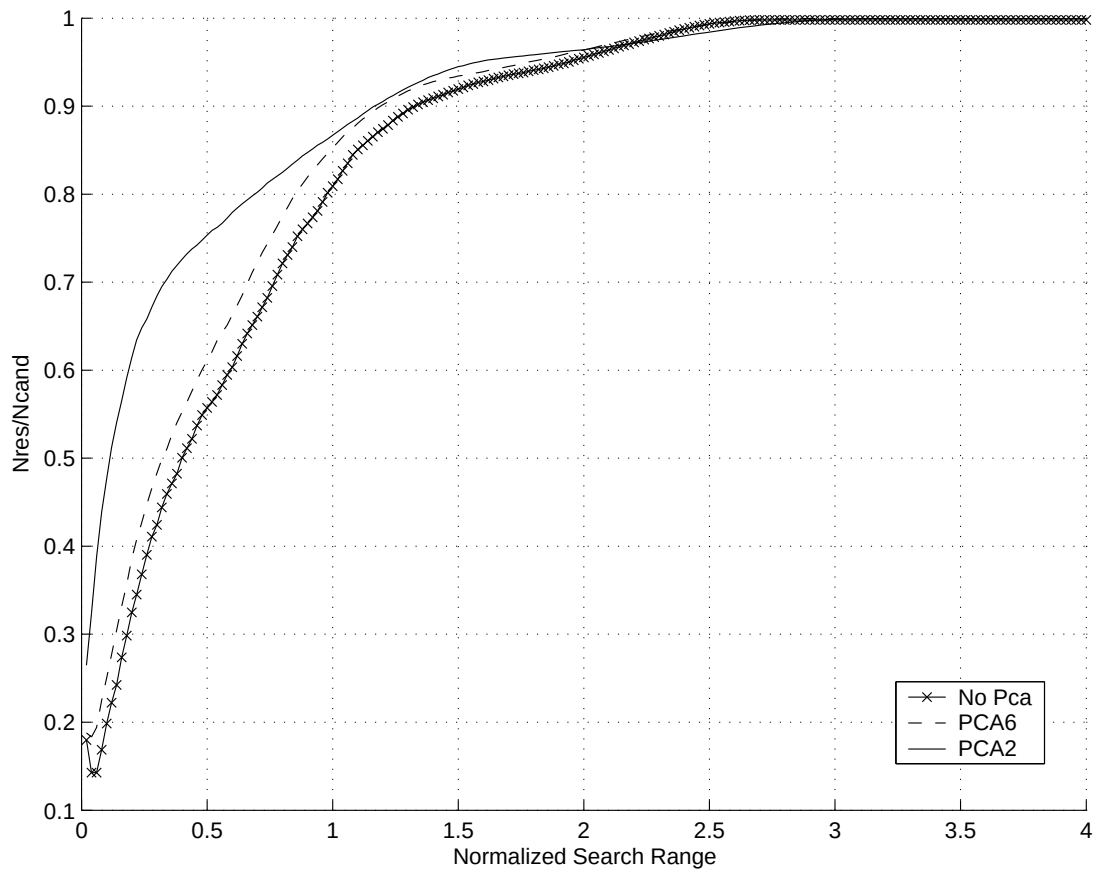


Figure 4.4: Comparing Different PCA by $\frac{N_{res}}{N_{cand}}$

Table 4.2: Comparing Different PCA by $\phi(r)$, $r = 4$

	No PCA	PCA 6	PCA 2
$\phi(r)$	0.6178	0.7499	0.6564

Table 4.3: Comparing Different Combinations of Minkowski Metrics and PCA Using Different Settings by $\phi(r)$, $r = 4$

	l_1	l_2	l_∞
No PCA	0.3779	0.4852	0.6178
PCA6	0.5521	0.5947	0.6564
PCA2	0.7567	0.7442	0.7499

performance. Table 4.2 lists the values of $\phi(r)$ for the three cases. The results listed in Table 4.2 agree with the conclusion obtained from Figure 4.4. Therefore, data extraction methods using PCA2 can make the search methods more efficient. Table 4.3 lists values of $\phi(r)$ of different combinations of Minkowski metrics and data extraction methods using PCA. For original data without PCA, the method using l_∞ yields the largest value of $\phi(r)$, which is much larger than that of the method using l_1 . For extraction methods using PCA6, the difference between three $\phi(r)$ of different Minkowski metrics are small. For extraction methods using PCA2, three $\phi(r)$ are even more close, and the one related to l_1 is the largest. This result suggests that l_∞ is the best choice for original data. When the number of principle components kept decreases, the difference of search efficiency between three metrics become smaller.

4.2 Utility Evaluation of Data Extraction Methods and Metrics

The two experiments in Section 4.1.1 and 4.1.2 demonstrate how to use $\frac{N_{res}}{N_{cand}}$ and $\phi(r)$ as criteria to evaluate and compare the performance of different data extraction methods. However, the search performance evaluated by this criterion is the search efficiency. In other words, this criterion answers the question of how efficient a search method is. Another critical problem that users care about is the utility of search results. In other words, whether or not does the search method return results that are useful to users? It is not a problem for the discussions in other chapters, because the search methods obtain the same results in those chapters. However, using different metrics and data extraction methods will yields different search results. Therefore, the utility of search results associated with different metrics and data extraction methods needs to be evaluated. Since an automatic evaluation method is not available, the utilities are evaluated manually here as a preliminary result.

The 1092 photos of the rowing team data set belong to various types, including photos looking down on a boat with rowers, photos of boats in a race at different locations and events, taken from the side, and photos of a rowing team in groups. During evaluation, all the photos are divided into 8 types, and 3 photos are chosen from each type as search targets. Four example photos belonging to different types are presented in Figure 4.5. Then, the results are judged manually and a score is assigned for different data extraction methods. The utility is indicated by the sum of scores. Both the utilities of different metrics and data extraction methods using PCA are compared. Because three different metrics are compared, the scores are 1, 2 and 3 (rank order).



(a) Photo Looking Down on a Boat with Rowers



(b) Photo of a Boat in a Race (at Southeast Regional Event) Taken from the Side



(c) Photo of a Boat in a Race (at Oak Ridge) Taken from the Side



(d) Photo of a Rowing Team Standing in Group

Figure 4.5: Examples Photos Belong to Different Types

For different PCA data extractions, because three different methods are used, scores are also 1, 2 and 3. A smaller score means the search result is more useful to users. An example of how to score different results is presented in Figure 4.6 and 4.7. The normalized search range in this example is 0.06.



(a) Target Photo



(b) Result #1 of Method
Using l_1



(c) Result #1 of Method
Using l_2



(d) Result #2 of Method
Using l_2



(e) Result #3 of Method
Using l_2



(f) Result #4 of Method Us-
ing l_2

Figure 4.6: Search Results of Methods Using Different Metrics (1)



(a) Result #1 of Method Using l_∞



(b) Result #2 of Method Using l_∞



(c) Result #3 of Method Using l_∞



(d) Result #4 of Method Using l_∞



(e) Result #5 of Method Using l_∞



(f) Result #6 of Method Using l_∞



(g) Result #7 of Method Using l_∞



(h) Result #8 of Method Using l_∞



(i) Result #9 of Method Using l_∞

Figure 4.7: Search Results of Methods Using Different Metrics (2)

Table 4.4: Utility Scores for Different Minkowski Metrics

	l_1	l_2	l_∞
Scores	50	39	37

Table 4.5: Utility Scores for Different Data Extraction Methods

	No PCA	PCA 6	PCA 2
Scores	34	36	51

The result of the search using the l_1 metric only contains one photo that is the same as the target. The results of l_2 contain 4 photos that are all very similar to the target. In the 9 results of l_∞ , 6 of them are very similar to the target, while 3 of them are not. Therefore, the scores assigned to l_1 , l_2 and l_∞ are 3, 1 and 2, respectively. This scoring is subjective, in that, given the l_2 result set, it appears that the l_1 result set did not contain a sufficient number of very similar photos. Similarly, three of the nine photos of the l_∞ result set were substantially different for the others. In this manner, all result sets of 24 targets are evaluated for both different metrics and data extraction methods. Some examples of search results for different metrics or data extraction methods are presented in Section 4.3 to demonstrate search utility. The final scores are listed in Table 4.4 and 4.5. The results using l_2 metric have the smallest score, while those using l_∞ has the highest one. Therefore, the search utility of l_∞ is lowest, although its search process is most efficient indicated by $\phi(r)$. Comparing performance of different data extraction methods, the method without PCA achieves the highest utility, and PCA2 has the lowest utility.

Comparing the search utilities of different methods to the results using the $\frac{N_{res}}{N_{cand}}$ and $\phi(r)$ criteria, it is clear that while the later can successfully evaluate and compare the

search efficiencies of different metrics and data extraction methods, search utility is also a critical issue to be considered for overall search performance. Therefore, an improved evaluation criterion that considers both search efficiency and utility is desirable and is a suitable topic for future research.

4.3 Sample Results of Image Database Search

Several examples that typify the result sets obtained during the scoring and evaluation process described in the last section. All photos are from the data set of rowing images and illustrate both the diversity of images in the data set and the utility of the search methodologies.

4.3.1 Search Results of Different Minkowski Metrics

Two examples of search results are presented in Figure 4.8, 4.9, 4.10 and 4.11. The normalized search range is 0.06. Figure 4.8 shows the target photo and the result set using the l_1 metric. Figure 4.9 shows the result set using the l_∞ metric with the same target. Figures 4.10 and 4.11 show similar results for a different target image.

4.3.2 Search Results of Different PCA

Three examples of search results are presented in Figure 4.12, 4.13, 4.14, 4.15, 4.16, 4.17 and 4.18. The normalized search range is 0.06. Figure 4.12 show the target photo and the result sets using no PCA and PCA6 data extractions. Figure 4.13 shows the result set for the PCA2 data extraction. Figure 4.14, 4.15, 4.16, 4.17 and 4.18 show two additional cases using different target images.



