

**MS- α - AN EFFICIENT, BALANCED AND
SCALABLE APPROACH TO SPHERICAL
RANGE SEARCH IN HIGH-DIMENSIONAL
MULTIMEDIA DATABASES**

By
XIAOYONG DUAN
Bachelor of Science
Fudan University
Shanghai, China
1988

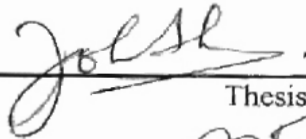
Master of Science
Fudan University
Shanghai, China
1991

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
August, 2003

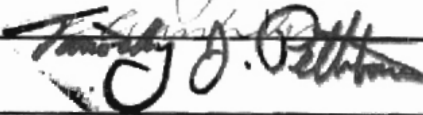
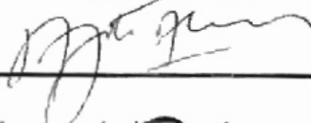
Oklahoma State University Library

**MS- α - AN EFFICIENT, BALANCED AND
SCALABLE APPROACH TO SPHERICAL
RANGE SEARCH IN HIGH-DIMENSIONAL
MULTIMEDIA DATABASES**

Thesis Approval:



Thesis Advisor



Dean of the Graduate College

PREFACE

- Subject:** High-dimensional multimedia database search
- Purpose:** Proposing MS- α as a solution for the *curse of dimensionality* problem in high-dimensional multimedia database search.
- Scope:** Quantitative analysis of the *curse of dimensionality* problem.
Mathematical proof for MS- α .
Analysis of traditional and recently proposed methods.
Experiments for comparative study and performance analysis.

ACKNOWLEDGEMENTS

I would like to express my appreciation to the many people who helped me with this thesis.

First I would like to thank **Dr. Johnson P. Thomas** for helping me organizing the thesis committee and precious advise on various issues, also for the encouragement and guidance I received from him through this whole process.

My deep gratefulness also goes to **Dr. Khanh Vu** who established the mathematical foundation of the MS- α approach and gave me invaluable guidance in technical details such as literature understanding, scope of study and experimental methodology.

I thank Dr. **Adjith P. Abraham**, as a committee member he contributed his precious time in chairing the Oral Qualifying and Final Examination meetings.

Special thanks to **Dr. Roger Weber** at Swiss Federal Institute of Technology for providing the VA-File source code and for his prompt response to my questions.

I should also thank my wife and my daughter for their whole-hearted support of my thesis work, allowing me to devote more time to my study.

TABLE OF CONTENTS

1-	Introduction.....	1
1.1	Spherical Range Search.....	1
1.2	The Curse of Dimensionality.....	2
1.3	MS- α in a Nutshell.....	6
1.4	Outline.....	8
2-	Literature Review.....	9
2.1	The VA-File.....	9
2.2	The LPC-File.....	11
2.3	iDistance.....	14
3-	MS- α in Detail.....	18
3.1	Definitions and Notations.....	18
3.2	Iso- μ Hyper-planes.....	19
3.3	Iso- σ Hyper-cylinders.....	21
3.4	α -Constant Planes.....	22
3.5	Approximation using Hyper-surfaces.....	24
3.5.1	Bound of the Means.....	25
3.5.2	Bound of the Standard Deviations.....	25
3.5.3	Bound of the α -Constants.....	26
3.5.4	The MS- α Bounding Shape.....	28

3.6	Subspace Bounds and MS- α Configurations.....	29
3.7	Comparison with Existing Approaches.....	31
4-	Objectives.....	33
5-	Technical Methodology.....	34
5.1	Data Preparation.....	34
5.2	Algorithm Implementation.....	35
5.3	Experiment Methodology.....	35
6-	Comparative Study and Performance Analysis.....	37
6.1	MS- α Optimal Configuration.....	37
6.2	Analysis of the Filtering Stage.....	38
6.3	Performance Study at 256 Dimensions.....	40
7-	Conclusions.....	44
	Reference.....	45

LIST OF FIGURES

1. $E[nn^{dist}]$ as a function of dimensionality	4
2. Search area when $2\varepsilon > edim$	5
3. Axial bounds vs. diagonal bounds.....	7
4. Data points and their approximation bit-string in a VA-File.....	10
5. Local Polar Coordinates in 2 and 3 dimensional data spaces.....	12
6. Illustration of d_{min} and d_{max} in 3 dimensional data space.....	13
7. Illustration of d_{min} and d_{max} in 2 dimensional data space for LPC- and VA-Files.....	14
8. iDistance search area in 2 dimensional data space.....	15
9. iDistance data partitioning scheme in 2 dimensional data space, and the search area (the lined area) for query point Q	16
10. Iso- μ hyper-planes, Iso- σ hyper-cylinders and α -Constant Planes in 3 dimensional data space.....	20
11. Hyper-plane $\mathcal{P}(\mathcal{N}, Q)$ in figure 10.....	24
12. The Iso- μ hyper-plane which passes through query point Q	27
13. Shape of the MS- α approximation.....	29
14. Subspace bounding on a 2 dimensional projection.....	31
15. MS- α configurations at 256 dimensions.....	37

16. MS- α optimal configuration partitions (k) at various dimensions.....	38
17. Filtering Stage Response Time at various dimensions.....	39
18: Approximation file size at 256 dimensions on 15,766 byte-sized data points.....	40
19. Filtering Stage % Return at 256 dimensions and various search radii.....	41
20. Filtering Stage Response Time at 256 dimensions and various search radii.....	42
21. Total Response Time at 256 dimensions and various search radii (Base data: 256 byte color histograms).....	43

CHAPTER 1

Introduction

1.1 Spherical Range Search

In today's multimedia applications, efficient similarity search is in high demand due to ever-increasing availability of digital multimedia data including image, video, voice, scientific data, time-series data, etc ([2], [3], [4], [5], [6], [7], [8], [9]). Objects in these applications are abstracted as a collection of features represented by numerical coordinates of a data point in a multidimensional data space, an important operation of these applications is to find similar or close objects to a given object (point), this operation is often called similarity search.

Similarity is measured by distance between objects, different metrics (L_1, L_2, L_3 , etc.) can be used to measure the distances, among which the L_2 (Euclidean) metric is the most popular one. We choose to use the Euclidean metric through out this thesis. In an n dimensional data space D_n , given points P and Q , the Euclidean distance $D(P, Q)$ is defined as,

$$D(P, Q) = \left[\sum_{i=1}^n (p_i - q_i)^2 \right]^{\frac{1}{2}}$$

Similarity searches can be characterized as the task of finding neighboring points around point Q (called query point), k nearest neighbor (k -NN) and distance based spherical range searches are two most important paradigms. As the names imply, k -NN search is to

find $k(s)$ points with smallest $D(P, Q)$, spherical range search is to find points within the hyper-sphere defined by the centroid Q and search radius ϵ , mathematically,

$$D(P, Q) = \left[\sum_{i=1}^n (p_i - q_i)^2 \right]^{\frac{1}{2}} \leq \epsilon \quad (1)$$

k -NN search can be accomplished by iterative spherical range searches starting from a small search radius ϵ , and gradually increasing the search radius until $k(s)$ nearest points are found ([11]), and reasonable estimation on the starting ϵ can be calculated based on dimension and the number of points in the data space ([1]). Thus, performance of spherical range search can be used as a benchmark for comparison of various similarity search approaches.

1.2 The Curse of Dimensionality

In large multimedia databases, a straightforward simple scan is expensive, since its time complexity is linear in terms of the size of the database. In order to achieve better performance than a simple scan, spherical range searches are typically performed in two stages, the first aims to quickly prune out the search area and returns a relatively small candidate set, and the second rejects anything beyond the exact search radius. Research has been focused on the first stage, since its effectiveness will dictate how much the load is reduced in the second stage. Various approaches have been proposed and they are all affected by dimensionality.

The number of features (dimensions) in multimedia objects ranges between moderate (4-8 in [5], 45 in [6], large (315 in [4]) and extremely large (over 900 in [3]), thus high dimensional similarity search becomes a necessity in many applications. But as

dimensionality increases, the *curse of dimensionality* (a phenomenon that the performance of search algorithms degrades drastically as dimensionality increases.) kicks in, we'll explain this phenomenon below. In following discussions, we assume coordinates of all dimensions are normalized between 0.0 and 1.0, this follows that the extension of each dimension *edim* is exactly 1.0.

As dimensionality increases, data space becomes more sparsely populated, sparsity can be measured in terms of expected NN-distance (reads expected nearest neighbor distance). Following Berchtold et al. [15] and Webber et al. [1], let $P[Q, r]$ be the probability that the NN-distance is at most r for query point Q , the expected NN-distance for query point Q can be expressed as,

$$E[Q, nn^{dist}] = \int_0^1 r \cdot \frac{\partial P[Q, r]}{\partial r} dr$$

it follows that the expected NN-distance for any query point Q is the average of $E[Q, nn^{dist}]$ over all possible query points Q in D_n ,

$$E[nn^{dist}] = \int_{Q \in D_n} E[Q, nn^{dist}] dQ$$

good approximations for such integrals can be obtained by the Monte-Carlo method, i.e. generating random points within the data space, summing the values of the function for this set of points, and dividing the sum by the total number of points. Experiments have been performed to estimate $E[nn^{dist}]$ and the results are shown in the following figure ([1]).

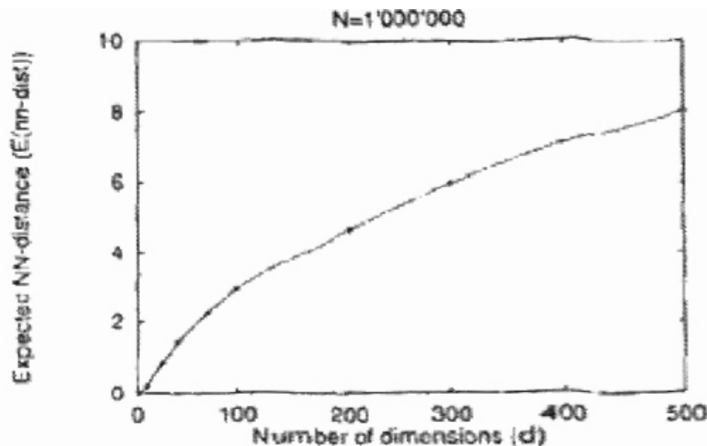


Figure 1: $E[nn^{dist}]$ as a function of dimensionality

Figure 1 shows that $E[nn^{dist}]$ increases with dimensionality, size of the database is 1,000,000 points, at 100 dimensions, $E[nn^{dist}]$ is already 2 times more than $edim$. Now we can explain why performance of various similarity search algorithms degrades with dimensionality or fails to address the dimensionality problem sufficiently.

Similarity search algorithms can be categorized as follows,

- Hyper-cube based search area approximation methods including balanced split ([13]), pyramid ([14]) and Γ partitioning ([15]).

These methods ultimately will under-perform a simple scan at sufficiently high dimensions, because when search radius ϵ exceeds $edim/2$ as shown in Figure 2, a hyper-cube with 2ϵ as the length of each dimension clearly contains all data points in the search space, the pruning first stage becomes an overhead of simple scan rather than an improvement.

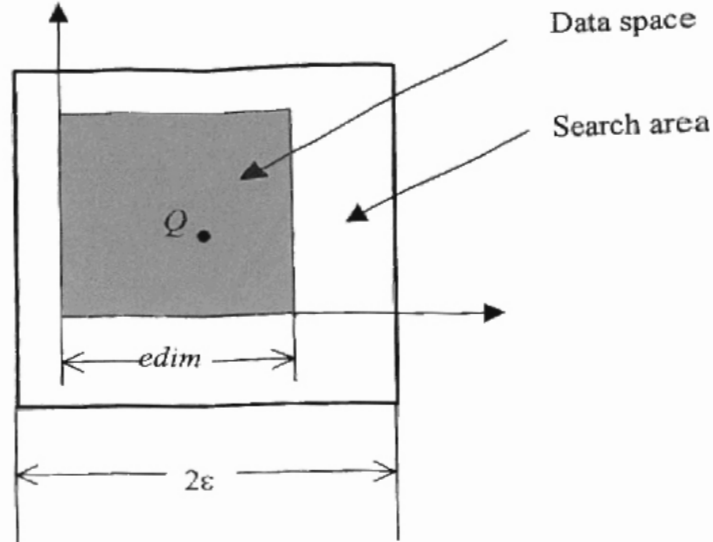


Figure 2: Search area when $2\epsilon > edim$

- Dimensionality reduction methods based on DFT ([16], [17], [7], [18]), DCT ([19]), DWT, etc.

