FINDING OPTIMIZED BOUNDING BOXES OF POLYTOPES IN D-DIMENSIONAL SPACE AND THEIR PROPERTIES IN K-DIMENSIONAL PROJECTIONS

By

Salman Shahid

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science–Doctor of Philosophy

2014

ABSTRACT

FINDING OPTIMIZED BOUNDING BOXES OF POLYTOPES IN D-DIMENSIONAL SPACE AND THEIR PROPERTIES IN K-DIMENSIONAL PROJECTIONS

By

Salman Shahid

Using minimal bounding boxes to encapsulate or approximate a set of points in d-dimensional space is a non-trivial problem that has applications in a variety of fields including collision detection, object rendering, high dimensional databases and statistical analysis to name a few. While a significant amount of work has been done on the three dimensional variant of the problem (i.e. finding the minimum volume bounding box of a set of points in three dimensions), it is difficult to find a simple method to do the same for higher dimensions. Even in three dimensions existing methods suffer from either high time complexity or suboptimal results with a speed up in execution time. In this thesis we present a new approach to find the optimized minimum bounding boxes of a set of points defining convex polytopes in d-dimensional space. The solution also gives the optimal bounding box in three dimensions with a much simpler implementation while significantly speeding up the execution time for a large number of vertices. The basis of the proposed approach is a series of unique properties of the k-dimensional projections that are leveraged into an algorithm. This algorithm works by constructing the convex hulls of a given set of points and optimizing the projections of those hulls in two dimensional space using the new concept of Simultaneous Local Optimal. We show that the proposed algorithm provides significantly better performances than those of the current state of the art approach on the basis of time and accuracy. To illustrate the importance of the result in terms of a real world application, the optimized bounding box

algorithm is used to develop a method for carrying out range queries in high dimensional databases. This method uses data transformation techniques in conjunction with a set of heuristics to provide significant performance improvement.

For Rabeel, Ashhad & Hashir

ACKNOWLEDGMENTS

I'm grateful to my wife for her continuous support and help throughout my PhD without which all of this would not have been possible. I'd also like to thank Dr. Sakti Pramanik for his kindness, advice and continuous help in guiding me through this process. I'm also indebted to Dr. Charles Owen for his help in giving key parts of my work direction as well as the International Institute of Education, New York for the Fulbright Grant that made this work possible. Finally, I'm deeply obliged and thankful to my parents for the love and prayers that helped me through this and every other part of my academic career.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **ix**

LIST OF FIGURES . . . . . **xi**

LIST OF ALGORITHMS . . . . . **xiii**

Chapter 1 Introduction **1**
1.1 Motivation 1
1.1.1 Minimum Bounding Box For 3-Dimensional Cross-Polytopes 2
1.1.2 Minimum Bounding Box For Arbitrary D-Dimensional Polytopes . . . 3
1.1.3 Querying And Indexing High-Dimensional Databases 4
1.2 Basic Concepts 5
1.2.1 Definitions 6
1.3 Contributions & Structure 7

Chapter 2 Related Work **9**
2.1 Finding Bounding Boxes in 2-Dimensions 9
2.2 Finding Bounding Boxes in 3-Dimensions 10
2.3 Finding Bounding Boxes in d-Dimensions 12
2.4 Querying And Indexing Techniques in High Dimensions 13
2.4.1 Collision Detection Using Bounding Boxes 16

Chapter 3 Minimum Bounding Box of a Three Dimensional Cross-Polytope 18
3.1 Two Edges Flush 18
3.2 One Face Flush 20
3.2.1 **Case - I: *Angle Between Adjacent Edges is 90 Degrees:*** . . . 20
3.2.2 **Case - II: *Angle Between Adjacent Edges is 60 Degrees*** 22
3.3 Bounding Box Under Projection 28
3.3.1 Minimality of Projected Bounding Box 28
3.4 Cross-Polytope under Projection 30
3.4.1 Definitions 30
3.4.2 Projected Polygons 32
3.5 Minimal Bounding Box by Optimal Projections 33
3.5.1 Unique Simultaneous Local Optimality 33
3.5.2 Simultaneous Local Optimal By Combination Reduction 36
3.6 Summary 38

Chapter 4 Optimized Bounding Box of Arbitrary D-Dimensional Polytope 40
4.1 Minimizing Bounding Box Based On Locally Optimal Projections 43

4.1.1 Definitions . . . 43
4.1.2 Minimality of Projected Bounding Box . . . 44
4.1.3 Edges Flush in d-Dimensions . . . 45
4.1.4 Existence of Simultaneous Local Optimal For All d-Dimensional Polytopes . . . 47
4.1.5 Minimality of D-Dimensional Bounding Box . . . 47
4.1.6 Convergence of Projections . . . 48
4.1.7 Period of Projected Bounding Box in 2D . . . 55
4.2 Optimized Bounding Box Algorithm . . . 60
4.2.1 The Simultaneous Local Optimal Algorithm . . . 62
4.2.2 The Minimum Bounding Box Candidate Algorithm - Exhaustive Approach . . . 66
4.2.3 Minimum Bounding Box Candidate Algorithm - Heuristic Approach . 70
4.2.3.1 Heuristic 1 . . . 73
4.2.3.2 Heusristic 2 . . . 73
4.2.3.3 Heuristic 3 . . . 74
4.2.4 Local Optimal Clustering . . . 75
4.2.5 Minimum Bounding Box Candidate Algorithm - Grid Based Clustering 77
4.2.5.1 Optimizations to Algorithm . . . 78
4.2.6 The Clustered Simultaneous Local Optimal Algorithm) . . . 80
4.3 Experimental Results . . . 80
4.3.1 Experimental Setup . . . 82
4.3.2 Theoretical Analysis . . . 84
4.3.2.1 Optimized Bounding Box - SLO . . . 84
4.3.2.2 Optimized Bounding Box - MBBC(Exhaustive) . . . 84
4.3.2.3 Optimized Bounding Box - MBBC(Heuristic) . . . 85
4.3.3 Minimum Bounding Boxes For 3-Dimensional Polytopes . . . 85
4.3.3.1 Effect of Varying Heuristics . . . 86
4.3.3.2 Effect of Increasing Vertices . . . 87
4.3.3.3 Effect of Increasing Granularity . . . 88
4.3.4 Minimum Bounding Boxes For d-dimensional Polytopes . . . 89
4.3.4.1 Effect of Increasing Granularity . . . 93
4.3.4.2 Effect of Increasing Dimensions . . . 93
4.3.4.3 Effect of Increasing Vertices . . . 96
4.3.4.4 Effect of Varying Heuristics . . . 97
4.4 Summary & Conclusion . . . 98

Chapter 5 Range Query In High Dimensional Databases Using Topological Transformation . . . 100
5.1 Motivation . . . 101
5.2 Methodology . . . 102
5.3 Topological Transformation in Multi-Dimensional Space . . . 104
5.3.1 Topological Transformation For d ≤ 3 . . . 105
5.3.2 Topological Transformation for d > 3 Using SLO . . . 107
5.3.2.1 Tranformation Property . . . 110

5.3.2.2 Transformation Heuristic . . . 111
5.3.3 Performance Evaluation . . . 113
5.3.3.1 Experimental Setup . . . 113
5.3.3.2 Effect of Increasing Dimensions . . . 113
5.3.3.3 Effect of Increasing Ranges . . . 115
5.3.3.4 Effect of Increasing Database Size . . . 115
5.3.3.5 I/O Optimization . . . 116
5.4 Summary & Conclusion . . . 117

Chapter 6 Conclusion . . . 118

BIBLIOGRAPHY . . . 121

LIST OF TABLES

Table 4.1 Cluster Statistics With Varying Dimensions 77

Table 4.2 Standard Deviation of Volume With Varying Vertices & Granularity in % for 3D 80

Table 4.3 Standard Deviation of Volume With Varying Vertices & Granularity in % for 4D 80

Table 4.4 Effects of Heuristics in 3D - 10V 90

Table 4.5 Effect of Heuristics in 3D - 50V 90

Table 4.6 Effect of Heuristics in 3D - 100V 90

Table 4.7 Effect of Heuristics on 3D - 200V 91

Table 4.8 Effect of Heuristics in 3D - 500V 91

Table 4.9 Comparison of Optimization For Polytopes in d-dimensions 95

Table 4.10 Optimization With Changing Heuristics in 4D 97

Table 4.11 Optimization of Volume With Changing Heuristics in 5D 98

Table 4.12 Volume Optimization with Changing Heuristics in 6D 98

Table 5.1 Volume Optimization for Bounding Box for Unit Hyper-Diamond . . 112

Table 5.2 Run Time in Seconds with Increasing Dimensions for a Range of 0.1 . 114

Table 5.3 Run Time in Seconds with Increasing Dimensions For a Range of 0.2 114

Table 5.4 Run Time in Seconds with Increasing Dimensions For a Range of 0.3 115

Table 5.5 Run Time in Seconds with Increasing Dimensions For a Range of 0.4 115

Table 5.6 Execution Time With Increase in Database Size 116

LIST OF FIGURES

Figure 3.1 Projection of the Octahedron on the X-Y Plane 21

Figure 3.2 Volume of the MBR as a function of θ 21

Figure 3.3 Position of Octahedron Against the Adjacent Faces of MBB for Case-I 22

Figure 3.4 Position of Octahedron Against the Adjacent Faces of MBB for Case-II 23

Figure 3.5 Local Maxima of X 27

Figure 3.6 Local Maxima of Y 28

Figure 3.7 Local Maxima of Z 28

Figure 3.8 3D-diamond in upright position 30

Figure 3.9 Projection of 3D-diamond on XY-plane 31

Figure 4.1 Examples of Clusters for a Random Polytope 77

Figure 4.2 Examples of Random Polytopes Tested 82

Figure 4.3 Execution Time for SLO With High Number of Vertices 89

Figure 4.4 Execution Time for SLO and ORourke 89

Figure 4.5 % Accuracy of SLO for Different Granularities 92

Figure 4.6 % Accuracy of SLO & PCA for Increasing No. of Vertices 92

Figure 4.7 % Error of Polytopes Not Matching ORourke 92

Figure 4.8 %Volume Optimization For a 4D-Polytope using SLO 94

Figure 4.9 %Volume Optimization For a 5D-Polytope using SLO 94

Figure 4.10 %Volume Optimization For a 6D-Polytope using SLO 94

Figure 4.11 %Execution Time For D-Dimensions With A Granularity of 1 95

Figure 4.12 %Optimization in d-dimensions using SLO 96

Figure 4.13 %Execution Time in 4D With Varying Vertices 97

Figure 5.1 Transformation of Range Query 105

Figure 5.2 3D-diamond in upright position 107

Figure 5.3 Time Optimization With Increasing Range 116

Figure 5.4 Comparison of I/O Optimization 117

LIST OF ALGORITHMS

Algorithm 4.1 Algorithm-SLO: Simultaneously Local Optimal Algorithm 65

Algorithm 4.2 Algorithm-MBC: Minimum Bounding Box Candidates - Exhaustive 68

Algorithm 4.3 Algorithm-MBC: Minimum Bounding Box Candidates - Heuristic . 72

Algorithm 4.4 Algorithm-MBC: Minimum Bounding Box Candidates - Clustering 81

Chapter 1

Introduction

1.1 Motivation

Using boxes as abstractions to represent an object or a region of space encompassing a given set of points is a common technique that has been used extensively in a variety of applications. The objective is usually to simplify the calculations involved and provide an improved method of describing the abstracted object/point's interaction with its environment.

To exemplify, bounding boxes in d-dimensions form the basis of many spatial and data indexing schemes such as R-Trees, R*-Tress, Box-Trees amongst others. These methods use the bounding box abstraction to subdivide space/data to facilitate analysis and queries on the data space.

They also form an integral part of a number of problems in computational geometry and computer graphics such as ray tracing, frustum culling, collision detection and hidden object detection amongst others.

Interference and collision detection are fundamental problems in a variety of application domains, such as physically-based modeling, robotics, animation, computer-aided design, manufacturing, and computer simulated environments.

The tightness of the bounding box used to approximate the underlying abstraction or object is an important performance constraint in these applications. Thus, having a minimum axis-aligned bounding box results in an automatic improvement in the performance

achieved irrespective of the application. Extensive work has also been done on detecting interpenetration of objects in static and dynamic environments using oriented bounding box hierarchies.

The approach presented in our work has interesting implications for these techniques since it presents a fast mechanism to produce minimized bounding boxes and thus tighter fitting OBB-Trees.

These specific problems solved using the techniques presented here are now discussed in detail.

1.1.1 Minimum Bounding Box For 3-Dimensional Cross-Polytopes

The ability to determine the minimum bounding box of a regular octahedron simply and accurately has significant implications for a variety of applications. Minimum bounding boxes in three dimensions are used among other things to hierarchically partition a given set of points. Creating and maintaining these partitions is an integral part of applications like collision detection and scene rendering in computer graphics. Furthermore, statistical applications like storing and performing range queries on large databases utilize these constructs to a large extent to optimally partition and look-up data. The are also extensively used in collision detection algorithms in three dimensional graphic rendering applications. They are good approximations for the component polyhedrons used as building blocks in these techniques and also lend themselves well to the indexing structures they utilize. We focus on a particular aspect of this problem, namely the determination of minimum volume bounding boxes for sets of points that define regular, uniform octahedrons in three dimensional space. Given that the set of points to be partitioned in time-sensitive applications like database range-search queries define octahedrons in three-dimensional space, optimizing

minimum bounding box calculation for such constructs becomes rather important. We prove the existence of a constraining condition that makes it quite simple to determine the bounding boxes of regular cross-polytopes. Furthermore, we also show how finding a minimum box in higher dimensions guarantees minimum bounding boxes in lower dimensions. At the end we prove that for three dimensions, the minimum bounding boxes for the projections of the encapsulated points in two dimensions, can be used to find the minimum volume bounding box for those points in three dimensions. We also provide proof of the forms the convex hulls of the projected points take in two dimensions.

1.1.2 Minimum Bounding Box For Arbitrary D-Dimensional Polytopes

There exists a significant body of work focusing primarily on the three dimensional variant of the problem. These methods suffer from drawbacks such as inefficient performance in terms of time when the maxim volume optimization is obtained to sub-optimal results when performance is improved. However, there hasn't been a lot work done on the d-dimensional variant of the problem. To the best of our knowledge excepting the theoretical technique using diameter estimation defined in [1] and the possible extension of PCA to estimate the bounding box, there does not exist a simple algorithm to find the optimized bounding bounding box of a set of points defining convex polytopes in d-dimensional space. As mentioned previously having an optimized bounding box in d-dimension can have interesting implications for database indexing and computer graphic algorithms. The tightness of the bounding box encapsulating a set of data objects in d-dimensional space in these applications is an important factor in the improving querying processing and intersection determination

respectively.

The results used to determine the minimum bounding box of a three dimensional cross polytope as well the properties if their *d-1* dimensional projections of these polytopes can be generalized to some extent to d-dimensional arbitrary polytopes and projections in k dimensional subspace where $k < d$. To determine the optimized bounding boxes of such polytopes additional proofs and results are presented here that show that the bounding box of a d-dimensional polytope can be defined in terms of their simultaneously local projections in d dimensions. Combining these results we present a general method to find the optimized bounding box for such polytopes.

The techniques presented here could be considered to lie in the class of exhaustive search in the the d-dimensional rotation group $SO(d)$ with specific heuristics such that the search space is minimized. $SO(d)$ is the group of all rotations about the origin of d-dimensional space R^d. The heuristic utilized limits the time of execution by a constant factor while giving the exact optimal for all cases tested in three dimensions. Furthermore, it extends very easily to higher dimensions providing optimized bounding boxes again with tightly controlled execution time. It decomposes the d-dimensional problem to a two dimensional one, minimizing in $2d$ for the solution like certain variants of PCA. However, it is not dependent on a provided or calculated direction like those variants and thus escapes the associated drawbacks.

1.1.3 Querying And Indexing High-Dimensional Databases

For most large scale databases returning data points lying within a specified range efficiently is extremely important. Range queries or similarity queries based on L_1 distance are used commonly in large scale multimedia databases containing sets of feature vectors. This chap-

ter focuses on a method to improve this functionality for queries in d-dimensional L1 space. Using the L_1 distance measure to carry out the range query results in an interesting side effect. The range query forms a convex cross-polytope in d-dimensional space. For three dimensions this a regular octahedron.

To achieve the targeted aim of improved efficiency in terms of improved I/O and execution time, the range query is approximated with a minimal bounding box query using a topological transformation base on the optimization approach for d-dimensional bounding boxes. This query in conjunction with some heuristics is used to return equivalent results while achieving the stated objectives.

For efficient execution of range queries in very large databases, usually a multi-dimensional index is created and queries are implemented using this index. The effectiveness of an index for implementing the range query is determined by the number of pages (amount of I/O) accessed in the index tree and the time taken to return the data matching the query given.

The application methodology presented here to solve the range query problem provides simple isometric transformations based on minimized bounding boxes in d-dimensional space that offers significant performance improvement in terms of CPU time expended while utilizing a simple heuristic to match or improve upon the page accesses necessary. We also show how the improvements remain stable with increase in database size.

1.2 Basic Concepts

In the following sections the terms octahedron, cross-polytope and hyper-diamond refer to regular octahedrons or cross-polytopes unless stated otherwise. Similarly, a polytope is a geometric object with flat sides defined by a set of points in d-dimensions

1.2.1 Definitions

We first define some terminologies and definitions, which will serve as the basis of our discussion.

Definition 1.2.1 (Axis Aligned Bounding Box)**.** An axis aligned bounding box of a d-dimensional polytope P is defined as the box whose body diagonal is delimited by the minimum and maximum axial coordinate values of P.

The edges of an axis aligned bounding box are parallel to the axes of the coordinate space.

Definition 1.2.2 (Oriented Bounding Box)**.** An oriented bounding box of a d-dimensional polytope P is a tightly fitting rectangular bounding box of an arbitrary orientation in the d-dimensional space.

The edges of an oriented bounding box may not be parallel to the axes.

Definition 1.2.3 (Orientation in d-Dimensions)**.** Given an arbitrary polytope P in d-dimensions, all orientations of P can be obtained by rotating P in parallel with one or more of the axial planes in the d-dimensional space while the vertex coordinates of the axes that are not involved in the rotation remain fixed.

Definition 1.2.4. For a given k $(0 \leq k \leq)$ and d, if the axis-aligned bounding box of a $d-k$ dimensional projection of a d-dimensional polytope is its arbitrarily oriented minimum bounding box, then the projection is said to be locally optimal.

1.3 Contributions & Structure

In this work we present algorithms that attempt to provide a solution to the problems just discussed. While significant work has been done to solve the three dimensional variant of the problem including one that guarantees an exact minimum bounding box, each set of methods has a few problems that we attempt to address through the solution presented here. The exact minimum bounding algorithm is quite difficult to implement and exponential in terms of the number of vertices of the polytope being bounded. In practice it rapidly becomes infeasible to utilize in real world applications for polytopes having more than a few tens of vertices. The fastest set of methods are based on using versions of PCA which has been proven to be unbounded in the worst case and significantly sub-optimal in most. Other techniques provide approximation bounds that isolate the optimal only up to certain level of accuracy and could miss the exact optimal in most cases. Still others require several runs to isolate an approximately optimal solution, which affects the feasibility of their use in real-time applications. Applications like range queries in high dimensional multimedia databases and collision detection in static and dynamic environments utilize bounding boxes to implement the functionality required. Having a fast, simple method to optimize bounding boxes in d dimensional space has interesting implications for these applications. In the thesis we investigate initially a specific aspect of the problem, namely finding the minimum bounding boxes of three dimensional regular cross-polytopes. We present additional theorems on the nature of the projections of these bounding boxes in lower dimensions. We then show how some of these results can be generalized to arbitrary d-dimensional polytopes and provide additional theoretical results that can leveraged into a algorithm to find optimized bounding boxes in d-dimensional space. To illustrate the implications in terms of real world

applications we discuss range queries in high dimensional data bases using topological transformations based on the algorithm. We show how these transformations in conjunction with some interesting heuristics significantly improve performance over existing state of the art.

Chapter 2

Related Work

Using boxes as abstractions to represent an object or a region of space encompassing a given set of points is a common technique that has been used extensively in a variety of applications. The objective is usually to simplify the calculations involved and provide an improved method of describing the abstracted object/point's interaction with its environment.

Other methods use constructs like a prism [2], the simplex [3], sphere [4] or cones [5] to encapsulate points. However, these constructs do not lend themselves as well to optimized partitioning as axis aligned minimum volume bounding boxes. In most cases the applicability of the bounding abstractions in real world applications is limited compared to axis aligned boxes.

There exist several methods in literature that present ways to determine a box bounding of a given set of points. However, finding the exact minimum bounding boxes for set of points in d-dimensions is an extant problem that has proved difficult to solve in practice.

2.1 Finding Bounding Boxes in 2-Dimensions

The simplest method for finding the bounding box of a given set of points is to use the maximum range in each dimension to create an axis-aligned bounding box. The R-Tree [6], packed R-Tree [7], and R*-Trees [8] are common methods used in database indexing that utilize two dimensional versions of these measure for hierarchical partitioning. An optimal

solution to the problem does exist in two dimensions. This solution, known as the rotating callipers was proposed by Godfried Toussaint in [9] based on an idea first presented by [10]. This algorithm constructs the convex hull of the point set given, and checks to see which edge of the convex hull when coincident with an edge of a bounding box gives the smallest area. It executes in linear time $\vartheta(n)$ for the number of vertices/points and being simple to implement provides an effective solution to the problem.

2.2 Finding Bounding Boxes in 3-Dimensions

Given that most practical applications involve minimum bounding boxes for three dimensional objects, the greatest amount of significant work has been done on the three dimensional variant of the problem. The problem of finding minimal volume boxes circumscribing a given set of three-dimensional points was investigated by O'Rourke in [11]. This work demonstrated that for a three dimensional polyhedron defined by such a set of points, a minimum volume bounding box would necessarily have two faces flush with two adjacent edges of the enclosed polyhedron. It is the current best exact algorithm for the three dimensional problem, having a time complexity of $\vartheta(n^3)$ in the number of vertices/points of the convex hull. This algorithm is quite difficult to implement and as evinced by its time complexity slows down considerably as the number of vertices increases. [12] [13].

All remaining algorithms attempting to solve this problem use approximation techniques based on various heuristics. The primary objective in most cases is to speed up execution while maintaining an acceptable level of accuracy. According to [14], these approaches can classified into two main categories. The first one is based on brute force search using certain heuristics to minimize the search space. These heuristics can involve sampling the search

space based on a uniform distribution as in [13] or formulating the search as an optimization problem over the space of rotation matrices as in [14]. A combination of genetic algorithms is then used to minimize the space searched. These methods provide bounding boxes in three dimensions that can be quite accurate. However, the accuracy achieved is dependent on running the algorithms multiple times to converge to an optimal result. For a given execution, due to the randomness inherent to the genetic algorithms used the results produced can be sub-optimal. Other methods use geometrical characteristics such as coincidence of edges or faces of the convex hull with the faces of the bounding box as in [11], [15]. Another example of this class of algorithms is [16] which uses directions defined by pairs of points to indicate the two dimensional projection to be optimized for minimum bounding box. Variations of this method are further elaborated upon in [12] to obtain minimum bounding boxes with a given approximation factor. [17] gives the approximated bounding box in three dimensions by solving the associated 2D problem. However, the space optimization obtained is sub-optimal as it is based on the based on the assumption that solving the 2D problem in a single instance in $(SO)_3$ is sufficient to obtain the minimum bounding box . This limits the search space significantly causing it to miss the optimal orientation in most instances. The problem with most of these algorithms is the lack of accuracy (space optimization) that goes hand in hand with the speed up obtained. Furthermore, if the heuristic used is not accurate or good enough to sufficiently restrict the search space the gain in speed can be reduced and even reversed.

The second category involves algorithms using principle component analysis (PCA) to estimate the minimum bounding box. PCA is applied on the vertices to determine principle axes of the frame. All or some of these axes can then be used to build up a bounding box. Variants of this technique have been used in works like [18] [19]. However, it has been

shown in recent work [20] [21] [22] that the difference between the actual optimal and the calculated optimal in variants of this method could be infinitely large. Thus, not only would the global optimal be missed in most cases but the volume optimization obtained could be negligible.

2.3 Finding Bounding Boxes in d-Dimensions

In d-dimensions other than axis aligned boxes, the only other unique method used aimed at finding an optimized bounding box is presented in [16] [1] as a theoretical proof. It leverages a technique to estimate the diameter of the given points set first proposed in [23] and uses that as a constant factor to approximate the bounding box. This method is based on work originally done by [24]. However, approximating the diameter using this method provides a relatively rough metric for estimating the minimum bounding box with the result that the optimization obtained is diminished. It proposes a bound of $2^d d! Vol(B_{opt}(P))$ for the final volume of the bounding box where d is the number of dimensions and B_{opt} is the theoretical minimum bounding box of a d-dimensional polytope P. This being a very large bound it can result in significantly sub-optimal bounding boxes.

The minimized bounding boxes obtained from the the approach presented here can have important implications like medical imaging and treatment [25], multimedia [26], [27], [28] and computer graphics [29] [30] amongst others. For high dimensional multimedia databases, there exists a significant amount of work on querying and indexing techniques. As the optimized bounding boxes presented in this work have a direct application for carrying out queries on indexed high dimensional data bases we present an overview of existing work in the field.