(a) Target Photo



(b) Result #1 of Method
Using l_1



(c) Result #1 of Method
Using l_2

Figure 4.8: Search Results of Methods Using Different Metrics (a1)



(a) Result #1 of Method
Using l_∞



(b) Result #2 of Method
Using l_∞



(c) Result #3 of Method
Using l_∞



(d) Result #4 of Method
Using l_∞

Figure 4.9: Search Results of Methods Using Different Metrics (a2)



(a) Target Photo



(b) Result #1 of Method Using l_1



(c) Result #2 of Method Using l_1



(d) Result #1 of Method Using l_2



(e) Result #2 of Method Using l_2



(f) Result #3 of Method Using l_2



(g) Result #4 of Method Using l_2

Figure 4.10: Search Results of Methods Using Different Metrics (b1)



(a) Result #1 of Method
Using l_∞



(b) Result #2 of Method
Using l_∞



(c) Result #3 of Method
Using l_∞



(d) Result #4 of Method
Using l_∞



(e) Result #5 of Method
Using l_∞



(f) Result #6 of Method Us-
ing l_∞

Figure 4.11: Search Results of Methods Using Different Metrics (b2)



(a) Target Photo



(b) Result #1 of Method
without Using PCA



(c) Result #1 of Method
Using PCA6



(d) Result #2 of Method
Using PCA6

Figure 4.12: Search Results of Methods Using Different PCA (a1)



(a) Result #1 of Method Using PCA2



(b) Result #2 of Method Using PCA2



(c) Result #3 of Method Using PCA2



(d) Result #4 of Method Using PCA2



(e) Result #5 of Method Using PCA2



(f) Result #6 of Method Using PCA2



(g) Result #7 of Method Using PCA2

Figure 4.13: Search Results of Methods Using Different PCA (a2)



(a) Target Photo



(b) Result #1 of Method
without Using PCA



(c) Result #1 of Method
Using PCA6



(d) Result #2 of Method
Using PCA6



(e) Result #3 of Method
Using PCA6



(f) Result #4 of Method Us-
ing PCA6

Figure 4.14: Search Results of Methods Using Different PCA (b1)



(a) Result #1 of Method Using PCA2



(b) Result #2 of Method Using PCA2



(c) Result #3 Method Using PCA2



(d) Result #4 of Method Using PCA2

Figure 4.15: Search Results of Methods Using Different PCA (b2)



(a) Target Photo



(b) Result of #1 of Method without Using PCA



(c) Result #2 of Method without Using PCA



(d) Result #3 of Method without Using PCA



(e) Result #4 of Method without Using PCA



(f) Result #5 of Method without Using PCA

Figure 4.16: Search Results of Methods Using Different PCA (c1)



(a) Result #1 of Method Using PCA6



(b) Result #2 of Method Using PCA6



(c) Result #3 of Method Using PCA6



(d) Result #4 of Method Using PCA6



(e) Result #5 of Method Using PCA6



(f) Result #6 of Method Using PCA6



(g) Result #7 of Method Using PCA6

Figure 4.17: Search Results of Methods Using Different PCA (c2)



(a) Result #1 of Method Using PCA2



(b) Result #2 of Method Using PCA2



(c) Result #3 of Method Using PCA2



(d) Result #4 of Method Using PCA2



(e) Result #5 of Method Using PCA2



(f) Result #6 of Method Using PCA2



(g) Result #7 of Method Using PCA2



(h) Result #8 of Method Using PCA2

Figure 4.18: Search Results of Methods Using Different PCA (c3)

4.3.3 Result Set Size Versus Search Range

Search utility can be affected by the number of search results, which will increase when the search range increases. Demonstration and analysis of the relation between result set size and search range would help to find the relation between search utility and search range. Figure 4.19 and 4.20 illustrate the result set size against different normalized search ranges for different metrics and data extraction methods respectively. In Figure 4.19, the original data are used, and in Figure 4.20, l_∞ is used as the metric. The result set sizes are different for different Minkowski metrics, while they are close for different data extraction methods using PCA. This result indicates that the effect of PCA on search utility is not as much as that of different metrics.

Since all the data objects are embedded in a metric space, the search target and results can be illustrated by points in the metric space. Four examples that illustrate the candidate sets and result sets under different search ranges are provided. Figure 4.22, 4.23, 4.24, and 4.25 illustrate the candidate sets and results sets for different metrics and search ranges using the search target presented in Figure 4.21(a), 4.21(b), 4.21(c), and 4.21(d), respectively. The projection of the feature vector of each image onto the basis corresponding to the first two singular values is plotted as a (black) point. Each of these figure illustrates the candidate and result sets for nine searches, which all use the same target image. The nine searches use all combinations of three metrics (l_1 , l_w , and l_∞) and three normalized search ranges (0.06, 0.16 and 0.4). These plots illustrate the relative sizes of the candidate and result sets, as well as the shapes of the result sets. The candidate objects are plotted as red stars, the search results are plotted as green circles, and the blue circles and vertices in figures indicate the search range.

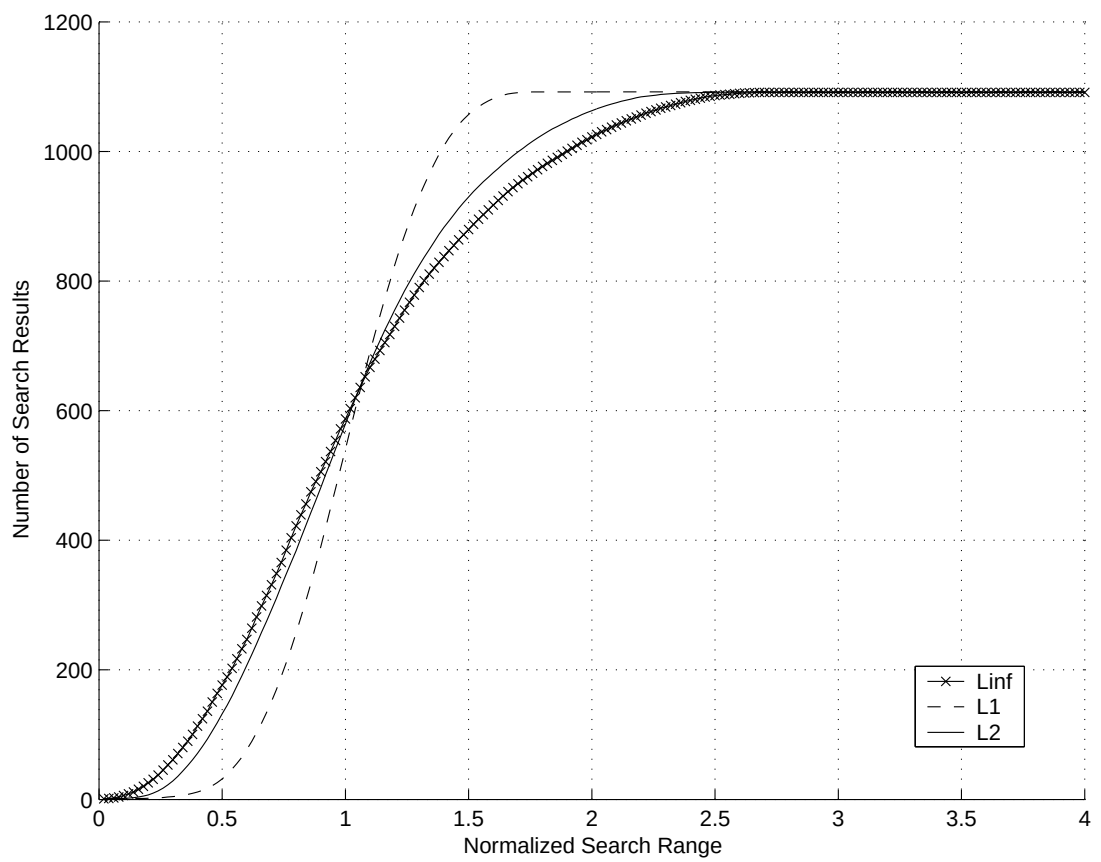


Figure 4.19: Number of Search Results vs. Normalized Search Range (Different Minkowski Metrics)

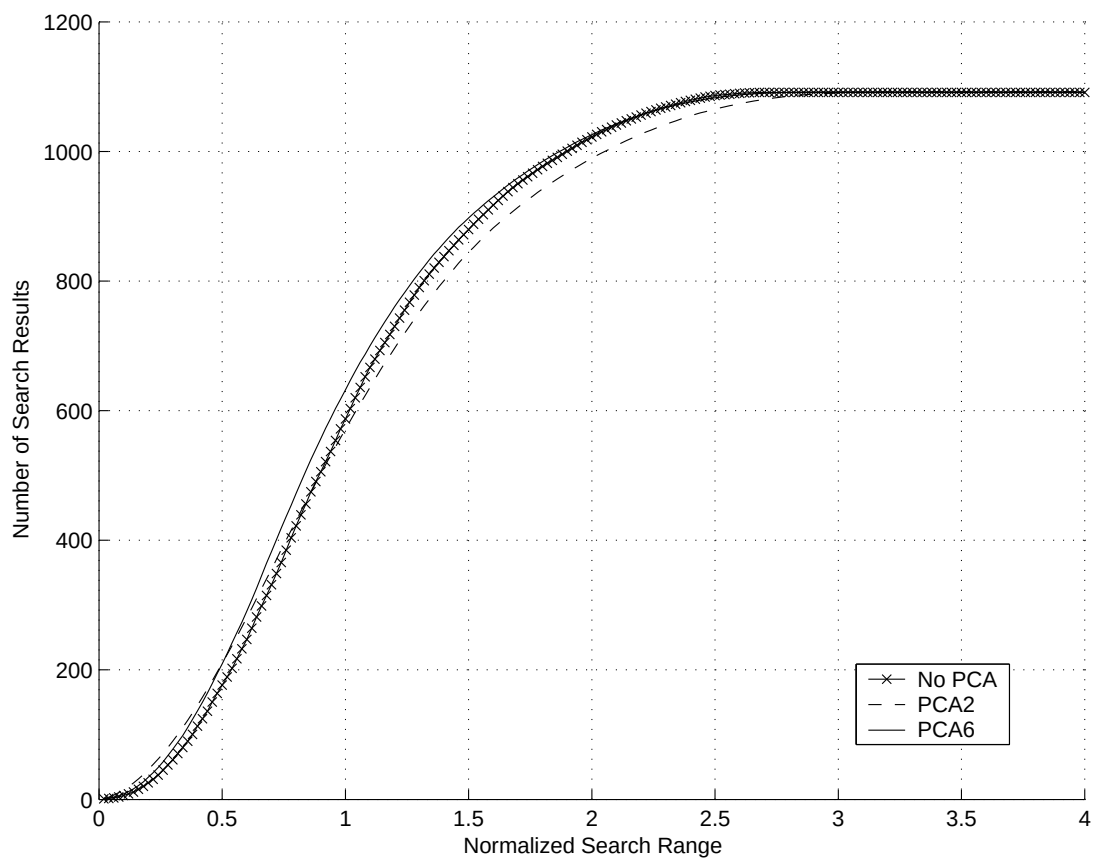


Figure 4.20: Number of Search Results vs. Normalized Search Range (Different Data Extraction Methods)