Since Hyper-cube approximation ultimately fails at high dimensionality, a reduction in dimensionality is naturally conceived. A simple discard of dimensions will result in unbounded search area, if dimension x_i is discarded, then the search area is unbounded along the x_i dimension.

When ϵ data density is very high, unboundedness will cause more false points to be returned to the candidate set resulting in inaccurate approximation.

Transformation schemes such as DFT, DCT and DWT have been proposed to mitigate the effect of unboundedness, but these schemes do not get rid of unboundedness, instead they still discard dimensions that

deemed to be *insignificant* in the transformed data space D'_n . Observations of these methods confirmed loss of precision of these methods ([16], [10]).

- Filter based methods including VA-File ([1]), LPC-File ([2]) and iDistance ([11]) etc.

These methods aim to compress the database size by reducing the information that each point contains. Data space is divided into cells or partitions, and each data point is approximated by its geometric information within its local cell or partition, this approximation information is used to generate a compressed file based on the original database.

A sequential scan on the approximation file can then be used to find candidate points (filtering stage). Since the scan is on a compressed file, better performance over a simple scan is expected.

But these methods have to make tradeoff between compression rate and accuracy. VA-File and LPC-File have better accuracy but bigger file size (proportionate to the number of dimensions); iDistance achieved compressed file whose size is not related to dimensionality but suffers from loss of accuracy.

We shall discuss these methods in details in Chapter 2.

1.3 MS- α in a Nutshell

We observe that the hyper-cube approximation fails at high dimensionality because the cube's sides sit parallel with dimensional axes, but the extension of each dimension is at

most *edim* (or 1.0 in normalized data space), approximation for search beyond *edim* will contain all points. Various methods have been proposed, but they all have to make some tradeoff between efficiency and accuracy.

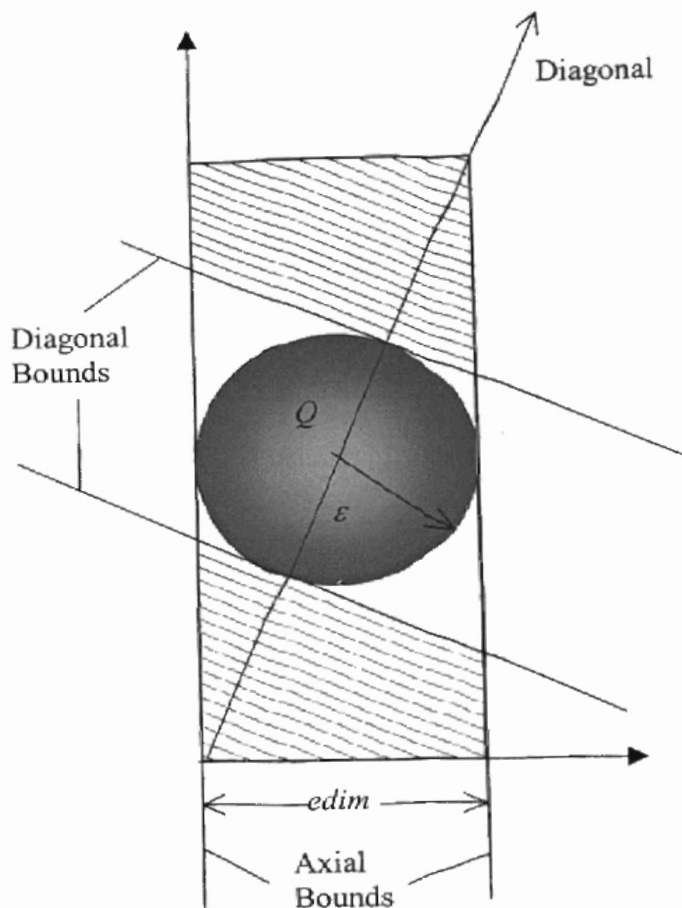


Figure 3: Axial bounds vs. diagonal bounds

We try to overcome the *edim* limitation by finding an approximation scheme that can search beyond *edim*. We observe that the length of the diagonal is greater than *edim*, in fact, the diagonal is the longest line within a normalized cubical data space (in D_n , the diagonal length is \sqrt{n}). If we can find a way to bind the search area along the diagonal,

we can search as far as the data space allows and still be able to approximate without containing all points. Figure 3 clearly shows this, the axial bounds already includes the whole data space, the diagonal bounds is still able to exclude points in the lined area. In fact, the diagonal bounds are hyper-planes generated by the means of points. We will bind the search area further by the standard deviation and the α -Constant, we shall explain the MS- α approximation in detail in Chapter 3.

1.4 Outline

We have defined spherical range search and identified the problems with existing approaches and briefly introduced our approach – the MS- α approximation. In Chapter 2 we will analyze 3 recently proposed approaches (VA-File, LPC-File and iDistance), in Chapter 3 a detailed description of MS- α will be given, Chapter 4 lays out what we want to accomplish in this thesis work, Chapter 5 gives descriptions and ramifications of how we will accomplish our work, Chapter 6 gives our results and finally Chapter 7 shall conclude our work.

CHAPTER 2

Literature Review

In Chapter 1 we categorized existing similarity search methods into 3 groups. Since hyper-cube based approximation and transformation based dimension reduction methods have been shown to either fail or display unsatisfactory performance at high dimensionality ([1], [16], [10]), we will not elaborate on them further, instead we shall devote our efforts to analyze recently proposed VA-File, LPC-File and iDistance. In this chapter, we'll review these 3 filter based approximation methods based on published materials, and figures from these materials are borrowed here for illustration.

2.1 The VA-File

The Vector Approximation File (VA-File) ([1]) divides each axis (x_i) of the D_n data

space into 2^{b_i} sections. Let $b = \sum_{i=1}^n b_i$, D_n is divided into 2^b rectangles (cells) each of

which is represented by a unique bit-string of length b . Each data point is approximated by the bit-string of the cell in which it is located, thus the approximation file is simply an array of bit-strings. Figure 4 illustrates this scheme for 4 points in a 2-dimensional data space, in this example, $b_1 = b_2 = 2$. Obviously, more bits means more cells and more granular partition of the data space, which leads to more accurate approximation of the data points.

Suppose each coordinate of a data point is a 4B (32-bit) float, if $b_1 = 4$ to $b_1 = 8$, the VA-File file size is 1/8 to 1/4 of the original database file (here we assume that the original

file is a string of floats and ignore the header size), a scan of the VA-File file is expected to take less time than the original file.

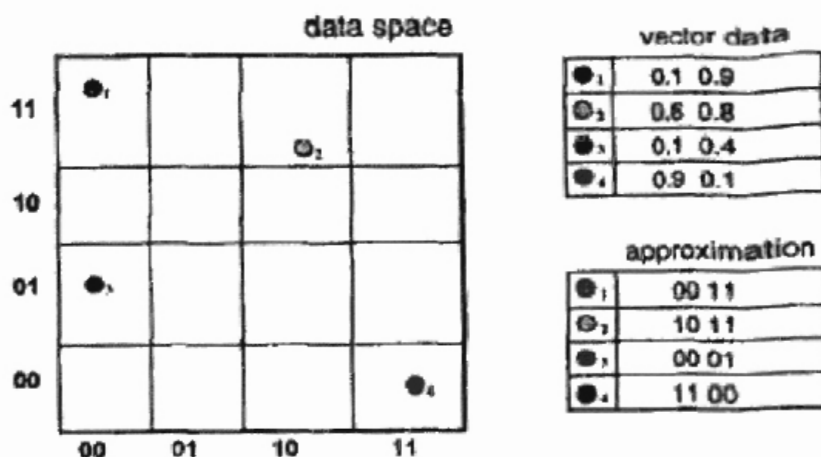


Figure 4: Data points and their approximation bit-string in a VA-File

Given a query point Q , the lower and upper bound of the distance between Q and each approximation cell d_{min} and d_{max} can be calculated based on the coordinates of Q and the cell (Figure 7 illustrates the lower and upper bound in a 2-dimensional data space). In k-NN search, both d_{min} and d_{max} are used in the filtering stage; in spherical range search, only d_{min} needs to be calculated, the filtering stage scans all approximation cells in the VA-File, if d_{min} of the cell is not greater than search radius ϵ , the data point falling within the cell is selected as a candidate point. Random accesses to the original file are then performed to find the real distance between points in the candidate set and Q . Experiments have shown that the VA-File outperforms ([1]) a simple scan in high dimensions, but its performance in the filtering stage degrades linearly with dimensionality, this is due to the fact that the VA-File size grows linearly with

dimensionality and the time complexity of d_{min} calculation is also $O(n*N)$ (N is the number of data points). Worse yet, if the original data is 1B in each dimension, as in color histograms, and 8 bits are used for each dimension, the VA-File accomplishes no compression at all.

The VA-File also has to make a tradeoff between accuracy and compression rate, if more bits are used, higher accuracy is expected, but performance of the filtering stage will degrade because of the increased approximation file size; on the other hand, if less bits are used, filtering should be more efficient, but accuracy suffers, more random access to the original file ensues.

2.2 The LPC-File

The Local Polar Coordinate File (LPC-File) ([2]) is an improvement over the VA-File in terms of approximation accuracy. Like the VA-File, it divides the data space into 2^b rectangles (cells). It uses the same bit-string as the VA-File plus local polar coordinates to approximate a data point.

Figure 5 illustrates the local polar coordinates for data point P in 2 and 3 dimensional data spaces. We can use the three tuple (b, r, θ) to represent the approximation, b is the bit-string as in VA-File, r is the distance between O and P , θ is the angle between the diagonal line of the cell and the line \overline{OP} . Since r and θ are coordinates local to O , we refer to them as local polar coordinates. In 2 dimensional space, as shown in Figure 5(a), P and P' have the same (b, r, θ) ; Figure 5(b) shows that in 3 dimensional space, points on the circle revolving around the local diagonal have the same (b, r, θ) . It is clear that

the introduction of r and θ gains more accuracy in approximation (The 'circle' as opposed to the whole cell).