2.4 Querying And Indexing Techniques in High Dimensions

A lot of work has been done on carrying out range queries and nearest neighbor queries using database indexes. k-NN (k-Nearest neighbor) queries can be considered a special case of range queries which only return the set of k records that are most similar to query record. A k-NN query can be implemented by keeping track of distance d_k of current k^{th} neighbor [31,32]. Any data node whose index bounding box is farther than d_k can be safely removed from consideration. Conceptually, this is equivalent to carrying out a range query with a range d_k. As the query is executed, the range decreases. [33, 34] present a detailed survey of these querying techniques.

There has been a lot of work on executing range queries and nearest neighbor queries using database indexes. A detailed discussion on these topics can be found in [35] [36]. Most of the existing work executes range queries by measuring distance of the query center from the minimum bounding rectangle (MBR) at each subtree and expanding a subtree only when certain distance criterion is satisfied [37] [6] [38] [39]. Other techniques utilize a distance metric distance or similar approximations integrated with the index structure [40] [41] [42] [43]. Box queries are executed by testing if the query box overlaps with the minimum bounding rectangle of a subtree in the index [44]. There a variety of high dimensional indexing techniques in dimensions that use a tree structure to organize the d-dimensional data. Techniques like the Box Tree [45] and the R-Tree [46] can be used to arrange the data into an index, which can then be used to return data matching specified range queries. [47] present algorithms to produce trees with small worst case complexity. Also a method is provided to convert Box Trees to R-Trees such that the query complexity is optimized. Other tree

based mechanisms include the STAR-Tree [48], the SS+-Tree [49], the HYBRID Tree [50], the X-Tree [51], G-Tree [52] and VA-File [53]. Any of these methods can used to carry range queries. The search space can be reduced whenever the distance between a given data page and the center of the query does not match the range criterion given. Some schemes try to incorporate distance metric (or other distance statistics) directly into the index structure itself. [54], [26] and [**?**, 55–57].

These methods remove nodes by first trying to determine the maximum distance of any data point in database being indexed from the from the subtree rooted at the node under consideration. The nodes not satisfying the distance metric are dropped.

In addition to indexing, query processing methods have also utilized transformation of data from one reference space to another to provide performance enhancement. An overview of these methods is provided in [58]. Those methods can be divided into two main categories, first one being those methods which map high-dimensional data down to lower dimensions before carrying out the range queries. These include techniques which map the d-dimensional data to one dimension using space filling curves such as z-curve [59] or hilbert curve [60]. For mapping to dimensions greater than one, [61–64] propose transformations that focus on reducing dimensionality in general rather targeting reduction to a specific dimension. These works use methods like Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) to achieve the dimension reduction thus avoiding the curse of dimensionality [65] and reducing redundancy.

The second category consists of methods which map polyhedrons or polygons to high dimensions. After the transformation has been affected then an established point access method such as the grid file [66] can be used to carry out the queries. The use of Z-curves to implement box queries has also been proposed in methods like [67, 68].

An interesting data space transformation based approach presented in [69, 70] as way to carry out approximate range queries and $k-$NN. This approach is based on the premise that the distances of similar data objects in the database from a specified set of reference points (reference objects) according to the specified distance metric will also be similar. Using this principle, the ranks of data objects can be used as an abstraction rather than storing the objects themselves. They discuss several heuristics to improve I/O and accuracy. This approach can used simultaneously with several different distance metrics such as Spearman Footrule distance, Kendall Tau distance and Spearman rho distance [70, 71]. The main drawback to these techniques is that they are sensitive to the statistical characteristics of the database and the determination of certain runtime parameters might require a data analysis affecting the complexity and accuracy. Additionally, the number of data objects needed for the determination of rank increases with increasing database size which the feasibility of using this method for dynamic databases.

Linear transformation in d-dimensional space is an integral part of linear algebra [72]. The application of these transformation with respect to their applications to database queries has not been extensively explored. To the best of our knowledge, other than the work in [73] ,there is no other work on mapping of range queries to box queries using data transformation. However, the transformation used there is inefficient in terms of the the amount of false positives in d-dimensions. The work focuses on using linear transformation within the framework of existing dynamic database indexes for improving page accesses for range queries. It also guarantees recall and scalability with increasing database size thus overcoming some of the disadvantages of prior work. The same set of transformations can have important implications for collision detection algorithms used in computer graphics to model object interaction in static and dynamic environments. An overview of existing work in this field

is provided in the next section.

2.4.1 Collision Detection Using Bounding Boxes

Interference and collision detection are fundamental problems in a variety of application domains, such as physically-based modeling, robotics, animation, computer-aided design, manufacturing, and computer simulated environments. CAD/CAM systems, for instance, use collision detection for clearance verification in an assembly [74], [75] and robot systems use collision detection for path planning [76] and, in computer graphics, collision detection works in conjunction with collision response to make animation appear more realistic and believable [77], [78] and [79].

Several different approximations for the objects within a given environment are possible [80], [79], [78]. However, due to the simplicity of calculation and shape, calculating intersecting bounding boxes is always more efficient than calculating intersecting objects. In addition to collision detection bounding boxes are also utilized for visible surface determination, re-projected pixel imaging [81] and view frustum culling [82]. [83] provides a good overview of some of the ways bounding boxes can be used in computer graphics. Modeling algorithms use bounding boxes to define complex shapes as combinations of simpler ones as well as to determine the minimum amount of space require to package components in an assembly [17]. Animation techniques used bounding boxes to approximate objects when simulating physical based motion [84]. However, he use of bounding boxes in collision detection is representative of its use in other algorithms [83]. The tightness of the bounding box used to approximate the underlying abstraction or object is an important performance constraint in these applications.

Extensive work has also been done on detecting interpenetration of objects in static and

dynamic environments using oriented bounding box hierarchies. OBB-Trees [19] were the first the propose to a hierarchical representation of models using tight fitting oriented bounding box trees. A runtime traversal of these trees tests for overlap between oriented bounding boxes in any two to determine collision. Other works [85], [86] [87] [88] [89] improve upon the OBB Trees presented here by using additional bounding volume abstractions and component decomposition respectively. Later works [90], [91]try to improve collision detection by improving upon the metrics and methods used to calculate the bounding box. However, the oriented bounding boxes used to approximate models in the constructed tree are sub-optimal in most cases. The standard method involves using PCA to find an orientation of bounding box of a model such that its volume is minimized. As discussed in [20] and [22] this technique while fast can result in bounding boxes that are significantly sub-optimal.

Most of these applications utilize the three dimensional variant of the problem, however [83] and [81] show that collision detection in R^d can also be carried out using d-dimensional bounding boxes. They provide bounds on the ratio of the box intersections and the object intersections. Furthermore, the bounds derived are shown be better applicable for convex polytopes in d-dimensional space. The minimization of the bounding boxes used to encapsulate these polytopes has a direct effect on the bounds proposed as tighter bounding boxes result in fewer intersections.

Chapter 3

Minimum Bounding Box of a Three Dimensional Cross-Polytope

The ability to minimize the time required for the determination of the minimum bounding box of a regular octahedron has significant implications for a variety of applications. Minimum bounding boxes in three dimensions are used to hierarchically partition a given set of points. Creating and maintaining these partitions is an integral part of applications like collision detection and scene rendering in computer graphics. Furthermore, statistical applications like storing and performing range queries on large databases utilize these constructs to a large extent to optimally partition and look-up data. We focus on a particular aspect of this problem, namely the determination of minimum volume bounding boxes for sets of points that define regular, uniform octahedrons in three dimensional space. Given that the set of points to be partitioned in time-sensitive applications like database range-search queries define octahedrons in three-dimensional space, optimizing minimum bounding box calculation for such constructs becomes rather important.

3.1 Two Edges Flush

The problem of finding minimal volume boxes circumscribing a given set of three-dimensional points was investigated by O'Rourke in [11]. This work demonstrated that for a three di-

mensional polyhedron defined by such a set of points, a minimum volume bounding box would necessarily have two faces flush with two adjacent edges of the enclosed polyhedron. Given this condition we state the following theorem for the special case where the convex hull of a set of points in three dimensional space defines a regular octahedron.

Theorem 1:

Every minimal volume bounding box of a set of points describing a regular uniform octahedron in three dimensional space must have at least two faces flush with two adjacent edges of the enclosed octahedron.

Proof:: The proof follows from O'Rourkes Theorem and is detailed in [Rourke84]. ■

Using this result we can develop an algorithm to determine the minimum volume bounding box for a regular octahedron. Such an algorithm would search through all possible combinations of adjacent edges in the octahedron, creating a bounding box for each pair, calculating its volume and isolating the one which has gives the smallest value. However, we can significantly improve upon this technique by leveraging one of the unique structural properties of a regular octahedron. This property is stated in the following lemma:

Lemma:*Any two adjacent edges of a uniform regular octahedron are separated by an angle of either 90 or 60 degrees*

This suggests that there can only be two possible unique combinations of adjacent edges with which the faces of a bounding box can be flushed. As the the octahedron is uniform, every other combination would be a reflection of these two. Utilizing this observation, we propose an additional necessary condition for the minimal volume bounding box of a regular octahedron that is significantly stricter than the one discussed above.

3.2 One Face Flush

In this section we further constrain the possible orientation of a minimum volume bounding box enclosing a cross polytope, by observing that for the special case of a regular octahedron the minimum volume bounding box would have one face flush with a face of the convex hull in addition to having two face flush with two adjacent edges of the enclosed octahedron.

Theorem 2:

A minimal volume bounding box must have at least one face flush with a face of the enclosed regular octahedron as well as having two flush with two adjacent edges.

Proof: We consider two cases, one in which the angle between adjacent edges is 90° and the other when the angle between the two is 60° as discussed earlier. Based on the geometry of the regular octahedron (the length of all edges being equal), all pairs of adjacent edges in the octahedron not bounding the same face of the octahedron, define three mutually orthogonal squares. Therefore, the angle between all such pairs of edges would be 90°. Conversely, in the case where two adjacent edges are at an angle of 60° , according to the geometry of the regular octahedron (the length of all edges being equal), every face of the octahedron is an equilateral triangle. Since the angle between two edges of such a triangle is always 60°, all pairs of adjacent edges bounding a face of the octahedron would have an angle of 60° between them.

3.2.1 Case - I: *Angle Between Adjacent Edges is 90 Degrees:*

In the first case, for simplicity we assume that the octahedron (or the 3D hyper-diamond) is rotated in such a way that one of the edges is flush with the X-Y plane and the other edge is flush with Z-X plane. Figure 1 shows the projection of the octahedron on the X-Y plane.

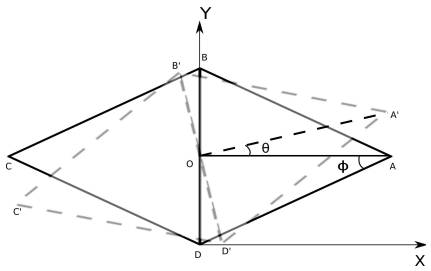

Figure 3.1: Projection of the Octahedron on the X-Y Plane

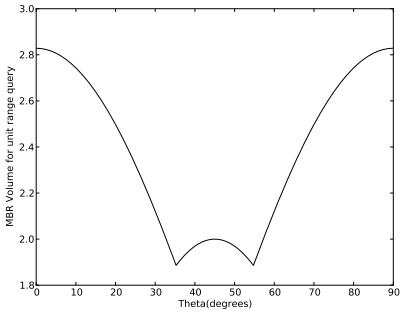

Figure 3.2: Volume of the MBR as a function of θ

Initially, the octahedron is positioned so as to have line OA parallel to the Z-X plane. Let θ be the angle by which the figure is rotated. As can be seen from the figure, the length l of the minimum volume bounding box for the octahedron is the projection of CA on to X-axis and its height h is equal to the projection of BD on to Y-axis. The breadth b of the box remains constant irrespective of θ. Hence, volume of the box can be calculated as V = lbh. As the polytope rotates Figure 2 shows volume of the MBR as a function of θ for $0 \leq \theta \leq 90$. Cases for other values of theta are similar to these with slight changes in the orientation, and are omitted hence. It can be seen that the function is discontinuous at two points which also happen to be the points of minimum volume. This is illustrated by the graph in fig. 2.

It can be shown using trigonometry that these two points correspond to the angles $\theta = \phi$ and $\theta = 90 - \phi, where \phi = 35.264\circ$. As the orientation of the cross polytope at each of these

two points ensures that two parallel faces of the octahedron are flush with the two parallel faces of the MBR, it follows that if the bounding box is oriented around any two adjacent edges with an angle of 90 between them, minimal volume can only be achieved when one of its faces is flush with a face of the octahedron.

3.2.2 Case - II: *Angle Between Adjacent Edges is 60 Degrees*

The starting condition is the same in this case. An axis-aligned bounding box is constructed with two adjacent edges of a regular octahedron having an angle of 60 between them flushed with two adjacent faces as shown in Fig.4. For simplicity we assume that this bounding box is placed at the origin with the octahedron oriented as shown in the figure (the face defined by vertices $V1, V2 and V3$ has the two edges with an angle of 60 between them).

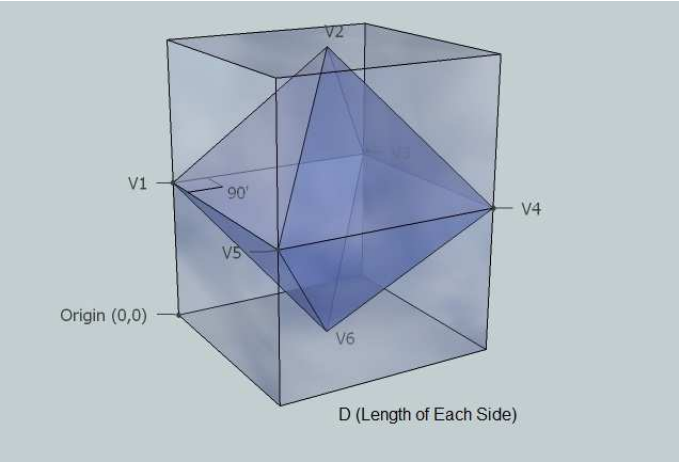

Figure 3.3: Position of Octahedron Against the Adjacent Faces of MBB for Case-I

Due to the constraint just mentioned (two edges need to be flushed with two adjacent faces), the only possible rotation of the cross-polytope will result in the movement of the vertice $V1$ on the Y-Axis, $V3$ on the Z-Axis and the edge $V1 - V2$ over the face of the bounding box. The distance from origin of this point is taken as a. Thus a varies over a finite range for the possible rotation.

In its initial position (as shown in Fig. 4), the three vertices coincident with faces of the bounding box are given by the following coordinates.

$V_1 = (0, a, 0)$

$V_2 = (x_2, y_2, 0)$

$V_3 = (0, 0, z_3)$

Based on the fixed distance D between the vertices V_1, V_2 and V_3,we derive the following equations,

$$D^2 = (x_2)^2 + (y_2 - a)^2$$

$$D^2 = (x_2)^2 + (y_2)^2 + (z_3)^2$$

$$D^2 = a^2 + (z_3)^2$$

where D is the length of a side of the cross-polytope.

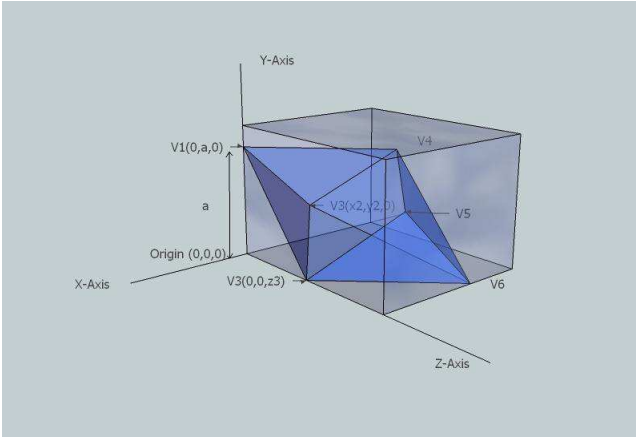

Figure 3.4: Position of Octahedron Against the Adjacent Faces of MBB for Case-II

Solving for x_2, y_2 and z_3, we can thus define each vertex in terms of a as given below,

$$V1 = (0, a, 0)$$

$$V2 = (\sqrt{a^2 - (\frac{2a^2 - D^2}{2a})^2}, \frac{2a^2 - D^2}{2a}, 0)$$

$$V3 = (0, 0, \sqrt{D^2 - a^2})$$

Using the fixed distances between the vertices of the cross-polytope, we can develop similar equations in terms of a and D. Solving for the variable a we obtain the following values for the coordinates of vertices $V4$,$V5$ and $V6$

$$V4 = (-(\frac{\sqrt{-a^2+d^2}(a^3-ad^2+\sqrt{2}\sqrt{-a^2(4a^4-5a^2d^2+d^4)})}{3(a^3-ad^2)}),$$
$$(\frac{\sqrt{4d^2-\frac{d^4}{a^2}}(4a^3-ad^2+\sqrt{2}\sqrt{-a^2(4a^4-5a^2d^2+d^4)})}{12a^3-3ad^2)},$$
$$\frac{(4a^3-ad^2-\sqrt{2}\sqrt{-4a^6+5a^4d^2-a^2d^4})}{3a^2})$$
$$V5 = (\frac{\sqrt{-a^2+d^2}(2a^3-2ad^2-\sqrt{2}\sqrt{-a^2(4a^4-5a^2d^2+d^4)})}{3(a^3-ad^2)},$$
$$\frac{2a^3+ad^2-2\sqrt{2}\sqrt{-4a^6+5a^4d^2-a^2d^4}}{6a^2},$$
$$\frac{4d^2-d^4/a^2\sqrt{(-4a^3+ad^2+2\sqrt{2}\sqrt{-a^2(4a^4-5a^2d^2+d^4)})}}{24a^3-6ad^2})$$
$$V6 = (\frac{\sqrt{-a^2+d^2}(2a^3-2ad^2-\sqrt{2}\sqrt{-a^2(4a^4-5a^2d^2+d^4)})}{3(a^3-ad^2)},$$
$$\frac{a^3-ad^2+\sqrt{2}\sqrt{-a^2(4a^4-5a^2d^2+d^4)}}{3a^2},$$
$$\frac{\sqrt{4d^2-d^4/a^2}(4a^3-ad^2+\sqrt{2}\sqrt{-a^2(4a^4-5a^2d^2+d^4]})}{12a^3-3ad^2})$$

Given that every coordinate of each vertex is a function of a, we can generalize the equation of a vertex of the cross polytope as,

$$V_i = (f_{xi}(a), f_{y_i}(a), f_{zi}(a)) \tag{3.1}$$

where i= 1,2,...,6 and f_{xi},f_{y_i} and f_{zi} are the functions defining the x,y and z coordinates respectively for each vertex.

The length of each dimension of the minimum bounding box is given by the minimum of the maximum values attained by the coordinates of each vertex. From above it is clear that the orientation of the octahedron is determined by the range over which a fluctuates. Based on the fact that V_1 has to remain on Y-Axis, the edge (V_1, V_2) on the XY-plane and assuming a unit octahedron for simplicity, we find the value of a has to be between $\frac{1}{\sqrt{2}}$ and $\frac{\sqrt{3}}{2}$ Utilizing basic trigonometry and the structural properties of the octahedron. The minimum of the maximum value attained by a coordinate in each dimension is thus given by,

$$Lengthof MBB_x = min_{1\leq i\leq 6}(max_{\frac{1}{\sqrt{2}}\leq a\leq\frac{\sqrt{3}}{2}}(f_{xi}(a))) \tag{3.2}$$

$$Lengthof MBB_y = min_{1\leq i\leq 6}(max_{\frac{1}{\sqrt{2}}\leq a\leq\frac{\sqrt{3}}{2}}(f_{y_i}(a)) \tag{3.3}$$

$$Lengthof MBB_Z = min_{1\leq i\leq 6}(max_{\frac{1}{\sqrt{2}}\leq a\leq\frac{\sqrt{3}}{2}}(f_{zi}(a))) \tag{3.4}$$

where i = *1,2,...6.*

To isolate the limits of the range of rotation of the polyhedron, we assume a unit polyhedron. Once the limits of the possible movement of the octahedron along its single degree of freedom are identified, specified in terms of the variable used as frame of reference i.e. a, we can obtain results for equations 2-4 above. This is done by finding first the local maxima and than the global minima of the coordinate equations for each vertex. After finding the local maxima for each dimension for every vertex we found that along the X-dimension only the equations for the x-coordinates of vertices $V2$ and $V6$ are differentiable and display

monotone convergence as functions of a over the given range. The x-coordinate equation for $V5$ is also differentiable, it retains a lesser range of values over the given range and thus is irrelevant when considering local maxima. While the function for the x-coordinate of $V2$ is monotonically increasing, the value of the function for the x-coordinate of the other vertex is monotonically decreasing. Thus, over the given interval the global minima would be point of convergence. The equations are,

$$x_2 = \sqrt{(a^2 - ((2a^2 - d^2)/(2a))^2)}$$

$$x_6 = \frac{\sqrt{-a^2 + d^2}(2a^3 - 2ad^2 - \sqrt{2}\sqrt{-a^2(4a^4 - 5a^2d^2 + d^4)})}{3(a^3 - ad^2)}$$

However for the y-coordinate equations, no such convergence occurs. Here we determine the local maxima for the y-coordinate of $V4$, providing a measure of the height of the minimum bounding box by the following equation.

$$y_4 = \frac{\sqrt{4d^2 - \frac{d^4}{a^2}}(4a^3 - ad^2 + \sqrt{2}\sqrt{-a^2(4a^4 - 5a^2d^2 + d^4)})}{12a^3 - 3ad^2)}$$

Similarly, the z coordinate equations for vertices $V4$ and $V3$ taken as a function of a are monotonically increasing and monotonically decreasing over the given range respectively. Thus, these functions are differentiable everywhere on the given interval allowing us to determine the global minima in the next step. Also, as these two functions converge to the same point over the given interval the global minima would be the point of convergence. Hence, we find that the length of the minimum bounding box in Z-dimension is given by the following equations.

$$z_4 = \frac{(4a^3 - ad^2 - \sqrt{2}\sqrt{-4a^6 + 5a^4d^2 - a^2d^4})}{3a^2}$$

$$z_3 = \sqrt{(d^2 - a^2)}$$

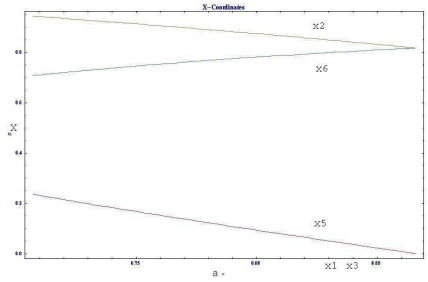

Figure 3.5: Local Maxima of X

The properties discussed above are easily verified by plotting the coordinate equations of each vertex as a function of a over the range of rotation identified as shown in figures 5-7. These three extremities give the minimum volume bounding box since $Volume of MBB = Length * Breadth * Height$. Minimizing the value of the x,y and z-coordinates over the given range, we find the orientation of the octahedron (given by the value of a at point of global minima) inside the minimum bounding box. This minimization results identifying the orientation with $a = \frac{\sqrt{3}}{2}$ as the point of global minimum. As we can see from Fig. 3, this orientation corresponds to the previous case where the angle $\theta = \phi$ and $\theta = 90 - \phi$, where $\phi = 35.264\circ$ and two parallel faces of the octahedron are flush with two parallel faces of the bounding box. Thus it follows that if the bounding box is oriented around any two adjacent edges with an angle of 60 degrees between them, minimal volume can only be achieved when one of its faces is flush with a face of the octahedron. ■

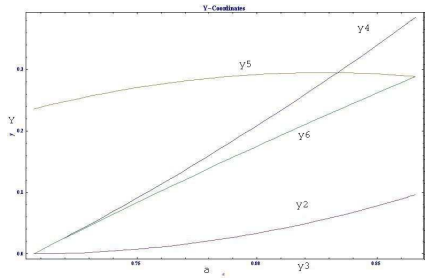

Figure 3.6: Local Maxima of Y

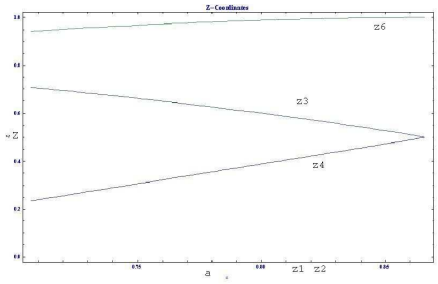

Figure 3.7: Local Maxima of Z

3.3 Bounding Box Under Projection

We now have a trivial method to determine the bounding box of a three dimensional cross polytope. In the following section we postulate about the characteristics of a three dimensional cross-polytope under projection when encapsulated with such a bounding box. The discussion is made with the assumption that the cross-polytope has already been optimally oriented such that its axis aligned bounding box is also the arbitrarily oriented minimum bounding box.

3.3.1 Minimality of Projected Bounding Box

If the bounding box of a three dimensional regular cross polytope is minimal, then we postulate about the nature of its projections in lower dimensions as follows,

Theorem 3:

Given an optimally oriented convex 3-polytope P and its axis-aligned minimum bounding box B, any projection $p : R^3 \rightarrow R^2$ of B will also be a minimum bounding box for the corresponding projection of P.