(a) Search Target #1



(b) Search Target #2

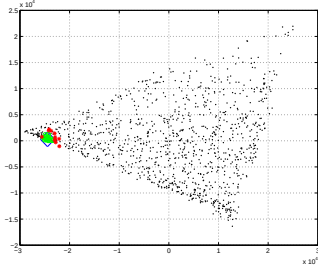


(c) Search Target #3

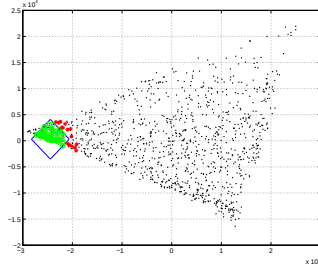


(d) Search Target #4

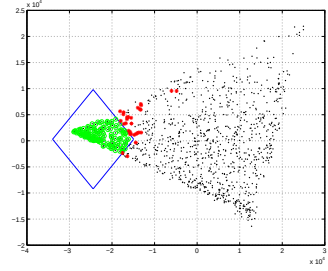
Figure 4.21: Search Targets



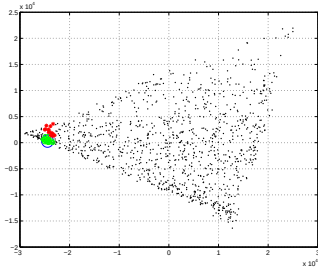
(a) Results of Method Using l_1 , $r_{normalized} = 0.06$



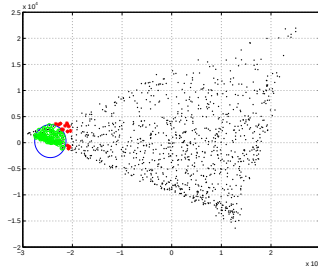
(b) Results of Method Using l_1 , $r_{normalized} = 0.16$



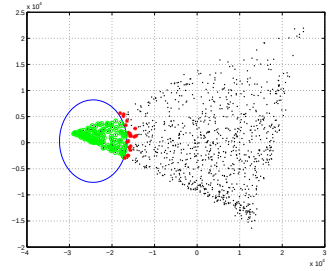
(c) Results of Method Using l_1 , $r_{normalized} = 0.4$



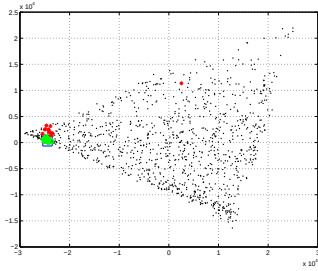
(d) Results of Method Using l_2 , $r_{normalized} = 0.06$



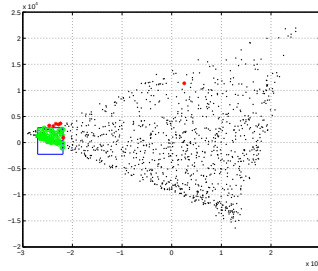
(e) Results of Method Using l_2 , $r_{normalized} = 0.16$



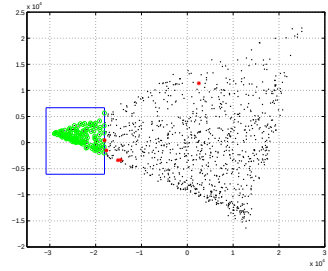
(f) Results of Method Using l_2 , $r_{normalized} = 0.4$



(g) Results of Method Using l_∞ , $r_{normalized} = 0.06$

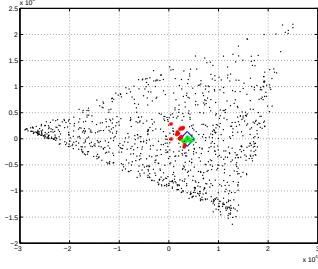


(h) Results of Method Using l_∞ , $r_{normalized} = 0.16$

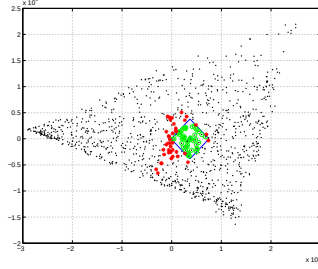


(i) Results of Method Using l_∞ , $r_{normalized} = 0.4$

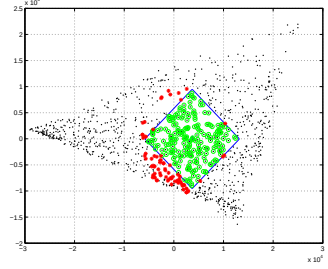
Figure 4.22: Search Results and Candidate Set of Different Metrics and Search Ranges Using Search Target #1



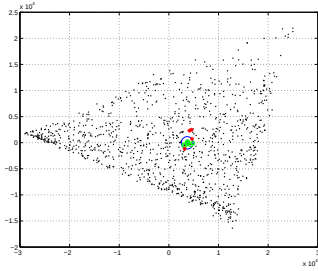
(a) Results of Method Using l_1 , $r_{normalized} = 0.06$



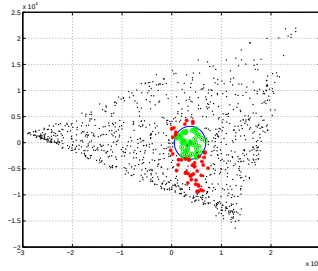
(b) Results of Method Using l_1 , $r_{normalized} = 0.16$



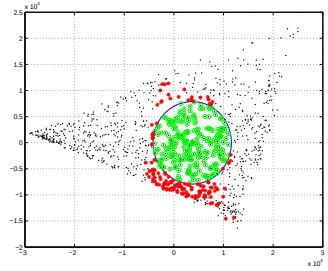
(c) Results of Method Using l_1 , $r_{normalized} = 0.4$



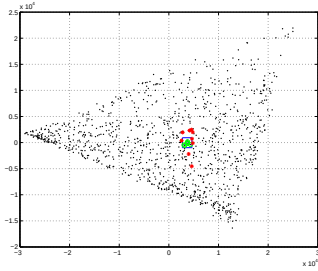
(d) Results of Method Using l_2 , $r_{normalized} = 0.06$



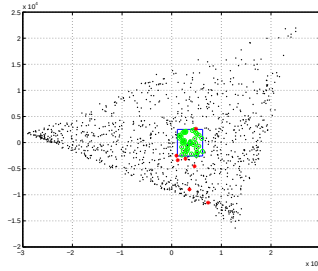
(e) Results of Method Using l_2 , $r_{normalized} = 0.16$



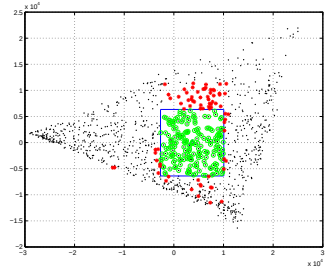
(f) Results of Method Using l_2 , $r_{normalized} = 0.4$



(g) Results of Method Using l_∞ , $r_{normalized} = 0.06$

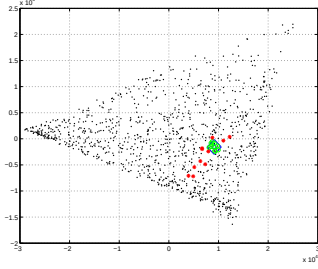


(h) Results of Method Using l_∞ , $r_{normalized} = 0.16$

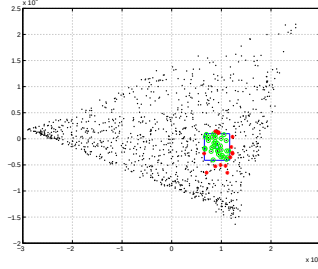


(i) Results of Method Using l_∞ , $r_{normalized} = 0.4$

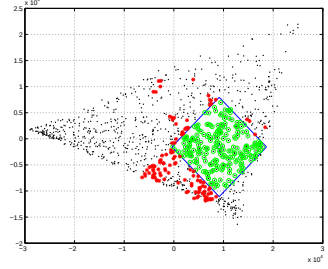
Figure 4.23: Search Results and Candidate Set of Different Metrics and Search Ranges Using Search Target #2



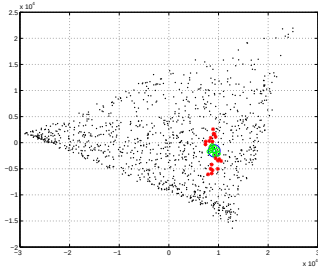
(a) Results of Method Using l_1 , $r_{normalized} = 0.06$



