A Triangulation Based Coverage Path Planning For a Mobile Robot With Circular Sensing Range

Vatana An
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/etd
University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
https://stars.library.ucf.edu/etd/5683
A TRIANGULATION BASED COVERAGE PATH PLANNING FOR A MOBILE ROBOT WITH CIRCULAR SENSING RANGE

by

VATANA AN
B. S. E. E. University of Florida, 2002
B. S. C. E. University of Florida, 2002
M. S. E. E. University of Central Florida, 2005

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the College of Engineering and Computer Science at the University of Central Florida Orlando, Florida

Fall Term
2017

Major Professor: Zhihua Qu
ABSTRACT

In this dissertation, two coverage path planning (CPP) approaches for a nonholonomic mobile robot are proposed. The first approach is the Local Coverage Path Planning (LCPP) approach which is designed for all sensing ranges. The second approach is the Global Coverage Path Planning (GCPP) approach which is designed for sufficient sensing range that can observe all points of interests in the target region (TR). The LCPP approach constructs CP after finding observer points for all local regions in the TR. The GCPP approach computes observer points after CP construction. Beginning with the sample TR, the LCPP approach requires 8 algorithms to find a smooth CP and sufficient number of observers for complete coverage. The Global Coverage Path Planning approach requires 17 algorithms to find the smooth CP with sufficient number of observers for completed coverage. The worst case running time for both approaches are quadratic which is consider to be very fast as compared to previous works reported in the literature. The main technical contributions of both approaches are to provide a holistic solution that segments any TR, uses triangulation to determine the line of sights and observation points, and then compute the smooth and collision-free CP. Both approaches provide localization, speed control, curvature control, CP length control, and smooth CP control. The first approach has applications in automate vacuum cleaning, search and rescue mission, spray painting, and etc. The second approach is best used in military and space applications as it requires infinite sensing range which only resource rich organizations can afford. At the very least, the second approach provides simulation opportunity and upper bound cost estimate for CPP. Both approaches will lead to a search strategy that provides the shortest CP with the minimum number of observer and with the shortest running time for any sensing range.
To my family
ACKNOWLEDGMENTS

My special thank of the Office of Naval Research for funding my PhD studies. I would like to thank Mr. David Vickers for letting me know about this wonderful PhD funding opportunity provided by Naval Surface Warfare Center (Panama City Division) which led me to receive more than $300,000 of research funding and ultimately the PhD degree from the university of Central Florida.

I would like to thank my professor, Professor Zhihua Qu, for patiently advising me throughout my graduate studies for the last decade. Professor Qu’s vast knowledge in the field of control engineering is truly admirable.

Finally, I would like to thank my family for their patience, encouragement, and love. My father always fosters his children to love continuous learning. My mother always cooks us great food so that we have energy to accomplish our goals. My wife always encourages me to make progress in all situations. She is also the greatest chef in addition to many leadership roles that she performs for the family. Her patience allows me to finish all of my goals while rearing our two beautiful and intelligent children.
# TABLE OF CONTENTS

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

LIST OF TABLES ........................................................................................................................................ xvii

LIST OF COMMON ABBREVIATIONS ................................................................................................. xx

CHAPTER 1 INTRODUCTION .................................................................................................................. 1

1.1 Background and Motivation ........................................................................................................... 2

1.2 Previous Work .................................................................................................................................... 5

1.3 Statement of Contributions ............................................................................................................. 10

CHAPTER 2 REVIEW OF COVERAGE PATH PLANNING CONTROL .............................................. 13

2.1 Introduction ...................................................................................................................................... 13

2.2 Heuristic Decomposition ................................................................................................................ 14

2.3 Approximate Decomposition .......................................................................................................... 14

2.4 Partial-approximate Decomposition ............................................................................................... 15

2.5 Exact Decomposition ....................................................................................................................... 16

2.6 Triangulation .................................................................................................................................... 18

2.7 Traveling Salesman Algorithm ......................................................................................................... 19

2.8 Discontinuous Coverage Path Planning ......................................................................................... 20

2.9 Cubic Spline Interpolation ............................................................................................................. 21

2.10 Conclusions .................................................................................................................................... 23

CHAPTER 3 A LOCAL COVERAGE PATH PLANNING APPROACH ........................................... 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Visible Polygon of a Regular Triangulation (RT)</td>
<td>26</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Visible Polygon</td>
<td>30</td>
</tr>
<tr>
<td>3.3</td>
<td>Row-Based Observer Placement Algorithm</td>
<td>32</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Row Based Observer Placement Theorem</td>
<td>36</td>
</tr>
<tr>
<td>3.4</td>
<td>Previous Next Waypoint Coverage Constraint Algorithm</td>
<td>42</td>
</tr>
<tr>
<td>3.5</td>
<td>Static Collision Avoidance</td>
<td>52</td>
</tr>
<tr>
<td>3.6</td>
<td>Smooth Adaptive Curvature Control</td>
<td>52</td>
</tr>
<tr>
<td>3.7</td>
<td>A Local Coverage Path Planning Approach</td>
<td>57</td>
</tr>
<tr>
<td>3.8</td>
<td>Coverage Path Planning For a Group of Robots</td>
<td>66</td>
</tr>
<tr>
<td>3.9</td>
<td>Conclusions</td>
<td>69</td>
</tr>
<tr>
<td>CHAPTER 4</td>
<td>A GLOBAL COVERAGE PATH PLANNING APPROACH</td>
<td>71</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>71</td>
</tr>
<tr>
<td>4.2</td>
<td>The RedPath Algorithm</td>
<td>77</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Foundational Coverage Path Algorithm, the Graham Scan’s</td>
<td>78</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Input of the RedPath Algorithm</td>
<td>78</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Output of the RedPath Algorithm</td>
<td>79</td>
</tr>
<tr>
<td>4.3</td>
<td>The OrangePath Algorithm</td>
<td>80</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Relative Position of Line Segment Algorithm, the BURL’s algorithm</td>
<td>81</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Input of the OrangePath Algorithm</td>
<td>84</td>
</tr>
</tbody>
</table>
4.3.3 Output of the OrangePath Algorithm ................................................................. 84

4.4 The YellowPath Algorithm .................................................................................... 86
  4.4.1 Collision Avoidance Algorithms, external tangent’s and internal tangent’s .... 87
  4.4.2 Input of the YellowPath Algorithm ................................................................. 88
  4.4.3 Output of the YellowPath Algorithm ............................................................... 88

4.5 The GreenPath Algorithm .................................................................................... 90
  4.5.1 Visibility of Crossing Coverage Path Algorithms, CSIRT’s ....................... 94
  4.5.2 Visible circle path modification .................................................................... 98
  4.5.3 Input of the GreenPath Algorithm ................................................................. 99
  4.5.4 Output of the GreenPath Algorithm ............................................................... 99

4.6 The BluePath Algorithm ....................................................................................... 101
  4.6.1 Differentiability Algorithm, Disk Enlargement’s ............................................ 102
  4.6.2 Input of the BluePath Algorithm ................................................................. 102
  4.6.3 Output of the BluePath Algorithm ............................................................... 103

4.7 The IndigoPath Algorithm ..................................................................................... 103
  4.7.1 Visibility of Non-Crossing RT, Region to Visible Curve Segment Mapping .... 105
  4.7.2 LosPoRT with multiple RTs ........................................................................... 115
  4.7.3 Input of the IndigoPath Algorithm ................................................................. 122
  4.7.4 Output of the IndigoPath Algorithm ............................................................. 122

4.8 The VioletPath Algorithm ..................................................................................... 124
LIST OF FIGURES

FIGURE 1. A SAMPLE TARGET REGION (TR) WITH 18 DISKS .......................................................... 3
FIGURE 2. A ROBOT AND ITS SENSING RANGE (Rs) ................................................................. 4
FIGURE 3. APPROXIMATE DECOMPOSITION ............................................................................. 15
FIGURE 4. PARTIAL-APPROXIMATE DECOMPOSITION .............................................................. 15
FIGURE 5. AN EXACT PARTITION OF THE SAMPLE TARGET REGION INTO 24 RTs ................. 17
FIGURE 6. REGULAR TRIANGULATIONS (RTs) OF THE SAMPLE TR ........................................ 19
FIGURE 7. AN RT WITH ELEMENTARY NOTATION ..................................................................... 26
FIGURE 8. A PROGRESSION OF LINE SEGMENTS (LSs) TO DETERMINE A VP ...................... 27
FIGURE 9. A PROGRESSION OF LSs TO DETERMINE VISIBLE POLYGON ................................. 28
FIGURE 10. A PROGRESSION OF LSs TO HALF PLANES OF DISK 8 ......................................... 28
FIGURE 11. HALF PLANE INTERSECTION AND THE VP OF AN RT ........................................ 29
FIGURE 12. TRANSITION FROM RT TO A VISIBLE POLYGON (VP) .......................................... 31
FIGURE 13. A TYPICAL RT, ITS VP, ITS VP’S CENTROID, AND LENGTH ................................ 32
FIGURE 14. RT WITH THREE OBSERVERS COVERAGE/PATH .................................................. 34
FIGURE 15. ROW BASED OBSERVER PLACEMENT (RBOP) ALGORITHM ............................... 35
FIGURE 16. COMPLETE COVERAGE (ONE OR MORE OBSERVER PER RT) ............................ 37
FIGURE 17. COMPLETE COVERAGE (ONE OR MORE OBSERVER WAYPOINTS PER RT) ......... 37
FIGURE 18. COMPLETE COVERAGE (ONE OBSERVER PER RT) ............................................... 38
FIGURE 19. COMPLETE COVERAGE WITH LOCAL PLANNING ............................................... 38
FIGURE 20. COMPLETE COVERAGE PATH WITH LOCAL PLANNING ...................................... 39
FIGURE 21. COMPLETE COVERAGE PATH WITH LOCAL PLANNING ...................................... 39
FIGURE 22. THE ROW BASED APPROACH TO FIND SUFFICIENT NUMBER OF OBSERVERS. .. 40
FIGURE 23. RBOP IN RT 5 WHEN Rs IS 50. OBSERVERS ARE LEFT TO RIGHT AND FROM BOTTOM TO UP. ............................................................................................................................................................................. 40

FIGURE 24. COVERAGE OF AN RT PRIOR TO PNWCC ALGORITHM. ............................................. 41

FIGURE 25. COVERAGE OF AN RT DURING PNWCC ALGORITHM. ................................................ 41

FIGURE 26. A TYPICAL LINEAR SPLINE TOUR, Rs IS 130.4 .......................................................... 43

FIGURE 27. AN RT WITH OBSERVING SENSORS TURNED ON/OFF, Rs IS 130.4 ......................... 44

FIGURE 28. SEARCHING FOR A “BETTER” WAYPOINT FOR A RT.............................................. 44

FIGURE 29. LCPP WHEN Rs IS 130.4 .................................................................................................. 48

FIGURE 30. A LCPP AND AN IMPROVED LCPP WITH PNWCC, Rs IS 130.4 ............................. 49

FIGURE 31. LCPP, RED, TO IMPROVED LCPP, BLACK, WITH DIFFERENTIABLE CP, Rs IS 163.... 49

FIGURE 32. COMPLETE COVERAGE PATH FOR Rs EQUAL TO 163 ................................................ 50

FIGURE 33. COVERAGE OF AN RT PRIOR TO AND POST PNWCC ALGORITHM. .................... 51

FIGURE 34. COVERAGE OF AN RT AFTER 1 ITERATION OF PNWCC ALGORITHM. .................... 51

FIGURE 35. ADAPTIVE CURVATURE CONTROL FOR INPUT TOUR .............................................. 53

FIGURE 36. ADAPTIVE CURVATURE CONTROL FOR INPUT TOUR .............................................. 54

FIGURE 37. TRANSITION FROM ACs TO DESIRE TANGENT LS.................................................... 54

FIGURE 38. A SAMPLE TARGET REGION WITH 10 DISKS AND ITS 10 RTs................................. 59

FIGURE 39. TRANSITION FROM VPs TO CENTROIDAL CP FOR SUFFICIENT SENSING RANGE..... 60

FIGURE 40. VISIBILITY THROUGH VPVs AND STRAIGHT LINE INTERSECTION .......................... 60

FIGURE 41 (A). ACs AND THEIR REFERENCE LSS............................................................................. 61

FIGURE 42. TWO DIFFERENTIABLE CPS OF THE TR ................................................................. 61

FIGURE 43. THE TRANSITION FROM THE SAMPLE TR TO A SMOOTH CP WITH A LCPP ALGORITHM. ............................................................................................................................................................................. 62
FIGURE 44. CENTROIDAL LINEAR SPLINE TOUR................................................................................. 63
FIGURE 45. ACs OF THE LINEAR SPLINE CP.................................................................................. 63
FIGURE 46. SMOOTH CENTROIDAL CP DUE AC CONTROL ............................................................. 64
FIGURE 47. CENTROIDAL CP AFTER ONE ITERATION OF PNWCC ALGORITHM ......................... 65
FIGURE 48. CENTROIDAL CP AFTER TWO ITERATIONS OF PNWCC ALGORITHM ....................... 65
FIGURE 49. CENTROIDAL CP AFTER THREE ITERATIONS OF PNWCC ALGORITHM ................. 65
FIGURE 50. SMOOTH CENTROIDAL CP AND ITS THIRD ITERATION WITH PNWCC ALGORITHM .... 65
FIGURE 51. GROUP COVERAGE PATH PLANNING WITH LCPP APPROACH (RS IS 130.4) ............ 66
FIGURE 52. GROUP COMMUNICATION/SENSING MATRIX FOR Q NODES ................................. 67
FIGURE 53. LCPP WITH RBOP AND PNWCC ALGORITHMS, RS IS 130.4................................. 68
FIGURE 54. TRIANGULAR-BASED CPTS. A RAINBOW CP IS THE SHORTEST OF ALL THREE CPTS. 72
FIGURE 55. THE RAINBOW CP PLANNING ALGORITHM............................................................ 72
FIGURE 56. TRANSITION FROM THE TR TO THE RAINBOW CP PLANNING ALGORITHM RESULT .... 75
FIGURE 57. RAINBOW CPTS FOR ALL SEVEN PHASES............................................................. 77
FIGURE 58. THE REDPATH ALGORITHM .................................................................................... 77
FIGURE 59. THE REDPATH OF THE SAMPLE TARGET REGION .................................................. 79
FIGURE 60. THE ORANGE PATH OF THE SAMPLE TR ............................................................... 80
FIGURE 61. THE BURL ALGORITHM ON THE REDPATH............................................................ 82
FIGURE 62. EXAMPLE OF THE BURL ALGORITHM .................................................................... 82
FIGURE 63. AN EXAMPLE OUTPUT OF THE ORANGE PATH ALGORITHM ................................. 85
FIGURE 64. AN EXAMPLE INPUT AND OUTPUT OF THE YELLOW PATH ALGORITHM .............. 86
FIGURE 65. C++ COLLISION CORRECTION ALGORITHM............................................................ 88
FIGURE 66. AN EXAMPLE OUTPUT OF THE YELLOW PATH ALGORITHM ................................... 89
Figure 67. An example output of the YellowPath algorithm ........................................ 89
Figure 68. An example input and output of the GreenPath algorithm ................ 91
Figure 69. RTs of the TR with the YellowPath ............................................................ 92
Figure 70. RTs 5, 13, and 16 with CSIRT ................................................................. 93
Figure 71. A typical RT through CSIRT partition algorithm .................................. 95
Figure 72. CSIRT algorithm analysis of observer points placement .................... 95
Figure 73. Lemma 1’s illustration ............................................................................ 96
Figure 74. Theorem 1 illustration ........................................................................... 97
Figure 75. Visible path modification with visible circle concept ......................... 98
Figure 76. Visible path modification with visible circle concept ......................... 100
Figure 77. An example output of the GreenPath algorithm ..................................... 100
Figure 78. An example input and output of the BluePath algorithm ..................... 101
Figure 79. An example output of the BluePath algorithm ....................................... 103
Figure 80. An example input and output of the IndigoPath algorithm ................... 104
Figure 81. A given $T(A)$, with and without disks, to be observed from a coverage path
.......................................................................................................................................... 107
Figure 82. LosPort’s example and its vertices’ relation. ......................................... 108
Figure 83. A partitioned region and its typical coverage path ............................... 109
Figure 84. Inclusive VCS in the absent of Disk ......................................................... 109
Figure 85. Highlight the 4 cases of observability of two vertices on disk $j$ .......... 110
Figure 86 (a). BCS due to Disk $i$ .............................................................................. 111
Figure 87. Observability of vertices B, C, G, R, and Y on disk $j$ with the inclusion of
 Disk $i$ ................................................................................................................................... 112
Figure 111. The visible polygon of RT 7 due to e1, e2, e3, e4, e5, and e6....................... 143
Figure 112. A DLL for PNWCC algorithm ................................................................. 145
Figure 113. A declaration of waypoint data structure in C++................................. 146
Figure 114. A DLL class implementation in C++.................................................. 146
Figure 115. A robot in the presence of moving obstacles........................................... 153
Figure 116. A robot in the presence of “static” obstacles........................................... 154
Figure 117. Robot and moving obstacles’ trajectories.............................................. 162
Figure 118. Robot’s first modified trajectory based on obstacles seen..................... 163
Figure 119. Robot’s final trajectory based on obstacles seen................................... 164
Figure 120. Speed control (inch per second)............................................................ 165
Figure 121. Steering control (inch per second)......................................................... 166
Figure 122. Turning angle in degree with respect to time........................................ 166
Figure 123. The LCPP approaches in red and blue and the RCPP approach in green
respectively. .............................................................................................................. 174
LIST OF TABLES

TABLE 1. A SAMPLE TARGET REGION WITH 18 DISKS ............................................................ 4
TABLE 2. A VISIBLE POLYGON’S ALGORITHM ......................................................................... 31
TABLE 3. ROW BASED OBSERVERS PLACEMENT ALGORITHM .................................................. 35
TABLE 4. LCPP’S PERFORMANCE BASED ON SENSING RANGE ............................................. 40
TABLE 5. PREVIOUS-NEXT WAYPOINTS COVERAGE CONSTRAINT ALGORITHM 1 ................ 46
TABLE 6. PREVIOUS-NEXT WAYPOINTS COVERAGE CONSTRAINT ALGORITHM 2 ............. 47
TABLE 7. RBOP’S AND PNWCC’S PERFORMANCE ................................................................... 50
TABLE 8. STATIC COLLISION AVOIDANCE ALGORITHM .......................................................... 52
TABLE 9. ADAPTIVE CIRCLE ALGORITHM ............................................................................... 54
TABLE 10. DC ALGORITHM ...................................................................................................... 56
TABLE 11. FOUR PHASES OF THE TRIANGULATION-BASED COVERAGE PATH PLANNING APPROACH AND THEIR DESCRIPTIONS. ................................................................. 58
TABLE 12. AUXILIARY ALGORITHMS USED BY THE LCPP ALGORITHM ............................... 58
TABLE 13. THE FOUR PHASES OF LCPP APPROACH AND THEIR SUB ALGORITHMS ........... 59
TABLE 14. THE STATIC WORKSPACE CONFIGURATION OF A SAMPLE TARGET REGION 2 ....... 59
TABLE 15. THE FOUR PHASES OF LCPP APPROACH AND THEIR SUB ALGORITHMS .......... 60
TABLE 16. RUNNING TIME OF THE LOCAL COVERAGE PATH PLANNING APPROACH ............ 63
TABLE 17. LCPP’S PERFORMANCE BASED ON RS AND VORONOI PARTITION (RS IS 130.4) ........ 67
TABLE 18. GROUP COVERAGE PATH PLANNING (LCPP VS ILCPP), RS IS 130.4 .................... 68
TABLE 19. LCPP APPROACH IN THE SEQUENCE OF STEPS .................................................. 69
TABLE 20. INPUTS AND OUTPUTS OF THE LCPP APPROACH .................................................. 70
TABLE 21. IL CPP APPROACH’S FAILURE CONDITIONS ............................................................ 70
TABLE 45. RAINBOW ALGORITHM’S VERTICES, LSs, AND CSs .......................................................... 131
TABLE 46. INPUTS AND OUTPUTS OF THE GCPP’S APPROACH .................................................. 132
TABLE 47. GCPP APPROACH’S FAILURE CONDITIONS ................................................................... 133
TABLE 48. PSEUDO CODE TO COMPUTE EPs .................................................................................. 137
TABLE 49. RUNNING TIME TO COMPUTE THE VISIBLE POLYGON OF RT 7 ................................. 141
TABLE 50. A LOCAL COVERAGE PATH PLANNING APPROACH IN THE SEQUENCE OF STEPS ...... 144
TABLE 51. EXPECTED RUNNING TIME FOR LCPP APPROACH ...................................................... 144
TABLE 52. A DLL WORSE CASE RUNNING TIME .......................................................................... 145
TABLE 53. EXPECTED RUNNING TIME FOR PNWCC ALGORITHM .................................................. 147
TABLE 54. EXAMPLE LINESEGMENT CLASS ................................................................................. 148
TABLE 55. SOME NOTATION .......................................................................................................... 148
TABLE 56. A DLL WORSE CASE RUNNING TIME .......................................................................... 149
TABLE 57. RUNNING TIME OF THE RAINBOW ALGORITHM ......................................................... 150
TABLE 58. COVERAGE PATH PERFORMANCE AS A FUNCTION OF SENSOR’S RANGE ................. 174
# LIST OF COMMON ABBREVIATIONS

<table>
<thead>
<tr>
<th></th>
<th>ACD</th>
<th>Adaptive Circle Diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>ACC</td>
<td>Adaptive Curvature Control</td>
</tr>
<tr>
<td>3</td>
<td>BCS</td>
<td>Blind Curve Segment</td>
</tr>
<tr>
<td>4</td>
<td>BPP</td>
<td>Baseline Path Planning</td>
</tr>
<tr>
<td>5</td>
<td>BURL</td>
<td>Bottom, Upper, Right, and Left</td>
</tr>
<tr>
<td>6</td>
<td>CET</td>
<td>Common External Tangent</td>
</tr>
<tr>
<td>7</td>
<td>CIT</td>
<td>Common Internal Tangent</td>
</tr>
<tr>
<td>8</td>
<td>CP</td>
<td>Coverage Path</td>
</tr>
<tr>
<td>9</td>
<td>CP</td>
<td>Coverage Path</td>
</tr>
<tr>
<td>10</td>
<td>CPL</td>
<td>Coverage Path Length</td>
</tr>
<tr>
<td>11</td>
<td>CS</td>
<td>Curve Segment</td>
</tr>
<tr>
<td>12</td>
<td>CS(A,B)</td>
<td>Curve Segment Connecting Vertices A and B</td>
</tr>
<tr>
<td>13</td>
<td>CSIRT</td>
<td>Curve Segment Intersecting Regular Triangulation</td>
</tr>
<tr>
<td>14</td>
<td>CVCS</td>
<td>Candidate Visible Curve Segment</td>
</tr>
<tr>
<td>15</td>
<td>CW</td>
<td>Clockwise</td>
</tr>
<tr>
<td>16</td>
<td>RBOP</td>
<td>Circular Waypoint Coverage Placement</td>
</tr>
<tr>
<td>17</td>
<td>DC</td>
<td>Discontinuous to Continuous</td>
</tr>
<tr>
<td>18</td>
<td>DLL</td>
<td>Doubly Link List</td>
</tr>
<tr>
<td>19</td>
<td>DT</td>
<td>Delaunay Triangulation</td>
</tr>
<tr>
<td>20</td>
<td>DT</td>
<td>Delaunay Triangulation</td>
</tr>
<tr>
<td>21</td>
<td>EP</td>
<td>Extreme Point</td>
</tr>
<tr>
<td>22</td>
<td>HLT</td>
<td>Horizontal Line Test</td>
</tr>
<tr>
<td>23</td>
<td>HLT</td>
<td>Horizontal Line Test</td>
</tr>
<tr>
<td>24</td>
<td>HV</td>
<td>Hole Vertex</td>
</tr>
<tr>
<td>25</td>
<td>L(A,B,r)</td>
<td>LS connecting points A and B is a Red LS</td>
</tr>
<tr>
<td>26</td>
<td>LOP</td>
<td>Local Observer Planning</td>
</tr>
<tr>
<td>27</td>
<td>LosPoRT</td>
<td>Line of sight Partition of Regular Triangulation</td>
</tr>
<tr>
<td>28</td>
<td>LP(A,B)</td>
<td>Line Passing Through Endpoints A and B</td>
</tr>
<tr>
<td>29</td>
<td>LS</td>
<td>Line Segment</td>
</tr>
<tr>
<td>30</td>
<td>LS</td>
<td>Line Segment</td>
</tr>
<tr>
<td>31</td>
<td>LS(A,B)</td>
<td>Line Segment With Endpoints A and B</td>
</tr>
<tr>
<td>32</td>
<td>MER</td>
<td>Minimum Enveloping Rectangle</td>
</tr>
<tr>
<td>33</td>
<td>NN</td>
<td>Nearest Neighbor</td>
</tr>
<tr>
<td>34</td>
<td>OP</td>
<td>Observer Placement</td>
</tr>
<tr>
<td>35</td>
<td>OP</td>
<td>Observer Placement</td>
</tr>
<tr>
<td>36</td>
<td>OW</td>
<td>Observer Waypoint</td>
</tr>
<tr>
<td>37</td>
<td>PNWCC</td>
<td>PNWCC</td>
</tr>
<tr>
<td>38</td>
<td>PNWCC</td>
<td>PNWCC</td>
</tr>
<tr>
<td>39</td>
<td>POR</td>
<td>Path and Observer Replanning</td>
</tr>
<tr>
<td>40</td>
<td>RBOP</td>
<td>Row Based Observer Placement</td>
</tr>
</tbody>
</table>
41 RG Region
42 RT Regular Triangulation
43 RT Regular Triangulation
44 SCA Static Collision Avoidance
45 SMS Sensor Maximum Square
46 SMSD Sensor Maximum Square Dimension
47 SPR Smooth Path Replanning
48 SR Sub Region
49 LCPP Triangulation-Based Coverage Path Planning
50 TR Target Region
51 TR Target Region
52 TSP Traveling Salesman Problem
53 VC Visible Circle
54 VCPM Visible Circle Path Modification
55 VCS Visible Curve Segment
56 VD Visible Disk
57 VL Vertical Line
58 VLS Visibility Line Segment
59 VLS Visibility Line Segment
60 VLT Vertical Line Test
61 VLT Vertical Line Test
62 VP Visible Polygon
63 VP Visible Polygon
64 VPV Visible Polygon Vertex
65 WP Waypoint
66 WRT With Respect To
CHAPTER 1 INTRODUCTION

Coverage path planning is widely study in the literature [1]-[70]. However, coverage path planning with nonholonomic (movement constraint) mobile robot with optimization capability is difficult to find in the literature. Canny introduced a path planning method that uses a map of the environment. Our coverage path planning method is very similar to Canny’s method because both methods construct an initial map which required a one-time fixed cost and then using the map to generate paths required additional cost [29]. The ideal approach to solve this problem is through incremental approach which is a very popular approach to solve a complicate problem with multiple systems. The overall system in coverage path planning involves many complicate system parameters such as environmental, sensor, mobile platform, image processing capability, networking, path planning algorithm, and etc. Our contributions will be detailed in Chapter 3 and Chapter 4 in which are about novel coverage path planning algorithms based on ideal environmental conditions. Our assumptions about the system parameters are detail in the immediate Section, Section 1.1. Based on these assumptions, our contributions are on algorithmic approaches to coverage path planning. Currently, most mobile coverage path planning researches and algorithms development generate a zigzag coverage path within the target environment [1] and [2]. Zigzag coverage path is not desirable because it is not energy efficient and time efficient because the robot has to stop completely at the corner before it can turn and then accelerate in the new segment of the coverage path. Decelerating and accelerating consume a lot of energy and time. In addition, frequent turning due to coverage path discontinuity is not desirable for data collection. As a result, data collected by sensor(s) during turns are removed from analysis activity due to its inaccuracy.
Most coverage path planning or watchman route problems are studied on the following shapes [71]: Convex, star, spiral, monotone, trapezoidal, orthogonal polygon, and L shape. While the shape of the region of interest, the shape of mobile sensor’s field of view, and the shape of obstacles play a role in how system of coverage path planning is designed, the triangulation approach that we adopted in this dissertation provides many advantages to coverage path planning. This dissertation investigates coverage path planning control for nonholonomic vehicles with all circular sensing range in both static and dynamic environments. In both limited and unlimited sensing range, the sensor footprint is larger than the platform footprint, see Figure 2. If the sensor footprint and the platform footprint are the same size, then our algorithms still function properly with regard to coverage path planning design and it is fitting exactly as a vacuum. Our main focus will be on exact partition and complete coverage of the target region and with a first order differentiable coverage path. Triangulation approach is the simplest partition for coverage sensing as it is the most basic polygon. Due to this advantageous property, even a non-convexity of a regular triangulation introduce into a triangulation with a circular obstacle can be transformed into a convex visible polygon for coverage observer planning as well as flexible smooth coverage path planning and re-planning. Due to its simplicity, a triangulation methodology is widely used to prove that other polygons with more than 3 vertices are observable with a few and sufficient observers [72]. Before providing our contributions in Chapters 3, 4, and 5, we begin with some background and preliminary knowledge in the field of coverage path planning in Chapter 1 in which our works are built upon.