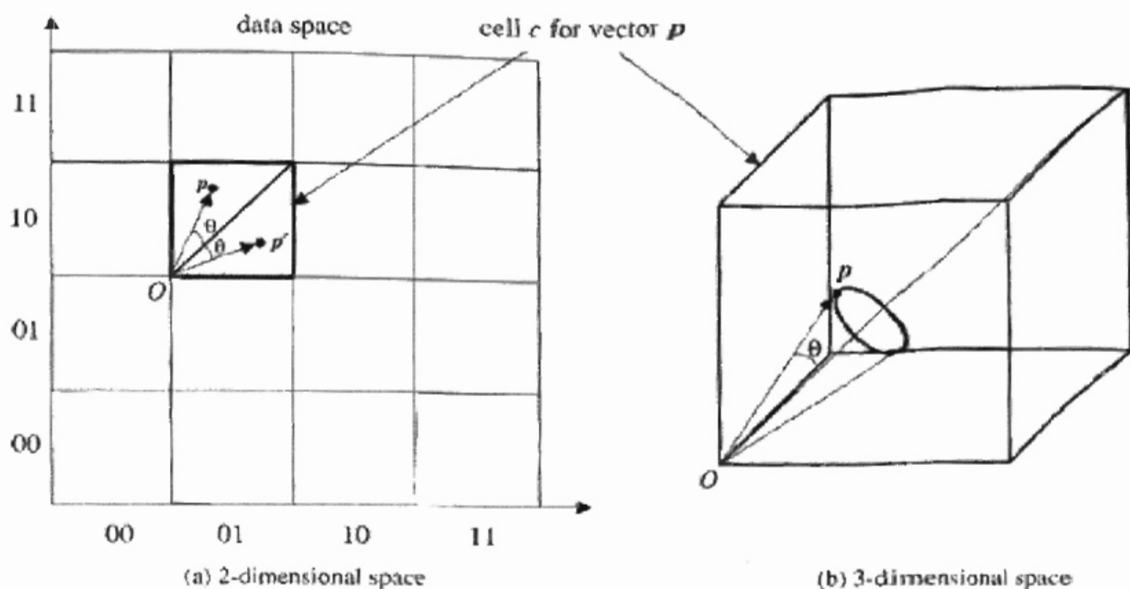


Figure 5: Local Polar Coordinates in 2 and 3 dimensional data spaces

As shown in Figure 6, if B is the query point, A and C are the points where the hyperplane formed by O, D, B intersects with the 'circle', the lower and upper bound of the distance between B and the 'circle' d_{min} and d_{max} are $D(B, C)$ and $D(B, A)$ and can be calculated as follows.

$$d_{min} = \left(D(O, A)^2 + D(O, B)^2 - 2 * D(O, A) * D(O, B) * \cos |\theta_1 - \theta_2| \right)^{\frac{1}{2}}$$

$$d_{max} = \left(D(O, A)^2 + D(O, B)^2 - 2 * D(O, A) * D(O, B) * \cos(\theta_1 + \theta_2) \right)^{\frac{1}{2}}$$

Once d_{min} and d_{max} are calculated, same algorithm on the VA-File can be used on the LPC-File in the filtering and the random access stages.

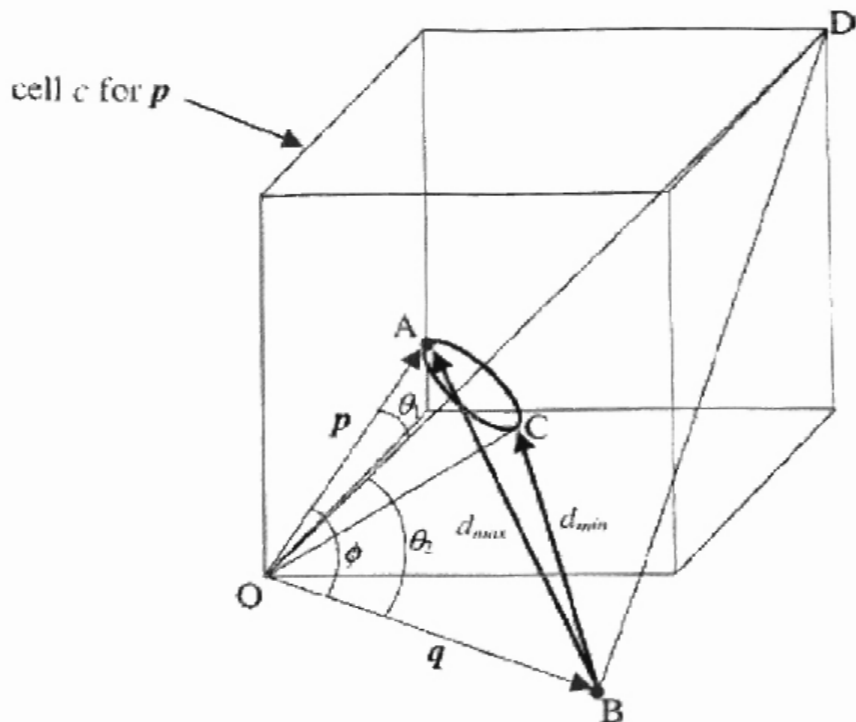


Figure 6: Illustration of d_{min} and d_{max} in 3 dimensional data space

The approximation accuracy gain achieved by the LPC-File can also be measured by the larger d_{min} and smaller d_{max} than those of the VA-File, as shown in Figure 7.

The LPC-File's approach of adding 3B (2B for r and 1B for θ) to the VA-File bit-string, though relatively small compared with the bit-string (256B for 256 dimensions using 8 bits for each dimension), is still a tradeoff between accuracy and compression rate.

Its performance degrades linearly just as the VA-File, and it gains no compression at all for 1B histograms, also calculation of d_{min} is more expensive than VA-File. As experiments show ([2]), its performance gain over the VA-File is marginal under clustered data distribution and high data density, but the penalty it carries when

calculating d_{min} may outweigh the precision gain when data distribution and density is not in its favor.

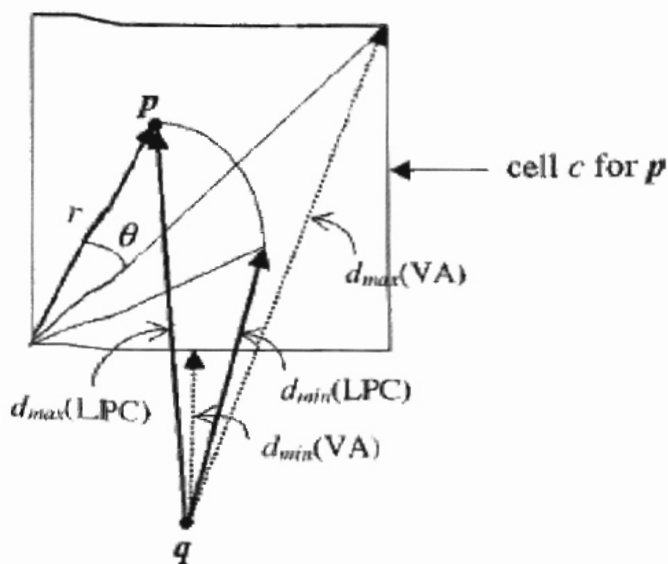


Figure 7: Illustration of d_{min} and d_{max} in 2 dimensional data space for LPC- and VA-Files

2.3 The iDistance

The iDistance ([11]) method divides the data space into certain number of partitions, and assigns a reference point for each partition. A data point P is represented as the coordinate d in a one dimensional data space as follows.

$$d = i * c + D(P, R_i)$$

Point P is located in partition i of which R_i is the reference point, and c is a constant to stretch the coordinate, so that the coordinates for all data points in partition i falls within the range of $i * c$ to $(i + 1) * c$. We can see that the partition ID i is simply the quotient of d divided by c . Let d_{max}^i be the maximum distance between R_i and all points in partition

i , obviously c must be greater than d_{\max}^i . Essentially, the one dimensional coordinate is a rough approximation of the original data.

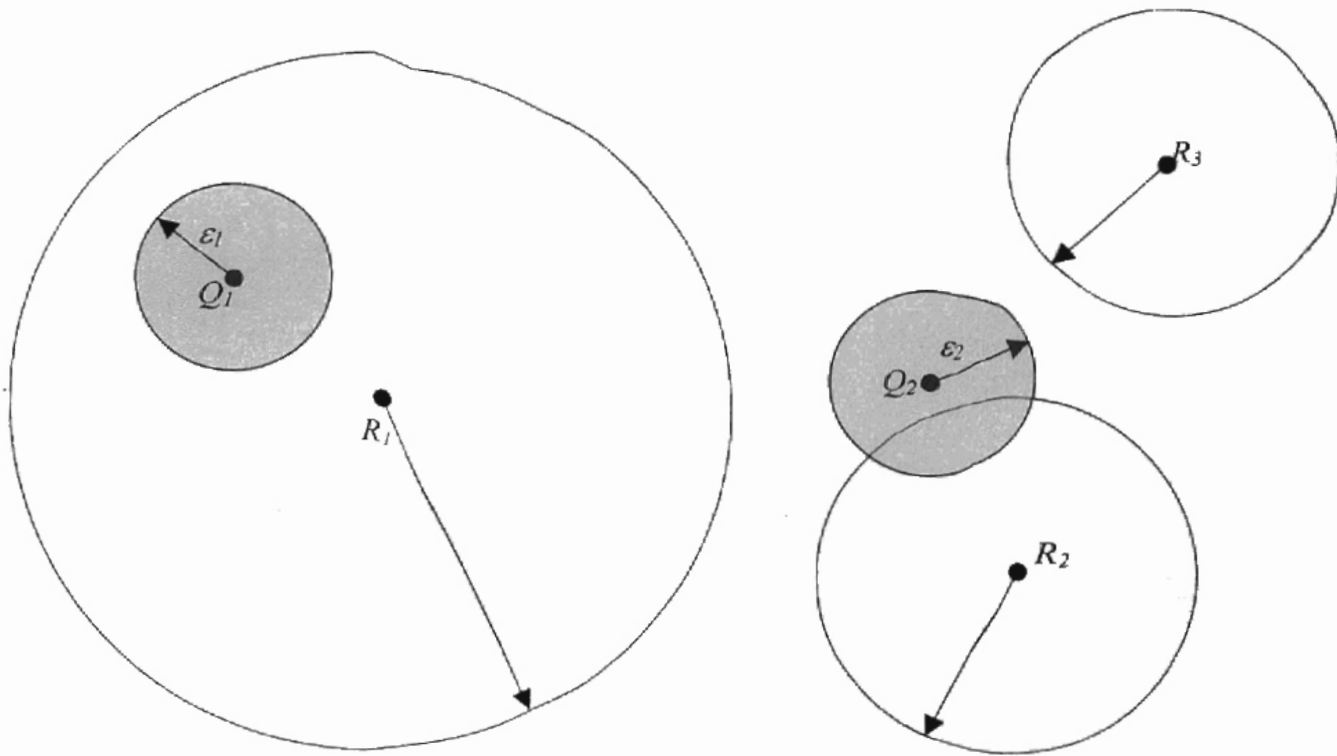


Figure 8: iDistance search area in 2 dimensional data space

The distance between the query point and the reference point determines which area of a partition needs to be searched. In Figure 8, R_1 , R_2 , and R_3 are reference points of partitions 1, 2 and 3, each partition is the space enclosed in the hyper-sphere around the reference point, Q_1 and Q_2 are two query points with search radii ϵ_1 and ϵ_2 . Given the fact that all points within some partition and having the same distance from the reference point are represented using the same one dimensional coordinate, for Q_1 the light gray area within partition 1 needs to be searched, partitions 2 and 3 need not be searched; for Q_2 only the

light gray area within partition 2 needs to be searched, partitions 1 and 3 need not be searched. It is straight forward to show that for partition i , given query point Q and search radius ϵ , a point within it needs to be searched only when its distance from the reference point falls within the range $\max(0, D(O_i, Q) - \epsilon)$ to $\min(d'_{\max}, D(O_i, Q) + \epsilon)$.

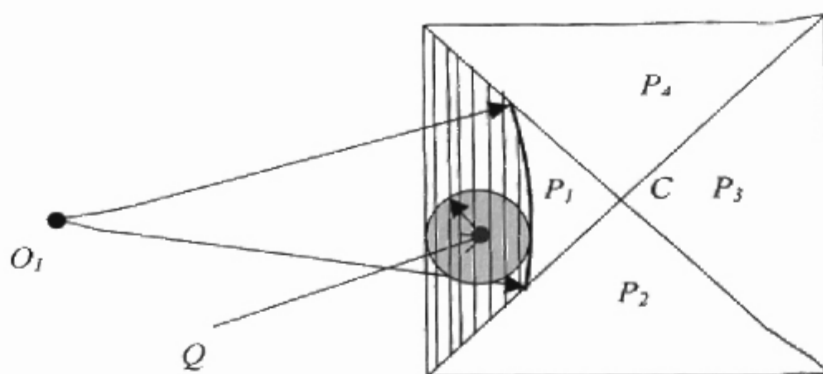


Figure 9: iDistance data partitioning scheme in 2 dimensional data space, and the search area (the lined area) for query point Q

Data space is portioned in a pyramid like scheme as shown in Figure 9, the centroid of the data space is used for the apex of all pyramids, and the border hyper-planes of the data space are used as bases, each pyramid forms a data partition, as in P_1, P_2, P_3 and P_4 . Experiments ([11]) show that the further the reference point is from the partition, the better performance.