Proof: By way of contradiction, assume there exists a three dimensional cross polytope P and a rotation R which when applied to P results in its axis-aligned minimum bounding box B also becoming its minimum arbitrarily oriented minimum bounding box. Assume there exists a projection *p'* of P such that p' belongs to the space R^2 and the bounding box *b'* (corresponding to the projection of B along the same dimensions)is not an arbitrarily oriented minimum bounding box of the projection *p'* of P rotated by R. Then there exists an arbitrarily oriented minimum bounding box *b"* of the projection *p'* which is different from *b'*. Let R' be the rotation that rotates b'' such that it is an axis-aligned bounding box. The area of the bounding box is the product of the ranges in each dimension. Since projection discards dimensions not subject to the projection, any rotation of the projection does not change the dimensions of the axis not subject to projection for an axis aligned bounding box. However, there does now exist a rotation of the projection such that an axis aligned bounding box is smaller than b'. Hence, the composition of R and R' is a rotation that will rotate the cross-polytope P such that a smaller axis-aligned minimum bounding box exists. This contradicts our earlier assumption of the optimality of the three dimensional bounding box and hence is not possible. ■

3.4 Cross-Polytope under Projection

The theorems just discussed confirm that if the bounding box in three dimensions is optimal then the bounding boxes of the two dimensional projections are also optimal. This also leads us to a modified definition of the local optimality of the projection of a cross-polytope. This is stated as:

3.4.1 Definitions

Local Optimality: *If the axis-aligned bounding box of a two-dimensional projection of a 3-dimensional cross-polytope (where k = 1..2) is it's arbitrarily oriented minimum bounding box requiring no rotation of the projection to make it smaller, then the projection is said to be locally optimal.* For example, the diamond ABCD of fig.3.9 is the 2D-projection in XY

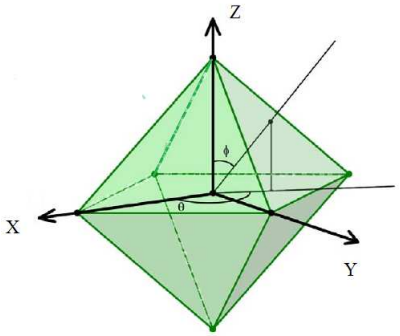

Figure 3.8: 3D-diamond in upright position

plane of the 3D-regular octahedron of 5.2. Projection ABCD is not locally optimal because the axis aligned bounding box for this projection is A'B'C'D' as shown by the dotted line in the figure, which is not the smallest bounding box of the projection. We get this optimal axis aligned bounding box by rotating the projection ABCD by an angle of 45°. Note that we are not rotating the 3D diamond, instead, we are just rotating the projection if it's bounding box is not already optimal.

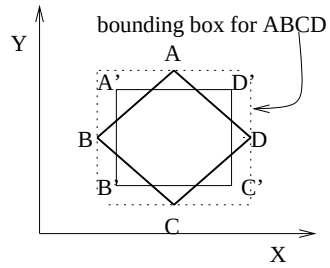

Figure 3.9: Projection of 3D-diamond on XY-plane

On the other hand, if the 3D-regular diamond itself is rotated from it's upright position of fig.5.2 by a 45°parallel to the XY- plane, the XY-projection of this newly oriented 3D-diamond is a square and bounding box of a square is the square itself. Thus, this projection is locally optimal. Using this definition we can further elaborate on the situations where multiple projections of the same cross-polytope exhibit local optimality at the same time. Thus:

Simultaneous Local Optimality: *If the two-dimensional projections of the* 3-dimensional *cross-polytope in each coordinate plane are locally optimal then it is said to have simultaneous locally optimal projections.*

From *Theorem 3*, we see that minimum bounding boxes of each two-dimensional projection of the 3-*dimensional* optimally oriented cross-polytope, corresponds to the two-dimensional projections of the minimum bounding box of the object in the corresponding planes. This, therefore, leads to the following corollary:

Corollary: *Given a convex three dimensional cross-polytope P, there exists at least one set of simultaneously locally optimal two- dimensional projections.*

3.4.2 Projected Polygons

In the case of a three dimensional cross-polytope, we can make additional statements about the nature of the projections of the polytope in two dimensions. The first of these statements is given the subsequent theorem,

Theorem 5:

Given a convex three dimensional cross-polytope P the convex hull of any projection $p : R^d \rightarrow R^{d_1}$, $1 < d_1 < 3$ of P will either be a hexagon, a rhombus or a rectangle.

Proof: The structural properties of a regular octahedron ensure that only two vertices of the polytope can be collinear at one time. Given this property, a vertex first projection onto one of the axial planes for any orientation of the octahedron would require that either one pair of vertices, two pairs of vertices or no pair of vertices be simultaneously coincident in the same planar projection.

Case - I: One Vertex Pair Coincident - The point formed from the projected pair would be bound by the convex hull of the remaining four and the planar projection obtained is rectangular.

Case - II: Two Vertex Pairs Coincident - The two pairs needs must lie on two parallel edges of the polytope thus forming two opposite points on the planar projection. The remaining two vertices are also opposite to each other and thus when projected onto the plane, form two more opposing points creating a rhombus.

Case - III: Three Vertex Pairs Coincident - Each vertice is projected as a point onto the plane forming a hexagonal envelope. ■

3.5 Minimal Bounding Box by Optimal Projections

Utilizing the unique characteristics of the two dimensional projections of a three-dimensional cross-polytope just discussed, a method can be developed to isolate the minimum bounding box of the cross-polytope by identifying the corresponding combination of locally optimal projections. The first step is to determine the constraints on the simultaneous local optimal projections possible, as is discussed in the subsequent section.

3.5.1 Unique Simultaneous Local Optimality

We can use the result of the theorem just discussed and Theorem-3 to define the optimal orientation of the three dimensional cross-polytope in terms of simultaneously locally optimal projections. This can be stated as:

Theorem 6:

Given a three dimensional cross-polytope P centered at the origin with each of its vertices lying on a coordinate axis, there exists only one combination of two dimensional projections in the coordinate planes that are simultaneously locally optimal, corresponding to the optimal orientation of the polytope.

Proof: **Case - I:** ***No Two Adjacent Edges Flush(Object Orientation is Sub-Optimal) :***

For this scenario we assume that no two adjacent edges of the object P are flush with two adjacent faces of the bounding box B. This implies that the orientation of P is sub-optimal and the bounding box B of P is not minimal. From *Theorem-5* we see that the convex hulls of the two dimensional projections in the coordinate planes are a combination of rhombuses, hexagons and rectangles. We now consider each projected shape in turn and determine the

conditions under which they become locally optimal.

Rhombus: Consider a projection *p'* of P on the XY-plane which forms a rhombus. For *p'* to be locally optimal, one of its edges must be coincident with one of the axes. However, we see that rotating p' to obtain the required orientation corresponds to rotating P such that one of its faces is coincident with a face of B. By Theorem-1 this corresponds to the optimal orientation of the object, contradicting our starting assumption that no two adjacent edges are flush with adjacent faces of B. Therefore, any combination of two dimensional projections containing a rhombus cannot be simultaneously locally optimal.

Hexagon: Consider a projection *q'* of of P on the XZ-plane which forms a hexagon. For *q'* to be locally optimal, at least two of its edges also need to be parallel with one of the axes and at least one must be coincident. Applying the rotation required to make the edges parallel, results in a corresponding change in the orientation of P such that the edges of P projected onto the XZ-plane to give the edges of q' being made parallel, become coincident with opposing faces of B. Due to the structure of P, any such pair of edges are joined with an adjacent edge that becomes coincident with an adjacent face of B contradicting our starting assumption. This precludes the possibility of a locally optimal hexagonal projection under the given conditions.

Rectangle: Proving the non-existence of simultaneous local optimality in all cases where the combination of projections on the coordinate planes contain a hexagon or a rhombus, leaves us with a single scenario where all of the projections are rectangles. A rectangular projection requires two pairs of adjacent edges forming a closed ring such that the angle between each pair is 90 degrees. Due to the structure of the octahedron, at no point in time can there be more than one such ring for any orientation of the polytope. Therefore, a combination of three rectangular projections is not possible.

Hence, there is no possibility of a simultaneously locally optimal combination of projections occurring when no two adjacent edges of the cross-polytope are flushed with the faces of the bounding box.

Case - II: ***Two Adjacent Edges Flush: Object Orientation May Be Optimal:*** As discussed earlier, any two adjacent edges of the polytope will have an angle of either 60° or 90° between them. Assume that the polytope is oriented such that two edges with an angle of 60° between them are flush with adjacent faces of the bounding box. A projection P of the object on the XY-plane forms a hexagon, given that no two vertices of the polytope have the same coordinates in the plane. For this hexagon to be locally optimal, it needs to have at least two edges parallel to one of the axes, with one being coincident to it. Since this is not the case here, it is not locally optimal. From the orientation of the polytope and from Theorem-1, the range of rotation of the polytope is limited such that for the given range, the projection of the polytope on the XY-plane remains a non-locally optimal hexagon except at the boundary where it becomes a rhombus. At this boundary the projection is optimal. For all other orientations it is sub-optimal, meaning that the combination of projections cannot be simultaneously locally optimal.

Now, for the case where two adjacent edges with an angle of 90° between them are flush with two adjacent faces of the bounding box. As we see from Fig.2, the projection of the polytope on the XY-plane is a rhombus for the entire range of possible rotations. For the rhombus to be locally optimal, one of its edges must be coincident with one of the axes. From the discussion above, this is only possible for the optimal orientation of the polytope. This precludes the existence of a simultaneous local optimal combination of projections for all possible orientations of the polytope under the given conditions (the rhombus is sub-optimal for all these orientations).

Therefore, we see that there can only exist a single unique combination of two projections that is simultaneously locally optimal. ■

3.5.2 Simultaneous Local Optimal By Combination Reduction

The uniqueness of the simultaneous local optimal projections possible for the three-dimensional cross-polytope combined with the limitations on the polygonal characteristics of those projections can now be leveraged into a process for isolating the combination of projections that correspond to the minimum bounding box.

From the previous theorem we know that there are three possible polygonal convex hulls for the two dimensional axial planar projections of the octahedron namely a hexagon (H), a rhombus (diamond)(D) and a rectangle(R). Since were only looking for a unique combination of projections that can occur simultaneously on the three coordinate planes, the order in which they occur is not important and we need only consider the possible combinations of these projections occurring simultaneously on the three coordinate planes xy,yz and zx. This leaves us with the following possible combinations that could correspond to the simultaneous local optimal,

HHH HHR HHD HRH HRR HRD HDD RRR RRD RDD DDD

Now, we try to remove all projections that are not possible due to the structural constraints of the octahedron. In the set of combinations just listed, there are several with multiple rectangular projections. However, if there are two rectangular axial planar projections, the projection in the third plane has to be rectangular too since the coordinate values it shares with the other planes would result in equivalent parallel edges in the plane. This is obviously not possible as a combination of three rectangular projections would result in a cube in three dimensions. Thus, removing all combinations with multiple rectangles:

HHH HHR HHD HRD HDD RDD DDD

For the octahedron to be optimally oriented, it must have at least two edges flush with two adjacent faces of the axis aligned minimum bounding box. This implies that the angle between any two dimensional projection of these edges would be 90° . Thus, the projected shape containing these two edges would need to have an internal angle of 90° w.r.t to the coordinate plane in which it exists.Hence, we only consider combinations with rectangular projections. Thus we have,

HHR HRD RDD

From Theorem 1, we know that if the three dimensional object is optimally oriented each projection has at least one edge flush with an edge of its axis aligned minimum bounding box. Given the structural constraints of an optimally oriented octahedron, if a diamond has one edge flush on one plane and the rectangle is locally optimal on another then the third projection cannot be a diamond without disturbing the optimality of the other two projections. Thus we are left with,

HHR HRD

Similarly, any orientation which has a hexagonal projection occurring simultaneously with a rectangle, such that both of them have at least one edge flush with one of the axes, cannot have a hexagonal projection in the third plane without compromising the optimality of the other two. This leaves only one combination of two dimensional projections that can correspond to an optimally oriented octahedron which is:

HRD.

This leads us to the following result,

Corollary: *Given a convex three dimensional uniform regular octahedron P, any simul-*

taneous locally optimal projections $p_i : R^d \rightarrow R^{d_1}$, $1 < d_1 < 3$, $0 < i < 4$ *of P will consist of a hexagon, a rectangle and a rhombus.*

Proof: The proof follows from *Theorems 5 and 6* and the process just discussed. ■

Thus the minimum bounding box of a three dimensional cross-polytope can be identified by projecting the polytope onto the coordinate planes and orienting the polytope such that the combination of those projection satisfy the constraints on the structure of their convex hulls.

3.6 Summary

To summarize, this chapter provides a discussion of methods to determine the minimum bounding box of a three dimensional cross polytope as well the discussing the characteristics of the two dimensional projections of these polytopes when encapsulated with a minimum bounding box. Identifying a minimum bounding box on the basis of face coincidence results in a significant speed up in the calculation of minimum volume bounded boxes. Given that three-dimensional boxes which enclose sets of points are used for maintaining hierarchical partitioning of sets of points. These data structures have important applications in computer graphics (e.g., for fast rendering of a scene or for collision detection) and statistics (for storing and performing range-search queries on a large database of samples). We focus on a particular aspect of this problem, namely the determination of minimum volume bounding boxes for sets of points that define regular octahedrons in three dimensional space. Given that the set of points to be partitioned in time-sensitive applications like database range-search queries define octahedrons in three-dimensional space, optimizing minimum bounding box calculation for such constructs becomes rather important.

In the next chapter we try to generalize some of the characteristics identified here for arbitrary n-dimensional polytopes. We try to postulate about the nature of the projections of such a polytope in d-k dimensions. We will then leverage those results to find a simultaneously locally optimal combination of projections corresponding to the optimal orientation of the polytope, such that finding the minimum bounding box of such polytopes becomes an exercise in finding a simultaneously locally optimal combination of projections in two dimensions. These results could have significant usage in statistical analysis, high dimensional databases and computational geometry.

Chapter 4

Optimized Bounding Box of Arbitrary D-Dimensional Polytope

Using minimal bounding boxes to encapsulate or approximate a set of points in d-dimensional space is a non-trivial problem that has applications in variety of fields including collision detection, object rendering, high dimensional databases and statistical analysis to name a few. While a significant amount of work has been done on the three dimensional variant of the problem (i.e. finding the minimum volume bounding box of a set of points in three dimensions), it is difficult to find a simple method to do the same for higher dimensions. Even in three dimensions existing methods suffer from either high time complexity or results which can be significantly suboptimal when approximation is used to speed up in execution time. In this chapter a new approach to find the optimized minimum bounding boxes of a set of points in d-dimensional space is presented. The solution also gives the optimal bounding box in three dimensions with a much simpler implementation while significantly speeding up the execution time for a large number of vertices. The proposed approach works by constructing the convex hulls of a given set of points and optimizing the projections of those hulls in two dimensional space using the new concept of *Simultaneous Local Optimal*.

Finding minimum bounding boxes in *d*- dimensional space is important for a variety of applications. The three dimensional variant or subset of the problem has greater utilization,

as illustrated by [14] which provides an excellent listing of the many applications of having an optimal method to find minimum bounding boxes in three dimensions. It mentions the importance of finding a solution to this problem in applications like querying and indexing high-dimensional databases [92], collision detection [93] [94] [95] [19], rendering scenes quickly [96]or mesh reparameterization [97]. In most of these cases the bounding box is used as an approximation of convex objects, as its regularity lends itself well to fast computations compared to an object's convex hull. While other bounding shapes can also be used, such as bounding spheres [4], bounding cylinders and cones [5] all of which have their own drawbacks and advantages [13], none lends itself as well to approximation or has the optimization advantages as a axis-aligned minimum bounding box.

Applications for optimized bounding boxes in higher dimensions are more sparse. However,recent work [98], illustrates the significant advantages to be obtained in querying and indexing high dimensional/multimedia databases by using a bounding box (in conjunction with some heuristics) to represent a range query in high dimensions, thereby reducing overlap and significantly improving upon the I/O cost. Initial experiments indicate that using optimized bounding boxes for such an application results in significant improvement in CPU time and I/O over standard range queries and the work mentioned.

Finding optimized bounding boxes for a set of points in a d-dimensional space has generally involved making an estimate of the diameter of the points provided in each dimension and using that estimate to create a tightly fitting axis aligned bounding box. Axis aligned bounding boxes (taking the maximum range in each dimension and using the points thus obtained to define the corners of the bounding box) provide the simplest solution to the problem. However, as is obvious they are significantly sub-optimal. While a solution to the three dimensional variant of the problem exists [11], the version that guarantees an optimal

bounding box is quite slow and non-trivial to implement. Other methods use heuristics to approximate an optimal bounding box while speeding up the execution time. In most, this unfortunately results in an appreciable decrease in the accuracy of the optimized bounding box obtained. The objective of the presented technique is to find a solution to the problem of finding the optimized minimum-volume oriented bounding box of a finite set of P points in d-dimensions defined by $P \subset \mathrm{R}^d$. In this chapter it is shown how the orientation of an optimal bounding box in d-dimensions can be computed by an optimization algorithm that minimizes the two dimensional projections of the convex hull of P. Using experimental results we also show empirically how, for the three dimensional variant of the problem, the optimized minimum bounding boxes obtained correspond to the optimal (minimum) bounding boxes.

In the subsequent section discussion on the optimality of the projections of a cross-polytope when encapsulated within a minimum bounding box discussed in the previous chapter is extended to the general case of an arbitrary d-dimensional polytope. It, therefore, also applies to the special case of arbitrary three dimensional polytopes and their bounding boxes. The discussion is made with the assumption that the polytope has already been optimally oriented such that its axis aligned bounding box is also the arbitrarily oriented minimum bounding box. We also provide certain definitions based on that result which we utilize in subsequent sections. We then define the formulation of the problem in terms of the optimization of two dimensional projections. Our Simultaneous Local Optimal Algorithm is then defined and described. In the final section we discuss its implementation and comparison with existing methods. We conclude by discussing further optimizations and possible applications that show promise based on initial investigation.

4.1 Minimizing Bounding Box Based On Locally Optimal Projections

In the following section the theorems postulated discuss the characteristics of a polytope under projection when encapsulated within a minimum bounding box . The result presented is applicable to arbitrary d-dimensional polytopes and their bounding boxes. The discussion is made with the assumption that the polytope has already been optimally oriented such that its axis aligned bounding box is also the arbitrarily oriented minimum bounding box. In order to facilitate this discussion, the following definitions and axioms are provided to illustrate the primary concepts involved.

4.1.1 Definitions

Definition 4.1.1. *Tightly Fitting Bounding Box:* A Tightly Fitting Bounding Box (TFBB) of a d-dimensional polytope P is defined as having at least one vertex of P coincident with each of its faces.

Definition 4.1.2. *Axis Aligned Bounding Box:*An Axis Aligned Bounding Box (AABB) of a d-dimensional polytope P is defined as the box whose body diagonal is delimited by the minimum and maximum axial coordinate values of P.

Definition 4.1.3. *Oriented Bounding Box:*An Oriented Bounding Box (OBB) of a d-dimensional polytope P is a tightly fitting rectangular bounding box of arbitrary orientation in d-dimensional space

Definition 4.1.4. *Orientation in d-Dimensions:*Given an arbitrary polytope P in d-dimensions, all orientations of P can be obtained by rotating P parallel to one or more of the axial planes

(vertex coordinates for all axes involved in the rotation remaining fixed) in d-dimensional space.

Definition 4.1.5. *Local Optimal:* For a given k, $0 \leq k \leq$ d-1, if the axis-aligned bounding box of a d-k dimensional projection of a d-dimensional polytope is its arbitrarily oriented minimum bounding box (requiring no rotation of the projection to make it smaller), then the projection is said to be local optimal.

Using this definition we can further elaborate on the situations where multiple projections of the same polytope exhibit local optimality at the same time. Thus:

Definition 4.1.6. *Simultaneously Local Optimal:* For a given k, $0 \leq k \leq$ d-1, if the *d-k* dimensional projections of the *d-dimensional*-polytope in each d-k dimensional plane are locally optimal then the projections are simultaneously local optimal.

4.1.2 Minimality of Projected Bounding Box

If the bounding box of a d-dimensional polytope is minimal, then we postulate about the nature of its projections in lower dimensions by the following theorem.

Theorem 4.1.1. *Given an optimally oriented convex d-polytope P and a given k, $0 \leq k \leq d-1$, projections of the object P in d-k dimensional subspaces are simultaneously local optimal.*

Proof: By way of contradiction, assume there exists a d-polytope P and rotation R such that an axis-aligned minimum bounding box B of P is also its minimum arbitrarily oriented minimum bounding box. Assume that the projections of the object P, rotated by R, in all (permutations) of $d-k$ dimensions for a given k, $0 \leq k \leq d-1$, are not simultaneously local optimal. Thus, there exists a projection p' of P in $d-k$ dimensions rotated by R such that

the bounding box b' (corresponding to the projection of B along the same $d-k$ dimensions)is not an arbitrarily oriented minimum bounding box of the projection p' of P rotated by R. Then there exists an arbitrarily oriented minimum bounding box b" of the projection p' of P rotated by R which is different from b'. Let R' be the rotation that rotates b'' such that it is an axis-aligned minimum bounding box. The volume of the bounding box is the product of the ranges in each dimension. Since projection discards dimensions not subject to the projection, any rotation of the projection does not change the range values on the axes in the orthogonal dimensions that are not subject to projection for an axis aligned bounding box. However, there does now exist a rotation of the projection such that an axis aligned bounding box for p' is smaller than b'. Hence, the composition of R and R' is a rotation that will rotate the d-polytope P such that a smaller axis-aligned minimum bounding box exists. This contradicts our earlier assumption of the optimality of the bounding box of the d-polytope. Hence, the projections of P in $d-k$ dimensions are simultaneously locally optimal. □

The necessity of an optimally oriented d-dimensional polytope having simultaneously locally optimal projections can be leveraged to further elucidate and identify the characteristics and conditions of the polytope in that orientation as discussed subsequently.

4.1.3 Edges Flush in d-Dimensions

Using the result just obtained we can come up with a necessary condition for the minimum bounding box of an n-dimensional cross polytope. This condition is stated by the following theorem:

Theorem 4.1.2. *Given a convex d-dimensional polytope P and an arbitrarily aligned min-*

imum bounding box B, at least d-1 edges must be flush with d-1 orthogonal faces of B.

Proof: Without loss of generality, assume P and B are rotated such that B is an axis-aligned minimum bounding box. By the previous theorem, any $2d$ projection of B must also be a minimum bounding box of the $2d$ projection of P. By a previous theorem [9], at least one edge of the $2d$ projection of P must be flush with one edge of the $2d$ projection of B. Such an edge will have identical values for each of the dimensions corresponding to the bounding box edge and the values must be either the minimum or maximum values of that dimension. Otherwise the bounding box would not be minimum since some value in one dimension is beyond the range of the bounding box. An edge that is flush with the edge of the bounding box in a $2d$ projection is also flush with the corresponding face of B, since the values in that dimension are equal for both edges of the edge and are the minimum or maximum value in that dimension. Hence, for every possible pair of dimensions an edge must exist that is flush with a face of B for one of those dimensions. It is possible for an edge to serve as the flush edge for more than one $2d$ projection involving a given dimension d. The edge will always have the minimum or maximum value for dimension d for every 2d projection that includes that dimension. The minimum number of edges that can satisfy this constraint for all possible pairs of dimensions is d. Assume it is less than d, then there exist two dimensions for which a flush edge does not exist. Hence, the 2d projection would not have a flush edge and we have a contradiction. □

4.1.4 Existence of Simultaneous Local Optimal For All d-Dimensional Polytopes

Theorem 4.1.1 confirms that if the bounding box in dimension d is optimal then the bounding boxes of the projections in lower dimensions are also optimal. From Theorem 4.1.2, we also see that minimum bounding boxes of each *d-k* dimensional projection of the *d-dimensional* optimally oriented object, correspond to the *d-k* dimensional projections of the minimum bounding box of the object in the corresponding planes. This, therefore, leads to the following corollary:

Corollary. Given a convex d-polytope P, and a given k, $0 \leq k \leq$ d-1, there exists at least one combination of simultaneously local optimal d-k dimensional projections.

Proof: This follows from 4.1.1, since the d-k dimensional projections corresponding to the global optimal orientation of the convex d-polytope P are simultaneously locally optimal and there exists a global optimal orientation for P such that the volume of its bounding box is minimized there exists at least one combination of simultaneously local $d-k$ dimensional projections for P. □

4.1.5 Minimality of D-Dimensional Bounding Box

We leverage the result of the 4.1.1 to define the optimal orientation of a d-dimensional polytope in terms of simultaneously local optimal projections in two dimensions. This can be described by the following proposition:

Proposition 4.1.3. Given a d-dimensional polytope P and a given k, $0 \leq k \leq d-1 \mid d-k = 2$, for all orientations of P having its $d-k$ projections in the axial planes simultaneously local

optimal, the bounding boxes of P defined by bounding boxes of its locally optimal projections form the candidate set for the global minimum volume bounding box.

Proof. By $Theorem1$, if the object's orientation is globally optimal then projections of the object on the axial planes are simultaneously local optimal. This implies that any orientation of P with simultaneous local optimal projections has the possibility of corresponding to the global optimal. Thus, those orientations of the object whose projections are simultaneously locally optimal are candidates for the global optimal orientation. □

This proposition implies that if a method could be found to isolate all orientations of an arbitrary d-dimensional polytope that satisfy the simultaneous local optimal condition in its projections, the orientation corresponding to the bounding box with the minimal volume (the bounding box of each orientation being constructed from the bounding boxes of the projections) would be the global optimal. With that in mind we show how starting from any orientation of a d-dimensional polytope, the $d-k$ axial projections can be optimized to converge to a simultaneous local optimal.