1.1 Background and Motivation

Given a target region defined by the blue line as shown in Figure 1 with a finite number of statically and moving disks, how do we design a differentiable and continuous path for one or
more mobile sensors with a fixed sensing range, coverage, movement (nonholonomic), and time constraints that can sweep the given region without collision while satisfying all four assumptions below?

**Assumption 1:** The robot being studied is a two-wheeled robot, enveloped by a two-dimensional obstacle, with the center at \( O(t) = (x, y) \) and of radius \( R_r \). Its motion obeys a nonholonomic constraint with velocity vector expressed as \( v_r(t) \). Note that the position and velocity are a function of time because the robot is continuously moving.

**Assumption 2:** The radius or range of robot’s motion sensor is \( R_s \). \( R_s \) is greater than \( R_r \). \( R_s \) is large enough to observe all vertices of the given regions.

**Assumption 3:** Both type of objects, static and dynamic, are denoted by the symbol \( O_i(t) \), where the subscript \( i = 1, \ldots, n \) represented the obstacle number. For example, an \( i^{th} \) object with radius \( R_i \) will be represented by obstacles centered at point \( O_i(t) \). For moving objects, the origin \( O_i(t) \) is time varying and moves with piecewise linear velocity.

**Assumption 4:** The set \( \Omega \) to be covered is two-dimensionally connected, with respect to a disk of the robot’s radius, \( R_r \).

![Figure 1. A sample target region (TR) with 18 disks](image)

Given an initial position and orientation of the robot represented by \( P_i \) and \( \theta_i \) and the environment under assumptions 1-4, we find a piecewise, continuous steering control under which the robot moves collision-free, and covers all points in the set \( \Omega \) overtime. This problem will be solved with limited and infinite sensing range shown in Figure 2.
Table 1. A SAMPLE TARGET REGION WITH 18 DISKS

<table>
<thead>
<tr>
<th></th>
<th>$O_{lx}$</th>
<th>$O_{ly}$</th>
<th>$R_i$</th>
<th>$O_{lx}$</th>
<th>$O_{ly}$</th>
<th>$R_i$</th>
<th>$O_{lx}$</th>
<th>$O_{ly}$</th>
<th>$R_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1656</td>
<td>2927</td>
<td>22</td>
<td>7</td>
<td>2198</td>
<td>3028</td>
<td>26</td>
<td>13</td>
<td>2566</td>
</tr>
<tr>
<td>2</td>
<td>1670</td>
<td>2579</td>
<td>40</td>
<td>8</td>
<td>2281</td>
<td>2896</td>
<td>21</td>
<td>14</td>
<td>2615</td>
</tr>
<tr>
<td>3</td>
<td>1830</td>
<td>3051</td>
<td>85</td>
<td>9</td>
<td>2289</td>
<td>2756</td>
<td>27</td>
<td>15</td>
<td>2664</td>
</tr>
<tr>
<td>4</td>
<td>2033</td>
<td>2497</td>
<td>21</td>
<td>10</td>
<td>2382</td>
<td>2997</td>
<td>7</td>
<td>16</td>
<td>2715</td>
</tr>
<tr>
<td>5</td>
<td>2050</td>
<td>2765</td>
<td>13</td>
<td>11</td>
<td>2485</td>
<td>2711</td>
<td>27</td>
<td>17</td>
<td>2772</td>
</tr>
<tr>
<td>6</td>
<td>2071</td>
<td>2966</td>
<td>35</td>
<td>12</td>
<td>2562</td>
<td>2818</td>
<td>24</td>
<td>18</td>
<td>2788</td>
</tr>
</tbody>
</table>

Figure 2. A robot and its sensing range (Rs)

Mathematically, the problem is to determine a differentiable path $s(t)$ by ensuring conditions represented by equations (1.1) and (1.2) hold:

$$\min_{t \in [t_0, t_0 + T]} \| q - s(t) \| \phi(q, t) \leq R_r, \forall q \in \Omega,$$  \hspace{1cm} (1.1)

where $\phi(q, t)$ is a weighting function which can be chosen at design time. The choice of $\phi(q, t)$ may produce $s(t)$ that passes through a given point $q$ once or several times and at certain desirable time. $q$ is the point in the set. $T$ is the time for the robot to complete it maneuver between adjacent pair of points. $\phi(q, t)$ is most obvious when we obtain some of the coverage paths of the Local Coverage Path Planning (LCPP) approach and the Rainbow coverage path
planning (RCCP) approach. Equation (1.1) is a standard mathematical expression for CP planning. Equation (1.2) ensures that at any point in time, the path that the robot is going to travel on is collision free.

\[ \|s(t) - O_i(t)\|_{t \in [t_0, t_0+T]} \geq R_r + R_i \forall i \in \{1, \ldots, n\}. \] (1.2)

The result of our dissertation will provide the necessary foundation to obtain the highest qualities performance parameters concurrently which is the shortest possible coverage path, minimal number of observers, minimal number of turns, and in the shortest running time.

1.2 Previous Work

This section highlights some of the early contributions of coverage path planning. A number of results in the literature solve the length of the coverage path, the number of observers and position, and the number of turns needed to achieve certain level of coverage. In any coverage path planning algorithm, complexity is usually measured with the length of distance traveled and memory required to store input information. Past works published in the literature yield very slow speed with coverage path planning and with undesirable coverage path. Optimal coverage path planning often obtained with NP-hard complexity. NP hard means "Non-deterministic Polynomial acceptable problems". Very few works have been done on obtaining a smooth coverage path. To the best of our knowledge, no result has been obtained with shortest possible coverage path, minimal number of observers, minimal number of turns, and in the shortest running time.

Kinematic and boundary constraints are also problems that have puzzled many researchers for decades. Efficient mobile coverage requires the path to be smooth or differentiable while avoiding obstacles. An important question to answer is how a coupled issue
such as path differentiability and a complete coverage is obtained? Combining trajectory planning with coverage path planning makes the task of finding a differentiable coverage path while ensuring complete coverage a very daunting task. In the span of one decade, references [1], [4], and [5] achieved some degrees of success with nonholonomic constraint coverage path planning. However, some fundamental issues need to be addressed. In the following paragraphs we are showing some coverage path planning approaches with some mathematical formulations.

The Art Gallery Theorem guarantees that a sensor with sufficient range can observe any three vertices polygon simply known as the triangle. The Art Gallery Theorem extends the result to basic polygon which requires no more than $\lfloor n/3 \rfloor$ observers. Any basic polygon is first partitioned or transformed into several triangles and then deduces the fact that each triangle can be observed by a single observer. The observer or guard in the Art Gallery Theorem refers to an observer that can see 360 degrees about it fixed position. Fisk and chvatal prove the theorem through triangulation technique. The Art Gallery Theorem is also extends to mobile observer. Fewer mobile observers are required to observe the same closed polygon observe by stationary observer. The number of mobile observers require for n vertices polygon are $\lfloor n/4 \rfloor$. The coverage path of the mobile observer is consists of multiple LSs which mean that they are discontinuous. In addition, the Art Gallery Theorem does not solve the problem dynamic and uncertain environment. Discontinuity and dynamic problems are solved by our novel approaches in Sections 3.6, 4.6, and existing approach in Chapter 6.

Reference [26] applied online algorithm, Spiral-STC, to cover an approximate area of the target region using a grid of cells each of size 2D where D is the size of the tool or sensor used to cover a cell. This is an example of an approximate decomposition technique. This algorithm requires access to a position and orientation sensor and a range sensor. It constructs a spanning
tree to follow a coverage path. It avoid repeat coverage with the used of sub-cells. Spiral-STC is a recursive algorithm which scans neighboring cells counterclockwise from the current cell’s parent. It repeats this until all neighbors are exhausted and then return. Spiral-STC algorithm generates smooth coverage path. Repeat coverage or overlapped coverage is also considered in our approaches. Overlapped coverage can be controlled depending on other parameters such as coverage path length, curvature control, observer placement, and obstacle avoidance.

Scan STC method does not take into account dynamic environment. Scan STC method allows skipping of coverage of horizontal cells if vertical coverage behavior is desired. Full scan STC provides the same advantages and disadvantages to the Scan-STC algorithm that the full spiral STC provides to the preliminary spiral STC. Coverage occurs using linear time and linear memory which are corresponding to the number of accessible free sub cells. The number of overlapped coverage can be computed by the following equation:

$$\text{pathlength} \leq (n + m)D$$

(1.3)

D is the tool size. The number of repetitive coverage is at most m. This reference does not solve the problem for dynamic and uncertain environment.

Reference [87] provides a unifying Geometric, probabilistic, and potential field approaches to multi-robot deployment. Several different strategies are integrated for deploying groups of robots in an environment. A specialized cost function representing different multi robot deployment tasks are discussed. Distributed controllers are obtained from the gradient of the cost function and are proved to converge to a local minimum of the cost function. Three special cases are derived as examples: a Voronoi based coverage control task, a probabilistic
minimum variance task, and a task using artificial potential fields. The combine cost function is formulated to be:

\[ H(P) = \int_\mathcal{Q} g(f(p_1, q), \ldots, f(p_n, q))\phi(q) dq, \quad (1.4) \]

where \( \phi: \mathbb{R}^d_+ \mapsto \mathbb{R}_{>0} \) is a weighting of importance over the region \( \mathcal{Q} \). The notation \( \mathbb{R}_{>0} \) represents the set of positive real numbers. The cost of the group of robots sensing at a single arbitrary point \( q \) is represented by the integrand \( g(f(p_1, q), \ldots, f(p_n, q)) \). Integrating over all points in \( \mathcal{Q} \), weighted by their importance \( \phi(q) \) gives the total cost of a configuration of the robots. The equation above can be further extends to include the mixing function. The mixing function determines how information from the sensors of multiple robots is to be combined. This reference does not solve the problem for dynamic and uncertain environment.

Reference [4] provides a coverage solution to a convex region with minimum number of sensors and then connects the center point of the sensor with the closest neighbor points in a recursive algorithm until all center points are connected. It does not take into account overlapped region for coverage path length improvement, coverage path continuity, and smooth turning. Given any convex region, the proposed solution in this reference find a minimum enclosing rectangle and then the algorithm computes the position of minimum number of circles required to observe the rectangle with the formulation shown below:

\[
[x_c^{kl}, y_c^{kl}] = \begin{cases} 
[0.5 + (l - 1)\frac{3}{2} R_c, (k - 1)\sqrt{3}R_c], & \text{if } l \text{ is an odd integer}; \\
[0.5 + (l - 1)\frac{3}{2} R_c, \frac{\sqrt{3}}{2} R_c + (k - 1)\sqrt{3}R_c], & \text{if } l \text{ is an even integer} 
\end{cases}
\]  

(1.5)

The number of circles needed in each column and row, \( n \) and \( m \), was formulated to be
\[ n = \begin{cases} 
\text{Int} \left( \frac{y_w}{\sqrt{3}r} \right) + 1, & \text{Rem} \left( \frac{y_w}{\sqrt{3}r} \right) \leq \frac{1}{2} \\
\text{Int} \left( \frac{y_w}{\sqrt{3}r} \right) + 2, & \text{Rem} \left( \frac{y_w}{\sqrt{3}r} \right) > \frac{1}{2} 
\end{cases} \tag{1.6} \]

\[ m = \begin{cases} 
\text{Int} \left( \frac{x_w}{1.5r} \right) + 1, & \text{Rem} \left( \frac{x_w}{1.5r} \right) \leq \frac{2}{3} \\
\text{Int} \left( \frac{x_w}{1.5r} \right) + 2, & \text{Rem} \left( \frac{x_w}{1.5r} \right) > \frac{2}{3} 
\end{cases} \tag{1.7} \]

The int in equations (1.6) and (1.7) is the integer operation and int(x) equals the integer part of x, Rem(x) = x − Int(x). \( x_w \) is the length of the rectangular edge along the x-axis, \( y_w \) is the length of the rectangular edge along the y-axis. Once the points to be traversed are computed, the reference implemented feasible trajectory published in reference [81] so that the robot moves collision-free and covers all the points in a dynamic and uncertain environment. It is clear that our new contributions are better than the contribution of this reference because our approach provide bias toward a number of desirable features such as smooth curvature, shorter path length, and flexible observer placement.

Reference [100] proposed a greedy iterative algorithm implementing with level set framework. It took several thousand iterations to find the optimal path. This algorithm achieved complete visibility coverage of the target region while attaining a local minimum of path length.

The Watchman Problem is an optimization problem with the objective to compute the shortest route a watchman should take to guard an entire area with obstacles given only a map of the area. The watchman peeks behind every corner to determine the best order in which corners should be visited. The problem may be solved in polynomial time when the area to be guarded is a simple polygon. This reference does not solve the problem for dynamic and uncertain environment.
1.3 Statement of Contributions

Contributions of this dissertation can be summarized into three distinct items: The main technical contributions of both approaches, a LCPP approach and a RCPP approach, are to provide a holistic solution that segments any target region, uses triangulation to determine the line of sights and observation points, and then compute the smooth and collision-free coverage path. The novelties of our contributions are that our approaches partition the target region into different shapes with different properties need to obtain complete coverage while achieving first order path differentiability. In addition, our proposed methods can be readily generalized to address problems of higher dimensions, and it is scalable with respect to the size of the target region and the number of robots. Our novel approaches can obtain the desirable coverage path and observers very quickly and with great flexibility.

Our algorithms compute the solutions with quadratic run-time. Fundamentally, the algorithms are fast and the CP obtained is nearly optimal while providing overlapped control, flexibility with observer placement, and localization. In addition, the coverage path is smooth which is very practical for the nonholonomic system which will be presented in Chapter 6. In Chapter 2 we provide an overview of existing coverage control techniques which include heuristic, approximate, partial-approximate, and exact decomposition. Definition of each coverage control technique is defined based on reference [29]. All of our contributions as will be seen in Chapters 3 and 4 are exact decomposition coverage control based on triangular technique.

In Chapter 3 we introduce the implementation of the exact decomposition technique via Delaunay triangulation. Regular Triangulation or RT is a simple cell of the TR partitioned by the Delaunay triangulation technique. When the sensing range is large relative to the RT, the
visible polygon concept guarantees complete coverage of the RT with a single observer. The
discovery of the visible polygon of the RT results in our first paper [69]. Our first paper
addresses coverage control locally, in each and every RT, for all sensing range. Since each RT
in the TR may be very different from each other, several algorithms are developed. When the
RT is large relative to the sensing range, our Row Based Observer Placement (RBOP) algorithm
will find the necessary number of observers and their placement location for complete coverage.
The running time for all new algorithms are analyzed and proved to be quadratic as will be seen
in Chapter 5. Conclusion on the conditions needed for the algorithms and their requirements for
successful run will be presented in Chapter 5.

Traveling Salesman’s algorithm, nearest neighbor algorithm, is used to link all observer
points found by the LCPP approach. The zigzag CP constructed by the NN algorithm is
smoothen by either the Cubic Spline algorithm or the newly developed adaptive curvature
control algorithm. The last step of the coverage path planning technique is to steer the mobile
robot from a given arbitrary position and orientation to the desired position and orientation so
that it is collision free with dynamic obstacle. This technique is implemented by the existence
algorithm developed by Qu et all [82].

In addition, Chapter 3 also covers the contribution of our second paper which improves
the result of the first paper through local improvement technique [70]. Section 3.4 describes the
Previous Next Waypoint Coverage Constraint (PNWCC) algorithm which reduces the number of
observers by 30% or more as well as reduces the coverage path length by more than 20%. The
basic idea with the PNWCC algorithm is by modifying the position of the current observer’s
position with respect to the previous and the next observers in the coverage path if the coverage
path length can be reduced. The current observer may be deleted if it is not needed. Similar to
the first paper, the second paper requires multiple steps to find observe points and the coverage path that can observe the entire target region. Both, the first paper and the second paper cover all sensing ranges, however, they are both Local Coverage Path Planning approaches.

In Chapter 4 we will be presenting the second coverage path planning approach which is a global approach, the Rainbow Coverage Path Planning approach. For brevity, we will be referring to the Rainbow Coverage Path Planning approach as Rainbow algorithm or RCPP approach. Just like the LCPP approach, the RCPP approach requires several steps to obtain the final desirable coverage path and observer for complete coverage of the target region. The RCPP approach consists of a family of algorithms which begin by finding the first path around the center of interior obstacles or disks of the target region through the Graham Scan algorithm. The first path is then used as a foundational path to generate a collision free, first order differentiable, and observable coverage path through a series of input and output transformation algorithms which together form the Rainbow algorithm. The last algorithm of the RCPP approach finds a sufficient number of observers on the coverage path needed to observe the target region.

The two LCPP approaches and the RCPP approach are compared in Chapter 7. With sufficient sensing range in the same target region, the RBOP algorithm requires 26 observers and has a length of 3216m. The PNWCC algorithm requires 12 observers and has a length of 2390m. The Rainbow algorithm requires 20 observers and has a length of 2001m. All three algorithms can be further improved to obtain the optimal CP length, the fewest number of observers, and to compute the solution in the shortest possible time based on their limitation. It is beyond the scope of this dissertation to get the optimal solution. It is a step in the right direction for further contributions.
CHAPTER 2 REVIEW OF COVERAGE PATH PLANNING CONTROL

Many coverage path planning algorithms are published in the literatures [1]-[70]. Applications of coverage path planning can be found in many areas including farming, cleaning, surveying, security patrolling. Below are some of the approaches used in coverage and coverage path planning and their implementation really influence complexity of the problem.

2.1 Introduction

Coverage path planning for mobile robot has recently become an increasingly popular research topic [1]-[68]. Typical issues that coverage path researchers want to solve are how to reduce high computation power usage that arises from massive information receive from sensory devices, time to completion, and number of turn for the robot to cover the interested region. Also important issues are how to take advantage of a priori information, how to perform online coverage efficiently, and how to balance work among multiple robots while avoiding collision among the robots themselves and the robot with the static disks in the given target environment.

The type of coverage such as approximate coverage or complete coverage is an important factor in coverage path planning research. Reference [29] defined four different types of coverage: heuristic decomposition, approximate decomposition, partial approximate decomposition, and exact decomposition. Any type of cellular decomposition divides the interest region into “simple” cells. In any coverage algorithm, complexity is usually measured with two different parameters. They are distance traveled by the robot and memory required to store input information.
2.2 Heuristic Decomposition

A heuristic decomposition technique is a simple rule of thumb that may work to satisfaction, but do not guarantees complete coverage. This approach consists of a simple set of behaviors such as following a line or randomly moving and avoiding any disk encounter. There are pro and con of this approach. However, in demining application in a large body of water, a heuristic method is analogous to a faulty mine detector [29]. Advantage of heuristic approach include random search which does not need costly localization sensors and due to its low computational resources for calculating the robot’s position. In some simulated scenario for robots build without localization capabilities which can be built at one fifth the cost, it may be useful to performed heuristic search verse methodical search strategies. Since our coverage path planning approaches guaranteed complete coverage, we do not implement heuristic decomposition of the interested region. The heuristic decomposition approach is introduced here since it is one of the four types of coverage.

2.3 Approximate Decomposition

An approximate decomposition technique is a grid base representation of the free space where the cells are the same size and shape, but the union of the cells only approximates the interested region [29]. This approach normally assumes that once the robot enters a cell it has covered or observed the cell. Coverage is complete when the robot visits each and every cell in the decomposition. Two examples of approximate cellular decompositions are conventional wave-front or distance transform algorithm and Spanning Tree Covering (STC). This technique is beneficial for mobile robots because the planner can be implemented to reduce the number of turns. An approximate decomposition approach has applications in vacuum cleaning, automate target recognition, mine hunting application, and etc. [29] and [104].
2.4 Partial-approximate Decomposition

A partial-approximate coverage algorithm relies on a partial discretization of space where cells are fixed in width but the ceiling and floor can have any shape [29]. Cells may be of the same shape and size for this approach. This approach has application in terrain-covering problem. This approach is advantageous in that it provide complete coverage without assuming any a priori information about the robot’s free space. In addition, a linear size memory is sufficient to implement the algorithm if the boundary of the interested region can be described by a semi-algebraic set [29]. Figure 4 illustrates partial approximate decomposition.
2.5 Exact Decomposition

An exact cellular decomposition technique results in the set of non-intersecting regions and their union is the interested region [29]. Examples of exact cellular decomposition include trapezoidal decomposition, boustrophedon decomposition, triangular decomposition, and rectangular decomposition. In a trapezoidal cells approach, coverage in each cell is achieved with simple back and forth motions. Coverage of the environment is achieved by visiting each cell in the adjacency graph. For large sensing range, existing exact decomposition approach is disadvantageous because complete coverage required the robot to visit each and every cell. Our approach in this dissertation is also an exact cellular decomposition, but with a triangular decomposition approach. A triangular decomposition is preferred because of its simplicity and because theories and algorithms are available to solve it such as the Art Gallery Theorem and Delaunay triangulation. With our approach, in the case of large sensing range, we are able to relax the requirement of full coverage without having the robot to visit each and every cell. Detail of this relaxation will be seen in Section 4.7. This section ends the introduction on cell decomposition.
Figure 5. An exact partition of the sample target region into 24 RTs

Figure 5 is a sample target region that we will be using throughout the dissertation as an example due to the variety of different RTs with different properties. In Chapter 3, we will study how effective local planning is in the static target region. In Chapter 4, we will study how effective global planning is in the static target region. We will see how the complete coverage constraint affects our actual result in the static environment whether it is a LCPP approach or a RCPP approach. In Chapter 5, we will study how fast the run time is for our methods, local and global path planning. Finally, Chapter 6 we will show how to extend our static coverage path planning in the dynamic environment and why our results naturally fit with chained form control of nonholonomic system.
2.6 Triangulation

Delaunay triangulation allows unique and consistent partition of the interested region into simple cells [71]. Simple cell allows us to compute visible polygon (VP) where a decision can be made to find the number of observer necessary to cover each cell based on the sensing range and the shape of the RT. Popular triangulation techniques that run in $O(N \log N)$ time [71] include Fortune sweep line algorithm, divide and conquer algorithm, and incremental algorithm. We refer to the simple cells that result from the Delaunay triangulation as RT throughout this dissertation. The different between RT and triangulation is that RT may contain weight site, point, or vertex whereas triangle only contains points as vertices. A disk is a perfect example of weighted point where the point is the center of the disk.

One of the important bookkeeping tasks before and after computing triangulation is how to order the disks and the RT within the target region. The disk’s order is very straightforward; they are ordered in ascending order based on the x-value of their center. Ordering the RT of three neighboring disks is a little tricky since each RT contains three disks and any disk can be a member of several RTs. To overcome this problem, an RT is ordered based on the sum of their disks’ number. In case of a tie, the RT with the smallest global disk identity has the priority. For example, RT1 of the sample TR in Figure 5 is $1+2+5 = 8$, RT2 is $1+3+5=9$, RT3 is $2+4+5=11$, RT22 is $12+16+18=46$, and RT23 is $14+15+17=46$. Obviously RT22 and RT23 have the same values with respect to the accumulation of their three disks, sum of 46, which result in a tie. RT 22 is the corresponding RT for the three disks with the smallest disk identity being 12 where RT 23 corresponding to the RT for the three disks with the smallest disk identity being 14.
Figure 5 presented the triangulation of Figure 1 with RT being ordered as mentioned in the above paragraph. The sum of all disks in RT number 1, 2, and 3 are 8, 9, and 11 respectively. There are 24 RTs in our sample target region as seen in Figure 6.

![Figure 6. Regular Triangulations (RTs) of the sample TR](image)

In addition to illustrating the RT, Figure 6 also illustrates some important properties of the triangulation. For example, RT number for RT 1 is labeled in blue and the global disk number is labeled in black for exterior disk and red for interior disk. The local disk number for each disk inside a RT is labeled in green. In future figures, we will not label the RT number and disk number in different color.

2.7 **Traveling Salesman Algorithm**

Any Traveling Salesman Problem (TSP) algorithm would find a tour through all the waypoint and return to the starting waypoint. A tour of $\Omega$ with bounded length, where $\Omega$ is a set
of m waypoints, can be computed with the following equation [72]:

\[
s(x) = d(p_{\pi(m)}, p_{\pi(1)}) + \sum_{i=1}^{m-1} d(p_{\pi(i)}, p_{\pi(i+1)})
\]

The minimum theoretical bound of any tour is computed by summing all row or column minima. The tour that was obtained after applying the TSP algorithm contained sharp edges which are not realistic for the nonholonomic robots. The Novel PNWCC algorithm in Section 3.4 can improve the sharp edge problem as well as reducing the local and global tour’s length and the undesirable curvature which limit the speed of the nonholonomic robot.

### 2.8 Discontinuous Coverage Path Planning

In 2004, Qu et all [82] contributed an analytical and real-time technique to maneuver a nonholonomic robot from an arbitrary initial position to an arbitrary final position. However, most coverage path planning works published in the literature recently failed to take advantage of Qu’s result. In 2009 reference [9] proposed Virtual Door algorithm along with the Task Area Allocation Algorithm which is a template-based coverage path planning approach. The Virtual Door algorithm divides a given region into several small regions. One divided area or region is allocated to one robot. Each divided area is rectangular. Task Area Allocation Algorithm eventually leads to the output coverage path that is zigzag and discontinuous. In 2011, reference [5] attempted to smooth the path with a high resolution grid map representation and coarse to fine constrained inverse distance transform techniques, however discontinuity in the CP still exists. In 2011, reference [8] proposed a state machine model which includes a number of behaviors such as right wall following navigation, left wall following navigation, obstacle avoidance navigation, zigzag path navigation, spiral path navigation, random path navigation,
and goal seeking navigation. It is obvious with the names of some of these behaviors that the output coverage path obtained by the model is zigzag and discontinuous. In 2014, reference [1] proposed a generalized Voronoi diagram-based CP planning, however, the resulting CP is also discontinuous.

In the above paragraph we see many studies which obtained limited results, even in the local coverage path planning scenario. For this reason, an analytical approach with chained form cannot be utilized for an autonomous coverage path planning for differential robots. As will be seen in Chapter 6, zigzag coverage path cannot satisfy the boundary conditions dictates by chained form’s requirements which can be referred to as the singularity condition.

The singularity condition in the chained form for differential robot can be avoided with multiple-phase coverage path planning as will be seen in Chapters 3 and 4. The Local Coverage Path Planning approach may use existing algorithm such as Cubic Spline to remove discontinuity in the planned linear spline coverage path. Our Local Coverage Path Planning approach employs an adaptive curvature control algorithm to remove discontinuity. Our Global Coverage Path Planning approach employs a BluePath algorithm to remove discontinuity. In either case, our result fit very well with an analytical approach to real-time vehicle maneuver contributed by [82].

2.9 Cubic Spline Interpolation

From the set of piecewise continuous functions obtained in the previous section, a single curve interpolating all points \( p_l \) for \( l = 1, 2, ..., n \) can be obtained by applying a three degree Cubic spline method so that all waypoints on the curve is continuous and twice differentiable. The cubic spline turns the otherwise discontinuous tour into the continuous tour so that a
nonholonomic system can follow. A general form of a cubic spline with its first and second derivatives can be expressed as follows [73]:

\[ p_k(x) = a_k + b_k x_k + c_k x_k^2 + d_k x_k^3, \]  

(2.2)

\[ \dot{p}_k(x) = b_k + 2c_k x_k + 3d_k x_k^2, \]  

(2.3)

\[ \ddot{p}_k(x) = 2c_k + 6d_k x_k, \]  

(2.4)

\[ \dddot{p}_k(x) = w_{l-1} + \frac{w_l - w_{l-1}}{x_l - x_{l-1}} (x_l - x_{l-1}), \]  

(2.5)

\[ \dot{p}_k(x) = \int \dddot{p}_k(x) dx, \]  

(2.6)

\[ p_l(x) = A_l (x - x_{l-1}) + B_l (x_l - x) + C_l (x - x_{l-1})^3 + D_l (x_l - x)^3, \]  

(2.7)

\[ A_l = \frac{y_l}{x_l - x_{l-1}} - \frac{w_l}{6(x_l - x_{l-1})^3} \]  

(2.8)

\[ B_l = \frac{y_{l-1}}{x_l - x_{l-1}} - \frac{w_{l-1}}{6(x_l - x_{l-1})}, \]  

(2.9)

\[ C_l = \frac{w_l}{6(x_l - x_{l-1})}, \]  

(2.10)

\[ D_l = \frac{w_{l-1}}{6(x_l - x_{l-1})}, \]  

(2.11)

Where \( a_l, b_l, c_l, d_l, w_l, A_l, B_l, C_l, \) and \( D_l \) are constants for \( l = 1, 2, ..., n \). From equation (2.2), a cubic spline has the following properties that make coverage control for nonholonomic system possible when interpolating linear spline tour into continuous and differentiable tour. Equivalent relation between equations (2.2) and (2.7) yields the coefficients in equations (2.8)-(2.11).
Property 1. The spline is continuous in \([x_0, x_n]\).

