1-1-2015

A Comparative Study of Dual-tree Algorithms for Computing Spatial Distance Histogram

Chengcheng Mou

University of South Florida, chengcheng@mail.usf.edu

Follow this and additional works at: http://scholarcommons.usf.edu/etd

Part of the Computer Engineering Commons

Scholar Commons Citation

http://scholarcommons.usf.edu/etd/5836

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
A Comparative Study of Dual-tree Algorithms for Computing Spatial Distance Histogram

by

Chengcheng Mou

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science Department of Computer Science and Engineering College of Engineering University of South Florida

Major Professor: Yicheng Tu, Ph.D.
Sagar Pandit, Ph.D.
Yan Zhang, Ph.D.

Date of Approval:
February 23, 2015

Keywords: Scientific Databases, Query Processing, Performance, Quad-tree, Oct-tree, k-d tree

Copyright © 2015, Chengcheng Mou
DEDICATION

I dedicate my thesis work to my family and many friends. A special feeling of gratitude to my loving parents, Yuanyun Mou and Jiali Wang, who have always loved me unconditionally and whose words of encouragement and push for tenacity ring in my ears.

I also dedicate this thesis to my girlfriend, Xiaolin, who has been a constant source of support and encouragement during the challenges of graduate school and life. I’m so grateful to have you in my life. Last but not least, I dedicate this thesis to the church who have supported me throughout the process.
ACKNOWLEDGMENTS

Foremost, I would like to express my sincerest gratitude to my master thesis advisor Dr. Yicheng Tu for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my thesis study. I would like to extend my appreciation to the committee members for their support and encouragement.
# TABLE OF CONTENTS

LIST OF TABLES  
iii

LIST OF FIGURES  
iv

ABSTRACT  
vii

CHAPTER 1 INTRODUCTION  
1

CHAPTER 2 RELATED WORK  
4
  2.1 Scientific Data Management  
4
  2.2 The $n$-body Correlation Function  
4

CHAPTER 3 PRELIMINARY  
7
  3.1 Density Map  
7
  3.2 SDH Algorithm  
7
  3.3 Implementation of Density Map  
10
  3.4 Dual-Tree Algorithm on Quad-tree and Oct-tree  
12

CHAPTER 4 COMPARATIVE ANALYSIS ON TREE STRUCTURES  
13
  4.1 Tree Structure  
13
    4.1.1 Quad-Tree  
13
    4.1.2 $k$-d Tree  
13
    4.1.3 Oct-Tree  
14
    4.1.4 Building the Tree Structures  
14
  4.2 DM-SDH Travels Among The Tree Structures  
15
    4.2.1 DM-SDH's Starting and Stopping Condition in 2D System  
17
    4.2.2 DM-SDH's Starting and Stopping Condition on 3D System  
18

CHAPTER 5 UNRESOLVABLE RATIO ANALYSIS  
21
  5.1 Overview of Our Approach  
21
  5.2 Analysis of DM-SDH Starting at Level of Rectangular Cell  
22
    5.2.1 Bucket Region  
22
    5.2.2 Coverable Regions  
24
      5.2.2.1 The First Bucket  
24
      5.2.2.2 The Second Bucket and Beyond  
28
  5.3 Analysis of DM-SDH Starting at Level of Square Cell  
36
    5.3.1 Coverable Regions  
37
      5.3.1.1 The First Bucket  
37
      5.3.1.2 The Second Bucket and Beyond  
40
LIST OF TABLES

Table 4.1  Controlled trials in 2D system. 18
Table 4.2  Controlled trials in 3D system. 20
Table 5.1  Values of $\frac{\alpha(m+1)}{\alpha(m)}$ when DM-SDH algorithm starts at rectangular cell. 33
Table 5.2  Values of $\frac{\alpha(m+1)}{\alpha(m)}$ when DM-SDH algorithm starts at square cell. 45
LIST OF FIGURES

Figure 3.1 Two density map of different resolutions on a simulated 2 dimensional space. 7
Figure 3.2 Three cases to consider the relative position of two cells that generates the maximum an minimum distances. 9
Figure 3.3 Tree structure, the p-count (number in each node), next (dotted lines), child (thin solid lines), and p-list (lines connecting to a ball). 10
Figure 4.1 Running time of building the tree structures on the same number of points 15
Figure 4.2 Quadtree and $k$-tree data structure. 16
Figure 4.3 Octtree and $k$-tree data structure. 16
Figure 4.4 Starting condition in 2D system. 17
Figure 4.5 Starting condition in 3D system. 19
Figure 5.1 Theoretical boundaries of bucket 1 and bucket 2 regions of cell A, with the bucket width $p$ being exactly $\sqrt{2}$$\delta$. 21
Figure 5.2 Illustration on how to calculate the area of inner rhombus-like region. 23
Figure 5.3 The coverable region of the first bucket, $m = 1$. 25
Figure 5.4 The coverable region of the first bucket, $m = 2$. 25
Figure 5.5 The coverable region of the first bucket, $m = 3$. 26
Figure 5.6 The coverable region of the first bucket, $m = 4$. 26
Figure 5.7 The coverable region of the first bucket, $m = 5$. 27
Figure 5.8 Inner boundaries of the coverable region with $m = 1$. 28
Figure 5.9 Inner boundaries of the coverable region with $m = 2$ and $m = 3$. 29
Figure 5.10 Inner boundaries of the coverable region with $m = 4$ and $m = 5$. 30
Figure 5.11 An illustration on how to calculate the area of region bounded by four arcs. 31
Figure 5.12 Theoretical boundaries of bucket 1 and bucket 2 regions of cell A, with the bucket width $p$ being exactly $\sqrt{2}$$\delta$. 37
Figure 5.13 The coverable region of the first bucket, \( m = 1 \).

Figure 5.14 The coverable region of the first bucket, \( m = 2 \) and \( m = 3 \).

Figure 5.15 The coverable region of the first bucket, \( m = 4 \) and \( m = 5 \).

Figure 5.16 Inner boundaries of the coverable region, \( m = 1 \).

Figure 5.17 Inner boundaries of the coverable region, \( m = 2 \) and \( m = 3 \).

Figure 5.18 Inner boundaries of the coverable region, \( m = 4 \) and \( m = 5 \).

Figure 5.19 An illustration on how to calculate the area of region bounded by four arcs in Figure 5.17 and 5.18.

Figure 6.1 Growth rate of running time of brute-force computation in 2D system

Figure 6.2 Growth rate of running time of brute-force computation in 3D system

Figure 6.3 DM-SDH algorithm travels on Quad-tree and \( k \)-d tree: case 1

Figure 6.4 The relationship between \( f(n) \) and \( g(n) \) in case 1

Figure 6.5 DM-SDH algorithm travels on Quad-tree and \( k \)-d tree: case 2

Figure 6.6 The relationship between \( f(n) \) and \( g(n) \) in case 2

Figure 6.7 DM-SDH algorithm travels on Quad-tree and \( k \)-d tree: case 3

Figure 6.8 The relationship between \( f(n) \) and \( g(n) \) in case 3

Figure 6.9 DM-SDH algorithm travels on Quad-tree and \( k \)-d tree: case 4

Figure 6.10 The relationship between \( f(n) \) and \( g(n) \) in case 4

Figure 6.11 DM-SDH algorithm travels on Oct-tree and \( k \)-d tree: case 1

Figure 6.12 The relationship between \( f(n) \) and \( g(n) \) in 3D case 1

Figure 6.13 DM-SDH algorithm travels on Oct-tree and \( k \)-d tree: case 2

Figure 6.14 The relationship between \( f(n) \) and \( g(n) \) in 3D case 2

Figure 6.15 DM-SDH algorithm travels on Oct-tree and \( k \)-d tree: case 3

Figure 6.16 The relationship between \( f(n) \) and \( g(n) \) in 3D case 3

Figure 6.17 DM-SDH algorithm travels on Oct-tree and \( k \)-d tree: case 4

Figure 6.18 The relationship between \( f(n) \) and \( g(n) \) in 3D case 4

Figure 6.19 DM-SDH algorithm travels on Oct-tree and \( k \)-d tree: case 5

Figure 6.20 The relationship between \( f(n) \) and \( g(n) \) in 3D case 5
Figure 6.21 DM-SDH algorithm travels on Oct-tree and k-d tree: case 6

Figure 6.22 The relationship between $f(n)$ and $g(n)$ in 3D case 6

Figure 6.23 DM-SDH algorithm travels on Oct-tree and k-d tree: case 7

Figure 6.24 The relationship between $f(n)$ and $g(n)$ in 3D case 7

Figure 6.25 DM-SDH algorithm travels on Oct-tree and k-d tree: case 8

Figure 6.26 The relationship between $f(n)$ and $g(n)$ in 3D case 8

Figure 6.27 DM-SDH algorithm travels on Oct-tree and k-d tree: case 9

Figure 6.28 The relationship between $f(n)$ and $g(n)$ in 3D case 9

Figure 6.29 Cache performance on Quad-tree

Figure 6.30 Cache performance on k-d tree

Figure 7.1 Performance of the DM-SDH running on Quad-tree and k-d with 2 bucket query in 2D data.

Figure 7.2 Performance of the DM-SDH running on Quad-tree and k-d with 3 bucket query in 2D data.

Figure 7.3 Performance of the DM-SDH running on Quad-tree and k-d with 3 bucket query in 3D data.

Figure 7.4 Performance of the DM-SDH running on Quad-tree and k-d with 4 bucket query in 3D data.

Figure 7.5 Performance of the DM-SDH running on Quad-tree and k-d with 7 bucket query in 3D data.
ABSTRACT

Particle simulation has become an important research technique in many scientific and engineering fields in latest years. However, these simulations will generate countless data, and database they required would therefore deal with very challenging tasks in terms of data management, storage, and query processing. The two-body correlation function (2-BCFs), a statistical learning measurement to evaluate the datasets, has been mainly utilized to measure the spatial distance histogram (SDH). By using a straightforward method, the process of SDH query takes quadratic time. Recently, a novel algorithm has been proposed to compute the SDH based on the concept of density map (DM), and it reduces the running time to $\Theta(N^{3/2})$ for two-dimensional data and $\Theta(N^{5/3})$ for three-dimensional data, respectively. In the DM-SDH algorithm, there are two types of DMs that can be plugged in for computation: Quad-tree (Oct-tree for three-dimensional data) and k-d tree data structure. In this thesis paper, by using the geometric method, we prove the unsolvable ratios on the k-d tree. Further, we analyze and compare the difference in the performance in each potential case generated by these DM-SDH algorithms. Experimental results confirm our analysis and show that the k-d tree structure has better performance in terms of time complexity in all cases. However, our qualitative analysis shows that the Quad-tree (Oct-tree) has an advantage over the k-d tree on aspect of space complexity.
CHAPTER 1
INTRODUCTION

Many scientific fields rely on computer simulation to determine and visualize the uncertainty of their researches [14], [15], [16]. These simulations usually generate large amount of spatiotemporal data. Generally, such scientific data has multiple dimensions, and the users who access scientific datasets are concerned with high-level analytical and reasoning queries [17], [18], [19]. Therefore, it places many challenges to the design of database management systems in terms of data management, storage, and query processing. As a result, scientific data management has attracted much attention in the database research community [20], [21], [22], [23]. Recent studies suggest that many efforts have been made in developing suitable data management techniques for processing scientific data [3], [4], [5], [6], [7].

Other than the problems of data storage and the expensive I/O cost due to large volumes of scientific data, the existing Database Management Systems (DBMSs) are not able to satisfy the demands of scientific analysis, because we still confront many difficulties and issues to answer the specific analytical queries [24], [25]. Scientific analysis usually requires precise mathematical computations to answer the analytical queries, and these computations usually take non-linear time. One remarkable example of scientific analysis is the $n$-body problem, which is a classical problem of predicting the individual motions of a group of particles interacting with each other mutually, and it has been broadly used in natural science [8], [26], [27], [28]. For instance, $n$-body correlation function ($n$-BCF) is statistical measurement of all $n$-point subsets of the entire dataset. In a dataset with $N$ points, $n$-BCF takes $O(N^n)$ time to compute in a straightforward method. In practice, the $n$-BCF could be extended to, such as, nearest-neighbor classification, gaussian process learning and prediction, kernel density estimation, two-point correlation.

The 2-BCF is a spatial statistic which has broad applicability within statistical learning, such as material science, astro-physics, biomedical science, etc. It can be thought of roughly as a distribution of pairwise distances among a given number of particles [10], [11], [12]. The most straightforward approach to compute
the 2-BCFs of relatively large volumes of coordinates that are generated from computer simulation is to compare each particle to each other one, and it requires quadratic time. While the algorithms of interest work on general 2-BCFs, our study concentrates on a query type named spatial distance histogram (SDH), which approximates the distribution of all pairwise distances. The SDH query can be formally stated as follows:

“Given the coordinates of $N$ points in a (2D or 3D) Cartesian coordinates system, draw a histogram that depicts the distribution of the pairwise distances between the $N$ points.” [1]

A histogram in database research field is a data structure held by DBMS to internally summarize the data and provide size estimates for queries. In this study, we specify the histogram with a parameter $l$, which is the total number of buckets. Because the dataset is generated from a simulation system with a fixed dimension (2-dimensional or 3-dimensional), the maximum distance between any two point in the system $L_{\text{max}}$ is fixed. In this study, we deal with the standard SDH, whose buckets are of the same width, the width of buckets $p = L_{\text{max}}/l$, named histogram resolution, is usually used as a parameter of a query. Specifically, with a given histogram resolution $p$, SDH retrieves the database and asks for the number of pairwise distances that fall in into ranges $[0, p), [p, 2p), [2p, 3p), ..., [(l-1)p, lp)$, respectively. Essentially, SDH discretely represents a continuous statistical distribution function, which indicates the density varies as a function of distance from a referenced point, named the radial distribution function (RDH). RDH could be used to visualize physical/chemical features of a natural system, such as pressure, tension, energy.

In a dataset with $N$ particles, 2-BCF requires $O(N^2)$ computation time by means of straightforward algorithm, which directly computes all the pairwise distances. There are some efficient SDH algorithms discussed in databases [2] and data mining research community [8], [9]. This efficient SDH algorithm only takes $O(N^{2d-1}/2^d)$ to compute the SDH query for 2- or 3-dimensional dataset, where $d$ is number of dimensions of dataset. Instead of computing each pairwise distances for all points, the main idea of these proposed algorithms is to analyze the distances between two groups of points. These groups are represented by nodes in a space-partitioning tree structure, called density map. These two density maps have been specified as Quad-tree (Oct-tree for 3-dimensional) in [2] and $k$-d tree in [8], [9]. The majority of cuts on running time is caused by the fact that the tedious brute-force computations are substituted by recursively resolving two groups of points. We are going to elaborate on the method of resolution on two groups of points in later chapter. In addition, because the resolution only proceeds on two disjoint subtrees, these
algorithms are named dual-tree algorithms [8]. The objective of this thesis is to further study and compare the performance of Quad-tree-based (Oct-tree-based in 3-dimensional data) dual-tree algorithm and $k$-d tree-based dual-tree algorithm.

This thesis is organized as follows: chapter 2 summarizes the related works done to compute SDH query, chapter 3 presents the preliminaries related to this thesis study, chapter 4 discusses the differences among the tree structures (Quad-tree, $k$-d tree, and Oct-tree), chapter 5 presents our geometric model to analyze the unresolvable ratio of DM-SDH algorithm running on the $k$-d tree, chapter 6 discusses the performance of DM-SDH algorithm running on different tree structures, chapter 7 shows the experimental results, and chapter 8 discusses the conclusion and future work.
CHAPTER 2
RELATED WORK

2.1 Scientific Data Management

The scientific research community generates vast volume of datasets that require large storage and new scientific approaches to analyze and organize the data through computer simulation. The volume of data is growing extremely fast, presumably, it doubles every year. Moving the copies of data is a costly and trivial task. As a result, the idea of scientific data center has been proposed. It is a large collection of datasets and separated from data analysis. Generally, instead of moving the raw data and its applications to the user end, scientific data center holds all raw data and pre-defined operations. It only requires to communicate the questions and answers to the users indeed.

On the other hand, scientific data requires very high precision. The quality and quantity of data are therefore competitively increasing. Other than general data, scientific data have several features: (1) the volume of dataset can be as large as petabyte and even exabyte scale; (2) data usually has complicated structures (multidimensional or distributed on a continuous domain); and (3) data-analysis queries are more complex. Nevertheless, these ever-increasing scientific databases are still built under existing relational DBMS. Thus, the former three features of scientific data introduced three corresponding challenges to DBMS: (1) I/O bandwidth not be able to catch up with storage capacity; (2) the performance of data-analysis algorithms are super-linear; and (3) more complex algorithms require more operators. Particle simulation is a concrete example of such scientific database management, which provides the particle prototype to visualize the large-scale structure of dynamical system.

2.2 The $n$-body Correlation Function

According to physics and astronomy, $n$-body simulation is a simulation of particles. It monitors the influence of the physical force, where $n$ is number of correlated particles. $n$-body correlation function ($n$-
BCF) is a distribution function that depicts the $n$-particle correlated distances within a given volume. Of the particle simulation, the configurations of particle may include types, velocities, coordinates, etc., but in this paper, we merely concern the coordinates of particle as the configurations of particle simulation, because it is the only parameter that we are using to compute the distances in the Cartesian Coordinates System. By using the brute-force algorithm, $n$-BCF computational complexity is equivalent to the number of all $n$-combinations of $N$:

$$T_{n-BCF}(N) = \binom{N}{n}$$

$$= \frac{N!}{n! \cdot (N-n)!}$$

$$\leq N \cdot (N-1) \cdot (N-2) \cdot \ldots \cdot (N-n+1)$$

$$\leq N^n$$  \hspace{1cm} (2.1)

Thus, $T_{n-BCF}(N) = O(N^n)$. As a given number of particles $N$, it requires $O(N^n)$ time complexity to compute the $n$-BCF in a straightforward method (brute-force algorithm).

The 2-body correlation function (2-BCF) also named spatial distance histogram (SDH), which describes the distribution function of pairwise distances within a given volume. It is a direct estimation of a continuous statistical distribution function, as known as radial distribution functions (RDF) [10], [11], [12]. The RDF is defined as follows:

$$g(r) = \frac{N(r)}{4\pi r^2 \delta r \rho}$$  \hspace{1cm} (2.2)

where $N(r)$ is the number of points in the space between $r$ and $r + \delta r$, $\rho$ is the average density of points in the entire system, and $4\pi r^2 \delta r \rho$ is the area of that space.

According to the thermodynamics, by using the Equation 2.2, we can easily get the physical quantities, such as total pressure:

$$p = \rho kT - \frac{2\pi}{3} \rho^2 \int dr r^2 u'(r)g(r, \rho, T)$$

and energy:

$$\frac{E}{NkT} = \frac{3}{2} + \frac{\rho}{2kT} \int dr 4\pi r^2 u'(r)g(r, \rho, T)$$
Apparently, \( g(r) \) is a vital factor that will be used by the formulae above. In this thesis, we will not go through the detail of these formulae, our goal is to discuss the efficiency of SDH in particle simulation. Correspondingly, by plugging 2-bodies into the Equation 2.1, SDH computational complexity is

\[
T_{2-BCF}(N) = \binom{N}{2} \leq N^2
\]  

(2.3)

Thus, \( T_{2-BCF}(N) = O(N^2) \). Given the number of particles \( N \), it requires \( O(N^2) \) time complexity to compute the SDH in a straightforward method (brute-force algorithm).
CHAPTER 3

PRELIMINARY

3.1 Density Map

Since the range of buckets is greater than zero, given a pair of points, if we know a range of their distance, we can determine which bucket this distance belongs to. Theoretically, as the range of bucket increases, the chance that any fixed distances will fall into a bucket increases as well. We interchangeably use the terms range of buckets, bucket width, and size of bin in this thesis paper. Other than directly calculating all point-to-point distances, we can save some running time on SDH computation by counting the number of distances that fall into a bucket.