4.1.6 Convergence of Projections

Theorem 4.1.4. *Given an d-dimensional polytope* $\boldsymbol{P}$ *and its k dimensional parallel axial projections* $p_i : R^d \rightarrow R^k$, $1 < k < d$, $1 \leq i \leq m$ *where* $m = C\binom{d}{k}$, *orienting* $\boldsymbol{P}$ *to minimize volume* $\boldsymbol{V}$ *of the bounding box* $\boldsymbol{B}$ *by rotating it successively around each k-dimensional projection according to the projected minimum in that subspace, causes these projections to converge to a simultaneous local optimal.*

Proof: Let B be the volume of an axis-aligned bounding box (AABB) of the convex hull of

a given set of points defining a polytope in d-dimensional space, where

$$B \geq B_{opt} > 0$$

and B_{opt} is volume of the minimum volume bounding box of $\mathbf{P}$ or in other words the volume of $\mathbf{B}$ at the optimal orientation of $\mathbf{P}$. Assume that the polytope P is projected into m k-dimensional contiguous subspaces bound by the axes of the dimensions involved in the projection where $m = C\binom{d}{k}$ [99]. Given that $\mathbf{P}$ can be rotated around a given k-dimensional subspace, let the volume of the bounding box of the i-th k-dimensional projection of $\mathbf{P}$ be, ${V^j}_i$ where $1 \leq i \leq m$ and $m = C\binom{d}{k}$ [99] at rotation j and,

$$B^j = {V^j}_i * \prod_{p \in (d-k_i)} l_p \tag{4.1}$$

where B^j is the volume of the bounding box of $\mathbf{P}$ after rotation j and k_i defines the set of the lengths of the dimensions of the bounding box involved in projection i. A rotation is defined as rotating $\mathbf{P}$ around a single k-dimensional subspace such that the bounding box of the projection of $\mathbf{P}$ is minimized. For example,${V^0}_1$ is the volume of the bounding box of the first k-dimensional projection at rotation 0 i.e. at the starting orientation of $\mathbf{P}$. The volume of the bounding box of the projection in the contiguous subspace 2 would be ${V^0}_1$. Similarly, ${V^1}_1$ would indicate the volume of the bounding box of the projection in subspace 1 after the first rotation and so on. One iteration consists of successively rotating in each k-dimensional subspace once. Thus, one iteration consists of m rotations. Thus, ${V^1}_2$ would indicate the volume of the bounding box of the projection in subspace 2 around which $\mathbf{P}$ is rotated in the subsequent rotation.

Projecting $\mathbf{P}$ onto the first k-dimensional subspace and rotating the projection by a

rotation r^1 such that it is minimized to its local optimal,

$$
\begin{aligned}
&(V_{1opt} = {V^1}_1 \leq {V^0}_1) \\
&\Rightarrow 0 < {V^1}_1 \leq {V^0}_1 \\
&\Rightarrow 0 < ({V^1}_1 / {V^0}_1) \leq 1
\end{aligned}
$$

Applying this rotation back to $\mathbf{P}$ results in changing the volume of $\mathbf{B}$. This is obvious since rotating around the k-dimensional subspace has no effect in the remaining $d-k$ dimensions which retain their original values. Thus volume of the bounding box after the first rotation would be given by,

$$
\begin{aligned}
B^1 &= {V^1}_1 * \prod_{p \in (d-k_1)} l_p \\
\Rightarrow B^1 &= \frac{{V^1}_1}{{V^0}_1} B^0 \because \prod_{p \in (d-k_1)} l_p = \frac{B^0}{{V^0}_1}
\end{aligned}
$$

given that the lengths of the $d-k_1$ remaining dimensions are unchanged.Projecting $\mathbf{P}$ in the next k-dimensional subspace in sequence and rotating it to make it locally optimal,

$$
\begin{aligned}
&(V_{2opt} = {V^2}_2 \leq {V^1}_2) \\
&\Rightarrow 0 < {V^2}_2 \leq {V^1}_2 \\
&\Rightarrow 0 < ({V^2}_2 / {V^1}_2) \leq 1
\end{aligned}
$$

and the corresponding volume of the bounding box would be,

$$B^2 = {V^2}_2 * \prod_{p \epsilon (d-k_2)} l_p$$
$$\Rightarrow B^2 = \frac{{V^2}_2}{{V^1}_2} B^1 \because \prod_{p \epsilon d-k_2} l_p = \frac{B^1}{{V^1}_2}$$
$$\Rightarrow B^2 = \frac{{V^2}_2}{{V^1}_2} \frac{{V^1}_1}{{V^0}_1} B^0 \because B^1 = \frac{{V^1}_1}{{V^0}_1} B^0$$

Successively rotating through each k-dimensional subspace, minimizing in that subspace only while letting the other dimensions remain unchanged, we can generalize the inequality defining the local optimal for the m-th rotation as,

$$0 < \frac{{V^m}_m}{{V^{m-1}}_m} \leq 1 \tag{4.2}$$

and define the volume of the bounding box of $\mathbf{P}$ after m rotations or one complete sequence of rotations around each k-dimensional subspace by the following equation,

$$B^m = \frac{{V^m}_m}{{V^{m-1}}_m} B^{m-1}$$

From **??** and 4.1.6 it's also obvious that the B^m can also be defined in terms of the volume B^0 of $\mathbf{P}$ at the starting orientation. This is done by successively composing the volume of $\mathbf{P}$ at each orientation at each rotation j in terms of rotation $j-1$. Thus,

$$B^m = \frac{{V^1}_1}{{V^0}_1} * \frac{{V2}_2}{{V^1}_2} * ..... * \frac{{V^m}_m}{{V^{m-1}}_m} B^0 \tag{4.3}$$

where,

$$0 < \frac{V^1{}_1}{V^0{}_1} * \frac{V2_2}{V^1{}_2} * ..... * \frac{V^m{}_m}{V^{m-1}{}_m} \leq 1$$

from 4.2.

This can also be generalized for a random number of rotations j to,

$$0 < \frac{V^j{}_i}{V^{j-1}{}_i} \leq 1$$

and,

$$B^j = \frac{V^j{}_i}{V^{j-1}{}_i} B^{j-1}$$

and $q = 1, 2, ...$ Furthermore, by definition each $V^j{}_i$ is the local optimal for projection i at the corresponding orientation defined by rotation j.

Also,

$$B^j = \frac{V^1{}_1}{V^0{}_1} * \frac{V2_2}{V^1{}_2} * ..... * \frac{V^j{}_i}{V^{j-1}{}_i} B^0 \tag{4.4}$$

where,

$$0 < \frac{V^1{}_1}{V^0{}_1} * \frac{V2_2}{V^1{}_2} * ..... * \frac{V^j{}_i}{V^{j-1}{}_i} \leq 1$$

from 4.2.

where $i = \begin{cases} j & j \leq m \\ j - qm & j > qm \end{cases}$

Repeating the above process for a finite number of rotations n, eq. 4.4 can be generalized

to,

$$B^n = \frac{V^{n-m+1}{}_1}{V^{n-m}{}_1} * \frac{V^n - m + 2_2}{V^{n-m+1}{}_2} * ..... * \frac{V^n{}_m}{V^{n-1}{}_m} B^{n-m} \tag{4.5}$$

$$\Rightarrow B^n = \frac{\prod_{s=1,r=n-m+1}^{m,n} V^r{}_s}{\prod_{s=1,r=n-m}^{m,n-1} V^{r-1}{}_s} B^{n-m} \tag{4.6}$$

and,

$$0 < \frac{\prod_{s=1,r=n-m+1}^{m,n} V^r{}_s}{\prod_{s=1,r=n-m}^{m,n-1} V^{r-1}{}_s} \leq 1$$

Assume,

$$\frac{\prod_{s=1,r=n-m+1}^{m,n} V^r{}_s}{\prod_{s=1,r=n-m}^{m,n-1} V^{r-1}{}_s} = \mathbf{V}_n$$

Thus, the above equation becomes,

$$B^n = \mathbf{V}_n B^{n-m} \tag{4.7}$$

In the above equation $\mathbf{V}_n = 1$ only if each $V^j{}_i = V^{j-1}{}_i$ for a given subspace i, i.e. the volume of the projections in each subspace does not change after a finite number of rotations $n - m$. Otherwise, from the inequality above $0 < \mathbf{V}_n < 1$. Therefore,

$$\begin{cases} B^n = B^{n-m} & if \mathbf{V}_n = 1 \\ B^n \neq B^{n-m} & if 0 < \mathbf{V}_n < 1 \end{cases}$$

Thus, the volume of a bounding box converges to a local optimal corresponding to a simultaneous local optimal of its projections only if $\mathbf{V}_n = 1$. If there does not exist a value of n such that this condition holds true then it is always the case that $0 < \mathbf{V}_n < 1$ and the volume does not converge. For an infinite number of rotations this would mean that'

$$\begin{aligned}
\lim_{n\to\infty} B^n &= \lim_{n\to\infty} (\mathbf{V}_n B^{n-m}) \\
&= \lim_{n\to\infty} (\mathbf{V}_n \mathbf{V}_{n-1} \mathbf{V}_{n-2} \ldots\ldots \mathbf{V}_1 B^0) \\
&= B^0 \lim_{n\to\infty} (\mathbf{V}_n \mathbf{V}_{n-1} \mathbf{V}_{n-2} \ldots\ldots \mathbf{V}_1) \\
&= B^0 * 0 = 0
\end{aligned}$$

This contradicts our starting assumption that $B \geq B_{opt} > 0$, which is not possible. Hence, there exists a finite number of rotations j such that $\mathbf{V}_n = 1$ for $n > j$ and the volume of the bounding box of each k-dimensional subspace converges to a local optimal. □

In practice minimizing k-dimensional projections by rotating around the corresponding k-dimensional subspace is non-trivial for $k > 3$. For $k \leq 3$, algorithms exist that can be used to find the exact minimum bounding boxes of projections in the lower dimensions. However, the only algorithm that guarantees an exact minimum bounding in three dimensions is quite difficult to implement as well as being quite slow for a large number of vertices. Minimization in two dimensions, in contrast, is relatively simple to implement and quite fast. Furthermore, any d-dimensional rotation can be decomposed to two dimensional planar rotations lending itself well to generalization. Thus for the specific case of projections where $k = 2$, it can be stated that,

Corollary. *Given an d-dimensional polytope* $\boldsymbol{P}$ *and its* 2*-dimensional parallel axial projec-*

tions $p_i : R^d \to R^2$, , $1 \leq i \leq m$ *where* $m = C\binom{d}{2}$, *orienting* $\boldsymbol{P}$ *to minimize volume* $\boldsymbol{V}$ *of the bounding box* $\boldsymbol{B}$ *by rotating it successively around each* 2*-dimensional projection according to the projected minimum in that plane, causes these projections to converge to a simultaneous local optimal.*

Proof. The proof follows from the 4.1.4 for the case where $k = 2$. □

In the next section we show how combining the result of the above theorem for the special case of $k = 2$ with the earlier proofs, results in a simple method for finding minimum volume bounding boxes in d-dimensions. Prior to this we discuss some additional characteristics of the case where $k = 2$, that can be leveraged to optimize the algorithm proposed.

4.1.7 Period of Projected Bounding Box in 2D

For a given 2-dimensional projection, rotating it a round its center results in changing the dimensions and area of its axis-aligned bounding box. This change, in the case of the lengths of the bounding box, is quantified by the following theorem

Theorem 4.1.5. *Given an arbitrary two-dimensional parallel axial projection* p_i *of d- dimensional polytope* $\boldsymbol{P}$ *such that* $p_i : R^d \to R^2$, $1 \leq i \leq m$ *where* $m = C\binom{d}{2}$, *and its corresponding* 2*-dimensional bounding box* B_i *having length* l_i *and breadth* b_i, *rotating* B_i *around its center to minimize its area results in the values of* l_i *and* b_i *repeating regularly by a period of* π.

Proof: Let X, Y be the set of 2-dimensional vertices defining an arbitrary 2-dimensional projection with (a, b) the center of its axis-aligned bounding box. The values of length l_i and breadth b_i are the difference between the maximum and minimum values of their corresponding axial coordinates. This also implies that the value of l_i and b_i would be the

same irrespective of where the center (a,b) is. Hence, the projection can be rotated around the origin instead of (a,b) without affecting the changing values of l_i and b_i. Thus,

$$l_i = Y_{max} - Y_{min} = (Y - b)_{max} - (Y - b)_{min} \tag{4.8}$$

$$b_i = X_{max} - X_{min} = (X - a)_{max} - (X - a)_{min} \tag{4.9}$$

Since we are now rotating the projection around the origin, this rotation for an angle θ can now be given by,

$$\begin{aligned} X', Y' &= X, Y \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}, \\ &= Xcos\theta - Ysin\theta, Xsin\theta + Ycos\theta, \end{aligned}$$

where X', Y' are the set of points of the new projection after rotation. Let $l_i\theta$ and $b_i\theta$ be the functions of θ, representing the value of l_i and b_i respectively after rotation θ. Then,

$$\begin{aligned} l_i(\theta) &= Y'_{max} - Y'_{min} \\ &= (Xsin\theta + Ycos\theta)_{max} - (Xsin\theta + Ycos\theta)_{min} \\ b_i(\theta) &= X'_{max} - X'_{min} \\ &= (Xcos\theta - Ysin\theta)_{max} - (Xcos\theta - Ysin\theta)_{min} \end{aligned}$$

$$
\begin{aligned}
l_i(\theta+\pi) &= (Xsin(\theta+\pi)+Ycos(\theta+\pi)0_{max} - (Xsin(\theta+\pi)+Ycos(\theta+\pi))_{min} \\
&= (-Xsin\theta - Ycos\theta)_{max} - (-Xsin\theta - Ycos\theta)_{min} \\
&= (-(Xsin\theta + Ycos\theta))_{max} - (-(Xsin\theta + Ycos\theta))_{min} \\
&= -(Xsin\theta + Ycos\theta)_{min} + (Xsin\theta + Ycos\theta)_{max} \\
&= (Xsin\theta + Ycos\theta)_{max} - (Xsin\theta + Ycos\theta)_{min} \\
&= l_i(\theta)
\end{aligned}
$$

Similarly, for b_i,

$$
\begin{aligned}
b_i(\theta+\pi) &= (Xcos(\theta+\pi) - Ysin(\theta+\pi))_{max} - (Xcos(\theta+\pi) - Ysin(\theta+\pi))_{min} \\
&= (-Xcos\theta - Ysin\theta)_{max} - (-Xcos\theta - Ysin\theta)_{min} \\
&= (-(Xcos\theta - Ysin\theta))_{max} - (-(Xcos\theta - Ysin\theta))_{min} \\
&= -(Xcos\theta - Ysin\theta)_{min} + (Xcos\theta - Ysin\theta)_{max} \\
&= (Xcos\theta - Ysin\theta)_{max} - (Xcos\theta - Ysin\theta)_{min} \\
&= b_i(\theta)
\end{aligned}
$$

Hence proved. □

Based on the result above, we can also postulate about the period with which the length of the bounding box in a given projection plane lags behind the breadth, when rotating **P** around that plane.

Theorem 4.1.6. *Given an arbitrary two-dimensional parallel axial projection p_i of d- dimen-*

sional polytope $\boldsymbol{P}$ *such that* $p_i : R^d \to R^2$, $1 \le i \le m$ *where* $m = C\binom{d}{2}$, *and its corresponding* 2*-dimensional bounding box* B_i *having length* l_i *and breadth* b_i, *rotating* B_i *around its center to minimize its area results in the values of* b_i *will lag behind the value of* l_i *by a period of* $\pi/2$.

Proof:

$$\begin{aligned} l_i(\theta + \frac{\pi}{2})) &= (X sin(\theta + \frac{\pi}{2}) + Y cos(\theta + \frac{\pi}{2}))_{max} - (X sin(\theta + \frac{\pi}{2}) + Y cos(\theta + \frac{\pi}{2}))_{min} \\ &= (X cos\theta - Y sin\theta)_{max} - (X cos\theta - Y sin\theta)_{min} \\ &= b_i(\theta) \end{aligned}$$

By substituting $\theta = \varphi - \frac{\pi}{2}$,we get

$l_i(\varphi) = b_i(\varphi - \frac{\pi}{2})$

□

Combining these two results, it can be determined that the area of the bounding box of a given projection p_i also repeats in a regular manner, as $\mathbf{P}$ is rotated around plane i. This can be stated formally as,

Theorem 4.1.7. Theorem*:*

Given an arbitrary two-dimensional parallel axial projection p_i *of d-dimensional polytope* $\boldsymbol{P}$ *such that* $p_i : R^d \to R^2$, $1 \le i \le m$ *where* $m = C\binom{d}{2}$, *and its corresponding* 2*-dimensional bounding box* B_i *having length* l_i *and breadth* b_i, *rotating* B_i *around its center to minimize its area results in the area repeating regularly by a period of* $\frac{\pi}{2}$.

Proof:

$$
\begin{aligned}
A(\theta) &= l_i(\theta)b_i(\theta) \\
A(\theta + \frac{\pi}{2})) &= l_i(\theta + \frac{\pi}{2}))b_i[(\theta + \pi) - \frac{\pi}{2}] \\
&= l_i[(\theta + \pi)b_i(\theta) \\
&= l_i(\theta)b_i(\theta)
\end{aligned}
$$

□

Since the bounding box of p_i corresponds to the projection of the bounding box of $\mathbf{P}$ on plane i, the the result presented above provides an interesting insight into the repeatability of the bounding box volume of $\mathbf{P}$ given by the following corollary,

Corollary. *Given an arbitrary d-dimensional polytope $\boldsymbol{P}$,its axis aligned bounding box $\boldsymbol{B}$ and a given projection p_i of $\boldsymbol{P}$ such that $p_i : R^d \rightarrow R^2$, rotating $\boldsymbol{P}$ in the two dimensional axial plane of the projection results in the volume of $\boldsymbol{B}$ repeating by a period of $\frac{\pi}{2}$*

.

Proof. The volume of the bounding box of the polytope is given by,

$$B = A_i * \prod_{j \epsilon (d - dp_i)} l_j$$

where l_j defines the set of the lengths of the dimensions not involved in projection p_i and A_i is the area of the bounding box of p_i. Rotating $\mathbf{P}$ by an angle θ in the plane of p_i corresponds

to rotating p_i by θ, the volume of $\mathbf{B}$ is given by,

$$B(\theta) = A_i(\theta) * \prod_{j \in (d - d_{p_i})} l_j$$

and,

$$B(\theta + \frac{\pi}{2}) = A_i(\theta + \frac{\pi}{2}) * \prod_{j \in (d - d_{p_i})} l_j$$

From 4.1.7, $A_i(\theta + \frac{\pi}{2}) = A_i(\theta)$, thus

$$B(\theta + \frac{\pi}{2}) = A_i(\theta) * \prod_{j \in (d - d_{p_i})} l_j$$

$$B(\theta + \frac{\pi}{2}) = B(\theta)$$

□

All the results discussed in this section point towards to a method to isolate optimized bounding for polytopes in d-dimensions based optimizing their projections in k dimensions where $k = 1, 2, ..., d-1$. Optimizing projections in two dimensions offers the simplest method to leverage these results into a viable optimization algorithm in d-dimensions.

4.2 Optimized Bounding Box Algorithm

4.1.1 shows that an optimally oriented d-dimensional polytope will also have simultaneously local optimal d-k-dimensional projections, where $k = 1, 2, ...d - 1$. Proposition 4.1.3 gives the candidate set for the globally optimal orientations of a d-dimensional polytope, using simultaneously local optimal projections to isolate each candidate orientation. Corollary-4.1.4

proves that iteratively optimizing the orientation of a d-dimensional polytope by minimizing its two dimensional axial planar projections causes those projections to converge to a simultaneous local optimal.

Leveraging these results we can come up with a way to build up a candidate set for the global minimum bounding box by selectively sampling the space of starting orientations for a given d-dimensional polytope. For each of these orientations successive optimization of the axial projections results in convergence to a candidate orientation corresponding to a simultaneous local optimal. Thus the optimization performed on the projection results in optimizing the orientation of the d-dimensional polytope such that volume of the axis aligned bounding box at that point is locally minimal. We devise an algorithm to determine the minimum bounding box for a set of points defining a convex polytope in d-dimensions.

As mentioned previously, optimization methods are non-trivial for $k > 3$ (a motivation for the current algorithm), therefore optimization in the subspace where $k = 2$ is used. Given that any d-dimensional rotation can be defined in terms of a composition of two dimensional planar rotations [100] [99] as well as the simplicity of finding minimum bounding boxes for points in two dimensions, we use two dimensional axial planar projections as the basis for d-dimensional rotations. This can be accomplished with *Givens* rotations, which allow for the annihilation of arbitrary elements of a matrix via a 22 rotation matrix. The implication is that starting with a rotation matrix, a diagonal matrix can be obtained via Givens rotations. But, since orthogonality of a matrix is preserved by multiplication with another orthogonal matrix, this means that the diagonal matrix must be orthogonal, and hence, must contain only 1's and -1's. Additional rotations then reduce this matrix to the identity. We use the general matrix presented in [101] based on the projections of d-dimensional data coordinate axes. For the rotation of the d-dimensional polytope around a plane (say, $x - y$ plane) by an

angle θ, the corresponding rotation matrix is given by,

$$R_{a,b} = \begin{cases} r_{i,j} = cos(\theta) & i = x, j = x \\ r_{i,j} = cos(\theta) & i = y, j = y \\ r_{i,j} = -sin(\theta) & i = x, j = y \\ r_{i,j} = sin(\theta) & y, j = x \\ r_{j,j} = 1 & j \neq x, j \neq y \\ r_{i,j} = 0 & elsewhere \end{cases}$$

We now discuss the details of the algorithm utilization the two dimensional rotations just discussed to obtain the simultaneous local optimal corresponding to an arbitrary starting orientation for a d-dimensional polytope.

4.2.1 The Simultaneous Local Optimal Algorithm

The *Simultaneous Local Optimal Algorithm* or *Algorithm-SLO* leverages the result proved in Theorem-4.1.4 and the method used to do so to determine the locally optimal volume for the d-dimensional polytope. Algorithm-SLO sequentially projects the vertices of polytope P into each plane i where $i = 1, 2, .., m$ constructing the convex hull of each projection p_i. For a given plane i, unless it is already locally optimal, p_i is rotated by angle θ_i such that its axis-aligned bounding box becomes its minimum bounding box. Thus,

$$B(p_i(\theta_i)) = B_{opt}p_i$$

This transformation is applied back to P by rotating the entire polytope by θ_i around the same plane i. This results creates a new orientation of the polytope from which it is projected into the next plane $i+1$ in the sequence and the process repeated.

From Theorem-4.1.7 we see that the area of the bounding box of a given two dimensional projection repeats with a period of $\frac{\pi}{2}$. This leads to the following proposition,

Proposition 4.2.1. *For each local optimal (specific to the plane in which the rotation takes place) obtained during an intermediate rotation of P, there exist at least three more orientations of the polytope that would give the same optimal area in the projected plane.*

Since the repeatability of the local optimal by the given period corresponds in essence to a realignment of the axis with which the relevant edge of the projection is flush, the convex hulls of the projections in the remaining planes would also be repeated for each optimal. However, the specific axial plane in which they occur would change. Choosing the planes with the same projection with the same sequence as in the original local optimal would lead to the same simultaneous local optimal. This shows that for each local optimal in an intermediate orientation at least three more orientations of $\mathbf{P}$ lead to the same SLO.

This indicates that for a given starting orientation k, the actual orientations of P that lead to the same simultaneous local optimal can be given by,

$$\mathbf{S} = P_k \cap \mathbf{I} \cap \mathbf{I'} \cap SLO_k$$

where P_k is the orientation of P at k, $\mathbf{I}$ is set of intermediate orientations of P leading to the SLO and $\mathbf{I}$' the is set of matching orientations of P resulting in the same local optimals

for a given p_i. The number of these orientations is given by,

$$|\mathbf{S}_k| = 4n + 2 \tag{4.10}$$

and n is the number of orientations P goes through before reaching the orientation giving the simultaneous local optimal.

This process is repeated iteratively for all the m two dimensional projections, wrapping around until all the projections are locally optimal at the same time indicated by all the check variables being equivalent to one (according to 4.1.4 there exist a finite number of iterations for which the projections will converge to a simultaneous local optimal).

At this point *Algorithm-SLO* returns the volume of the axis aligned bounding box as the local minimum and the corresponding orientation of the d-dimensional polytope defined by the rotation matrix *Orientation*. The pseudocode for the algorithm is given in **??**

This algorithm provides a way to find a locally minimal volume and corresponding orientation that satisfy the necessary condition for the minimum bounding box of the d-dimensional polytope. We now need a method to isolate the set of all such orientations. The fact that every such orientation is associated with a specific starting orientation indicates that by selectively sampling the space of the starting orientations of the d-dimensional polytope we can obtain a set of locally optimal volumes the minimum of which is the absolute minimum or close to the absolute minimum bounding box of the polytope.