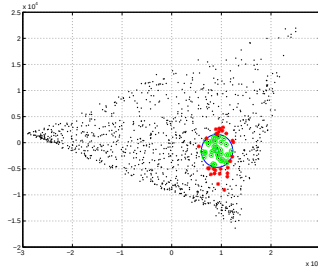
(b) Results of Method Using l_1 , $r_{normalized} = 0.16$



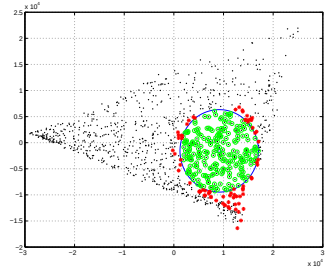
(c) Results of Method Using l_1 , $r_{normalized} = 0.4$



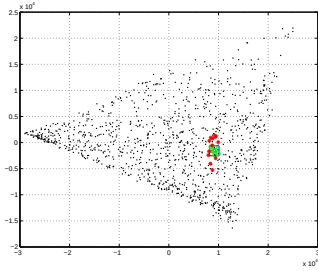
(d) Results of Method Using l_2 , $r_{normalized} = 0.06$



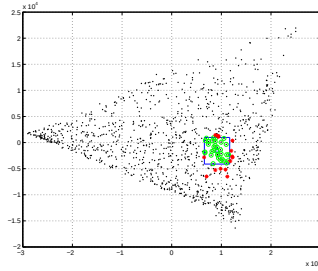
(e) Results of Method Using l_2 , $r_{normalized} = 0.16$



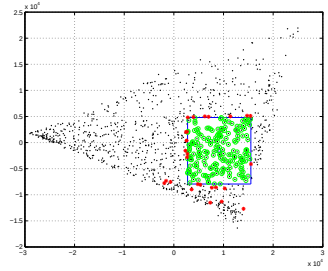
(f) Results of Method Using l_2 , $r_{normalized} = 0.4$



(g) Results of Method Using l_∞ , $r_{normalized} = 0.06$

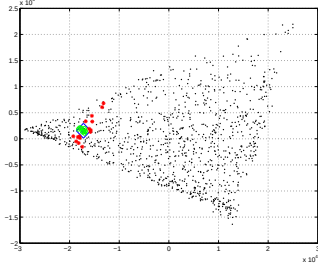


(h) Results of Method Using l_∞ , $r_{normalized} = 0.16$

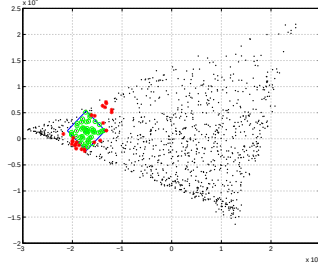


(i) Results of Method Using l_∞ , $r_{normalized} = 0.4$

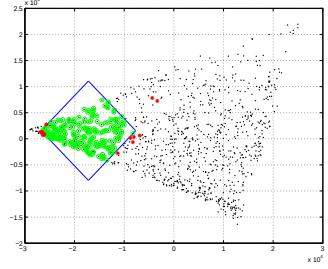
Figure 4.24: Search Results and Candidate Set of Different Metrics and Search Ranges Using Search Target #3



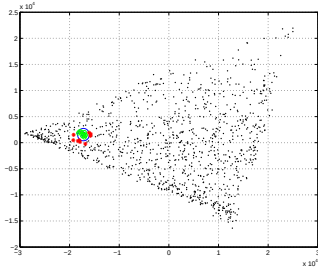
(a) Results of Method Using l_1 , $r_{normalized} = 0.06$



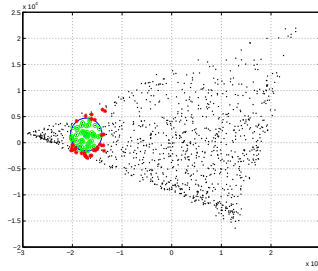
(b) Results of Method Using l_1 , $r_{normalized} = 0.16$



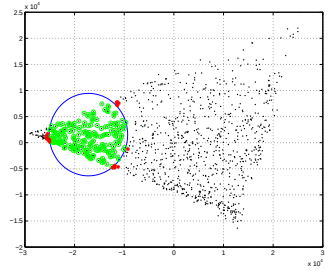
(c) Results of Method Using l_1 , $r_{normalized} = 0.4$



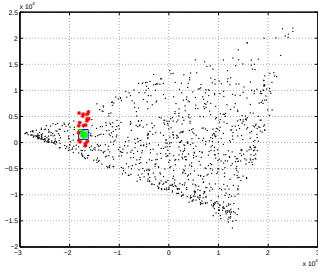
(d) Results of Method Using l_2 , $r_{normalized} = 0.06$



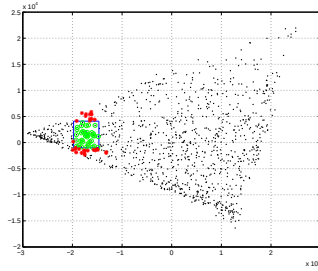
(e) Results of Method Using l_2 , $r_{normalized} = 0.16$



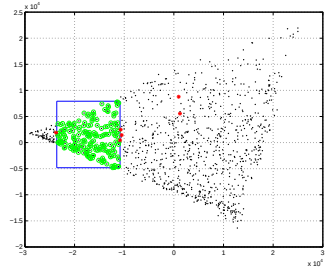
(f) Results of Method Using l_2 , $r_{normalized} = 0.4$



(g) Results of Method Using l_∞ , $r_{normalized} = 0.06$



(h) Results of Method Using l_∞ , $r_{normalized} = 0.16$



(i) Results of Method Using l_∞ , $r_{normalized} = 0.4$

Figure 4.25: Search Results and Candidate Set of Different Metrics and Search Ranges Using Search Target #4

4.4 Summary

A criterion to evaluate and compare the performance of different metrics and data extraction methods is presented. Two example using this criterion to evaluate the performance of methods using different Minkowski metrics and PCA keeping different numbers of principal components are presented. The results imply that l_∞ and PCA2 yield the best search performance. However, the performance evaluated by this criterion only considers search efficiency. Search utility is also an important aspect of search performance. A preliminary result about search utility obtained by manually judging the search results is presented. An automatic way to evaluate the search accuracy and a new criterion to evaluate methods with respect to both search efficiency and utility is desirable and may be the subject of future work. Some sample search results for the rowing photo data set are presented to demonstrate search utilities of different methods.

Chapter 5

Issues Related to Tree Index Structures

5.1 Overview of Issues Related to Tree Index Structures

There are many issues that will affect the performance of a tree-structured index, such as the number of objects stored at leaves and the number of branches descending from each node. Because of the special characteristics of tree-structured indices based on metric spaces, the solutions for these problems are different from those in normal cases. In this chapter, some experiments are performed to demonstrate how these two issues affect search performance, and several conclusions are drawn.

5.2 Maximum Leaf Size

A tree-structured index is built by iteratively splitting the object set until the leaf level is reached. Therefore, one of the important issues of a tree-structured index is selection of the criterion used to stop the iteration. In other words, how many objects

are allowed to be stored in a leaf node? As mentioned in Section 2.2.3, the number of metric computations required to resolve a query is mainly determined by the size of the candidate set, which is composed of the objects contained in the candidate leaf nodes. Thus, the leaf size can affect search performance of a tree-structured index. Several experiments are performed on data sets containing color histograms of images, mitochondrial DNA sequences and rowing photos. Figure 5.1, 5.2, and 5.3 illustrate the numbers of metric computations required by search queries using different maximum leaf sizes.

In Figure 5.1, 5.2 and 5.3, the number of metric computations decreases when the leaf size decreases. The tree with maximum leaf size equal to one yields the best performance. This result is somewhat different from the normal case. Usually, a smaller leaf size means a taller tree, which will cause longer tree traversal times. However, in similarity search based on metric spaces the time required for metric computations is much longer than the tree traversal time. Thus, the main difference between a tree index based on metric spaces and a normal tree is that the tree traversal time is dominated by the number of metric computations, which becomes the key to search performance. Under this criterion, the leaf size of an optimal tree should be as small as possible, i.e., one.

5.3 Number of Branches

The number of branches is another critical parameter of a tree-structured index. Since in this research the tree is stored in memory, I/O time is not an issue. Thus, a tree with a larger branch number usually will have a better performance, because more branches at each node yields a shorter tree and shorter traversal times. As mentioned in Section 5.2,