![Figure 3.1: Two density map of different resolutions on a simulated 2 dimensional space.](image)

3.2 SDH Algorithm

The basic idea of this approach is building top of data structure as known as density map. As shown in Figure 3.1, each cell counts the points that are bounded by the four coordinates of the cell itself. The reciprocal of the cell diagonal in a density map is named as resolution. In order to proceed with SDH computation, the different resolutions of density maps have been arranged from coarse to fine by bisecting
the both x- and y-dimension on 2D data/space (all x-, y-, and z-dimension on 3D data/space). Thus, in the 2-dimensional system, a density map divides a simulated space, it’s usually a square, into four equivalent sub-squares to the next level of density map. Correspondingly, a density map divides the simulated space into eight equivalent sub-squares to next level of density map in the 3-dimensional system. For instance, the simulated space is initially divided into six cells in Figure 3.1a, each with diagonal length $2\sqrt{2}$. Then, each of the six cells have been divided into 4 sub-cells in Figure 3.1b, each with diagonal length $\sqrt{2}$. However, the number of points that are contained by a lower resolution cell XA and four higher resolution sub-cells X0A0, X0A1, X1A0, and X1A1 are identical.

As shown as Algorithm 1, the core process of the algorithm is the procedure ResolveTwoCell, which recursively resolve the two cells $m_1$ and $m_2$ on the same density map. In ResolveTwoCell, in order to check $m_1$ and $m_2$ are resolvable (line 1), we firstly compute the minimum and maximum distances between any points from $m_1$ and $m_2$. Since $m_1$ and $m_2$ are in a Cartesian coordinates system, they have a relative position, such that the minimum and maximum distance are meaningful. This process only take $\Theta(1)$ to be accomplished by computing the corner coordinates of two cells $m_1$ and $m_2$.

<table>
<thead>
<tr>
<th>Algorithm 1: DM-SDH (Tu 2009)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong>: all data points, specified density maps, and bucket width $p$;</td>
</tr>
<tr>
<td><strong>Result</strong>: an array of counts $h$ for histogram</td>
</tr>
<tr>
<td>initialize all elements in $h$ to 0;</td>
</tr>
<tr>
<td>find the first density map $DM_i$ whose cells have diagonal length $k \leq p$;</td>
</tr>
<tr>
<td><strong>for all cells in</strong> $DM_i$ <strong>do</strong></td>
</tr>
<tr>
<td>$n \leftarrow$ number of particles in the cell;</td>
</tr>
<tr>
<td>$h_1 = h_1 + \frac{1}{2}n(n-1)$;</td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
<tr>
<td><strong>for any two cells</strong> $m_j$ <strong>and</strong> $m_k$ <strong>in</strong> $DM_i$ <strong>do</strong></td>
</tr>
<tr>
<td>ResolveTwoCells ($m_j, m_k$);</td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
<tr>
<td><strong>return</strong> $h$</td>
</tr>
</tbody>
</table>

As shown in Figure 3.2, the red dash line represents the maximum distances, and blue solid line represents the minimum distance. When both minimum and maximum distances fall into a same histogram bucket $i$, we claim these two cells $m_1$ and $m_2$ are resolvable on density map $DM_i$. If it dose so, the corresponding bucket of histogram will be updated by incrementing $n_1 \cdot n_2$ times. $n_1$ and $n_2$ are number of points that bounded by $m_1$ and $m_2$, respectively. Inversely, if two cells $m_1$ and $m_2$ are not resolvable on density map $DM_i$, we move to next Density map $DM_{i+1}$ which has higher resolution, and go over the previous
Algorithm 2: ResolveTwoCells \((m_1, m_2)\)

check if \(m_1\) and \(m_2\) are resolvable;

if \(m_1\) and \(m_2\) are resolvable then

\[ i \leftarrow \text{index of the bucket } m_1 \text{ and } m_2 \text{ resolve into}; \]
\[ \text{current section becomes this one}; \]
\[ n_1 \leftarrow \text{number of particles in } m_1; \]
\[ n_2 \leftarrow \text{number of particles in } m_2; \]
\[ h_i \leftarrow h_i + n_1 n_2; \]

end

if \(n_1\) or \(n_2\) equal to 0 then

\[ \text{return} \]

end

if \(m_1\) and \(m_2\) are on the last density map then

for each particle \(A\) in \(m_1\) do

for each particle \(B\) in \(m_2\) do

\[ f \leftarrow \text{distance between } A \text{ and } B; \]
\[ i \leftarrow \text{the bucket } f \text{ falls into}; \]
\[ h_i \leftarrow h_i + 1; \]

end

end

else

\[ DM' \leftarrow \text{next density map with higher resolution}; \]

for each partition \(m'_1\) of \(m_1\) on \(DM'\) do

for each partition \(m'_2\) of \(m_2\) on \(DM'\) do

ResolveTwoCells \((m_j, m_k)\)

end

end

end

Figure 3.2: Three cases to consider the relative position of two cells that generates the maximum and minimum distances.
steps to check each of four children in $m_1$ to each of four children in $m_2$, so on and so forth. However, if $m_1$ and $m_2$ are not resolvable on the last level (highest resolution) of density map, we have to calculate the direct distances of all points that in the non-resolvable cell. In addition, if $n_1 = 0$ or $n_2 = 0$ at certain level of density map, the procedure will directly exit, because there is not any distance that crosses this two cells. The DM-SDH algorithm starts at density map while the diagonal length less and equal to bucket width.

$$\delta \leq \frac{p}{\sqrt{d}}$$

where $\delta$ is cell side length, $d$ is dimension of data (2 for 2D, 3 for 3D). Clearly, none pair of cells will be resolved at diagonal greater than bucket width. Literally, we categorize all the distances into three classes: (1) intra-node distance; (2) inter-node distance; and (3) direct distance. The intra-node distances are all resolved at once the algorithm started (DM-SDH algorithm line 4 to 5), it takes constant time. Inter-node distance are resolved by recursively call the algorithm to check their resolvability. The direct distances are resolved by directly computing the distances.

### 3.3 Implementation of Density Map

![Figure 3.3: Tree structure, the p-count (number in each node), next (dotted lines), child (thin solid lines), and p-list (lines connecting to a ball).](image)

In DM-SDH, as a given dataset, we have to build the density map before we invoke the algorithm. The pervious works construct the cells on different density map into a point region (PR) Quad-tree (Oct-tree on
The nodes in the tree contain the following information:

\[(p - \text{count}, x_1, x_2, y_1, y_2, \text{child}, p - \text{list}, \text{next})\]  

(3.1)

where, \(p - \text{count}\) represents the number of points in this cell, \((x_1, y_1)\) and \((x_2, y_2)\) are coordinates of this cell, \(\text{child}\) is a pointer that point to the first child on next level, \(p - \text{list}\) indicates the head of a list data structure which store the real particle data, \(\text{next}\) represents a pointer that chain all the siblings together at same level. Theoretically, we have the well organized density map as shown in Figure 3.3, which is exactly same dataset as Figure 3.1. In order to building the tree, we use the most straightforward space partition approach:

1) the region represented by each cell is strictly set to be a square (cube for 3D space). i.e., we have \(|x_1 - x_2| = |y_1 - y_2|\);
2) we always partition each dimension by bisecting it into exactly TWO equal segments.

Generally, SDH queries come with different bucket width \(p\), thus we need to set up a series of density maps from the coarsest resolution to the finest. At the beginning, we just set a single cell map, which covers the entire simulated space, to initial the coarsest resolution. On the other hand, how finest resolution should be in the density map? This is a subtle issue: first, given any bucket width \(p\), the percentage of resolvable cells increases with the level of the tree. However, the number of pairs of cells also increases dramatically (i.e., increase \(2^d!\) for each unresolvable pairs). In addition. Because of computing the minimum and maximum distances even more costly than direct distances, it’s not going to have any merit that to compute minimum and maximum distances for two cells. Therefore, the tree height of a density map on the Quad-tree (Oct-tree for 3D) \(H\) is set to be

\[H = \left\lfloor \log_2 \frac{N}{\beta} \right\rfloor + 1\]  

(3.2)

where \(d\) is the degree of dimensions, \(\beta\) is the average number of points in each leaf node. The tree height of a density map on the \(k\)-d tree \(H\) is set to be

\[H = \left\lfloor \log_2 \frac{N}{\beta} \right\rfloor + 1\]  

(3.3)
where $\beta$ is the average number of points in each leaf nodes. In the practical cases, we set the threshold $\beta$ equal to 4 (8 for 3D), which is the condition that we stop building the tree.

### 3.4 Dual-Tree Algorithm on Quad-tree and Oct-tree

As discussed in [1], [2], the unresolvable ratio of DM-SDH algorithm, running on Quad-tree (Oct-tree in 3D system) are based on following theorem.

**Theorem 1** “Let $DM_0$ be the first density map where the DM-SDH algorithm starts running, and $\alpha(m)$ be the percentage of pairs of cells that are not resolvable on the density map that lies $m$ levels below $DM_0$ (i.e. map $DM_m$). We have $\lim_{p \to 0} \frac{\alpha(m+1)}{\alpha(m)} = \frac{1}{2}$” (Tu 2011)

Based on the unresolvable ratio, if the particles are uniformly distributed in space, the time complexity of DM-SDH algorithm running on Quad-tree (Oct-tree in 3D system) is $\Theta\left(N \frac{2^{d-1}}{d}\right)$ where $d \in \{2, 3\}$ which is the number of dimensions of the data.
CHAPTER 4

COMPARATIVE ANALYSIS ON TREE STRUCTURES

4.1 Tree Structure

We are using two different data structures to implement the density map: Quad-tree (Oct-tree in 3D system) and $k$-d tree. In this section, we will discuss the time that are required to build Quad-tree, $k$-d tree, and Oct-tree.

4.1.1 Quad-Tree

A tree structure partitions the physical space by recursively bisecting the each of two dimensions into four quadrants. The each of internal nodes (non-leaf nodes) have four children. Building a Quad-tree takes,

$$T(n) = 4T(n/4) + \Theta(1)$$  \hspace{1cm} (4.1)

By using the master method,

$$a = 4, b = 4, \log_b a = \log_4 4 = 1, n^{\log_b a} = n^{\log_4 4} = n, f(n) = \Theta(1) = n^0$$

Since $f(n) = O(n^{1-\epsilon})$, where $\epsilon = 1$

Case 1 of master theorem applies: $T(n) = \Theta(n^{\log_b a})$ When $f(n) = O(n^{\log_b a - \epsilon})$

Thus the solution is $T(n) = \Theta(n^{\log_4 4}) = \Theta(n)$

4.1.2 $k$-d Tree

A special case of binary tree, which alternatively bisects each of the its dimensions. For instance, in 2D system, it alternatively divides $x$- and $y$-dimension into two sub-region; in 3D system, it alternatively
divides $x$, $y$, and $z$-dimension into two sub-region. Building a $k$-tree takes,

$$T(n) = 2T(n/2) + \Theta(1)$$  \hspace{1cm} (4.2)

By using the master method,

$a = 2, b = 2, \log_b a = \log_2 2 = 1, n^{\log_b a} = n^1 = n, f(n) = \Theta(1) = n^0$

Since $f(n) = O(n^{1-\epsilon})$, where $\epsilon = 1$

Case 1 of master theorem applies: $T(n) = \Theta(n^{\log_b a})$ When $f(n) = O(n^{\log_b a - \epsilon})$

Thus the solution is $T(n) = \Theta(n^{\log_2 2}) = \Theta(n)$

### 4.1.3 Oct-Tree

A tree structure partitions the physical space by recursively bisecting the each of three dimensional space into eight sub-regions. the internal nodes have eight children. Building a oct-tree takes,

$$T(n) = 8T(n/8) + \Theta(1)$$  \hspace{1cm} (4.3)

By using the master method,

$a = 8, b = 8, \log_b a = \log_8 8 = 1, n^{\log_b a} = n^1 = n, f(n) = \Theta(1) = n^0$

Since $f(n) = O(n^{1-\epsilon})$, where $\epsilon = 1$

Case 1 of master theorem applies: $T(n) = \Theta(n^{\log_b a})$ When $f(n) = O(n^{\log_b a - \epsilon})$

Thus the solution is $T(n) = \Theta(n^{\log_8 8}) = \Theta(n)$

### 4.1.4 Building the Tree Structures

Generally, when we talk about time complexity of a specific algorithm, by default, we consider that the input size approaches to infinity. These three tree structures therefore take linear time to build, if we don’t concern the input size. However, in our study, even though we have to build these tree structures before we invoke the algorithm, the time that take to build these trees are subtle: given a firmed particles number $N$, the time that are required to build them are different. We study the differences of these tree structures by assuming the points in the simulated space are uniform distributed. Consequently, these tree structures are
all balanced, and it is easily to get their running time. Quad-tree takes $\log_4 N$, and $k$-d tree takes $\log_2 N$.

$$\log_4 N = \frac{1}{2} \cdot \log_2 N$$

Therefore, the quad-tree takes the running time as half as $k$-d tree to build, as shown in Figure 4.1(a). In the 3D system simulation, Oct-tree takes $\log_8 N$, similarly,

$$\log_8 N = \frac{1}{3} \cdot \log_2 N$$

The k-d tree takes the running times as three times as Oct-tree to build, as shown in Figure 4.1(b).

![Figure 4.1: Running time of building the tree structures on the same number of points](image)

### 4.2 DM-SDH Travels Among The Tree Structures

In this section, we are going to talk about the performance of DM-SDH algorithm running on different density maps. In 2D system, we compare the performance of Algorithm 1 running on Quad-tree and $k$-d tree; in 3D system, we compare the performance of the same algorithm running on Oct-tree and $k$-d tree.

As shown in Figure 4.2(a), the simulated space (a 2D plane) has been simultaneously partitioned on $x$- and $y$-dimension, it yields a Quad-tree structure. Figure 4.2(b) shows that the simulated space has been partitioned on $x$-dimension at the first step, and both two sub-spaces have been partitioned on $y$-dimension at the next step, it yields a binary tree structure. $k$-d tree therefore introduces a interim level.
Figure 4.2: Quadtree and $k$-tree data structure.

Figure 4.3: Octtree and $k$-tree data structure.

Clearly, in the 3D system, $k$-d tree introduces two interim levels compared to the Oct-tree, shown as Figure 4.3. Figure 4.3(a) shows that the simulated space (a 3D space) has been simultaneously partitioned on $x$-, $y$-, and $z$-dimension. Figure 4.3(b) shows that the simulated space follows on $x$-, $y$-, and $z$-dimensional order to be alternatively partitioned.

The DM-SDH algorithm once started on certain level of the density map, it travels all the way to the leaf level of density map. Moreover, the bucket width $p$ determines where the algorithm to be started on the density map, and the number of particles simulated space $N$ determines the height of tree. Therefore, the algorithm might start/end at (1). Corresponding level, $k$-d tree starts/ends at $y$-dimension in 2D system, or $z$-dimension in 3D system. (2). Different level, $k$-d tree starts/ends at the interim level, $x$-dimension in 2D system, or either $x$-dimension or $y$-dimension in 3D system. In the rest of section, we are going to detail the starting/stoping condition in 2D system and 3D system.
4.2.1 DM-SDH’s Starting and Stopping Condition in 2D System

As we discussed in the earlier chapter, the DM-SDH algorithm starts at density map while the diagonal less and equal to the bucket width, consequently, the algorithm may start at different levels of Quad-tree and k-d tree. For instance, given the bucket width of a specific query, the DM-SDH algorithm scan the Quad-tree from the very top to bottom, the algorithm starts while it first hits the starting condition, bucket width less or equal too diagonal, at DM₁. However, given the same query, DM-SDH algorithm may first hit the starting condition at DM₂ of k-d tree. Intuitively, the cell diagonal of interim level on the k-d tree is more likely to satisfy the starting condition, because it lies between two contiguous levels of Quad-tree. Let’s define the starting condition more rigorously.

According to the algorithm starting condition \( R \leq p \) (i.e. \( \delta \leq \frac{p}{\sqrt{2}} \)). As shown in Figure 4.4, \( R_i \) represents the resolution at \( i \)-th level of the Quad-tree, \( R'_{2i} \) represents the the resolution at \( 2i \)-th level of the 2-dimensional k-d tree. \( P \) represents the bucket resolution. The algorithm will proceed the following 4 scenarios:

Scenario 1, when \( p \) falls into section (1) \( R_i \leq R'_{2i} \leq p \leq R_{i-1} \mid R'_{2i-2} \), the algorithm will be started at \( i \)-th level of Quadtree and \( (2i - 1) \)-th level of k-d tree.

Scenario 2, when \( p \) falls into section (2) \( R_i \mid R'_{2i} \leq p \leq R'_{2i} \leq R_{i-1} \), the algorithm will be started at \( i \)-th level of Quadtree and \( 2i \)-th level of k-d tree.

Scenario 3, when \( p \) falls into section (3) \( R_{i+1} \leq R'_{2i+1} \leq p \leq R_{i} \mid R'_{2i} \), the algorithm will be started at \( (i + 1) \)-th level of Quadtree and \( (2i + 1) \)-th level of k-d tree.

Scenario 4, when \( p \) falls into section (4) \( R_{i+1} \mid R'_{2i+2} \leq p \leq R'_{2i+1} \leq R_{i} \), the algorithm will be started at \( (i + 1) \)-th level of Quadtree and \( (2i + 2) \)-th level of k-d tree.

Since scenario 1 and 3 are described the algorithm starts at different level, and scenario 2 and 4 are described the algorithm starts at corresponding level, we can just sum them up in 2 main cases: (1)The
algorithm starts at the corresponding on Quad-tree and k-d tree, and (2) The algorithm starts at interim level that only half step ahead to the Quad-tree on k-d tree.

The height of the tree depends on number of points in simulated space, the more points system simulate, the higher tree height will be generated. However, either Quad-tree or k-d tree will cease when the average number of points in each leaf nodes is less or equal to 4. Namely, the average number of points of each level must greater than 4. Consequentially, this condition may force Quad-tree and k-d tree stop at different level. For example, given a firmed number of points in the simulated space, say 4000 points, both Quad-tree and k-d start building up the tree structure at level 0 (DM₀). On the Quad-tree, the average number of points at level 4 and level 5 are 15.625 and 3.906, respectively. Therefore, the Quad-tree ceases at 4. However, on the k-d tree, the average number of points at level 8, level 9, and level 10 are 15.625, 7.813, and 3.906, respectively. Instead of stopping at corresponding level 8, k-d tree just stopped at the level 9. Once DM-SDH started, it always travels to the last level of density map. Rigorously, as given number of points, we can easily get the height of tree by plugging the Formula 3.2 and Formula 3.3. Therefore, we also can conclude that there are two cases that the algorithm stops: (1) The algorithm stop at the corresponding on Quad-tree and k-d tree, and (2) The algorithm stop at interim level no the k-d tree.

In the 2D system, we design 4 groups of controlled trials, as shown in Table 4.1, to analyze the performance of DM-SDH algorithm running on Quad-tree and k-d tree.

<table>
<thead>
<tr>
<th>Case</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Corresponding level</td>
<td>Corresponding level</td>
</tr>
<tr>
<td>2</td>
<td>Corresponding level</td>
<td>Different level</td>
</tr>
<tr>
<td>3</td>
<td>Different level</td>
<td>Corresponding level</td>
</tr>
<tr>
<td>4</td>
<td>Different level</td>
<td>Different level</td>
</tr>
</tbody>
</table>

**4.2.2 DM-SDH’s Starting and Stopping Condition on 3D System**

In 3D system, the DM-SDH has to travel on Oct-tree and k-d tree, as shown in Figure 4.3. Because k-d partitions x-, y-, and z-dimensions step by step, k-d introduces two interim levels compared to the Oct-tree. The starting condition can be defined as following.
According to the algorithm starting condition $R \leq p$ (i.e. $\delta \leq \frac{p}{\sqrt{2}}$). As illustrated in Figure 4.5, $R_i$ represents the resolution at $i$-th level of the Quadtree, $R'_i$ represents the resolution at $2i$-th level of the 3-dimensional $k$-d tree. $P$ represents the bucket resolution. The algorithm will proceed the following 6 scenarios:

**Scenario 1**, when $p$ falls into section (1): $R_i \leq R'_{3i-2} \leq p \leq R'_{3i-3}$, the algorithm will be started at $i$-th level of Oct-tree and $(3i-2)$-th level of $k$-d tree.

**Scenario 2**, when $p$ falls into section (2): $R_i \leq R'_{3i-1} \leq p \leq R'_{3i-2} \leq R_{i-1}$, the algorithm will be started at $i$-th level of Oct-tree and $(3i-1)$-th level of $k$-d tree.

**Scenario 3**, when $p$ falls into section (3): $R_i || R'_{3i} \leq p \leq R'_{3i-1} \leq R_{i-1}$, the algorithm will be started at $i$-th level of Oct-tree and $3i$-th level of $k$-d tree.