The original iDistance algorithm uses B^+ -tree to index the one dimensional approximation data, for comparison with VA-File, we can store the approximation in a flat file and use the familiar sequential scan for the filtering stage, without loss of fairness. Clearly, iDistance has achieved much smaller compression rate on the

approximation file, suppose the one dimensional coordinate is a 4B float, in 256 float coordinates space, the approximation is only $1/256$ of the original file and stays unchanged as dimensionality increases. But iDistance suffers from poor accuracy, as confirmed by experiments ([11]). In figure 9, the search area (lined area) for query point Q is much bigger than the hyper-sphere (gray area). It is clear that when dimensionality increases, the search area grows larger; also, if the data density is very high, more points fall in the search area, both conditions lead to performance degradation. The time complexity on calculating $D(O_i, Q)$ is worth discussing, let n be the number of dimensions, N the number of data points, if we embed the calculation when evaluating each approximation, the time complexity is $O(n*N)$, if we calculate it before evaluation, the complexity is $O(n*n)$ because the number of partitions is $2*n$, we choose to implement iDistance using the latter approach, because our N is much bigger than n , we can see at high dimensionality, this portion of the calculation is very expensive.

CHAPTER 3

MS- α in Detail

This chapter will describe in detail iso- μ hyper-planes, iso- σ hyper-cylinders and α -Constant planes, and how these hyper-surfaces can be used to bind the search area.

3.1 Definitions and Notations

Table 1 describes the notations we use in this thesis.

Notations	Description
ε	Search radius
$P(p_1, p_2, \dots, p_n) \in D_n$	D_n point P with coordinates (p_1, p_2, \dots, p_n)
$D_m \subseteq D_n$	A projection of D_n onto m ($1 \leq m \leq n$) sub-dimensions
$\dim(D_m)$	Dimensions of D_m
$\mu_p, \mu_p^{(m)}$	The mean of point P in D_n and D_m respectively
$\sigma_p, \sigma_p^{(m)}$	The standard deviation of point P in D_n and D_m respectively
\mathcal{N}	The diagonal line
\mathcal{C}	A hyper-cylinder revolving around \mathcal{N}
$\mathcal{S}(Q, \varepsilon)$	A hyper-sphere in D_n formed by centroid Q and radius ε
$\mathcal{P}(\mathcal{N}, Q)$	A hyper-plane formed by \mathcal{N} and Q

Table 1: Notations

Definition 1 (Mean) The mean of point $P(p_1, p_2, \dots, p_n)$ in D_m ,

$$\mu_p^{(m)} = \frac{\sum_{i \in \text{dim}(D_m)} p_i}{m}$$

Definition 2 (Standard Deviation) The standard deviation of point $P(p_1, p_2, \dots, p_n)$ in D_m ,

$$\sigma_p^{(m)} = \left[\frac{\sum_{i \in \text{dim}(D_m)} p_i^2}{m} - (\mu_p^{(m)})^2 \right]^{\frac{1}{2}}$$

3.2 Iso- μ Hyper-planes

Theorem 1 All data points on a hyper-plane perpendicular to \mathcal{N} have the same μ .

Proof: Let's take a look at Figure 10, points M, P and K are on a hyper-plane perpendicular to \mathcal{N} and intercepts \mathcal{N} at M ,

$$\begin{aligned} D(P, M)^2 &= \left[\sum_{i=1}^n (p_i - m)^2 \right] \\ &= \left[\sum_{i=1}^n p_i^2 - 2 * m * \sum_{i=1}^n p_i + \sum_{i=1}^n m^2 \right] \\ &= \left[\sum_{i=1}^n p_i^2 - 2 * \frac{\mu_M * \mu_P}{n} + \frac{\mu_M * \mu_M}{n} \right] \\ \frac{d(D(P, M)^2)}{d(\mu_M)} &= (\mu_M - \mu_P) * 2/n \end{aligned}$$

$D(P, M)$ is minimum only when $\mu_M = \mu_P$, for the same reason, we have $\mu_M = \mu_K$, we call these planes Iso- μ hyper-planes. We also have,

$$D(O, M) = \left[\sum_{i=1}^n m_i^2 \right]^{\frac{1}{2}} = m\sqrt{n} = \mu_P \sqrt{n} \quad (2)$$

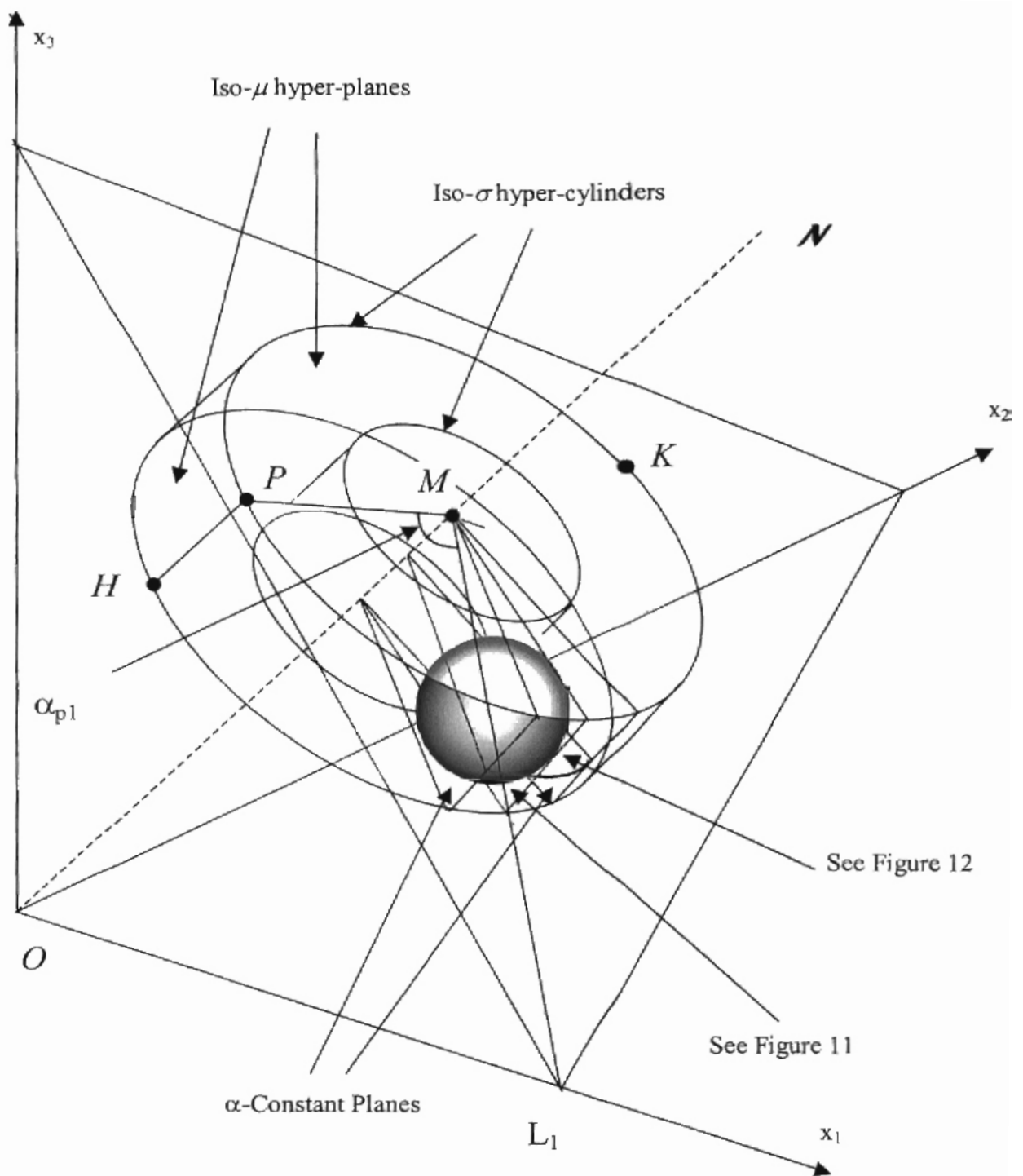


Figure 10: Iso- μ hyper-planes, Iso- σ hyper-cylinders and α -Constant Planes in 3 dimensional data space

3.3 Iso- σ Hyper-cylinders

Theorem 2 All data points on a hyper-cylinder revolving around \mathcal{N} have the same σ .

Proof: Let's take a look at Figure 10, points M, P and K are on a hyper-cylinder revolving around \mathcal{N} , H and P are on a line parallel to \mathcal{N} , P and K are on the circle formed by the intersection of hyper-sphere $\mathcal{S}(Q, D(O, P))$ and the hyper-plane $\mathcal{P}(\mathcal{N}, P)$. Since geometric relationships between all points on the hyper-cylinder can be derived from the relationships between M, P and K , we only need to prove these 3 points have the same σ .

Since \overline{HP} is parallel to \mathcal{N} , it must be true that $p_i = h_i + d$ for some constant d and

$1 \leq i \leq n$,

$$\begin{aligned}
 \sigma_P &= \left[\frac{\sum_{i=1}^n p_i^2}{n} - \left(\frac{\sum_{i=1}^n p_i}{n} \right)^2 \right]^{\frac{1}{2}} \\
 &= \left[\frac{\sum_{i=1}^n h_i^2 + \sum_{i=1}^n d^2 + 2 \sum_{i=1}^n h_i * d}{n} - \frac{(\sum_{i=1}^n h_i)^2 + (\sum_{i=1}^n d)^2 + 2 * \sum_{i=1}^n h_i * \sum_{i=1}^n d}{n^2} \right]^{\frac{1}{2}} \\
 &= \left[\frac{\sum_{i=1}^n h_i^2 + n * d^2 + 2 * d \sum_{i=1}^n h_i}{n} - \frac{(\sum_{i=1}^n h_i)^2 + n^2 * d^2 + 2 * n * \sum_{i=1}^n h_i * d}{n^2} \right]^{\frac{1}{2}} \\
 &= \left[\frac{\sum_{i=1}^n h_i^2}{n} - \frac{(\sum_{i=1}^n h_i)^2}{n^2} \right]^{\frac{1}{2}} \\
 &= \sigma_H
 \end{aligned}$$

and since P and K are on $\mathcal{S}(Q, D(O, P))$ and $\mathcal{P}(\mathcal{N}, P)$, we have $\sum_{i=1}^n p_i^2 = \sum_{i=1}^n k_i^2$ and

$$\sigma_p = \left[\frac{\sum_{i=1}^n p_i^2}{n} - \mu_p^2 \right]^{\frac{1}{2}} = \left[\frac{\sum_{i=1}^n k_i^2}{n} - \mu_K^2 \right]^{\frac{1}{2}} = \sigma_K$$

we call these cylinders Iso- σ hyper-cylinders.