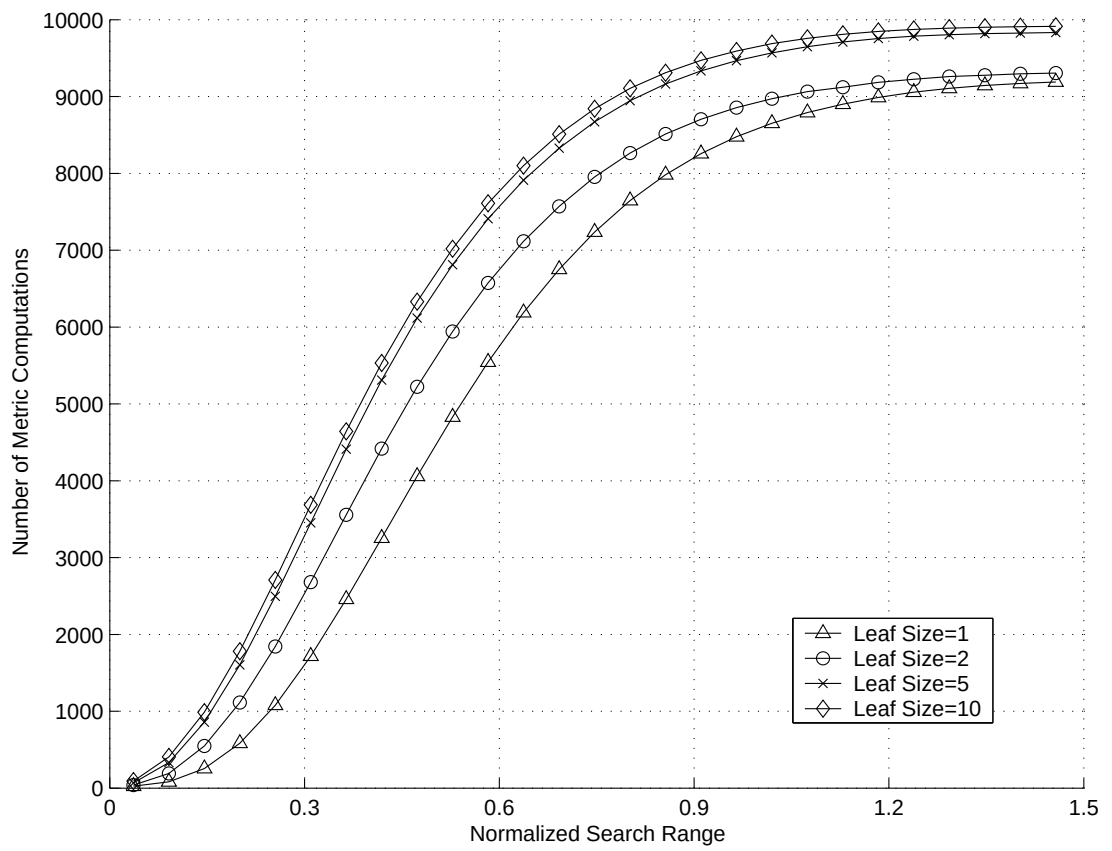


Figure 5.1: Number of Metric Computations under Different Search Range (Color Histogram of Images)

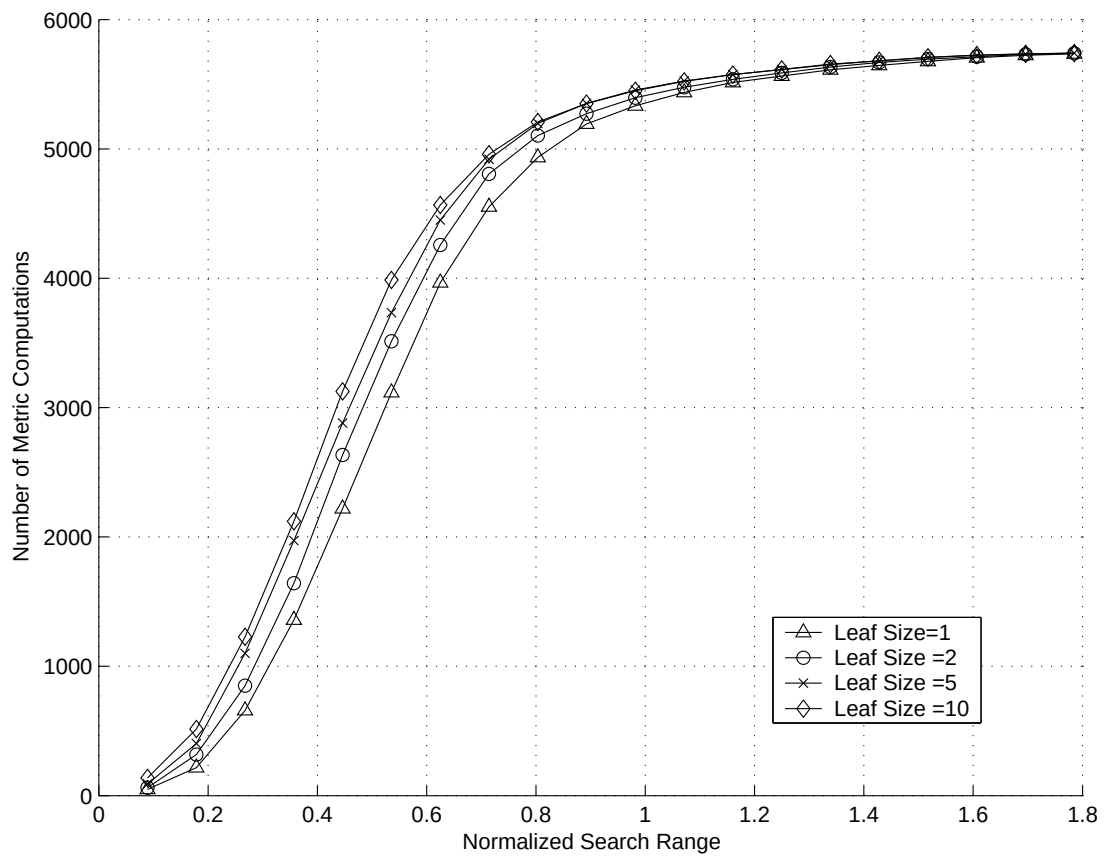


Figure 5.2: Number of Metric Computations under Different Search Range (Mitochondrial DNA Sequences)

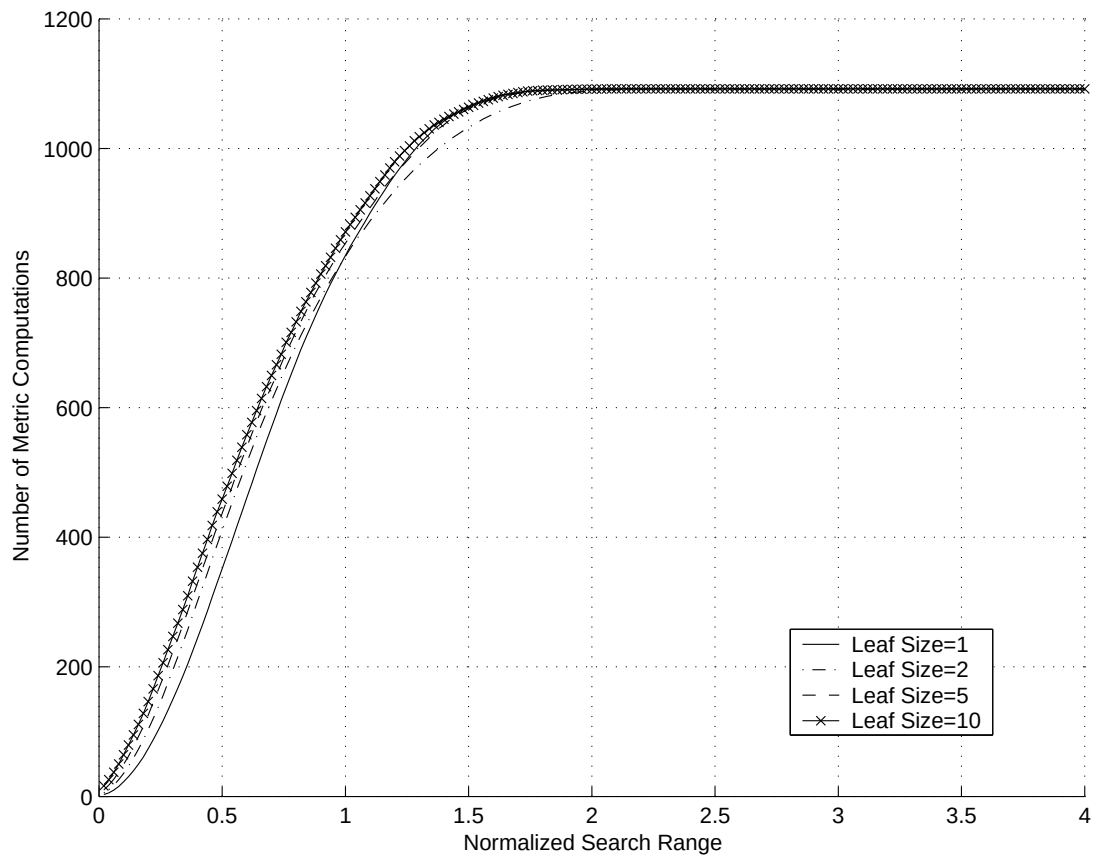


Figure 5.3: Number of Metric Computations under Different Search Range (Rowing Photos)

for a tree-structured index based on metric spaces, this rule is no longer effective. The number of metric computations is the dominant factor for trees based on metric spaces. Figure 5.4, 5.5, and 5.5 illustrate search performance for different number of branches for data sets of image color histograms, mtDNA sequences, and rowing photos, respectively. In all three experiments, the more branches a tree has, the poorer its search performance will be, and the binary trees yields the best performance. This conclusion conflicts with the normal case, which is also because of the special characteristic of a tree-structured index based on metric spaces. However, why a smaller leaf size and number of branches will lead to less metric computation is still unknown. The following section will further explore this issue.

5.4 Tree Height

From the experiments discussed in Section 5.2 and 5.3, a tree with minimum leaf size and minimum number of branches will lead to the least metric computations and the best search performance when computation of the metric dominates. Both these conditions yield a taller tree. This conclusion suggest that an optimal indexing tree should be as tall as possible. A set of experiments was designed to demonstrate the effects of tree height on the number of metric computations. In these experiments, binary trees are used, and no threshold on the leaf size is enforced. Instead, a tree height is predefined. A tree will grow to the predefined height, no matter how small the leaf nodes are. The results of the experiments are shown in Figure 5.7, 5.8, and 5.9. The experimental results show that in all the experiments search performance improves when the tree's height increases. Actually, in both experiments the sizes of most nodes are already reduced to 1 before level 16. In other words, most nodes are not splitted below level