Property 2. The points at the end of each interior waypoint must be equal, \(p_l(x_{l+1}) = p_{l+1}(x_{l+1})\) for \(l = 1, 2, ..., n - 1\).

Property 3. The first derivative at the end of each interior point is continuous or \(\dot{p}_l(x_{l+1}) = \dot{p}_{l+1}(x_{l+1})\) for \(l = 1, 2, ..., n - 1\).

Property 4. The second derivative is continuous at each interior point or \(\ddot{p}_l(x_{l+1}) = \ddot{p}_{l+1}(x_{l+1})\) for \(l = 1, 2, ..., n - 1\).

Property 5. The end points are free boundaries which means that the second derivative at the end points are zero.

2.10 Conclusions

We introduce several decomposition techniques in this Chapter. A heuristic decomposition technique is a simple rule of thumb that may work to satisfaction, but does not guarantee complete coverage. An approximate decomposition technique is a grid base representation of the free space where the cells are the same size and shape but the union of the cells only approximates the interested region. A partial-approximate decomposition technique relies on a partial discretization of space where cells are fixed in width but the ceiling and floor can have any shape. An exact decomposition technique results in the set of non-intersecting regions and their union is the interested region. In local coverage path planning with limited sensing range, observers must be determined for each and every “simple” cell and then all observers are connected with a linking algorithm such as the Traveling Salesman’s Nearest
Neighbor algorithm to form a tour. Finally a smoothing algorithm such as the Cubic Spline algorithm can be used to smoothen the coverage path or the tour to get a differentiable tour which is desired for a nonholonomic robot.
CHAPTER 3 A LOCAL COVERAGE PATH PLANNING APPROACH

3.1 Introduction

The previous chapter presented several approaches to coverage decomposition. In this Section we will focus only on our new approaches in coverage path planning for nonholonomic vehicles, which employ an exact partitioning technique of triangular decomposition. Some preliminary materials were introduced in Chapter 2. Section 3.2 introduces the Visible Polygon (VP) concept which allows the rest of the local coverage path planning algorithms to make determination on where to place observers and therefore the coverage path for complete coverage path planning of the entire target region. Once the VP of an RT is determined, the centroid of the VP can then be computed. Depending on the size of Rs, different algorithms in this Chapter are used to find observer points which will be used as waypoints to find a tour with the TSP’s NN algorithm. Initially, the tour being generated which we refer to as the baseline coverage path is a linear spline or piecewise continuous tour which may be involved in collision with the static obstacle in the target region. This baseline coverage path may be refine or enhance with a number of algorithms presented in this chapter to find a desired smooth coverage path that the robot(s) can follow to observe the entire target region that is collision free.

We summarize our contributions of the local observer placement in VP algorithm, RBOP theorem, RBOP algorithm, and PNWCC algorithm. Section 3.7 organizes the novel algorithms in this Chapter into a Local Coverage Path Planning approach. Finally, the performance of coverage path planning for a group of robots with RBOP algorithm and PNWCC algorithm is presented in Section 3.8 follows by conclusion in Section 3.9.
3.2 Visible Polygon of a Regular Triangulation (RT)

A half plane is exploited to study the visibility line segments and it direction of coverage is toward the other end point of the LS that it is perpendicular to within the RT. We will show how this directionality is determined in Chapter 5. For example, the corresponding half plane of LS\(s_{58}\), the LS that begins at the center of Disk 5 and ends at the center of Disk 8, is \(h_{5827}\) (see Figure 8(b)). The area of coverage is determined by an inequality \(h_{5827} \geq l_{5827}\) or \(h_{5827} \leq l_{5827}\) which will also be discussed in Chapter 5 when we discuss the running time in computing the visible polygon. \(l_{5827}\) represents a line due to edge \(s_{58}\). Generally, the area of coverage is expected to be the region between the two EPs of each of the edge. The LS \(l_{5827}\) represents the visibility line of sight, a LS, that is perpendicular to the LS\(s_{58}\) and is crossing the extreme point (EP) at \(e_2\), which can be seen in Figure 8. RT number is the last number, number 7, in \(l_{5827}\).

![Image of RT with elementary notation](image)

(a) An RT of disks 5, 6, and 8. Each disk has two EPs. The EPs are sorted by their x-coordinate relative to each other on the boundary of the disk. We do not prefer this notation.

(b) An RT of disks 5, 6, and 8. Each disk has two EPs. The EPs are sorted by their x-coordinate relative to each other in the RT. This is the preferred notation.

Figure 7. An RT with elementary notation
Figure 7 illustrates the notation of any RT. Note that there are a number of ways to label each LS, each EP of a disk, and each half plane. The EPs are the points where each Disk intersects with the edge that connects the origin of any two Disks within the triangulation of the three Disks. For example, $e_1, e_2, e_3, e_4, e_5,$ and $e_6$ in Figure 7 are EPs. However, for consistency and simplicity, the notation as in Figure 7 is adopted throughout this dissertation. The green line in Figure 7(b), $l_{5827}$, can be read as the visibility line that is perpendicular to $LS_{58}$ at $e_2$ and in RT 7. In the next few illustrations, we will introduce more visibility LSs. An addition of $l_{5617}$ slightly affect the overall visible half plane of disk 5 with respect to RT 7 as seen in Figure 8 (c). The transition from Figure 9 (b) to Figure 9 (c) is very small, slightly change. This is not always the case as seen in Figure 9 with disk 6 because there is significant/observable change from Figure 9 (b) to Figure 9 (c).

(a) $LS_{5827}$ relative to $LS_{58}$. The green arrow denotes the direction of the half plane of $l_{5827}$.

(b) Half plane $h_{5827}$ relative to the RT of disks 5, 6, and 8, and $LS_{5827}$.

(c) Half plane of disk 5 within the RT of disks 5, 6, and 8. A half plane of any disk in any RT is the intersection of two half planes.

Figure 8. A progression of Line Segments (LSs) to determine a VP
Figure 9. A progression of LSs to determine visible polygon

Basically, the notation is $l_{ijkp}$ where $i$ and $j$ are global disk identifiers, $k$ is the EP identifier within an RT which is within the range of 1 to 6, and $p$ is the RT number. Figures 8-9 treat the visibility half plane of a single disk. However, the process of finding the visible polygon requires the treatment of all three disks within the RT. Figure 11 (a) shows all visible LSs of all three disks.

Figure 10. A progression of LSs to half planes of disk 8

(a) LS $l_{6847}$ relative to LS$s_{68}$ and RT of disks 5, 6, and 8.
(b) LS $l_{5617}$ relative to LS$s_{56}$ and RT of disks 5, 6, and 8.
(c) Half plane of disk 6 within the RT of disks 5, 6, and 8.

(a) LS$l_{5827}$ relative to LS$s_{58}$ and RT of disks 5, 6, and 8.
(b) LS$l_{6817}$ relative to LS$s_{68}$ and RT of disks 5, 6, and 8.
(c) Half plane of disk 8 within the RT of disks 5, 6, and 8.
We know that if a point observer of sufficient range is placed anywhere inside the green polygon as shown in Figure 11(b), then the whole RT is observed. However, the process of keeping the green polygon shown in Figure 11(b) requires complicate constraint checking since the green polygon also covers area outside of the RT. For some target region, allowing the VP of an RT to be outside of the RT may yield un-tractable computation. It is beyond the scope of this dissertation to obtain a VP that is outside its RT due to computation tractability. For this reason, we limit the visible polygon to be bounded within the RT.

In Figure 12 (c), the green area within the RT is a VP, bounded to be within the RT, the white areas within the RT are the blind-spot polygons, and the red disks are the static disks. A sensor, of any range, cannot cover the entire RT if it is placed inside the blind-spot polygon. Mathematically, the EPs and the visible polygon may be found by solving the constraint equations (3.1) and (3.2).

\[(x - x_i)^2 + (y - y_i)^2 = R_i^2 \quad (3.1)\]

\[y - y_i = m(x - x_i) + b \quad (3.2)\]
The points $x_i$ and $y_i$ are the center of disk $i$. $R_i$ is the radius of disk $i$.

From each EP on the statically disks, as shown in Figure 12(b), draw a line from the EP that is perpendicular to the edge that contains the EP and connect it with the other edge of the same static disk. The polygonal area, bounded by the lines above and the three edges of the RT, is a visible polygon. The difference of the RT and the visible polygon(s) is the blind-spot polygon(s). The visible polygon of a RT is convex and can be computed in linear time. Running time analysis will be shown in Chapter 5. The convex region form by the visibility half plane can be described by equation (3.3):

$$vp_p = (\bigcap \sum_{e=1}^{6} h_{ijep}) \cap (\Delta_{I_{JQ}}). \quad (3.3)$$

$\Delta_{I_{JQ}}$ is the triangle formed by points I, J, and Q which are the center of disks i, j, and q of the same RT respectively. Now that we have seen how a VP is computed as a function of the RT, EPs, and half planes, we are ready to present a simpler version of visible polygon derivation which is the model used to implement the VP algorithm.

### 3.2.1 Visible Polygon

If it exists, a visible polygon, inside a triangulation of three Disks, has at most nine vertices [69]. Visible Polygon is a subset of the RT to be covered. The Visible Polygon is found by the VP algorithm shown in Table 2.
TABLE 2. AVISIBLE POLYGON’S ALGORITHM

| Input:  | An RT of disks $i,j$, and $k$ (e.g. RT 7 in Figure 12). |
| Output: | The convex visible polygon. |
| Algorithm Steps Description: | |
| 1. Compute all six EPs of the RT. |
| 2. Reduce all disks of the RT into points to get a triangle. |
| 3. Initialize the VP to the triangle in step 2. |
| 4. For each of the six EPs found in step 1, compute the corresponding VLS. Compute the intersection of the VLS with the VP. Remove the region to the left, right, below, or above the two EPs on the LS that spawn the VLS. Update the VP. |

Steps 2 and 3 of the VP algorithm initialize the VP to a triangle formed by the centers of all three disks within the RT. It is dictated by equation (3.3). Step 4 computes the intersection of the RT’s triangle with all half planes to find the resulting VP. Illustration for step 1 of the algorithm is shown in Figure 12(b). Illustrations for steps 2 and 3 are shown in Figure 12(c) and (f).

![Figure 12](31)

Figure 12. Transition from RT to a visible polygon (VP).
Illustrations for step 4 are shown in Figure 12(g)-(l). Figure 12(d)-(l) include the bounding rectangle enclosing the triangle. It is shown to show that half planes due to EPs can be finite.

While Figure 12(g) and (h) show VLS due to EP A as the first operation to find the VP, it is not necessary to begin with EP A or any sorted order EP. What is important is the consistency in VP computation which eventually lead to a unique result whether we started with EP C or F or $e_2$.

Once all EPs are considered, the unique VP of the RT in Figure 12(a) is shown in Figure 12(l), the green VP.

### 3.3 Row-Based Observer Placement Algorithm

If the sensing range is sufficiently large, a local coverage path planning algorithm can just place an observer anywhere within the VP of the RT and then be able to observe that RT. What happen if the sensing range is not large enough? What is the governing equation that guaranteed that the sensing range can observe an RT? What make the VP useful is it convexity property. The VP compute by the VP algorithm is always convex. Due to its convexity, equations (3.4)-(3.6) can be used to find the area of the VP as well as it centroid. The number of vertices is $n$ and $i$ is the vertex number. Equation (3.7) represents the Euclidean length between any two points.

**Figure 13.** A typical RT, its VP, its VP’s centroid, and length.

\[
A_{VP} = \frac{1}{2} \sum_{i=0}^{n-1} (x_i y_{i+1} - x_{i+1} y_i)
\]  

(3.4)
\[ X_C = \frac{1}{6A_{VP}} \sum_{i=0}^{n-1} (x_i + x_{i+1})(x_iy_{i+1} - x_{i+1}y_i) \]  \hspace{1cm} (3.5)  \\

\[ Y_C = \frac{1}{6A_{VP}} \sum_{i=0}^{n-1} (y_i + y_{i+1})(x_iy_{i+1} - x_{i+1}y_i) \]  \hspace{1cm} (3.6)  \\

\[ L(P_1, P_2) = \sqrt{(P_{1x} - P_{2x})^2 + (P_{1y} - P_{2y})^2} \]  \hspace{1cm} (3.7)  \\

\[ R_s \geq \max_{e_i \in E} L(e_i, G) \]  \hspace{1cm} (3.8)  \\

\[ R_s \geq \max_{e_i \in E, v_j \in V} L(e_i, v_j) \]  \hspace{1cm} (3.9)  

If condition expressed in equation (3.8) is met, then there exists at least one observer point, at the centroid of the VP, that can observe an entire RT. If condition expressed in equation (3.9) is met, then there exist infinitely many possible locations of an observer within the VP that can observe the entire RT. What happen if the sensing range, \( R_s \), is less than that expressed in equation (3.8)?

A Row-Based Observer Placement (RBOP) algorithm can find a sufficient number of observer or waypoint needed to cover each RT based on the sensing range, \( R_s \) [69] and [70]. Before formulating the algorithm, let us see a few examples. Multiple observers in an RT may be required for complete coverage when condition (3.8) or (3.9) is not available.

In this section, sufficient conditions to observe an RT is formulated, derived, and proved. RBOP algorithm begins by computing the necessary quantities to find the sufficient number of observers to observe the RT. The first example shows that three observer waypoints can cover the whole RT by placing each observer waypoint at each pair of the EPs of any disk. See Figure 14. Since we assume a circular observer, equation (3.1) yields two possible solutions with the constraint of two EPs. Ideally, we need the solution that place the observer inside of the RT. With many possible combinations of RT and sensing range, how can we design an algorithm that
guaranteed that all observers will be inside of the RT? As important is how many observers are sufficient? Figure 15(a) suggests that the solution is very simple when the sensing range is large enough so that only one observer is required per RT. The solution is also simple when the sensing range is much smaller than the RT such that several rows of observers are required to fill the RT. For sensing range in between, the best way to check that a sufficient number of observer can be found to observe the RT and that all observers will be inside the RT is by solving equation (3.1) against the two EPs of all three disks as shown in Figure 14. If there exists a solution for any pair of EPs, then there exist two observer locations which can be check whether one of the solutions is inside the RT. If no solution is inside of the RT, then reducing the size of the sensing range by some percentage point, 5% or 10% or at user’s discretion, and then repeat the new reduced sensing range with the two EPs. This process can solve the problem, although it may require more observer than necessary, but it is required for complete local coverage control.

After placement of the three observer waypoints in Figure 14(a), a linking algorithm such as the TSP’s NN algorithm maybe applied to link the three waypoints as a tour and then the cubic spline or adaptive curvature control can be applied to turn the linear spline into a continuous

![Figure 14. RT with three observers coverage/path](image)

(a) Three observer waypoints coverage.  
(b) Three observer waypoints and coverage path in blue and interpolated path in purple.

Figure 14. RT with three observers coverage/path
coverage path as in Figure 14(b). Now we are ready to formulate a RBOP theorem which guarantee sufficient number of observer and complete coverage of the RT. For clarity, we present a typical RT that we can refer to as in Figure 15.

\[(a) \ R_s \geq \max_{e \in E} D(e, p_l) \quad (b) \ R_s < \max_{e \in E} D(e, p_l)\]

Figure 15. Row Based Observer Placement (RBOP) algorithm

<table>
<thead>
<tr>
<th>TABLE 3. ROW BASED OBSERVERS PLACEMENT ALGORITHM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> An RT of disks (i, j,) and (k,) and an observer’s sensing range, (R_s.) The centers of disks (i, j,) and (k) are points I, J, and K respectively.</td>
</tr>
<tr>
<td><strong>Output:</strong> All observer points and their coordinates and their region to be observed.</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. Select (LS(I, K)) as the base of the RT. Once (LS(I, K)) is selected as the base of the RT, then the other two LSs are indirectly selected as the height of the RT.</td>
</tr>
<tr>
<td>2. Compute (l_1) and (l_2) such that they are perpendicular to the base of the RT at points (P_{1IK}) and (P_{2IK}).</td>
</tr>
<tr>
<td>3. Compute (row_\Delta,) and then compute (l_1, l_2, \ldots, l_{row_\Delta}.) Also compute (K_1, K_2, \ldots, K_{row_\Delta}.)</td>
</tr>
<tr>
<td>4. Compute (row_1, row_2, \ldots, row_{\max}, \max = row_\Delta.)</td>
</tr>
<tr>
<td>5. Compute the observer’s rectangle in each row. The centroid of each of the rectangle is the observer’s location. Note that the sides of the rectangle are equal to the SMSSD or smaller.</td>
</tr>
</tbody>
</table>
3.3.1 Row Based Observer Placement Theorem

Given any RT of 3 disks with one visible polygon, as shown in Figure 15, a sufficient number of circular observers needed for complete coverage is \( \sum_{i=1}^{\text{row}_\Delta} \text{row}_{Oi} \), where \( \text{row}_{Oi} \) is the number of circular observers in row \( i \). \( \text{row}_\Delta \) is the total number of row in the RT.

Proof:

It is easy to observe the LSs’ relationship in Figure 15(b) that
\[ l_j \perp l_{AK} \quad j = 1, 2, \quad \text{(3.10)} \]
\[ l_1 \parallel l_2. \quad \text{(3.11)} \]
\[ \text{row}_\Delta = \text{INT} \left[ \frac{\max(D(P_1AB, P_1BK), D(P_2AB, P_2BK))}{R_s\sqrt{2}} \right] \quad \text{(3.12)} \]
\[ \text{row}_{Oi} = \text{INT} \left[ \frac{\max(D(P_{iAB}, P_{iBK}), D(P_{i+1AB}, P_{i+1BK}))}{R_s\sqrt{2}} \right] \quad \text{(3.13)} \]

\( P_{iAB}, P_{iB}, P_{iBK}, P_{iAK} \) are points that can be found using the disk’s and the line’s formulae. Lines \( l_{AB}, l_{BK}, l_{AK}, l_j \) can be computed since the origin of disks A, B, and K are known. \( D(P_1, P_2) \) is the Euclidean distance between points \( P_1 \) and \( P_2 \). The area of a RT shown in Figure 15 (b) is calculated to be 48326 units square if it were to be completely covered by a circular sensor of sensing radius \( R_s = 9.43 \). The area of all the circular observers of radius \( R_s = 9.43 \) covering the RT is 79075, which result in a ratio of 1.6.

With RBOP algorithm, the target region in Figure 1 can be completely cover by 41 observers with a sensing range of \( R_s = 130.4 \). This may seems complicate as we begin by placing observers throughout the target region. A simpler example with RBOP finding observers for a single RT will be provided later.
In Figure 16, the observers are number in no particular order. The purpose of the number is only to count the total number of observer. More examples of complete local coverage with different observer’s sensing range are illustrated in Figure 18, Figure 19, and Figure 20.

Figure 17. Complete coverage (one or more observer waypoints per RT)
Figure 18. Complete coverage (one observer per RT)

Figure 18 is an example of sufficient sensing range relative to the RT. If an RT has a single visible polygon, then only one observer is required for that RT. If an RT has no single visible polygon, then it is partitioned into two sub regions. Each sub region is compute to find it visible polygon. Eventually, the observer’s position as the centroid of the visible polygon is computed.

Figure 19. Complete coverage with local planning
Figures 20 and 21 show examples of two different local coverage paths due to different sensing ranges. The linear spline LSs are the result of nearest neighbor’s algorithm which connected all observer points. The smooth curve segment is the result of performing cubic spline on the connected observer points.

Figure 20. Complete coverage path with local planning

Figure 21. Complete coverage path with local planning
Table 4 tabulated the numerical result of two different coverage paths based on their sensing range.

**Table 4. LCPP’S PERFORMANCE BASED ON SENSING RANGE**

<table>
<thead>
<tr>
<th>$R_s$</th>
<th>Number of observer</th>
<th>Optimal tour</th>
<th>Tour length</th>
<th>Length ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>36</td>
<td>2700</td>
<td>3439</td>
<td>1.26</td>
</tr>
<tr>
<td>280</td>
<td>26</td>
<td>2608</td>
<td>3216</td>
<td>1.23</td>
</tr>
</tbody>
</table>

Figure 22. The row based approach to find sufficient number of observers. Figure 22 illustrates two examples of RBOP algorithm with 4 rows of observers and 10 rows of observers. With the four rows observers for this particular example, the sensing range is 50, and there are 11 observers as seen in Figure 23(b). Figure 23(c) shows the three observers for three extreme regions closest to each of the disk. It is a simple illustration around the three disks. It is clear that the other 8 regions that are not block by the disk are observable.

Figure 23. RBOP in RT 5 when $R_s$ is 50. Observers are left to right and from bottom to up.
Figure 24 illustrates one of the shortest linear spline coverage path found by connecting the observers in Figure 23 with the TSP’s nearest neighbor algorithm. Figure 24(b) show row two being covered by two observers. Row 2 requires 4 observers four complete coverage of row. It is clear that if all four observers in row 2 are turned on, then row 2 is completely covered.

Figure 24. Coverage of an RT prior to PNWCC algorithm.

Figure 25 illustrates two possible modification of the RBOP arrangement to reduce the length of the coverage path produced by the Nearest Neighbor’s algorithm. The shorter of the two configurations is chosen as the temporary modified coverage path due to observer number six in row 2. If the clockwise ordering is chosen, then the next observer to be check for possible coverage path length modification is observer number 2 in row 1. The final coverage path for this example will be shown later in this Chapter.

Figure 25. Coverage of an RT during PNWCC algorithm.
3.4 Previous Next Waypoint Coverage Constraint Algorithm

When an infinite sensing range is available, coverage path planning can be studied through local and global analysis. A local and complete coverage path planning algorithm may employ the Previous Next Waypoint Coverage Constraint (PNWCC) algorithm presented in this section to reduce the distance, the number of observers, and the curvature of the coverage path. The PNWCC algorithm is useful for all sensing ranges. This is possible because the local coverage path planning approach with algorithm such as RBOP have plenty of overlap.

PNWCC algorithm takes the coverage linear spline path which may be formed by the TSP algorithm as input and then output a new path that is shorter, contains fewer sharp angles, and with reduces curvatures. The output coverage path remains a linear spline until we perform cubic spline transformation or adaptive curvature transformation to get a differentiable coverage path.

Consider a typical linear spline tour in Figure 26 which is generated by connecting the LCPP algorithm’s output and linked by the TSP’s NN algorithm. The sequence of the linear spline coverage path generated is 1, 3, 6, 5, 4, 9, 7, 8, 12, 10, 13, 11, 14, 39, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 40, 41, 38, 36, 37, 35, 34, 33, 25, 27, 26, 28, 29, 32, 2, 30, 31, and 1. Now let consider just one RT in Figure 27 which is presented in Figure 26 with observer waypoints 1, 2, and 3. For convenience, let the sensor be displayed in different colors, black, blue, and green as in Figure 27. We can clearly see that the RT in Figure 27(a) is completely covered by the 3 sensors. We can also see that there are plenty of excess sensing that are outside of the RT. The experiment is to check if moving the observer waypoints around will shorten the length while maintaining complete coverage. Any of the three waypoints maybe moved to reduce the path
length, but let choose the blue observer waypoint which is waypoint number 2 as the starting waypoint.

Now waypoint number 2 is the current waypoint. Based on the clockwise coverage tour sequence presented in Figure 26, waypoint number 32 is the previous waypoint, and waypoint number 30 is the next waypoint. By turning observer waypoint number 2 off, we have some part of RT uncover. This uncover region may be referred to as hole. Observer waypoints 1 and 3 are remained to cover the RT. The resulting hole has 5 EPs, \{e_1, e_2, e_3, e_4, e_5\}, to be covered by the new waypoint. The new waypoint is constrained by the five EPs as well as the previous and the next observer waypoints which are waypoints 32 and 30 respectively.

Figure 26. A typical linear spline tour, Rs is 130.4
A possible new waypoint is computed as a function of the EPs that surface as a result of turning off the observer waypoint number 2. These EPs are known as the hole vertices (HVs). The number of computation is two to the power of (HVs) plus one for the centroid. Since there are 5 HVs for this particular example, there are thirty three different values are computed and then compared to find the best result. If the new result is not reducing the path length, then
PNWCC algorithm keeps the old result. Figure 27 show how the center of the new waypoint is derived and computed.

Figure 27. An RT with observing sensors turned on/off; Rs is 130.4

(a) Right triangle with $R_s^2 = 0.5h^2$

(b) Right triangle with $R_s^2 = 0.25h^2 + v^2$

Figure 28. Searching for a “better” waypoint for a RT
From Figure 28, the equation that constraint the new waypoint to be selected or not is found to be

\[ D(p_k, p'_l) + D(p'_l, p_m) < D(p_k, p_l) + D(p_l, p_m). \] (3.14)

In equation (3.14) \( p'_l \) is the possible center of the new waypoint to replace the current waypoint, \( p_l \). The previous and the next waypoints are \( p_k \) and \( p_m \) respectively. The modified waypoints are given by equations (3.15) and (3.16):

\[ p'_{lx} = \frac{-k \pm \sqrt{k^2 - 4ac}}{2a}, \] (3.15)
\[ p'_{ly} = mp_{lx} + b, \] (3.16)

where

\[ m = \frac{e_{x_1}-e_{x_2}}{e_{y_2}-e_{y_1}}, \]
\[ b = \frac{e_{x_2}^2-e_{x_1}^2+e_{y_2}^2-e_{y_1}^2}{2(e_{y_2}-e_{y_1})}, \]
\[ a = m^2 + 1, \]
\[ k = \begin{cases} 2bm - 2e_{x_2} - 2e_{y_2}m, & \text{if } R_s^2 = 0.5h^2 \\ 2bm - 2e_{x_1} - 2e_{y_1}m, & \text{if } R_s^2 = 0.25h^2 + v^2 \end{cases} \]
\[ c = \begin{cases} e_{x_2}^2 + e_{y_2}^2 + b^2 - 2e_{y_2}b - 0.5d(p_{e_1}, p_{e_2})^2, & \text{if } R_s^2 = 0.5h^2 \\ e_{x_1}^2 + e_{y_1}^2 + b^2 - 2e_{y_1}b - R_s^2, & \text{if } R_s^2 = 0.25h^2 + v^2 \end{cases} \]
\[ h = D(e_i, e_j), \text{ where } e_i, e_j \in E \text{ and } i \neq j \]
\[ (e_{lx} - p'_{lx})^2 + (e_{ly} - p'_{ly})^2 = R_s^2, \text{ where } e_l \in E. \]
The modified waypoint for the centroid of the hole to be covered is given by equations (3.17) and (3.18):

\[ p'_{lx} = \frac{1}{6A} \sum_{i=0}^{n-1} (x_i + x_{i+1})(x_iy_{i+1} - x_{i+1}y_i), \]  
(3.17)

\[ p'_{ly} = \frac{1}{6A} \sum_{i=0}^{n-1} (y_i + y_{i+1})(x_iy_{i+1} - x_{i+1}y_i), \]  
(3.18)

<table>
<thead>
<tr>
<th>Table 5. Previous-Next Waypoints Coverage Constraint Algorithm 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A linear spline tour, ( s(x) = G(V, E) ), consisting of WPs with known beginning and ending, (e.g., CP in Fig. 33 (a)), and all regions in all RTs.</td>
</tr>
<tr>
<td><strong>Output:</strong> A better linear spline tour, ( G'(V', E') ), with fewer WPs and with shorter tour length (e.g., CP in Fig. 33 (f)).</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. Initialize all Regions in the RT to COVERED state, number of HVs for all WPs to zero, and all WP’s state to MODIFIABLE.</td>
</tr>
<tr>
<td>2. Select a WP to begin with if it state is MODIFIABLE.</td>
</tr>
<tr>
<td>3. If the select WP is MODIFIABLE, remove the selected WP from the Region it is associated with and then initialize the Region it is supposed to be covered to NOT COVER. Else, go to step 7.</td>
</tr>
<tr>
<td>4. Identify the row that the select WP is in. If more than three WPs are in the row, turn on the two closest neighboring WPs while turn off the selected WP (see Fig. 33(a)-(d)). Identify all HVs that surface and compute their cartesian coordinates, see Fig. 33 (b)-(d) for illustration. Else if only one WP is in the row, turn off the selected WP. Identify all HVs that surface and compute their cartesian coordinates.</td>
</tr>
<tr>
<td>5. Since the sensor is circular, only two HVs found in the previous step may be used at a time to compute the new WP. Use equations 3.14-3.18 to find the best new WP ( P'_i ) to replace ( P_i ). This may require 2 to the power of the number of HVs. The new WP must be able to restore the Region it has altered to COVERED state by verifying that all HVs are in the circle of the new WP as well as shorten the distance in equation 3.14. If the new WP cannot make improvement, keep the old WP. The new WP may be in the same Region or move to another Region. Once the new WP is determined, update the DLL of the new WP to NOT MODIFIABLE and their HV counter to 0.</td>
</tr>
<tr>
<td>6. If the next WP in the tour is not the last WP, select the next WP in the tour and repeat steps 3, 4, and 5. Else go to step 7.</td>
</tr>
<tr>
<td>7. All WPs have been considered for improvement. Check each and every Region for excess number of WP(s) and remove them. The new tour ( G'(V', E') ) is a set of piecewise continuous functions, and it is at least as good as the old tour or better. See Fig. 33(f).</td>
</tr>
</tbody>
</table>
In equations (3.17) and (3.18), \( n \) is the number of vertices of the resulting hole when an observer waypoint sensor is turned off. Now we are ready to formulate an algorithm for the PNWCC.