We also have

$$\begin{aligned} D(M, P) &= \left[\sum_{i=1}^n (p_i - m)^2 \right]^{\frac{1}{2}} \\ &= \left[\sum_{i=1}^n p_i^2 - 2 * m * \sum_{i=1}^n p_i + \sum_{i=1}^n m^2 \right]^{\frac{1}{2}} \\ &= \left[\sum_{i=1}^n p_i^2 - 2 * \frac{(\sum_{i=1}^n p_i)^2}{n} + \frac{(\sum_{i=1}^n p_i)^2}{n} \right]^{\frac{1}{2}} \\ &= \sigma_p \sqrt{n} \end{aligned} \tag{3}$$

3.4 α -Constant Planes

Theorem 3 Let L_k be the intercept of Iso- μ hyper-plane $\mathcal{P}(\mathcal{N}, P)$ and axis $x_k, 1 \leq k \leq n$,

and α_{p_k} be the angle between lines \overline{MP} and \overline{ML}_k .

$$\cos \alpha_{p_k} = \frac{p_k - \mu_p}{\sigma_p \sqrt{n-1}} \tag{4}$$

Proof: To have some intuition, let's take a look at the angle between \overline{MP} and \overline{ML}_1 in

figure 10,

$$\cos \alpha_{p_k} = \frac{\overline{MP} * \overline{ML_k}}{D(M, P) * D(M, L_k)}$$

$$\begin{aligned} \overline{MP} * \overline{ML_k} &= -(p_1 - m)m - (p_2 - m)m - \dots + (p_k - m)\left(\sum_{i=1}^n p_i - m\right) - \dots - (p_n - m)m \\ &= -m \sum_{i=1}^n p_i + nm^2 + p_k \sum_{i=1}^n p_i - m \sum_{i=1}^n p_i \\ &= -mn\mu_p + n\mu_p^2 + p_k n\mu_p - n\mu_p^2 \\ &= n\mu_p(p_k - \mu_p) \end{aligned}$$

$$\begin{aligned} D(M, L_k) &= \left[\left(\sum_{i=1}^n p_i - m \right)^2 + (n-1)m^2 \right]^{\frac{1}{2}} \\ &= \left[\left(\sum_{i=1}^n p_i \right)^2 - 2 \left(\sum_{i=1}^n p_i \right) m + nm^2 \right]^{\frac{1}{2}} \\ &= \left[n^2 \mu_p^2 - 2n\mu_p^2 + n\mu_p^2 \right]^{\frac{1}{2}} \\ &= \mu_p \left[n(n-1) \right]^{\frac{1}{2}} \end{aligned}$$

$$\begin{aligned} \cos \alpha_{p_k} &= \frac{\overline{MP} * \overline{ML_k}}{D(M, P) * D(M, L_k)} \\ &= \frac{n\mu_p(p_k - \mu_p)}{(\sigma_p \sqrt{n})(\mu_p [n(n-1)]^{\frac{1}{2}})} \\ &= \frac{p_k - \mu_p}{\sigma_p \sqrt{n-1}} \end{aligned}$$

We call the hyper-planes with \mathcal{N} buried inside “ α -Constant planes”, the angle between these planes and any $\overline{ML_k}$ (L_k being the intercept of any Iso- μ hyper-plane and axis $x_k, 1 \leq k \leq n$, and M being the intercept point of \mathcal{N} and the Iso- μ hyper-plane) can be calculated using Equation (4).

3.5 Approximation using Hyper-surfaces

Theorems 1, 2 and 3 have defined hyper-surfaces using a point's μ , σ and α constants, this section will try to approximate the search space without false dismissals using these surfaces, clearly the tightest bounding surfaces are those passing through the tangent points of $\mathcal{S}(Q, \varepsilon)$.

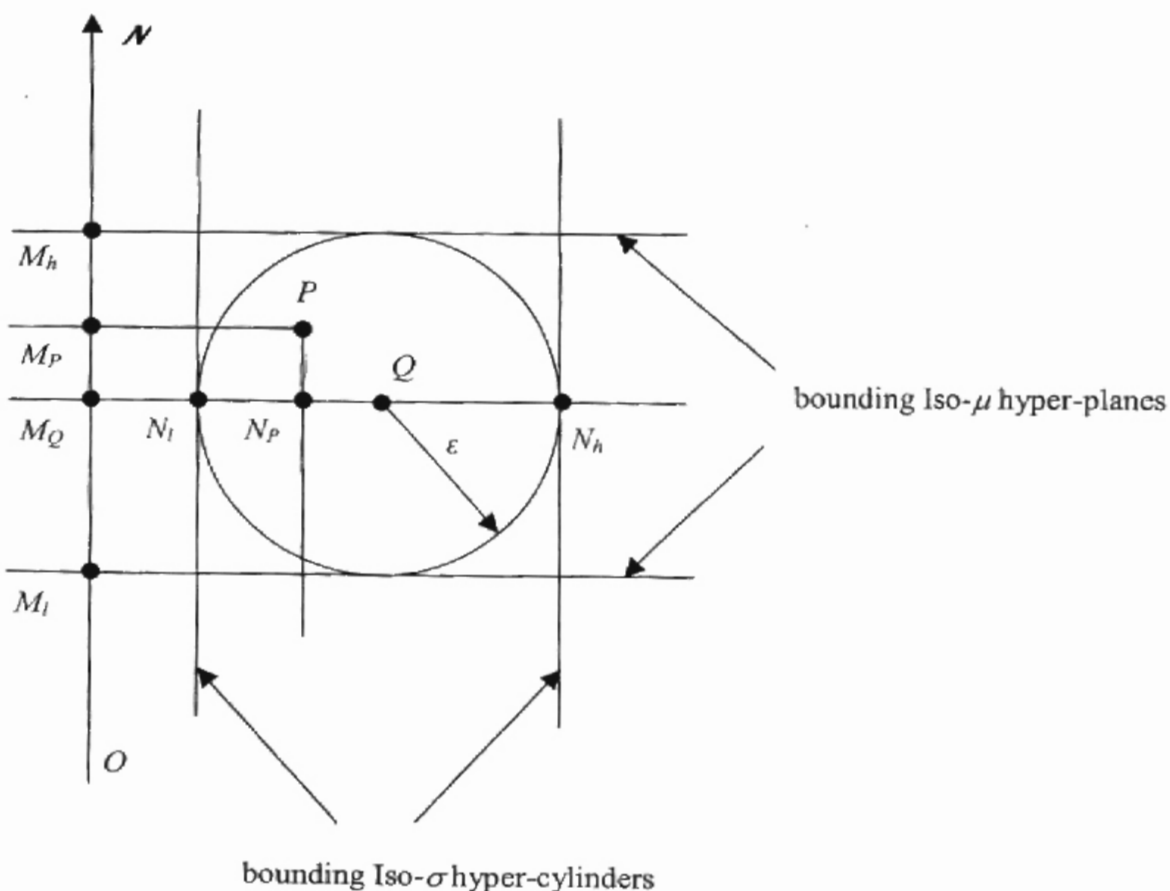


Figure 11: Hyper-plane $\mathcal{P}(\mathcal{N}, Q)$ in figure 10

3.5.1 Bound of the Means

Lemma 1 Let Q be the query point, ε the search radius and P any point in the search sphere $\mathcal{S}(Q, \varepsilon)$ i.e. P satisfying Equation 1, we have,

$$|\mu_P - \mu_Q| \leq \frac{\varepsilon}{\sqrt{n}}$$

Proof: Figure 11 is the detailed illustration of hyper-plane $\mathcal{P}(\mathcal{N}, Q)$ of Figure 10, the bounding Iso- μ hyper-planes pass through the ‘upper’ and ‘lower’ tangent points of $\mathcal{S}(Q, \varepsilon)$ and intersect with \mathcal{N} at points M_h and M_l .

$$\begin{aligned} D(O, M_l) &\leq D(O, M_P) \leq D(O, M_h) \\ D(O, M_Q) - \varepsilon &\leq D(O, M_P) \leq D(O, M_Q) + \varepsilon \\ |D(O, M_P) - D(O, M_Q)| &\leq \varepsilon \\ |\mu_P \sqrt{n} - \mu_Q \sqrt{n}| &\leq \varepsilon \\ |\mu_P - \mu_Q| &\leq \frac{\varepsilon}{\sqrt{n}} \end{aligned}$$

Clearly, bounds by these hyper-planes are not confined and will lead to large false returns. In the next section, we shall bind the search area further by the iso-hyper-cylinders.

3.5.2 Bound of the Standard Deviations

Lemma 2 Let Q be the query point, ε the search radius and P any point in the search sphere $\mathcal{S}(Q, \varepsilon)$ i.e. P satisfying Equation 1, we have,

$$|\sigma_P - \sigma_Q| \leq \frac{\varepsilon}{\sqrt{n}}$$

Proof: Figure 11 is the detailed illustration of hyper-plane $\mathcal{P}(\mathcal{N}, Q)$ of Figure 10, the bounding Iso- σ hyper-cylinders pass through the ‘right’ and ‘left’ tangent points of $\mathcal{S}(Q, \varepsilon)$ and intersect with \mathcal{N} at points N_h and N_l .

$$\begin{aligned}
 D(M_Q, N_l) &\leq D(M_Q, N_p) \leq D(M_Q, N_h) \\
 D(M_Q, Q) - \varepsilon &\leq D(M_Q, N_p) \leq D(M_Q, Q) + \varepsilon \\
 |D(M_Q, N_p) - D(M_Q, Q)| &\leq \varepsilon \\
 |\sigma_p \sqrt{n} - \sigma_Q \sqrt{n}| &\leq \varepsilon \\
 |\sigma_p - \sigma_Q| &\leq \frac{\varepsilon}{\sqrt{n}}
 \end{aligned}$$

Lemmas 1 and 2 already confine the search area in a hyper-ring (see Figure 10), but its volume is still quite large compared with the search sphere, we shall bind the search area further by the α -Constant Planes.

3.5.3 Bound of the α -Constants

Lemma 3 Let Q be the query point, ε the search radius and P any point in the search sphere $\mathcal{S}(Q, \varepsilon)$ i.e. P satisfying Equation 1, we have,

$$\left| \arccos\left(\frac{p_k - \mu_p}{\sigma_p \sqrt{n-1}}\right) - \arccos\left(\frac{q_k - \mu_Q}{\sigma_Q \sqrt{n-1}}\right) \right| \leq \arcsin\left(\frac{\varepsilon}{\sigma_Q \sqrt{n}}\right)$$

Proof: Figure 12 is the detailed illustration of the Iso- μ hyper-plane which passes through query point Q in Figure 10, the bounding α -Constant planes pass through the tangent points of $\mathcal{S}(Q, \varepsilon)$ at the smallest and biggest α angles possible.

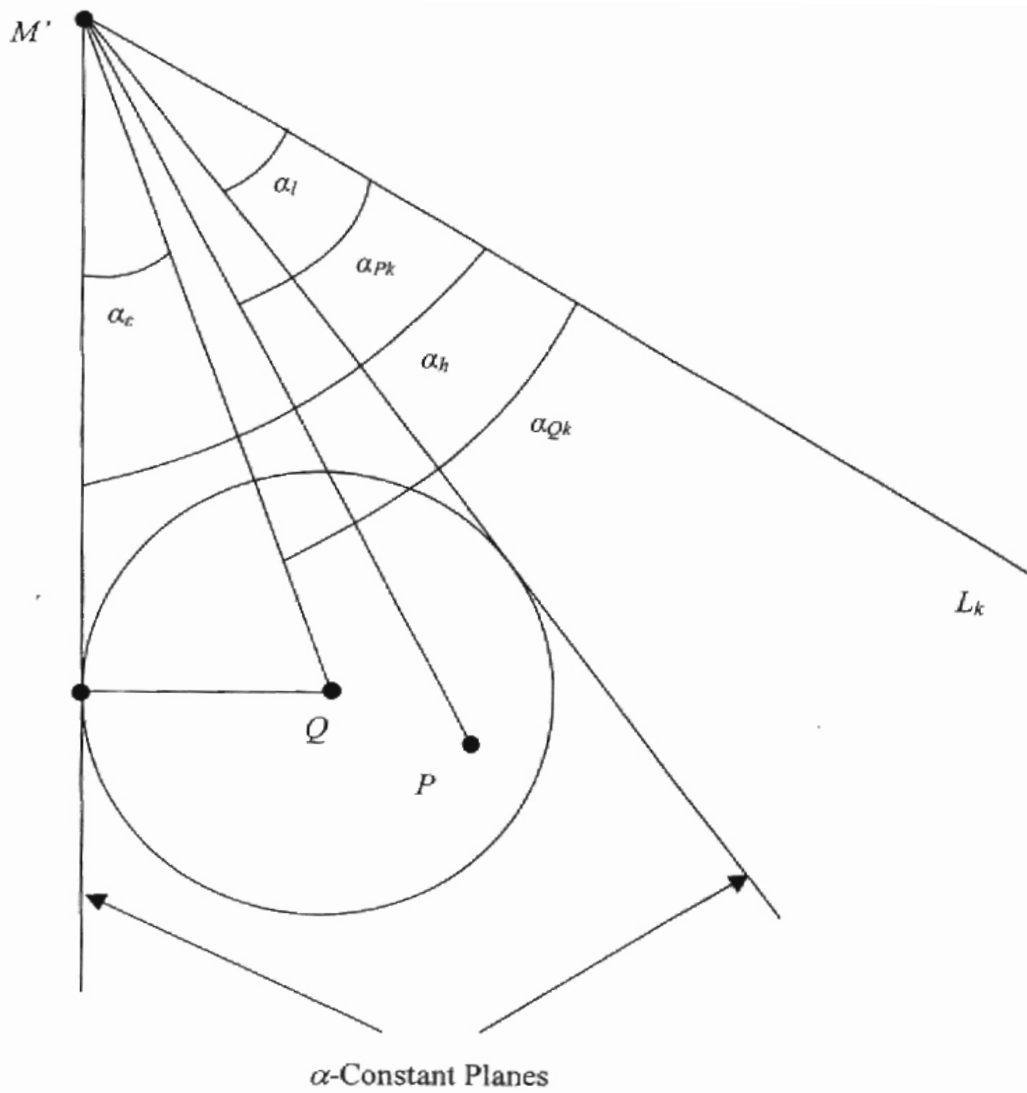


Figure 12: The Iso- μ hyper-plane which passes through query point Q

$$\sin \alpha_c = \frac{\epsilon}{D(M', Q)} = \frac{\epsilon}{\sigma_Q \sqrt{n}}$$