We first discuss the general case where the entire space of starting orientation is sampled exhaustively using a grid-based technique such that the distance between the grid points dictates the number of points that need to be sampled. We then discuss a series of heuristics that can be used to reduce the number of starting points that need to be sampled

Algorithm 4.1 Algorithm-SLO: Simultaneously Local Optimal Algorithm

Input: Data points in d-dimensions.
  $P \leftarrow Datapoints$
  $Proj \leftarrow$ All 2D projections of P
  $n \leftarrow NumberofProjections$
  Initialize all $check[i]$ to 0. ▷ Until All Projections Are Optimal
  while $check[1] * check[2] * ..... * check[n] \neq 1$ **do**
    for $i = 1 \rightarrow n$) **do**
      if $IsOptimal(Proj[i]$ **then**
        $check[i] \leftarrow 1$
      else
        $R \leftarrow$ RotatingCallipers($Proj[i]$). ▷ Minimize Each 2D Plane
        Apply R to P.
        $Proj \leftarrow$ All 2D projections of P
        $check[i] \leftarrow 0$
      end if
    end for
  end while
  $FinalVolume \leftarrow$ Volume of P.
  $Orientation \leftarrow$ Optimal Rotation of P.
  Output: $FinalVolume$, $Orientation$

(thus improving efficiency) while still maintaining an accurate level of optimization. These heuristics are based on the characteristics of the starting orientations and orientations that correspond to the simultaneous local optimal. The paradigm followed in each version is similar though, as the algorithm successively rotates the polytope in each k dimensional subspace where $k = 1, 2, ..., d - 1$, building up the set of candidate orientations for the global optimal by taking its k dimensional projections and finding the simultaneous local optimal for those projections. The orientation giving the lowest volume from the entire candidate set is taken as the globally optimized value. In case of multiple orientations corresponding to the same volume value, any one can be used to define the global optimal. The accuracy and speed of the algorithm can be controlled by controlling the granularity of the rotation in the k-dimensional subspace. The granularity in essence controls the number of starting orientations that are sampled to build up the candidate set. In each case, the algorithm is

independent of the method used to optimize the projection in the subspace. So another optimization algorithm can be used to replace the SLO for subspace optimization if necessary, say for example of the optimization is being carried out in three dimensions instead of two.

4.2.2 The Minimum Bounding Box Candidate Algorithm - Exhaustive Approach

From Theorem-4.1.4 we see that for given starting orientation of an arbitrary d-dimensional polytope P, there exists a corresponding orientation P' at which the volume of its bounding box is locally minimal. Quantifying the space of all possible starting orientations for P, therefore provides a method to quantify the the set of locally optimal orientations the lowest of which. Since any orientation or rotation of a polytope in d-dimensional space can be defined in terms of a series of rotations around $m = C\binom{d}{2}$ axial planes [99], the polytope P at orientation o defined by angle θ is given by,

$$P(\theta) = P * \prod R_i\theta_i) \tag{4.11}$$

where $R_i(\theta_i)$ defines the rotation in the ith two dimensional plane where $i = 1, 2, .., m$. The product of these rotations is equivalent to rotating P by θ in d-dimensional space.

From 4.11, the entire space of orientations can now be defined in terms of the rotations of P around each of its two dimensional planes. Expanding on this idea the first part of the algorithm involves building up an exhaustive set of starting orientations for the d-dimensional object using a grid based decomposition of the m-dimensional orientation space R^m where each axis represents the set of rotations in a single two dimensional axial plane. The set of starting orientations of P given by,

$$\mathbf{SO} = SO_1 \times SO_2 \times SO_3 \times ..... \times SO_m \tag{4.12}$$

where SO_i is the set of orientations obtained by the two dimensional rotation of P around plane i stepped by a granularity g. Each grid point corresponds to a unique orientation of P and is given by,

$$P_o = (\theta_1, \theta_2, ...\theta_m)$$

The orientations are obtained by doing a complete 360 degree rotation of the object around each of the $C\binom{d}{2}$ two dimensional axial planes. The number of unique isolated orientations and therefore, the corresponding coverage of orientation space is dependent upon g with which the rotation is stepped. This granularity defines the distance between each grid point. Let $\mathbf{O}$ be the set consisting of all possible orientations of $\mathbf{P}$ and O' be the set of starting orientations for the *MBB-Candidate* algorithm . Then it could be said that,

$$O'(g) = \frac{360^m}{g}$$

$$\lim_{g \to 0} O'(g) = \mathbf{O}$$

This granularity is set at the start of the algorithm and can be used to control the overall speed and accuracy. This is, in essence, a discretization of the space of starting rotations. For example, in three-dimensional space, there are $360 \times 360 \times 360$ distinct starting orientations (corresponding a complete rotation in the first plane times a complete rotation in the second plane and then the third) when the rotation is stepped with a granularity of 1 degree. If the granularity is changed to 0.1 the number of distinct starting rotations becomes $3600 * 3600 * 3600$.

Algorithm 4.2 Algorithm-MBC: Minimum Bounding Box Candidates - Exhaustive

Input: Enter d-dimensional data points.
Input: Granularity. ▷ Interval for rotation
$D \leftarrow Datapoints$
$G \leftarrow Granularity$
$i \leftarrow 1$
for all Axial Planes (Starting With An Orthogonal Plane) **do** ▷ Rotate each plane through 360 degrees
$j \leftarrow 1$
for $\theta = 0 \rightarrow 360$ and $\theta \leftarrow \theta + G$ **do**
$R \leftarrow rotmatrix(\theta)$
Rotate D by R
$[Volume, Orientation] \leftarrow SLO(\mathrm{D})$
$MinVolCand[j] \leftarrow Volume$ ▷ Store min. volume for each orientation
$MinOCand[j] \leftarrow Orientation$ ▷ Store corresponding orientation
$j \leftarrow j + 1$
end for
$[MinVolPlane[i], In] \leftarrow MinVol(MinVolCand)$ ▷ Find minimum volume for current planar rotation
$MinOPlane[i] \leftarrow MinOCand[In]$ ▷ Use index to find corresponding orientation
$i \leftarrow i + 1$
end for
$[MinimumVolume, Index] \leftarrow MinVol(MinVolPlane)$ ▷ Fined minimum volume for all planar rotations
$MinimumOrientation \leftarrow MinOPlane[Index]$
▷ Get corresponding orientation **Output:** $MinimumVolume, MinimumOrientation$
function ROTMATRIX(angle)
$Rotation \leftarrow$ Calculate Rotation for Current Plane **Output:** $Rotation$
end function

Thus, the granularity becomes a major factor in determining the execution time and accuracy. Decreasing the granularity makes the likelihood of finding the global optimal greater (since coverage of the starting orientation space increases) but takes more time and vice versa. This has the added advantage that increasing the number of vertices becomes a linear factor on the time complexity (affecting only the determination of the local optimal in rotating callipers), while the number of starting points tested (themselves linear with increasing granularity) determine the time required to isolate the global optimal. This means that the $MBB-Candidate$ is independent of the number of vertices present in the polytope. However, as we will show in the subsequent section the granularity can be manipulated to provide a significant speed-up while still maintaining close to 100% accuracy.

The rotation of the polytope around a plane (say, $x-y$ plane) by an angle θ is defined by the rotation matrix,

$$R_{a,b} = \begin{cases} r_{i,j} = cos(\theta) & i = x, j = x \\ r_{i,j} = cos(\theta) & i = y, j = y \\ r_{i,j} = -sin(\theta) & i = x, j = y \\ r_{i,j} = sin(\theta) & y, j = x \\ r_{j,j} = 1 & j \neq x, j \neq y \\ r_{i,j} = 0 & elsewhere \end{cases}$$

The pseudocode for the algorithm is defined in Algorithm 0.

Each orientation obtained in this manner is passed to the simultaneous local optimal algorithm. For each starting orientation, the corresponding local minimum volume and the orientation(in terms of a series of planar rotations in each two dimensional plane) returned by SLO are stored. After the object has been rotated around each axial plane, the minimum

of all the local minimum volumes returned by the Algorithm-SLO is obtained along with the rotation matrix defining the corresponding orientation. This gives the globally optimized minimum volume bounding box for the d-dimensional points.

4.2.3 Minimum Bounding Box Candidate Algorithm - Heuristic Approach

The exhaustive approach provides a good way to cover as much of the starting orientation space as possible, thus building up as complete a set of candidate locally optimal orientations from the the SLO-Algorithm for the final isolation of the globally optimized volume. However, as is clear from the manner in which it selects those orientations, it is exponential in the number of planes as the number of dimensions increases. Experimentally, we find that this method while efficient enough for $d \leq 4$ becomes prohibitively expensive as d goes beyond that point. To counter this effect we try to come up with a way to selectively sample the space of starting orientations while still trying to ensure that the bounding box is sufficiently optimized. This sampling is based on the following propositions,

Proposition 4.2.2. *If $\boldsymbol{S}$ is the set of all starting orientations of P and $\boldsymbol{O}$ is the set of all locally optimal orientations of P, then $\boldsymbol{O} \subseteq \boldsymbol{S}$ and $|\boldsymbol{O}| \ll |\boldsymbol{S}|$*

The proposition stated above follows from the eq.4.10,Theorem-**??** and the fact that every locally optimal orientation corresponds to a starting orientation itself.

Extrapolating from Theorem-4.1.4 we can come up with way of uniformly sampling starting orientations of P across d-dimensions. Instead of using the cartesian product of the sets of possible axial planar rotations of P to generate the set of starting orientations, we instead utilize a union of these sets i.e.,

$$\mathbf{SO} = SO_1 \cup SO_2 \cup SO_3 \cup ..... \cup SO_m$$

where SO_i is the set of the two dimensional rotations of P around plane i stepped by granularity g. The members of this set thus define m-mutually orthogonal lines passing through the origin of orientation space R^m providing the basis of a uniform distribution of sampling points in the orientation space.

In addition to the results of Theorem-4.1.7, eq.4.10 and Proposition-4.2.2, this sampling method also provides adequate coverage of the rotation space by the leveraging the fact that for a fixed value of rotation in one plane, rotating in the second orthogonal plane is sufficient to cover the the set of starting orientations obtained from rotating in the third plane since both would lead to the same local optimal if the starting projection optimized for the $SLO - Algorithm$ is in the second plane.

Using the optimization just mentioned for three dimensional space,for example, we need only check 360×360 points to cover the entire space. This indicates that the choice of the starting plane for the SLO-Algorithm becomes an important consideration in controlling the uniformity of the distribution of orientations sample as well the region and amount orientation space covered, resulting in a further decrease in the number of distinct starting orientations that need to be tested while maintaining the same coverage. The pseudocode for the algorithm is provided in 0.

Based on the choice of the starting plane in the SLO-Algorithm we can define a series of heuristics that result in changing the distribution and sampling region in R^m of the starting orientations covered for the given set candidate locally optimal orientations obtained.

Algorithm 4.3 Algorithm-MBC: Minimum Bounding Box Candidates - Heuristic

Input: Enter d-dimensional data points.
Input: Granularity. ▷ Interval for rotation
$D \leftarrow Datapoints$
$G \leftarrow Granularity$
$i \leftarrow 1$
for all Axial Planes (Starting With An Orthogonal Plane) **do** ▷ Rotate each plane through 360 degrees
$j \leftarrow 1$
for $\theta = 0 \rightarrow 360$ and $\theta \leftarrow \theta + G$ **do**
$R \leftarrow rotmatrix(\theta)$
Rotate D by R
$[Volume, Orientation] \leftarrow SLO(\mathrm{D})$
$MinVolCand[j] \leftarrow Volume$ ▷ Store min. volume for each orientation
$MinOCand[j] \leftarrow Orientation$ ▷ Store corresponding orientation
$j \leftarrow j + 1$
end for
$[MinVolPlane[i], In] \leftarrow MinVol(MinVolCand)$ ▷ Find minimum volume for current planar rotation
$MinOPlane[i] \leftarrow MinOCand[In]$ ▷ Use index to find corresponding orientation
$i \leftarrow i + 1$
end for
$[MinimumVolume, Index] \leftarrow MinVol(MinVolPlane)$ ▷ Fined minimum volume for all planar rotations
$MinimumOrientation \leftarrow MinOPlane[Index]$
▷ Get corresponding orientation **Output:** $MinimumVolume, MinimumOrientation$
function ROTMATRIX(angle)
Rotation ← Calculate Rotation for Current Plane **Output:** *Rotation*
end function

4.2.3.1 Heuristic 1

The first heuristic proposes that for each two dimensional plane i where $1 \leq i \leq m$ around which P is rotated by an angle θ to determine a starting orientation, the plane chosen in the SLO-Algorithm to start the optimization process is always the same corresponding to the first or the xy-plane. The set of starting orientations sampled under this heuristic would therefore consist of planes in the starting orientation space R^m (defined by axes Θ_1,Θ_2,....,Θ_m), with at least one axis being defined by the set of rotations of P around plane-1. This is given by,

$$SO_{H1} = \bigcup_{i=1}^{m} \Theta_{1i}$$

where Θ_{1i} represents the set of orientations of P in the plane of rotations in R^m defined by the axes θ_1 (corresponding to rotations around the first or xy-plane) and θ_i (corresponding to rotations around the ith plane). The actual coverage of the R^m provided by SO_H1 is augmented by the optimization discussed in the previous section combined with result from proposition-4.2.1. For each orientation, the other steps to isolate the global minimal bounding box remain the same as discussed on the previously.

4.2.3.2 Heusristic 2

The second heuristic proposes that for each two dimensional plane i where $1 \leq i \leq m$ around which P is rotated by an angle θ to determine a starting orientation, the plane chosen in the SLO-Algorithm to start the optimization process is selected randomly. However, the selection is restricted by the constraint that the plane be orthogonal to the plane of the starting orientation and repeated with the only condition being that a particular plane does not repeat more than a given number of times t. The set of starting orientations sampled

under this heuristic would therefore consist of planes in the starting orientation space R^m (defined by axes Θ_1,Θ_2,....,Θ_m), with each axis being defined by the set of rotations of P around planes-i and j. This is given by,

$$SO_{H2} = \bigcup_{\substack{i=1 \\ j \in m}}^{m} \Theta_{ij}$$

where Θ_{ij} represents the set of orientations of P in the plane of rotations in R^m defined by the axes θ_i and θ_j. In this case $i \neq j$ and j does not repeat more than $m-1$ times.

4.2.3.3 Heuristic 3

The third heuristic proposes that for each two dimensional plane i where $1 \leq i \leq m$ around which P is rotated by an angle θ to determine a starting orientation, the plane chosen in the SLO-Algorithm to start the optimization process is orthogonal and disjoint in the sense that it is not repeated as a choice for any of the other starting planes. The set of starting orientations sampled under this heuristic would therefore consist of planes in the starting orientation space R^m (defined by axes Θ_1,Θ_2,....,Θ_m), with each axis being defined by the set of rotations of P around planes-i and j where,

$$SO_{H3} = \bigcup_{\substack{i=1 \\ j \in m}}^{m} \Theta_{ij}$$

where Θ_{ij} represents the set of orientations of P in the plane of rotations in R^m defined by the axes θ_i and θ_j. In this case $i \neq j$ and j does not repeat at all. In other words, the set Θ_{ij} is unique and has no overlap or intersection with any other axes or plane of rotation in $\mathbf{R}^m$.

This heuristic provides the greatest coverage and uniformly spread sampling of R^m. Each unique combination of planar rotations covers an independent region of space, with no overlap with any other sampled region. The number of points sampled being the same as the previous two heuristics this ensures that the likelihood of finding distinct candidates for the optimized orientation of P from the SLO-Algorithm is increased. In consequence the optimization on the volume of the bounding box of the polytope is also increased.

4.2.4 Local Optimal Clustering

Empirical observation also provided an additional characteristic of the locally optimal orientations. Experimental results show that these orientatins tend to occur contiguously in clusters. An example of the type of clusters occurring for individual shapes as well as the example shape used for three dimensions is shown in figure 4.1. Starting orientations close to each other lead to the same cluster. It is observed that the number of clusters detected increases with the decrease in the granularity of the minimum bounding box candidate algorithm utilized. However, this increase is bounded by an upper limit in the fineness of the granularity (specific to each d-dimensional shape) which when reached results in an an increase the members of individual clusters rather than further increase in the number of clusters themselves. As we can see in the figures, the clusters are not very tight but spread out in space with lots of scattering. The scattering is indicated by the presence of singletons that occur in random portions of the d-dimensional angular space. However, the no of singletons is much less than the total number of SLOs covered by the larger clusters. Again, this scattering an be decreased to an extent by decreasing the granularity. The clusters were observed for polytopes in higher dimensional space as well and exhibit properties along the same line as discussed above. Some statistics defining clusters formed for random polytopes

in 3, 4 and 5 dimensions are provided in table 4.1. The table defines the number of clusters observed for a given number of vertices in a specific dimension. The maximum deviation of volume of the minimum bounding boxes corresponding to the SLOs within the clusters from the observed minimum is also noted as well as the threshold value of granularity beyond which the number of clusters stop increasing. The decrease in deviation with increase in dimensions is in part due to the method used for generating random polytopes. As we increase dimensions, the polytope generated gets closer and closer to a d-dimensional sphere which means that the bounding boxes for the SLOs isolated shall have smaller differences between their volumes.

The occurrence of clusters could be explained in terms of the optimization of two dimensional planes used as the primary basis of the overall volume minimization. For a given set of adjacent starting orientations, the two dimensional projection of a given d-dimensional polytope in a specific plane provides a set of similarly shaped two dimensional polygons (same or similar number of edges with similar angles between them), therefore the likelihood of each polygon in the set converging to a local optimal that varies only slightly from others in the group is increased. The same principle applied to each of the two dimensional projections used in the optimization process could result in the final orientation of the corresponding d-dimensional local optimals to be quite close to each other resulting in clusters in $m = C\binom{d}{2}$-dimensional angular orientation space. These clusters can be leveraged to create another version of the MBB-Candidate algorithm focused on isolating the global optimal from clusters of local optimals which we now discuss.

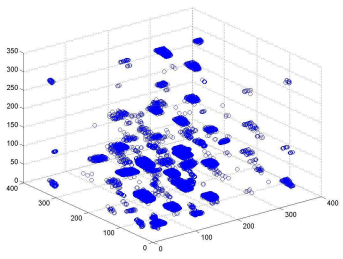
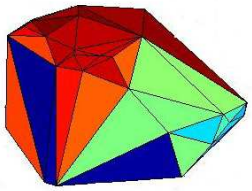

Figure 4.1: Examples of Clusters for a Random Polytope

Table 4.1: Cluster Statistics With Varying Dimensions

Vertices	Dim	Points	Clusters	Vol. Dev. From Min.	Threshold
35	3	2160	61	1.79	0.65
35	4	2880	69	1.67	0.80
35	5	3600	85	1.64	0.95

4.2.5 Minimum Bounding Box Candidate Algorithm - Grid Based Clustering

The primary motivation behind this version of the algorithm is the empirically observed characteristic that is presented in the following proposition.

Proposition 4.2.3. *Given an orientation P_k any orientation within a distance Δ of the P_k leads to the same cluster of locally optimal orientations.*

Since the locally optimal orientations occur in clusters a coarse grained version of the MBB-Algorithm utilizing any one of the heuristics can be used to identify the initial set of simultaneous local optimals in conjunction with SLO-algorithm. A clustering algorithm can then be applied to the points obtained to isolate regions where further locally optimal orientations can occur. Using the boundaries of these regions as limits for isolating corresponding starting points, a fine grained version of the same MBB-Algorithm can be used to identify candidate locally optimal orientations.

For a given set of starting points in d-dimensional angular space the corresponding set of simultaneous local optimals forms a cluster in the same set of dimensions. The number of clusters is fixed and as the granularity gets smaller and smaller the clusters stop increasing after a specific threshold and the number of simultaneous local optimals within each cluster starts to increase. However, depending upon the granularity used there do exist a set of singleton simultaneous local optimals that manifest in the d-dimensional angular space. It is observed that the number of unique singletons decreases as the granularity utilized for starting the candidate algorithm is decreased. However, as the number of singletons observed is very small compared to the total number of simultaneous local optimals, the likelihood of the global optimals occurring within those singletons is commensurately low. Furthermore, the bounding volumes corresponding to these singletons are very close to the bounding volumes for the simultaneous local optimals lying within specific clusters in terms of absolute values. The standard deviation of the values for bounding volumes of corresponding to simultaneous local optimals lying within clusters for a specific data set is presented in the table.The deviation from the value of the minimum volume bounding box observed is for 3 and 4 dimensional objects each having $10 - 30$ vertices as indicated in Table 4.2 and in 4.3 . It can be seen from the values presented, within each cluster the variation in the values of the bounding volumes is low.

4.2.5.1 Optimizations to Algorithm

An important factor in the efficiency of this method is the clustering method used. If the clusters obtained manage to encompass most of the local optimal orientations identified in the first phase then the likelihood of obtaining the global optimal is increased and vice versa. On the other hand any clustering technique which is exhaustive enough to cover all the data

points identified takes significantly longer to run thus reducing the efficiency in comparison with other methods. Balancing these two constraints becomes an important factor in this version of the candidate algorithm.

The clustering technique utilized after isolation of the simultaneous local optimals plays a major part in determining the time required to actually isolate the global optimal. The starting points can be selected using either of the two approaches mentioned earlier. Using the heuristic based method method is preferable since the one to one correspondence between the the starting points and SLOs means that the heuristic approach would have a fewer number of points to cluster. The algorithm can be optimized to obtain a balance between the time required to carry out a detailed clustering of the simultaneous local optimals obtained for a given granularity (thus resulting in greater accuracy) and the time required to do the clustering process. The following two clustering techniques were tested:

- Agglomerative Clustering
- Hierarchical Clustering

Based on the time taken and accuracy of the clustering methods considered , the SLOs are obtained using an agglomerative clustering algorithm using the normalized weighted distance between the simultaneous local points in the d-dimensional space being used as a metric to determine their membership in a given cluster. The primary objective of using this algorithm is the fact that it does not require either cluster size or number of clusters as a metric while carrying out clustering. The only metric required and provided is the threshold distance necessary for identifying cluster membership based on proximity.

4.2.6 The Clustered Simultaneous Local Optimal Algorithm)

The basic process is similar to the methods discussed earlier. The heuristic based candidate algorithm combined with the simultaneous local optimal algorithm is used with a mid level granularity to isolate an initial set of simultaneous local optimals. These local optimals are clustered using the clustering method discussed earlier. For each cluster obtained the starting points corresponding to the outliers of each observed cluster are identified and the heuristic based candidate algorithm is used again within each set of starting points with fine granularity. The lowest bounding volume from the set of volumes corresponding to the local optimals thus obtained is the minimized bounding box. The pseudocode for the algorithm is provided in 0.

Table 4.2: Standard Deviation of Volume With Varying Vertices & Granularity in % for 3D

Granularity	0.5	1	2	5
10	1.62	1.83	1.97	2.22
20	1.81	1.82	1.93	2.17
30	1.75	1.87	2.1	2.35

Table 4.3: Standard Deviation of Volume With Varying Vertices & Granularity in % for 4D

Granularity	0.5	1	2	5
10	1.71	1.88	1.72	1.98
20	1.64	1.75	1.91	2.32
30	1.68	1.79	1.98	2.57

4.3 Experimental Results

In this section we present experimental results that demonstrate the efficacy of using the Simultaneous Local Optimal algorithm to find the minimum volume bounding boxes of

Algorithm 4.4 Algorithm-MBC: Minimum Bounding Box Candidates - Clustering

Input: Enter d-dimensional data points.
Input: Granularity. ▷ Interval for rotation
$D \leftarrow Datapoints$
$G \leftarrow Granularity$
$i \leftarrow 1$
for all Axial Planes (Starting With An Orthogonal Plane) **do** ▷ Rotate each plane through 360 degrees
$j \leftarrow 1$
for $\theta = 0 \rightarrow 360$ and $\theta \leftarrow \theta + G$ **do**
$R \leftarrow rotmatrix(\theta)$
Rotate D by R
$[Volume, Orientation] \leftarrow SLO(\mathrm{D})$
$MinVolCand[j] \leftarrow Volume$ ▷ Store min. volume for each orientation
$MinOCand[j] \leftarrow Orientation$ ▷ Store corresponding orientation
$j \leftarrow j + 1$
end for
$[MinVolPlane[i], In] \leftarrow MinVol(MinVolCand)$ ▷ Find minimum volume for current planar rotation
$MinOPlane[i] \leftarrow MinOCand[In]$ ▷ Use index to find corresponding orientation
$i \leftarrow i + 1$
end for
$ClusterMinVolPLane$ ▷ Cluster the SLOs obtained
$IsolateClusterOutliers$
$SP \leftarrow StartingPointsCorrespondingtoOutliers$
for all Starting Points Ranges in SP **do**
$RepeatMBC - Heuristicwithfinergranularity$ ▷ Isolate new set of simultaneous local optimals
end for
$[MinimumVolume, Index] \leftarrow MinVol(MinVolPlane)$ ▷ Find minimum volume for all planar rotations
$MinimumOrientation \leftarrow MinOPlane[Index]$
▷ Get corresponding orientation **Output:** $MinimumVolume, MinimumOrientation$
function ROTMATRIX(angle)
$Rotation \leftarrow$ Calculate Rotation for Current Plane **Output:** $Rotation$
end function

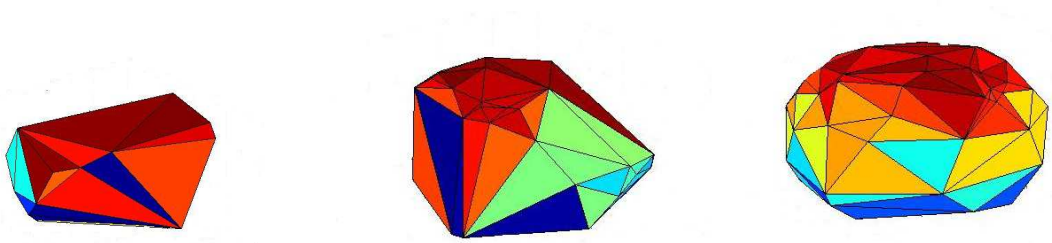

Figure 4.2: Examples of Random Polytopes Tested

general d-dimensional polytopes. We first discuss the experimental setup involved in running the experiments, followed by a description of the experimental paradigm. This is followed by a runtime analysis of the SLO-Algorithm and the different versions of the MBB-Candidate algorithm. We then discuss the application of this paradigm to randomly generated three dimensional vertices and compare the results obtained with the only known method that guarantees a minimum bounding box for polytopes in three dimensions as. This is followed by a discussion of the paradigm as applied to four to twenty dimensional convex polytopes and the advantages obtained therein in the optimization of volume compared to standard axis aligned bounding boxes and the estimation method presented in [**?**] to solve this problem. Next the effect of each of the three heuristics on the quality of optimization obtained is elaborated upon and we see how, specially in the higher dimensions choosing the correct heuristic can have an appreciable effect on the optimization obtained.