**Algorithm 2**

**Input:** A linear spline tour, \( s(x) = G(V, E) \), consisting of WPs with known beginning and ending, and all RTs and their respective VPs.

**Output:** A better linear spline tour, \( G'(V', E') \), with fewer WPs and with shorter tour length.

**Algorithm Steps Description:**

1. Initialize all RTs in the TR to COVERED state, number of HVs for all WPs to zero, and all WP’s state to MODIFIABLE.
2. Select a WP to begin with if it state is MODIFIABLE.
3. If the select WP is MODIFIABLE, remove the selected WP from the RT it is associated with and then initialize the RT it is supposed to be covered to NOT COVER. Else, go to step 8.
4. Check the number of WPs in the RT. If the number of WP is less than 4 and \( R_s \) is less than \( R_s \) in 3.8, then turn on all of the WPs in the RT while turn off the selected WP. Identify all HVs that surface as a result of removing the WP in this step and compute their Cartesian coordinates.
5. If an RT has only 1 WP and \( R_s \) meet the criteria in equation 3.9, compute the intersection of straightline \( L S(P_k, P_m) \) and the VP. If no intersection found, find the new WP with the vertices of the VP and equation 3.4-3.6. Keep WP \( P_l \) if neither the intersection nor the vertices condition is resulting in shorter \( L S(P_k, P_m) \). Go to step 7.
6. Since the sensor is circular, only two HVs found in the previous step may be used at a time to compute the new WP \( P'_l \). Use equation 3.14 to find the best new WP to replace the removed WP. This may require 2 to the power of the number of hole’s vertices. The new WP must be able to restore the RT(s) it has altered to COVERED state as well as shorten the distance of the WP it has modified. If the new WP cannot make improvement, keep the old WP. The new WP may be in the same RT or move to another RT. Once the new WP is determined, update the DLL of the new WP to NOT MODIFIABLE and their HV counter to 0.
7. If the next WP in the tour is not the last WP, select the next WP in the tour and repeat steps 3, 4, 5, and 6. Else go to step 8.
8. All WPs have been considered for improvement. Check each and every RT for excess number of WP(s) and remove them. The new tour \( G'(V', E') \) is a set of piecewise continuous functions, and it is at least as good as the old tour or better.
For convenience we used the linear spline tour or CP found by linking the output of the RBOP algorithm with the TSP’s NN algorithm as an input. The PNWCC algorithm may substantially improve the input CP in one iteration. It may be run for several iterations. The following illustrations show the original linear spline CP in blue, the enhance CP in red, after one iteration of PNWCC improvement of the blue CP, and the second iteration of PNWCC improvement of the blue CP in green. The third iteration of the CP with PNWCC algorithm is shown in black. The CPs in red, green, and black are pretty much similar except where the red segments and the green segments of the CPs are very visible near WP 7 and WP 33.

![Figure 29. LCPP when Rs is 130.4](image)

The linear spline input and the linear spline output of the LCPP approaches can also be found in Figure 30 with the input’s and the output’s tour sequences listed in the following paragraph.
Input linear spline path, $G(V, E)$. 

Output linear spline path, $G'(V', E')$. 

Figure 30. A LCPP and an Improved LCPP with PNWCC, Rs is 130.4

Input coverage path $G(V, E)$ has the clockwise tour sequence of (1-3-6-5-4-9-7-8-12-10-13-11-14-39-15-16-17-18-19-20-21-22-23-24-40-41-38-36-37-35-34-33-25-27-26-28-29-32-2-30-31-1). Output coverage path $G'(V', E')$ has the clockwise tour sequence of (1-3-6-5-4-12-10-13-11-14-39-16-17-19-20-21-22-23-24-36-35-33-25-27-26-28-2-30-1). Note that the waypoints in the input tour sequence and the output tour sequence may be represented by the same numerical number, but they are not necessarily equal in coordinate.

Figure 31. LCPP, red, to Improved LCPP, black, with differentiable CP, Rs is 163
Table 7 summarizes the result of both LCPP algorithms with different sensing range. Clearly, the PNWCC algorithm improves the tour length by about 25% while reduces the number of observer by up to 50%. The sensing range of 150 and 163 are producing very similar performance. Figure 32 show how much better the Improved LCPP’s CP is compare to CP due to the original CP due to a LCPP. The black coverage path of Figure 32 (a) show the first iteration of PNWCC algorithm. The green coverage path of Figure 32 (b) show the second iteration of PNWCC algorithm.

Table 7. RBOP’S AND PNWCC’S PERFORMANCE

<table>
<thead>
<tr>
<th>Rs</th>
<th>CWCP NC</th>
<th>PNWCC NC</th>
<th>CWCP Area (Million)</th>
<th>PNWCC Area (Million)</th>
<th>CWCP Length</th>
<th>PNWCC Length</th>
<th>Length Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>130.4</td>
<td>41</td>
<td>29</td>
<td>2.19</td>
<td>1.55</td>
<td>3872</td>
<td>2876</td>
<td>0.74</td>
</tr>
<tr>
<td>150</td>
<td>36</td>
<td>19</td>
<td>2.54</td>
<td>1.34</td>
<td>3439</td>
<td>2693</td>
<td>0.78</td>
</tr>
<tr>
<td>163</td>
<td>34</td>
<td>19</td>
<td>2.84</td>
<td>1.59</td>
<td>3439</td>
<td>2690</td>
<td>0.78</td>
</tr>
<tr>
<td>195.6</td>
<td>29</td>
<td>14</td>
<td>3.48</td>
<td>1.68</td>
<td>3228</td>
<td>2394</td>
<td>0.74</td>
</tr>
</tbody>
</table>

(a) Nonholonomic tour generated by LCPP algorithms in black and red. (b) PNWCC generated nonholonomic tour in one iteration in black and waypoint 10 run 2 times in green.

Figure 32. Complete coverage path for Rs equal to 163.
Figure 33 illustrated a relatively smaller size sensing range with the RT. Figure 33 illustrates a much smaller sensing range relative to the RT. The RBOP algorithm found 4 rows of observers which equate to 11 observers for the RT (see Figure 33(a)). The hole generated by WP 6 of row 2 is shown in Figure 33(b). With the RT’s row constraint, there are only 4 HVs. Running one iteration of the PNWCC algorithm yield the new CP shown in Figure 33(f).

![Figure 33](image)

(a) coverage with OWs 5 and 7.  
(b) Hole 6.  
(c) Hole 6’s vertices.  
(d) New OW.  
(e) Old and new observers in the RT.  
(f) Modified CP due to new observers.

Figure 33. Coverage of an RT prior to and post PNWCC algorithm.

Figure 34 pictorially show how the whole RT is observed based on one row at a time.

Note that after linking all observer points with the TSP’s NN algorithm, collision in the CP may exist. It is not treated right away, even after PNWCC algorithm. It will be treated in the next section with the static collision avoidance algorithm.

![Figure 34](image)

(a) Observed row 1.  
(b) Observed row 2.  
(c) Observed rows 3 and 4.

Figure 34. Coverage of an RT after 1 iteration of PNWCC algorithm.
3.5 Static Collision Avoidance

The Static Collision Avoidance (SCA) algorithm modifies the collision segment of the CP so that no collision occurs. SCA finds an equation of the line that is perpendicular to the LS that involve in collision and that also passes through the center of the disk. Any collision involved with a disk will have two possible solutions. The goal is to pick the intermediate point on the disk for the detour that provides the shorter distance. Figure 39(d) illustrates two points in gray dots that provide the solution and they are on the enlarged disk 5. For the example collision, the point with the shorter distance is to the right of the original LS that encounter collision. Note that all disks in Figure 39(d) are enlarged by the size of the robot.

A collision correction required an insertion of a WP into the CP. This means that two LSs are replacing a damage LS by inserting a new WP to detour the collision. A damage LS is a LS that collide with a disk. This new WP is not an OW. Additional details of the SCA algorithm can be found in Table 8.

**TABLE 8. STATIC COLLISION AVOIDANCE ALGORITHM**

| Input: A line segment with two end points I, J and the disk involved in collision, disk k. |
| Output: Two new LSs that detour the disk involve in collision, LS(I, K) and LS(K, J). |
| Algorithm Steps Description: |
| 1. Compute LS(K_B, K_G) that pass through the center of disk k and is perpendicular to LS(I, J). Note that K_B and K_G are on the enlarged circle of disk k and the robot’s radius. K_B ≤ \[ LS(I, J) < K_G \]. |
| 2. Compare and select the point K_B or K_G that result in shorter distance, LS(I, K_B) and LS(K_B, J) or LS(I, K_G) and LS(K_G, J). Name the selected point as K. |

3.6 Smooth Adaptive Curvature Control

Alternative to Cubic Spine algorithm is our Adaptive Curvature Control which allows more control on the CP. Applying Adaptive Curvature Control (ACC) algorithm on all LSs of the CP, the smooth CP for the TR can be found. Figure 36 illustrate the working process of Adaptive Circle (AC).
AC algorithm, one of two components of the ACC algorithm, begin with the observer point being considered and the two LSs connecting to it, LS(1,2) and LS(1,3). Figure 36 is a segment of the CP shown in Figure 41. To find an AC which is proportional to the 2 LSs, LS(1,2) and LS(1,3), the segment is first transformed into a triangle by connecting the open endpoints in Figure 35(a) to get Figure 35(b). Then find the minimum enclosing rectangle to wrap around the triangle. The third base of the triangle, LS(2,3), is always the opposite length to the WP under consideration. After finding the triangle as shown in Figure 35(b), the minimum enclosing rectangle maybe found with the three vertices of the triangle. The vertical line test (VLT) and the horizontal line test (HLT) at WP1 will help determine the center of the detour AC which is required to be on the VLT or the HLT that intersect the boundary of the rectangle and the third base of the triangle. Figure 35(d) illustrated the case that the HLT intersect the boundary rectangle as well as the third base of the triangle.

\[
ACD = \frac{1}{2} \min(L(P_1, P_2), L(P_1, P_3), (P_2, P_3))
\]  

(3.19)

This can be visualize with the leftmost side of the rectangle that pass through WP1; it did not intersect the third base. Figure 36(a) shows that the center of an adaptive circle is on the HLT as required. There may be two ACs to consider when both the HLT and the VLT meet the condition in step 3 of the AC algorithm. Either one may be used. Adaptive circle’s diameter is equate in equation (3.19). It may also be set to a certain size for optimal condition depending on

53
the vehicle and its operational requirements rather than with adaptive size. It is up to the user to
set this fixed size so that AC is neither too small nor too large.

![Figure 36. Adaptive curvature control for input tour.](image)

**Table 9. Adaptive Circle Algorithm**

<table>
<thead>
<tr>
<th>Input: WP, (I, J), and (I, K). Points I, J, and K are neighboring points with respect to the CP (See Figure 36 for illustrations).</th>
<th>Output: AC I, AC I will be used to find a single smooth curve segment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm Steps Description:</td>
<td></td>
</tr>
<tr>
<td>1. Find the third leg, LS(J,K), to get a triangle.</td>
<td></td>
</tr>
<tr>
<td>2. Find a rectangle enclosing points I, J, and K.</td>
<td></td>
</tr>
<tr>
<td>3. Find a HLS or a VLS that pass through WP I and also intersect with LS(J,K).</td>
<td></td>
</tr>
<tr>
<td>4. Find an AC with center on the HLS or VLS in step 3. Set AC ( D ) = 0.5(min(LS(I,J), LS(I,K), LS(J,K))) or equal to user specified-value.</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 37. Transition from ACs to desire tangent LS](image)
Figure 37 provided illustration for discontinuous to continuous transformation algorithm shown in Table 10. ACs 7 and 8 in Figure 37(a) may be referred to as ACs I and J respectively in the DC algorithm. In this example as well as in all of the WPs in Figure 41A, LS(7,8) represent the LS from points 7 to 8. LP(7,8) represent a line passing through both points 7 and 8. LP(7,8) is longer than LS(7,8) and it contains all elements of LS(7,8). The concept of LP is required to generalize a LS that is not long enough to intersect a circle at two points when it could if it were longer. For example LS(7,8) of Figure 41A(c) intersect AC 7 at two points and intersect AC 8 at only 1 point due to its length. The extension of LS(7,8) to LP(7,8) will intersected AC 7 at two points and AC 8 at two points. The LS(7,8) of Figure 41A(d) does not need to be extended to LP(7,8) as it is already intersecting AC 7 at two points and AC 8 at two points. The extension is necessary for some LSs to find a true tangent LS for smooth CP. Obvious LSs that required extension include LS(7,8), LS(7,5E), and LS(5,5E). A piecewise continuous CP is not desirable for nonholonomic robot. Before the CP can be transformed into a smooth and continuous CP with AC, the relationship between the two ACs and the piecewise continuous CP’s LS must be known so that an internal or external tangent LS can be found. Figure 41A(a) and (b) must be transformed into Figure 41A(c) and (d) respectively.

The different between Figure 41A(a) and (c) is that Figure 41A(c) also connected a LS from center of two ACs that was connected with a LS of two WPs on the two ACs. The LS that connect two ACs’ centers is called a reference LS. For example WPs 1 and 2 are connected on two ACs, now both ACs are connected center to center to form a reference LS.
If both counterpart in Figure 41A(d) will require an external tangent LS modification. After finding the example, LS(7,8) in Figure 41A(c) will require an internal tangent LS modification, but its PNWCC algorithm may alter the relationship between a WP LS and a reference LS. For LS(4,5) and the reference LS(AC4, AC5) will required an internal tangent LS modification. The relationship between LS(1,2) and the smooth CP can be obtained as shown in Figure 37(h). If I is less than LP(I, J), set \( I_t = I_b \). Else set \( I_t = I_g \).

10. Compute the intersection of \( \text{LP}(I_t, J_t) \) and AC J. If only one intersection is found, set \( J_t = J_b \). Else if two intersections are found, set \( J_t = J_g \) and \( J_j = J_j \).

11. If both State \( I = True \) and State \( J = True \), end the program.

The relationship between LS(1,2) and LS(AC1, AC2) will require an external tangent LS modification and the tangent line is to the left of the reference LS. The relationship between LS(4,5) and the reference LS(AC4, AC5) will required an internal tangent LS modification. The PNWCC algorithm may alter the relationship between a WP LS and a reference LS. For example, LS(7,8) in Figure 41A(c) will require an internal tangent LS modification, but its counterpart in Figure 41A(d) will require an external tangent LS modification.
Figure 41B with the DC process illustrated in Figure 37. While Figure 41B showed only one iteration of DC transformation, some ACs and LSs may take up to 3 iterations to find the smooth CSs, but no more than 3 iterations. Each arc in Figure 41B has 3 WPs. Arcs 1, 4, 5, 7, 8, 10, 9, and 6 have 2 WPs and 1 OW. Arc that detours around disk 5 has 3 WPs with no OW. As a reminder, an OW is an observer WP as well as a WP. Note the fundamental different between (11) and (15). All WPs in (11) are OWs, but not all WPs in (15) are OWs. Once all ACs and their WPs are found for a piecewise continuous CP, their desirable arcs as illustrated in Figure 41B(d) can be found by drawing a LS to connect the two tangent WPs, \( I_t \) and \( J_t \), and then keep the arc that contains all three WPs. The smooth CPs are obtained as in Figure 42 by connecting all arcs and all tangential LSs.

### 3.7 A Local Coverage Path Planning Approach

Having present new contributed algorithms from Sections 3.1 to 3.6, it is now appropriate to organize the order of precedence for a local triangulation-based coverage path planning (LCPP) approach which consists of four phases and 6 auxiliary algorithms as shown in the tables below. Figure 43 illustrates a sample target region with 9 disks which will be used to demonstrate a suite of algorithms which together form the LCPP approach. Obviously, there are 10 RTs in the sample TR.

Figure 41A illustrated the piecewise continuous CP of the sample TR given in Figure 43. It is the same as the CP in Figure 39(d), except that the actual disks of the TR are not shown; only the CP and ACs are shown for clarity. Initially, the number of AC is equal to the number of WP.
### Table 11. Four Phases of the Triangulation-Based Coverage Path Planning Approach and their Descriptions.

1. **Local Observer Planning (LOP)** – Developed to place observer(s) in each disjoint subregion of the TR which is called regular triangulation (RT). Three different algorithms are developed to place observer in the RT depending on the relative size of the RT and $R_s$:

2. **Baseline Path Planning (BPP)** – Developed to generate a CP based on the observers found in phase 1. The CP obtained in this phase may have a collision with the known disks in the TR. All collisions may be removed as required while completely observing the TR. If collisions are not removed in this phase, they will have to be removed in the next phase.

3. **Path and Observers Replanning (POR)** – Developed to reduce the CP length while also reducing the number of observer required to observe the TR. All collisions will be removed in this phase if they are not already removed in phase 2. This phase is optional for efficiency. A Doubly Linked List data structure is used to systematically organize and access data.

4. **Smooth Path Replanning (SPR)** – Developed to smooth the coverage path for continuity and efficiency while still maintaining complete coverage of the TR.

### Table 12. Auxiliary Algorithms Used by the LCPP Algorithm.

1. **Visible Polygon (VP)** – developed to compute the convex polygon inside the RT from which a sufficient range observer can be placed within the convex polygon to observe the entire RT.

2. **Row Based Observers Placement (RBOP)** – developed to find a set of observers within the RT that can collectively observe the entire RT. This algorithm is used when an RT is relatively larger than the sensing range, $R_s$.

3. **Static Collision Avoidance (SCA)** – developed to replace the colliding line segment of the CP with two new line segments that detour around the disk that involve in collision.

4. **Previous Next Waypoint Coverage Constraint (PNWCC)** – developed to reduce the path length while maintaining complete coverage. This algorithm may move observers within the TR or delete them as needed.

5. **Adaptive Circle (AC)** – developed to find smooth curvature between three waypoints and the two LSs connecting them.

6. **Discontinuous to Continuous (DC)** – developed to transform the linear spline CP, piecewise continuous CP, into smooth CP while maintaining the same observer location.
TABLE 13. THE FOUR PHASES OF LCPP APPROACH AND THEIR SUB ALGORITHMS

1. Local Observer Planning
   (E) Delaunay Triangulation
   (D) Visible Polygon
   (D) Circular Coverage Placement
   (D) Row Based Observer Placement

2. Baseline Path Planning
   (E) Nearest Neighbor
   (D) Static Collision Avoidance

3. Path and Observers Replanning
   (D) Previous Next Waypoint Coverage Constraint
   (D) Static Collision Avoidance

4. Smooth Path Replanning
   (D) Adaptive Circle algorithm
   (D) Discontinuous to Continuous algorithm

TABLE 14. THE STATIC WORKSPACE CONFIGURATION OF A SAMPLE TARGET REGION 2

<table>
<thead>
<tr>
<th>Disk</th>
<th>O_{lx}</th>
<th>O_{ly}</th>
<th>R_i</th>
<th>Disk</th>
<th>O_{lx}</th>
<th>O_{ly}</th>
<th>R_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>100</td>
<td>30</td>
<td>6</td>
<td>600</td>
<td>900</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>900</td>
<td>30</td>
<td>7</td>
<td>630</td>
<td>600</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>350</td>
<td>500</td>
<td>40</td>
<td>8</td>
<td>1000</td>
<td>100</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>400</td>
<td>100</td>
<td>30</td>
<td>9</td>
<td>1000</td>
<td>900</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>550</td>
<td>300</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 38. A sample target region with 10 disks and its 10 RTs
**TABLE 15. THE FOUR PHASES OF LCPP APPROACH AND THEIR SUB ALGORITHMS**

1. **Local Observer Planning**
   (E) Delaunay Triangulation
   (D) Visible Polygon
   (D) Row Based Observer Placement

2. **Baseline Path Planning**
   (E) Nearest Neighbor
   (D) Static Collision Avoidance

3. **Path and Observers Replanning**
   (D) Previous Next Waypoint Coverage Constraint (Optional, if implement, the CP may be shorter, smoother and contains fewer observers).
   (D) Static Collision Avoidance

4. **Smooth Path Replanning**
   (D) Adaptive Circle algorithm
   (D) Discontinuous to Continuous algorithm

---

(a) A TR and its RTs. (b) VPs’ centroids. (c) Centroidal CP. (d) Modify CP.

**Figure 39. Transition from VPs to Centroidal CP for sufficient sensing range.**

(a) Straightline is not observable. VPs of RT 6 are above the straight line $P_3P_9$.
(b) Shortest observable path $P_3P_9$ cannot be used. It is replaced by two new LSs: $P_3P_6$ and $P_9P_6$.
(c) Straight line, $P_8P_9$, is observable.
(d) Observable points are any point between two blue dots inclusive.

**Figure 40. Visibility through VPVs and straight line intersection.**
Figure 41 (A). ACs and their reference LSs.

Figure 41B. One iteration of tour reduction through PNWCC and AC control

Figure 42. Two differentiable CPs of the TR.

Most of the WPs in Figure 41A are OWs, except for the WP between WPs 5 and 7 which are collision avoidance WP. In Figure 41A(a), it is the WP closest to the enlarged disk 5 which is label as 5E. At this point, it should be clear that the CP obtain after the PNWCC algorithm reduced the CP length, reduce the number of observers, and also reduce the number of turns.
LS(2,4) in Figure 41A(a) will be replaced with external tangent LS because it does not intersect with LS(AC2, AC4) as shown in Figure 41A(c). If the two were to be intersected, then LS(2,4) will be replaced with internal tangent LS. Figure 36 illustrated the process of replacing the piecewise continuous LS for smoothness. Note that OWs are not replaced, they remain at the same location. Only the CP’s LS leading to the OWs are replaced and linked to OW by an arc of the AC. See Figure 41B(d) for the complete transformation from piecewise continuous CP to the smooth CP and the preservation of the OWs.

![Figure 43. The transition from the sample TR to a smooth CP with a LCPP algorithm.](image)
Table 16. Running Time of The Local Coverage Path Planning Approach

<table>
<thead>
<tr>
<th>Phase</th>
<th>Algorithm</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Local Observer Planning</strong></td>
<td>Triangulation algorithm</td>
<td>$N \log N$</td>
</tr>
<tr>
<td></td>
<td>Visible Polygon algorithm</td>
<td>$O(M)$</td>
</tr>
<tr>
<td></td>
<td>RBOP algorithm</td>
<td>$O(KM) \approx O(M)$</td>
</tr>
<tr>
<td><strong>Baseline Path Planning</strong></td>
<td>Traveling Salesman algorithm</td>
<td>$O(M^2)$</td>
</tr>
<tr>
<td></td>
<td>Static Collision Avoidance</td>
<td>$O(M)$</td>
</tr>
<tr>
<td><strong>Path and Observers Replanning</strong></td>
<td>PNWCC</td>
<td>$O(M)$</td>
</tr>
<tr>
<td></td>
<td>Static Collision Avoidance</td>
<td>$O(M)$</td>
</tr>
<tr>
<td><strong>Smooth Path Replanning</strong></td>
<td>Adaptive Curvature algorithm</td>
<td>$O(M)$</td>
</tr>
<tr>
<td>Summing all running time in all 4 phases</td>
<td></td>
<td>$O(M^2)$</td>
</tr>
</tbody>
</table>

Figure 44 to Figure 50 provides additional examples of LCPP approach with a sample target region of 18 disks. The sensing range in these examples is considered sufficient because each RT required only one observer. Figure 45 shows the result of the target region being partitioned into 24 simple cells or 24 RTs. Each of the 24 RTs are processed to find its VP which is shown in cyan color. Then the centroid of each VP is computed to place an observer due to sufficient sensing range. Once all observers are placed, a NN algorithm is called to find the linear spline coverage path shown in Figure 44.

Figure 44. Centroidal Linear spline tour
Figure 45. ACs of the Linear Spline CP

Figure 45 shows adaptive circles of each of the observer point which is a continuation of the LCPP approach shown in Figure 44 except that the VP of each RT is not shown for clarity. Each adaptive circle is different from each other due to the length of the two edges connecting with the observer. Figure 46 show how segment of each of an adaptive circle is integrated with the linear spline CP due to discontinuous to continuous algorithm. The coverage path is very smooth which is desirable for the nonholonomic system.

Figure 46. Smooth centroidal CP due AC control

Figure 44 to Figure 46 show the LCPP approach without the use of PNWCC algorithm. As a result, the CP is not efficient in term of CP length. It is evidence in Figure 50 that the CP can be shorter and contain fewer ripples. Figure 47 show how CP in Figure 44 can be improved, in only one iteration. The PNWCC algorithm may be run in many iterations, but it can be control to continue running until “no improvement” is notice so that the computer process can do other works. Figure 50 compare the two LCPP results.
Figure 47. Centroidal CP after one iteration of PNWCC algorithm

Figure 48. Centroidal CP after two iterations of PNWCC algorithm

Figure 49. Centroidal CP after three iterations of PNWCC algorithm

Figure 50. Smooth centroidal CP and its third iteration with PNWCC algorithm
### 3.8 Coverage Path Planning For a Group of Robots

The Voronoi partition, \( V(p) = \{V_1, V_2, \ldots, V_N\} \), of a target region \( \Omega \) depends on the number of \( n \) sites or \( n \) generators desired. If a point \( p_l \) lies in the cell corresponding to a site \( V_l \in V \) then the following conditions must hold:

\[
V_l = \{p \in \Omega | D(p, p_l) \leq D(p, p_k), \forall l \neq k\}, \quad \text{and} \quad (3.20)
\]

\[
V_i \cap V_j = 0 \text{ for } \forall i \neq j. \quad (3.21)
\]

Voronoi diagram can partition the given target region into smaller areas that meet the constraints above in \( O(N(\log N)) \) time. Partition the target region with Delaunay Triangulation and then partition the TR into 5 groups with the 5 sites displayed in solid black dot, green dot, blue dot, red dot, and cyan dot yields the 5 corresponding tours after filling in observers with the RBOP algorithm.

![Voronoi Diagram](image.png)

**Figure 51.** Group coverage path planning with LCPP approach (Rs is 130.4)
Note that summing the tours’ length in Figure 51 yield greater length than just a single tour. The travel path for each tour is describes in Table 17 along with the length and the efficient ratio.

Table 17. LCPP’s performance based on sensing range and Voronoi partition

<table>
<thead>
<tr>
<th>Robot</th>
<th>Length</th>
<th>Ratio</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>682</td>
<td>1.39</td>
<td>1-3-6-5-4-2-1</td>
</tr>
<tr>
<td>Green</td>
<td>821</td>
<td>1.5</td>
<td>7-8-9-12-10-37-35-34-7</td>
</tr>
</tbody>
</table>

Total length for five tours = 4439  Ratio = 1.38

The communication or sensing matrix for multiple robots presented in Figure 51 can be represented by a matrix shown in Figure 52. \( S(t) = S(t_k^s) \) for all \( t \in [t_k^s, t_{k+1}^s] \), where \( s_{ii} = 1 \) and \( s_{ij}(t) = 1 \) if the \( j^{th} \) system is in the sensor range of the \( i^{th} \) system at time \( t \), and \( s_{ij} = 0 \) if otherwise.

\[
S(t) = \begin{bmatrix}
S_1(t) \\
S_2(t) \\
... \\
S_q(t)
\end{bmatrix}
= \begin{bmatrix}
S_{11} & s_{12}(t) & \cdots & s_{1q}(t) \\
S_{21}(t) & S_{22} & \cdots & s_{2q}(t) \\
\vdots & \vdots & \ddots & \vdots \\
s_{q1}(t) & s_{q2}(t) & \cdots & s_{qq}
\end{bmatrix}
\]

Figure 52. Group communication/sensing matrix for q nodes

Figure 53 shows that the Previous Next Waypoints Coverage Constraint algorithm is also effective for coverage path planning for a group of mobile robots. For instance, the Black path in Figure 51 is reduced to length of 651 from 682. From a quick glance we can see that each and
Figure 53. LCPP with RBOP and PNWCC algorithms, Rs is 130.4

Every tour in Figure 52(b) is shorter and smoother than its corresponding tour in Figure 51. Their numerical tour length, ratio, and path sequence are tabulated in Table 18. Except the Blue CP, all others improved by 20% or more. The Blue CP improves by just 2% in term of path length, but the curvature improves drastically and is very desirable by the nonholonomic system.