$$\alpha_l \leq \alpha_{P_k} \leq \alpha_l$$

$$\alpha_{Q_k} - \alpha_c \leq \alpha_{P_k} \leq \alpha_{Q_k} + \alpha_c$$

$$|\alpha_{P_k} - \alpha_{Q_k}| \leq \alpha_c$$

$$\left| \arccos\left(\frac{p_k - \mu_p}{\sigma_p \sqrt{n-1}}\right) - \arccos\left(\frac{q_k - \mu_q}{\sigma_q \sqrt{n-1}}\right) \right| \leq \arcsin\left(\frac{\epsilon}{\sigma_Q \sqrt{n}}\right)$$

3.5.4 The MS- α Approximation Shape

From Theorems 1, 2 and 3, μ , σ and α values essentially define some hyper-surfaces, μ defines a iso- μ hyper-plane and vice versa, same for σ and α , when we say μ , the iso- μ hyper-plane is also implied.

Theorem 4 *Let Q be the query point, ϵ the search radius, the search sphere $\mathcal{S}(Q, \epsilon)$ can be approximated by a piece of a hyper-ring, the hyper-ring is defined by the following μ and σ thresholds, the piece is cut by the α -Constant planes defined by the following α thresholds.*

$$\mu_{\min} = \max(0, \mu_Q - \epsilon/\sqrt{n})$$

$$\mu_{\max} = \mu_Q + \epsilon/\sqrt{n}$$

$$\sigma_{\min} = \max(0, \sigma_Q - \epsilon/\sqrt{n})$$

$$\sigma_{\max} = \sigma_Q + \epsilon/\sqrt{n}$$

$$\alpha_{\min} = \max(0, \alpha_Q - \alpha_\epsilon)$$

$$\alpha_{\max} = \alpha_Q + \alpha_\epsilon$$

Proof: These thresholds should be straightforward from Lemmas 1, 2 and 3.

In fact the shape of the approximation is illustrated in Figure 13. Since the means bound parallels the diagonal line which is the longest line in the data space, as long as the search radius does not exceed the longest possible length in the data space, the MS- α approximation will never include the entire data space, essentially this approximation scheme solves the dimensionality curse by a tight yet efficient geometric approximation of the search area.

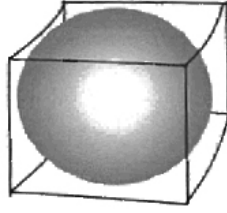


Figure 13: Shape of the MS- α approximation

3.6 Subspace Bounds and MS- α Configurations

The search sphere $\mathfrak{S}(Q, \epsilon)$ can be projected to an m dimensional subspace D_m , for $1 \leq m \leq n$, we have,

$$D^{(m)}(P, Q) = \left[\sum_{i=1}^m (p_i - q_i)^2 \right]^{\frac{1}{2}}$$

$$D(P, Q) = \left[\sum_{i=1}^n (p_i - q_i)^2 \right]^{\frac{1}{2}}$$

$$D^{(m)}(P, Q) \leq D(P, Q) \leq \epsilon$$

which tells us that the projected sub search sphere satisfies the same condition for the n dimensional search sphere. Since P satisfying the same pre-condition for Lemmas 1, 2 and 3 both in the n and the m dimensions, we can use the lemmas to prove the following corollaries.

Corollary 1 Let Q be the query point, ϵ the search radius and P any point in the search sphere $\mathfrak{S}(Q, \epsilon)$ i.e. P satisfying Equation 1, for $1 \leq m \leq n$, we have,

$$|\mu_P^{(m)} - \mu_Q^{(m)}| \leq \frac{\epsilon}{\sqrt{m}}$$

Corollary 2 Let Q be the query point, ε the search radius and P any point in the search sphere $\mathcal{S}(Q, \varepsilon)$ i.e. P satisfying Equation 1, for $1 \leq m \leq n$, we have,

$$|\sigma_P^{(m)} - \sigma_Q^{(m)}| \leq \frac{\varepsilon}{\sqrt{m}}$$

Corollary 3 Let Q be the query point, ε the search radius and P any point in the search sphere $\mathcal{S}(Q, \varepsilon)$ i.e. P satisfying Equation 1, for $1 \leq m \leq n$, we have,

$$\left| \arccos\left(\frac{P_k - \mu_P^{(m)}}{\sigma_P^{(m)} \sqrt{m-1}}\right) - \arccos\left(\frac{q_k - \mu_Q^{(m)}}{\sigma_Q^{(m)} \sqrt{m-1}}\right) \right| \leq \arcsin\left(\frac{\varepsilon}{\sigma_Q^{(m)} \sqrt{m}}\right)$$

We can use these corollaries to bind the search area in groups of dimensions, an illustration of subspace bounding is given in Figure 14.

Subspace bounding introduces the concept of MS- α configuration that is the combination of the number of subspaces and the number of dimensions in each subspace. The configuration of n subspaces (1 dimension in each subspace) is actually the hyper-cube approximation; the configuration of 1 subspace reduces the dimensions to 3 (μ , σ and α), an important aspect of the MS- α approach is to find the configuration for optimal performance. For a configuration of k subspaces with same number of dimensions, $m = n/k$ is the number of dimensions in each subspace and the dimensionality is reduced to $3k$. Choice of a configuration should take into account of both the approximation accuracy and the number of the reduced dimensions ($3k$), a configuration that renders the most accurate approximation may not be the optimal one if it requires more dimensions than a less accurate configuration.

As for iDistance, a comparison of Figure 9 and Figure 14 shows that MS- α is more accurate. iDistance's accuracy suffers more at higher dimensionality and denser data space, we'll try to demonstrate this in our experiments. In chapter 2, we established that VA-File and LPC-File incurs $O(n*N)$ in calculating d_{min} ; iDistance has a $O(n*n)$ portion in calculating query distances from reference points and $O(N)$ for accessing all approximation data; MS- α has to calculate $(\mu, \sigma$ and $\alpha)$ for the query point, time complexity for this part is $O(k)$, it's clear that accessing and evaluating all approximation data carries $O(k*N)$ complexity. Since k is a number from 1 to 4, MS- α is obviously more efficient than VA-File and LPC-File in the filtering stage, our experiments also show that it is in fact more efficient than iDistance at high dimensionality due to the $O(n*n)$ portion that iDistance carries.

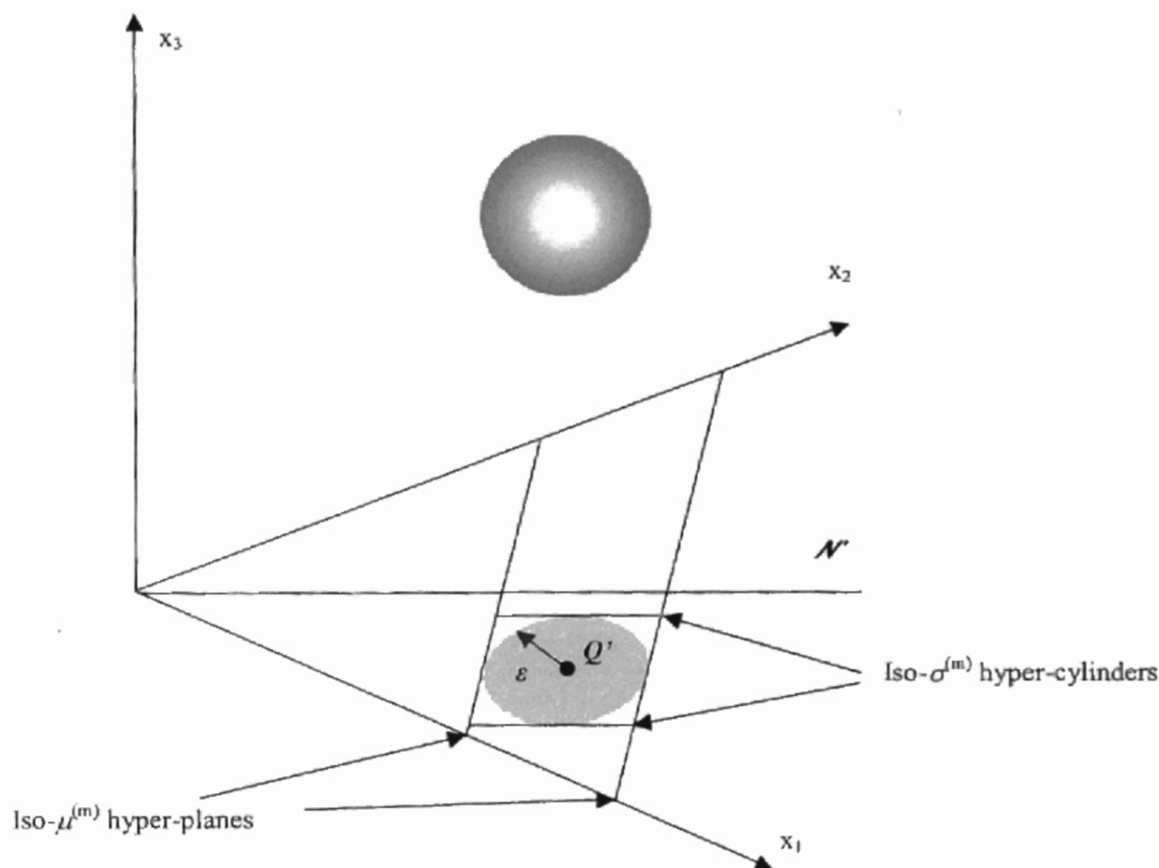


Figure 14: Subspace bounding on a 2 dimensional projection

3.7 Comparison with Existing Approaches

Using the same filtering stage as in VA-File and LPC-File, the compression rate of the MS- α is determined by the number of subspaces k in a configuration. For $k = 1$, the approximation data point consumes 12B ($3 \times 4 = 12$, assuming μ , σ and α are 4B floats), suppose each coordinate of a data point is a 4B (32-bit) float, in 256 dimensions, the compression rate for MS- α is less than 0.012 as opposed to 0.125 ~ 0.25 for VA-File and LPC-File, for one byte data point, it is less than 0.047 as opposed to 1 (no compression). Although k maybe higher for optimal configuration, it is clear that the optimal k is not linear to n , because if $k = n$, MS- α degenerates to the most ineffective hyper-cube.

CHAPTER 4

Objectives

Experiments show that MS8 (the approximation scheme using only the means and standard deviation bounds with a configuration of $k=8$) significantly outperforms simple scan, hyper-cubic based approximation schemes and data transformation schemes in high dimensionality ([10]).