**Scenario 4**, when $p$ falls into section (4): $R_{i+1} \leq R'_{3i+1} \leq p \leq R_i || R'_{3i}$, the algorithm will be started at $(i+1)$-th level of Oct-tree and $(3i+1)$-th level of $k$-d tree.

**Scenario 5**, when $p$ falls into section (5): $R_{i+1} \leq R'_{3i+2} \leq p \leq R'_{3i+1} \leq R_i$, the algorithm will be started at $(i+1)$-th level of Oct-tree and $(3i+2)$-th level of $k$-d tree.

**Scenario 6**, when $p$ falls into section (5): $R_{i+1} || R'_{3i+3} \leq p \leq R'_{3i+2} \leq R_i$, the algorithm will be started at $(i+1)$-th level of Oct-tree and $(3i+3)$-th level of $k$-d tree.

Scenario 1 and 4 are described the algorithm starts at different level, first interim level of $k$-d tree. Scenario 2 and 5 are described the algorithm starts at different level, second interim level of $k$-d tree. Scenario 3 and 6 are described the algorithm starts at the corresponding level. We can sum them up to 3 main cases: (1) The algorithm starts at first interim level that ahead to the Oct-tree on $k$-d tree, (2) The algorithm starts at second interim level that ahead to the Oct-tree on $k$-d tree, and (3) the algorithm starts at the corresponding on Oct-tree and $k$-d tree.

Similar with 2D system, the DM-SDH will visit the last level of density map. However, the threshold that stops further building the two tree structures is set to be 8. Also, compared to the Oct-tree, the $k$-d tree
may cease at either first interim level (right after cut on $x$-dimension) or second interim level (right after cut on $y$-dimension). Therefore, there are three cases that the algorithm will be terminated: (1) The algorithm ends at first interim level on k-d tree, (2) The algorithm ends at second interim level on k-d tree, and (3) the algorithm starts at the corresponding on Oct-tree and k-d tree. Thus, we found out 9 groups controlled trials to analyze the performance of Oct-tree and $k$-d tree, As shown in Table 4.2.

In order to analyze these cases, we have to utilize the unresolvable ratio to discuss the resolution on each tree structure. The unresolvable ratio of Quad-tree (Theorem 1) has already been discussed in paper [1], but the unresolvable ratio of $k$-d tree has not been studied yet. Thus, we are going to first study the unresolvable ratio of $k$-d tree in Chapter 5, and then we will come back for these cases in Chapter 6.

<table>
<thead>
<tr>
<th>Case 1</th>
<th>Corresponding level</th>
<th>Corresponding level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 2</td>
<td>Corresponding level</td>
<td>First interim level</td>
</tr>
<tr>
<td>Case 3</td>
<td>Corresponding level</td>
<td>Second interim level</td>
</tr>
<tr>
<td>Case 4</td>
<td>First interim level</td>
<td>Corresponding level</td>
</tr>
<tr>
<td>Case 5</td>
<td>First interim level</td>
<td>First interim level</td>
</tr>
<tr>
<td>Case 6</td>
<td>First interim level</td>
<td>Second interim level</td>
</tr>
<tr>
<td>Case 7</td>
<td>Second interim level</td>
<td>Corresponding level</td>
</tr>
<tr>
<td>Case 8</td>
<td>Second interim level</td>
<td>First interim level</td>
</tr>
<tr>
<td>Case 9</td>
<td>Second interim level</td>
<td>Second interim level</td>
</tr>
</tbody>
</table>
CHAPTER 5
UNRESOLVABLE RATIO ANALYSIS

5.1 Overview of Our Approach

Figure 5.1: Theoretical boundaries of bucket 1 and bucket 2 regions of cell A, with the bucket width \( p \) being exactly \( \sqrt{2}\delta \).

This chapter is going to discuss a geometric model, which is similar to the approach that discussed in [1], to analyze DM-SDH algorithm. Given any cell \( A \) on density map \( DM_0 \), we first quantify the area of a theoretical region that be able to contain all particles that can possibly resolve into the \( i \)-th bucket with any particle in \( A \), we named this region as \( \text{bucket } i \text{ region} \), and denote it as \( A_i \). However, other than Quad-tree, there are two cases for DM-SDH to start with, either starts at the level of square cell or the level of rectangular cell. In Section 5.2, we are going discuss the case that the algorithm starts at the level of rectangular cell. The case that algorithm starts at the level of square cell is similar to the algorithm starts at Quad-tree, and the analysis can be found in Section 5.3. In a 2D system, as illustrated in Figure 5.1, a cell...
A is a rectangle which draws with four points \(O_1, O_2, O_3, \) and \(O_4, A_1\) therefore bounded by 4 arcs and 4 line segments connected by points \(C_1\) through \(C_8\).

If the cells that are resolvable into bucket \(i\) with any subcells in \(A\) also from a region, we named such region as coverable region and denote it as \(A'_i\). Since the shape of the subcells is alternatively to be rectangle and square, when DM-SDH algorithm visits more levels of the tree, the boundary of \(A'_i\) shows as zigzag pattern, and eventually approaches to \(A_i\), as shown in Figure 5.4, 5.5, 5.6, 5.7. We define the ratio of \(\sum_i A'_i\) to \(\sum_i A_i\) as the covering factor, which is a critical quantity to measure how much area are “covered” by the resolvable cells. Namely, non-covering factor is the percentage of area that is not resolvable. i.e.,

\[
\text{non-covering factor} = 1 - \text{covering factor}
\]

5.2 Analysis of DM-SDH Starting at Level of Rectangular Cell

5.2.1 Bucket Region

Again, as shown in Figure 5.1, the bucket 1 region for cell \(A\) is connected by \(C_1\) through \(C_8\). Typically, \(C_1C_2, C_3C_4, C_5C_6, \) and \(C_7C_8\) are all line segments; \(C_2C_3, C_4C_5, C_6C_7, \) and \(C_8C_1\) are all 90-degree arcs with a same radius \(p\) and centered at \(O_2, O_3, O_4, \) and \(O_1\), respectively. Apparently, the area of this region is \(\pi p^2 + 2p\delta + p\delta + \frac{\delta^2}{2}\). The bucket 2 region of \(A\) is identical with bucket 1 region but the radii of the four arcs are \(2p\), as illustrated in Figure 5.1, bucket 2 region is connected by \(D_1\) all the way around to \(D_8\). However, if the points are too close to \(A\), they will only be resolved into bucket 1, because their distances to any points in \(A\) will always be shorter than \(p\). These points formed a region, which is connected by four arcs \(Q_1Q_2, Q_2Q_3, Q_3Q_4, \) and \(Q_4Q_1\) with a same radius \(p\) and centered at opposite corner of \(A\). The bucket 2 region should not take count of such inner region. This inner region is a football-like region \(Q_1Q_2Q_3Q_4\) (in the Figure 5.1), and is fourfold of the area of region \(Q_4Q_1D\), as illustrated in Figure 5.2. To get area of \(\hat{Q}_4Q_1D\), we first calculate the area of sector \(Q_4Q_1O_3\),

\[
\angle Q_4O_3Q_1 = \frac{\pi}{2} - \angle Q_4O_3F - \angle Q_1O_3A = \frac{\pi}{2} - \arcsin \frac{\delta}{4p} - \arcsin \frac{\delta}{2p}
\]
Figure 5.2: Illustration on how to calculate the area of inner rhombus-like region.

\[
S_{\overline{Q_4Q_1O_3}} = \frac{\pi}{2} - \arcsin \frac{\delta}{4p} - \arcsin \frac{\delta}{2p} \cdot p^2
\]

then, take away the area of region \(\triangle Q_4O_3B\) and \(\triangle Q_1O_3C\).

\[
\begin{align*}
S_{\triangle Q_4O_3B} &= \frac{\delta}{8} \sqrt{p^2 - \left(\frac{\delta}{4}\right)^2} \\
S_{\triangle Q_1O_3C} &= \frac{\delta}{4} \sqrt{p^2 - \left(\frac{\delta}{2}\right)^2}
\end{align*}
\]

Notice that, by doing the subtraction, we subtract the quadrilateral twice, and only once for each of two triangles. Thus, we have to put them back by adding only once the area of rectangle \(O_3BDC\). Obtains,

\[
S_{Q_1Q_2Q_3Q_4} = 4S_{\overline{Q_4Q_1D}} = 4(S_{\overline{Q_4Q_1O_3}} - S_{\triangle Q_4O_3B} - S_{\triangle Q_1O_3C} + S_{O_3BDC}) = 2 \left(\frac{\pi}{2} - \arcsin \frac{\delta}{4p} - \arcsin \frac{\delta}{2p}\right) p^2 - \frac{\delta}{2} \sqrt{p^2 - \left(\frac{\delta}{4}\right)^2} - \delta \sqrt{p^2 - \left(\frac{\delta}{2}\right)^2} + \frac{\delta^2}{2}
\]  

(5.1)
The method to plot the bucket $i(i > 2)$ regions is the same as bucket 2 region does except expending radii $p$ to radii $ip$. Recall that, the algorithm starting condition is $p \geq \text{diagonal}$, for the convenience, the we just set $p = \text{diagonal}$, i.e. $p = \frac{\sqrt{5}\delta}{2}$. As we will see later, $p > \text{diagonal}$ will not affect our analysis. We therefore have the general formula $g(i)$ to measure the area of bucket i region.

\[
g(i) = \begin{cases} 
\left(\frac{5\pi}{4} + \frac{3\sqrt{5}+1}{2}\right)\delta^2 & \text{if } i = 1 \\
\frac{5\pi i^2}{4} + \frac{3\sqrt{5}}{2}i \\
-\left[\left(\frac{5\pi}{4} - \frac{5}{2}\text{arcsin}\frac{\sqrt{5}}{10(i-1)} - \frac{5}{2}\text{arcsin}\frac{\sqrt{5}}{5(i-1)}\right)(i-1)^2 \\
-\frac{1}{2}\sqrt{\frac{5}{4}(i-1)^2 - \frac{1}{26}} - \sqrt{\frac{5}{4}(i-1)^2 - \frac{1}{4}}\right]\delta^2 & \text{if } i > 2
\end{cases}
\]  

(5.2)

5.2.2 Coverable Regions

Similar to bucket region, the coverable region must be calculated by using the outer boundaries to subtract the inner boundaries. The rest of this section will discuss how to calculate the coverable regions.

5.2.2.1 The First Bucket

First of all, let's discuss the situation of bucket 1. In Figure 5.3, 5.4, 5.5, 5.6, and 5.7, we illustrate the coverable regions of five different density maps $m = 1$, $m = 2$, $m = 3$, $m = 4$, and $m = 5$, respectively. The thin solid line with zigzagged pattern indicates the coverable region of cell $A$, denotes as $A'$. Its region contains all the points can be resolved into coverable region of cell $A$, i.e. the minimal and maximal distances from any point in cell $A$ (subcell of $A$) to the points in coverable region $A'$ fall into the range $[(i-1)p, ip]$. In order to calculate the area of $A'$, we adopt a approximated boundaries, as shown as dashed line. As the $m$ increases, the boundaries of $A'$ (zigzagged pattern) are approaching to the approximated boundaries.

\[
S_{\text{sector}} = \frac{\arccos\frac{\delta}{4p}}{2\pi} \cdot \pi p^2 = \arccos\frac{\delta}{4p} \cdot \frac{p^2}{2}
\]

\[
S_{\Delta} = \sqrt{p^2 - \left(\frac{\delta}{4}\right)^2} \cdot \frac{\delta}{4} \cdot \frac{1}{2} = \frac{\delta\sqrt{p^2 - \left(\frac{\delta}{4}\right)^2}}{8}
\]
For $m = 1$, we have

$$S_{A'} = 4(S_{sector} - S_{\triangle}) = 4 \left( \arccos \frac{\delta}{4p} \cdot \frac{p^2}{2} - \frac{\delta \sqrt{p^2 - \left(\frac{\delta}{4}\right)^2}}{8} \right)$$

(5.3)

Figure 5.3: The coverable region of the first bucket, $m = 1$.

For $m = 2$, we have

$$S_{A'} = 4 \cdot S_{sector} = \pi p^2$$

(5.4)

Figure 5.4: The coverable region of the first bucket, $m = 2$. 
For $m = 3$, we have

$$S_{A'} = 4S_{sector} + 2S_{rectangle} = \pi p^2 + \delta p$$  \hspace{1cm} (5.5)$$

Figure 5.5: The coverable region of the first bucket, $m = 3$.

For $m = 4$, we have

$$S_{A'} = 4S_{sector} + 2S_{rectangle_1} + 2S_{rectangle_2} + S_{rectangle} = \pi p^2 + \delta p + \frac{\delta}{2} \cdot p + \frac{\delta^2}{8}$$  \hspace{1cm} (5.6)$$

Figure 5.6: The coverable region of the first bucket, $m = 4$. 
For \( m = 5 \), we have

\[
S_{A'} = 4S_{\text{sector}} + 2S_{\text{rectangle}_1} + 2S_{\text{rectangle}_2} + S_{\text{rectangle}}
\]

\[
= \pi p^2 + \frac{3\delta}{2} \cdot p + \frac{\delta}{2} \cdot p + \frac{3\delta^2}{16}
\]

(5.7)

Figure 5.7: The coverable region of the first bucket, \( m = 5 \).

By combining the above 5 cases, when \( i = 1 \), we have the general formula for \( S_{A'} \) to calculate the area if region \( A' \).

\[
S_{A'} = \begin{cases} 
4p^2 \arccos \frac{\delta}{4p} - \frac{\delta \sqrt{p^2 - \left( \frac{\delta}{4} \right)^2}}{2} & m = 1 \\
\pi p^2 + 2p\left(1 - \frac{2}{2^{\frac{m-1}{2}}}\right)\delta + 2p\left(1 - \frac{2}{2^{\frac{m}{2}}}\right)\frac{\delta}{2} & m \geq 2, \text{ and } m \text{ is even} \\
\pi p^2 + 2p\left(1 - \frac{1}{2^{\frac{m-1}{2}}}\right)\delta + 2p\left(1 - \frac{2}{2^{\frac{m}{2}}}\right)\frac{\delta}{2} & m \geq 2, \text{ and } m \text{ is odd} \\
+(1 - \frac{2}{2^{\frac{m-1}{2}}})\delta \cdot (1 - \frac{2}{2^{\frac{m}{2}}})\frac{\delta}{2} & m \geq 2, \text{ and } m \text{ is even}
\end{cases}
\]
5.2.2.2 The Second Bucket and Beyond

The buckets beyond the first one are similar to how we calculate bucket region \( i \) \((i > 2)\). First of all, we have to have the outer boundaries of region \( A_i' \). This region could be simply expending the radii of arcs \( p \) to \( i p \), with \( p = \frac{\sqrt{5} \delta}{2} \), yields,

\[
S_{\text{out}(i)} = \begin{cases} 
\left[ \frac{5}{4} \pi i^2 + \frac{3\sqrt{5}}{2} \left( 1 - \frac{2}{2^{m-1}} \right) + \frac{1}{2} \left( 1 - \frac{2}{2^{m-1-1}} \right) \right] \delta^2 & m \geq 2, \text{ and } m \text{ is even} \\
\left[ \frac{5}{4} \pi i^2 + \sqrt{5} \left( 1 - \frac{1}{2^{m-1}} \right) + \frac{\sqrt{5}}{2} \left( 1 - \frac{2}{2^{m-1-1}} \right) \right] \delta^2 & m \geq 2, \text{ and } m \text{ is odd}
\end{cases}
\]

Figure 5.8: Inner boundaries of the coverable region with \( m = 1 \).

Then, let’s study the inner boundaries of the coverable region. Figure 5.8 shows an example with \( m = 1 \) against the second and third bucket. Clearly, any cell that crossed by a segment of the theoretical inner boundary, as shown in Figure 5.8 thick solid line, will not be able to resolve into bucket \( i \), because they are only resolvable to bucket \((i - 1)\). In addition, there are more cells that are not resolvable to either bucket i
or \((i-1)\). In Figure 5.8, \(m = 1\), dashed line separate the regions which are resolvable and which are not. This boundaries were being plotted as follow: for each quadrant of cell A, we draw a arc (dashed line) with radius \((i-1)p\) and centered at the corner of the subcell of A. Consequently, any cell that crossed by this arc cannot resolve into bucket \(i\), because they are to close to \(A\). Such boundary also approximates the real inner boundaries (zigzagged pattern), and we have formula to calculate the area of this approximated boundaries.

\[
\pi (ip)^2 + \delta ip - \pi [(i-1)p]^2 - \delta(i-1)p
\]  

(5.8)

Figure 5.9(a), 5.9(b), 5.10(a), and 5.10(b), illustrate the cases when \(m = 2\), \(m = 3\), \(m = 4\), and \(m = 5\), respectively. For the cases of \(m \geq 2\), we can use the same method as case of \(m = 1\) to generate the real inner boundaries and approximated inner boundaries. Notice that, as \(m\) increases, the point \(C\) is approaching to point \(O\). Thus, with a same radius \((i-1)p\) the approximated inner boundaries are approaching to the theoretical inner boundaries. The following subsection will elaborate how to calculate the area of the approximated inner boundaries.

**Figure 5.9: Inner boundaries of the coverable region with \(m = 2\) and \(m = 3\).**
First of all, we need to calculate angle $\beta$ that encloses the shade area, as illustrated in Figure 5.11.

$$\beta = \angle DCB = \frac{\pi}{2} - \angle JCD - \angle KCB = \begin{cases} 
\beta_{\text{even}} = \frac{\pi}{2} - \arcsin \frac{\theta_m \cdot \delta}{p} - \arcsin \frac{\theta_m \cdot \delta}{p} & m \text{ is even} \\
\beta_{\text{odd}} = \frac{\pi}{2} - \arcsin \frac{\theta_m \cdot \delta}{p} - \arcsin \frac{\theta_m + 2 \cdot \delta}{p} & m \text{ is odd} 
\end{cases}$$  \hspace{1cm} (5.9)

where $\theta$ is a function of $m$ for the convenience in further discussions. Theoretically, when $m$ is even, line segment $DJ \neq BK$, in other words, the subcell is a rectangle with two different sides. When $m$ is odd, line segment $DJ = BK$, correspondingly, the subcell is a square. We therefore define.

$$\theta_m = \frac{1}{2} - \frac{1}{2 \cdot \pi}$$

By using the $\beta$, we can easily to calculate the area of the Sector $\widehat{BDC}$,

$$S_{\widehat{BDC}} = \frac{\beta}{2\pi} \cdot \pi p^2 = \frac{\beta p^2}{2} \hspace{1cm} (5.10)$$
Figure 5.11: An illustration on how to calculate the area of region bounded by four arcs.