Table 18. GROUP COVERAGE PATH PLANNING (LCPP VS ILCPP), RS IS 130.4

<table>
<thead>
<tr>
<th>Robot</th>
<th>Length</th>
<th>Ratio</th>
<th>Path (Number of Circle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>682</td>
<td>1.39</td>
<td>1-3-6-5-4-2-1 (6)</td>
</tr>
<tr>
<td>Green</td>
<td>921</td>
<td>1.5</td>
<td>7-8-9-12-10-37-35-34-7 (8)</td>
</tr>
<tr>
<td>Cyan</td>
<td>983</td>
<td>1.39</td>
<td>25-27-26-30-31-32-28-29-33-25 (9)</td>
</tr>
</tbody>
</table>

(b) ILCPP, Total length = 3769

<table>
<thead>
<tr>
<th>Robot</th>
<th>Length</th>
<th>Ratio</th>
<th>Path (Number of Circle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>651</td>
<td>1.33</td>
<td>1-3-5-4-2-1 (5)</td>
</tr>
<tr>
<td>Green</td>
<td>697</td>
<td>1.27</td>
<td>7-9-10-35-34-7 (9)</td>
</tr>
<tr>
<td>Red</td>
<td>781</td>
<td>1.25</td>
<td>41-19-20-21-23-24-41 (6)</td>
</tr>
<tr>
<td>Cyan</td>
<td>845</td>
<td>1.2</td>
<td>25-27-26-30-32-33-25 (6)</td>
</tr>
</tbody>
</table>

Table 19 shows the set of steps for PNWCC algorithm. Both, LCPP approaches required triangulation as the first step.
Table 19. **LCPP APPROACH IN THE SEQUENCE OF STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Delaunay Triangulation</td>
<td>Performs Step 1 to divide the target region into a finite number of RT.</td>
</tr>
<tr>
<td>2</td>
<td>RBOP algorithm</td>
<td>Performs Step 2 to locally place observer in each RT to ensure complete coverage.</td>
</tr>
<tr>
<td>3</td>
<td>Nearest neighbor algorithm</td>
<td>Performs Step 3 to link all observers into a single coverage path. The resulting coverage path is a linear spline tour.</td>
</tr>
<tr>
<td>4</td>
<td>PNWCC algorithm</td>
<td>Performs Step 4 to reduce coverage path length, its curvature, and sharp angles.</td>
</tr>
<tr>
<td>5</td>
<td>Adaptive curvature</td>
<td>Performs Step 5 to transform the linear spline coverage path into a continuous and differentiable coverage path.</td>
</tr>
<tr>
<td>6</td>
<td>Dynamic obstacle avoidance algorithm</td>
<td>Performs Step 6 to steer the robot from the starting point to the ending point to perform coverage task if a dynamic obstacle is detected.</td>
</tr>
</tbody>
</table>

### 3.9 Conclusions

In this chapter we show how exact decomposition technique is performed on the target region to obtain disjoint RT. From the RT, we found a visible polygon, RBOP algorithm, and PNWCC algorithm. We show that LCPP approach with RBOP, and PNWCC algorithms achieve complete coverage through exact partition. Our LCPP approaches achieve differentiable coverage path. The steps to obtain the CP with the LCPP approach are included below to study the failure conditions in each of the phases.

There are two main failure conditions for the local coverage path planning algorithm. The first failure condition is the target region which cannot have all disks in the target region to be collinear in term of their centers. If this happens, Delaunay triangulation would encounter degeneracy. This failure condition may be avoided by padding a non collinear point in the target region provided that all disks in the target region met assumption 4 in the formulation of our coverage problem. The second failure condition is an assumption 4. The set to be covered is connected with respect to the size of the robot. This means that the robot can maneuver in all simple cells, RTs, of the target region.
Table 20. **INPUTS AND OUTPUTS OF THE LCPP APPROACH**

<table>
<thead>
<tr>
<th>step</th>
<th>phase</th>
<th>Input</th>
<th>Algorithm</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Sample TR</td>
<td>Delaunay Triangulation</td>
<td>Multiple RTs in the TR</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>TR with “simple cells”, Multiple RTs</td>
<td>Visible Polygon</td>
<td>VP of each RT if exist</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>RT with VP</td>
<td>RBOP</td>
<td>Observers in the TR</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>RT with sufficient observer</td>
<td>Nearest Neighbor</td>
<td>Linear spline CP</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>Linear Spline CP</td>
<td>PNWCC</td>
<td>Linear Spline CP that may be shorter and has fewer observer</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>Linear Spline CP</td>
<td>SCA</td>
<td>Collision free Linear Spline CP</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>Collision free Linear Spline CP</td>
<td>AC</td>
<td>Collision free Linear Spline CP with ACs.</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Collision free Linear Spline CP</td>
<td>DC</td>
<td>Final smooth CP.</td>
</tr>
</tbody>
</table>

Table 21. **ILCPP APPROACH’S Failure Conditions**

<table>
<thead>
<tr>
<th>step</th>
<th>phase</th>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Delaunay Triangulation</td>
<td>Failure condition occurs if all disks’ centers in the TR are collinear.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Visible Polygon</td>
<td>A single VP in an RT may not exist. Splitting the RT into 2 SRs may be required. Further partition of the SR may be done as well.</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>RBOP</td>
<td>This algorithm is implemented if the RT is larger than the sensor which means that sensing range is smaller than equation 3.8. Failure condition occurs if the observer is outside of the RT. Under such condition, the sensing range can be reduced by a small fraction and RBOP is called again with the new pair of RT and reduced sensing range.</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Nearest Neighbor</td>
<td>Since collision is allowed in this step, there is no failure condition.</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>PNWCC</td>
<td>Since collision is allowed in this step, there is no failure condition.</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>SCA</td>
<td>Provided that assumption 4 holds, this algorithm has no failure condition.</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>AC</td>
<td>There is no failure condition.</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>DC</td>
<td>There is no failure condition.</td>
</tr>
</tbody>
</table>
CHAPTER 4 A GLOBAL COVERAGE PATH PLANNING APPROACH

4.1 Introduction

Our earlier coverage path planning approaches such as a LCPP approach and an improved LCPP approach result in fundamental engineering problems. As sensing range approach infinity, the LCPP approach generated a coverage path that is formed by many ripples. Since an improved LCPP approach takes the output of the LCPP approach as input, it is also a local planner which is trying to reduce the number of observer point, the length of coverage path, and the curvature of the coverage path through the previous, the current, and the next waypoints as the constraint. An improved LCPP approach improves the result of a LCPP approach as seen in Table 18.

A LCPP approach and an improved LCPP approach generate differentiable coverage path through Discontinuous to Continuous transformation algorithm. Although significant advantage can be seen with improved LCPP algorithm’s result such as a new coverage path that is shorter, has fewer sharp angles, and has reduced curvatures, an improved LCPP is far from achieving optimal result because of saturation. The saturation example can be seen in Figure 54. One complete iteration of the PNWCC with the input data being the output of LCPP algorithm at Rs = 200 give the result shown in blue in the same figure. No further improvement can be made after the second iteration. This shows that the algorithm converges very quickly. The green coverage path suggests that shorter coverage path is possible if it can be found to wrap around some of the interior obstacles more tightly.
The answer is a Rainbow coverage path planning approach, also known as the GCPP approach, which is illustrated in Figure 55. The Rainbow coverage path planning algorithm involves seven phases with each phase having several steps. All phases of the Rainbow algorithm are briefly described in Table 22.

The wonderful things about the Rainbow algorithm include simplicity, fast, accurate, efficient, effective, and programmable. The simplicity of the Rainbow algorithm is that each phase of the Rainbow algorithm is fast and provides simple output which allows it to be further processed as input by the next phase. The Rainbow algorithm is an input to output transformation algorithm.
Table 22. **RAINBOW ALGORITHM PHASE DESCRIPTION**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 RedPath</td>
<td>Generate a foundational path based on the interior disks’ position. The result of this algorithm is a convex hull that has all interior disks’ center lie either inside the convex hull or on the convex hull.</td>
</tr>
<tr>
<td>2 OrangePath</td>
<td>Processes the RedPath to obtain a coverage path that wrap around all interior disks whose center is on the convex hull/RedPath. The OrangePath may intersect/collide with one or more disks.</td>
</tr>
<tr>
<td>3 YellowPath</td>
<td>Processes the OrangePath to obtain a coverage path that wrap around all interior disks as well as avoiding collision with all known disks.</td>
</tr>
<tr>
<td>4 GreenPath</td>
<td>Processes the YellowPath to obtain a coverage path that can observe all partitioned regions of the target region that the coverage path crosses. The GreenPath inherits the collision-free property of the YellowPath.</td>
</tr>
<tr>
<td>5 BluePath</td>
<td>Processes the GreenPath to obtain a first order differentiable coverage path. The BluePath inherits the collision-free’s and the observability’s properties of the GreenPath.</td>
</tr>
<tr>
<td>6 IndigoPath</td>
<td>Processes the BluePath to obtain a coverage path that can observe all partitioned regions of the target region that the BluePath does not cross. The IndigoPath is actually the BluePath with the ability to observe all regions of the target region.</td>
</tr>
<tr>
<td>7 VioletPath</td>
<td>Process the IndigoPath to obtain a coverage path that can observe all partitioned regions of the target region. The VioletPath is actually the IndigoPath with all observe points found to observe the entire target region.</td>
</tr>
</tbody>
</table>

Table 23 provides a summary of technique(s) or algorithm(s) that each phase need.

Some of the algorithms are existing algorithms found in the literature such as Graham’s scan algorithm, but the majority of the algorithms are our new contributions.

Table 23. **MAIN ALGORITHM/TECHNIQUE OF THE RAINBOW ALGORITHM’S PHASE**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Technique(s) or Algorithm(s) employed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 RedPath</td>
<td>Graham Scan algorithm.</td>
</tr>
<tr>
<td>2 OrangePath</td>
<td>BURL algorithm and simple algebra.</td>
</tr>
<tr>
<td>3 YellowPath</td>
<td>BURL algorithm and simple algebra.</td>
</tr>
<tr>
<td>4 GreenPath</td>
<td>Delaunay Triangulation, visible polygon, CSIRT, and Visible Circle Path Modification.</td>
</tr>
<tr>
<td>5 BluePath</td>
<td>Algebraic relation.</td>
</tr>
<tr>
<td>6 IndigoPath</td>
<td>LosPoRT, LosPoRT to visible curve segment transformation.</td>
</tr>
<tr>
<td>7 VioletPath</td>
<td>Line intersection for grid and visible curve segment(s) intersection and observer point determination algorithm.</td>
</tr>
</tbody>
</table>
The GreenPath algorithm, which is phase number 4, is the most involve phase of the Rainbow algorithm. It is the phase that decides the overall runtime of the Rainbow algorithm as will be seen in Chapter 5. Each phase of the Rainbow algorithm is described in one full section for readability. Table 23 shows the main algorithm(s) employ in each of the phase. Existing algorithms that we implemented include Graham Scan, Delaunay Triangulation, and Line intersection. Our main contribution in the family of Rainbow algorithm include BURL, visible polygon, CSIRT, VCPM, LosPoRT, LosPoRT to visible curve segment mapping, and observer point determination algorithm. The algorithms described above will be explained as we present each of the phases in their corresponding sections.

Pictorial representation of each of the phase is illustrated in Figure 56. The target region to be searched is shown in Figure 56(a). The sample target region shown in Figure 1 is feed to the RedPath algorithm as input. The output of the Redpath algorithm is a convex hull shown in Figure 56(b). The convex hull obtained by the RedPath algorithm is feed into the OrangePath algorithm as input. The OrangePath algorithm produce an output coverage path that envelope all interior disks and the result is shown in Figure 56(c). At this stage the coverage path may intersect with interior disk(s) and exterior disk(s), this phenomenon is simply known as collision, and discontinuity may exist in the coverage path on the curve segment around the disk whose center is on the convex hull. Discontinuity is highly visible around disks 11 and 12 in our example of Figure 56(c). The objective of YellowPath algorithm is to remove coverage path collision with all interior disks and all exterior disks.
Figure 56. Transition from the TR to the Rainbow CP Planning algorithm result

Figure 56(d) shows how internal tangent principle regenerates the new coverage path that is collision free with disk 7. At this stage our coverage path is collision with all known disks, interior disks and exterior disks. The YellowPath or Yellow phase’s coverage path is then input
into the GreenPath algorithm which is the most extensive phase of the Rainbow algorithm. The GreenPath algorithm check, and if necessary modified the Yellow coverage path to the Green coverage path. The Green coverage path is a coverage path that guaranteed to have observer points for all RTs that the Green coverage path intersects. At this phase of the Rainbow algorithm, the coverage path is collision free with all interior and exterior disks and existence of observers to observe all RTs that the coverage path intersects as seen in Figure 5(e). The Green coverage path is then feed into the BluePath algorithm to correct any discontinuity problem as seen in Figure 5(f). An IndigoPath or Indigo coverage path is the same as the Blue coverage path, but has more information encoded into the coverage path which is the coordinate of the candidate visible curve segments (CVCS). The CVCSs produce candidate observers, some of which are needed to observe the entire target region. The VioletPath algorithm produces and processes the candidate observers to find the sufficient number of observer needed for the TR. The result of the VioletPath algorithm is presented in Figure 5(h).

The result of each and every phase of the rainbow coverage path planning is presented in Figure 56. The convex hull representing the RedPath is shown in dash-red. The OrangePath is not visible in the figure because it is beneath the YellowPath. The majority of the GreenPath is visible, but some segment of it is overlay by the BluePath. Most of an IndigoPath is visible as double thin lines around the GreenPath, the BluePath, and the YellowPath. Finally, the result of the VioletPath are visible as observer circles: small red disk surrounded by slightly larger green circles and purple circles.
4.2 The RedPath Algorithm

The RedPath algorithm occurs during the first phase of the seven phases in a Rainbow Coverage Path Planning algorithm. Its purpose is to generate a foundational path so that the next phase can process it to wrap around all interior disks. As such, only the center point of the interior disk is considered. Many algorithms can be selected to compute the RedPath. The Graham Scan algorithm is chosen due to its popularity and its log linear computation speed.
Figure 58 presented the input and the output of the RedPath algorithm. Only the interior disks are process to find the foundational coverage path. The exterior disks remain un-touch. The RedPath algorithm involves three steps as shown in Figure 58. At the end of the Redpath, the interior disks remain points which will be restored to their original size in the OrangePath algorithm. Readers are refer to reference [96] for the logic of the Graham Scan’s algorithm.

**Table 24. The RedPath Algorithm.**

<table>
<thead>
<tr>
<th>Input: The TR with all known static disks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: A convex hull of interior disks’ center.</td>
</tr>
<tr>
<td>Algorithm Steps Description:</td>
</tr>
<tr>
<td>1. Remove all exterior disks of the TR.</td>
</tr>
<tr>
<td>2. Reduce all remaining interior disks to points, interior points.</td>
</tr>
<tr>
<td>Find convex hull of interior points with Graham Scan algorithm.</td>
</tr>
</tbody>
</table>

4.2.1 **Foundational Coverage Path Algorithm, the Graham Scan’s**

The foundation path compute by the RedPath algorithm for our example problem is shown in Figure 59. The points that made up the RedPath in clockwise direction are points 5, 11, 15, 12, 10, and 6. The points that are inside of the RedPath are points 8 and 9.

4.2.2 **Input of the RedPath Algorithm**

Input to this module is the target region and all known static disks. The known static disks are $O_{si}$ where $si \in \{1, 2, \ldots, N\}$ with N equal to 18 for our example. With the given configuration of the target region in Figure 1, the interior disks are $O_{io}$ where $io \in \{5, 6, 8, 9, 10, 11, 12, 15\}$. 

78
4.2.3 **Output of the RedPath Algorithm**

![The RedPath of the sample target region](image)

The convex hull approach to path determination is the first phase of the seven phases approach to differentiable path determination. The idea is very simple to implement since the given problem with finite number of disks and the boundary enclosing the disks are known. The idea of the convex hull path, we prefer to call it the Redpath, algorithm is simply removing all disks that are on the boundary of the piecewise convex hull. Only the remaining interior disks are process to find the Redpath. All remaining interior disks are first reduce to point and then apply one of the fastest running algorithms to find the convex hull of the interior points such as Graham Scan’s algorithm which can compute the convex hull in $N\log(N)$ time where $N$ is the number of points [96]. In our problem, $N$ is the number of interior disks. The RedPath algorithm has 3 steps as shown in Table 24.
4.3 The OrangePath Algorithm

The YellowPath algorithm occurs during the second phase of the seven phases in a Rainbow Coverage Path Planning algorithm. Its purpose is to envelop all interior disks so that the next phase can process it to deter collision with all interior disks and exterior disks. As such, only the disk with center point on the RedPath is considered. To transform the RedPath into an OrangePath, the relative position of all RedPath’s LSs must be known. The relative position of all RedPath’s LSs can be determined by the Bottom, Upper, Right, and Left (BURL) algorithm. Figure 62 illustrates the detail of the BURL algorithm.

![Figure 60. The OrangePath of the sample TR](image)

The OrangePath algorithm is presented in Figure 60 with 5 steps as shown in Table 25, and the first step involve the BURL algorithm which is shown in Table 26. The second step of the OrangePath algorithm discusses how the LS of the RedPath are moved to their new position based on their relative position with respect to the RedPath’s convex hull and the size of the point’s original disk. The key point is that the original LS have two end points at the center of the two disks while the new LS have two end points on the exterior of the two disks. The exact coordinate of the OrangePath’s LS endpoints are influence by the result of the respective RedPath’s LS as determined by the BURL algorithm. Once the LSs of the OrangePath are determined, their respective curve segment connecting any two LSs can be determined.
TABLE 25. AN ORANGEPath ALGORITHM. THE SECOND RAINBOW ALGORITHM.

<table>
<thead>
<tr>
<th>Input: A convex hull of interior disks’ center point (RedPath).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: A set of CSs enveloping the hull disks completely (OrangePath).</td>
</tr>
</tbody>
</table>

**Algorithm Steps Description:**
1. Determine the RedPath’s LS status with the BURL algorithm.
2. Enlarged all points of the input to their original radius plus the robot’s radius.
3. Compute CET LS to a pair of enlarged disks WRT equation (4), the status of each RedPath’s LS, and the end points of each LS.
4. Apply the index set of the input LS to the output LS.
5. Compute CS between any two vertices on the same enlarged disk as computed in step 2.

4.3.1 Relative Position of Line Segment Algorithm, the BURL’s algorithm

The BURL algorithm involves four steps as shown in Table 26. The algorithm begins by determining the minimum rectangular box that enclosed the convex hull or the RedPath. Figure 61 shows the input and some of the output of the BURL algorithm. In Figure 61, only three LSs with their vertical line tests are shown and they are LSs LS(5,6), LS(5,11), and LS(11,15). The vertical line test is drawn vertically through the midpoint of the LS as dictated by the second step of the algorithm.

<table>
<thead>
<tr>
<th>TABLE 26. A RELATIVE CURVE SEGMENT TEST (BURL) ALGORITHM.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BURL (Bottom, upper, right, left) Algorithm</td>
</tr>
<tr>
<td>Input: A closed path.</td>
</tr>
<tr>
<td>Output: Relative status of each CS in a closed path.</td>
</tr>
<tr>
<td>Algorithm Steps Description:</td>
</tr>
<tr>
<td>1. Find a MER for a closed path and determine VLT and HLT.</td>
</tr>
<tr>
<td>2. Find the midpoint of each CS.</td>
</tr>
<tr>
<td>3. Draw a VLT at the midpoint of each CS as shown in Figure 62 and then compute $L_0$ and $L_T$. Determine the relationship between $L_0$ and $L_T$. Get the CS status as either “upper” or “bottom”. If $L_0$ is not found or the slope of the CS is undefined, go to step 4. Else return status of the CS.</td>
</tr>
<tr>
<td>4. Draw a HLT at the midpoint of each CS then compute $L_0$ and $L_T$. Determine the relationship between $L_0$ and $L_T$. Get the CS status as either “left” or “right”. Return status of the CS.</td>
</tr>
</tbody>
</table>
The vertical line test yields two intersection points for consideration, $L_0$ and $L_T$. Since our problem is 2 dimensional, the coordinates of $L_0$ and $L_T$ are $(x_o, y_o)$ and $(x_T, y_T)$ respectively. The relationship between $y_o$ and $y_T$ determines whether the LS is an upper or bottom LS. $L_T$ is the midpoint of the LS that we want to know its status (e.g. Bottom or upper). Similarly, a horizontal line test allows us to determine whether the LS is a left or right LS within the convex hull. Our example problem does not require a horizontal line test. However its implementation is straightforward. The corresponding result of a horizontal line test, if compared to the vertical line test as shown in Figure 62 (b), would replace the test coordinate of $(y_T < y_o)$ with $(x_T < x_o)$. 

![Figure 61. The BURL algorithm on the RedPath](image)

![Figure 62. Example of the BURL algorithm](image)
Note that the minimum rectangle that enveloped the convex hull in Fig. 62 is drawn larger than it should be for readability. The actual size is slightly smaller as it left-side and right-side should have the x-coordinate of \( \min(\{r_i_x\}) \) and \( \max(\{r_i_x\}) \) respectively. Correspondingly, the upside and the downside of the rectangle should have the y-coordinate of \( \max(\{r_i_y\}) \) and \( \min(\{r_i_y\}) \) respectively.

The OrangePath algorithm establishes the relationship between any connected Redpath vertices, \( i \) and \( j \), as \( LS(i,j,r) \) with the algebraic equation shown in equation (4.1).

\[
y(i,j,r) = y(j,i,r) = m(i,j,r)x + b(i,j,r), \quad \text{where} \quad m(i,j,r) = m(j,i,r) = \frac{y_i - y_j}{x_i - x_j}, \quad \text{and} \quad b(i,j,r) = y_i - m(i,j,r)x_i.
\]  

Equation (4.1) and the BURL algorithm provide a solution to equation (4.2). Each vertex found by equation (4.2) is processed further to ensure that they form LS that are external tangent to both enlarged disks. It follows from equation (4.1) that the exterior point to an enlarged disk \( i \) for external tangent with enlarged disk \( j \) is on the line that is perpendicular to the red line.
connecting disks i and j at their centers. The equation is given by

\[ y(i, j, o) = m(i, j, o)x + b(i, j, o) \]

where

\[ m(i, j, o) = m(j, i, o) = \frac{-1}{m(i, j, r)} \quad \text{and} \quad b(i, j, o) = y_i - m(i, j, o)x_i. \]

(4.2)

4.3.2 Input of the OrangePath Algorithm

Input to this module is the convex hull vertices index set and the convex hull LS index set. For our example, the convex hull vertices index set is \( V_{rl} \) where \( rl \subset \{5, 6, 10, 11, 12, 15\} \). The convex hull LS index set in clockwise order is \( \text{LS}(V_{r5}, V_{r11}), \text{LS}(V_{r11}, V_{r15}), \text{LS}(V_{r15}, V_{r12}), \text{LS}(V_{r12}, V_{r10}), \text{LS}(V_{r10}, V_{r6}), \) and \( \text{LS}(V_{r6}, V_{r5}) \).

4.3.3 Output of the OrangePath Algorithm

The output generated by this module is the external tangent path that wraps all enlarged interior disks whose centers are the hull vertices. The OrangePath algorithm takes \( h \) number of vertices as input and then outputs \( 2h \) number of new vertices which are related to the input vertices. The OrangePath Algorithm also takes the RedPath’s LSs as input and then outputs disconnected LSs which are linked by the curve of the enlarged disk in which the output vertices are on. However they may undesirably intersect under rare circumstances which cause discontinuity. Examples of this kind of discontinuity can be seen on disk 11 and disk 12. This kind of discontinuity problem will be removed by the subsequent algorithm, the BluePath algorithm.

The output vertices generated by this module are \( V_{ol} \) with \( ol \subset \{5, 6, 10, 11, 12, 15\} \) and \( V_{or} \) with the same set order which is corresponding to the input vertices. Note that \( V_{ol}(1) \) is not necessarily equal to \( V_{or}(1) \). The OrangePath index set is important because it allows the data
structure to keep track of the OrangePath properly. The OrangePath LSs index set in clockwise order is $LS(V_{or5}, V_{ol11})$, $LS(V_{or11}, V_{ol15})$, $LS(V_{or15}, V_{or12})$, $LS(V_{ol12}, V_{or10})$, $LS(V_{ol10}, V_{or6})$, and $LS(V_{ol6}, V_{ol5})$. Also in clockwise order for the curve segments that connect the two LSs of the OrangePath are $CS(V_{ol11}, V_{or11})$, $CS(V_{ol15}, V_{or15})$, $CS(V_{or12}, V_{ol12})$, $CS(V_{or10}, V_{ol10})$, $CS(V_{or6}, V_{ol6})$, and $CS(V_{ol5}, V_{or5})$. $LS(V_{or5}, V_{ol11})$ is a LS connecting the right orange vertex of disk 5 and the left orange vertex of disk 11. $CS(V_{11}, V_{11})$ is a curve segment connecting the left orange vertex of disk 11 and the right orange vertex of disk 11. Figure 63 shows the relationship among OrangePath’s vertices, LSs, and curve segments. It also shows the relationship between the RedPath and the OrangePath. The OrangePath’s vertices are shown in small red dots which are the endpoints of the OrangePath’s LSs which are shown in thick orange LSs. The OrangePath’s curve segments are shown in yellow which are the arc of the disk whose center is on the RedPath.

Figure 63. An example output of the OrangePath algorithm
4.4 The YellowPath Algorithm

The input to this module is the vertices, LSs, and curve segments of the OrangePath and all disks in the target region. Every LS of the OrangePath is tested for collision with each and every disk that is in the target region except for disks whose center is the RedPath’s vertices because they have been addressed already. The reason is that we already know that the OrangePath goes around the disk that has a center on the RedPath so there is no collision. If a LS is tested positive for collision, a correction is prevented with either the external tangent or internal tangent collision correction algorithm depending on the status of the LS and the status of the disk involved. The status of the LS is either bottom, upper, right, or left. The status of the disk is either interior or exterior.

![Diagram](image)

Figure 64. An example input and output of the YellowPath algorithm

Note that the free space to be observed is painted gray in the output of the target region because it is very difficult to see the yellow LSs on the white background. The gray has no significant other than for readability and it is equal to white color in this case. The YellowPath algorithm follows two steps as listed in Table 27. An example of a LS detected to have collision with disk 7 is illustrated in Figure 64. For convenience, the LS that causes collision is drawn in dash black, see the right of Figure 64. The replacement LSs are shown in yellow lines with their endpoints in red.
### YellowPath Algorithm

**Input:** OrangePath and all disks in the TR.

**Output:** The YellowPath.

**Algorithm Steps Description:**

1. Check if any input LS intersects any enlarged disk. If a LS is not intersecting any enlarged disk, promote the input LS to yellow LS. Else find a pair of LSs to replace the LS that causes collision based on the input LS and the enlarged disk involved in collision. Ensure the new LSs are CIT/CET to the two enlarged disks that they are connecting.

2. Update all vertices and CS data structure of the YellowPath.

### 4.4.1 Collision Avoidance Algorithms, external tangent’s and internal tangent’s

The collision avoidance algorithm to transform the OrangePath LS into the YellowPath LSs has two states which result in eight different possibilities. The two different states are the status of the LS and the status of the disk. The status of the LSs can be “bottom”, “under,” “right,” or “left.” The status of the disk can be “interior” or “exterior.”

Figure 65 illustrates 4 different types of collision outcomes. We do not consider collision with the left or right LSs to conserve space. \( L_0(i,j) \) is the OrangePath’s LS in compact form with \( i \) and \( j \) represent the endpoint vertices. Disk(k) is the disk k with it radius enlarged by the robot’s radius while the robot itself is reduced to point.
4.4.2 **Input of the YellowPath Algorithm**

The input to this module is the vertices, LSs, and curve segments of the OrangePath and all disks in the target region. See the output section of the OrangePath algorithm for complete details. The only disks that are not tested for collision with the LS of the OrangePath are the disks whose center forms the RedPath.

4.4.3 **Output of the YellowPath Algorithm**

The YellowPath algorithm follows two steps as listed in Table 27. An example of a LS detected to have collision with disk 7 and is repaired is illustrated in Figure 67.
Two LSs replace one LS involves in collision. This results in four vertices replacing two vertices due to collision.