4.3.1 Experimental Setup

In order to test the algorithms exhaustively, we randomly generated sets of convex d-dimensional points with variable number of vertices. The primary constraint for the points comprising each test case was that they describe a convex polytope in d-dimensional space.

Some examples of the polytopes generated are shown in Figure.4.2.

For tests in three dimensions (for the purpose of comparison with other algorithms), we generated a hundred convex polytopes each for a given number of vertices , with the number of vertices ranging from eight to five hundred. In addition data sets consisting of polytopes of up to $10,000$ vertices were also generated to test execution times of the optimization algorithm and effect of granularity on the volume optimization obtained.

For tests in four to eight dimensions we similarly generated ten tests cases each with number of vertices varying from ten to fifty. As the primary object of the experiments in higher dimensions was to illustrate the effectiveness of the approach when applied to d-dimensions (significant minimization of bounded volume) compared to axis aligned bounding boxes rather than a comparison with existing methods (as in the three dimensional case) the number of test cases generated are much less. However, they are sufficient to indicate the minimization trend that occurs with the increase in dimensions. Additionally, for all experiments we used the rotating callipers algorithm [9] to minimize the two dimensional projections of the polytope within the SLO. This algorithm is quite simple to implement and has the added advantage of giving the minimum bounding box of a two dimensional polygons in linear time. Depending upon the nature of the polytopes being optimized this algorithm can be replaced by other algorithms to improve the execution time if required.

All computations and experiments have been carried out using Matlab ®7.14.0 (R2012a) on a Dual-Core Intel ®$Core^{TM}$ i5-2410M 2.30 GHz with 6GB RAM. We measure all results with a precision of 10^{-4}.

4.3.2 Theoretical Analysis

4.3.2.1 Optimized Bounding Box - SLO

The simultaneous local optimal algorithm uses the optimization of two dimensional projections as the mechanism to isolate the locally optimal orientation of the d-dimensional polytope. The optimization method used therefore becomes the primary driving factor behind the run time of the SLO-Algorithm. We utilize the rotating callipers algorithm presented in [9] to find the minimum bounding box of the projection of the polytope in a given two dimensional plane. Since the rotating callipers is $O(n)$ in the number of vertices projected onto the two dimensional plane, for a finite number of rotations r required by SLO to converge to a simultaneous local optimal the run time would be given by $O(r*n)$. This indicates that SLO is linear in terms of the number of vertices of the polytope. $O(n*r)$ where n is number of vertices required for constructing a convex hull during rotating callipers and r is the number of intermediate rotations.

4.3.2.2 Optimized Bounding Box - MBBC(Exhaustive)

For the exhaustive MBB-Candidate algorithm, the number of orientations selected is dependant on the dimension d and the granularity g of the grid. Since the objective is to sample every starting orientation for a given g, this requires a complete rotation around each plane i times the next plane and so on till plane m stepped by g . Therefore, the runtime for the algorithm can be given by $O(2\pi/g)^m$ or $O(1/g)^m$, indicating that the key factor for the algorithm is granularity chosen. Decreasing it increases run time and vice versa.

4.3.2.3 Optimized Bounding Box - MBBC(Heuristic)

For the heuristic based MBB-Candidate algorithm, the number of orientations selected is again dependant on the dimension d and the granularity g of the grid. However, since the orientation space is sampled selectively in this case, with the sampling being based a series of strategically placed orientation planes in space it only requires a single complete rotation around each plane i independently stepped by granularity g. The number of starting orientation planes chosen for each of the heuristics mentioned is similar so the run time in each case would be the same. Therefore, the runtime for the algorithm can be given by $O(2\pi m/g)$ or $O(1/g)m$. The key factor is again the granularity, however as is clear the number of orientations sampled is now linear in the number of the planes rather than exponential as was the case for the exhaustive approach.

4.3.3 Minimum Bounding Boxes For 3-Dimensional Polytopes

In three dimensions, the optimized bounding box algorithm was run initially on randomly generated convex polytopes with 8 to 500 vertices. For a given number of vertices, the algorithm was applied to a 100 test cases (i.e. generating 100 sets of the same number of vertices). In each case it was found that for any given orientation of the polytope (defined by its rotation around each of the axial planes) the SLO converged to a simultaneous local optimal within a maximum of ten iterations through the axial planes at most and three iterations in general. The minimum value of the volumes defined by the multiple simultaneous local optimals was taken as the global optimal. To check the accuracy of the optimized bounding boxes thus obtained we compared them to the results obtained by applying ORourke's Algorithm [11] to the same test cases. This is the only algorithm that is known to give

a guaranteed minimum bounding box in three dimensions. The version utilized was implemented in Matlab and provided in [14]. Furthermore, a comparison was also performed with the commonly used PCA method for finding the optimized bounding box to see how SLO fares in comparison to the fastest prevalent method. Given the significant increase in execution time for O'Rourke's method with increasing number of vertices, the first ten test cases for each vertex group were compared. The results of this comparison are discussed in detail below.

4.3.3.1 Effect of Varying Heuristics

The heuristics discussed are focused on maximizing the space covered and optimally selecting the starting orientations of a given polytope. In three dimensions, only three two dimensional planes are available to make choices for the orthogonal starting plane of the SLO. Given the orientation space is thus quite small and the space coverage optimization provided by the intermediate SLO orientations still holds, the optimal is arrived at with little effect from the choice of the starting plane. As shown in previous sections, in three dimensions the granularity is key factor in determining whether the minimum bounding box is isolated or not. The optimized values of the volume of arbitrary three dimensional polytopes having $10-500$ vertices are compared against each other, keeping the granularity fixed at 0.5. From the Tables 4.4, 4.5, 4.6 and 4.7, 4.8 we see that while heuristic 3 is better than 1 at times where the optimized bounding box does not match the global optimal, the improvement is fractional. In most cases both the heuristics match the global optimal as shown. The improvement of optimization produced by the heuristics discussed is more apparent in higher dimensions where the choices available for the starting orientation and the hence the space to be covered is greater.

4.3.3.2 Effect of Increasing Vertices

Increasing the number of vertices in the convex polytopes to be optimized, has slightly contrasting effects on the SLO and ORourke's algorithms. For SLO the most important factor is the general increase in the number of edges of the convex polytope defined by the vertices (data points). Since the candidate orientations in the first part of the algorithm and the number of planes to be optimized is constant across any number of vertices for a given granularity, the only variable is the general increase in the number of edges in the two dimensional projections being optimized using the rotating callipers.

Since this algorithm gives results in linear time with respect to the number of vertices, we see a more or less linear increase in the execution time for SLO as the number of vertices are increased as illustrated in Figure.4.4. However, this increase is regulated by the orientation of the vertices. Since the points for the test cases are randomly generated, the additional vertices for each successive set of test cases might not cause an increase in the number of edges. If the additional vertices are oriented such that a larger number of data points lie on the same boundaries, they might decrease the number projected edges. Hence the decrease in execution time for some of the test cases with larger number of vertices (exemplified by the test cases for 100 and 200 vertices in Figure. 4.4).

ORourke's algorithm runs in cubic time with respect to the number of vertices and as we can see the execution time increases significantly with increase in the number of vertices. The execution times for ORourkes and SLO with a granularity of 0.5 are comparable for a low number of vertices but for polytopes with vertices greater than 50, SLO has a clear advantage. An advantage that gets better considerably as the number of points gets larger and larger. In contrast, while PCA remains quite fast with increasing vertices comparable to

the fastest version of SLO, its performance in terms of accuracy is extremely weak, matching the global optimal less than 30% of the time in some cases and hovering much below that for most as shown in Figure. 4.6.

The linearity of execution time is quite visible from fig.4.3 where polytopes bounded by thousands vertices are tested. As can be seen the time required to calculate the minimum bounding box of each polytope increases slowly even with a large increase in the number of vertices.

4.3.3.3 Effect of Increasing Granularity

The number of candidate orientations are controlled by granularity (interval of rotation) around each axial plane. The smaller the granularity the greater the number of candidate starting orientations and thus the greater the likelihood of arriving at the optimal minimum bounding box. However, this results in a corresponding increase in the execution time as the SLO has to be run for every candidate orientation. The objective is to achieve a good balance between accuracy and speed by controlling the granularity. From Figure.4.5 we can see that SLO with a granularity of 0.1 gives the optimal bounding box a hundred percent of the time for all test cases. Its execution time, on the other hand, for vertex sets less than 150 vertices while still linear is more than ORourke's as shown in Figure.4.4. Above 150 as the execution time for ORourke increases enormously its linearity allows it to post a significant advantage. Conversely, increasing the granularity to 1.5 and above cuts the execution time by a factor $(360)/G$ where G is the granularity as illustrated in Figure4.4. Based on experimental results, a granularity of 0.5 provides the best balance between speed and accuracy. It matches the exact optimal 95 to 100 percent for all sets of polytopes with a specified number of vertices and its execution time while slightly slower than O'Rourke's

for vertices ≤ 30, is much faster for vertices ≥ 50 (Figure.4.4). However, it is interesting to note that for all granularities great than 0.1, the average % error by volume for test cases not matching the optimal is extremely low as illustrated in Figure.4.7. In fact, excepting three or four datapoints, all other non-matching test cases display an average %error of less than 0.007%. Thus, for a chosen granularity of 0.5 and a given vertice set the 5% – 10% of times the result does not match the exact optimal, it differs only in third or fourth decimal point.

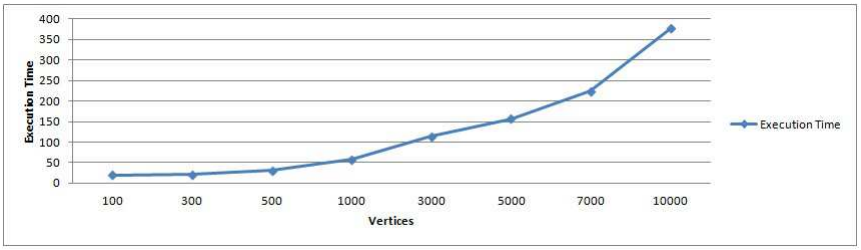

Figure 4.3: Execution Time for SLO With High Number of Vertices

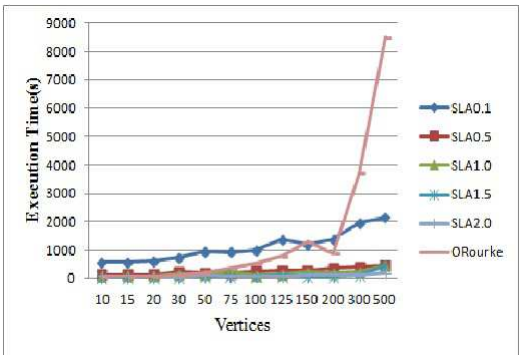

Figure 4.4: Execution Time for SLO and ORourke

4.3.4 Minimum Bounding Boxes For d-dimensional Polytopes

In higher dimensions, the SLO was run initially on randomly generated convex polytopes with 10,15,20 and 50 vertices. These tests were run for $4, 5, 6, 7$ and 8 dimensions to determine the general trends for execution time and optimization with increasing number of dimensions.

Table 4.4: Effects of Heuristics in 3D - 10V

Instances	ORourke	SLO-H1	SLO-H3
1	2.13716	2.13716	2.13716
2	1.545414	1.545414	1.545414
3	2.861554	2.861554	2.861554
4	2.95988	2.95988	2.95988
5	1.980998	1.980998	1.980998
6	1.81546	1.81546	1.81546
7	2.552137	2.552137	2.552137
8	2.456895	2.456895	2.456895
9	2.724014	2.724062	2.724062
10	2.653667	2.653667	2.653667

Table 4.5: Effect of Heuristics in 3D - 50V

Instances	ORourke	SLO-H1	SLO-H3
1	5.90747	5.90747	6.483248
2	5.912747	5.912747	6.659173
3	5.546646	5.546646	6.857865
4	5.857574	5.857574	6.686615
5	5.807628	5.807628	6.251951
6	6.105782	6.105782	6.757059
7	5.870182	5.870182	6.735433
8	5.892193	5.892193	6.429091
9	5.796868	5.796868	6.827371
10	5.845011	5.845011	6.678335

Table 4.6: Effect of Heuristics in 3D - 100V

Instances	ORourke	SLO-H1	SLO-H3
1	6.483248	6.483248	6.483248
2	6.659173	6.659177	6.659284
3	6.857865	6.857904	6.857866
4	6.686615	6.686615	6.686615
5	6.251951	6.251951	6.251951
6	6.757059	6.757059	6.757059
7	6.735433	6.735433	6.735433
8	6.429091	6.429091	6.429091
9	6.827371	6.827371	6.827371
10	6.678335	6.678335	6.678335

Table 4.7: Effect of Heuristics on 3D - 200V

Instances	ORourke	SLO-H1	SLO-H3
1	7.134995	7.134995	7.631631
2	6.875339	6.875339	7.636308
3	7.141376	7.141376	7.666846
4	7.098962	7.098962	7.603187
5	7.236272	7.23627	7.593752
6	7.260731	7.260731	7.535476
7	7.123031	7.123031	7.66344
8	7.236821	7.236809	7.545288
9	7.28961	7.28961	7.61791
10	7.206606	7.206606	7.622973

Table 4.8: Effect of Heuristics in 3D - 500V

Instances	ORourke	SLO-H1	SLO-H3
1	7.631631	7.631631	7.631631
2	7.636308	7.636309	7.636318
3	7.666846	7.666846	7.666846
4	7.603187	7.603187	7.603187
5	7.593752	7.593752	7.593752
6	7.535476	7.535476	7.535476
7	7.66344	7.66344	7.66344
8	7.545288	7.545287	7.545288
9	7.61791	7.617916	7.61791
10	7.622973	7.622973	7.622973

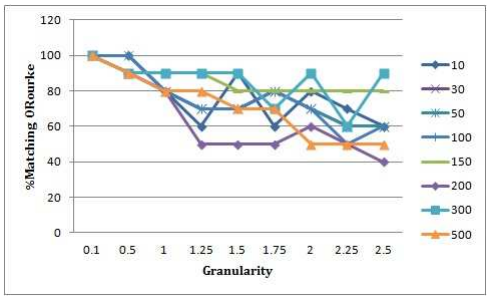

Figure 4.5: % Accuracy of SLO for Different Granularities

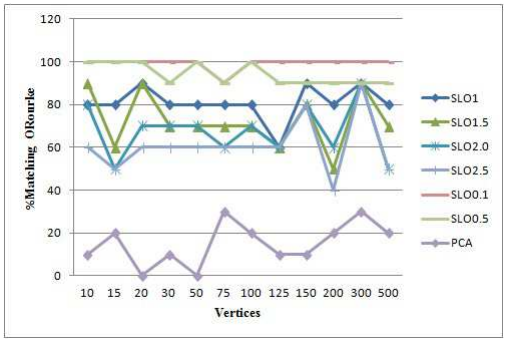

Figure 4.6: % Accuracy of SLO & PCA for Increasing No. of Vertices

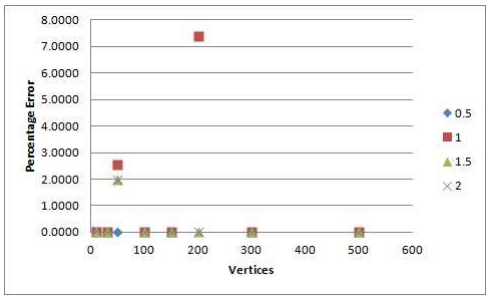

Figure 4.7: % Error of Polytopes Not Matching ORourke

For a given number of vertices, the algorithm was applied to ten test cases. The smaller number of test cases is sufficient to display the trend of the optimization obtained over axis aligned bounding boxes for d-dimensional polytopes. To study the effect of varying heuristics sets of randomly generated polytopes with vertices varying between $10-20$ vertices are used while keeping the granularity set at 1. The optimized volume obtained is compared with the volume of the axis-aligned bounding in d-dimensions and the optimized volume obtained from the technique based on diameter estimation provided in [**?**]. As can be seen the

optimization obtained from this method is much less than the optimization obtained from the Optimized Bounding Box Algorithm.

4.3.4.1 Effect of Increasing Granularity

Granularity also plays an important part in high dimensions in controlling the optimization obtained. In fact, the control provided by the granularity could be more useful in higher dimensions in predicting the absolute global optimal by providing an asymptotic mechanism to define the point at which optimization obtained cannot be improved upon. We show the effect of changing granularity in figures. 4.8 to 4.10.

Each figure shows the decrease in volume for a particular d-dimensional polytope with decreasing granularity. The range of granularities chosen is much larger in order to indicate the trend of the volume optimization. For granularity smaller than 0.5 the optimization remains constant or only changes fractionally which can allow us to predict the absolute global optimal for the given polytope with a high level of accuracy. The lowest volume can also be isolated earlier as seen from fig. 4.10 if the SLO defining the grid point corresponds to a starting orientation that is part of the set of orientations generated with a larger granularity. In general, the volume of the bounding box tends to be higher for a larger granularity, since the number of starting orientations sampled are much smaller.

4.3.4.2 Effect of Increasing Dimensions

As we increase the number of dimensions, the number of two dimensional axial projections also increases. For a d dimensional polytope the number of 2D parallel axial projections are given by $m = C\binom{d}{2}$. Thus, in addition to the number of vertices, m also becomes a factor in the increasing execution time. While SLO still gives the optimized bounding box the time

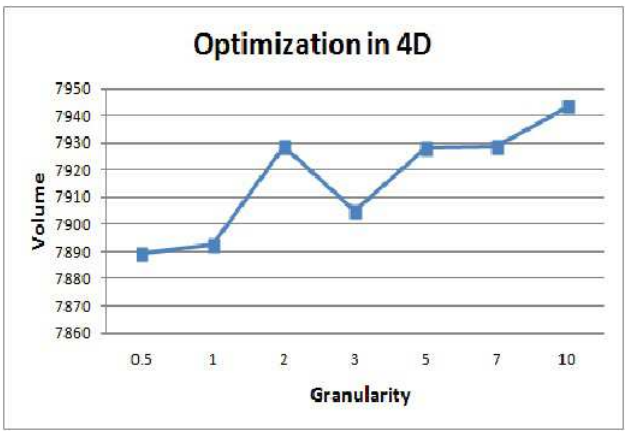

Figure 4.8: %Volume Optimization For a 4D-Polytope using SLO

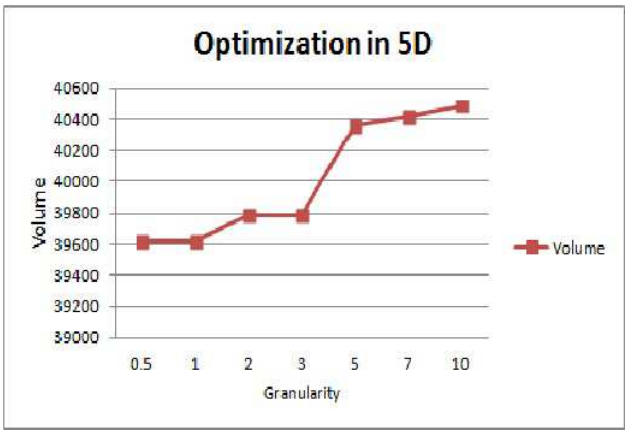

Figure 4.9: %Volume Optimization For a 5D-Polytope using SLO

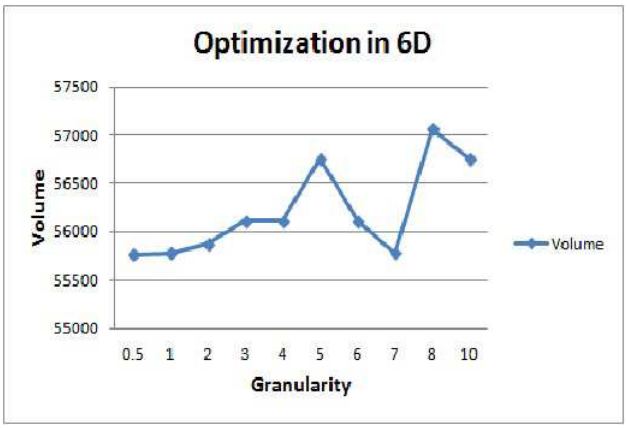

Figure 4.10: %Volume Optimization For a 6D-Polytope using SLO

taken to isolate the minimal volume could increase quickly with an increase in dimensions. This can be seen from fig.4.11 Therefore, for dimensions greater than 6 larger sizes of the

granularity can be used to reduce the time taken to arrive at an optimal volume. Given the scale of optimization obtained for $d > 3$ as shown in fig. this could be considered an acceptable choice. If maximizing the decrease in the volume is the objective for a particular application, then the granularity could be reduced further to achieve the required optimization.

As the number of dimensions are increased, there is a commensurate increase in the optimization obtained. This means that for higher dimensions, the decrease in volume of the optimized bounding is very large when compared to the axis aligned bounding box. Thus for applications where a high dimensional bounding box is used to encapsulate a given set of points (querying multimedia databases), utilizing this transformation could have significant impact. The optimization obtained is illustrated by Figure.4.12. The comparison between optimization obtained for instances of $4-10$ dimensional polytopes is shown in Table 4.9.

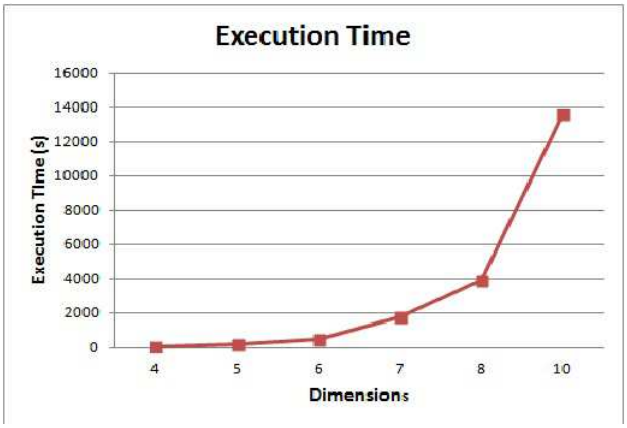

Figure 4.11: %Execution Time For D-Dimensions With A Granularity of 1

Table 4.9: Comparison of Optimization For Polytopes in d-dimensions

Dim	4	5	6	7	8	10
AABB	36824.027	133542.736	465692.231	12219776.7	72234570.73	22621605161
SLO	9815.319	39616.433	59063.183	175240.158	990024.990	376762123.4
DE	35240.594	124996.001	440079.159	11498809.9	69561891.61	21083336010
%OptS	73.34	70.33	87.31	98.56	98.62	98.33
%OptD	4.3	6.4	5.5	5.9	3.7	6.8

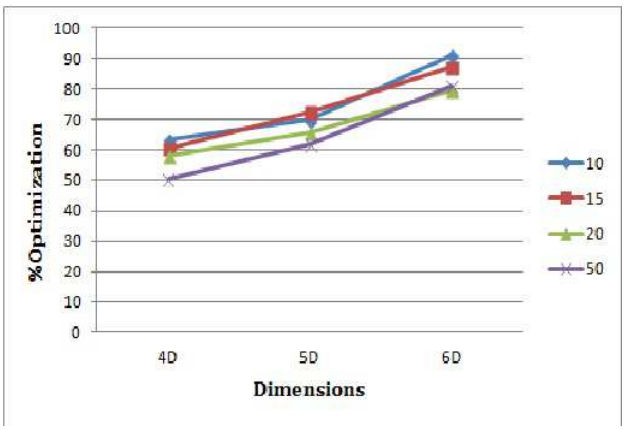

Figure 4.12: %Optimization in d-dimensions using SLO

4.3.4.3 Effect of Increasing Vertices

Increasing the number of vertices increases the execution time slightly in a manner similar to the three dimensional case. The same reasoning applies here as the increased edges lead to increased edge combinations that rotating callipers needs to check to obtain minimum bounding boxes for the two dimensional projections. Effect of increase in execution time is indicated 4.13 where we use the four dimensional case as an exemplar for higher dimensions. However, the optimization obtained starts to decrease slightly as the vertices are increased. This can be explained by the method used to generate the random collection of points in higher dimensions. As the number of vertices increases it becomes more likely that the points settle into an annulus yielding a fairly sphere like convex d-dimensional polytope that starts becoming too symmetric for significant optimization to be obtained. For a specific shape, say the d-dimensional hyper-diamond the optimization remains in the same average range with increasing number of vertices on the convex hull.