Then, we can calculate the area of the polygon $BFDC$, by using the formula,

\[ S_{\text{polygon}} = S_{\Delta BHC} + S_{\Delta DIC} - S_{IFHC} \]  

(5.11)

where $S_{\Delta BHC}$, $S_{\Delta DIC}$, and $S_{IFHC}$ are defined as following,

\[
S_{\Delta BHC} = \begin{cases} 
\sqrt{p^2 - (\theta_m \delta)^2} \cdot \theta_m \delta \cdot \frac{1}{2} & m \text{ is even} \\
\sqrt{p^2 - (\theta_{m+1} \delta)^2} \cdot \theta_{m+1} \delta \cdot \frac{1}{2} & m \text{ is odd}
\end{cases}
\]

\[
S_{\Delta DIC} = \sqrt{p^2 - (\theta_m \delta)^2} \cdot \theta_m \frac{\delta}{2} \cdot \frac{1}{2}
\]

\[
S_{IFHC} = \begin{cases} 
\theta_m^2 \cdot \frac{\delta^2}{2} & m \text{ is even} \\
\theta_{m-1} \cdot \theta_{m+1} \cdot \frac{\delta^2}{2} & m \text{ is odd}
\end{cases}
\]

Last, the area of the square $LEFG$ is fixed,

\[ S_{\text{RectangleLEFG}} = \frac{\delta^2}{8} \]  

(5.12)

Thus, by plugging above four equations, we obtain the area of region bounded by four arcs.

\[ S_{\text{shade}} = S_{\text{sector}} - S_{\Delta DIC} - S_{\Delta BHC} + S_{\text{RectangleIFHC}} - S_{\text{RectangleLEFG}} \]  

(5.13)
For the $i$-th bucket, we can get the general equation to calculate the area of $S_{\text{shade}(i)}$.

\[
S_{\text{shade}(i)} = \begin{cases}
\frac{5\beta_{\text{even}}}{8} - \frac{\theta_m}{4} & \sqrt{\frac{5}{4} - \frac{\theta_m^2}{4} - \frac{\theta_{m-1}^2}{4} - \frac{\theta_{m+1}^2}{4}} \delta^2 \\
\frac{5\beta_{\text{even}}}{8} - m \text{ is even} \\
\frac{5\beta_{\text{even}}}{8} - \frac{\theta_{m-1}}{4} & \sqrt{\frac{5}{4} - \frac{\theta_{m-1}^2}{4} - \frac{\theta_{m+1}^2}{4} - \frac{\theta_{m-1}^2}{4}} \delta^2 \\
\frac{5\beta_{\text{odd}}}{8} - m \text{ is odd}
\end{cases}
\]

We denote the area of the coverable region $A'$ for bucket $i$ under different $m$ values $f(i, m)$. i.e.,

\[
f(i, m) = S'_A = S_{\text{out}(i)} - 4 \cdot S_{\text{shade}(i-1)} - S_A
\]

(5.14)

We use the non-covering factor $\alpha(m)$ (Equation 5.16) to analyze the percentage of unresolvable pairs of cell at each level, i.e. the ratio of $\alpha(m+1)$ to $\alpha(m)$. By simulating the unresolvable ratio with specific
parameter $m$ and $i$, we got a set of very interesting statistics, as shown in Table 5.1.

$$\alpha(m) = 1 - c(m) = \frac{\sum_{i=1}^{l}[g(i) - f(i,m)]}{\sum_{i=1}^{l} g(i)} \quad (5.16)$$

<table>
<thead>
<tr>
<th>$\frac{\alpha(m+1)}{\alpha(m)}$</th>
<th>i=2</th>
<th>i=4</th>
<th>i=8</th>
<th>i=16</th>
<th>i=32</th>
<th>i=64</th>
<th>i=128</th>
<th>i=256</th>
<th>i=512</th>
<th>i=1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>m=1</td>
<td>0.74197</td>
<td>0.64118</td>
<td>0.61973</td>
<td>0.61336</td>
<td>0.61305</td>
<td>0.61297</td>
<td>0.61295</td>
<td>0.61295</td>
<td>0.61295</td>
<td>0.61295</td>
</tr>
<tr>
<td>m=2</td>
<td>0.67732</td>
<td>0.6691</td>
<td>0.6721</td>
<td>0.66679</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
</tr>
<tr>
<td>m=3</td>
<td>0.74807</td>
<td>0.74909</td>
<td>0.74968</td>
<td>0.74997</td>
<td>0.74999</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=4</td>
<td>0.67521</td>
<td>0.6688</td>
<td>0.6719</td>
<td>0.6667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
</tr>
<tr>
<td>m=5</td>
<td>0.74448</td>
<td>0.74809</td>
<td>0.74941</td>
<td>0.74995</td>
<td>0.74999</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=6</td>
<td>0.67473</td>
<td>0.66891</td>
<td>0.6726</td>
<td>0.66682</td>
<td>0.66671</td>
<td>0.6667</td>
<td>0.6667</td>
<td>0.6667</td>
<td>0.6667</td>
<td>0.6667</td>
</tr>
<tr>
<td>m=7</td>
<td>0.74276</td>
<td>0.74762</td>
<td>0.74929</td>
<td>0.74994</td>
<td>0.74998</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=8</td>
<td>0.67464</td>
<td>0.66903</td>
<td>0.6732</td>
<td>0.66685</td>
<td>0.66672</td>
<td>0.66668</td>
<td>0.6667</td>
<td>0.6667</td>
<td>0.6667</td>
<td>0.6667</td>
</tr>
<tr>
<td>m=9</td>
<td>0.74193</td>
<td>0.74739</td>
<td>0.74923</td>
<td>0.74978</td>
<td>0.74994</td>
<td>0.74998</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=10</td>
<td>0.67464</td>
<td>0.6691</td>
<td>0.6736</td>
<td>0.66686</td>
<td>0.66672</td>
<td>0.66668</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
</tr>
<tr>
<td>m=11</td>
<td>0.74151</td>
<td>0.74728</td>
<td>0.7492</td>
<td>0.74977</td>
<td>0.74994</td>
<td>0.74998</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=12</td>
<td>0.67465</td>
<td>0.66915</td>
<td>0.6738</td>
<td>0.66687</td>
<td>0.66672</td>
<td>0.66668</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
</tr>
<tr>
<td>m=13</td>
<td>0.74131</td>
<td>0.74723</td>
<td>0.74919</td>
<td>0.74977</td>
<td>0.74994</td>
<td>0.74998</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=14</td>
<td>0.67466</td>
<td>0.66917</td>
<td>0.6739</td>
<td>0.66687</td>
<td>0.66672</td>
<td>0.66668</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
</tr>
<tr>
<td>m=15</td>
<td>0.74121</td>
<td>0.7472</td>
<td>0.74918</td>
<td>0.74977</td>
<td>0.74994</td>
<td>0.74998</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=16</td>
<td>0.67466</td>
<td>0.66918</td>
<td>0.674</td>
<td>0.66687</td>
<td>0.66672</td>
<td>0.66668</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
</tr>
<tr>
<td>m=17</td>
<td>0.74116</td>
<td>0.74718</td>
<td>0.74918</td>
<td>0.74977</td>
<td>0.74994</td>
<td>0.74998</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=18</td>
<td>0.67467</td>
<td>0.66919</td>
<td>0.674</td>
<td>0.66687</td>
<td>0.66672</td>
<td>0.66668</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
</tr>
<tr>
<td>m=19</td>
<td>0.74113</td>
<td>0.74718</td>
<td>0.74918</td>
<td>0.74977</td>
<td>0.74994</td>
<td>0.74998</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=20</td>
<td>0.67467</td>
<td>0.66919</td>
<td>0.674</td>
<td>0.66687</td>
<td>0.66672</td>
<td>0.66668</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
</tr>
</tbody>
</table>

Now, let’s study the ratio $\frac{\alpha(m+1)}{\alpha(m)}$ more rigorously.

$$\frac{\alpha(m+1)}{\alpha(m)} = \frac{\sum_{i=1}^{l}[g(i) - f(i,m+1)]}{\sum_{i=1}^{l} g(i) - \sum_{i=1}^{l} f(i,m)} \quad (5.17)$$

From the experimental results, as shown in Table 5.1, we observe that when $m$ is an even, the ratio of $\alpha(m+1)$ to $\alpha(m)$ is converged to exactly $\frac{2}{3}$ (we merely rounded the results to four digits after the decimal point), and the ratio of $\alpha(m+2)$ to $\alpha(m+1)$ is converged to exactly $\frac{3}{4}$. First of all, let study the ratio
\( \alpha(m+1) \) to \( \alpha(m) \), when \( m \) is an even, we assume \( \frac{\alpha(m+1)}{\alpha(m)} = \frac{2}{3} \), obtaining,

\[
\frac{\alpha(m+1)}{\alpha(m)} = \frac{\sum_{i=1}^{l} g(i) - \sum_{i=1}^{l} f(i, m+1)}{\sum_{i=1}^{l} g(i) - \sum_{i=1}^{l} f(i, m)} = \frac{2}{3}
\]

\[
\Rightarrow 3 - \frac{3 \sum_{i=1}^{l} f(i, m+1)}{\sum_{i=1}^{l} g(i)} = 2 - \frac{2 \sum_{i=1}^{l} f(i, m)}{\sum_{i=1}^{l} g(i)}
\]

\[
\Rightarrow 3 \sum_{i=1}^{l} f(i, m+1) - 2 \sum_{i=1}^{l} f(i, m) = \sum_{i=1}^{l} g(i)
\]

yields,

\[
\sum_{i=1}^{l} [3f(i, m+1) - 2f(i, m)] = \sum_{i=1}^{l} g(i) \quad (5.18)
\]

Thus, if we can prove the Equation 5.18 is hold, then our assumption \( \frac{\alpha(m+1)}{\alpha(m)} = \frac{2}{3} \) is proved. Let’s discuss the left part and right part of the equation, respectively.

\[
left = \delta^2 \sum_{i=2}^{l} \left\{ 3 \left[ \frac{5}{4} \pi i^2 + 3 \sqrt{5} \theta_m i - \frac{5 \beta_{\text{even}}}{2} (i - 1)^2 \right.ight.
\]

\[
+ \theta_m \sqrt{\frac{5}{4} (i - 1)^2 - \frac{\theta_m^2}{4}} + 2 \theta_m \sqrt{\frac{5}{4} (i - 1)^2 - \frac{\theta_m^2}{4}} \right]
\]

\[
- 2 \left[ \frac{5}{4} \pi i^2 + 2 \sqrt{5} \theta_{m+2} i + \sqrt{5} \theta_m i - \frac{5 \beta_{\text{odd}}}{2} (i - 1)^2 \right.
\]

\[
+ \theta_m \sqrt{\frac{5}{4} (i - 1)^2 - \frac{\theta_m^2}{4}} + 2 \theta_{m+2} \sqrt{\frac{5}{4} (i - 1)^2 - \frac{\theta_{m+2}^2}{4}} \right\} \quad (5.19)
\]

\[
right = \delta^2 \sum_{i=2}^{l} \left\{ \frac{5}{4} \pi i^2 + 3 \sqrt{5} i
\]

\[
- \left[ \left( \frac{5}{4} \pi - \frac{5}{2} \arcsin \frac{\sqrt{5}}{10(i - 1)} - \frac{5}{2} \arcsin \frac{\sqrt{5}}{5(i - 1)} \right)(i - 1)^2 \right.
\]

\[
- \frac{1}{2} \sqrt{\frac{5}{4} (i - 1)^2 - \frac{1}{16}} - \sqrt{\frac{5}{4} (i - 1)^2 - \frac{1}{4}} \right\} \quad (5.20)
\]
\[
\left. \begin{array}{l}
\delta^2 \sum_{i=2}^{l} \left\{ \left( -3\sqrt{5}\theta_m + 6\sqrt{5}\theta_{m+2} - \frac{3\sqrt{5}}{2} \right) \cdot i \\
\left( -\frac{3\sqrt{5}}{2}\theta_m + 3\sqrt{5}\theta_{m+2} - \frac{3\sqrt{5}}{4} \right) (i-1) \\
\left[ 5\beta_{\text{odd}} - \frac{15}{2}\beta_{\text{even}} + \frac{5}{4}\pi - \frac{5}{2} \left( \arcsin \frac{\sqrt{5}}{10(i-1)} + \arcsin \frac{\sqrt{5}}{5(i-1)} \right) \right] (i-1)^2 \right\}
\end{array} \right\} \tag{5.21}
\]

Since the \( m \) is level of the density map, when \( m \) getting larger, the approximated boundary will approach to the theoretical boundary. Therefore when \( m \) approaches to infinity, the \( \theta \) approaches to \( \frac{1}{2} \), thus, we can replace all the \( \theta \) by \( \frac{1}{2} \) into the above equation, and when \( l \to \infty \), obtains,

\[
\sum_{i=2}^{l} \left( -3\sqrt{5} \cdot \frac{1}{2} + 6\sqrt{5} \cdot \frac{1}{2} - \frac{3\sqrt{5}}{2} \right) i = 0
\]

\[
\sum_{i=2}^{l} \left( -\frac{3\sqrt{5}}{2} \cdot \frac{1}{2} + 3\sqrt{5} \cdot \frac{1}{2} - \frac{3\sqrt{5}}{4} \right) (i-1) = 0
\]

\[
\beta_{\text{even}} = \frac{\pi}{2} - \arcsin \frac{\theta_m \sqrt{5}}{5i} - \arcsin \frac{2\sqrt{5}\theta_{m+2}}{5i} \to \frac{\pi}{2}
\]

\[
\beta_{\text{odd}} = \frac{\pi}{2} - \arcsin \frac{\theta_m \sqrt{5}}{5i} - \arcsin \frac{2\sqrt{5}\theta_{m+2}}{5i} \to \frac{\pi}{2}
\]

\[
\frac{5}{2} \left( \arcsin \frac{\sqrt{5}}{10(i-1)} + \arcsin \frac{\sqrt{5}}{5(i-1)} \right) \to 0
\]

\[
\sum_{i=2}^{l} \left[ 5\beta_{\text{odd}} - \frac{15}{2}\beta_{\text{even}} + \frac{5}{4}\pi - \frac{5}{2} \left( \arcsin \frac{\sqrt{5}}{10(i-1)} + \arcsin \frac{\sqrt{5}}{5(i-1)} \right) \right] (i-1)^2 \to 0
\]

Therefore, \( \left. left = right \right. \), the Equation 5.18 is proved. We could conclude that when \( m \) is an even, \( \frac{\alpha(m+1)}{\alpha(m)} = \frac{2}{3} \) holds.

Then, let’s look at the ratio \( \alpha(m+2) \) to \( \alpha(m+1) \). Accordingly, \( m \)-th level and \( (m+2) \)-th level are two consecutive levels on the Quad-tree. Regarding to the Theorem 1, we have

\[
\frac{\alpha(m+2)}{\alpha(m)} = \frac{1}{2}
\]

35
such that
\[
\frac{\alpha(m + 1)}{\alpha(m)} \cdot \frac{\alpha(m + 2)}{\alpha(m + 1)} = \frac{1}{2}
\] (5.22)

Since we already have \(\frac{\alpha(m+1)}{\alpha(m)} = \frac{2}{3}\), we can obtain,
\[
\frac{\alpha(m + 2)}{\alpha(m + 1)} = \frac{3}{4}
\] (5.23)

From the above mathematical proof, we can get a new theorem, which describes as following:

**Theorem 2** Let \(DM_i\) be the first density map where the DM-SDH algorithm starts running on a k-d tree, and \(\alpha(m)\) be the percentage of pairs of cells that are not resolvable on the density map that lies \(m\) levels below \(DM_i\).

if \((i + m)\) is an even, we have
\[
\lim_{p \to 0} \frac{\alpha(m + 1)}{\alpha(m)} = \frac{3}{4}
\]
\[
\lim_{p \to 0} \frac{\alpha(m + 2)}{\alpha(m + 1)} = \frac{2}{3}
\] (5.24)

if \((i + m)\) is an odd, we have
\[
\lim_{p \to 0} \frac{\alpha(m + 1)}{\alpha(m)} = \frac{2}{3}
\]
\[
\lim_{p \to 0} \frac{\alpha(m + 2)}{\alpha(m + 1)} = \frac{3}{4}
\] (5.25)

5.3 Analysis of DM-SDH Starting at Level of Square Cell

Given any cell \(A\) on density map \(DM_0\), as illustrated in Figure 5.12, a cell \(A\) is a square which draws with four points \(O_1, O_2, O_3,\) and \(O_4\), bucket region \(A_1\) therefore bounded by 4 arcs and 4 line segments connected by points \(C_1\) through \(C_8\). The bucket region \(A_2\) is similar to bucket region \(A_1\), the radii of the four arcs are scaled to \(2p\), and not include the inner region. So on and so forth, we have general formula \(g(i)\) to calculate the bucket region \(A_i\).
Figure 5.12: Theoretical boundaries of bucket 1 and bucket 2 regions of cell A, with the bucket width $p$ being exactly $\sqrt{2} \delta$.

$$g(i) = \begin{cases} (2\pi + 4\sqrt{2} + 1)\delta^2 & i = 1 \\ 2\pi i^2 + 4\sqrt{2}i \\ -(i - 1)^2 \cdot \left(8 \arctan \sqrt{8(i - 1)^2 - 1} - 2\pi \right) \\ -\sqrt{8(i - 1)^2 - 1} \delta^2 \\ \end{cases}$$

(5.26)

5.3.1 Coverable Regions

5.3.1.1 The First Bucket

Figure 5.13: The coverable region of the first bucket, $m = 1$. 
\[ S_{\text{sector}} = \frac{\arccos \frac{\delta}{2p} \cdot \pi p^2}{2\pi} = \arccos \frac{\delta}{2p} \cdot \frac{p^2}{2} \]

\[ S_{\Delta} = \sqrt{p^2 - \left(\frac{\delta}{2}\right)^2} \cdot \frac{\delta}{2} \cdot \frac{1}{2} = \frac{\delta \sqrt{p^2 - \left(\frac{\delta}{2}\right)^2}}{4} \]

\[ S_{A'} = 4(S_{\text{sector}} - S_{\Delta}) = 4 \left( \arccos \frac{\delta}{2p} \cdot \frac{p^2}{2} - \frac{\delta \sqrt{p^2 - \left(\frac{\delta}{2}\right)^2}}{4} \right) \] (5.27)

For \( m = 2 \), we have

\[ S_{A'} = 4 \cdot S_{\text{sector}} = \pi p^2 \] (5.28)

For \( m = 3 \), we have

\[ S_{A'} = 4 \cdot S_{\text{sector}} + 2 \cdot S_{\text{rectangle}} = \pi p^2 + \delta p \] (5.29)
Figure 5.15: The coverable region of the first bucket, \( m = 4 \) and \( m = 5 \).

For \( m = 4 \), we have

\[
S_{A'} = 4 \cdot S_{\text{sector}} + 4 \cdot S_{\text{rectangle}} + S_{\text{square}} = \pi p^2 + 2\delta p + \frac{\delta^2}{4} \tag{5.30}
\]

For \( m = 5 \), we have

\[
S_{A'} = 4 \cdot S_{\text{sector}} + 2 \cdot S_{\text{rectangle}_1} + 2 \cdot S_{\text{rectangle}_2} + S_{\text{square}} = \pi p^2 + p\delta + 2p(\delta - \frac{\delta}{4}) + \frac{\delta}{2}(\delta - \frac{\delta}{4}) \tag{5.31}
\]

By combining above 5 cases, we have \( S_{A'} \) for bucket 1.

\[
S_{A'} = \begin{cases} 
\left( 4 \cdot \arccos \frac{\sqrt{3}}{4} - \frac{\sqrt{3}}{2} \right) \delta^2 & m = 1 \\
2\pi + 4\sqrt{2} \left( 1 - \frac{2}{2^m} \right) + \left( 1 - \frac{2}{2^m} \right)^2 \delta^2 & m \geq 2, m \text{ is even} \\
2\pi + 2\sqrt{2} \left( 1 - \frac{2}{2^{m-1}} \right) + 2\sqrt{2} \left( 1 - \frac{2}{2^{m+1}} \right) + \left( 1 - \frac{2}{2^{m-1}} \right) \left( 1 - \frac{2}{2^{m+1}} \right) \delta^2 & m > 2, m \text{ is odd}
\end{cases}
\tag{5.32}
\]
5.3.1.2 The Second Bucket and Beyond

As we mentioned earlier, the cases of buckets beyond the first one are similar to how we calculate bucket region when \( i > 2 \). First of all, we have to have the outer boundaries of region \( A' \). This region could be simply extended to the radius of arcs \( \sqrt{2}\delta \) to \( i \cdot \sqrt{2}\delta \), yields,

\[
S_{out(i)} = \begin{cases} 
2\pi i^2 + 4\sqrt{2}i \left( 1 - \frac{2}{2m \pi} \right) + \left( 1 - \frac{2}{2m \pi} \right)^2 \delta^2 & m \geq 2, m \text{ is even} \\
2\pi i^2 + 2\sqrt{2}i \left( 1 - \frac{2}{2m+1} \right) + 2\sqrt{2i} \left( 1 - \frac{2}{2m+1} \right) + \left( 1 - \frac{2}{2m+1} \right) \left( 1 - \frac{2}{2m+1} \right) \delta^2 & m > 2, m \text{ is odd}
\end{cases}
\]  

(5.33)

![Diagram](image.png)

Figure 5.16: Inner boundaries of the coverable region, \( m = 1 \).

Similarly, let’s study the inner boundaries of the coverable region. Figure 5.16 shows an example with \( m = 1 \) regarding to the second and the third bucket. Clearly, any cell that crossed by a segment of the theoretical inner boundary, as shown in Figure 5.16 thick solid line, will not be able to resolve into bucket \( i \), because they are only resolvable to bucket \((i - 1)\). In addition, there are more cells that are not resolvable to either bucket \( i \) or \((i - 1)\). In Figure 5.16, the dashed line separates the regions which are resolvable and with which are not at \( m = 1 \). This boundaries were being plotted as following: for each quadrant of cell \( A \), we draw a arc (dashed line) with radius \((i - 1)p\) and centered at the corner of the subcell of \( A \). Consequently,
any cell that crossed by this arc cannot resolve into bucket $i$, because they are too close to $A$. Such boundary also approximates the real inner boundaries (zigzagged pattern), and we have a formula to calculate the area of this approximated boundary.

$$
\pi (ip)^2 + 2(ip)\delta - \pi [(i-1)p]^2 - 2(i-1)p\delta \quad (i \geq 2)
$$

(5.34)

Figure 5.17: Inner boundaries of the coverable region, $m = 2$ and $m = 3$.