Recently filter based VA-File, LPC-File and the iDistance methods have been brought to our attention, we believe MS- α should outperform or at least be a competitor for these approaches. We aim to demonstrate our thoughts by experimenting these methods in spherical range search, all 4 methods were implemented in a way (see the next chapter) such that experimental results on them can yield fair comparison.

Though $k=8$ has been shown to be the optimal configuration for MS at 256 dimensions ([10]), we think a smaller k might be the optimal for MS- α because of the introduction of the α -Constant bounds which significantly improves approximation accuracy at or near the center of the data space. We performed experiments to find the optimal configuration for MS- α .

Besides experimenting with various configurations, we also experimented with various dimensions and search radii. Both approximation accuracy and response time were used in the comparative study.

CHAPTER 5

Technical Methodology

In this chapter we shall describe our method that aims to provide a fair platform for our comparative study of the 4 search approaches (MS- α , VA-File, LPC-File and iDistance).

5.1 Data Preparation

The (μ, σ, α) values of computer generated random data have a sharp bell shaped data distribution which renders MS- α ineffective in the filtering stage, fortunately real data has a more uniform distribution of these 3 tuples ([10]), we used real data in our experiments.

Real data were obtained by sampling images in RGB color format at 256 locations which constitutes the image's 256 feature vector (a data point in D_{256} data space), the R color at each location is a byte value and used as the coordinate of one dimension. Lower dimension data were obtained by selecting the lower dimension coordinates of each 256 feature vector. More than 15,000 data points were used in our experiments.

Approximation data $\{(\mu, \sigma, \alpha)$ tuple for MS- α , bit-string for VA-File, (b, r, θ) tuple for LPC-File and $i * c + D(P, R_i)$ for iDistance} were generated based on base data and stored in a flat file (the compressed file) in the same order as the base data so that the base data point can be easily located based on a candidate's position in the compressed file.

5.2 Algorithm Implementation

Spherical range search were implemented for all 4 methods in a two-stage (the filtering and the random access stage) algorithm. We choose to use 8-bit to approximate each dimension for VA-File and LPC-File, though smaller number of bits can be used at high dimensions, 8-bit is chosen because it demonstrates more evidently that their performance deteriorates linearly with dimensions.

The filtering stage is a simple scan of the compressed file, based on each of the 4 method's filtering criteria {the threshold values in Theorem 4 for MS- α , local distance within $\max(0, D(O_i, Q) - \epsilon)$ to $\min(d_{\max}^i, D(O_i, Q) + \epsilon)$ for iDistance and $d_{\min} \leq \epsilon$ for VA-File and LPC-File} the ID of the candidate approximation data is stored in a memory heap.

The random access stage traverses the candidate heap, performs a random access to the base file to find the 'true' data point based on the ID of the candidate, and only those points that satisfy Equation 1 are kept in the result set.

Because same file and memory access methods were used for all 4 methods and the framework is the same, our implementation provides a fair platform for comparative study.

5.3 Experiment Methodology

For a certain combination of search method, dimensionality and search radius, 99 query points were extracted from the base data file upon each of which a spherical range search was performed, and the average result were used for our study.

To find the optimal configuration for MS- α , we conducted experiments with n (number of dimensions) in {8, 16, 32, 64, 128, 256}, k (number of subspaces) in {1, 2, 4, 8, 16}, the k value which generates the smallest candidate set and consumes the least response time is the optimal configuration for the specific n and was used in our comparative experiments.

In our comparative study, experiments were performed at 8, 16, 32, 64, 128, 256 dimensions, search radii in {0.2, 0.6, 1, 1.4, 1.8, 2.2, 2.6, 3}. We watched the %return (number of candidate points as opposed to total number of points) and response time in the filtering stage to evaluate the performance of all methods in the filtering stage. Total response time was used as the yardstick for overall performance evaluation.

CHAPTER 6

Comparative Study and Performance Comparison

Our experiments were performed on a Dell Dimension L1000R PC running Windows 2000 with 1GHZ CPU and 256MEG RAM.

6.1 MS- α Optimal Configuration

In order to find the optimal configuration of MS- α , we conducted spherical range searches for various k and search radii, results for 256 dimensions are given in figure 15.

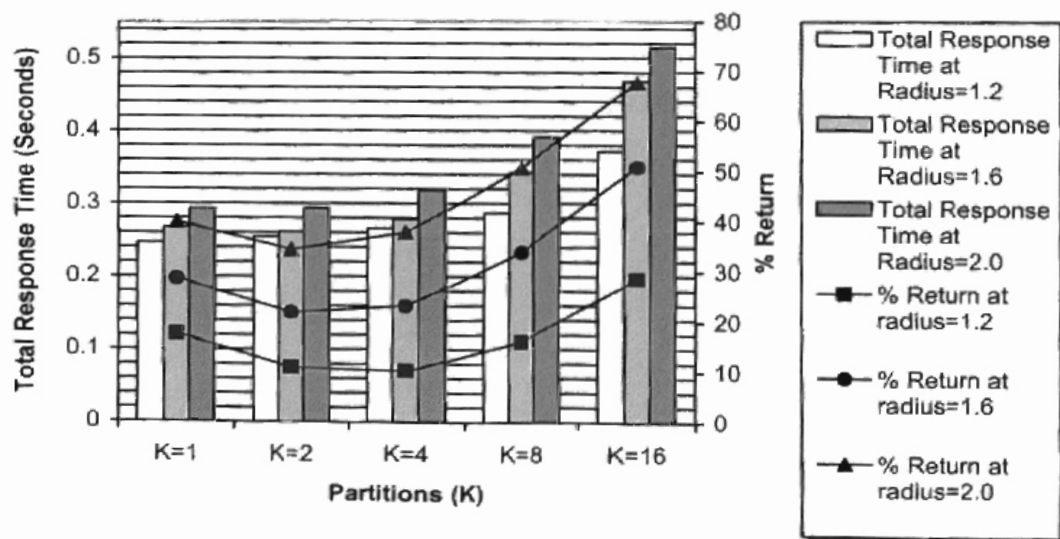


Figure 15: MS- α configurations at 256 dimensions

Search radii 1.2, 1.6 and 2.0 were chosen because they are in the medium radius range of our comparative study (0.2 - 3.0). From the above figure, the configuration at $k=2$

provides the smallest %return and near optimal total response time. The experiments were performed on byte sized base data, for 2 byte short integer and 4 byte float base data, $k=2$ configuration should provide more evident performance advantage over $k=1$ because smaller %return in the filtering stage saves more time in the random access stage for bigger base data file. We decided to designate $k=2$ as the optimal configuration for 256 dimensions and used it in the following experiments.

Similar experiments were performed for dimensions 8, 16, ..., 128, and the optimal configurations are plotted in Figure 16. Clearly k grows very slowly with dimensionality (sub-logarithm indeed).

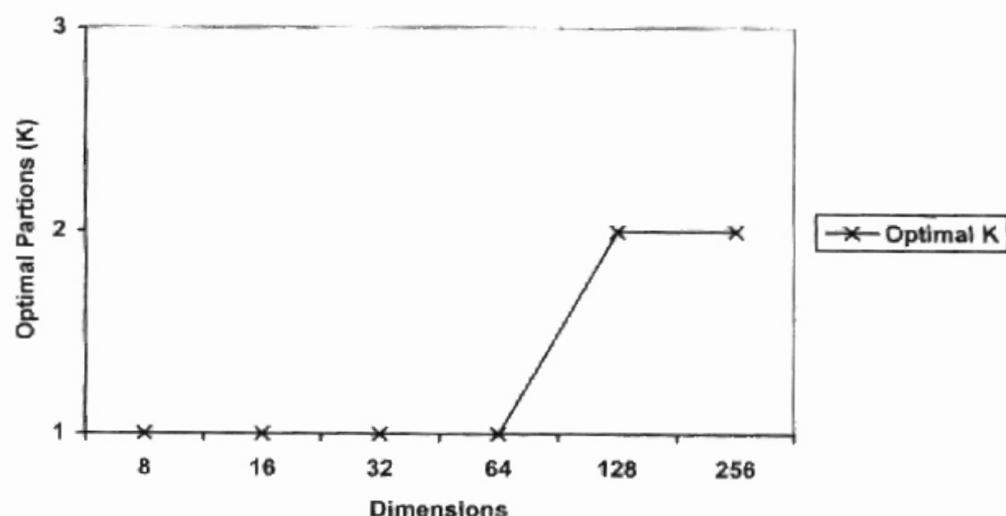


Figure 16: MS- α optimal configuration partitions (k) at various dimensions

6.2 Analysis of the Filtering Stage

In section 3.7, we discussed the performance of the four methods in the filtering stage. To prove our analysis, we conducted spherical range searches across dimensions 8, 16 to 256. We chose to use a big search radius for each of these searches so that the filtering

stage returns all data points, thus excluding the affect of search radius in the filtering stage and provide a fair comparison. The results are given in Figure 17.

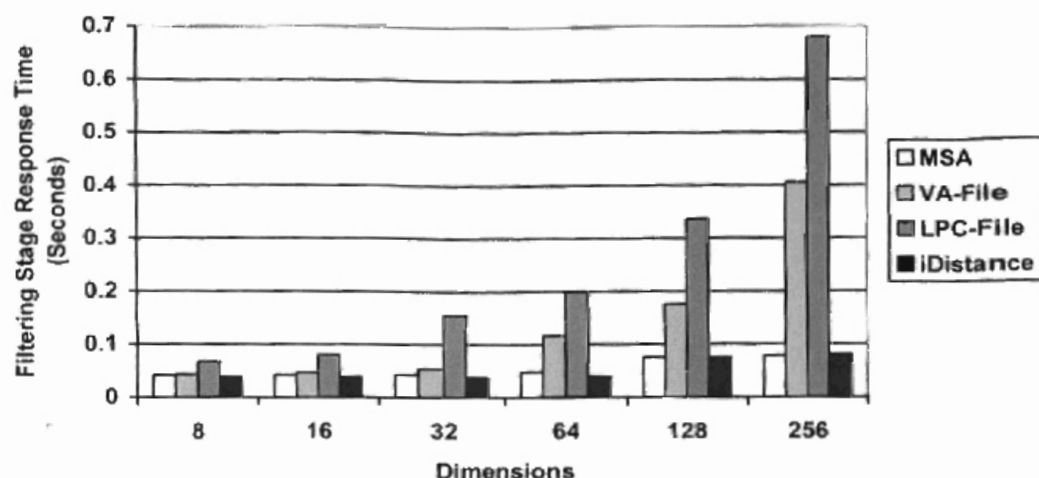


Figure 17: Filtering Stage Response Time at various dimensions

The response time of the filtering stage for MSA stays near constant for dimensions 8 through 64 and then doubles at 128 but almost no change at 256 dimensions, this is expected because optimal k only grows to 2 for 128 and 256 dimensions and time complexity is $O(k*N)$.

Figure 17 clearly shows the near linear growth of response time for VA-File and LPC-File, and the penalty that LPC-File suffers for the time it spends on calculating the angles and distances.

iDistance shows comparative filtering stage response time with MSA, based on the discussion in section 3.7, we believe the spike at dimensions 128 and 256 is due to the $O(n*n)$ portion of the calculations.

6.3 Performance Study at 256 Dimensions

We performed spherical range searches across search radii 0.2, 0.6,, 3.0 at 256 dimensions. We paid attention to approximation file size, %return, filtering stage response time and total response time.

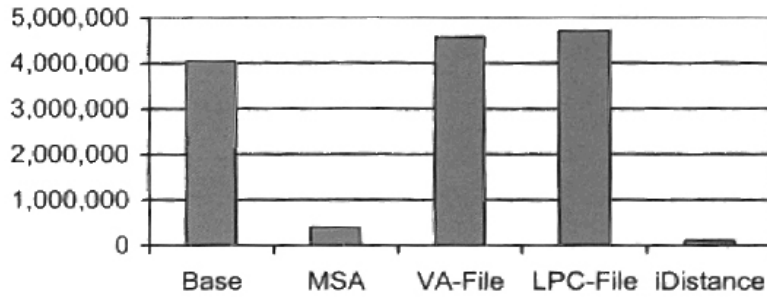


Figure 18: Approximation file size at 256 dimensions on 15,766 byte-sized data points

Figure 18 plots the approximation file sizes of the base data and the 4 approximation methods. Using 8-bit approximation scheme, VA-File and LPC-File achieve no compression. iDistance has the smallest approximation file as is expected. The approximation file size for MS- α is based on optimal configuration at 256 dimensions ($k=2$), it is roughly $1/10^{\text{th}}$ of the base file size.