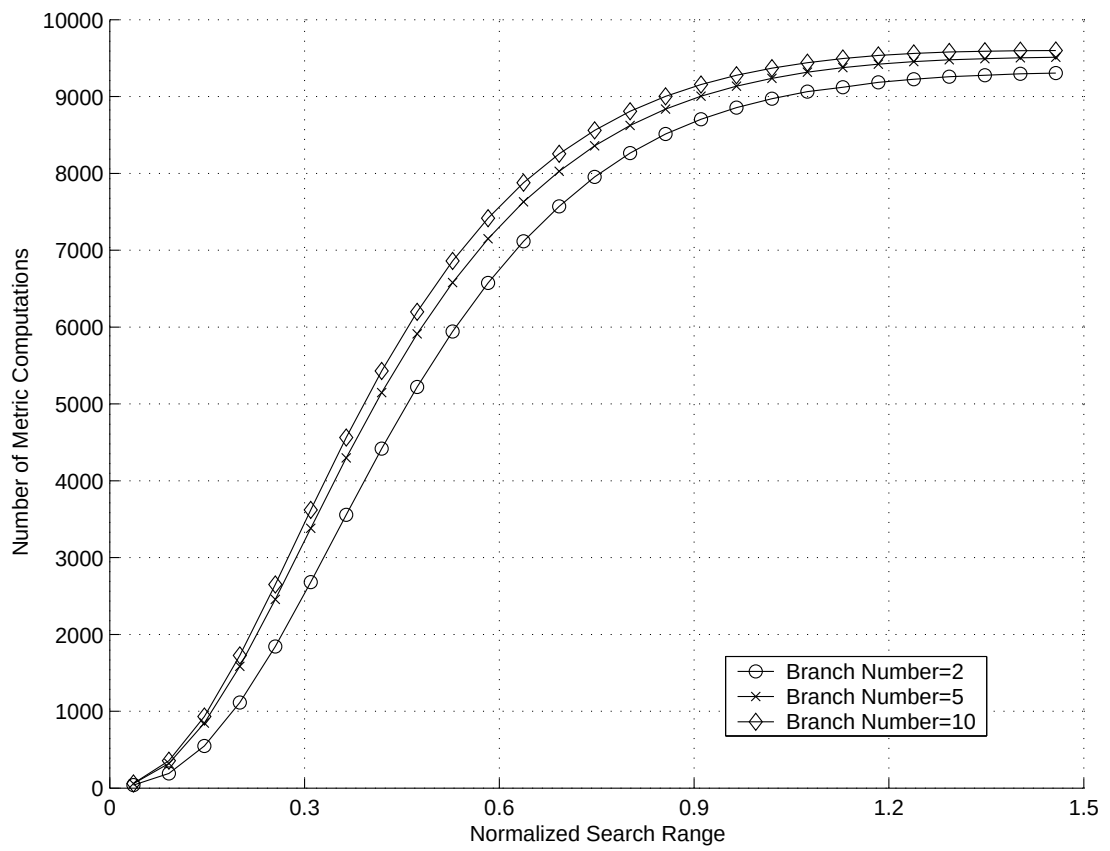


Figure 5.4: Number of Metric Computations under Different Number of Branches (Color Histograms of Images)

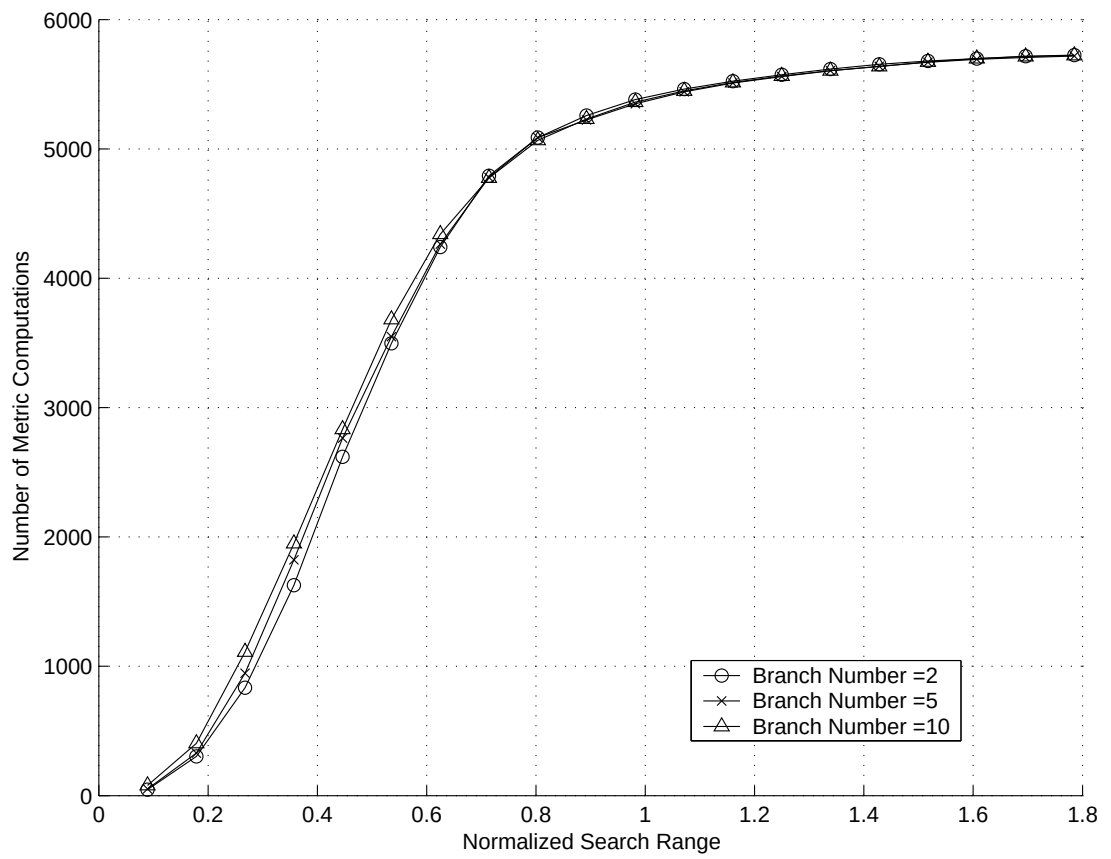


Figure 5.5: Number of Metric Computations under Different Number of Branches (Mitochondrial DNA Sequence)

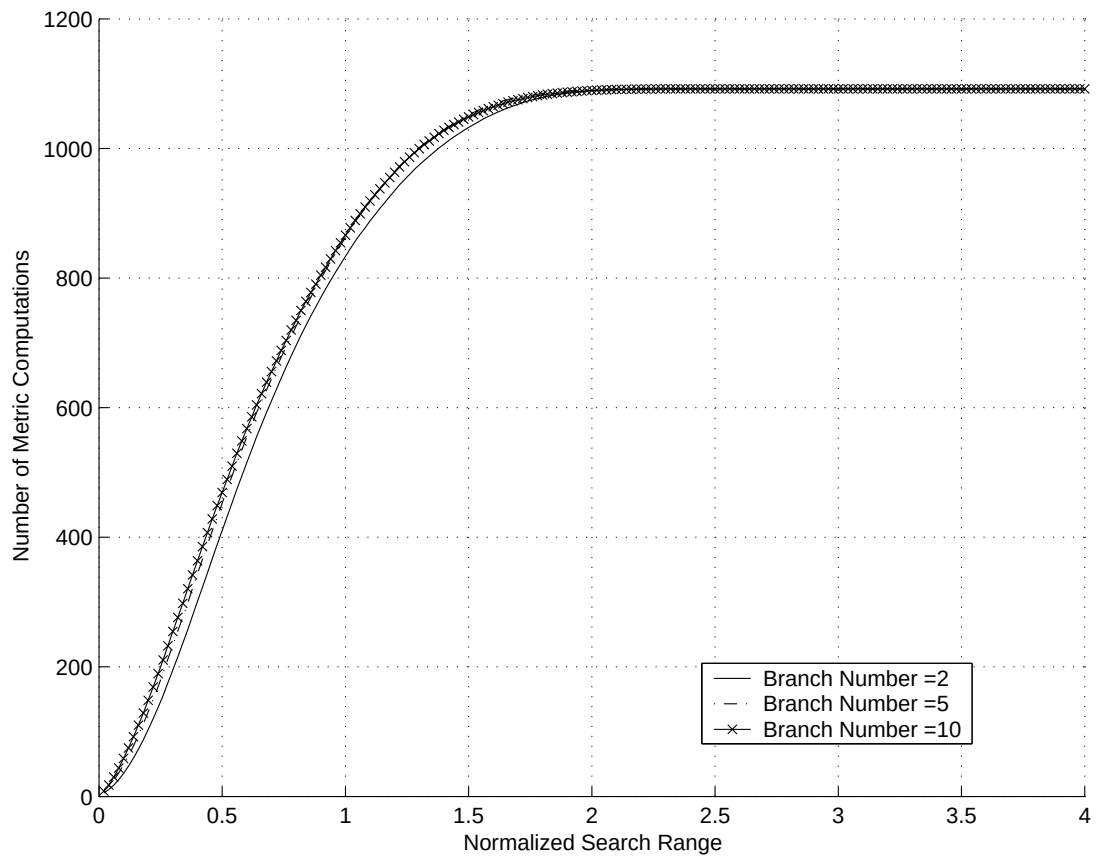


Figure 5.6: Number of Metric Computations under Different Number of Branches (Rowing Photos)