Figure 5.18: Inner boundaries of the coverable region, $m = 4$ and $m = 5$.  

41
Figure 5.17(a), 5.17(b), 5.18(a), and 5.18(b) illustrate the cases when \( m = 2, m = 3, m = 4, \) and \( m = 5 \), respectively. For the case of \( m \geq 2 \), we can use the same method for case of \( m = 1 \) to generate the real inner boundaries and approximated inner boundaries. As \( m \) increases, the point \( C \) is approaching to point \( O \), therefore, with a same radius \((i-1)p\) the approximated inner boundaries are approaching to the theoretical inner boundaries. The following subsection will introduce how to calculate the area of the approximated inner boundaries.

\[ \beta = \angle DCB = \begin{cases} 
\frac{\pi}{2} - \arcsin\left(\frac{\theta_m \delta}{p}\right) - \arcsin\left(\frac{\theta_{m+2} \delta}{p}\right) & \text{if } m \text{ is odd} \\
\frac{\pi}{2} - 2 \arcsin\left(\frac{\theta_m \delta}{p}\right) & \text{if } m \text{ is even}
\end{cases} \]

where \( \theta \) is a function of \( m \) for the convenience in further discussions. Theoretically, when \( m \) is odd, line segment \( DJ \) will not equals to line segment \( BK \), in other words, the subcell is a rectangle with two different sides. Thus, we define \( \theta \) as following,

\[ \theta_m = \frac{1}{2} - \frac{1}{2^{m+1}} \]

when \( m \) is even, line segment \( DJ = BK \), correspondingly, the subcell is a square. Physically, it represents at the interim level, the side length keeps the same on the \( x \)-dimension, but on \( y \)-dimension, the side length will be cut a half of the side length of the previous level.
By using the $\beta$, we can easily calculate the area of the Sector $\widehat{BDC}$,

$$S_{\widehat{BDC}} = \frac{\beta}{2\pi} \cdot \pi p^2$$

$$= \frac{\beta p^2}{2}$$

(5.35)

then, we can calculate the area of the polygon $BFDC$, by using the formula,

$$S_{\text{polygon}} = S_{\triangle BHC} + S_{\triangle DIC} - S_{IFHC}$$

(5.36)

where $S_{\triangle BHC}$, $S_{\triangle DIC}$, and $S_{IFHC}$ are defined as following.

$$S_{\triangle BHC} = \sqrt{p^2 - (\theta_m \delta)^2} \cdot \theta_m \delta \cdot \frac{1}{2}$$

$$S_{\triangle DIC} = \sqrt{p^2 - (\theta_{m+2} \delta)^2} \cdot \theta_{m+2} \delta \cdot \frac{1}{2}$$

$$S_{IFHC} = \begin{cases} 
\theta_m \cdot \theta_{m+2} \cdot \delta^2 & \text{if } m \text{ is odd} \\
(\theta_m)^2 \cdot \delta^2 & \text{if } m \text{ is even}
\end{cases}$$

Last, the area of the square $LEFG$ is firmed,

$$S_{\text{square}} = \frac{\delta^2}{4}$$

(5.37)

plug in all above four equations, we can get the area of the region bounded by four arcs.

$$S_{\text{shade}} = \begin{cases} 
S_{\text{sector}} - (S_{\triangle BHC} + S_{\triangle DIC} - S_{IFHC}) - S_{\text{square}} & \text{if } m \text{ is odd} \\
S_{\text{sector}} - (2S_{\triangle BHC} - S_{IFHC}) - S_{\text{square}} & \text{if } m \text{ is even}
\end{cases}$$

For the $i$-th bucket, we can get the general equation to calculate the area $S_{\text{shade}(i)}$. 

43
$$S_{\text{shade}(i)} = \begin{cases} 
\frac{\beta_{\text{odd}}(ip)^2}{2} - \left[ \sqrt{p^2 - (\theta_m\delta)^2} \cdot \theta_m\delta \cdot \frac{1}{2} 
+ \sqrt{p^2 - (\theta_{m+2}\delta)^2} \cdot \theta_{m+2}\delta \cdot \frac{1}{2} - \theta_m \cdot \theta_{m+2} \cdot \delta^2 \right] \cdot \frac{\delta^2}{4} & \text{if } m \text{ is odd} \\
\frac{\beta_{\text{even}}(ip)^2}{2} - \left[ 2\sqrt{p^2 - (\theta_m\delta)^2} \cdot \theta_m\delta \cdot \frac{1}{2} 
- (\theta_m)^2 \cdot \delta^2 \right] \cdot \frac{\delta^2}{4} & \text{if } m \text{ is even} 
\end{cases}$$

$$S_A' = S_{\text{out}(i)} - 4 \cdot S_{\text{shade}(i-1)} - S_A$$ (5.38)

With the Equation 5.38, we obtain the general formula \( f(i, n) \) to calculate the area of coverable region for bucket \( i \).

$$f(i, m) = \begin{cases} 
(4 \cdot \arccos \sqrt{\frac{7}{8} - \frac{\sqrt{3}}{8}}) \delta^2 & \text{if } i = 1, m = 1 \\
(2\pi + 8\sqrt{2}\theta_m + 4\theta_m^2) \delta^2 & \text{if } i = 1, m \geq 2, \text{ and } m \text{ is even} \\
\left[ 2\pi + 4\sqrt{2}(\theta_m + \theta_{m+2}) + 4\theta_m\theta_{m+2} \right] \delta^2 & \text{if } i = 1, m > 2, \text{ and } m \text{ is odd} \\
2\pi i^2 + 2\sqrt{2}i - 2\pi(i - 1) - 2\sqrt{2}(i - 1) \delta^2 & \text{if } i > 1, m = 1 \\
\end{cases}$$

$$\begin{cases} 
\{ 2\pi i^2 + 8\sqrt{2}i\theta_m + 4\theta_m^2 
-4 \left[ \beta_{\text{even}}(i - 1)^2 - \left( \theta_m \sqrt{2(i - 1)^2 - \theta_m^2} - \theta_m^2 \right) \right] \delta^2 & \text{if } i > 1, m \geq 2, \text{ and } m \text{ is even} \\
\{ 2\pi i^2 + 4\sqrt{2}(\theta_m + \theta_{m+2})i + 4\theta_m\theta_{m+2} 
-4 \left[ \beta_{\text{odd}}(i - 1)^2 - \left( \theta_m \sqrt{2(i - 1)^2 - \theta_m^2} \right) \right. \
\left. + \frac{\theta_{m+2} \sqrt{2(i - 1)^2 - \theta_{m+2}^2} - \theta_m \cdot \theta_{m+2}}{2} \right] \delta^2 & \text{if } i > 1, m > 2, \text{ and } m \text{ is odd} 
\end{cases}$$
We use the non-covering factor $\alpha(m)$ (Equation 5.39) to analyze the percentage of unresolvable pairs of cell at each level, i.e. the ratio of $\alpha(m+1)$ to $\alpha(m)$.

$$\alpha(m) = 1 - c(m) = \frac{\sum_{i=1}^{l}[g(i) - f(i, m)]}{\sum_{i=1}^{l} g(i)} \quad (5.39)$$

### Table 5.2: Values of $\frac{\alpha(m+1)}{\alpha(m)}$ when DM-SDH algorithm starts at square cell.

<table>
<thead>
<tr>
<th>$\frac{\alpha(m+1)}{\alpha(m)}$</th>
<th>$i=2$</th>
<th>$i=4$</th>
<th>$i=8$</th>
<th>$i=16$</th>
<th>$i=32$</th>
<th>$i=64$</th>
<th>$i=128$</th>
<th>$i=256$</th>
</tr>
</thead>
<tbody>
<tr>
<td>m=1</td>
<td>0.8068</td>
<td>0.8898</td>
<td>0.9413</td>
<td>0.9697</td>
<td>0.9846</td>
<td>0.9922</td>
<td>0.9961</td>
<td>0.9980</td>
</tr>
<tr>
<td>m=2</td>
<td>0.7596</td>
<td>0.7522</td>
<td>0.7505</td>
<td>0.7502</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
</tr>
<tr>
<td>m=3</td>
<td>0.6696</td>
<td>0.6670</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
</tr>
<tr>
<td>m=4</td>
<td>0.7545</td>
<td>0.7510</td>
<td>0.7502</td>
<td>0.7501</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
</tr>
<tr>
<td>m=5</td>
<td>0.6677</td>
<td>0.6667</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
</tr>
<tr>
<td>m=6</td>
<td>0.7521</td>
<td>0.7504</td>
<td>0.7502</td>
<td>0.7501</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
</tr>
<tr>
<td>m=7</td>
<td>0.6670</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
</tr>
<tr>
<td>m=8</td>
<td>0.7510</td>
<td>0.7502</td>
<td>0.7501</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
</tr>
<tr>
<td>m=9</td>
<td>0.6668</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
</tr>
<tr>
<td>m=10</td>
<td>0.7505</td>
<td>0.7501</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.7500</td>
</tr>
<tr>
<td>m=11</td>
<td>0.6668</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
</tr>
<tr>
<td>m=12</td>
<td>0.7502</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=13</td>
<td>0.6667</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
</tr>
<tr>
<td>m=14</td>
<td>0.7501</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>m=15</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
<td>0.6666</td>
</tr>
<tr>
<td>m=16</td>
<td>0.75006</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Now, let’s study the ratio $\frac{\alpha(m+1)}{\alpha(m)}$ more rigorously.

$$\frac{\alpha(m+1)}{\alpha(m)} = \frac{\sum_{i=1}^{l}[g(i) - f(i, m+1)]}{\sum_{i=1}^{l} g(i)} = \frac{\sum_{i=1}^{l} g(i) - \sum_{i=1}^{l} f(i, m+1)}{\sum_{i=1}^{l} g(i) - \sum_{i=1}^{l} f(i, m)} \quad (5.40)$$

First of all, let’s study the ratio $\alpha(m+1)$ to $\alpha(m)$, when $m$ is an even, we assume $\frac{\alpha(m+1)}{\alpha(m)} = \frac{3}{4}$, obtaining,

$$\frac{\alpha(m+1)}{\alpha(m)} = \frac{\sum_{i=1}^{l} g(i) - \sum_{i=1}^{l} f(i, m+1)}{\sum_{i=1}^{l} g(i)} = \frac{3}{4}$$

$$4 - \frac{4\sum_{i=1}^{l} f(i, m+1)}{\sum_{i=1}^{l} g(i)} = 3 - \frac{3\sum_{i=1}^{l} f(i, m)}{\sum_{i=1}^{l} g(i)}$$

$$4 \sum_{i=1}^{l} f(i, m+1) - 3 \sum_{i=1}^{l} f(i, m) = \sum_{i=1}^{l} g(i)$$

45
yields,
\[ \sum_{i=1}^{l} [4f(i, m + 1) - 3f(i, m)] = \sum_{i=1}^{l} g(i) \]  \hspace{1cm} (5.41)

Thus, if we can prove the Equation 5.41 is hold, then our assumption \( \frac{a(m+1)}{a(m)} = \frac{3}{4} \) is proved. Let’s discuss the left part and right part of the equation, respectively.

\[
left = \sum_{i=2}^{l} \left\{ 4 \left[ 2\pi i^2 + 4\sqrt{2}(\theta_m + \theta_{m+2})i + 4\theta_m \theta_{m+2} \ight. \\
- 4 \left[ \beta_{odd}(i-1)^2 - \left( \frac{\theta_m \sqrt{2(i-1)^2 - \theta_m^2}}{2} \right. \\
+ \frac{\theta_{m+2} \sqrt{2(i-1)^2 - \theta_{m+2}^2}}{2} - \theta_m \cdot \theta_{m+2} \right] \right\} \\
- 3 \left\{ 2\pi i^2 + 8\sqrt{2}i\theta_m + 4\theta_m^2 \\
- 4 \left[ \beta_{even}(i-1)^2 - \left( \theta_m \sqrt{2(i-1)^2 - \theta_m^2} \right. \right. \right. \\
\left. \left. \left. - \theta_m^2 \right) \right]\right\} \right\} \\

right = \sum_{i=2}^{l} g(i) \\
= \sum_{i=2}^{l} \left[ 2\pi i^2 + 4\sqrt{2}i \\
- (i-1)^2 \cdot \left( 8 \arctan 2 \sqrt{2(i-1)^2 - \frac{1}{4}} - 2\pi \right) \\
- 2 \sqrt{2(i-1)^2 - \frac{1}{4}} \right] \hspace{1cm} (5.43)

\[
left - right = \sum_{i=2}^{l} \left\{ \left[ 16\sqrt{2}(\theta_m + \theta_{m+2}) - 24\sqrt{2}\theta_m - 4\sqrt{2} \right] \cdot i \\
+ \left[ 8\sqrt{2}(\theta_m + \theta_{m+2}) - 12\sqrt{2}\theta_m - 2\sqrt{2} \right] \cdot (i-1) \\
+ \left[ -16\beta_{odd} + 12\beta_{even} + 8 \arctan \sqrt{8(i-1)^2 - 2\pi} \right] \cdot (i-1)^2 \right\} \hspace{1cm} (5.44)
\]

Since the \( m \) is level of the density map, when \( m \) getting larger, the approximated boundary will approach to the theoretical boundary, as shown in Figure 5.16. Therefore, when the \( m \) approaches to \( \infty \), the \( \theta \)
approaches to $\frac{1}{2}$, thus, we can substitute the $\theta$ by $\frac{1}{2}$ into above equation, and when $l \to \infty$, yields,

\[
\sum_{i=2}^{l} \left[ 16\sqrt{2}(\frac{1}{2} + \frac{1}{2}) - 24\sqrt{2} \times \frac{1}{2} - 4\sqrt{2} \right] \cdot i = 0
\]

\[
\sum_{i=2}^{l} \left[ 8\sqrt{2}(\frac{1}{2} + \frac{1}{2}) - 12\sqrt{2} \times \frac{1}{2} - 2\sqrt{2} \right] \cdot (i - 1) = 0
\]

\[
\beta_{odd} = \frac{\pi}{2} - \arcsin \frac{\theta_m}{\sqrt{2i}} - \arcsin \frac{\theta_{m+2}}{\sqrt{2i}} + \frac{\pi}{2}
\]

\[
\beta_{even} = \frac{\pi}{2} - 2 \arcsin \frac{\theta_m}{\sqrt{2i}} + \frac{\pi}{2}
\]

\[
\arctan \sqrt{8(i-1)^2} \to \frac{\pi}{2}
\]

\[
\sum_{i=2}^{l} \left[ -16\beta_{odd} + 12\beta_{even} + 8 \arctan \sqrt{8(i-1)^2} - 2\pi \right] \cdot (i - 1)^2 \to 0
\]

\[
\text{left} = \text{right}
\]

the equation 6 is proved. We could conclude that when $m$ is an even, $\frac{\alpha(m+1)}{\alpha(m)} = \frac{3}{4}$ holds.

Then, let’s look at the ratio $\alpha(m + 2)$ to $\alpha(m + 1)$. Accordingly, $m$-th level and $(m + 2)$-th level are two consecutive levels on the Quad-tree. Regarding to the Theorem 1, we have

\[
\frac{\alpha(m + 2)}{\alpha(m)} = \frac{1}{2}
\]

such that

\[
\frac{\alpha(m + 1)}{\alpha(m)} \cdot \frac{\alpha(m + 2)}{\alpha(m + 1)} = \frac{1}{2}
\]

(5.45)

Since we already have $\frac{\alpha(m+1)}{\alpha(m)} = \frac{3}{4}$, by substituting Equation 5.45, we can obtain,

\[
\frac{\alpha(m + 2)}{\alpha(m + 1)} = \frac{2}{3}
\]

(5.46)
CHAPTER 6

PERFORMANCE OF DM-SDH

In computer science community, the analysis of algorithm is the determination of the amount of resources (time and storage) that are required to return the results. This criterion is an important method to evaluate the performance of algorithm. In order to analyze the performance of DM-SDH algorithm, we are going to study the computational complexity, which includes time complexity and space complexity, of the DM-SDH algorithm running on the Quad-tree (Oct-tree in 3D system) and $k$-d tree.

6.1 Time Complexity

In this section we are going discuss the theoretical time measurement of the DM-SDH algorithm running on two tree structures. As we discussed in the earlier chapter, the Quad-tree (Oct-tree in 3D system) only has one case, but $k$-d tree has 4 cases (9 cases in 3D system). We therefore compare the case of Quad-tree (Oct-tree) with each four cases (nine cases in 3D system) of $k$-d tree.

6.1.1 DM-SDH Algorithm Running on $k$-d Tree

According to the “cookbook” of algorithm [13], time complexity describes a function that quantifies the amount of time taken by an algorithm on certain input. When DM-SDH traveling on $k$-d tree, the time is merely spent on two operations (i), recursively resolving the cell pairs; operation (ii), computing the direct pairwise distances for the cells that are unresolvable at leaf level.

For the operation (i) in 2D system, as a given bucket width $p'$, DM-SDH starts at $DM_0$. Assume there are $I'$ pairs of cells need to be resolved on $DM_0$, the next level will generate $2^2 I'$ pairs of subcells in total. Also, leaving $2^2 I' \alpha_0$ pairs unresolved on the $DM_1$. So on and so forth, we have $2^{2(n-1)} I \alpha_0 \alpha_1 \cdots \alpha_{n-1}$
pairs unresolved on $D M_n$. Adding them together, we obtain the running time $T'_i(N)$ of operation (i).

$$ T'_i(N) = I' + 2^2 I' \alpha_0 + 2^4 I' \alpha_0 \alpha_1 + \cdots + 2^{2(n-1)} I' \alpha_0 \alpha_1 \cdots \alpha_{n-1} $$  \hspace{1cm} (6.1)

where $n$ is the total number of levels in $k$-d tree visited by the algorithm. According to the Theorem 2, by assuming DM-SDH algorithm starts at square cell (as we will see later, by assuming the algorithm starts at rectangular cell will not affect our mathematical analysis), we have $\frac{\alpha(m+1)}{\alpha(m)} = \frac{\sqrt[3]{2}}{2}$ and $\frac{\alpha(m+2)}{\alpha(m+1)} = \frac{\sqrt[3]{2}}{3}$, plugging these ratio into Equation 6.1, yields,

$$ T'_i(N) = I' \left[ 1 + 2^2 \cdot \frac{3}{4} + 2^4 \cdot \frac{1}{2} + 2^6 \cdot \frac{1}{2} \cdot \frac{3}{4} + 2^8 \cdot \left( \frac{1}{2} \right)^2 \right. $$

$$ + 2^{10} \cdot \left( \frac{1}{2} \right)^2 \cdot \frac{3}{4} + 2^{12} \cdot \left( \frac{1}{2} \right)^3 + \cdots $$

$$ + \left. 2^{2(n-2)} \cdot \left( \frac{1}{2} \right)^{(n-1)} \cdot \frac{3}{4} + 2^{2(n-1)} \cdot \left( \frac{1}{2} \right)^{(n-1)} \right] $$

(6.2)

Clearly, the value of $n$ increases by 2 when $N$ increases to $2^2 N$. Thus by revisiting the above equation, we have the following recurrence:

$$ T_i(N) = 2^3 \cdot T_i \left( \frac{N}{2^2} \right) $$

By applying the 1st case of master theorem, obtains

$$ T_i(N) = \Theta \left( N^{\log_{2^2} 2^3} \right) = \Theta \left( N^{\frac{3}{2}} \right) $$

Then, let's discuss the cost of operation (ii). When height of tree increases one more level, the system size has to be scaled to $2N$, and the number of pairs that have to compute pairwise distances of the particles becomes $2^{n-1} I' \alpha$ where $\alpha \in \left\{ \frac{3}{4}, \frac{2}{3} \right\}$.

$$ \begin{cases} T_{ii}(N) = 4 \times \frac{3}{4} T_{ii} \left( \frac{N}{2} \right) & \text{if } \alpha = \frac{3}{4} \\ T_{ii}(N) = 4 \times \frac{2}{3} T_{ii} \left( \frac{N}{2} \right) & \text{if } \alpha = \frac{2}{3} \end{cases} $$

(6.3)
Figure 6.1: Growth rate of running time of brute-force computation in 2D system
By using the master theorem, gives

\[
\begin{align*}
T_{ii}(N) &= \Theta(N^{\log_2 3}) \\
T_{ii}(N) &= \Theta(N^{\log_2 8 \times \frac{3}{3}})
\end{align*}
\]  

(6.4)

Surprisingly, we found the \( N^{\frac{3}{2}} \) is geometric mean of \( N^{\log_2 3} \) and \( N^{\log_2 8 \times \frac{3}{3}} \), i.e.,

\[
\sqrt{N^{\log_2 3} \cdot N^{\log_2 8 \times \frac{3}{3}}} = \sqrt{N^{\log_2 3 \times \frac{8 \times 3}{3}}} = N^{\frac{3}{2}}
\]  

(6.5)

According to the Figure 6.1, the blue line represents the growth rate of brute-force computation starting at square shape cell, and the red line represents the growth rate of the brute-force computation starting at rectangle shape cell. These two cases alternatively merge into the middle line, which is exactly \( N^{\frac{3}{2}} \). The growth rate changes depending on much levels \( \lfloor \log_2 \frac{N}{4} \rfloor + 1 \) of tree will be increased at once. In terms of time complexity of a particular algorithm, we don’t really care how large is input size will be changed at once. We merely focus on the input size is constantly increasing and approaching to infinity. Thus, even though the operation (ii) includes these two cases, we might generalize them into one: when \( N \) approaches to infinity, the running time of brute-force computation is \( \Theta(N^{\frac{3}{2}}) \). Therefore, we can conclude that the time complexity of DM-SDH running on a \( k \)-d in 2D system is

\[
T(N) = T_i(N) + T_{ii}(N) + \Theta(N) = \Theta(N^{\frac{3}{2}})
\]

where \( \Theta(N) \), as we discussed in Section 4.1, is the running time to build a \( k \)-d tree on 2D system.