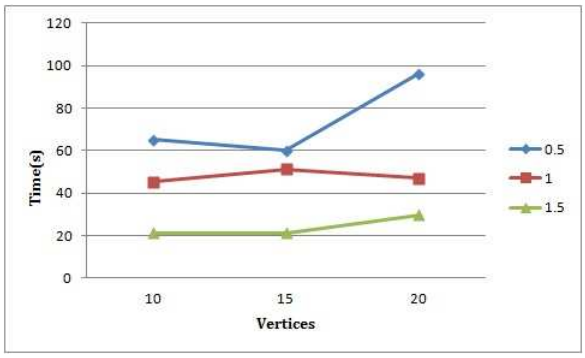

Figure 4.13: %Execution Time in 4D With Varying Vertices

4.3.4.4 Effect of Varying Heuristics

For dimensions greater than 3, the effect of the choice of the starting plane of the simultaneous local optimal comes into play. Increasing the number of dimensions increases this effect. From the tables for instances of $4-6$ dimensional polytopes we can see that Heuristic-3 gives the best result for a given d-dimensional polytope. In each table the leftmost corner value is always the lowest corresponding to the sampling with lowest granularity using Heuristic 3. The results also show how the optimization obtained increases with the increase in the number of dimensions. Since the coverage given by Heuristic-3 is the most the result reflects the advantage in optimization that it provides.

Table 4.10: Optimization With Changing Heuristics in 4D

Granularity	1	2	3	5	7	10
H1	10097.65079	10109.55	10109.43	10138.59	10127.59	10157.65
H2	10097.65079	10109.55	10109.43	10138.59	10127.59	10127.47
H3	10097.41194	10097.96	10097.96	10116.7	10097.96	10097.96

Table 4.11: Optimization of Volume With Changing Heuristics in 5D

Gran	0.5	1	2	3	5	7	10
H1	31397.74	31363.2137	31523.82	31499.76	31702.63	31910.11	31974.85
H2	31272.73	31256.6369	31403.46	31484.57	31516.01	31981.42	31621.43
H3	31256.64	31242.1895	31425.57	31450.79	31484.57	31981.42	31750.63

Table 4.12: Volume Optimization with Changing Heuristics in 6D

Granularity	1	2	3	5	7	10
H1	88284.1377	87723.4354	89272.9911	90902.5557	911327.313	94307.3415
H2	87723.4354	91005.1063	87723.4354	85014.6478	89805.4971	94187.9184
H3	84321.2178	85014.6478	84321.2178	83179.8467	86327.4354	87723.4354

4.4 Summary & Conclusion

In this chapter we postulate about the the properties of the minimum bounding box a d-dimensional polytope under projection. The properties of these k-dimensional projections specially the *Simulatneous Local Optimal* and the convergence of optimized projections to a local optimal in the associated higher dimension can be leveraged into a novel technique for finding the optimized bounding boxes of points defining arbitrary convex polytopes in d-dimensional space. The details of this method are provided and discussed. We also proposed heuristics that could be used to significantly improve the execution time of the algorithm proposed while maintaining a very high level of optimization. The comparison of these results of the method in three dimensions with the only known algorithm that guarantees a absolute minimum as well as the fastest approximation method used to generate bounding boxes with an adequate level of accuracy shows the optimization is obtained matches the maximum level possible in most cases while providing an enormous amount of speed up in execution time. Individual factors affecting the performance of the algorithm were isolated and discussed for both threes and higher dimensions, and appropriate values for these were

proposed that provide a good balance between accuracy and speed. The discussion on applying this technique for polytopes in higher dimensions show how the advantages obtained their are significantly more than for three dimensions. While optimization methods exist in 3D, no significant work has been done to solve this problem in d-dimensions excepting an approximation technique based on diameter estimation that does not improve significantly over the volume of the axis aligned boudning box for each polytope. Therefore, the technique presented fills an important gap literature whereby applications utilizing high dimensional constructs could take advantage of the approximation presented.

Chapter 5

Range Query In High Dimensional Databases Using Topological Transformation

The ability to construct minimized bounding boxes for a set of points in d-dimensional space could have important implications in a number of applications. To prove the efficacy of the optimization method propose and illustrate its importance in an application domain, we discuss in this chapter how optimized bounding boxes obtained by the algorithm presented in the previous chapter can be used to carry out range queries on high-dimensional databases. The results of also show how the technique presented improves upon existing state of the art methods in terms of the I/O and execution time.

For most large scale databases returning data points lying within a specified range efficiently is extremely important. Range queries or similarity queries based on L_1 distance are used commonly in large scale multimedia databases containing sets of feature vectors. This chapter focuses on a method to improve this functionality for queries in d-dimensional L1 space. To achieve the targeted aim of improved efficiency in terms of improved I/O and execution time, the range query is approximated with a minimal bounding box query. This query in conjunction with some heuristics is used to return equivalent results while achieving

the stated objectives.

5.1 Motivation

Extensive work has been done to carrying out range queries on high dimensional databases using a variety of indexing technique.Approximating range queries by box queries, however, and using them in conjunction with bounding box indexing techniques was recently proposed in [98]. The primary objective there is to to take advantage of the alignment of the edges of the box queries with the bounding rectangles used to index data in techniques like the R-Tree [46] and R*-Tree [102]. It takes advantage of the fact the range query in two dimensional L1-space describes a regular diamond and rotating it by 45 deg results in an exact box which can then be applied to the data points in the database which have been similarly transformed. For higher dimensions the data was decomposed into a series of disjoint two dimensional planes on which the same 45 deg transformation was applied and dimensions not divisible by 2 left as is. The resulting querying process provides a significant improvement in I/O time over existing state of the art methods used to carry out standard range queries. However, while the transformation is exact in the two dimensions, for $d > 2$ the bounding box constructed using disjoint planar rotations is quite inefficient in terms of the empty space encapsulating the actual range query points. This empty space can be removed by heuristics proposed in [98] such that the effect on the I/O is minimized. But the performance of the algorithm in terms of the execution time is significantly affected by the amount of empty space/false positives that have to be removed. Since this space increases exponentially with increase in dimensions, minimizing it such that the improvement in I/O is still maintained becomes a major issue.

In this chapter we show how using a Minimum Bounding Box (MBB) transformation instead of the the disjoint planar rotation transformation presented in [98] to implement range queries in d-dimensional space can result in significant performance optimization in terms of execution time while matching or slightly improving upon the I/O optimization over standard range queries.

5.2 Methodology

For a set of d-dimensional feature vector or data records the range query with a given radius r at centered at a point p can be formally given by,

$$R(p) = \{q | q \in D \wedge d(p,q) \leq r\} \tag{5.1}$$

where, $d(p,q)$ is the distance function that serves as the measure of dissimilarity between the points. In the case of the application under discussion this function is the L_1 distance measure which is formally defined as,

$$d(p,q) = L_1(p,q) = |q_1 - p_1| + \cdots + |q_d - p_d| \tag{5.2}$$

where, $p_i (1 \leq i \leq d)$ is i^{th} dimension of point p.

Using the L_1 distance measure to carry out the range query results in an interesting side effect. The range query forms a convex cross-polytope in d-dimensional space. For three dimensions this a regular octahedron as shown in fig.5.2. From [98] we can extrapolate that using axis aligned bounding boxes to represent range queries minimizes collision with the

similarly axially aligned bounding boxes of the index nodes of an indexing technique such as the R*-Tree.

As mentioned previously to execute queries efficiently in large databases, the data present is referenced by a multidimensional index. Queries are then applied to the index rather than the data itself. This index can be created using a variety of techniques depending upon the requirement and application. A common indexing technique used for high-dimensional database querying is the R-Tree [46] based indexing method the R*-Tree [102]. This method utilizes d-dimensional bounding boxes to efficiently index d-dimensional data points. The efficiency of an indexing technique like the one just mentioned is measured in terms of the number of data pages (amount of IO) accessed in the index tree during the execution of a given query as well the time it takes to return the data satisfying the given range.

The bounding boxes for arbitrary d-dimensional polytopes returned by the simultaneous local optimal algorithm are axis-aligned and provide a very high level of optimization in terms of reducing the empty space (false positives in case of a range query) as shown. Therefore, using the SLO-Algorithm we can come up with a transformation T for a d-dimensional range query such that its axis-aligned bounding box is minimal. This bounding box when applied as a query to an indexed d-dimensional database (the contents of which have also been rotated by T) results in significant performance optimization. For d dimensions this minimized bounding box query (MBB) with range $r_i = [min_i, max_i]$ in dimension i encapsulating a range query R with radius r is given by,

$$B(r_i, r_2, ..., r_d) = \left\{ q \;\middle|\; \begin{array}{l} q \in D \\ \wedge \\ min_i \leq q_i \leq max_i \text{ for } 1 \leq i \leq d \end{array} \right\} \tag{5.3}$$

where D is the set of all d-dimensional data points in the database.

The false positives still present after the optimal transformation can be removed by heuristics similar to the ones proposed in [98] as shall be discussed subsequently.

5.3 Topological Transformation in Multi-Dimensional Space

Using bounding box queries in lieu of range queries, requires transformation of both the data space and the range queries themselves. The space transformation is performed offline on the entire data set in the database, before constructing the index. The query transformation is performed on the fly as each range query is submitted. The key factor determining the validity of this transformation is the satisfaction of the property that the result of the transformed query over the transformed database is either equal to, or at the least a superset of, the result of the original query over the original database. It should not miss any results which could have been returned by the original. In this regard using the minimal bounding box query is a direct extrapolation, since by definition it encapsulates the entire polytope that a submitted range query defines in d dimensions. Using the bounding does introduce false positives (i.e., the points that do not satisfy the original query but do satisfy the transformed query). However, as long as there are no false negatives (i.e, the points that are

in the original query but are not in the transformed query) the validity of the transformation is maintained.

As mentioned previously, using the disjoint planar rotational transformation for dimensions greater than two results in bounding boxes that have a large amount of empty space acting as false positives. The amount of extra space increases as the dimensionality increases, leading to the requirement of more CPU time to prune the extra space. This space is removed using a set of heuristics that while ensuring the improvement in I/O cause an increase in the CPU time expended. The time taken is directly proportional to the amount of false positives present. The greater the space the greater the amount of CPU time expended in determining how much of the transformed data intersects with the bounding rectangles of the index. In this section, we introduce a topological transformation algorithm utilizing the simultaneous local optimal algorithm to minimize the bounding box querying time for d-dimensional range queries, for $d > 2$.

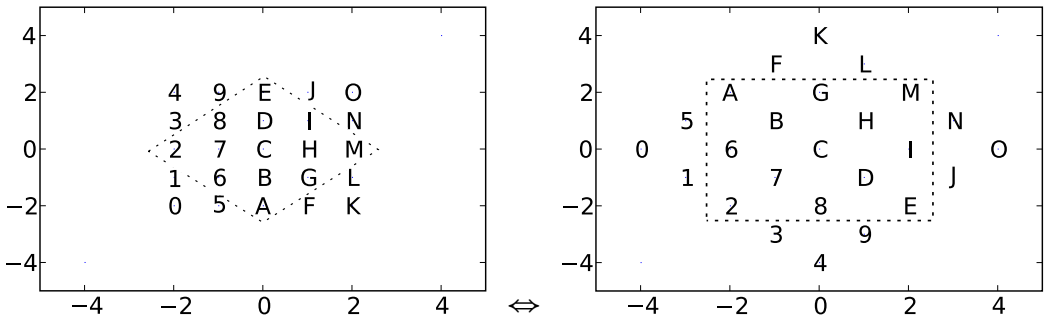

Figure 5.1: Transformation of Range Query

5.3.1 Topological Transformation For d ≤ 3

In the two dimensional case presented in [98], the transformation obtained is exact in the sense that after application of the transformation the range query is exactly mapped to

the box query with no extra space.false positives that need to be removed. This can be exemplified by the example given in fig.5.1 [98].

The edges of the range query follow the line vectors $\langle 1, 1\rangle$ and $\langle -1, 1\rangle$. If we align the query space's axes to align with these vectors $\langle 1, 1\rangle$ and $\langle 1, -1\rangle$ instead of the unit vectors $\langle 1, 0\rangle$ and $\langle 0, 1\rangle$, the query space is transformed into the space shown in **??**. The minimum bounding box given by the box query-2, 2-2, 2 precisely defines the original range query in 5.1 [98].

The in-exactness comes into play for dimensions greater than two. Finding an exact transformation that would result in an axis aligned bounding box for the cross-polytope representing a range query is non-trivial and results in a dimensional blow-up which is infeasible for building efficient indices. Therefore, to decrease overlap with the bounding boxes of the index an axis aligned bounding box can be used to encapsulate the range query.

For $d > 3$ there exist methods [15], [11] for finding the exact minimum bounding box of a cross-polytope.

For example, the 3D regular diamond shown in 5.2 is in the up-right position (centered at the origin with each of its vertices on the coordinate planes), said vertices being i.e. (1,0,0), (-1,0,0), (0,1,0), (0, -1,0), (0,0,1) and (0,0,-1) The optimal bounding box for this 3D cross-polytope can be obtained by rotating parallel to the XY-plane by 45°and then rotating clockwise by 54.7327°parallel to the YZ-plane [15]. However, when $d > 3$, there are no such algorithms for finding optimal bounding boxes in prior literature. Thus, for $d > 3$ SLO can used to find an optimal bounding box. We discuss the transformations and the steps involved in this process subsequently. In the d-dimensional case, the query is a d-dimensional cross-polytope and if it is optimally oriented then, from Theorem.4.1.1 all of it's 2-D projections are simultaneous local optimal. We use this property in the SLO-Algorithm to to compute

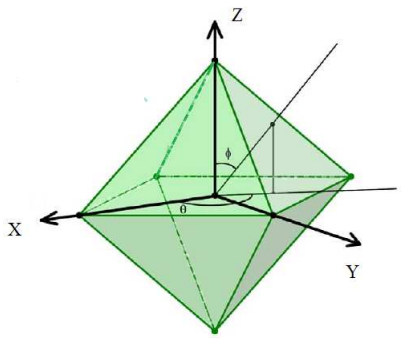

Figure 5.2: 3D-diamond in upright position

an optimized bounding box. In this section, we present an algorithm that computes an optimized axis aligned bounding box for a d-dimensional cross polytopes.

5.3.2 Topological Transformation for d > 3 Using SLO

For dimensions greater than three, SLO can be used to determine the minimal bounding box, which can be used as the querying construct instead of the standard range query. Mathematically, $Rp_1, p_2, ..., p_d$ denotes the set of all the points that are within a specified range r based on L_1 distance from the point $q = (q_1, q_2, ..., q_d)$ where the query is centered and q_i gives the coordinate in the i-th dimension. Geometrically, all the points in $Rp_1, p_2, ..., p_d$ form a cross-polytope or hyper-diamond with $2n$ points: $(p_1 + r, p_2, ..., p_d)$, $(p_1 - r, p_2, ..., p_d)$,$(p_1, p_2 + r, ..., p_d)$, $(p_1, p_2 - r, ..., p_d)$,....., $(p_1, p_2, ..., p_d + r)$, $(p_1, p_2, ..., p_d - r)$. Given this range query, its associated axis aligned bounding box with 2^d endpoints forming the corresponding box query is given by,

$$BQ = R(min_1 : max_1, min_2 : max_2, ..., min_d : max_d). \tag{5.4}$$

which gives the range in each dimension for the set of points defined by $Rp_1, p_2, ..., p_d$.

Using the SLO-Algorithm we can isolate a transformation T^d that when applied to $R(p_i)$ results in its axis aligned bounding box becoming minimal. To simplify the extraction of the transformation for a dimensions d, the initial orientation of the d-dimensional cross-polytope corresponding to $Rp_1, p_2, ..., p_d$ can be given by the $2d \times d$ matrix R. After applying SLO to R, the final optimal orientation of R will be given by the matrix R'. Based on the final orientation of the polyhedron given by the algorithm, we can obtain the transformation matrix, which then can be applied directly to transform the corresponding d-dimensional space. Formally for a given dimension, the topological transformation matrix T^d is given by

$$T^d = (R)(R'^{-1}) \tag{5.5}$$

where the matrix T^d gives the transformation in dimension d.

Since for a dimension d every range query forms a cross polytope with the same initial orientation, the transformation need only be calculated once. Once calculated it is then applied to the entire database offline and an index built up from the transformed data. Thus, for a given d-dimensional query RQ in the original space the corresponding optimal orientation RQ' will be given by $RQ' = RQ * T^d$. The bounding box for this orientation can be calculated using 5.4. From this it is also obvious that T^d is invertible. The inverse transformation $T^{d(-1)}$ can then be used return data values returned from the query on the fly.

This is exemplified by taking the unit three dimensional range query given by,

$$R = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

optimizing it using SLO. From 5.5, the transformation matrix would be,

$$T = \begin{bmatrix} -0.4082 & -0.7071 & -0.5774 \\ 0.8165 & -0.0000 & -0.5774 \\ 0.4082 & -0.7071 & 0.5774 \end{bmatrix}$$

and applied to P giving the final orientation of the polyhedron given by,

$$R = \begin{bmatrix} -0.4082 & -0.7071 & -0.5774 \\ 0.4082 & 0.7071 & 0.5774 \\ 0.8165 & -0.0000 & -0.5774 \\ -0.8165 & 0.0000 & 0.5774 \\ 0.4082 & -0.7071 & 0.5774 \\ -0.4082 & 0.7071 & -0.5774 \end{bmatrix}$$

The corresponding minimal bounding box for this query,

MBBQ = -0.8165:0.8165, -0.7071:0.7071, -0.5774:0.5774

equivalent to the range in each dimension is used as a box query applied to the data set that has been rotated by the same transformation.

We have computed the volumes of optimized bounding box for a simple d-dimensional cross-polytope where the polytope is centered at the origin and has a unit radius in L1 norm. Table. 5.1 compares the volumes of the bounding boxes for d-dimensional cross-polytope for the the standard range query, the method using DPR and the method using SLO. The table shows that SLO provides enormous optimization with increasing dimensions up to 99.99% better than the bounding box volume over the bounding box constructed using the DPR approach in 16 dimensions. This corresponds to an equivalent optimization in the amount of false positives that need removal.

5.3.2.1 Tranformation Property

Proposition 5.3.1. *For any two points in d-dimensional space* $p(d_1, d_2, ..., d_d)$ *and* $q(d_1, d_2, ..., d_d)$ *if* $D(p, q)$ *measures is the function defining their distance in original space then* $D(p, q) = D(T^d(p), T^d(q))$ *where* $D(T^d(p), T^d(q))$ *defines the distance in the transformed space where* T^d *is the transformation in dimension d.*

Proof. The proof follows from the fact that all the transformations involved in the SLO process are rotations. Each rotation in the m 2D-planes being an isometry, the distance between any two points in the database being transformed would be maintained before and after the transformations. Since the final transformation T^d is a composition of these rotations [101], T^d would also be distance preserving. □

This property can leveraged into a modified version of the heuristic proposed in [98] to remove the false positives still present after the minimization obtained through the SLO.

5.3.2.2 Transformation Heuristic

The heuristic proposed takes advantage of the observation that if the distance between the range query center and a bounding hyper-rectangle of the index tree can be estimated, the branches of the tree that do not contain any actual matches can be removed from consideration.

Proposition 5.3.2. ***Heuristic:*** *Given an index bounding box B overlapping with the query box Q with range r and query center q, if the distance between q and the closest point p in B is greater than r then B is removed from consideration.*

The closest point for any bounding box would lie on its boundary. Since boundary information for a specific bounding rectangle is provided in the in the corresponding index node, it could be checked without increasing the I/O. Assuming that b is the closest point in the in the overlapping index bounding box B to query center q_c. Using the above heuristic, we now formally the define minimal bounding box query ($MBBQ$) with range r as,

$$MBBQ = \left\{ q \;\middle|\; \begin{array}{l} q \in D \\ \wedge \;\; (min_i \leq q_i \leq max_i \\ \text{for } 1 \leq i \leq d) \;\; \wedge \\ \text{D(p,q)} \leq r \end{array} \right\} \tag{5.6}$$

Including this heuristic as part of the query removes the false positives present while resulting in greater efficiency for query execution. The efficiency is only valid as long as we ensure that no point satisfying the range query is removed from consideration after transformation T.

Theorem 5.3.3. *For every d-dimensional data point p that satisfies the range query* $R(q_1, q_2, ..., q_d)$, *p is also satisfies the minimized bounding box query* $B(T^d(q_1, q_2..., q_d))$ *contained in the result of the PBQ.*

Proof. Assume that range query R with range r is centered at point $q = (q_1, q_2..., q_d)$ in L_1 space. Also, there exists a point p such that, $p \in R(q)$ but p' is not contained in the box query b, where $p' = T^d(p)$.

This holds true only if the index bounding box B' containing p' was removed by the minimum bounding box query at some point due to the specified heuristic, i.e. the estimated distance between B' and q' ($q' = T^d(q)$) was greater than r. Let $s' = T^d(s)$ be the closest point in B' to q'. This implies that while s' was the closest point to q' in the transformed domain, s was not the closest point to q in the original data domain. This contradicts the distance preservation property defined in proposition 5.3.1. Hence, such a point p does not exist. In other words, result set returned by the $MBBQ$ contains all the points from the one returned by the original range query. □

Using the optimized orientation of the d-dimensional cross-polytopes and the associated transformation we carried out range queries on the test databases. Each range query was implemented as a MBBQ by using its axis-aligned bounding box as the querying construct.

Table 5.1: Volume Optimization for Bounding Box for Unit Hyper-Diamond

Dim	4	6	8	10	12	16
RQ	16	64	256	1024	4096	32768
DPR	4	8	16	32	64	256
SLO	1.005	0.9473	0.3076	0.1354	0.0435	0.0022
SLO-DPR(%)	74.875	88.1575	98.0775	99.57688	99.93203	99.99914

5.3.3 Performance Evaluation

5.3.3.1 Experimental Setup

The experiments were conducted on uniformly distributed synthetic datasets. The data points were normalized to unit (hyper)cube. The page size of for the index nodes was 4K bytes. All results presented here are based on averaging the I/O of one hundred random queries. Similarly, the run time is measured for a set of one hundred queries with equivalent ranges for each method. All the experiments were run on AMD Opteron 2.2 GHz systems running GNU/Linux. In the experiments discussed here the R*-tree [8] is used. However, other indexing schemes may also be used for these experiments with the caveat that they used axis-aligned bounding boxes. R*-tree index nodes contain minimum bounding rectangles for the child nodes and both range and bounding box queries can be implemented using this index structure.The labels used for various methods in the figures and tables are as follows: RQ - traditional range query on an R*-Tree, PBQ - Pruning Box Query on R*-Tree and MBBQ - Minimized Bounding Box Query.

5.3.3.2 Effect of Increasing Dimensions

Tables-5.2,5.3, 5.4 and 5.5 show the run time performance of range queries, for query ranges varying from 0.1 through 0.4. The queries are made using standard range queries(RQ), the pruning bounding box (PBQ) query using the DPR presented in [98] and the minimized bounding box query (MBBQ) of the SLO algorithm. For the data presented the database size is fixed at 1 million feature vectors. From the tables it is clear the performance in terms of run time increases significantly with the increase in the number of dimensions in comparison to the PBQ as well as the standard range query. This is explained by the fact that amount of

false positives increases exponentially with the increasing dimensions if the DPR are used to derive the optimizing transformation as shown in table.5.1. Removing these false positives increase CPU time expended as the heuristics utilized perform inverse transformations and comparisons for all points satisfying the box query. Since the number of false positives are minimized in the MBBQ and it does not need to do an inverse transformation to apply the heuristic involved, it takes less time to run while providing the same results. The improvement in time of the PBQ over RQ is explained by the improvement in I/O which improvement is matched by the MBBQ. From the table it can seen that this improvement is about 33% for 20 dimensions.

Table 5.2: Run Time in Seconds with Increasing Dimensions for a Range of 0.1

Dimensions	RQ	PBQ	MBBQ	%Opt MBBQ over PBQ
4	6.253	5.337	4.979	6.707888327
8	22.845	19.978	17.8003	10.90049054
12	60.694	47.13	39.683	15.80097602
15	144.703	108.96	83.899	23.00018355
16	174.184	137.428	104.72	23.80009896
20	359.472	226.924	152.294	32.88766283

Table 5.3: Run Time in Seconds with Increasing Dimensions For a Range of 0.2

Dimensions	RQ	PBQ	MBBQ	%Opt MBBQ over PBQ
4	31.06	28.425	26.406	7.102902375
8	129.91	116.291	103.4989	11.00007739
12	225.259	188.384	160.711	14.68967641
15	380.213	309.939	236.864	23.57722003
16	327.556	312.873	237.47	24.10019401
20	507.16	389.854	264.71	32.10022213

Table 5.4: Run Time in Seconds with Increasing Dimensions For a Range of 0.3

Dimensions	RQ	PBQ	MBBQ	%Opt MBBQ over PBQ
4	69.058	65.498	60.978	6.900974076
8	168.735	177.676	157.634	11.28008285
12	255.748	248.304	209.071	15.80038984
15	340.951	332.459	256.392	22.88011454
16	328.724	334.064	254.222	23.90021074
20	507.803	543.857	363.04	33.24715872

Table 5.5: Run Time in Seconds with Increasing Dimensions For a Range of 0.4

Dimensions	RQ	PBQ	MBBQ	%Opt MBBQ over PBQ
4	108.212	97.213	90.116	7.30046393
8	181.111	204.418	181.727	11.10029449
12	254.755	272.036	226.605	16.70036319
15	308.778	331.397	255.6727	22.8500258
16	330.628	319.763	242.7	24.10003659
20	507.859	514.618	341.706	33.6000684

5.3.3.3 Effect of Increasing Ranges

For a changing values of range the improvement obtained is shown in the fig.The database size is still fixed at 1 million feature vectors and the number of dimensions is fixed. The figure show that the improvement over PBQ remains more or less constant with increasing range size. This can explained by the fact that the MBB optimization is dependant on the orientation of the cross-polytope defining the query and not its size. For a comparable size the false positive optimization of MBBQ over PBQ remains constant. Thus, this indicates that this improvement percentage is independent of the values of query ranges.