Figure 66. An example output of the YellowPath algorithm

Figure 67. An example output of the YellowPath algorithm

Figure 66 illustrates an example of that required internal tangent collision correction. Figure 67 illustrates and example of two collisions that required both internal tangent
collision correction and external tangent collision correction. In Figure 67, disk 9 is enlarged so that a scenario that required external tangent collision is required. Note that the output of the Yellow path may contain discontinuity. In both examples of Figure 66 and Figure 67, the discontinuity occurs at connecting curve segment of disk 11. The discontinuity problem is solved by the BluePath algorithm.

4.5 The GreenPath Algorithm

The GreenPath algorithm begins by checking that all RTs that the YellowPath crosses are observable from the coverage path. The coverage path at this stage is the Yellow coverage path. This is where an exact partition of the target region is required for observability analysis. There are two types of exact partitions required for the GreenPath algorithm. The first exact partition is done by computing the Delaunay triangulation of the target region which results in mutually disjoint RTs which together form the target region. Then each RT that the YellowPath crosses, the visible polygon is computed. If the curve segment of the YellowPath crosses a visible polygon of the RT, then the RT is observable as we have seen from Section 3.6. Otherwise a second type of exact partition is required to process the YellowPath’s LS for possible promotion to GreenPath’s LS. The second type of exact partition requires to observe the RT along the coverage path is called the Curve Segment Intersecting RT (CSIRT) algorithm. The GreenPath algorithm only checks for the observability of the RT that the YellowPath crosses. For RT that is completely encapsulated by the YellowPath, the LosPoRT algorithm will be employed along with the result of Lemma 2. This kind of problem will be solved in Section 4.7. For the RT that the YellowPath crosses, but cannot be observed through visible polygon or CSIRT algorithm, the visible path modification is required to correct the observability problem in the interest of complete coverage of the target region. Before showing the required algorithms for the
GreenPath, let us first introduce the input and output relationship of the GreenPath algorithm in the Figure 68.

![Diagram of GreenPath algorithm](image)

**Figure 68. An example input and output of the GreenPath algorithm**

The GreenPath algorithm presented in Figure 68 is detailed in Table 28 with six steps. The required inputs and the resulting outputs are briefly described in Figure 68. The most important step is to compute the Delaunay triangulation as presented in Figure 69. The result of the Delaunay triangulation is taken into consideration with the YellowPath’s curve segments to see how the GreenPath’s curve segment can be generated. Step 2 of the GreenPath’s algorithm required our result done in Chapter 3.

**TABLE 28. THE GREENPATH ALGORITHM.**

<table>
<thead>
<tr>
<th>Input:</th>
<th>The YellowPath, the TR, and all disks in the TR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>A set of CSs forming a closed path. This new closed path can observe all RTs that it crosses (crossing RTs).</td>
</tr>
</tbody>
</table>

**Algorithm Steps Description:**

1. Perform DT[16] on the TR to find RT. Compute each RT’s VP.
2. For each RT that the YellowPath crosses, add it to the crossing RT list.
3. For each crossing RT, check if the CS crosses it VP and if so promote the CS to green CS.
4. Else check if the RT and the CS can be partitioned by CSIRT algorithm and promote this CS to a green CS.
5. Else check if the RT and the CS can be partitioned by LosPoRT 3 and promote this CS to a green CS. Otherwise demote the CS to a black CS.
6. If the CS is a black CS, perform path modification through the VCPM algorithm and then pass the new CS(s) found to step 3. Else repeat step 3 with the new CS. If the current CS is the last CS in the list, end the program.
Figure 69 illustrates the transition from the YellowPath to the GreenPath. This figure is the output of step 1 of the GreenPath algorithm. Table 29 records the relationship between the YellowPath’s LS and the RT. IRT represents intersect RT which means that LS $L(5,11,y)$ is
intersecting RT 3 when true. L(5,11,y) represents the YellowPath’s LS from disk 5 to disk 11. IVP stand for LS intersecting visible polygon. For example if L(5,11,y)’s IVP is true, it is mean that all RTs that L(5,11,y) intersect also intersect all RTs’ visible polygons. This is when the YellowPath’s LS can be promoted to the GreenPath’s LS. An example of when the IVP is false is L(6,5,y). L(6,5,y) cannot be promoted to Green LS because it intersects RT 1, RT 2, and RT 4, but it only intersect VP 1 and VP 4. VP 2 is not intersected. As a result L(6,5,y) has to be repaired by a Visible Circle Path Modification algorithm as will be seen later in this Section (See Figure 70). Our focus is on the Curve Segment Intersecting RT (CSIRT) for now. CSIRT is relevant to RT 5, RT 8, RT 16, and RT 13. It is for the case of promoting the YellowPath LS to the GreenPath LS when the LS is intersecting the RT, but not the RT’s visible polygon. Figure 70 illustrates some of the output of the GreenPath algorithm.

![Figure 70. RTs 5, 13, and 16 with CSIRT](image)
4.5.1 Visibility of Crossing Coverage Path Algorithms, CSIRT’s

CSIRT algorithm may be required to observe the RT that the coverage path crosses. Example RTs that need the CSIRT algorithm are RTs 5, 8, 13, and 16. CSIRT algorithm follows 7 steps as shown in Table 30. The inputs to the CSIRT algorithm are the curve segment that crosses the interested RT and the RT itself. The outputs of the CSIRT algorithm are the mutually disjoined sub regions and the observer points that jointly observe all of the RT.

An example result of the CSIRT algorithm is presented in figure 71. The visibility LSs of disk 9, \( l_1 \) and \( l_2 \), visibility LSs of disk 11, \( l_3 \) and \( l_4 \), and visibility LSs of disk 14, \( l_5 \) and \( l_6 \). Step 2 of the algorithm determines that disk 11 is the visible disk because it has both visibility LSs, \( l_3 \) and \( l_4 \), intersect with the Yellow coverage path within the RT. Step 3 of the algorithm determines that \( l_3 \) and \( l_4 \) are intersected with each other within the RT and \( A_4 \) is nonzero.

**TABLE 30. CURVE SEGMENT INTERSECTING REGULAR TRIANGULATION ALGORITHM.**

| Input: An RT and CS pair (eg. RT16 and CS(RT16)). |
| Output: Observers \( O_1, O_2, \) and \( O_3 \) on the input CS and mutually disjoined SRs which together form an RT. |
| Algorithm Steps Description: |
| 1. Compute the VLSs \( l_i, i \in 1,2, ...,6 \) as shown in Figure 71(a). |
| 2. Find a VD. Return false if not found. |
| 3. Check whether the two VLSs of every disk intersect with each other within the RT. |
| 4. Determine SRs \( A_1, A_2, A_3, \) and \( A_4 \) based on step 3’s result. |
| 5. In CW direction, find the first VLS of the VD that intersects with CS within the RT and designate the intersection point as \( O_1 \). |
| 6. Repeat step 5 to find the second VLS that intersects with CS within the RT and designate the intersection point as \( O_2 \). |
| 7. Compute the average x-value of \( O_1 \) and \( O_2 \) then find the corresponding y-value on the CS. This is \( O_3 \). Return true. |
The mutually disjoined sub regions $A_1, A_2, A_3,$ and $A_4$ are determined. The result of step 3 allows steps 4, 5, and 6 to determine the three observers required for complete coverage of the RT.

Figure 71. A typical RT through CSIRT partition algorithm

Of the three disks in RT 16, only disk 11 has both visibility LSs intersecting with the yellow curve segment. Note that visibility LS is perpendicular to the RT’s boundary LS that the EP is originated. Now we are ready to present Lemma 1.

Figure 72. CSIRT algorithm analysis of observer points placement
Lemma 1: The CSIRT algorithm provides a set of observers, \( O_1, O_2, \) and \( O_3 \), on a CS that crosses an RT, but does not cross its VP, that jointly observe the whole RT if all of the following conditions hold:

1) The CSIRT algorithm returns true with an RT and the CS pair as inputs.
2) \( O_1 \) is in the VP of \( A_1 \). Likewise \( O_2 \) is in the VP of \( A_3 \).
3) \( O_1, O_2, \) and \( O_3 \) have the following properties:
   i. A center slicing \( O_1 I_1 \) intersects with a secant slicing \( O_3 I_1 \) at \( I_1 \).
   ii. A center slicing \( O_2 I_2 \) intersects with a secant slicing \( O_3 I_2 \) at \( I_2 \).

Proof: Condition 1 of the lemma guarantees that the combination of the input CS and an RT pair, partitions an RT into SRs with all three observers and all four SRs with properties needed for visibility analysis. Condition 2 of the lemma guarantees that \( O_1 \) has full coverage of \( A_1 \), and \( O_2 \) has full coverage of \( A_3 \). Condition 3 guarantees that \( O_1, O_2, \) and \( O_3 \) can jointly observe \( A_2 \). \( A_4 \), if greater than zero, is always seen by either \( O_1, O_2, \) or \( O_3 \) because it is not blocked. □

Figure 73. Lemma 1’s illustration
Condition 1 of lemma 1 returns observer points $O_1$, $O_2$, and $O_3$. Condition 3 is shown with points $I_1$ and $I_2$ connected to observer points $O_1$ and $O_2$ respectively and points $I_1$ and $I_2$ are connected to point $O_3$.

To reduce the number of observers for the target region, we investigate if it is possible to reduce the number of observers provided by the CSIRT algorithm. The result of Theorem 1 allows us to eliminate observer number 3. By reducing the number of the observer when possible, we are able to save computational power as well as processing power.

**Theorem 1:** A RT that can be partitioned by the CSIRT algorithm and is visible by a set of observer points can be reduced to just two observer points if observer point $O_1$ and observer point $O_2$ can both observe a common point on the visible disk.

Figure 74. Theorem 1 illustration
Figure 79 illustrates observer point $O_1$ is in visible polygon of $A_1$, the “light blue” region. Hint, all of $A_1$ is seen. Observer $O_2$ is in visible polygon of $A_3$, the orange region. Hint, all of $A_3$ is seen. Observer point $O_3$ is removed in favor of theorem 1.
Proof: Point $O_1$ can observe $A_1$ because it is on the visible polygon of $A_1$. Likewise, point $O_2$ can observe $A_3$ for the same reason. The fact that point $O_1$ and point $O_2$ can both observe a common point on the visible disk means that together they can observe $A_2$. Since $A_4$ is not blocked, either point $O_1$ or point $O_2$ can observe $A_4$.

4.5.2 Visible circle path modification

When CSIRT algorithm cannot be used to promote the Yellow curve segment to Green curve segment, the Visible Circle Path Modification (VCPM) provides the solution. Figure 75 illustrates how the Yellow curve segment, shown in black due to it inability to get a promotion, is replace by the Green curve segment. VCPM algorithm is illustrates in Table 31 with 4 steps.

![Figure 75. Visible path modification with visible circle concept](image-url)
Table 31. **Visible Circle Path Modification Algorithm.**

| **Input:** | A LS of the YellowPath and all RTs that it crosses. |
| **Output:** | A visible circle and two new LSs with endpoints that meet the external tangent requirement on all disks that the new LSs connect. |

**Algorithm Steps Description:**

1. Determine whether the LS to be replaced is a bottom, upper, right, or left LS with respect to the YellowPath with the BURL algorithm.
2. Determine the location of the RT and its visible polygon with respect to the LS to be replaced.
3. Compute the center of the visible circle so that exactly one point of the visible circle intersects the interested RT’s visible polygon at the vertex with the minimum distance from the end vertices of the LS to be replaced.
4. Compute the two new LSs that will replace the input LS. The two new LSs have endpoints that meet the external tangent requirement on all disks that the new LSs connect.

Step 1 of VCPM algorithm determines that the curve segment shown in Figure 75 to be repaired is a lower left curve segment. This allows the VCPM algorithm to know that the replacement curve is to the left of the original curve segment. Step 2 determines which RT is to be taken into account to find the crossing point of the new curve segment. The metric to be used is the distance from the vertices of the VP of RT 2 to both end points of the curve segment to be replaced.

### 4.5.3 Input of the GreenPath Algorithm

All disks in the target region are required to partition the TR into several disjoint simple cells known as the regular triangulations (RTs). The Delaunay Triangulation technique is used to partition the target region into several RTs. There are \( n_{Lines} = h + n_{CollisionLSs} \) and \( n_{Vertices} = 2h + 2n_{Collision} \) vertices along with N number of disks in the target region.

### 4.5.4 Output of the GreenPath Algorithm

The output to the GreenPath algorithm of our example problem is presented in Figure 76 along with the output of the previous phases such as the RedPath and the YellowPath. At this stage of the Rainbow algorithm, we know that the Green coverage path is collision free with the known static disks and can also observe the RTs that it crosses. Note that the Green path may
contain discontinuity which is to be repaired by the BluePath algorithm. One obvious result of discontinuity is the curve segment of the GreenPath around disk 11. Figure 77 illustrates just the Green coverage path obtained.

Figure 76. Visible path modification with visible circle concept

Figure 77. An example output of the GreenPath algorithm
4.6 The BluePath Algorithm

The BluePath algorithm checks, and if necessary modifies, the curve where any two LSs meet to ensure differentiability. Differentiability correction occurs when the enlarged disk $i$, where the two LSs are joined together by the arc of the enlarged disk $i$, is smaller than the user’s specified threshold value, see Figure 78. The BluePath algorithm repairs differentiability problems by enlarging the disk $i$ by the user specified value since an arc of an enlarged disk is known to be $S=(R_i + R_r)\theta$, where $\theta$ is the central angle.

![Figure 78. An example input and output of the BluePath algorithm](image)

The blue path algorithm follows 15 steps as illustrated in Table 32.
TABLE 32. THE BLUEPATH ALGORITHM. THE FIFTH RAINBOW ALGORITHM.

<table>
<thead>
<tr>
<th>BluePath Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> The GreenPath and all disks in the TR.</td>
</tr>
<tr>
<td><strong>Output:</strong> A first order differentiable CP, BluePath, and crossing RTs’ VCS and observers.</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. For ( (i = 1, i &lt; N_{\text{gcs}}, i + +) ), set ( ST(i) ) to false. Set temporary ( \text{CS}<em>{ti} = S</em>{gi}, L(V_{tti}, V_{tr(i-1)}) ) equal to ( L(V_{gli}, V_{gr(i-1)}) ), and ( L(V_{tri}, V_{tl(i+1)}) ) equal to ( L(V_{gri}, V_{gr(i+1)}) ).</td>
</tr>
<tr>
<td>2. For ( (i = 1, i &lt; N_{\text{gcs}}, i + +) ), perform steps 3-6:</td>
</tr>
<tr>
<td>3. Check that ( S_{ti} ) satisfies the user’s specified value. If so set ( ST(i) ) to true then go to step 6. Else set ( ST(i) ) to false.</td>
</tr>
<tr>
<td>4. If ( ST(i) ) is false, set ( ST(i - 1) ) and ( ST(i + 1) ) to false.</td>
</tr>
<tr>
<td>5. Enlarge ( R_{gi} ) to find ( S_{ti} ) that satisfies the user’s specified value. Compute and update ( L(V_{tti}, V_{tr(i-1)}) ) and ( L(V_{tri}, V_{tl(i+1)}) ). Set ( ST(i) ) to true.</td>
</tr>
<tr>
<td>6. Set ( i = i + 1 ).</td>
</tr>
<tr>
<td>7. Repeat steps 2-6 until all ( ST(i) ) are true and then promote ( S_{ti}, L(V_{tti}, V_{tr(i-1)}) ), and ( L(V_{tri}, V_{tl(i+1)}) ) to ( S_{bi}, L(V_{blt}, V_{br(i-1)}) ), and ( L(V_{bri}, V_{bl(i+1)}) ) respectively.</td>
</tr>
<tr>
<td>8. For ( (j = 1, j &lt; N_{\text{gcs}}, j + +) ) perform the following steps:</td>
</tr>
<tr>
<td>9. If ( S_{bj} ) and ( S_{gi} ) are not equal, perform the following:</td>
</tr>
<tr>
<td>10. Set ( \text{CS}<em>{bj} = \text{merge} \left( L(V</em>{bjj}, V_{br(j-1)}), S_{bj}, L(V_{brj}, V_{bl(j+1)}) \right) ). Check if ( \text{CS}_{bj} ) collide with any disk in the TR. If so, returns fail and exit the program. Else go to step 11.</td>
</tr>
<tr>
<td>11. Check if ( \text{CS}_{bj} ) cross all VPs of all crossing RTs. If so, go to step 14.</td>
</tr>
<tr>
<td>12. If any crossing RT’s VP is not crossed by ( \text{CS}<em>{bj} ) in step 10, check if ( \text{CS}</em>{bj} ) and the RT pair can be partitioned by the CSIRT algorithm. If yes, go to step 14. Else go to step 13.</td>
</tr>
<tr>
<td>13. Check if ( \text{CS}_{bj} ) and the RT pair return true with LosPoRT3. If yes, go to step 14. If not returns fail and exit the program.</td>
</tr>
<tr>
<td>14. Set ( j = j + 1 ).</td>
</tr>
<tr>
<td>15. For all crossing RTs, find their VCS(s), CSIRT, and VCPM observers.</td>
</tr>
</tbody>
</table>

4.6.1 Differentiability Algorithm, Disk Enlargement’s

The BluePath algorithm, which is shown in Table 32, shows that disk enlargement to prevent discontinuity is represents by \( L = (R_{i} + R_{o}) \theta \). Figure 79 illustrates the enlargement of disk 11 to prevent discontinuity.

4.6.2 Input of the BluePath Algorithm

Input of the BluePath algorithm includes the GreenPath and all disks in the TR.
4.6.3 Output of the BluePath Algorithm

Since the robot encounters the enlarged disk at a tangent, its path is first order differentiable. The result of the BluePath algorithm is a first order differentiable coverage path. Figure 79 illustrates an example of our BluePath correct differentiable issue. Disk 11 is enlarged to smooth the coverage path.

Figure 79. An example output of the BluePath algorithm

4.7 The IndigoPath Algorithm

In this phase of the Rainbow algorithm, we already know that our coverage path has the following properties: collision free with known disks, observability of all RTs that the coverage path crossed, and first order differentiable. The question to ask at this point is “can the noncrossing RTs be observed from the coverage path?” The question can be answered by the
technique discovered in this section. However, before we delve any further, let us introduce the input and the output diagrams for clarity.

Figure 80. An example input and output of the IndigoPath algorithm

Figure 80 shows the input/output to the left/right of the Indigo algorithm. The figure above can be algorithmically described by the algorithm in Table 33 below.

<table>
<thead>
<tr>
<th>IndigoPath Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> The BluePath, all noncrossing RTs, all noncrossing SRs, and all interior disks.</td>
</tr>
<tr>
<td><strong>Output:</strong> Noncrossing SRs and their corresponding VCSs.</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. For each RT, perform LosPoRT1 algorithm to obtain 3 SRs.</td>
</tr>
<tr>
<td>2. Find VCSs of each SR with all blocking interior disks.</td>
</tr>
<tr>
<td>3. If no VCS can be found, partition the SR into smaller SR with LosPoRT2 algorithm and then repeat step 2 with the smaller SR.</td>
</tr>
</tbody>
</table>

In the IndigoPath algorithm, LosPoRT stands for Line of sight Partition of RT. There are two variants of LosPoRT algorithms. If the noncrossing RT is simple, the LosPoRT algorithm 1 can find the visible curve segment(s) for each of the three partitioned region, see Figure 81 and Figure 82. Otherwise LosPoRT algorithm 2 is needed to further partition the region that has no corresponding visible curve segments. Our example problem does not required LosPoRT 2.
LosPoRT algorithm 2, just like LosPoRT 1, result in a sub region with just one adjacent disk. It is very straightforward to understand how LosPoRT algorithm 2 work if you understand LosPoRT algorithm 1. For our example, all noncrossing RTs have corresponding visible curve segment(s) determine by LosPoRT algorithm 1.

4.7.1 Visibility of Non-Crossing RT, Region to Visible Curve Segment Mapping

LosPoRT algorithm 1 is illustrated in Table 34 which involves 8 steps. The first step find the red, blue, and gray vertices which are used by steps 2 to 5 to find the three sub regions \( A_1, A_2, \) and \( A_3 \). The geometrical relationships of the red, blue, and gray vertices are presented in Figure 82. The fixed points of red, blue, and gray vertices allow us to compute the sub regions \( A_1, A_2, \) and \( A_3 \). The sub regions \( A_1, A_2, \) and \( A_3 \) are process by Region to Visible Curve Segment Mapping algorithm to find the corresponding visible curve segment(s) on the coverage path. If any sub region does not have a corresponding visible curve segment, then it is a candidate sub region for further partition which require LosPoRT algorithm 2. In short, all sub regions found by LosPoRT algorithm 1 or algorithm 2 are process by Region to Visible Curve Segment Mapping algorithm to find the visible curve segment(s) on the coverage path. The Region to Visible Curve Segment mapping is presented in Section 4.7.1 for a single RT and in Section 4.7.2 for multiple RTs.

For simplicity, illustration of LosPoRT algorithm 1 is done as follow with a single RT and a simple coverage path that contained the RT. Figure 83(a) shows a typical RT enveloped by a circular coverage path which is much simpler than the typical coverage path that are input into this phase of the Rainbow algorithm, see Figure 93 for the coverage path that encapsulate multiple RTs. A single RT example surrounded by the circular coverage
path is the simplest illustration that allows us to study the observability of the noncrossing RT.

Table 34. LOSPoRT 1 ALGORITHM.

<table>
<thead>
<tr>
<th>Line of sight Partition of RT Algorithm 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> RT of 3 disks (e.g., disks i, j, and l).</td>
</tr>
<tr>
<td><strong>Output:</strong> SRs A₁, A₂, and A₃. Each SR has only one adjacent disk.</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. Set w, o, and q as disks. Put disks i, j, k in the list, list LD.</td>
</tr>
<tr>
<td>2. Determine the longest of three edges of the RT. Compute the midpoint of the longest edge and name it as point R.</td>
</tr>
<tr>
<td>3. Name the disk opposite to R as disk o. Remove disk i, j, or k that is equal to disk o from LD. Set q = ( \text{min}(LD) ). Set w = ( \text{max}(LD) ).</td>
</tr>
<tr>
<td>4. Compute the midpoints of the other two edges. Name the midpoints as G and B so that CW order of B, R, and G are maintained for SR adjacent to disk o.</td>
</tr>
<tr>
<td>5. Connect points B, R, and G to form ( \overline{BR} ) and ( \overline{GR} ).</td>
</tr>
<tr>
<td>6. Find ( A₁ ) which is adjacent to disk q and it is bounded by ( CS(C_q, Y_q), \overline{Y_qR}, \overline{RB}, \text{ and } \overline{BC_q}. )</td>
</tr>
<tr>
<td>7. Find ( A₂ ) which is adjacent to ( A₁ ) and bounded by ( CS(C_o, Y_o), \overline{Y_oB}, \overline{BR}, \overline{RG}, \text{ and } \overline{GC_o}. )</td>
</tr>
<tr>
<td>8. Find ( A₃ ) which is adjacent to ( A₂ ) and bounded by ( CS(C_w, Y_w), \overline{Y_wG}, \overline{GR}, \text{ and } \overline{RC_w}. )</td>
</tr>
</tbody>
</table>

Table 35. LINE OF SIGHT PARTITION OF REGULAR TRIANGULATION ALGORITHM 2.

<table>
<thead>
<tr>
<th>Line of sight Partition of RT Algorithm 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> SRs A₁, A₂, and A₃ partitioned by LosPoRT algorithm.</td>
</tr>
<tr>
<td><strong>Output:</strong> SRs ( A_i ) where ( i \in 1,2,3 ) and M is a user-specified number, the number of partition of SR A₁, A₂, and A₃. Each SR is a LosPoRT SR.</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. If a region to be partitioned is ( A₂ ), and ( M = 2 ), then partition ( A₂ ) by drawing a line from the center of ( A₂ )’s adjacent disk to the R vertex.</td>
</tr>
<tr>
<td>2. Else if a region to be partitioned is a SR of ( A₂ ), then partition the SR by drawing a line from the center of the ( A₂ )’s adjacent disk to the midpoint of the edge that is opposite to the adjacent disk.</td>
</tr>
<tr>
<td>3. Else if a region to be partitioned is ( A₁, A₃ ), ( A₁ )’s or ( A₃ )’s SRs then partition the SR by drawing a line from the center of ( A₁ )’s or ( A₃ )’s adjacent disk to the midpoint of the edge that is opposite to the adjacent disk.</td>
</tr>
</tbody>
</table>

Figure 81 (b) show that in the absent of any disk, region A is visible from every point on the coverage path. The absent of any obstacle is the only scenario where the region is visible.
from every point, the visible curve segment is a closed set of all points in the absence of obstacles.

Figure 81. A given $T(A)$, with and without disks, to be observed from a coverage path $P(A)$.

<table>
<thead>
<tr>
<th>(a) $T(A)$ forms by disks $i, j, and k.$</th>
<th>(b) In the absence of any disk, Region $A$ is visible from every point on $P(A)$.</th>
</tr>
</thead>
</table>

Figure 82 (a) shows how the RT is partitioned by LosPoRT algorithm 1. There are three sub regions, $A_1, A_2, and A_3$. $A_2$, if needed, may be further partitioned into $\frac{1}{2}A$, $\frac{2}{3}A$ and $\frac{3}{4}A$. Likewise, $A_3$ may be partitioned into $\frac{1}{2}A$ and $\frac{3}{4}A$. Again, $A_2$ and $A_3$ may be partitioned into much smaller sub regions if necessary. It is beyond the scope of this dissertation to cover all possible cases. In Figure 82 (b), $(x_{i,j}, y_{i,j})$ and $(x_{i,j}, y_{j,j})$ are the coordinates of yellow EPs on disks $i$ and $j$ respectively.
Before we introduce Lemma 2, which guaranteed complete coverage of a sub region of $T(A)$, let us first provide the motivation of the two Line of sight Partition of RT (LosPoRT) algorithms which partition $T(A)$ or sub region in $T(A)$ into observable regions with just one disk included in each partitioned region. This idea is motivated by the fact that the union of two sets is at most the size of the smaller of the two sets. The first LosPoRT algorithm partitions $T(A)$ into three sub regions, and it is shown in Figure 83 (a). Note that the symbols for color represent by B, C, G, R, and Y are for blue, cyan, gray, red, and yellow respectively throughout the dissertation.

Now we are ready to introduce Lemma 2.

**Lemma 2:** The subset $A_2, A_2 \subseteq A = \bigcup_{i=1}^{3} A_i$, bounded by CW CSs, $BR, GR, GC_j, CS(C_j,Y_j)$, and $Y_jB$, as shown in Figure 83(a) can be observed from $P(i,j,k)$ with a single observer if there exists at least one VCS on $P(i,j,k)$ where an observer can be selected to see all vertices of $A_2$. 

---

---

**Figure 82.** LosPoRT’s example and its vertices’ relation.
Before proving Lemma 2, we need to note some of the observation necessary for the proof’s formalism. The goal of the LosPoRT algorithm is to have just one disk for any region or sub region to be covered from the coverage path. Figure 83(b) represented the region $A_2$. To see how $A_2$ can be observed, we begin by removing the only disk in the region, disk $j$, which is illustrated in Figure 84. The scenario presented in Figure 84 is the only time where the “visibility curve” contains all points on the curve.

Figure 83. A partitioned region and its typical coverage path

Figure 84. Inclusive VCS in the absent of Disk.
In the absence of disk, any point on the coverage path P can see all region of $A_2$.

For a region with one adjacent disk, the first step in visibility analysis is to isolate the blind curve segment from the visible curve segment. The blind curve segment can be removed by finding the half plane as illustrated in Figure 85. The two halfplanes are originate by the two vertices on the disk.

Figure 85. Highlight the 4 cases of observability of two vertices on disk j

The final curve segment where an observer can see both vertices C and Y in the presence of disk j is the green curve represented by $g_j(Y, C)$. Note that the halfplane represented by $h_c$ and $h_y$ in red are always drawn through the vertices that are on the disk. The halfplane LSs spawn by the vertices are perpendicular to their adjacent LS spawn by the same vertices. For example the halfplane $h_y$ and LS $BY$ are perpendicular.
Figure 86 (a). BCS due to Disk i

Figure 86 highlights the observability of vertices C and Y on disk j with inclusion of disk I using the concept of visible cone, the visible cone originate at the vertex C and tangent to the disk i ends at two points on the coverage path P, \( C_{il} \) and \( C_{ir} \). The final visible curve cannot be computed yet because not all points have been considered. The pattern will become clear as the visible cones of the vertices Y, B, R, and G are included.