In 3D system, similarly, the running time \( T_i(N) \) of operation (i) is same as 2D system (Equation 6.1), but different unresolvable ratios. Regarding to the Chapter 5, we are using a geometry approach to prove the unresolvable ratios in 2D system, but if we are using the same approach to prove the unresolvable ratios in 3D system, it is extremely complicated and trivial. Based on the experimental results, the unresolvable ratios are quickly converged to \( \frac{5}{6} \), \( \frac{4}{5} \), and \( \frac{3}{4} \), respectively, when the DM-SDH algorithm visits the \( k \)-d tree in
3D system. With this hypothesis, we plugging these ratios into Equation 6.1 again, gives,

\[ T_i(N) = I' \left[ 1 + 2^2 \cdot \frac{5}{6} + 2^4 \cdot \frac{4}{5} + 2^6 \cdot \frac{1}{2} 
\quad + 2^8 \cdot \frac{1}{2} \cdot \frac{5}{6} + 2^{10} \cdot \frac{1}{2} \cdot \frac{5}{6} \cdot \frac{4}{5} + 2^{12} \cdot \left( \frac{1}{2} \right)^2 
\quad + 2^{14} \cdot \left( \frac{1}{2} \right)^2 \cdot \frac{5}{6} + 2^{16} \cdot \left( \frac{1}{2} \right)^2 \cdot \frac{5}{6} \cdot \frac{4}{5} + 2^{18} \cdot \left( \frac{1}{2} \right)^3 
\quad + \ldots 
\quad + 2^{2(n-3)} \cdot \left( \frac{1}{2} \right)^{\frac{n-1}{3}-1} \cdot \frac{5}{6} + 2^{2(n-2)} \cdot \left( \frac{1}{2} \right)^{\frac{n-1}{3}-1} \cdot \frac{5}{6} \cdot \frac{4}{5} + 2^{2(n-1)} \cdot \left( \frac{1}{2} \right)^{\frac{n-1}{3}} \right] \tag{6.6} \]

The value of \( n \) increases by 3 when \( N \) increases to \( 2^{3N} \). Thus, by revisiting the above equation, we have the following recurrence:

\[ T_i(N) = 2^5 \cdot T_i\left( \frac{N}{2^3} \right) \]

Again, by applying the 1st case of master theorem, obtains

\[ T_i(N) = \Theta\left( N^{\log_{2^3} 2^5} \right) = \Theta\left( N^{\frac{5}{3}} \right) \]

When height of tree increases one more level, the system size has to be scaled to \( 2N \), and the number of pairs that are required to be computed by brute-force computation becomes \( 2^{\frac{2(n-1)}{3}} I' \alpha \), where \( \alpha \in \left\{ \frac{5}{6}, \frac{4}{5}, \frac{3}{4} \right\} \).

\[
\begin{cases}
T_{ii}(N) = 4 \times \frac{5}{6} T'_{ii}\left( \frac{N}{2} \right) \\
T_{ii}(N) = 4 \times \frac{4}{5} T'_{ii}\left( \frac{N}{2} \right) \\
T_{ii}(N) = 4 \times \frac{3}{4} T'_{ii}\left( \frac{N}{2} \right)
\end{cases} \tag{6.7}
\]

By using the master theorem, gives

\[
\begin{cases}
T_{ii}(N) = \Theta\left( N^{\log_{2^3} 20} \right) \\
T_{ii}(N) = \Theta\left( N^{\log_{2^3} 16} \right) \\
T_{ii}(N) = \Theta\left( N^{\log_{2^3} 12} \right)
\end{cases} \tag{6.8}
\]
Similarly, we found the $N^{\frac{5}{3}}$ is geometric mean of $N^{\log_2 20 \frac{20}{6}}$, $N^{\log_2 16 \frac{16}{5}}$, and $N^{\log_2 12 \frac{12}{4}}$

$$3^{\frac{1}{3}} N^{\log_2 20 \frac{20}{6}} N^{\log_2 16 \frac{16}{5}} N^{\log_2 12 \frac{12}{4}} = 3^{\frac{1}{3}} N^{\log_2 20 \frac{20}{6} \times 16 \frac{16}{5} \times 12 \frac{12}{4}} = N^{\frac{5}{3}}$$ (6.9)

According to the Figure 6.2, the blue line represents the growth rate of brute-force computation starting at cubic cell, the red line represents the growth rate of the brute-force computation starting at cuboid cell, and the green line represents the growth rate of the brute-force computation starting at another cuboid cell. These three cases alternatively merge into the middle line, which is exactly $N^{\frac{5}{3}}$. The growth rate changes depending on much levels ($\lfloor \log_2 \frac{N}{t} \rfloor + 1$) of tree will be increased at once. Again, we just care about the input size is constantly increased and approaching to infinity. Thus, the operation (ii) includes these three cases, we may generalize them into one: when $N$ approaches to infinity, the running time of brute-force computation is $\Theta(N^{\frac{5}{3}})$. Therefore, we can conclude that the time complexity of DM-SDH running on a $k$-d in 2D system is

$$T(N) = T_i(N) + T_{ii}(N) + \Theta(N) = \Theta(N^{\frac{5}{3}})$$

where $\Theta(N)$, as we discussed in Section 4.1, is the running time to build a $k$-d tree on 3D system. Compared to Quad-tree and Oct-tree, the DM-SDH running on $k$-d has the same time complexity $\Theta(N^{\frac{2d-1}{d}})$, where $d \in \{2, 3\}$. This running time analysis of the algorithm, which visits on those tree structures, is from the sight of principle approach of algorithm analysis. However, in the practical applications, a set of specific parameters will be past into algorithm. Thus, the algorithm will visit different number of levels on Quad-tree (Oct-tree in 3D system) and $k$-tree. As we mentioned in Section 4.2.1 and 4.2.2, the algorithm will start/stop at corresponding or different levels of the tree, it results the differences that the number of resolutions and brute-force computations have been called by the algorithm. With these quantitative analysis we have done in previous chapters and sections, in the next section, we are going to dig into the algorithm performance analysis in different cases.

### 6.2 Comparative Study on DM-SDH Running on Different Tree Structure

In this section, we will discuss the performance of MD-SDH algorithm running on Quad-tree (Oct-tree in 3D system) and $k$-d on each potential case. Sometimes, different queries may force the algorithm to visit same depth on tree structure. But in our analysis, we focus on the each that results from the same query.
Figure 6.2: Growth rate of running time of brute-force computation in 3D system
6.2.1 Quad-Tree VS k-d Tree: Case 1

The SDH algorithm starts and ends at the corresponding levels on Quad-tree and k-d tree, i.e. Quad-tree starts at level $i$ and ends at level $(i + n)$, k-d tree starts at level $2i$ and ends at level $2(i + n)$. In this case, since they end at the corresponding level, they have the same unresolvable pairs leaving at leaf level. The number of distances that are required to be computed are equal, such that, they consumed a same amount of time on brute-force computation. So, in order to distinguish the performance of these two tree structures, we only compare the number of resolutions have been called when the algorithm travels on each of them.

As shown in Figure 6.3, Since level $i$ of Quad-tree and $2i$ of k-d tree are corresponding level, they have same pairs of nodes $I$ need to be resolved. On the Quad-tree, we have $C(4^i, 2) = I$ pairs of nodes at level $i$, and $\alpha_0$ is unresolvable rate at level $i$. If a pair of nodes are not resolvable at current level, it will generate 16 pairs of nodes at its children’s level. So, after the algorithm resolved the nodes at level $i$, it leaves $16\alpha_0 I$ pairs unresolved at level $i + 1$. Recursively, at level $i + 2$, it leaves $16^2\alpha_0\alpha_1 I$ pairs of nodes to be resolved. So on and so forth, the algorithm recursively visits all the necessary pairs until it hits leaf level. So, we have a recursive formula to calculate the number of resolutions.

$$S_n = \begin{cases} I & n = 0 \\ 16 \cdot \alpha_{n-1} \cdot S_{n-1} & n \geq 1 \end{cases}$$

Figure 6.3: DM-SDH algorithm travels on Quad-tree and k-d tree: case 1
We sum up all the resolutions for each level, we have total number of resolutions $R$ called on Quad-tree.

$$R = \sum_{i=0}^{n} S_i = I(1 + 16\alpha_0 + 16^2\alpha_0\alpha_1 + 16^3\alpha_0\alpha_1\alpha_2 + \cdots + 16^n\alpha_0\alpha_1\cdots\alpha_{n-1})$$  \hspace{1cm} (6.10)

According to Theorem 1, we have $\frac{\alpha(m+1)}{\alpha(m)} = \frac{1}{2}$, thus, we have

$$R = \left[ 1 + 16\alpha_0 + 16^2\alpha_0 \left(\frac{1}{2}\right) + 16^3\alpha_0 \left(\frac{1}{2}\right)^2 + 16^4\alpha_0 \left(\frac{1}{2}\right)^3 + \cdots + 16^n\alpha_0 \left(\frac{1}{2}\right)^{n-1} \right]$$

$$= I + \alpha_0 \cdot I \cdot \sum_{i=1}^{n} 16^i \left(\frac{1}{2}\right)^{i-1}$$  \hspace{1cm} (6.11)

On the $k$-d tree, $C(2^{2i}, 2) = C(4^i, 2) = I$ pairs of nodes at level $2i$ need to be resolved, and $\beta_0$ is unresolvable rate at level $2i$. If a pair of nodes can not be resolved at current level, it will reproduce $4\beta_0I$ pairs of nodes at its children’s level. Thus, after the algorithm resolved the nodes at level $2i$, leaves $4\beta_0I$ pairs of nodes unresolved at level $2i + 1$. Recursively, at level $2i + 2$, it leaves $4^2\beta_0\beta_1I$ pairs of nodes to be resolved. Similarly, the algorithm recursively visits all the necessary pairs until it hits leaf level.

$$S'_n = \begin{cases} 
I & \text{n = 0} \\
4 \cdot \beta_{n-1} \cdot S_{n-1} & \text{n ≥ 1} 
\end{cases}$$

We sum up all the resolutions for each level, we have total number of resolutions $R'$ called on $k$-d tree.

$$R' = \sum_{i=0}^{n} S'_i = I(1 + 4\beta_0 + 4^2\beta_0\beta_1 + 4^3\beta_0\beta_1\beta_2 + \cdots + 4^n\beta_0\beta_1\cdots\beta_{n-1})$$

$$+ \cdots + 4^{2n}\beta_0\beta_1\cdots\beta_{2n-1}$$  \hspace{1cm} (6.12)

According to Theorem 2, $\frac{\beta(m+1)}{\beta(m)} = \frac{3}{4}$ and $\frac{\beta(m+2)}{\beta(m+1)} = \frac{2}{3}$, we have,

$$R' = I \left[ 1 + 4\beta_0 + 4^2\beta_0 \left(\frac{2}{3}\right) + 4^3\beta_0 \left(\frac{2}{3}\right)^2 \left(\frac{3}{4}\right) + 4^4\beta_0 \left(\frac{2}{3}\right)^3 \left(\frac{3}{4}\right)^2 + \cdots + 4^n\beta_0 \left(\frac{2}{3}\right)^i \left(\frac{3}{4}\right)^{\frac{i(i-1)}{2}} \right]$$

$$+ \cdots + 4^{2n}\beta_0 \left(\frac{2}{3}\right)^n \left(\frac{3}{4}\right)^{n-1}$$

$$= I + \beta_0 \cdot I \cdot \sum_{i=1}^{n} 4^i \left(\frac{2}{3}\right)^i \left(\frac{3}{4}\right)^{\frac{i(i-1)}{2}}$$  \hspace{1cm} (6.13)
In the Quad-tree, we can let the $R$ be a function of $n$, $R = f(n)$. Similarly, in $k$-d tree, we can let the $R'$ be another function of $n$, $R' = g(n)$. Thus, we have the figure to illustrate the relationship between $f(n)$ and $g(n)$, as shown in Figure 6.4. The blue curve indicates the growth rate of total resolutions have been called on Quad-tree, the red curve indicates the growth rate of total resolutions have been called on $k$-d tree.

We were using the Matlab to approximate these two curves. If we zoom in the figure, we found $f(n) > g(n)$ all the time, so $R > R'$. Consequently, in this case, we can conclude that the number of resolutions have been called on the Quad-tree is more than $k$-d tree, and performance of $k$-d is better than the performance of Quad-tree.

6.2.2 Quad-Tree VS $k$-d Tree: Case 2

The SDH algorithm starts at interim level (rectangle cell level) of $k$-d tree, preceding a half level of the Quad-tree, and both of them end at corresponding level, i.e. Quad-tree starts at level $i$ and ends at level $(i + n)$, $k$-d tree starts at level $(2i - 1)$ and ends at $2(i + n)$. Similarly, in this case, since they end at the corresponding level, they are going to call the same times of brute-force algorithm to compute the direct distances. Accordingly, the efficiency depends on how long they spent on tree traveling. We therefore compare the number of resolutions have been called when the algorithm travels on each of them.
As shown in Figure 6.5, the SDH starts on \(k\)-d tree a half level before the Quad-tree. On the Quad-tree, since minor modifications on bucket width will not impact the starting condition, such that we still have \(C(4^i, 2) = I\) pairs of nodes at level \(i\) to be resolved. Refers to the Equation 6.11, we have the same times of recursions called on Quad-tree.

\[
R = \sum_{i=0}^{n} S_i
\]

However, as we discussed in earlier chapter (Section 4.2.1), with the same parameter (bucket width), compared to the Quad-tree, the SDH will start a half level ahead on \(k\)-d tree. On the \(k\)-d tree, \(C(2^{2i-1}, 2) = I'\) pairs of nodes at level \(2i - 1\) need to be resolved, and \(\beta\) is uncoverable rate at the level \(2i - 1\). Similarly, after the algorithm resolved the nodes at level \(2i - 1\), it leaves \(4\beta I'\) pairs unresolved at level \(2i\). The rest of levels follow the same resolution method to resolve all the pairs of nodes until reach the leaf level. Therefore, we have the total number of recursions \(R'\) have been called on \(k\)-d tree.

\[
R' = I' + 4\beta I' + 4\beta I' \sum_{i=1}^{n} 4^i \left(\frac{3}{4}\right)^{\left\lfloor \frac{i}{2} \right\rfloor} \left(\frac{2}{3}\right)^{\left\lfloor \frac{i-1}{2} \right\rfloor}
\]  
(6.14)

We let \(R\) be a function of \(n\), \(R = f(n)\), and \(R'\) be another function of \(n\), \(R' = g(n)\). As shown in Figure 6.6, \(f(n)\) beats \(g(n)\) all the time, so \(R > R'\). In this case, we conclude that the number of resolutions have been called on the Quad-tree is more than \(k\)-d tree, and performance of \(k\)-d is better than the performance of Quad-tree.
6.2.3 Quad-Tree VS $k$-d Tree: Case 3

The SDH algorithm starts at corresponding level on Quad-tree and $k$-d, but ends at the interim level (rectangular cell level) of $k$-d tree, a half level further of Quad-tree, i.e. Quad-tree starts at level $i$ and ends at level $(i + n)$, $k$-d tree starts at level $2i$ and ends at $2(i + n) + 1$. In this case, since the $k$-d tree has a half more level over than Quad-tree, more nodes are resolved by the DM-SDH algorithm, less brute-force computations are required to compute the direct distances. Thus, simply compare the number of resolutions will not illustrate the difference of efficiency, we have to consider the number of brute-force computations.

Figure 6.7: DM-SDH algorithm travels on Quad-tree and $k$-d tree: case 3
As shown in Figure 6.7, the SDH starts on Quad-tree and $k$-d tree at the corresponding level, accordingly, $C(4^i, 2) = C(2^{2i}, 2)$, they hence have same pairs of nodes $I$ need to be resolved. Regarding to the case 1, the number of resolutions have been called on Quad-tree are still

$$R = \sum_{i=0}^{n} S_i$$

However, there are $2n + 1$ levels on the $k$-d tree, thus, obtains

$$R' = \sum_{i=0}^{2n+1} S'_i \quad (6.15)$$

Figure 6.8: The relationship between $f(n)$ and $g(n)$ in case 3

We let $R$ be a function of $n$, $R = f(n)$, and $R'$ be another function of $n$, $R' = g(n)$. As shown in Figure 6.8. With a sufficiently large $n$, $f(n) < g(n)$, so $R < R'$.

In addition, the number of brute-force computations are different at their leaf levels. We assume there are $J$ pairs of points left at leaf level $(i + n)$ of Quad-tree. Correspondingly, there are $J$ pairs left at level $2(i + n)$ of $k$-d tree, after one resolution on each pair, there are $\frac{3}{4}J$ pairs left at leaf level $2(i + n) + 1$ of
false

$k$-d tree. Each pair merely requires one direct distance computation, compared to the Quad-tree, we can easily recognize the $k$-d tree only require three quarters of brute-force computations. In practice, in terms of efficiency, resolving a quarter of pairs is a valuable improvement, because the leaf level of a complete tree structure usually has relatively large amount of nodes. Therefore, if we have large number of data, we significantly improve the efficiency by using the $k$-d tree.

6.2.4 Quad-Tree VS $k$-d Tree: Case 4

The SDH algorithm starts at the interim level (rectangular cell level) of $k$-d tree, preceding a half level of Quad-tree, and also ends at the interim level (rectangular cell level) of $k$-d tree, a half level further of Quad-tree, i.e. Quad-tree starts at level $i$ and ends at level $(i + n)$, $k$-d tree starts at level $2i - 1$ and ends at $2(i + n) + 1$. Likewise, the number of brute-force computations are different, in order to study the efficiency, we have to concern their number of resolutions and brute-force computations.

As shown in Figure 6.9, the SDH starts on $k$-d tree a half level before the Quad-tree. Similarly, at the starting level, the number of the pairs of nodes on $k$-d tree are a half of Quad-tree, i.e. $C(4^i, 2) = I$ pairs on the starting level of Quad-tree, $C(2^{2i-1}, 2) = I'$ pairs on the starting level of k-d tree. Regarding to the case 1, the number of resolutions have been called on Quad-tree are still

$$R = \sum_{i=0}^{n} S_i$$
However, compared to the Quad-tree, the SDH will start a half level ahead on \( k \)-d tree. On the \( k \)-d tree, \( I' \) pairs of nodes at level \( 2i - 1 \) need to be resolved, and \( \beta \) is uncoverable rate at the level \( 2i - 1 \). At level \( 2i - 1 \), it leaves \( 4 \beta \cdot I' \) pairs unresolved at level \( 2i \). The rest of levels follow the same pattern to resolve all the pairs of nodes until reach the leaf level \( 2(i + n) + 1 \). there are \( 2n + 1 \) levels in total. Therefore, we have the total number of resolutions \( R' \) have been called on \( k \)-d tree.

\[
R' = I' + 4 \cdot \beta \cdot I' + 4 \cdot \beta \cdot I' \cdot \sum_{i=1}^{n+2} 4^i \left( \frac{3}{4} \right)^{\frac{1}{2}i} \left( \frac{2}{3} \right)^{\frac{1}{2}i-1}
\]

(6.16)

![Figure 6.10: The relationship between \( f(n) \) and \( g(n) \) in case 4](image)

We let \( R \) be a function of \( n \), \( R = f(n) \), and \( R' \) be another function of \( n \), \( R' = g(n) \). As shown in Figure 6.10, \( f(n) < g(n) \), so \( R < R' \).