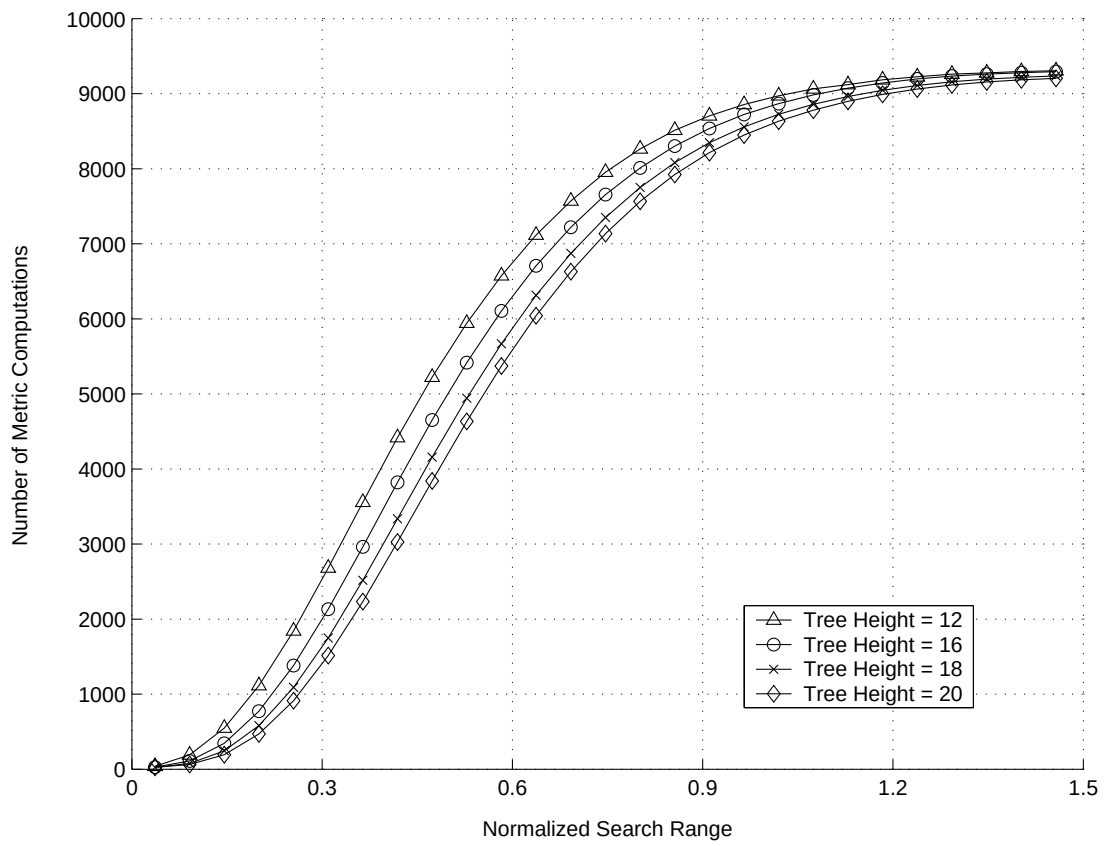


Figure 5.7: Number of Metric Computations under Different Tree Height (Color Histograms of Images)

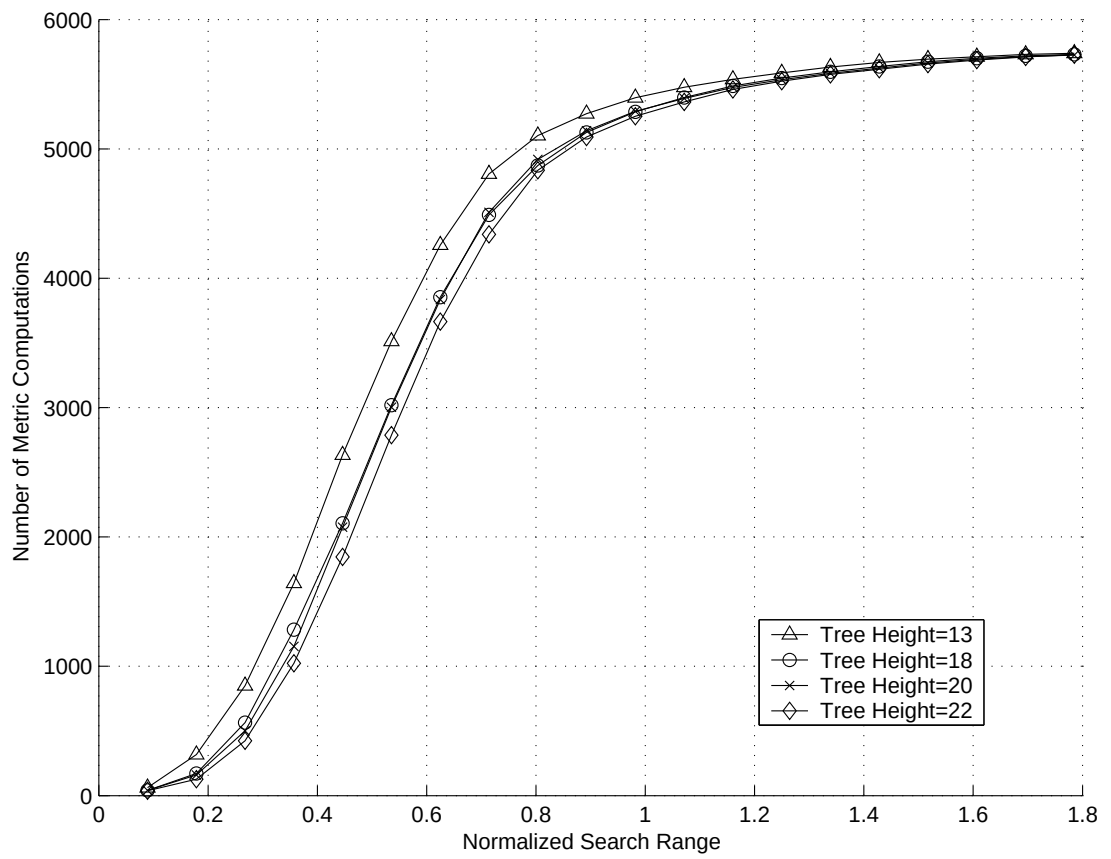


Figure 5.8: Number of Metric Computations under Different Tree Height (Mitochondrial DNA Sequence)

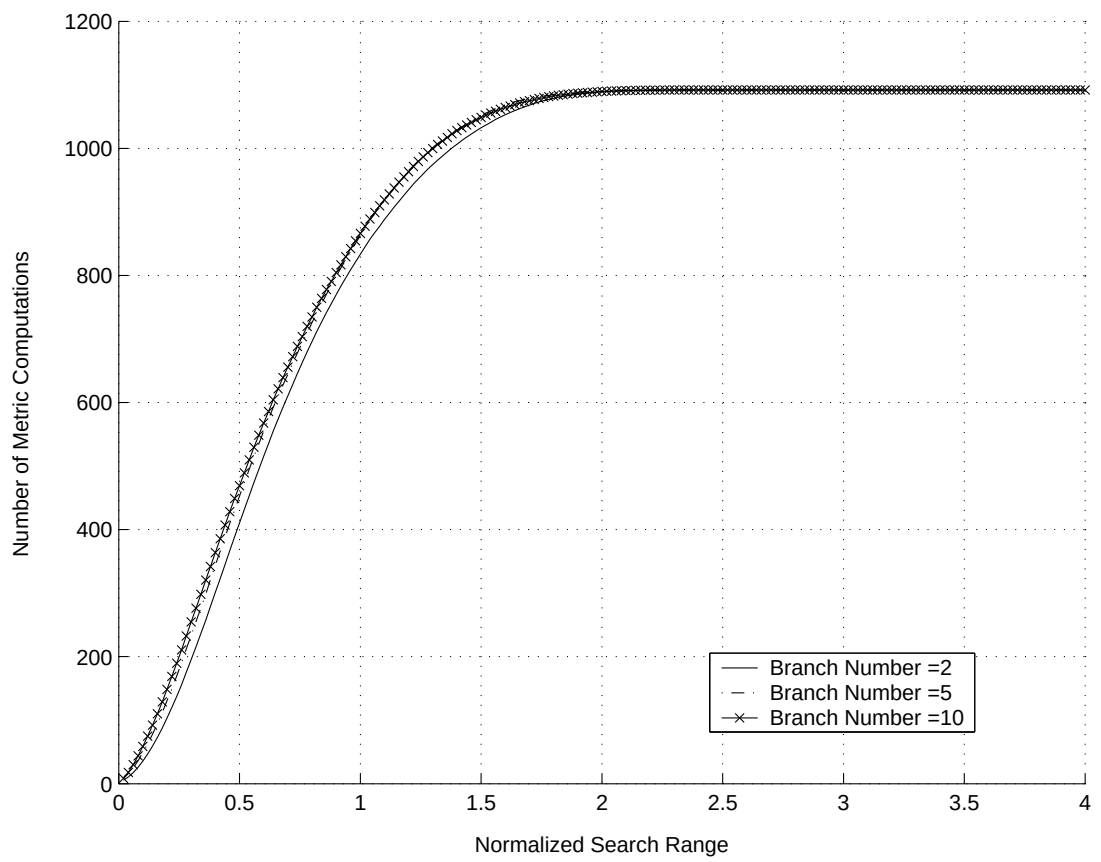


Figure 5.9: Number of Metric Computations under Different Tree Height (Rowing Photo)

16. Usually a tree will stop growing when a node can not be split, because that will cause a taller tree and longer search time. However, for trees based on metric spaces, the dependence of tree traversal time on the tree height is no longer a critical part of the search time. The search time is determined by the number of metric computations, which is

$$N_{metric} = N_{cand} + N_{ref}. \quad (5.1)$$

In some case, when the search range is so small that N_{cand} is close to N_{ref} , N_{ref} can not be neglected. An optimal tree structure should find a balance between N_{cand} and N_{ref} . However the candidate set is often much larger than the number of reference points. In this case, N_{ref} can be neglected, and N_{metric} is dominated by N_{cand} . A taller tree uses more reference points, which will lead to a larger total pruning rate and a smaller candidate set. Even if the size of a node is already reduced to 1, additional search constraints imposed by another reference point will increase the chance that the node is pruned from the candidate set. That is the reason a taller tree will yield a better search performance. Note that this conclusion may not be true, if all the nodes at each level did not use the same reference point. In that case, each additional level would increase the number of metric computations required to descend the tree by a factor in proportion to the fan out from the nodes. The main disadvantage of using more reference points is the time and memory required to construct the tree will increase a lot.

The conclusion drawn above leads to an efficient tree-structured index that is a binary tree with fixed height. The tree height is predefined, and the same number of reference points is always chosen. Then, the tree will grow until all the reference points are used, even if the size of a node is one. The key is to find an optimal tree height that

balances the search performance and tree building time.

5.5 Summary

In this chapter, two parameters of the tree-structured index are discussed. During the discussion, the tree height is found to be the key to good search performance. The conclusion that an optimal tree should be as tall as possible is drawn. An efficient tree-structured index based on this conclusion is recommended.