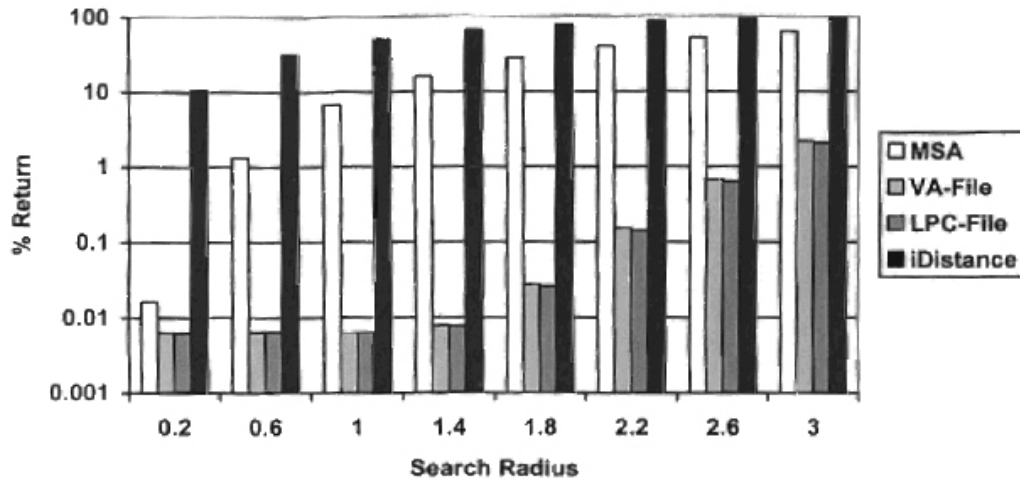


Figure 19: Filtering Stage % Return at 256 dimensions and various search radii

Figure 19 plots the %return of the 4 methods, VA-File and LPC-File provide excellent approximation accuracy, almost 100 times better than MS- α and iDistance. Note the scale of y-axis in this figure is logarithm, MS- α provides significant accuracy improvement over iDistance.

Figure 20 shows the response time in the filtering stage, MS- α outperforms all 3 other methods in this regard.

Figure 18, 19 and 20 demonstrate that MS- α strikes a balance between approximation accuracy and efficiency.

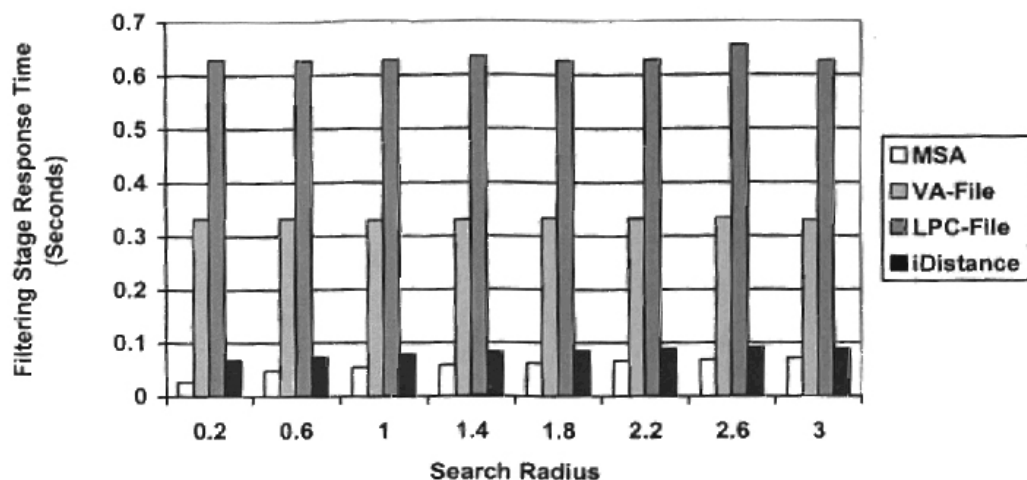


Figure 20: Filtering Stage Response Time at 256 dimensions and various search radii

Finally the yardstick (Total Response Time) for performance comparison is given in Figure 21, results for Simple Scan is also plotted. We can see that MS- α consistently outperforms all other methods including Scan, until search radius 3 when all 4 methods under-performs Scan. Search radius beyond 3 is not important for similarity search because 2% of data points are within the search sphere, for our more than 15,000 base data points, that's more than 300 points.

LPC-File actually is worse than Scan in our study because the data distribution is not highly skewed as in the original study ([2]), and data density is quite low (15,000 as opposed to 1,000,000 in ([2])). But LPC-File has its place when data distribution and density is in its favor.

At search radius 2.2, VA-File under-performs Scan, iDistance under-performs Scan at search radius 1.4.

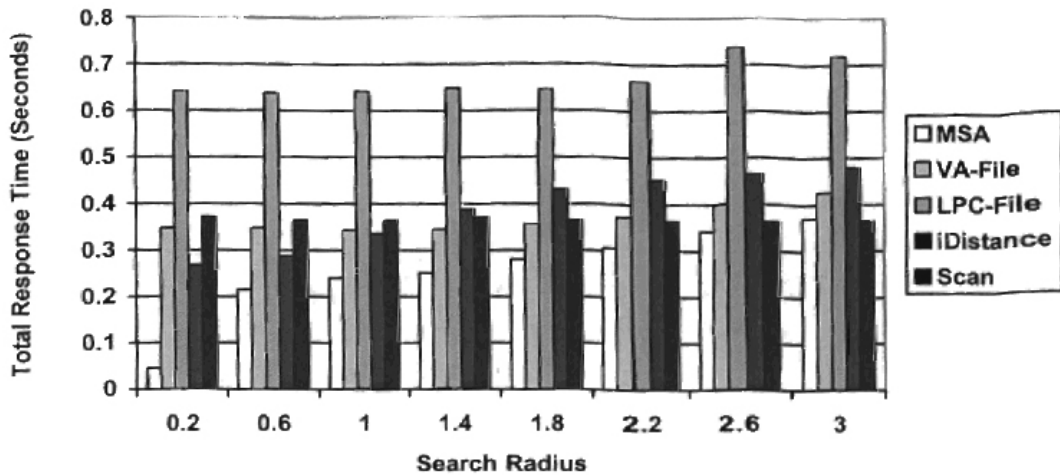


Figure 21: Total Response Time at 256 dimensions and various search radii

(Base data: 256 byte color histograms)

Experiments on 2-byte short integer and 4-byte float base data were also conducted (results not plotted here), VA-File outperforms MS- α around search radius 0.4 ~ 0.6 on these base data because its filtering power gains significant savings in the random search stage, MS- α still consistently outperforms LPC-File, iDistance and Scan.

CHAPTER 7

Conclusions

We proposed MS- α as a solution for high-dimensional multimedia database search. MS- α strikes a balance between approximation accuracy and efficiency, and scales well with dimensionality and search radii.

It outperforms Simple Scan, traditional hyper-cube based approximation, data transformation techniques and dimensionality reduction techniques.

Based on our experiments, MS- α consistently outperforms recently proposed LPC-File and iDistance methods. It consistently outperforms VA-File on byte sized base data in entire range of search radii, and in small radii on short integer and float base data.

We conclude that MS- α is an efficient solution for the *curse of dimensionality*, and is especially powerful in multimedia image databases such as color histograms.

REFERENCES

- [1] R. Webber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces" in *Proc. 24th Int. Conf. VLDB*, 1998, pp. 194–205.
- [2] Guang-Ho Cha, Xiaoming Zhu, Dragutin Petkovic, Fellow, IEEE, and Chin-Wan Chung, "An Efficient Indexing Method for Nearest Neighbor Searches in High-Dimensional Image Databases" in *IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 4, NO. 1, MARCH 2002*.
- [3] A. Csillaghy, "Information extraction by local density analysis: A contribution to content-based management of scientific data", *Ph.D. thesis, Institut für Informationssysteme*, 1997.
- [4] A. Dimai, "Differences of global features for region indexing" in *Technical Report 177, ETH Zürich*, Feb. 1997.
- [5] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system." in *Computer*, 28(9):23-32, Sept.1995.
- [6] M. Stricker and M. Orengo, "Similarity of color images. In Storage and Retrieval for Image and Video Databases" in *SPIE, San Jose, CA*, 1995.
- [7] D. Goldin and P. Kanellakis, "On similarity queries for time-series data: Constraint specifications and implementation." in *Proc. of Constraint Programming*, Sep 1995.

- [8] A. V. Oppenheim and R. W. Schaffer, "Digital Signal Processing", *PrenticeHall, Englewood Cliffs, N.J.*, 1975.
- [9] D. Rafiei and A. Mendelzon, "Similarity-based queries for time series data" in *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, May 1997.
- [10] Khanh Vu, Kien A. Hua and S.D. Lang, "Efficient Spherical Range Search in Multimedia Databases" in *IEEE (to appear)*
- [11] C. Yu, B. C. Ooi, K.-L. Tan, and H. V. Jagadish, "Indexing the distance: An efficient method to KNN processing" in *The VLDB Journal*, pages 421–430, 2001.
- [12] S. Berchtold, C. Böhm, B. Braunmüller, D. Keim, and H.-P. Kriegel, "Fast parallel similarity search in multimedia databases" in *Proc. of the ACM SIGMOD Int. Conf. On Management of Data*, pages 1-12, Tucson, USA, 1997.
- [13] N. Beckman, H. Kriegel, R. Schneider, and B. Seeger, "The r^* -tree: an efficient and robust access method for points and rectangles" in *ACM SIGMOD*, pages 322–331, May 1990.
- [14] S. Berchtold, C. Böhm, and H. Kriegel, "The pyramic-technique: Toward breaking the curse of dimensionality" in *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 142–153, 1998.
- [15] R. Orlandic, J. Lukaszuk, and C. Swietlik, "The design of a retrieval technique for highdimensional data on tertiary storage" in *SIGMOD Record*, 31(2):15–21, June 2002.
- [16] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases" in *Proc. of the FODO Conference*, Oct 1993.

- [17] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, “Fast subsequence matching in timeseries databases” in *Proc. ACM SIGMOD*, pages 419–429, May 1994.
- [18] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber, “Efficient and effective querying by image content” in *Journal of Intelligent Information Systems*, 3(3):231-262, 1994.
- [19] C. Li, P. Yu, and V. Castelli, “Hierarchyscan: A hierarchical similarity search algorithm for databases of long sequences” in *Proc. of ICDE*, pages 546–553, 1996.
- [20] K. Kanth, D. Agrawal, and A. Singh, “Dimensionality reduction for similarity searching in dynamic databases” in *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 166–176, 1998.

2.

VITA

Xiaoyong Duan

Candidate for the Degree of

Master of Science

Thesis: MS-a - AN EFFICIENT, BALANCED AND SCALABLE APPROACH TO SPHERICAL RANGE SEARCH IN HIGH-DIMENSIONAL MULTIMEDIA DATABASES

Major Field: Computer Science

Biographical:

Education: Graduated from Fuxing High School, Shanghai, China in July 1984; received a Bachelor of Science degree in Physics and a Master of Science degree in Physics from Fudan university, Shanghai, China in July 1988 and July 1991, respectively. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in August 2003.

Experience: Employed as a software engineer by Shanghai MPSI Ltd., 1991 to 1993; received training as a software engineer at MPSI Inc., Tulsa, Oklahoma, 1993 to 1994; employed as programmer/analyst by Diversified Systems Resources, Inc., 1994 to present.