The number of brute-force computations are different at their leaf levels. We assume there are \( J \) pairs of points left at leaf level \( (i + n) \) of Quad-tree. Correspondingly, there are \( J \) pairs left at level \( 2(i + n) \) of \( k \)-d tree, after one resolution on each pair, there are \( \frac{3}{4}J \) pairs left at leaf level \( 2(i + n) + 1 \) of \( k \)-d tree. The \( k \)-d tree eliminates a quarter of brute-force computations, so the time that are required to compute direct distance on \( k \)-d tree are shortened to three quarters of Quad-tree.
6.2.5 Oct-Tree VS $k$-d Tree: Case 1

The SDH algorithm starts and ends at the corresponding levels on Oct-tree and $k$-d tree, i.e. Oct-tree starts at level $i$ and ends at level $(i + n)$, $k$-d tree starts at level $3i$ and ends at level $3(i + n)$. In this case, since they end at the corresponding level, they have the same unresolvable pairs leaving at leaf level. The number of distances that are required to be computed are equal, such that, they consumed a same amount of time on brute-force computation. In order to distinguish the performance of these two tree structures, we merely compare the number of resolutions have been called when the algorithm travels on each of them.

![Image](attachment:image.png)

Figure 6.11: DM-SDH algorithm travels on Oct-tree and $k$-d tree: case 1

As shown in Figure 6.11, since level $i$ of Oct-tree and level $3i$ of $k$-d tree are corresponding level, $C(8^i, 2) = C(2^{3i}, 2)$, they both have $I$ pairs of nodes need to be resolved. On the Oct-tree, we have $I$ pairs at level $i$, and $\alpha_0$ is unresolvable rate at level $i$. If a pair is not resolvable at current level, it will generate 64 pairs at its children’s level. So, after the algorithm resolved the nodes at level $i$, it leaves $64\alpha_0 I$ pairs unresolved at level $i + 1$. Recursively, at level $i + 2$, it leaves $64^2\alpha_0\alpha_1 I$ pairs to be resolved. So on and so forth, the algorithm recursively visits all the necessary pairs until it reaches the leaf level.

$$S_n = \begin{cases} 
I & n = 0 \\
64 \cdot \alpha_{n-1} \cdot S_{n-1} & n \geq 1
\end{cases}$$

We sum up all the number of resolutions of each level, we have total recursions $R$ have been call on Oct-tree.

$$R = \sum_{i=0}^{n} S_i = I(1 + 64\alpha_0 + 64^2\alpha_0\alpha_1 + 64^3\alpha_0\alpha_1\alpha_2 + \cdots + 64^n\alpha_0\alpha_1\cdots\alpha_{n-1}) \quad (6.17)$$
Similarly, according to Theorem 1, we have \( \frac{\alpha(m+1)}{\alpha(m)} = \frac{1}{2} \), yields

\[
R = I \left[ 1 + 64\alpha_0 + 64^2\alpha_0 \left(\frac{1}{2}\right) + 64^3\alpha_0 \left(\frac{1}{2}\right)^2 + 64^4\alpha_0 \left(\frac{1}{2}\right)^3 + \cdots + 64^n\alpha_0 \left(\frac{1}{2}\right)^{n-1} \right]
\]

\[
= I + \alpha_0 \cdot I \cdot \sum_{i=1}^{n} 64^i \left(\frac{1}{2}\right)^{i-1}
\]

(6.18)

On the \( k \)-d tree, \( I \) pairs of nodes at level \( 3i \) need to be resolved, and \( \beta_0 \) is unsolvable rate at level \( 3i \). If a pair can not be resolved at current level, it will generate 4 pairs at its children’s level. So, after the algorithm resolved the nodes at level \( 3i \), it leaves \( 4\beta_0I \) pairs unresolved at level \( 3i + 1 \). Recursively, at level \( 3i + 2 \), it leaves \( 4^2\beta_0\beta_1I \) pairs of nodes to be resolved. So on and so forth, the algorithm recursively visits all the necessary pairs until it reaches the leaf level.

\[
S'_n = \begin{cases} 
I & n = 0 \\
4 \cdot \beta_{n-1} \cdot S_{n-1} & n \geq 1
\end{cases}
\]

By summing up all the number of resolutions for each level, we have total recursions \( R' \) have been called on \( k \)-d tree.

\[
R' = \sum_{i=0}^{n} I(1 + 4\beta_0 + 4^2\beta_0\beta_1 + 4^3\beta_0\beta_1\beta_2 + \cdots + 4^n\beta_0\beta_1\cdots\beta_{n-1})
\]

\[
+ \cdots + 4^2n\beta_0\beta_1\cdots\beta_{2n-1} + \cdots + 4^3n\beta_0\beta_1\cdots\beta_{3n-1})
\]

(6.19)

This equation is similar to 2D system (Equation 6.12). However, because of the difficulty, we did not mathematically prove the resolvable ratios of \( k \)-d in the 3D system. Fortunately, we did a certain number of experiments on 3D system, and we found that the resolvable ratios have same relationship: \( \alpha \cdot \beta \cdot \gamma = \frac{1}{2} \).

In addition, \( \alpha \), \( \beta \), and \( \gamma \) are quickly converged to \( \frac{5}{6}, \frac{4}{5}, \) and \( \frac{3}{4} \), respectively. Consequently, we use these unresolvable ratios \( \frac{\text{ResolveRate}(m+1)}{\text{ResolveRate}(m)} = \frac{5}{6}, \frac{\text{ResolveRate}(m+2)}{\text{ResolveRate}(m+1)} = \frac{4}{5}, \) and \( \frac{\text{ResolveRate}(m+3)}{\text{ResolveRate}(m+2)} = \frac{3}{4} \) to analyze the
cases of $k$-d tree in 3D system. Then, we have,

$$
R' = I \left[ 1 + 4\beta_0 + 4^2\beta_0 \left( \frac{4}{5} \right) + 4^3\beta_0 \left( \frac{4}{5} \right) \left( \frac{3}{4} \right) + 4^4\beta_0 \left( \frac{4}{5} \right) \left( \frac{3}{4} \right) \left( \frac{5}{6} \right) \\
+ \cdots + 4^n\beta_0 \left( \frac{4}{5} \right) \left( \frac{3}{4} \right) \left( \frac{5}{6} \right) \left( \frac{i-1}{3} \right) \right] + \cdots + 4^{3n}\beta_0 \left( \frac{4}{5} \right) \left( \frac{3}{4} \right) \left( \frac{5}{6} \right) \left( \frac{n-1}{3} \right) \right] \\
= I + \beta_0 \cdot I \cdot \sum_{i=1}^{3n} 4^i \left( \frac{4}{5} \right) \left( \frac{3}{4} \right) \left( \frac{5}{6} \right) \left( \frac{i-1}{3} \right) \right) \\
$$

(6.20)

In the Oct-tree, we can let the $R$ be a function of $n$, $R = f(n)$. Similarly, in $k$-d tree, we can let the $R'$ be another function of $n$, $R' = g(n)$. Thus, we used the Matlab to plot the curves $f(n)$ and $g(n)$, as shown in Figure 6.11. The blue curve indicates the total resolutions have been called on Oct-tree, red curve indicates indicates the total resolutions have been called on $k$-d tree.

Figure 6.12: The relationship between $f(n)$ and $g(n)$ in 3D case 1
According to the Figure 6.12, Similarly, if we zoom in this figure, $f(n)$ beats $g(n)$ all the time, so $R > R'$. Consequently, in this case, we can conclude that the number of resolutions have been called on the Oct-tree is more than $k$-d tree, and performance of $k$-d tree is better than performance of Oct-tree.

### 6.2.6 Oct-Tree VS $k$-d Tree: Case 2

The SDH algorithm starts at the corresponding levels on Oct-tree and $k$-d tree, i.e. starts at level $i$ of Oct-tree, starts at level $3i$ of $k$-d tree; the algorithm ends at a third level further on the $k$-d tree, i.e. ends at level $(i + n)$ of Oct-tree, ends at level $3(i + n) + 1$ of $k$-d tree. In this case, since $k$-d tree has a third of level over than Oct-tree, more nodes are resolved by the algorithm, and less brute-force computations are required to compute the direct distances. Thus simply compare the number of resolutions will not illustrate the difference of the performance, we have to discuss the brute-force computations as well. As shown in Figure 6.13, the SDH starts at the same of Oct-tree and $k$-d tree, accordingly, $C(8^i, 2) = C(2^{3i}, 2)$, they have same pairs of nodes I need to be resolved.

![Figure 6.13: DM-SDH algorithm travels on Oct-tree and $k$-d tree: case 2](image)

Regarding to the case 1, the number of resolutions have been called on Oct-tree are still

$$R = \sum_{i=0}^{n} S_i$$

However, there are $3n+1$ levels on the $k$-d tree, thus, by apply Equation 6.20, we have

$$R' = \sum_{i=0}^{n+1} S_i'$$
We let $R$ be a function of $n$, $R = f(n)$, and $R'$ be another function of $n$, $R' = g(n)$. As shown in Figure 6.14, $f(n) < g(n)$, so $R < R'$.

In addition, the number of brute-force computations are different at their leaf levels. We assume there are $J$ pairs of points left a leaf level $(i + n)$ of Oct-tree, Correspondingly, there are $J$ pairs left at level $3(n + i)$ of $k$-d tree, after one resolution on each pairs, there are $\frac{5}{6}J$ pairs left at leaf level $3(i + n) + 1$ of $k$-d tree. Each pair requires one direct distances computation, compared to the Oct-tree, $k$-d tree saves one sixth of brute-force computations. In practice, the leaf level has most number of nodes, $k$-d tree therefore reduces number of large number brute-force computations and improve the efficiency of the SDH algorithm.

6.2.7 Oct-Tree VS $k$-d Tree: Case 3

The SDH algorithm starts at the corresponding levels on Oct-tree and $k$-d tree, i.e. starts at level $i$ of Oct-tree, starts at level $3i$ of $k$-d tree; the algorithm ends at two levels further on the $k$-d tree, i.e. ends at level $(i + n)$ of Oct-tree, ends at level $3(i + n) + 2$ of $k$-d tree. In this case, since $k$-d tree has two third of level over than Oct-tree, more nodes are resolved by the algorithm, and less brute-force computations are
required to compute the direct distances. Thus simply compare the number of resolutions will not illustrate the difference of the performance, we have to discuss the brute-force computations as well. As shown in Figure 6.15, the SDH starts at the same of Oct-tree and $k$-d tree, accordingly, $C(8^i, 2) = C(2^{3i}, 2)$, they have same pairs of nodes I need to be resolved.

Regarding to the case 1, the number of resolutions have been called on Oct-tree are still

$$R = \sum_{i=0}^{n} S_i$$

However, there are $3n+2$ levels on the $k$-d tree, thus, by apply Equation 6.20, we have

$$R' = \sum_{i=0}^{n+2} S'_i$$

We let $R$ be a function of $n$, $R = f(n)$, and $R'$ be another function of $n$, $R' = g(n)$. As shown in Figure 6.16, $f(n) < g(n)$, so $R < R'$.

The number of brute-force computations are different at their leaf levels. We assume there are $J$ pairs of points left a leaf level $(i + n)$ of Oct-tree, Correspondingly, there are $J$ pairs left at level $3(n + i)$ of $k$-d tree, after one resolution on each pair, there are $\frac{5}{6}J$ pairs left at leaf level $3(i + n) + 1$ of $k$-d tree. After one more resolution on each pair, there are $\left(\frac{5}{6} \times \frac{4}{5}\right)J = \frac{2}{3}J$ pairs left at leaf level $3(i + n) + 2$ of $k$-d tree. Compared to the Oct-tree, $k$-d tree saves a third of brute-force computations. In practice, the leaf level has relatively large number of nodes, $k$-d tree therefore reduces number of large number brute-force computations and dramatically improve the efficiency of the SDH algorithm.
6.2.8 Oct-Tree VS k-d Tree: Case 4

The SDH algorithm starts at the interim level (rectangle cell level) of k-d tree, preceding a third level of the Oct-tree, i.e. starts Oct-tree at level $i$, starts k-d tree at level $3i - 1$; the algorithm ends at the corresponding level of both two trees, $i + n$ and $3(i + n)$ on Oct-tree and k-d tree, respectively. In this case, since they end at the corresponding level, the number of brute-force computations called on k-d tree is identical with Oct-tree. Thus, the difference merely lies on how long they spent on tree traveling. Similarly, we compare the number of resolutions have been called when the algorithm visits on each of them. As shown in Figure 6.17, the SDH algorithm starts on k-d a third level before the Oct-tree. On the Oct-tree, since minor modifications on bucket width will not impact the starting condition, such that we still have $C(s^i, 2) = I$ pairs of nodes at level $i$ to be resolved.

The number of resolutions on Oct-tree still

$$R = \sum_{i=0}^{n} S_i$$
However, as we discuss in earlier chapter (Section 4.2.2), unlike Oct-tree, the modifications of bucket width on certain range will change the algorithm starts at a third level ahead on $k$-d tree. On the $k$-d tree, $C(2^{3i-1}, 2) = I'$ pairs of nodes at level $3i - 1$ need to be resolved, and $\beta$ is unresolvable rate at the level $3i - 1$. After the algorithm resolved the nodes at level $3i - 1$, it leaves $4\beta I'$ pairs unresolved at level $3i$. The algorithm follows the same pattern to resolve all the internal nodes on rest of levels until reaches the leaf level. So, we can plug the Equation 6.18 into the rest of levels. Obtains,

$$R' = I' + 4\beta I' + 4\beta I' \cdot \sum_{i=1}^{3n} S'_i$$

Figure 6.17: DM-SDH algorithm travels on Oct-tree and $k$-d tree: case 4

Figure 6.18: The relationship between $f(n)$ and $g(n)$ in 3D case 4
Let \( R \) be a function of \( n \), \( R = f(n) \), and \( R' \) be another function of \( n \), \( R' = g(n) \). As shown in Figure 6.18, \( f(n) > g(n) \), so \( R > R' \). So in this case, we conclude that the number of resolutions have been called on the Oct-tree is more than \( k \)-d tree. The SDH algorithm running on the \( k \)-d three therefore has better performance.

### 6.2.9 Oct-Tree VS \( k \)-d Tree: Case 5

The SDH algorithm starts at the interim level (rectangle cell level) of \( k \)-d tree, preceding a third level of the Oct-tree, i.e. starts Oct-tree at level \( i \), starts \( k \)-d tree at level \( 3i - 1 \); the algorithm ends at one third level further on the \( k \)-d tree, i.e. ends at \((i + n)\) of Oct-tree, ends at \(3(i + n) + 1\) of \( k \)-d tree. In this case, since \( k \)-d tree has one third of level over than Oct-tree, more nodes are resolved by the algorithm, and less brute-force computations are required to compute the direct distances. Thus, in order to study the performance of these two trees, we have to discuss the number of resolutions and brute-force computations. As shown in Figure 6.19, the SDH algorithm starts on \( k \)-d a third level before the Oct-tree. Similarly, on the Oct-tree, minor modifications on bucket width will not impact the starting condition, we still have \( C(8^i, 2) = I \) pairs of nodes at level \( i \) to be resolved.

![Diagram](image)

Figure 6.19: DM-SDH algorithm travels on Oct-tree and \( k \)-d tree: case 5

The number of resolutions on Oct-tree still

\[
R = \sum_{i=0}^{n} S_i
\]

Other than Oct-tree, the modifications of bucket width on certain range will change the algorithm starts at a third level ahead on \( k \)-d tree. On the \( k \)-d tree, \( C(2^{3i-1}, 2) = I' \) pairs of nodes at level \( 3i - 1 \) need to
be resolved, and $\beta$ is unresolvable rate at the level $3i - 1$. After the algorithm resolved the nodes at level $3i - 1$, it leaves $4\beta I'$ pairs unresolved at level $3i$. The algorithm follows the same pattern to resolve all the internal nodes on rest of levels until reaches the leaf level $3(i + n) + 1$. So, we can plug the Equation 6.18 into the rest of levels. Obtains,

$$R' = I' + 4\beta I' + 4\beta I' \cdot \sum_{i=1}^{3n+1} S'_{i}$$

![Figure 6.20: The relationship between $f(n)$ and $g(n)$ in 3D case 5](image)

Let $R$ be a function of $n$, $R = f(n)$, and $R'$ be another function of $n$, $R' = g(n)$. As shown in Figure 6.20, $f(n) < g(n)$, so $R < R'$.

Similar to case 2, the number of brute-force computations are different at their leaf levels. We assume there are $J$ pairs of points left a leaf level $(i + n)$ of Oct-tree, Correspondingly, there are $J$ pairs left at level $3(n + i)$ of k-d tree, after one resolution on each pair, there are $\frac{5}{6}J$ pairs left at leaf level $3(i + n) + 1$ of k-d tree. Each pair requires one direct distances computation, compared to the Oct-tree, k-d tree saves one sixth of brute-force computations. In practice, the leaf level has most number of nodes, k-d tree therefore reduces number of large number brute-force computations and improve the efficiency of the SDH algorithm.
6.2.10  Oct-Tree VS k-d Tree: Case 6

The SDH algorithm starts at the interim level (rectangle cell level) of k-d tree, preceding a third level of the Oct-tree, i.e. starts Oct-tree at level $i$, starts k-d tree at level $3i - 1$; the algorithm ends at two third level further on the k-d tree, i.e. ends at $(i + n)$ of Oct-tree, ends at $3(i+n)+2$ of k-d tree. In this case, since k-d tree has two third of level over than Oct-tree, more nodes are resolved by the algorithm, and less brute-force computations are required to compute the direct distances. Thus, we have to discuss the number of resolutions and brute-force computations. As shown in Figure 6.21, the SDH algorithm starts on k-d a third level before the Oct-tree. Similarly, on the Oct-tree, minor modifications on bucket width will not impact the starting condition, we still have $C(8^{i}, 2) = I$ pairs of nodes at level $i$ to be resolved.

![Figure 6.21: DM-SDH algorithm travels on Oct-tree and k-d tree: case 6](image)

The number of resolutions on Oct-tree still

$$R = \sum_{i=0}^{n} S_{i}$$

Other than Oct-tree, the modifications of bucket width on certain range will change the algorithm starts at a third level ahead on k-d tree. On the k-d tree, $C(2^{3i-1}, 2) = I'$ pairs of nodes at level $3i - 1$ need to be resolved, and $\beta$ is unresolvable rate at the level $3i - 1$. After the algorithm resolved the nodes at level $3i - 1$, it leaves $4\beta I'$ pairs unresolved at level $3i$. The algorithm follows the same pattern to resolve all the internal nodes on rest of levels until reaches the leaf level $3(i+n)+2$. So, we can plug the Equation 6.18 into the rest of levels. Obtains,

$$R' = I' + 4\beta I' + 4\beta I' \cdot \sum_{i=1}^{3n+2} S'_{i}$$

73
Figure 6.22: The relationship between \( f(n) \) and \( g(n) \) in 3D case 6

Let \( R \) be a function of \( n \), \( R = f(n) \), and \( R' \) be another function of \( n \), \( R' = g(n) \). As shown in Figure 6.22, \( f(n) < g(n) \), so \( R < R' \).

Similar to case 3, the number of brute-force computations are different at their leaf levels. We assume there are \( J \) pairs of points left a leaf level \((i+n)\) of Oct-tree, Correspondingly, there are \( J \) pairs left at level \( 3(n+i) \) of \( k\)-d tree, after one resolution on each pair, there are \( \frac{5}{3}J \) pairs left at leaf level \( 3(i+n) + 1 \) of \( k\)-d tree. After one more resolution on each pair, there are \( \left( \frac{5}{3} \times \frac{4}{3} \right) J = \frac{20}{9}J \) pairs left at leaf level \( 3(i+n) + 2 \) of \( k\)-d tree. Compared to the Oct-tree, \( k\)-d tree saves a third of brute-force computations. In practice, the leaf level has relatively large number of nodes, \( k\)-d tree therefore reduces number of large number brute-force computations and dramatically improve the efficiency of the SDH algorithm.