Figure 86b. Highlight the visibility of vertices C and Y on disk f with the inclusion of disk i
Visible cones originate at the vertices C and Y ends at two points on the coverage path P, $C_{il}$ and $C_{ir}$ for visible cone C and $Y_{il}$ and $Y_{ir}$ for visible cone Y respectively. $g_{ji}(Y,C)$ represents the curve segment which points Y and C can be observed in the presence of disks $j$ and $i$. Note that the ordering is important because it denotes the area $A_2$ which is adjacent to disk $j$ and disk $i$ is the blocker. The long form of $g_{ji}(Y,C)$ is $g_{adjacent\_disk\_j\_blocker\_disk\_i}(Y,C)$.

Figure 87. Observability of vertices B, C, G, R, and Y on disk $j$ with the inclusion of disk $i$.

Visible cones originate at the vertices B, R, G, C, and Y ends at two points on the path P for each of the vertex for a total of ten end points (see Figure 87 (b)). With the result of the two half planes and blind curve segments known as in Figure 87 (b), the final visible curve segment can be determined.
Visible cones originate at the vertices B, R, G, C, and Y and end at two points on the coverage path P. The green curve segment shown in Figure 88(a) is the open-ended curve segment where any point on it can see $A_2$ completely.

Figure 89. Observability of $A_2$ with the inclusions of disk $k$. 

(a) Coverage path, disks j and k, and region $A_2$. 

(b) Visibility cone of vertex G on disk k.
Now let observe the visible curve segment of the region $A_2$ with disk $k$ as the blocking disk. The visible cone originates at the vertex $G$ ends at two points on the path $P$, $G_{kl}$ and $G_{kr}$. The final curve segment where a point can be selected to observe all of $A_2$ is $g_{jik}(A_2)$ or equivalently $g_{jik}(B, C, G, R, Y)$ which has a length of $C_{j2} = \int_{l_{max}}^{K_{min}} ds$. Now we are ready to prove Lemma 2.

![Diagram](image.png)

(a) All vertices of region $A_2$ and their visible cones with respect to disk $k$.  
(b) Enlargement of disk $k$’s “blind curve”.

Figure 90. Observability of $A_2$ with the introduction of disk $k$

![Diagram](image.png)

(a) Finding “blind curves” of all three disks.  
(b) Finding “blind curves” of $A_2$.

Figure 91. Observability of $A_2$ with the inclusion of disks $i, j, and k$
4.7.2 LosPoRT with multiple RTs

We have seen LosPoRT with just one RT and it seems easy because each RT has only three disks. Of the three disks, only two disks are the blocking disks in a single RT problem. How do we deal with observing a partitioned region with many disks as the blocking disks? The same concept of region to visible curve segment mapping applied to multiple disks as to just two

Proof: Any region partitioned by the LosPoRT algorithm has at most 5 vertices. Since the region is part of an RT that is formed by 3 disks, there exist at least three potentially VCSs on $P(i,j,k)$, corresponding to the number of disks, from which an observer can be chosen that can observe the whole set of $A_l$ where $l = 1,2,3$. The number of VCSs can also be directly obtained from drawing a visible cone from all vertices of subset $A_l$ to enclose each of the disks inside the CP. The rays that originate from each vertex are tangent to each side of each disk. Any point on the VCS can observe $A_l$. 

Figure 92. Observability of A2 with the inclusion of disks $j$, $i$, and $k$. 

(a) Finding “visibility curve” of all three disks.
(b) Result of “visibility curve” of all three disks.
disks. The first step to do is to compute the two half plane of the two vertices on the adjacent disk. Recall that we consider the adjacent disk of the interested sub region, e.g. Sub region A_2. Figure 93 shows the sub region 7A_2 with disk 6 as the adjacent disk. The candidate visible curve segment for region 7A_2 can be found by finding the two half planes of the two vertices on disk 6, namely Y and C vertices.

Figure 93. Candidate visible region 7A2

The outer curve segment surrounding the green region which is a subset of the Indigo coverage path is the candidate visible curve segment. Note that the blue circle and the yellow circle in Figure 94 denote the end point of the candidate visible curve segment of region 7A_2. The candidate visible curve segment of region 7A_2 is abbreviated as CVCS(7A_2). The clockwise CVCS(7A_2) in curve segment format is CS(Blue point, Yellow point). The visible curve segment of region 7A_2, VCS(7A_2), is the subset of CVCS(7A_2).
Figure 94 shows how easy it is to find the candidate visible curve segment of region $7A_2$ since we know the Indigo coverage path and the two vertices on disk 6, the adjacent disk of region $7A_2$. The next step is to find the visible curve segment of region $7A_2$, $VCS(7A_2)$ which we know is the sub set of $CVCS(7A_2)$. The problem boil down to finding the blocking disks within the candidate visible curve segment which is also within the sub region enveloped by the Indigo coverage path.

Table 36. FIND THE BLOCKING DISK ALGORITHM

<table>
<thead>
<tr>
<th>Find the blocking disk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> LosPoRT’s sub region of interest and its visibility LSs, LS of Yellow vertex and Cyan vertex, and all interior obstacles. Initialize all arrays to zero: YellowVLS[], YellowVLSIntersect[], YellowResult[], CyanVLS[], CyanVLSIntersect[], CyanResult[], and DiskCVCS[].</td>
</tr>
<tr>
<td><strong>Output:</strong> Interior disks blocking the candidate visible curve segment (CVCS).</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. Determine if center of disk I is greater/less than Yellow LS’s equation, if so set YellowVLS[I] to 1. Else if disk I intersect with the Yellow LS’s equation, set YellowVLSIntersect[I] to 1.</td>
</tr>
<tr>
<td>2. Perform logical Oring, YellowResult[I] = (YellowVLS[I] OR YellowVLSIntersect[I]).</td>
</tr>
<tr>
<td>3. Determine if center of disk I is greater/less than CyanLS’s equation, if so set CyanVLS[I] to 1. Else if disk I intersect with the CyanLS’s equation, set CyanVLSIntersect[I] to 1.</td>
</tr>
<tr>
<td>4. Perform logical Oring, CyanResult[I] = (CyanVLS[I] OR CyanVLSIntersect[I]).</td>
</tr>
<tr>
<td>5. Perform logical Anding, DiskCVCS[I] = (CyanResult[I] AND YellowResult[I]).</td>
</tr>
</tbody>
</table>
Tables 37, 38, and 39 provide work out examples with finding the blocking disk with find blocking disk algorithm. Table 37 illustrates the finding of blocking disk for region $7A_{2}$ shown in Figure 94. Clearly seen in Figure 94, the Yellow visibility LS does not intersect interior disks 5, 6, 9, 10, 11, 12, and 15. The array YellowVLSIntersect[] at indices 5, 6, 9, 10, 11, 12, and 15 remain zero. But because the Yellow visibility LS intersect interior disk 8, the array YellowVLSIntersect[] at index 8 is set to 1. With regard to the interior disks being inside the interested region $7A_{2}$ view, we see interior disks 5, 8, 9, 11, 12, and 15 are meeting the algorithm’s criteria. Oring the two arrays as required by step 3 of the algorithm, we find that the blocking disks of the Yellow visibility LS in YellowResult[] array are disks 5, 8, 9, 11, 12, and 15. Repeating the same procedures for the Cyan visibility LS, we discover that it intersects interior disk 5, but not any other interior disk. The CyanVLSIntersect[] at index 5 is set to 1 while the array at all other indices remain zero. For the interior disks that are to the right of the Cyan visibility LS, we found out that disk 6 is the only interior disk that are not included. All other interior disks are included.

Table 37. REGION 7A2 AND THE RESULTING DISKS BLOCKING ITS CVCS

<table>
<thead>
<tr>
<th>Step</th>
<th>RT 7 SR A2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><strong>Interior Disk</strong></td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>1. YellowVLS[]</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2. YellowVLSIntersect[]</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><strong>YellowResult[]</strong></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Step 3</td>
<td><strong>Step 3</strong></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1. CyanVLS[]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2. CyanVLSIntersect[]</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><strong>CyanResult[]</strong></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Step 5</td>
<td><strong>DiskCVCS</strong></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The CyanVLS[] remains zero at index 6 while it is set to 1 for all other indices. As a result, the CyanResult[] array has value of 0 at index 6 and a value of 1 at indices 5, 8, 9, 10, 11, 12, and 15. Finally, the algorithm found disks 5, 8, 9, 11, 12, and 15 to be blocking disks for region $A_2$’s candidate visible curve segment. The determined blocking disks will be used to remove all blind curve segment from the CVCS to find the VCS from the coverage path for the interested region. For regions $14A_2$ and $12A_2$, the Find the blocking disk algorithm yields the results shown in Table 38 and Table 39 respectively. The candidate visible curve segment of Region $14A_2$ is block by disks 8, 9, 11, 12, and 15. The candidate visible curve segment of Region $12A_2$ is block by disks 10 and 12.

Table 38. REGION 14 A2 AND THE RESULTING DISKS BLOCKING ITS CVCS

<table>
<thead>
<tr>
<th>RT14 SR A2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interior Disk</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>Step 1</td>
<td>3. YellowVLS[]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4. YellowVLSIntersect[]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Step 2</td>
<td>YellowResult[]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Step 3</td>
<td>3. CyanVLS[]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4. CyanVLSIntersect[]</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Step 4</td>
<td>CyanResult[]</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Step 5</td>
<td>DiskCVCS</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 39. REGION 12A2 AND THE RESULTING DISKS BLOCKING ITS CVCS

<table>
<thead>
<tr>
<th>RT12 SR A2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interior Disk</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>Step 1</td>
<td>5. YellowVLS[]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6. YellowVLSIntersect[]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Step 2</td>
<td>YellowResult[]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Step 3</td>
<td>5. CyanVLS[]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6. CyanVLSIntersect[]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Step 4</td>
<td>CyanResult[]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Step 5</td>
<td>DiskCVCS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Once the blocking disks are found, we have to find the “blind curve segments” generated by the blocking disks using the concept of visible cone. The visible cone algorithm is presented.
in Table 40 with 4 steps. After all “blind curve segments” are found, then they can be removed from the candidate visible curve segment to find the final visible curve segment for the region of interested.

**Table 40. VISIBLE CONE ALGORITHM**

<table>
<thead>
<tr>
<th>Visible cone algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> vertex and disk.</td>
</tr>
<tr>
<td><strong>Output:</strong> the two tangent points, $T_{li}$ and $T_{lr}$, which allow us to find equation $E_{vTlx}$, $x$ is for l or r.</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. Draw a straight LS, $LS_{bi}$, from the interested vertex to the center of the interested blocking disk and call it point i.</td>
</tr>
<tr>
<td>2. Draw another straight LS, $LS_{bili}$, that is perpendicular to $LS_{bi}$ at point i so that $LS_{bili}$ intersect the target interior disk at two points. The two points are at diameter length of the target disk from each other.</td>
</tr>
<tr>
<td>3. Compute the two intersecting points in step 2 and then name the two points as $T_{li}$ and $T_{lr}$.</td>
</tr>
<tr>
<td>4. Find equation of straight lines from the interested vertex in step 1 to $T_{li}$ and to $T_{lr}$.</td>
</tr>
</tbody>
</table>

Figure 95. Visible cone of the Yellow vertex WRT disk 9 for region 7A2

Figure 95 shows the blind visible cone of the Yellow vertex of region 7A2 with respect to disk 9. We need to compute all blind visible curve segments for all vertices of region7A2. Since disks 5, 8, 9, 11, 12, and 15 are blocking the CVCS of region7A2, the running time to compute
the blind visible curve segments is linear assuming that the number of blocking disks is greater than the number of vertices in the interested region which is known to be at most 5. Points $T_{Yi9L}$ and $T_{Yi9R}$ in Figure 95 represented the two points in step 3 of the visible cone algorithm.

![Image](image.png)

Figure 96. CVCS($7A_2$) and blind CS of the Yellow vertex generated by disk 9

Figure 96 shows the candidate visible curve segment, CS($C7A_2$, $Y7A_2$), of region $7A_2$ which is generated by the Yellow and the Cyan vertices of region $7A_2$. $C7$ and $Y7$ represent Cyan vertex and Yellow vertex of RT 7 respectively. Also shown is the blind curve segment generated by disk 9 where $Y9L$ and $Y9R$ represent the vertices originated by the Yellow vertex and blocking disk 9 at the left and the right respectively. Figure 96 shows only partial result of the VCS($7A_2$) to be determined. Table 41 and Figure 97 show the final result of VCS($7A_2$) which is much shorter than CVCS($7A_2$). This is expected because there are permutation of five vertices of the interested region which has 6 disks generating blind CS. The blind CS is to be
removed from the candidate visible curve segment to find the final VCS as thoroughly demonstrated in a single RT problem.

4.7.3 Input of the IndigoPath Algorithm

All RTs that do not intersect the coverage path, the BluePath, are considered as inputs to the IndigoPath algorithm along with the BluePath. For our example problem, the input RTs for this phase of the Rainbow algorithm are RTs 7, 9, 12, 14, 15, and 18.

4.7.4 Output of the IndigoPath Algorithm

Figure 97 illustrates the IndigoPath algorithm with the BluePath as input along with all RTs that the BluePath enveloped and the IndigoPath output. Figure 97 illustrates only the IndigoPath output for readability. RTs 7, 9, 12, 14, 15, and 18 are partitioned by the LosPoRT algorithm 1. The corresponding visible curve segment’s endpoints are shown along the IndigoPath, the coverage path in indigo color, with five different colors: red, yellow, gray, blue, and green for readability. Recall that the endpoints along the IndigoPath are not included because the visible curve is an open ended curve. Because every sub region shown has one or more visible curve segment, we do not need to partition any sub region into smaller sub regions with LosPoRT algorithm 2 unless minimization of the number of observer is required. Recall that all curves throughout this paper are in clockwise direction. However, further partitioning the sub regions with LosPoRT algorithm 2, may yield much better result in term of minimizing the number of observers needed for searching the entire target region.

The partition of RT 7 with LosPoRT algorithm 1 yields sub regions 7A1, 7A2, and 7A3. As shown in Table 41, all sub regions of RT 7 have one or more visible curve segments which make
them observable from the coverage path. In the VioletPath phase of the Rainbow algorithm, we will see how the observers are selected. For the moment, we only need to know that the existence of visible curve segment for a sub region guarantee an observability of that sub region from the coverage path. See Lemma 2’ proof for rigorous proof. In clockwise direction, the sub region 7A1 is observable from two visible curve segments, curve segments (71,73) and (57, 67).

![Figure 97. LosPoRT of selected RTs and the VCSs it generated.](image)

The colored points on the IndigoPath are the endpoints of visible curve segments and they are considered in clockwise order. The visible curve segments displayed are only for RTs that the IndigoPath does not intersect for the reason of readability. It is obvious that visibility is guarantee for the RT with its visible polygon being crossed by the BluePath such as RT 1.
Table 41. **Sub Regions (SR) and Their Visible Curve Segments As Shown in Figure 97**

<table>
<thead>
<tr>
<th>Sub Region</th>
<th>VCS</th>
<th>Sub Region</th>
<th>VCS</th>
<th>Sub Region</th>
<th>VCS</th>
<th>Sub Region</th>
<th>VCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>7A1</td>
<td>(71,73)</td>
<td>9A3</td>
<td>(76,5)</td>
<td>15A1</td>
<td>(18,25)</td>
<td>14A2</td>
<td>(28,32)</td>
</tr>
<tr>
<td>7A2</td>
<td>(57,67)</td>
<td>9A3</td>
<td>(7,12)</td>
<td>15A1</td>
<td>(31,36)</td>
<td>14A3</td>
<td>(49,56)</td>
</tr>
<tr>
<td>7A3</td>
<td>(9,15)</td>
<td>12A1</td>
<td>(21,26)</td>
<td>15A1</td>
<td>(40,42)</td>
<td>14A3</td>
<td>(60,68)</td>
</tr>
<tr>
<td>7A1</td>
<td>(74,6)</td>
<td>12A1</td>
<td>(33,37)</td>
<td>15A2</td>
<td>(59,64)</td>
<td>14A3</td>
<td>(3,4)</td>
</tr>
<tr>
<td>7A3</td>
<td>(8,11)</td>
<td>12A1</td>
<td>(38,44)</td>
<td>15A2</td>
<td>(50,56)</td>
<td>14A3</td>
<td>(15,21)</td>
</tr>
<tr>
<td>7A3</td>
<td>(66,70)</td>
<td>12A1</td>
<td>(47,51)</td>
<td>15A3</td>
<td>(14,20)</td>
<td>18A1</td>
<td>(30,36)</td>
</tr>
<tr>
<td>9A1</td>
<td>(10,17)</td>
<td>12A2</td>
<td>(48,56)</td>
<td>15A3</td>
<td>(75,1)</td>
<td>18A1</td>
<td>(41,45)</td>
</tr>
<tr>
<td>9A2</td>
<td>(46,53)</td>
<td>12A3</td>
<td>(58,63)</td>
<td>14A2</td>
<td>(16,23)</td>
<td>18A3</td>
<td>(24,31)</td>
</tr>
<tr>
<td>9A2</td>
<td>(72,2)</td>
<td>12A3</td>
<td>(14,22)</td>
<td>14A1</td>
<td>(58,61)</td>
<td>18A3</td>
<td>(43,46)</td>
</tr>
<tr>
<td>9A2</td>
<td>(65,69)</td>
<td>15A1</td>
<td>(47,54)</td>
<td>14A1</td>
<td>(47,54)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Now we are ready to provide a rigorous proof for Lemma 2.

**Lemma 2 proof:**

*Proof*: Any region partitioned by the LosPoRT algorithm has at most 5 vertices [see Figure 82 (a)]. Since the region is part of a RT that is formed by 3 disks, disks $i, j,$ and $k$, there exist at least three potentially visible curves, corresponding to the number of disks, from which an observer point can be chosen that can visualize the whole set of $A_l$ where $l = 1, 2, 3$. The number of visible curves can also be directly obtained from drawing a visible cone from all vertices of subset $A_l$ to enclose each of the disks inside the coverage path. The rays that originate from each vertex are tangent to each side of each disk. Any point on the “visible curve” can observe the interested region. More specifically, a point on the VCS can observe all vertices of the interested region.

### 4.8 The VioletPath Algorithm

Motivated by [88], the VioletPath algorithm simply computes a rectangular box that wraps around the coverage path computed in the previous phase, phase 6, and then divided the box
vertically with \((2N + 1)\) number of vertical lines where \(N\) represents the number of disks in the target region. The number of vertical lines required may be increased as desired to yield better performance or to ensure that every visible curve intersects with at least one vertical line. The LosPoRT algorithm and visible polygon algorithm generate visible curves along the coverage path for visibility analysis. However, the CSIRT algorithm and visible circle path modification generate observer points for visibility coverage. For this reason, observer points generated by CSIRT and visible circle path modification should be considered first when deciding on observer points for visibility coverage. Figure 98 illustrates the input/output relationship of the VioletPath algorithm.

![Figure 98. Example input and output of VioletPath algorithm](image)

Table 42 spell-out the required inputs and resulting output in words. The VioletPath algorithm consists of seven steps. The first step of the algorithm determines the length and width/height of the minimum rectangle which enveloped the input coverage path. The result of the first step allows the algorithm to determine the height and the number of vertical LS required to perform the search for potential observers in steps 3 and 4. The potential observers are actually the intersection point of the vertical LS to be tested and the visible curve segments or visible polygons found in the previous phases of the algorithm. The other type of vertical LS test in step 5 has the same height as the vertical line test determined in step 2. The different between
the two different vertical line tests are the spacing. The vertical line tests generated in step 5 occur at the observer points generated by the CSIRT algorithm.

### TABLE 42. VioletPath algorithm. The Last Rainbow algorithm.

<table>
<thead>
<tr>
<th>VioletPath algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> The BluePath, all crossing RTs, including crossing and noncrossing SRs, and their VCSs. CSIRT and VCPM observers. All SRs of the noncrossing RTs and their VCSs.</td>
</tr>
<tr>
<td><strong>Output:</strong> Observers that jointly observe the TR.</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. Find a MER based on the input CP. Find the height of VL test.</td>
</tr>
<tr>
<td>2. Divide the width of a MER in step 1 by ((2N + 1)) to determine the spacing for ((2N + 1)) number of VL tests.</td>
</tr>
<tr>
<td>3. For each VL determine its intersection(s) with the VCS along the “upper ” CS of the BluePath. Store the intersection as observer point, VLUpt, and store all regions observe by VLUpt in VLUrg.</td>
</tr>
<tr>
<td>4. Repeat step 3 for the “lower” CS of the CP with VLBpt and VLBrg as observer point and region storage respectively.</td>
</tr>
<tr>
<td>5. Draw a different line test for each and every CSIRT observers. Store the intersection in CSIRTpt and store all regions observe in CSIRTrg.</td>
</tr>
<tr>
<td>6. Draw a different line test for each and every VCPM observers. Store the intersection in VCPMpt and store all regions observe in VCPMrg.</td>
</tr>
<tr>
<td>7. Call the OP algorithm with the results of steps 3, 4, 5, and 6 as inputs.</td>
</tr>
</tbody>
</table>

Figure 99 illustrates vertical line tests generated by steps 1, 2, 3, and 4 of the algorithm. Some of the results of the vertical line test are tabulated in Table 43. Enlarged portion of Figure 99 is shown in Figure 100 (a) with some of the results of the vertical line test are shown in Figure 100 (b). The numbering on the observer points in Figure 99 are labeled in red, but the observer points themselves are colored red, yellow, gray, blue, and green in alternate and clockwise order. The last observer point is labeled in purple to denote the last observer point on the coverage path.
In Figure 100, the result of vertical lines 19 and 20 are also identical to vertical lines 18 and 21. The underline simply means that the vertical line is intersecting with the lower coverage path. In addition to intersecting 5 different visible curve segments, vertical line 18 also intersect visible polygon 10. Equivalently, we can also say that the observer created by vertical line 18 is
inside visible polygon 10 which allow it to see all of RT 10. In addition, this observer can also observed regions 7A1, 14A3, 9A1, 15A2, and 12A3.

![Diagram showing vertical line test and selected result](image.png)

<table>
<thead>
<tr>
<th></th>
<th>VL17</th>
<th>VL18</th>
<th>VL21</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(57,67)</td>
<td>(57,67)</td>
<td>(57,67)</td>
</tr>
<tr>
<td>2</td>
<td>(60,68)</td>
<td>(60,68)</td>
<td>(60,68)</td>
</tr>
<tr>
<td>3</td>
<td>(62,68)</td>
<td>(62,68)</td>
<td>(62,68)</td>
</tr>
<tr>
<td>4</td>
<td>(59,64)</td>
<td>(59,64)</td>
<td>(59,64)</td>
</tr>
<tr>
<td>5</td>
<td>VP10</td>
<td>(58,63)</td>
<td>(58,63)</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>VP10</td>
<td>VP10</td>
</tr>
</tbody>
</table>

(a) Vertical line test. (b) Selected result of vertical line test.

Figure 100. Vertical line test in RT 8 and its vicinity.

With (2N+1) vertical line test and with upper and lower coverage path test we are expecting to find many candidate observers. Topping the challenge that we are already facing, we also have CSIRT observers and VCPM observer(s) to handle. Because CSIRT observers and VCPM observer are point observers, not visible curve segment observer, they only observed specific regions, and maybe, by chance of luck able to observe some other regions nearby, they need to be processed first. That is why the CSIRT observers and the VCPM observer are processed before all other type of observers (e.g. Curve segment and vertical line test observer) as seen in steps 2 and 3 of the OPD algorithm.

### 4.8.1 Complete CoveragePath Algorithm, Observer Point Selection

Observer Point Determination (OPD) algorithm is a sub algorithm of the VioletPath algorithm. It helps determine sufficient number of observer necessary to observe the target
region. Table 44 shows the OPD algorithm with 7 steps. It is called by step 6 of the VioletPath algorithm.

### TABLE 44. OBSERVER PLACEMENT ALGORITHM.

<table>
<thead>
<tr>
<th>Observer Point Determination Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> VLUpt, VLUrg, VLBpt, VLBrg, CSIRTpt, CSIRTrg, VCPMpt, VCPMrg, and TargetRg, the TR with all RTs and their partitioned SRs.</td>
</tr>
<tr>
<td><strong>Output:</strong> A collection of observers that observe the entire TR together.</td>
</tr>
<tr>
<td><strong>Algorithm Steps Description:</strong></td>
</tr>
<tr>
<td>1. Initialize all regions of the TR, e.g., RT1 and 7A3, to “not observed”.</td>
</tr>
<tr>
<td>2. Merge VLUpt and VLBpt into VLpt. Sort VLpt based on the most number of regions in VLUrg and VLBrg that each VLpt can observe.</td>
</tr>
<tr>
<td>3. While removing CSIRTpt from the list one at a time, initialize TargetRg’s region to “observed” based on CSIRTpt’sCSIRTrg.</td>
</tr>
<tr>
<td>4. While removing VCPMpt from the list one at a time, initialize the TargetRg’s region to “observed” based on VCPMpt’sVCPMrg.</td>
</tr>
<tr>
<td>5. While removing VLpt from the list one at a time, initialize the TargetRg’s region to “observed” based VLpt’s regions in VLUrg and VLBrg.</td>
</tr>
<tr>
<td>6. If all observer points have been tested and there still exist “not observed” region, then place an observer where the CS of the BluePath first intersects the VCS of the “not observed” region in CW direction. Update the TargetRg’s region as “observed”. Else end this program.</td>
</tr>
<tr>
<td>7. Repeat step 5.</td>
</tr>
</tbody>
</table>

#### 4.8.2 Input of the VioletPath Algorithm

The input to the VioletPath algorithm include the visible polygons of all RTs that intersect the BluePath, all observer points output by the CSIRT algorithm and visible circle path modification algorithm for visibility analysis of selected RT with the BluePath’s curve segments, and all “visibility curves” of the RTs that required the LosPoRT algorithm for visibility analysis.

#### 4.8.3 Output of the VioletPath Algorithm

Figure 99 illustrates the output of steps 1 to 4 of the VioletPath algorithm. Some of the results of the vertical line test shown in Figure 99 are tabulated in Table 43. Figure 99 also
presented the endpoints of the visible curves in different colors which are also numbered in red. There are 76 endpoints shown on the coverage path and they are in clockwise order. Computers do not have problems computing the intersection of all visible curves with the vertical line test. However, it is a bit difficult for us to visualize.

**4.9 Conclusions**

Our Rainbow Coverage Path Planning approach results in a complete and first order differentiable coverage path with static disks avoidance which means it can be implemented with time and energy efficiency. The dynamic obstacle avoidance technique can be incorporated as will be shown in Chapter 6. Control points can be selected along the curve segments that form the VioletPath so that the third order nonholonomic vehicle can compute a trajectory through the algorithm presented in Chapter 6. In the case where complete coverage is not required, if other important parameters are also considered, a utility function can be imposed to weight and implement the decision. Integration under the curve can quickly find the area which consideration can be taken to cover a certain region of the target region or not based on the cost imposed on the distance of the path travel or the cost of computation on the sensor. A coverage path generated by our technique is shorter than a coverage path obtained in [1], [69], and [70] due to the foundation path obtained through the convex hull algorithm and interior disks. Our approach can also be modified to adapt to unknown environments by applying the seven templates based approach implemented in [94] and initial navigation through the known boundary of the target region.
Table 45. RAINBOW ALGORITHM’S VERTICES, LSS, AND CSS.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of vertices</th>
<th>Number of LS</th>
<th>Number of curve segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedPath</td>
<td>$N_h$</td>
<td>$N_h$</td>
<td>0</td>
</tr>
<tr>
<td>OrangePath</td>
<td>$2N_h$</td>
<td>$N_h$</td>
<td>$N_h$</td>
</tr>
<tr>
<td>YellowPath</td>
<td>$N_h + 2N_{col}$</td>
<td>$N_h + N_{col}$</td>
<td>$N_h + N_{col}$</td>
</tr>
<tr>
<td>GreenPath</td>
<td>$N_h + 2N_{col} + 2N_{VCPM}$</td>
<td>$N_h + N_{col} + N_{VCPM}$</td>
<td>$N_h + N_{col} + N_{VCPM}$</td>
</tr>
<tr>
<td>BluePath</td>
<td>$N_h + 2N_{col} + 2N_{VCPM}$</td>
<td>$N_h + N_{col} + N_{VCPM}$</td>
<td>$N_h + N_{col} + N_{VCPM}$</td>
</tr>
</tbody>
</table>

Figure 101. Example of observer points and the regions they can observe

Figure 101 shows an example of observers which can observe a number of regions in the TR. Observer VD 7L can observe regions 7A1, 7A3, 9A1, 9A2, and 14A3. Observer VD11L can observe regions 12A1, 15A1, and part of RT16. Both observers, VD11L and VD11R can together observe all of RT16. There are a few possible failure conditions to be careful when
utilize the Rainbow algorithm. The Rainbow algorithm is expanded into 17 steps in the following two tables to illustrate more states for clarity and for expected conditions.