Chapter 6

Conclusions

6.1 Overview

Database similarity search is to find the objects that are similar to a target using a specified similarity criterion. It is different from traditional exact searches. Usually, similarity search focuses on generic data and is based on contents. The traditional indexing and search techniques are not suitable for generic data similarity search. Database similarity search is used in many practical applications, such as finding partial match of genetic sequences, scenes similar to a given image, similar transients in time series, and similar land utilization patterns in geological database. Approaches based upon metric spaces are proposed in this research. Reference points are used to build a tree-structured index, and search requests are processed based on the triangle inequality. The purpose of this research is to explore factors that limit performance, as well as opportunities for new design methods. An estimation method of the performance limitation is provided. Some new search methods are proposed and inspected. The experimental results indicate that the methods have potential, but their utility depends upon both the requirements of

the application and the characteristics of the data collection.

6.2 Current Results

In this report, the problem of generic database similarity search is formulated. Tree-structured indexing techniques based on metric spaces are a recommended solution. There are two approaches of indexing a metric space. One is based on Voronoi partitions, and another is based on reference points. The approach based upon reference points is chosen as the theme of this thesis. Tree-structured indexing techniques and search methods of this approach are addressed in detail. The number of metric computations is used as a performance evaluation criterion.

Performance limitations of tree-structured indexing techniques based upon reference points in a metric space are discussed. After in-depth analysis, the performance limitations are found to be imposed by the inherent statistical characteristics of a data set. The distance variance of the data set determines the maximum performance and range threshold. Range threshold is a new concept defined in this report. It is the maximum feasible search range, beyond which the search performance will be unacceptable. A method to estimate the range threshold using the distance variance of the data set is provided. This estimation method makes it possible to check the feasibility of similarity search approaches based upon reference points for a data set without implementing and running a program. Users can compare the estimated value of range threshold to determine whether the search range of interest are likely to yield acceptable performance. If the range threshold is smaller than the search range of interest, a satisfactory performance is unlikely to be obtained using this approach. During the research, a sets of new criteria is designed to compare the performance of search methods using different

data set. One of them is a figure illustrating two ratios $\frac{N_{cand}}{N_{total}}$ and $\frac{N_{res}}{N_{total}}$ that can be used to evaluate the search performance without the effects of different characteristics of data sets. A normalized search range is also introduced to provide a standard range for evaluating performance across different data.

Metric selection and data extraction are the very first two steps of generic database similarity search. However, normal performance evaluation criteria do not work for different metrics and data extraction methods, because they will yield different result sets. A new method using $\frac{N_{res}}{N_{cand}}$ is proposed. Moreover, an improved evaluation criterion using the integration of $\frac{N_{res}}{N_{cand}}$ weighted with an exponentially distributed search range probability is designed. Two examples using this method to compare the performance of different Minkowski metrics and data extraction methods using PCA are presented. Both experiments show this performance evaluation method works. The search utility is another problem that users care about. The utilities of different metrics and data extraction methods are evaluated manually. Some sample results are presented to show the search utilities of different metrics, and data extraction methods. The effects of search ranges on search utility are also demonstrated.

Several issues of the index tree structure, such as leaf size and the number of branches, are analyzed. The performance of trees with different leaf sizes are compared. The results suggest that the leaf size should be as small as possible. A set of experiments are conducted to demonstrate how the number of branches affects the search performance. The binary tree is proved to be the best choice. All these conclusions suggest a hypothesis that a taller tree yields better performance. Therefore, a new tree structure with fixed height is designed. The experimental results of trees with different heights support the hypothesis. This conclusion is reasonable, because a

taller tree uses more reference points, which leads to a larger pruning rate and smaller candidate set. The candidate set size is the most critical parameter that affects search performance. The disadvantage of a taller tree is longer tree building time. Therefore, a balance between the building time complexity and search performance is needed.

6.3 Future Work

Possible topics for future work include designing a method to evaluate utilities of search results and integrating it into the current method to develop a more comprehensive way to compare the performance of different data extraction methods. Besides data extraction methods, this criterion can be extended to evaluate the performance of feature extraction methods. Another possible subject is how search ranges affect the search utility. The problem of performance limitations can also be explored further. The relation between performance limitations and mean of the pairwise distances of a data set is a good topic for future work.

Bibliography

Bibliography

- [1] Y-L Cheung A. W-C Fu, P. M-s Chan and Y. S. Moon. Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances. *VLDB Journal*, 9(2):154–173, 2000.
- [2] F. Aurenhammer. Voronoi diagramsa survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.
- [3] G. Navarro B. Bustos and E. Ch’avez. Pivot selection techniques for proximity searching in metric spaces, 2001.
- [4] R. Baeza-Yates and G. Navarro. Fast approximate string matching in a dictionary. In *Proceedings of the 5th International Symposium on String Processing and Information Retrieval (SPIRE’98)*, pages 14–22. IEEE CS Press, 1998.
- [5] A. Berman and L. G. Shapiro. Selecting good keys for triangle-inequality-based pruning algorithms. In *Proc. Intrl. Workshop on Content-Based Access of Image and Video Databases (CAIVD ’98)*, pages 12–19, Bombay, India, 1998.
- [6] T. Bozkaya. Index structures for temporal and multimedia databases, 1998.
- [7] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. pages 357–368, 1997.

- [8] S. Brin. Near neighbor search in large metric spaces. In *The VLDB Journal*, pages 574–584, 1995.
- [9] W. A. Burkhard and R. M. Keller. Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236, 1973.
- [10] E. Chávez and G. Navarro. Unbalancing: the key to index high dimensional metric spaces.
- [11] TC Chiueh. Content-based image indexing. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 582–593. Morgan Kaufmann Publishers Inc., 1994.
- [12] Y. Burago D. Burago and S. Ivanov. *A Course in Metric Geometry*. Amer Mathematical Society, 2001.
- [13] F. Dehne and H. Noltemeier. Voronoi trees and clustering problems. *Inform. Systems*, 4(2), 2002.
- [14] M.H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2003. DUN m 03:1 1.Ex.
- [15] R. A. Baeza-Yates E. Chavez, G. Navarro and J. L. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.
- [16] C. Faloutsos E. Siegel F. Korn, N. Sidiropoulos and Z. Protopapas. Fast nearest neighbor search in medical image databases. In *The VLDB Journal*, pages 215–226, 1996.
- [17] F. Rabitti G. Amato and P. Savino. Supporting image search on the web, 1998.

- [18] D. Gusfield. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge Univ. Press, 1997. Gusfield.
- [19] V. Ferrari T. Tuytelaars H. Shao, T. Svoboda and L. Van Gool. Fast indexing for image retrieval based on local appearance with re-ranking. In *International Conference on Image Processing*, Sept, 2003.
- [20] P. Ciaccia I. Bartolin and M. Patella. String matching with metric trees using an approximate distance. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE 2002)*, pages 271–283, Lisbon, Portugal, September 2002. Springer.
- [21] M. R. Wilson J. A. DiZinno K. L. Monson, K. W. P. Miller and B. Budowle. The mtdna population database: An integrated software and database resource for forensic comparison. *Forensic Science Communications*, 12(2):171–175, 1987.
- [22] C. Faloutsos J. Richardson M. Arya, W. F. Cod and A. Toya. QBISM: A prototype 3-d medical image database system. *Data Engineering Bulletin*, 16(1):38–42, 1993.
- [23] H. Blanken P. Apers and M. Houtsma. Multimedia databases in perspective. 1997.
- [24] M. Patella P. Ciaccia and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *The VLDB Journal*, pages 426–435, 1997.
- [25] U. Manber R. A. Baeza-Yates, W. Cunto and S. Wu. Proximity matching using fixed-queries trees. In M. Crochemore and D. Gusfield, editors, *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, number 807, pages 198–212, Asilomar, CA, 1994. Springer-Verlag, Berlin.

- [26] C. Traina Jr. R. F. Santos Filho, A. J. M. Traina and Christos Faloutsos. Similarity search without tears: The OMNI family of all-purpose access methods. In *ICDE*, pages 623–630, 2001.
- [27] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. pages 13–25, 1997.
- [28] W. Miller E. W. Myers S. F. Altschul, W. Gish and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 1990.
- [29] J. Macker S.C. Sahinalp, M. Tasan and Z.M. Ozsoyoglu. Distance based indexing for sequence proximity search. In *ICDE’03*, Bangalore, India, 2003.
- [30] M. B. Shapiro. Choice of reference points in best-match file searching. *j-CACM*, 20(5):339–343, May 1977.
- [31] G. Strang. *Linear Algebra and its Application*, 2nd. Ed. Academic Press, New York, San Francisco, London, 1980.
- [32] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179, 1991.
- [33] J. Padolina W. Liu C. R. Linder W. Xu, W. J. Briggs and D.P. Miranker. Using mobios’ scalable genome joins to find conserved primer pair candidates between two genomes, 2004.
- [34] P.M. Narendra W.L.G. Koontz and K. Fukunaga. A branch and bound clustering algorithm. *IEEE Trans. Comput.*, C(24):908–915, 1975.
- [35] N. Yazdani. Similarity search for sequences of different lengths: Matching and indexing.

- [36] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1993.
- [37] P. Yianilos. Excluded middle vantage point forests for nearest neighbor search, 1999.
- [38] A. Yoshitaka and T. Ichikawa. A survey on content-based retrieval for multimedia databases. *IEEE Trans. on Knowledge and Data Engineering*, 11(1):81–93, 1999.

Vita

Zeqian Shen was born in HangZhou, China on March 13, 1979. He went to Zhejiang University and received a B.S. in Computer Science in 2001. He was admitted by the Electrical and Computer Engineering Department at University of Tennessee, Knoxville in 2002. He joined Laboratory of Information Technologies as a graduate research assistant and did research in data mining, machine learning and parallel computing. This thesis marks the end of his master's study at the University of Tennessee, Knoxville.