### 6.2.11 Oct-Tree VS \( k\)-d Tree: Case 7

The SDH algorithm starts at the interim level (rectangle cell level) of \( k\)-d tree, preceding two third level of the Oct-tree, i.e. starts Oct-tree at level \( i \), starts \( k\)-d tree at level \( 3i - 2 \); the algorithm ends at the corresponding level of both two trees, \( i + n \) and \( 3(i + n) \) on Oct-tree and \( k\)-d tree, respectively. In this case, since they end at the corresponding level, the number of brute-force computations called on \( k\)-d tree is
identical with Oct-tree. Consequently, the difference merely lies on how long they spent on tree traveling. Similarly, we compare the number of resolutions have been called when the algorithm visits on each of them. As shown in Figure 6.23, the SDH algorithm starts on $k$-d a third level before the Oct-tree. On the Oct-tree, since minor modifications on bucket width will not impact the starting condition, such that we still have $C(8^i, 2) = I$ pairs of nodes at level $i$ to be resolved.

![Figure 6.23: DM-SDH algorithm travels on Oct-tree and $k$-d tree: case 7](image)

The number of resolutions on Oct-tree still

$$R = \sum_{i=0}^{n} S_i$$

However, as we discuss in earlier chapter (Section 4.2.2), unlike Oct-tree, the modifications of bucket width on certain range will change the algorithm starts at a third level ahead on $k$-d tree. On the $k$-d tree, $C(2^{3i-2}, 2) = I'$ pairs of nodes at level $3i - 2$ need to be resolved, and $\beta'$ is unresolvable rate at the level $3i - 2$. After the algorithm resolved the nodes at level $3i - 2$, it leaves $4\beta'I'$ pairs unresolved at level $3i - 1$. The algorithm follows the same pattern to resolve all the internal nodes on rest of levels until reaches the leaf level $3i$. So, we can plug the Equation 6.18 into the rest of levels. Obtains,

$$R' = I' + 4\beta'I' + 16\beta'\beta I + 16\beta'\beta I \cdot \sum_{i=1}^{3n} S_i'$$

Let $R$ be a function of $n$, $R = f(n)$, and $R'$ be another function of $n$, $R' = g(n)$. As shown in Figure 6.18, $f(n) > g(n)$, so $R > R'$. So, in this case, we conclude that the number of resolutions have been
Figure 6.24: The relationship between \( f(n) \) and \( g(n) \) in 3D case 7

called on the Oct-tree is more than \( k \)-d tree. The SDH algorithm running on the \( k \)-d three therefore has better performance.

6.2.12 Oct-Tree VS \( k \)-d Tree: Case 8

The SDH algorithm starts at the interim level (rectangle cell level) of \( k \)-d tree, preceding two third level of the Oct-tree, i.e. starts Oct-tree at level \( i \), starts \( k \)-d tree at level \( 3i - 2 \); the algorithm ends at one third level further on the \( k \)-d tree, i.e. ends at level \((i + n)\) of Oct-tree, ends at \( 3(i + n) + 1 \) of \( k \)-d tree. In this case, since \( k \)-d tree has one third of level over than Oct-tree, more nodes are resolved by the algorithm, and less brute-force computations are required to compute the direct distances. Thus, we have to discuss the number of resolutions and brute-force computations. As shown in Figure 6.25, the SDH algorithm starts on \( k \)-d a third level before the Oct-tree. Similarly, on the Oct-tree, minor modifications on bucket width will not impact the starting condition, we still have \( C(8i, 2) = I \) pairs of nodes at level \( i \) to be resolved. The number of resolutions on Oct-tree still

\[
R = \sum_{i=0}^{n} S_i
\]
Figure 6.25: DM-SDH algorithm travels on Oct-tree and k-d tree: case 8

Other than Oct-tree, the modifications of bucket width on certain range will change the algorithm starts at a third level ahead on k-d tree. On the k-d tree, \( C(2^{3i-2}, 2) = I' \) pairs of nodes at level \( 3i - 2 \) need to be resolved, and \( \beta' \) is unresolvable rate at the level \( 3i - 2 \). After the algorithm resolved the nodes at level \( 3i - 2 \), it leaves \( 4\beta'I' \) pairs unresolved at level \( 3i - 1 \). The algorithm follows the same pattern to resolve all the internal nodes on rest of levels until reaches the leaf level \( 3(i + n) + 1 \). So, we can plug the Equation 6.18 into the rest of levels. Obtains,

\[
R' = I' + 4\beta'I' + 16\beta'\beta I + 16\beta'\beta I \cdot \sum_{i=1}^{3n+1} S_i'
\]

Figure 6.26: The relationship between \( f(n) \) and \( g(n) \) in 3D case 8
Let $R$ be a function of $n$, $R = f(n)$, and $R'$ be another function of $n$, $R' = g(n)$. As shown in Figure 6.26, $f(n) < g(n)$, so $R < R'$.

Similar to case 2 and case 5, the number of brute-force computations are different at their leaf levels. We assume there are $J$ pairs of points left a leaf level $(i + n)$ of Oct-tree, Correspondingly, there are $J$ pairs left at level $3(n + i)$ of k-d tree, after one resolution on each pair, there are $\frac{5}{6}J$ pairs left at leaf level $3(i + n) + 1$ of k-d tree. Each pair requires one direct distances computation, compared to the Oct-tree, k-d tree saves one sixth of brute-force computations. In practice, the leaf level has most number of nodes, k-d tree therefore reduces number of large number brute-force computations and improve the efficiency of the SDH algorithm.

6.2.13 Oct-Tree VS k-d Tree: Case 9

The SDH algorithm starts at the interim level (rectangle cell level) of k-d tree, preceding two third level of the Oct-tree, i.e. starts Oct-tree at level $i$, starts k-d tree at level $3i - 2$; the algorithm ends at two third level further on the k-d tree, i.e. ends at $(i + n)$ of Oct-tree, ends at level $3(i + n) + 2$ of k-d tree. In this case, since k-d tree has two third of level over than Oct-tree, more nodes are resolved by the algorithm, and less brute-force computations are required to compute the direct distances. Thus, we have to discuss the number of resolutions and brute-force computations. As shown in Figure 6.27, the SDH algorithm starts on k-d a third level before the Oct-tree.

Similarly, on the Oct-tree, minor modifications on bucket width will not impact the starting condition, we still have $C(8^i, 2) = I$ pairs of nodes at level $i$ to be resolved. The number of resolutions on Oct-tree
\[ R = \sum_{i=0}^{n} S_i \]

Other than Oct-tree, the modifications of bucket width on certain range will change the algorithm starts at a third level ahead on \( k \)-d tree. On the \( k \)-d tree, \( C(2^{3i-2}, 2) = I' \) pairs of nodes at level \( 3i - 2 \) need to be resolved, and \( \beta' \) is unresolvable rate at the level \( 3i - 2 \). After the algorithm resolved the nodes at level \( 3i - 2 \), it leaves \( 4\beta'I' \) pairs unresolved at level \( 3i - 1 \). The algorithm follows the same pattern to resolve all the internal nodes on rest of levels until reaches the leaf level \( 3(i + n) + 2 \). So, we can plug the Equation 6.18 into the rest of levels. Obtains,

\[ R' = I' + 4\beta'I' + 16\beta'\beta'I + 16\beta'\betaI \cdot \sum_{i=1}^{3n+2} S'_i \]

![Figure 6.28: The relationship between \( f(n) \) and \( g(n) \) in 3D case 9](image)

Let \( R \) be a function of \( n \), \( R = f(n) \), and \( R' \) be another function of \( n \), \( R' = g(n) \). As shown in Figure 6.28, \( f(n) < g(n) \), so \( R < R' \).
Similar to case 3 and case 6, the number of brute-force computations are different at their leaf levels. We assume there are \( J \) pairs of points left at a leaf level \((i + n)\) of Oct-tree. Correspondingly, there are \( J \) pairs left at level \( 3(n + i) \) of \( k \)-d tree, after one resolution on each pair, there are \( \frac{5}{6}J \) pairs left at leaf level \( 3(i + n) + 1 \) of \( k \)-d tree. After one more resolution on each pair, there are \( \left( \frac{5}{6} \times \frac{4}{5} \right)J = \frac{2}{3}J \) pairs left at leaf level \( 3(i + n) + 2 \) of \( k \)-d tree. Compared to the Oct-tree, \( k \)-d tree saves a third of brute-force computations. In practice, the leaf level has relatively large number of nodes, \( k \)-d tree therefore reduces number of large number brute-force computations and dramatically improve the efficiency of the SDH algorithm.

### 6.3 Space Complexity

Other than time complexity, space complexity describes a function that counts the amount of memory (space) taken by an algorithm on certain input.

For the convenience, in the memory, we assume each of tree nodes takes one space unit. Therefore, in order to measure difference of space complexity, we can simply compare the sums of all the tree nodes on Quad-tree (Oct-tree in 3D) and \( k \)-d tree, respectively. By given number of points \( N \) in simulated system, we can have the height \( \log_4 \frac{N}{\beta} \) and total number of nodes \( S_Q \) in Quad-tree.

\[
S_Q = \sum_{i=1}^{\log_4 \frac{N}{\beta}} 4^i
\]

\[
= 4(1 - 4^{\log_4 \frac{N}{\beta}}) \quad \text{(6.21)}
\]

\[
= 4 \cdot \left( \frac{N}{\beta} - 1 \right)
\]

Similarly, by given number of points \( N \) in simulated system, we can have the height \( \log_2 \frac{N}{\beta} \) and total number of nodes \( S_k \) in \( k \)-d tree.

\[
S_k = \sum_{i=1}^{\log_2 \frac{N}{\beta}} 2^i
\]

\[
= 2(1 - 2^{\log_2 \frac{N}{\beta}}) \quad \text{(6.22)}
\]

\[
= 2 \cdot \left( \frac{N}{\beta} - 1 \right)
\]
Since \( \frac{S_k}{S_Q} = \frac{3}{2} \), we can conclude that \( k \)-d tree requires at least \( \frac{3}{2} \) more space than Quad-tree, because sometimes \( k \)-d may stop at the interim level, which a half further of Quad-tree.

Correspondingly, by given number of points \( N \) in simulated system, we can have the height \( \log_8 \frac{N}{\beta} \) and total number of nodes \( S_O \) in Oct-tree.

\[
S_O = \sum_{i=1}^{\log_8 \frac{N}{\beta}} 8^i = \frac{8(1 - 8^{\log_8 \frac{N}{\beta}})}{1 - 8} = \frac{8}{7} \cdot \left( \frac{N}{\beta} - 1 \right)
\]

By comparing the Oct-tree and \( k \)-d tree, we have \( \frac{S_k}{S_O} = \frac{7}{4} \), we can conclude that \( k \)-d tree requires at least 5.7 more space than Oct-tree, because sometimes \( k \)-d may stop at one of the two interim levels, which one third or two thirds further of Quad-tree.

### 6.4 Performance on the Memory Hierarchy

These tree structures are implemented on general CPU computation, so memory hierarchy may result different performances on different tree structure. Locality of reference is one of approaches to measure the performance on the memory hierarchy. There are two types of reference locality: temporal locality and spatial locality. The latter refers to the neighbors of a used data tend to be used soon. In this section, we are going to discuss the locality of Quad-tree (Oct-tree in 3D) and \( k \)-d tree. Theoretically, the DM-SDH algorithm is a Depth-First Search algorithm, it traverses from the root level (certain level of tree) and explores all the way down to the leaf level. As illustrated in Figure 6.29, there are two subtrees \( A \) and \( B \) that lie on the bottom of a Quad-tree. Node \( A_0 \) has four children \( A_1, A_2, A_3, \) and \( A_4 \); node \( B_0 \) has four children \( B_1, B_2, B_3, \) and \( B_4 \). Block frame is on-chip L1 cache in memory hierarchy. When CPU references some objects, it first looks up the L1 cache, if it miss the target object, and then looks up the higher level of caches until the main memory. However, we merely discuss the L1 cache to expound the locality issue on Quad-tree and \( k \)-d tree. We assume the block frame has 8 blocks, and each block holds a node of tree.

According to DM-SDH algorithm (Section 3.2), by using the LRU replacement policy, the \( A_0 \) and \( B_0 \) are brought into the block frame to replace the nodes that are perviously replaced in the block frame. If they
are resolvable, it returns and reports the result. However, if they are unresolvable, the algorithm goes ahead to resolve their children. Because $A_1$ is need to be resolved with each of $B_0$’s children, $A_1$, $B_1$, $B_2$, $B_3$ and $B_4$ are been orderly replaced in block frame. Then, the algorithm is going to resolve the $A_2$ with each of $B_0$’s children. Luckily, $B_1$, $B_2$, $B_3$, and $B_4$ are already in there, we only have to replace $A_2$ into block frame to complete these resolutions. So far, all the previous nodes are been replaced out, the next reference is going to replace out the $A_0$, because it is the oldest node that have been used. $A_3$ therefore replaces the $A_0$. $B_1$, $B_2$, $B_3$, and $B_4$ are still in the block frame, thus the resolutions of $A_3$ to four of them could be successfully finished. Similarly, the resolutions of $A_4$ to four children of $B_0$ could be accomplished without replacing $B_1$, $B_2$, $B_3$, and $B_4$ again. If none of pairs are resolvable, the total number of replacing operations that are took by Quad-tree is 10.

Similarly, there are two subtrees $A$ and $B$ lie on the bottom of a $k$-d, shown as Figure 6.30. Node $A_0$ has two children $A_1$ and $A_2$ and four grandchildren $A_3$, $A_4$, $A_5$, and $A_6$. Node $B_0$ has two children $B_1$ and $B_2$ and four grandchildren $B_3$, $B_4$, $B_5$, and $B_6$. By using the same configurations of Quad-tree, $k$-d tree follows the same approach to resolve the nodes. Here, we just omit the verbal description of replacing process on $k$-d, but the result of each replacement can be found in the Figure 6.30. Accordingly, if none of pairs are resolvable, the total number of replacing operations that are took by $k$-d tree is 20. Therefore, the Quad-tree has better cache performance than $k$-d tree.
Figure 6.30: Cache performance on k-d tree
We have implemented both algorithms with the C programming language and our experiments were run on an Apple Xserve server with two Intel quad-core 2.4GHz processors and 16GB of memory. The operating system was MAC OS X 10.6 (Snow Leopard). We used uniform distributed dataset in our experiments. We first compare the running time of the DM-SDH algorithm running on Quad-tree and $k$-d tree. Results using 2D data inputs are plotted in Figure 7.1 and 7.2.

### 7.1 Experiments on 2D System

<table>
<thead>
<tr>
<th>2 Bucket</th>
<th>Quad-tree</th>
<th>$k$-d tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>200K</td>
<td>17.091</td>
<td>15.265</td>
</tr>
<tr>
<td>400K</td>
<td>41.944</td>
<td>41.768</td>
</tr>
<tr>
<td>600K</td>
<td>81.278</td>
<td>81.319</td>
</tr>
<tr>
<td>800K</td>
<td>136.726</td>
<td>122.365</td>
</tr>
<tr>
<td>1M</td>
<td>208.268</td>
<td>176.542</td>
</tr>
<tr>
<td>1.2M</td>
<td>295.522</td>
<td>241.596</td>
</tr>
<tr>
<td>1.4M</td>
<td>279.439</td>
<td>273.481</td>
</tr>
<tr>
<td>1.6M</td>
<td>336.738</td>
<td>333.665</td>
</tr>
<tr>
<td>1.8M</td>
<td>403.042</td>
<td>401.338</td>
</tr>
<tr>
<td>2M</td>
<td>476.433</td>
<td>476.818</td>
</tr>
</tbody>
</table>

Figure 7.1: Performance of the DM-SDH running on Quad-tree and $k$-d with 2 bucket query in 2D data.
Figure 7.2: Performance of the DM-SDH running on Quad-tree and $k$-d with 3 bucket query in 2D data.

The first thing we can realize from the figures is, $N$ is small, the running time is almost identical when Quad-tree and $k$-d tree have exact same leaf level. However, when $N$ increases, once the leaf level of $k$-d tree further than Quad-tree, the performance of $k$-d tree beats the Quad-tree.

### 7.2 Experiments on 3D System

We have also compare the running time of the DM-SDH algorithm running on Oct-tree and $k$-tree. Results using 3D data inputs are plotted in Figure 7.3, 7.4, and 7.5.

In the 3D system, the $k$-d beats the Oct-tree more than 2D system. Similarly, when $N$ is small, the running time is almost identical when Oct-tree and $k$-d tree have exact same leaf level. However, when $N$ increases, once the leaf level of $k$-d tree further than Oct-tree one or two levels, the performance of $k$-d tree beats the Oct-tree.
Figure 7.3: Performance of the DM-SDH running on Quad-tree and \( k \)-d with 3 bucket query in 3D data.

<table>
<thead>
<tr>
<th></th>
<th>Quad-tree</th>
<th>( k )-d tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>100K</td>
<td>60.967</td>
<td>60.853</td>
</tr>
<tr>
<td>300K</td>
<td>532.116</td>
<td>377.747</td>
</tr>
<tr>
<td>600K</td>
<td>1148.982</td>
<td>1141.673</td>
</tr>
<tr>
<td>900K</td>
<td>2461.788</td>
<td>2455.561</td>
</tr>
<tr>
<td>1.2M</td>
<td>6572.759</td>
<td>3784.876</td>
</tr>
</tbody>
</table>

Figure 7.4: Performance of the DM-SDH running on Quad-tree and \( k \)-d with 4 bucket query in 3D data.

<table>
<thead>
<tr>
<th></th>
<th>Quad-tree</th>
<th>( k )-d tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>100K</td>
<td>82.016</td>
<td>81.53</td>
</tr>
<tr>
<td>300K</td>
<td>714.496</td>
<td>511.209</td>
</tr>
<tr>
<td>600K</td>
<td>1554.062</td>
<td>1547.374</td>
</tr>
<tr>
<td>900K</td>
<td>3340.212</td>
<td>3326.95</td>
</tr>
<tr>
<td>1.2M</td>
<td>7531.516</td>
<td>6349.861</td>
</tr>
</tbody>
</table>
Figure 7.5: Performance of the DM-SDH running on Quad-tree and \( k \)-d with 7 bucket query in 3D data.
CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 Conclusion

In this thesis paper, we evaluated the performance of DM-SDH algorithm running on Quad-tree (Oct-tree in 3D) and binary tree to compute the important query named spatial distance histogram in scientific datasets. We noticed that the unsolvable ratios of the algorithm running on $k$-d trees quickly converge to $\frac{3}{4}$ and $\frac{2}{3}$ from simulation results, we therefore mathematically proved the unresolvable ratios $\alpha = \frac{3}{4}$ and $\beta = \frac{2}{3}$, where $\alpha$ is the unresolvable ratio from square cells to rectangle cells, and $\beta$ is unresolvable ratio from rectangle cells to square cells. We have illustrated time complexity of binary tree is identical with Quad-tree (Oct-tree in 3D), i.e. $\Theta(N^{\frac{2d-1}{d}})$ where $d$ is the dimension of data. On the other hand, we have shown the differences of Quad-tree and $k$-d tree in practical cases. Generally speaking, binary has better performance than Quad-tree (Oct-tree in 3D) in all cases, whereas binary tree requires large memory space and has extra overhead on process of caching. In addition, experiments we designed have solidified our theoretical analysis.

8.2 Future Work

Our work on this topic can be extended in multiple directions. First, based on the 3D experimental results, we found $\alpha = \frac{5}{6}$, $\beta = \frac{4}{5}$, and $\gamma = \frac{3}{4}$, further, $\alpha \cdot \beta \cdot \gamma = \frac{1}{2}$. However, we didn’t mathematically prove these ratios, because we were using geometric method to prove the unresolvable ratios in 2D system. If we are using the same approach to prove these ratios in 3D system, it is extremely complicated to do so. In addition, we made a bold speculation, in the 4D system, $\alpha \cdot \beta \cdot \gamma \cdot \delta = \frac{1}{2}$ still holds, and $\alpha = \frac{7}{8}$, $\beta = \frac{6}{7}$, $\gamma = \frac{5}{6}$, $\delta = \frac{4}{5}$. Second, the process of caching was not been deeply discussed. Third, the I/O cost of the algorithm was not been discussed, the performance of algorithm could be improved by implementing the pre-fetching mechanism.
REFERENCES