5.3.3.4 Effect of Increasing Database Size

5.6 gives the performance improvement of Minimized Bounding Box (MBBQ) queries due to SLO over the the Pruning Box Queries (PBQ) due to DPR with increasing database size.

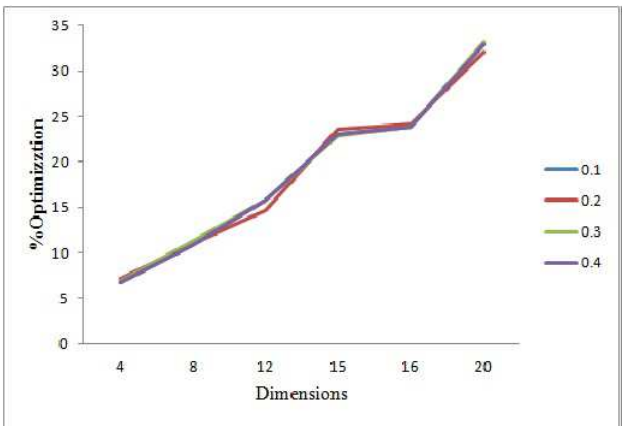

Figure 5.3: Time Optimization With Increasing Range

In this set of experiments the query range was fixed at 0.2 and the number of dimensions was set at 10. The performance of range query (RQ) in the original space is also included in the table for the purpose of comparison. We observe that the performance improvement of SLO over DPR is stable as the database size increases.

Table 5.6: Execution Time With Increase in Database Size

DB Size	0.5M	1M	2M	5M	10M
RQ	134.737	225.694	441.884	949.814	2165.921
DPR	101.1	188.384	299.31	670.336	1416.086
SLO	90.322	163.705	256.807	571.796	1192.486

5.3.3.5 I/O Optimization

The optimization obtained in terms of I/O is shown in fig.5.4. The queries are run by varying their ranges from 0.1 to 0.4 fora each type over a fixed database size of 1 million feature vectors. In terms of optimization obtained MBBQ matches the optimization obtained for the PBQ over the standard RQ. The primary I/O cost in terms of page accesses comes in to play after the application of heuristics for the removal of false positives. Both PBQ and MBBQ remove all the false postives before accessing the leaf nodes of the index tree to retrieve the

matching data points. Thus, the page accesses for both would be similar as shown in fig.5.4 and significantly better than those of range query RQ.

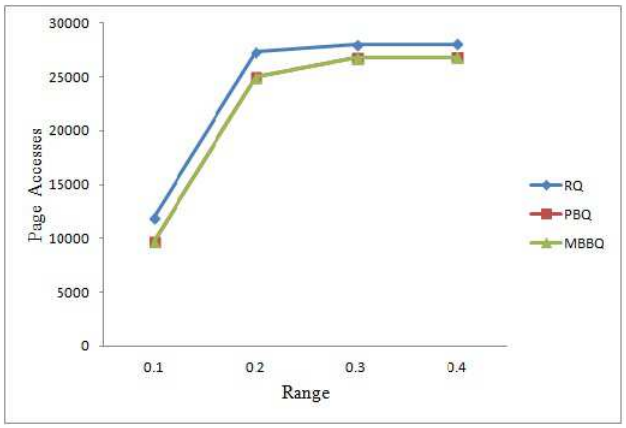

Figure 5.4: Comparison of I/O Optimization

5.4 Summary & Conclusion

In this chapter an efficient method to carry out range queries in d-dimensions is presented. The minimal bounding box queries presented here can implemented pretty quickly for applications like content based audio database indexing. For a specific dimension the transformation required need only be calculated once. Combined with the fact that the distance preserving nature of the rotational transformation prevents the need for the calculation of inverse transformations during false positive removal. This has an additional impact on the performance improvement to add to the significant optimization obtained from the removal of false positives during the SLO process. All of these results illustrate the interesting impact and implications of using optimized bounding boxes obtained from the simultaneous local algorithm in real world applications.

Chapter 6

Conclusion

In this thesis we identify a number of interesting properties for arbitrary d-dimensional polytopes and their projections. We prove the existence of the face coincidence property as an additional constraint for the existence of a minimum volume bounding box for a regular cross-polytope in three dimensions. We also discuss the characteristics of the *d-1* dimensional projections of these polytopes when encapsulated by a minimum bounding box. Based on these characteristics and the structural symmetry of the cross-polytope, we postulate about the forms the convex hulls of those projections. These results are then used to isolate the unique set of two dimensional projections that correspond to the orientation of the cross-polytope in three dimensions. This in conjunction with the other results proved the optimal orientation of a three dimensional polytope can be isolated in terms of its projections. Furthermore, these techniques can also be applied to identify the bounding boxes of other three dimensional polytopes such as icosahedrons or dodecahedrons. Using the reduction process with projections in two dimensions for optimization means that the calculations and determination of projections involved are relatively simple and the process can be completed quickly.

We subsequently discuss the properties of the minimum bounding box of a d-dimensional polytope under projection. The properties of these k-dimensional projections specially the *Simulatneous Local Optimal* and the convergence of optimized projections to a local optimal in the associated higher dimension can be leveraged into a novel technique for finding the

optimized bounding boxes of points defining arbitrary convex polytopes in d-dimensional space. The details of this method are provided and discussed. We also proposed heuristics that could be used to significantly improve the execution time of the algorithm proposed while maintaining a very high level of optimization. The comparison of these results of the method in three dimensions with the only known algorithm that guarantees a absolute minimum as well as the fastest approximation method used to generate bounding boxes with an adequate level of accuracy shows the optimization is obtained matches the maximum level possible in most cases while providing an enormous amount of speed up in execution time. Individual factors affecting the performance of the algorithm were isolated and discussed for both threes and higher dimensions, and appropriate values for these were proposed that provide a good balance between accuracy and speed. The discussion on applying this technique for polytopes in higher dimensions show how the advantages obtained their are significantly more than for three dimensions. While optimization methods exist in 3D, no significant work has been done to solve this problem in d-dimensions excepting an approximation technique based on diameter estimation that does not improve significantly over the volume of the axis aligned bounding box for each polytope. Therefore, the technique presented fills an important gap literature whereby applications utilizing high dimensional constructs could take advantage of the approximation presented.

While the discussion focuses on polytopes, the results are applicable to any set of points in d-dimensions. Bounding boxes for points defining curves or spheres in d dimensional space can also be obtained using a similar methodology. The only differentiating factor would be the choice of the optimizing algorithm in two dimensions. This algorithm would replace the rotating callipers used here to find the exact minimum bounding box in 2D.

As discussed earlier these results could have significant usage in high dimensional database

indexing and computational geometry. An implementation of the algorithm to range queries in high dimensional database indexing is given. The topological transformation presented based on the optimization algorithm (in addition to the interesting heuristics utiized) is shown to provide significant performance improvement over the existing state of the art technique used to carry out range queries on high dimensional indexes. The minimal bounding box queries presented here can implemented pretty quickly for applications like content based audio database indexing. For a specific dimension the transformation required need only be calculated once. Combined with the fact that the distance preserving nature of the rotational transformation prevents the need for the calculation of inverse transformations during false positive removal. This has an additional impact on the performance improvement to add to the significant optimization obtained from the removal of false positives during the SLO process

Since bounding boxes are inherent to a number of other applications such as collision detection algorithms, physically-based modeling, robotics, animation, computer-aided design, manufacturing, and computer simulated environments, the optimized transformation could also be applied in such applications to provide a commensurate performance improvement. The tightness of the bounding box used to approximate the underlying abstraction or object is an important performance constraint in these applications. Thus, having a minimum axis-aligned bounding box results in an automatic improvement in the performance achieved irrespective of the application.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Har-Peled, *Geometric approximation algorithms*, vol. 173. Amer Mathematical Society, 2011.

[2] O. D. Faugeras and J. Ponce, "Prism trees: a hierarchical representation for 3-d objects," in *Proceedings of the Eighth international joint conference on Artificial intelligence - Volume 2*, (San Francisco, CA, USA), pp. 982–988, Morgan Kaufmann Publishers Inc., 1983.

[3] Y. Zhou and S. Suri, "Algorithms for a minimum volume enclosing simplex in three dimensions," *SIAM J. Comput.*, vol. 31, pp. 1339–1357, May 2002.

[4] P. M. Hubbard, "Collision detection for interactive graphics applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, pp. 218–230, 1995.

[5] H. Samet, *Foundations of multidimensional and metric data structures*. The Morgan Kaufmann series in computer graphics and geometric modeling, Elsevier/Morgan Kaufmann, 2006.

[6] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, SIGMOD '84, (New York, NY, USA), pp. 47–57, ACM, 1984.

[7] N. Roussopoulos and D. Leifker, "Direct spatial search on pictorial databases using packed r-trees," in *Proceedings of the 1985 ACM SIGMOD international conference on Management of data*, SIGMOD '85, (New York, NY, USA), pp. 17–31, ACM, 1985.

[8] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The r*-tree: an efficient and robust access method for points and rectangles," in *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, SIGMOD '90, (New York, NY, USA), pp. 322–331, ACM, 1990.

[9] G. Toussaint, "Solving geometric problems with the rotating calipers," 1983.

[10] F. P. Preparata and M. I. Shamos.

[11] J. O'Rourke, "Finding minimal enclosing boxes," *International Journal of Parallel Programming*, vol. 14, pp. 183–199, 1985. 10.1007/BF00991005.

[12] G. Barequet and S. Har-Peled, "Efficiently approximating the minimum-volume bounding box of a point set in three dimensions," in *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '99, 1999.

[13] C. Ericson, *Real-Time Collision Detection*. No. v. 1 in Morgan Kaufmann Series in Interactive 3D Technology, Elsevier, 2005.

[14] C.-T. Chang, B. Gorissen, and S. Melchior, "Fast oriented bounding box optimization on the rotation group so(3,ℝ)," *ACM Trans. Graph.*, vol. 30, pp. 122:1–122:16, Oct. 2011.

[15] S. Shahid, S. Pramanik, and C. B. Owen, "Minimum bounding boxes for regular cross-polytopes," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, (New York, NY, USA), pp. 879–884, ACM, 2012.

[16] G. Barequet and S. Har-Peled, "Efficiently approximating the minimum-volume bounding box of a point set in three dimensions," *J. Algorithms*, vol. 38, no. 1, pp. 91–109, 2001.

[17] C. Chan and T. S.T., "Determination of the minimum bounding box an iterative approach," *Computers and Structures*, vol. 15, pp. 1433–1449, 2000.

[18] G. Barequet, B. Chazelle, L. J. Guibas, J. S. Mitchell, and A. Tal, "Boxtree: A hierarchical representation for surfaces in 3d," *Computer Graphics Forum*, vol. 15, no. 3, pp. 387–396, 1996.

[19] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: a hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, (New York, NY, USA), pp. 171–180, ACM, 1996.

[20] D. Dimitrov, C. Knauer, K. Kriegel, and G. Rote, "Bounds on the quality of the pca bounding boxes," *Comput. Geom. Theory Appl.*, vol. 42, pp. 772–789, Oct. 2009.

[21] D. Dimitrov, C. Knauer, K. Kriegel, and G. Rote, "On the bounding boxes obtained by principal component analysis," 2006.

[22] D. Dimitrov, C. Knauer, K. Kriegel, and G. Rote, "Bounds on the quality of the pca bounding boxes," *Comput. Geom. Theory Appl.*, vol. 42, pp. 772–789, Oct. 2009.

[23] S. Har-Peled, "A practical approach for computing the diameter of a point set," in *Proceedings of the seventeenth annual symposium on Computational geometry*, pp. 177–186, ACM, 2001.

[24] A. M. Macbeath, "A compactness theorem for affine equivalence-classes of convex regions," *J. Maths*, vol. 3, no. 1, pp. 54–61, 1951.

[25] M. Lahanas, T. Kemmerer, T. Kemmerer, N. Milickovic, D. Baltas, K. Karouzakis, D. Baltas, N. Zamboglou, O. Germany, M. Lahanas, and K. Offenbach, "Optimized bounding boxes for three-dimensional treatment planning in brachytherapy," 2000.

[26] P. Ciaccia, "Multimedia data indexing," pp. 1804–1808.

[27] Y. Sakurai, M. Yoshikawa, and S. Uemura, "Spatial indexing by virtual bounding rectangles for high-dimensional data," in *Information Organization and Databases* (K. Tanaka, S. Ghandeharizadeh, and Y. Kambayashi, eds.), vol. 579 of *The Springer International Series in Engineering and Computer Science*, pp. 267–279, Springer US, 2000.

[28] I. Kamel, "Indexing, hilbert r-tree, spatial indexing, multimedia indexing," pp. 507–512.

[29] T. Koziara and N. Biani, "Bounding box collision detection," in *In Proceedings of the 13th ACME Conference*, 2005.

[30] S. Suri, P. M. Hubbard, and J. F. Hughes, "Analyzing bounding boxes for object intersection," *ACM Trans. Graph.*, vol. 18, pp. 257–277, July 1999.

[31] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Trans. Database Syst.*, vol. 24, pp. 265–318, June 1999.

[32] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, SIGMOD '95, (New York, NY, USA), pp. 71–79, ACM, 1995.

[33] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top-k query processing techniques in relational database systems," *ACM Comput. Surv.*, vol. 40, no. 4, pp. 1–58, 2008.

[34] G. R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces (survey article)," *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 517–580, 2003.

[35] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top- k query processing techniques in relational database systems," *ACM Computing Surveys*, vol. 40, no. 4, pp. 1–58, 2008.

[36] G. R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces (survey article)," *ACM Trans. Database Syst.*, vol. 28, pp. 517–580, December 2003.

[37] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The x-tree: An index structure for high-dimensional data," in *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, (San Francisco, CA, USA), pp. 28–39, Morgan Kaufmann Publishers Inc., 1996.

[38] A. Kumar, "G-tree: A new data structure for organizing multidimensional data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 6, pp. 341–347, April 1994.

[39] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, (San Francisco, CA, USA), pp. 194–205, Morgan Kaufmann Publishers Inc., 1998.

[40] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces.," in *VLDB'97*, pp. 426–435, 1997.

[41] D. A. White and R. Jain, *Similarity indexing with the SS-tree*, pp. 516–523. IEEE Comput. Soc. Press, 1996.

[42] N. Katayama and S. Satoh, *The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries*, pp. 369–380. ACM Press, 1997.

[43] J. K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees.," *Inf. Process. Lett.*

[44] J. T. Robinson, "The k-d-b-tree: a search structure for large multidimensional dynamic indexes," in *Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, SIGMOD '81, (New York, NY, USA), pp. 10–18, ACM, 1981.

[45] Agarwal, de Berg, Gudmundsson, M. Hammar, , and H. J. Haverkort, "Box-trees and r-trees with near-optimal query time," *Discrete and Computational Geometry*, vol. 28, pp. 291–312, 2002. 10.1007/s00454-002-2817-1.

[46] A. Guttman, "R-trees: a dynamic index structure for spatial searching," *Proceedings of ACM SIGMOD*, pp. 47–57, 1984.

[47] P. K. Agarwal, M. de Berg, J. Gudmundsson, M. Hammar, and H. J. Haverkort, "Box-trees and r-trees with near-optimal query time," in *Proceedings of the seventeenth annual symposium on Computational geometry*, SCG '01, (New York, NY, USA), pp. 124–133, ACM, 2001.

[48] C. Procopiuc, P. Agarwal, and S. Har-Peled, "Star-tree: An efficient self-adjusting index for moving objects," in *Algorithm Engineering and Experiments* (D. Mount and C. Stein, eds.), vol. 2409 of *Lecture Notes in Computer Science*, pp. 178–193, Springer Berlin Heidelberg, 2002.

[49] R. Kurniawati, J. S. Jin, and J. A. Shepard, "Ss+ tree: an improved index structure for similarity searches in a high-dimensional feature space," pp. 110–120, 1997.

[50] K. Chakrabarti and S. Mehrotra, "The hybrid tree: an index structure for high dimensional feature spaces," in *Data Engineering, 1999. Proceedings., 15th International Conference on*, pp. 440–447, 1999.

[51] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The X-tree: An index structure for high-dimensional data," in *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 28–39, Morgan Kaufmann Publishers Inc., 1996.

[52] A. Kumar, "G-Tree: A new data structure for organizing multidimensional data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 6, no. 2, pp. 341–347, 1994.

[53] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 194–205, Morgan Kaufmann Publishers Inc., 1998.

[54] L. Arge, M. de Berg, H. J. Haverkort, and K. Yi, "The priority r-tree: a practically efficient and worst-case optimal r-tree," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, (New York, NY, USA), pp. 347–358, ACM, 2004.

[55] N. Katayama and S. Satoh, "The SR-tree: an index structure for high-dimensional nearest neighbor queries," in *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 369–380, ACM, 1997.

[56] J. K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees.," *Inf. Process. Lett.*, vol. 40, no. 4, pp. 175–179, 1991.

[57] D. A. White and R. Jain, "Similarity indexing with the SS-tree," in *Proceedings of the 12th International Conference on Data Engineering*, (Washington, DC, USA), pp. 516–523, IEEE Computer Society, 1996.

[58] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Comput. Surv.*, vol. 30, pp. 170–231, June 1998.

[59] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," *Technical report*, 1966.

[60] D. Hilbert, "Über die stetige abbildung einer linie auf ein flächenstück," vol. 38, p. 459460, 1890.

[61] C. C. Aggarwal, "On the effects of dimensionality reduction on high dimensional similarity search," in *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, (New York, NY, USA), pp. 256–266, ACM, 2001.

[62] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Trans. Database Syst.*, vol. 27, no. 2, pp. 188–228, 2002.

[63] K. V. Ravi Kanth, D. Agrawal, and A. Singh, "Dimensionality reduction for similarity searching in dynamic databases," *SIGMOD Rec.*, vol. 27, no. 2, pp. 166–176, 1998.

[64] K. Vu, K. A. Hua, H. Cheng, and S.-D. Lang, "A non-linear dimensionality-reduction technique for fast similarity search in large databases," in *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 527–538, ACM, 2006.

[65] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[66] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "The grid file: An adaptable, symmetric multikey file structure," *ACM Trans. Database Syst.*, vol. 9, pp. 38–71, March 1984.

[67] J. Orenstein, "A comparison of spatial query processing techniques for native and parameter spaces," *SIGMOD Rec.*, vol. 19, pp. 343–352, May 1990.

[68] H. Tropf and H. Herzog, "Multidimensional range search in dynamically balanced trees," *Applied Informatics*, pp. 71–77, 1981.

[69] G. Amato and P. Savino, "Approximate similarity search in metric spaces using inverted files," in *Proceedings of the 3rd international conference on Scalable information systems*, InfoScale '08, (ICST, Brussels, Belgium, Belgium), pp. 28:1–28:10, 2008.

[70] E. Chavez Gonzalez, K. Figueroa, and G. Navarro, "Effective proximity retrieval by ordering permutations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, pp. 1647–1658, September 2008.

[71] C. Gennaro, G. Amato, P. Bolettieri, and P. Savino, "An approach to content-based image retrieval based on the lucene search engine library," in *Proceedings of the 14th European conference on Research and Advanced Technology for Digital Libraries*, ECDL'10, (Berlin, Heidelberg), pp. 55–66, Springer-Verlag, 2010.

[72] S. Lang, *Linear Algebra*. New York: Springer-Verlag, 1987.

[73] S. Pramanik, A. Watve, C. R. Meiners, and A. Liu, "Transforming Range Queries To Equivalent Box Queries To Optimize Page Access," *Proceedings of the 36th International Conference on VLDB*, pp. 409–416, 2010.

[74] A. Garcia-Alonso, N. Serrano, and J. Flaquer, "Solving the collision detection problem," *IEEE Comput. Graph. Appl.*, vol. 14, pp. 36–43, May 1994.

[75] C. M. Hoffmann, *Geometric and solid modeling: an introduction*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989.

[76] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[77] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, "I-collide: an interactive and exact collision detection system for large-scale environments," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, I3D '95, (New York, NY, USA), pp. 189–ff., ACM, 1995.

[78] M. Held, J. T. Klosowski, and J. S. B. Mitchell, "Real-time collision detection for motion simulation within complex environments," in *ACM SIGGRAPH 96 Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH '96*, SIGGRAPH '96, (New York, NY, USA), pp. 151–, ACM, 1996.

[79] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," in *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, (New York, NY, USA), pp. 289–298, ACM, 1988.

[80] P. M. Hubbard, "Collision detection for interactive graphics applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, pp. 218–230, Sept. 1995.

[81] S. Suri, P. M. Hubbard, and J. F. Hughes, "Collision detection in aspect and scale bounded polyhedra," in *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, SODA '98, (Philadelphia, PA, USA), pp. 127–136, Society for Industrial and Applied Mathematics, 1998.

[82] A. Iones, S. Zhukov, and A. Krupkin, "On optimality of obbs for visibility tests for frustum culling, ray shooting and collision detection," in *Computer Graphics International, 1998. Proceedings*, pp. 256–263, IEEE, 1998.

[83] Y. Zhou and S. Suri, "Collision detection using bounding boxes: Convexity helps," in *In 8th Annual European Symposium on Algorithms (ESA 2000*, pp. 437–448, Yunhong, 2000.

[84] L. McMillan, Jr., *An image-based approach to three-dimensional computer graphics*. PhD thesis, Chapel Hill, NC, USA, 1997. UMI Order No. GAX97-30561.

[85] J.-W. Chang, W. Wang, and M.-S. Kim, "Efficient collision detection using a dual bounding volume hierarchy," in *Advances in Geometric Modeling and Processing* (F. Chen and B. Jttler, eds.), vol. 4975 of *Lecture Notes in Computer Science*, pp. 143–154, Springer Berlin Heidelberg, 2008.

[86] L. Yi, W. Yi, and X. Changqing, "Efficient collision detection based on component technique using obb trees in virtual assembly," in *Industrial Mechatronics and Automation, 2009. ICIMA 2009. International Conference on*, pp. 41–44, 2009.

[87] S. Redon, A. Kheddar, and S. Coquillart, "Fast continuous collision detection between rigid bodies," in *Computer graphics forum*, vol. 21, pp. 279–287, Wiley Online Library, 2002.

[88] S. Redon, Y. J. Kim, M. C. Lin, and D. Manocha, "Fast continuous collision detection for articulated models," in *Proceedings of the ninth ACM symposium on Solid modeling and applications*, pp. 145–156, Eurographics Association, 2004.

[89] A. Gregory, M. C. Lin, S. Gottschalk, and R. Taylor, "A framework for fast and accurate collision detection for haptic interaction," in *ACM SIGGRAPH 2005 Courses*, p. 34, ACM, 2005.

[90] S. Hu and L. Yu, "Optimization of collision detection algorithm based on obb," in *Proceedings of the 2010 International Conference on Measuring Technology and Mecha-*

tronics Automation - Volume 02, ICMTMA '10, (Washington, DC, USA), pp. 853–855, IEEE Computer Society, 2010.

[91] E. Ramrez, H. Navarro, R. Carmona, and J. D. Ramos, "Optimizing collision detection based on obb trees generated with a genetic algorithm," 2011.

[92] A. Iones, S. Zhukov, and A. Krupkin, "On optimality of obbs for visibility tests for frustum culling, ray shooting and collision detection," in *Computer Graphics International, 1998. Proceedings*, pp. 256 –263, jun 1998.

[93] J. J. Jiménez and R. J. Segura, "Collision detection between complex polyhedra," *Comput. Graph.*, vol. 32, pp. 402–411, Aug. 2008.

[94] A. Garcia-Alonso, N. Serrano, and J. Flaquer, "Solving the collision detection problem," *IEEE Comput. Graph. Appl.*, vol. 14, pp. 36–43, May 1994.

[95] P. Jimnez, F. Thomas, and C. Torras, "3d collision detection: A survey," *Computers and Graphics*, vol. 25, pp. 269–285, 2000.

[96] U. Assarsson and T. Mller, "Optimized view frustum culling algorithms," tech. rep., 1999.

[97] J.-F. Remacle, C. Geuzaine, G. Compre, and E. Marchandise, "High-quality surface remeshing using harmonic maps," *International Journal for Numerical Methods in Engineering*, vol. 83, no. 4, pp. 403–425, 2010.

[98] S. Pramanik, A. Watve, C. R. Meiners, and A. X. Liu, "Transforming range queries to equivalent box queries to optimize page access," *PVLDB*, vol. 3, no. 1, pp. 409–416, 2010.

[99] A. A. Ricardo and R. Prez-aguila, "General n-dimensional rotations," 2004.

[100] A. M. Noll, "A computer technique for displaying n-dimensional hyperobjects," *Commun. ACM*, vol. 10, pp. 469–473, Aug. 1967.

[101] K. L. Duffin and W. A. Barrett, "Spiders: A new user interface for rotation and visualization of n-dimensional point sets," in *In Proceedings of the Conference on Visualization (Los Alamitos*, pp. 205–211, IEEE Computer Society Press, 1994.

[102] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: an efficient and robust access method for points and rectangles," *SIGMOD Rec.*, vol. 19, no. 2, pp. 322–331, 1990.