<table>
<thead>
<tr>
<th>step</th>
<th>phase</th>
<th>Input</th>
<th>Algorithm</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Sample TR</td>
<td>Graham Scan</td>
<td>Convex hull</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Convex hull</td>
<td>BURL</td>
<td>Convex hull and their CS’ status.</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Convex hull and CS status</td>
<td>CIT and CET</td>
<td>CP that envelopes all interior disks.</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>CP that envelops all interior disks and the TR.</td>
<td>BURL</td>
<td>CP and their CS’ status.</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>CP, their CS’ status, and the TR</td>
<td>CIT and CET</td>
<td>Collision free CP</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>TR</td>
<td>Delaunay Triangulation</td>
<td>The TR is split into multiple RTs.</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>TR with multiple RTs</td>
<td>VP</td>
<td>The TR with multiple RTs and VPs.</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>TR with multiple RTs</td>
<td>CSIRT</td>
<td>TR with additional SRs</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>TR with multiple RTs</td>
<td>LosPoRT</td>
<td>TR with additional SRs</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>TR with multiple RTs, CP</td>
<td>VCPM</td>
<td>New CP with additional observers</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>CP</td>
<td>VCS, CSIRT, CET, and CIT,</td>
<td>Update CP for differentiability.</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>RT, CP</td>
<td>LosPoRT 3</td>
<td>Additional SRs may be added due to LosPoRT 3.</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>RT, CP, SR, and TR.</td>
<td>LosPoRT 1, BLOCKING DISK</td>
<td>SRs, VCS of enveloping RTs and SRs.</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>RT, CP.</td>
<td>LosPoRT 2, BLOCKING DISK</td>
<td>CP and TR with additional SRs.</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>CP</td>
<td>MER</td>
<td>Rectangle enveloping the CP</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>CP, MER</td>
<td>Grid</td>
<td>Candidate observers along the CP</td>
</tr>
<tr>
<td>17</td>
<td>7</td>
<td>CP, and all candidate observers</td>
<td>Observer Placement</td>
<td>Observer that jointly observe the TR.</td>
</tr>
</tbody>
</table>

There are three possible failure conditions for the Rainbow algorithm. The first failure condition occurs when all interior points are collinear. If this happens, Graham Scan algorithm
would encounter degeneracy and cannot find the convex hull. This failure condition may be avoided by padding a non collinear point in the set of the interior points. The second failure condition occurs when all disks in the target region have collinear centers. If this happens, Delaunay Triangulation algorithm would encounter degeneracy and cannot partition the target region.

Table 47. GCPP APPROACH’S FAILURE CONDITIONS

<table>
<thead>
<tr>
<th>step</th>
<th>phase</th>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Graham Scan</td>
<td>Failure occurs if all interior points (center of interior disks) are collinear.</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>BURL</td>
<td>No failure condition provide step 1 pass.</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>CIT and CET</td>
<td>No failure condition provide step 1 pass.</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>BURL</td>
<td>No failure condition provide step 1 pass.</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>CIT and CET</td>
<td>No failure condition provides step 1 pass and assumption 4 hold.</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>Delaunay Triangulation</td>
<td>Failure occurs when all disks’ centers are collinear.</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>VP</td>
<td>Failure occurs when there is no single VP. However this failure condition can be overcome by splitting the RT along the longest edge and the center of the disk opposite to the longest edge. This process can be repeated for SR or RT that has already been split.</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>CSIRT</td>
<td>Failure occurs when there is no visible disk. The failure can be overcome with either LosPoRT algorithm 3 or the VCPM algorithm.</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>LosPoRT</td>
<td>Failure condition occurs if assumption 4 does not hold.</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>VCPM</td>
<td>Failure condition occurs if assumption 4 does not hold.</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>VCS, CET, and CIT</td>
<td>Failure condition occurs if assumption 4 does not hold.</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>LosPoRT 1</td>
<td>Failure condition occurs if assumption 4 does not hold.</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>BLOCKING DISK</td>
<td>Failure condition occurs if assumption 4 does not hold.</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>LosPoRT 2, BLOCKING DISK</td>
<td>Failure condition occurs if assumption 4 does not hold.</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>MER</td>
<td>No failure condition.</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>Grid</td>
<td>No failure condition.</td>
</tr>
<tr>
<td>17</td>
<td>7</td>
<td>Observer Placement</td>
<td>No failure condition.</td>
</tr>
</tbody>
</table>
This failure condition may be avoided by padding a non collinear point in the target region provided that all disks in the target region met assumption 4 in the formulation of our coverage problem. The third failure condition occurs when an assumption 4 does not meet. This means that the set to be covered is not connected with respect to the size of the robot. In this circumstance, the robot cannot maneuver in all simple cells, RTs, of the target region which implies that complete coverage is not obtainable.
5.1 Introduction

This chapter is concerning with the running time of our approaches, both local and global. It is the result of our published papers. Some of our algorithms depend on some existing algorithms. In such case, the running time for existing algorithms already existed and can be found in the literatures. For existing algorithms, we will just reference the running time result from the literature. For our new algorithms, we will derive and show how their running times are computed. The organization of this Chapter follows the chronological order of our algorithms as shown in Chapter 3 and Chapter 4. Before we discuss our algorithms’ running time in Section 5.3 and other sections that followed, we introduce the standard running time metric known as the Big-O notation in Section 5.2.

5.2 Big-O Notation

In 1892, German mathematician introduced the big-O notation to compute the complexity of algorithms. The notation $O(N)$ represents the linear running time, also known as linear complexity, of the algorithm. The big-O notation is intended to express the qualitative behavior of the algorithm, instead of the quantitative behavior [97]. In term of qualitative analysis, the values of $O(N), O(2N)$, and $2(O(N))$ are consider the same. The expression $O(C(N^2 + N))$, with C being a constant, is qualitatively similar to $O(N^2)$ and generally express with the later notation for simplicity [97]. Readers are referred to reference [97] for details of running time and complexity analysis.
5.3 A Local Coverage Path Planning Approach

In this section, we will show how fast the running time is for our LCPP and ILCPP algorithms. With the target region given as in Figure 1, we perform Delaunay triangulation in $N \log(N)$ time to get exact partition of the target region and we called each of the cell obtained in the target region as RT or Regular Triangulation. With the target region divided into multiple RTs, we discovered/contributed two algorithms which are implemented in the LCPP approaches and we called them RBOP algorithm and PNWCC algorithm. This section is dedicated to computing the running time of our algorithms and we will begin with computing the visible polygon.

5.3.1 Visible Polygon

From chapter 3, we show how a visible polygon is found. In this section, we will show that the visible polygon of the RT can be computed in linear time. For simplicity, we presented the result in the following steps.

Step 1: Computing the RT’s EPs

Each RT has three disks, see Figure 102. Each disk has two EPs. Therefore each RT has six EPs. The running time to compute the six EPs is directly correlates with the loop counter which we determined to be a constant number. The result is that the EPs in an RT can be computed in linear time.
The running time to compute the six EPs is computed from the pseudo code below:

<table>
<thead>
<tr>
<th>Table 48. PSEUDO CODE TO COMPUTE EPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS(disk[i], disk[i+1]) = disk[i]; //solve for x and y of e1</td>
</tr>
<tr>
<td>LS(disk[i], disk[i+2]) = disk[i]; //solve for x and y of e2</td>
</tr>
<tr>
<td>LS(disk[i+1], disk[i]) = disk[i+1]; //solve for x and y of e1</td>
</tr>
<tr>
<td>LS(disk[i+1], disk[i+2]) = disk[i+1]; //solve for x and y of e2</td>
</tr>
<tr>
<td>LS(disk[i+2], disk[i]) = disk[i+2]; //solve for x and y of e1</td>
</tr>
<tr>
<td>LS(disk[i+2], disk[i+1]) = disk[i+2]; //solve for x and y of e2</td>
</tr>
</tbody>
</table>

Running time to compute all six EPs is constant time. The pseudo code presented in Figure 103 is completed in a single iteration for one RT. For M number of RTs, the running time is M which is linear.

Step 2: Sorting the EPs of an RT

The running time for step 2 is log linear, N log N. The running time result is given by several well known sorting algorithms such as merge sort and quick sort [97]. Figure 104 shows the sorted EPs based on their x-coordinate relative to the whole RT. With the ordering of the six EPs known, we can progress to Step 3.
Step 3: Computing the visibility lines of sight (VLoS)

The running time for this step is linear time as will be described in this section. Each RT has 3 edges of interest: $S_{56}$, $S_{58}$, and $S_{68}$. When we reduce the disks in the RT to points, we have a triangle which is a subset of a polygon (see Figure 106 (a)). Figure 105 shows one of the visibility line of sights, the necessary step in determining the visible polygon, spawns by one of the end points of edge $S_{58}$. Each end point of the edge of interest has two sweep directions. Performing a BURL test on the RT to determine relative position of all three edges, we know which direction the visibility line of sight is supposed to sweep. The other sweep direction is toward the other end point of the same edge. Figure 105 (b) shows an initial sweep of the visible polygon within the RT, since $e_2$ is one of the endpoints of $S_{58}$, then the sweep direction is toward $e_6$. We happen to choose $e_2$ as the first sweep point, but it does not have to be $e_2$. We can choose any EP to begin with. However, it is much easier to start from $e_4$ for programmability and usability.
Each edge generates two visibility lines of sight (VLoS) which may be considered an infinite line in theory or a finite LS for drawing purpose. All VLoSs are perpendicular to the edge segments that spawned them. The VLoSs are function of the RT itself. Continue to generate VLoS for all endpoints of the three edges as shown in Figure 104 we get Figure 106. Figure 106 illustrates the situation we left in Section 3.2. We surely want to have a visible polygon as large as possible, but it opens a can of worms to be addressed. To make our problem solvable in the time that we have, this dissertation limits the visible polygon to be within the RT. This mean that the visible polygon shown in Figure 106 (b) is reduces to the visible polygon shown in Figure 111(b) which is a subset of the visible polygon shown in Figure 106 (b).
Since each RT can be reduced to a convex polygon with three vertices as illustrated in Figure 106 (c) and we have 6 VLoS for each and every RT, we know that the running time is \( \sum_{i=3}^{8} O(\log i) \) from Theorem 3 of reference [99]. For M number of RT, the running time is M, linear, since \( 6O(\log 3) \) is a constant. For convenience, we include Theorem 3 from reference [99] as follows:

Theorem [99]: The intersection of an infinite line with a convex polygon with p vertices can be computed in \( O(\log p) \) time.

<table>
<thead>
<tr>
<th>(a) Input prior to ( e_1 ).</th>
<th>(b) VLoS ( l_{567} ) during ( e_1 ).</th>
<th>(c) Output VP due to ( e_1 ).</th>
</tr>
</thead>
</table>

Figure 106. Partial visible polygon of RT 7 due to \( e_1 \).

Figure 106 shows that RT 7 is first reduced to its elementary convex polygon. It is done by setting all disks of the RT to point. The elementary convex polygon is then processed with the VLoS generated by the first EP to obtain the result of step 1’s convex polygon. Based on the sorted vertices, we obtained the result tabulated in Table 49. The VLoS in Table 49, 6 VLoS, are generated by the three sides of RT 7. For example, VLoS \( l_{5827} \) and \( l_{5867} \) are generated by side \( s_{58} \) which shared \( e_2 \) and \( e_6 \). Each side generates 2 VLoSs.
Table 49. Running Time to Compute the Visible Polygon of RT 7

<table>
<thead>
<tr>
<th>Step</th>
<th>Extreme point</th>
<th>VLoS</th>
<th>Resulting Convex Polygon</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$e_1$</td>
<td>$l_{5617}$</td>
<td>Four vertices convex polygon (Figure 107 (a))</td>
<td>$O(\log 3)$</td>
</tr>
<tr>
<td>2</td>
<td>$e_2$</td>
<td>$l_{5827}$</td>
<td>Five vertices convex polygon (Figure 108 (b))</td>
<td>$O(\log 4)$</td>
</tr>
<tr>
<td>3</td>
<td>$e_3$</td>
<td>$l_{5637}$</td>
<td>Six vertices convex polygon (Figure 108 (b))</td>
<td>$O(\log 5)$</td>
</tr>
<tr>
<td>4</td>
<td>$e_4$</td>
<td>$l_{6847}$</td>
<td>Seven vertices convex polygon (Figure 110 (b))</td>
<td>$O(\log 6)$</td>
</tr>
<tr>
<td>5</td>
<td>$e_5$</td>
<td>$l_{6857}$</td>
<td>Eight vertices convex polygon (Figure 111 (b))</td>
<td>$O(\log 7)$</td>
</tr>
<tr>
<td>6</td>
<td>$e_6$</td>
<td>$l_{5867}$</td>
<td>Nine vertices convex polygon (Figure 112 (b))</td>
<td>$O(\log 8)$</td>
</tr>
</tbody>
</table>

Figure 108 shows the third step’s process of computing the visible polygon of RT 7. In this step, RT 7 has already been reduced to a four-vertex convex polygon. The result of this step generates a five-vertex convex polygon. See Figure 108 as an example.

![Figure 107](image1.png)  
(a) Input prior to $e_2$.  
(b) Output post $e_2$.

Figure 107. Partial visible polygon of RT 7 due to $e_1$ and $e_2$.  

141
Figure 108. Partial visible polygon of RT 7 due to e1, e2, and e3.

Figure 109. Partial visible polygon of RT 7 due to e1, e2, e3, and e4.

Figure 110. Partial visible polygon of RT 7 due to e1, e2, e3, e4, and e5.
5.3.2 Traveling Salesman Algorithm

A number of Traveling Salesman algorithms are published in the literature. Different variant of Traveling Salesman algorithms may generate different path length and may encounter different level of complexity. The best heuristic algorithm that guarantees a path length close to the optimal solution, but with complicate running time is the Sanjeev Arora algorithm [72]. The Arora algorithm guarantee the path length to be within \((1 + 1/c)\) of the optimal length for \((c > 1)\). However the running time is quite expensive and it is \(O(n(log_2 n)^{O(c)})\) for two-dimensional problem. To simplify our problem, we connect our coverage path with the nearest neighbor variant of the Traveling Salesman problem. The running time for the nearest neighbor variant is \(O(n^2)\).

5.3.3 Cubic Spline Interpolation

Section 3.5 introduced the Cubic Spline interpolation technique with detail mathematical derivation. The set of equations and the two boundary conditions make the cubic spline very
practical because of their tri-diagonality [98]. The cubic spline can solved our problem in $O(N)$ time which is linear time.

5.3.4 Row Based Observer Placement Algorithm

Section 3.7 detail how the LCPP algorithm works. Table 50 shows the steps involve in computing the coverage path with LCPP algorithm. The expected running time is shown in Figure 113.

<table>
<thead>
<tr>
<th>Step</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Delaunay Triangulation</td>
<td>Performs Step 1 to divide the target region into a finite number of RT.</td>
</tr>
<tr>
<td>2</td>
<td>RBOP</td>
<td>Performs Step 2 to locally place observer in each RT to ensure complete coverage.</td>
</tr>
<tr>
<td>3</td>
<td>Nearest Neighbor</td>
<td>Performs Step 3 to link all observers in a single coverage path. The resulting coverage path is a linear spline.</td>
</tr>
<tr>
<td>4</td>
<td>Cubic spline</td>
<td>Performs Step 5 to transform the linear spline coverage path into a continuous and differentiable coverage path.</td>
</tr>
<tr>
<td>5</td>
<td>Dynamic obstacle avoidance</td>
<td>Performs Step 6 to steer the robot from the starting point to the ending point to perform coverage task.</td>
</tr>
</tbody>
</table>

Table 51. Expected running time for LCPP approach

<table>
<thead>
<tr>
<th>Step</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Perform Delaunay Triangulation in $O(N \log N)$ time to get Delaunay Graph with $F$ number of RT. Note that from the Euler’s formula, $F + V - E = 1$. Where $F$, $V$, and $E$ represent the number of RT, vertex, and edge respectively. In this paper, $N = V$.</td>
</tr>
<tr>
<td>2</td>
<td>Find $M$ equals number of circles in $O(M)$ times to fill all $F$ RT to completely cover the TR. For identical sensor, $M = \sum_{i=1}^{F} M_i$ where $M_i$ is the number of circle needed to fill RT $F_i$. $M \geq F &gt; N$.</td>
</tr>
<tr>
<td>3</td>
<td>Perform nearest neighbor heuristic of the TSP $M$ times on the set of $M$ points and obtain the shortest tour from the set of $M$ tours. This step lets all points of set $M$ to have a chance running as the first node once. The running time for this step is $O(M^2)$.</td>
</tr>
<tr>
<td>4</td>
<td>Perform cubic spline interpolation on the tour with $M$ points obtained in step 3.</td>
</tr>
<tr>
<td>5</td>
<td>Summing all running time in all 4 steps above result in $O(M^2)$ running time since $O(M^2)$ is the dominant term.</td>
</tr>
</tbody>
</table>
5.3.5 PNWCC Algorithm

Array data structure allows fast element access in search application such as high density archipelago search, but it is difficult to resize. A Singly-Linked List is a possible tool for development of PNWCC algorithm, as resize can be done by adding or removing waypoints; however, it is inefficient if removing the tail is required. In addition, there is no constant-time to update the tail to point to the previous node or waypoint. A Circular-Linked List, like a Singly Linked List, has a next pointer and an element value. The nodes of Circular Linked List are linked into a cycle. However, traversal of the Circular-Linked List is a one way lane because of the absence of a previous pointer.

A Doubly-Linked List (DLL) is a perfect choice for our Novel PNWCC tour generation algorithm. A representation of a subset of the tour in a DLL is illustrates in Figure 114. The C++ instruction would look like Figure 114. The running time for each DLL’s operation is tabulated in Table 52.

Table 52. A DLL WORSE CASE RUNNING TIME

<table>
<thead>
<tr>
<th>Operation</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion at either head or tail</td>
<td>O(1)</td>
</tr>
<tr>
<td>Deletion at either head or tail</td>
<td>O(1)</td>
</tr>
<tr>
<td>Waypoint data structure access</td>
<td>O(n)</td>
</tr>
</tbody>
</table>

![Figure 112. A DLL for PNWCC algorithm](image)
typedef string WPoint; // list WPoint

class DWP { // DLL list node
private:
  int wpn;   // WP number
  WPoint State; // WP State
  int nFace[1..N]; // WP Cover Face
  int nVertex[1..V]; // WP has V vertices
  double xVertex[1..V]; // x coordinate of vertex
  double yVertex[1..V]; // y coordinate of vertex
  bool Improvable; // can wp improve tour?
  DWP* prev; // previous wp in tour
  DWP* next; // next wp in tour
friend class DLLinkedList; //DLInkedList
};

Figure 113. A declaration of waypoint data structure in C++

class DLLinkedList { // DLL
public: // wp is waypoint
  DLLinkedList(); // constructor
  ~DLLinkedList(); // destructor
  bool empty() const; // is list empty?
  const WPoint& front() const;
  const WPoint& back() const;
  void addFront(const WPoint& PtFront);
  void addBack(const WPoint& PtBack);
  void removeFront();
  void removeBack();
private:
  DWP* header;
  DWP* trailer;
protected:
  void add(DWP* wp, const WPoint& PtAdd);
  void remove(DWP* wp);
};

Figure 114. A DLL class implementation in C++
### Table 53. Expected Running Time for PNWCC Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Perform Delaunay Triangulation in $O(N \log N)$ time to get Delaunay Graph with $F$ number of RT. Note that from the Euler’s formula, $F + V - E = 1$. Where $F$, $V$, and $E$ represent the number of face (RT), vertex, and edge respectively. In this paper, $N = V$.</td>
</tr>
<tr>
<td>2</td>
<td>Find $M$ equals number of circles in $O(M)$ times to fill all $F$ RTs to completely cover the TR. For identical sensor, $M = \sum_{i=1}^{F} M_i$ where $M_i$ is the number of circle needed to fill RT $F_i$. $M \geq F &gt; N$.</td>
</tr>
<tr>
<td>3</td>
<td>Perform nearest neighbor heuristic of the TSP $M$ times on the set of $M$ points and obtain the shortest tour from the set of $M$ tours. This step lets all points of set $M$ to have a chance running as the first node once. The running time for this step is $O(M^2)$.</td>
</tr>
<tr>
<td>4</td>
<td>Perform Previous-Next Waypoint Coverage Constraint algorithm on the tour obtained from step 3. This algorithm will run $M$ times for one complete cycle. The result will be a new $M' \leq M$. This algorithm runs in $O(M)$ times.</td>
</tr>
<tr>
<td>5</td>
<td>Perform cubic spline interpolation on the tour with $M'$ points obtained in step 4.</td>
</tr>
<tr>
<td>6</td>
<td>Summing all running time in all 5 steps above result in $O(M^2)$ running time since $O(M^2)$ is the dominant term.</td>
</tr>
</tbody>
</table>

### 5.4 Global Coverage Path Planning

A number of data structures are required to compute the running time in term of big O notation. Most of the data structures are used in multiple phases of the Rainbow Coverage Path Planning. For example, the LineSegment class is used in all seven phases of the Rainbow algorithm ranging from the RedPath to the VioletPath. Some member variables and function members are only used in some phases. For example, a member function AddLineSegment() is only used in the YellowPath and the GreenPath. A member function ModifiedLineSegment() is used in the OrangePath, the YellowPath, the GreenPath, and the BluePath. An example of the LineSegment class is shown in Table 54. Many other classes are required such as RT, VP, and SubRegion to name a few.
Table 54. EXAMPLE LINESEGMENT CLASS

```
Class LineSegment //class name
{
    private:
    long ID; //LS id
    vector<point> e1, e2; //begin point and end point
    vector<double> m, b; //slope and y-intercept
    vector<BOOL> IRT; //intersecting one or more RT
    vector<BOOL> IVP; //intersecting one or more VP
    vector<long> CSIRTRT; //value of RT to be CSIRT
    vector<BOOL> CSIRTPF; //CSIRT status
    vector<long> SubSegment; //sub segment
    vector<point> sse1; //sub segment’s beginning point
    vector<point> sse2; //sub segment’s ending point
    vector<long> ConnectArc; //in clockwise direction
    long AdjacentLineSegment; //in clockwise direction
    vector<point> VPObserver; //one observer for one VP/RT
    vector<point> CSIRTObserver; //observers for one RT
    vector<int> LineStatus; //bottom, upper, right, left
    public:
    void IntersectRT(vector<BOOL>&);
    void IntersectVP(vector<BOOL>&); //LS intersects VP
    void ModifiedLineSegment(vector<long>&);
    void AddLineSegment(vector<long>&);
    void GetLineStatus(vector<int>&);
}
```

Table 55. SOME NOTATION

- $I$ is the number of disks.
- $I_i$ is the number of interior disks, $N_h \leq I_i < N$.
- $I_h$ is the number of interior disks with centers on the convex hull.
- $N_{col}$ is the number of disks involved in collision with OPLS.
- $N_h + N_{col} \leq N$.
- $M$ is the number of RTs, $(M + N - E = 1)$.
- $M = (M_{VP} + M_{CSIRT} + M_{VCPM} + M_{LOSPORT})$.
- $M_{SR}$ is the number of sub regions, $3M_{LOSPORT}$.
For convenience, Table 55 illustrates the common variables used in computing the running time. The number of vertices, LSs, and curve segments increase from the RedPath to the GreenPath and then remain constant from the GreenPath onward. Table 56 shows how much time it takes to compute the most intensive activities in each of the color phase of the Rainbow algorithm. The leftmost column of Table 57 represents by \{1, 2, 3, 4, 5, 6, 7\} are for \{RedPath, OrangePath, YellowPath, GreenPath, BluePath, IndigoPath, VioletPath\} respectively.

As seen in Table 57, none of the dominant activity is converged as the number of input \(N\) approach infinity. The slowest activity is in checking the intersection of the YPLS with the RT and the VP of the GreenPath algorithm. A nested for loops was implemented which result in quadratic runtime.

Locally, all of our algorithms do converge in at most three iterations where we compute the tangent LS to correct collision, visibility, and first order differentiable problems. The BURL algorithm and simple algebraic manipulation return the correct result of OPLS from the RPLS in only one iteration (See Table 57).

Table 56. A DLL WORSE CASE RUNNING TIME

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of vertices</th>
<th>Number of LS</th>
<th>Number of CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedPath</td>
<td>(N_h)</td>
<td>(N_h)</td>
<td>0</td>
</tr>
<tr>
<td>OrangePath</td>
<td>(2N_h)</td>
<td>(N_h)</td>
<td>(N_h)</td>
</tr>
<tr>
<td>YellowPath</td>
<td>(2N_h + 2N_{col})</td>
<td>(N_h + N_{col})</td>
<td>(N_h + N_{col})</td>
</tr>
<tr>
<td>GreenPath</td>
<td>(2N_h + 2N_{col} + 2N_{VCPM})</td>
<td>(N_h + N_{col} + N_{VCPM})</td>
<td>(N_h + N_{col} + N_{VCPM})</td>
</tr>
<tr>
<td>BluePath</td>
<td>(2N_h + 2N_{col} + 2N_{VCPM})</td>
<td>(N_h + N_{col} + N_{VCPM})</td>
<td>(N_h + N_{col} + N_{VCPM})</td>
</tr>
</tbody>
</table>
Table 57. Running time of the Rainbow Algorithm

<table>
<thead>
<tr>
<th>Selected activities</th>
<th>Running Time</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Graham Scan [23]</td>
<td>$N_i \log N_i$</td>
<td>-</td>
</tr>
<tr>
<td>2 BURL</td>
<td>$N_h \log N_h$</td>
<td>1</td>
</tr>
<tr>
<td>Compute vertices</td>
<td>$N_h$</td>
<td>1</td>
</tr>
<tr>
<td>3 Collision correction</td>
<td>$N_h(N - N_h)$</td>
<td>3</td>
</tr>
<tr>
<td>4 Triangulation [16]</td>
<td>$N \log N$</td>
<td>-</td>
</tr>
<tr>
<td>Compute VP</td>
<td>$M$</td>
<td>1</td>
</tr>
<tr>
<td>YPLS intersect RT</td>
<td>$(N_h + N_{col})M$</td>
<td>1</td>
</tr>
<tr>
<td>YPLS intersect VP</td>
<td>$(N_h + N_{col})M$</td>
<td>1</td>
</tr>
<tr>
<td>CSIRT</td>
<td>$M_{\text{CSIRT}}$</td>
<td>1</td>
</tr>
<tr>
<td>LosPoRT 3</td>
<td>$M_{\text{LosPoRT}}$</td>
<td>3</td>
</tr>
<tr>
<td>VCPM</td>
<td>$M_{\text{VCPM}}$</td>
<td>3</td>
</tr>
<tr>
<td>5 Disk enlargement</td>
<td>$N_h + N_{col} + N_{ucpm}$</td>
<td>3</td>
</tr>
<tr>
<td>6 Partition, LosPoRT</td>
<td>$M_{\text{LosPoRT}}$</td>
<td>1</td>
</tr>
<tr>
<td>Sub Region to CVCS mapping</td>
<td>$(N_h + N_{col} + N_{VCPM})(M_{\text{LosPoRT}})$</td>
<td>1</td>
</tr>
<tr>
<td>CVCS</td>
<td>$(N_h + N_{col} + N_{VCPM})(M_{\text{LosPoRT}})$</td>
<td>1</td>
</tr>
<tr>
<td>Find blocking disk(s)</td>
<td>$N_i$</td>
<td>1</td>
</tr>
<tr>
<td>in the CVCS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCS for each SR</td>
<td>$(M_{SR})(N_i)$</td>
<td>1</td>
</tr>
<tr>
<td>7 Worst case line intersection for vertical grid with the followings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSIRT Observer</td>
<td>$(M_{\text{CSIRT}} + N + 1)\log(M_{\text{CSIRT}} + N)$</td>
<td>1</td>
</tr>
<tr>
<td>VCPM Observer</td>
<td>$(M_{\text{VCPM}} + N + 1)\log(M_{\text{VCPM}} + N)$</td>
<td>1</td>
</tr>
<tr>
<td>VP observer</td>
<td>$(M_{\text{VP}} + N + 1)\log(M_{\text{VP}} + N)$</td>
<td>1</td>
</tr>
<tr>
<td>LosPoRT’s VCSs</td>
<td>$(M_{\text{VCS}} + N + 1)\log(M_{\text{VCS}} + N)$</td>
<td>1</td>
</tr>
</tbody>
</table>

5.5 Conclusion

In this chapter we show that the running time in computing the visible polygon of each RT. Section 5.3 shows that the result is linear running time for the visible polygon. However, due to the Traveling Salesman’s algorithm involved in both the RBOP algorithm and the PNWCC algorithm, the overall running time for the local coverage path planning is quadratic. For the global coverage path planning, the Rainbow algorithm also results in quadratic running time as shown in Table 57.
CHAPTER 6 KINEMATICS AND DYNAMICS OF THE MOBILE ROBOT

A class of nonholonomic vehicles with different degrees of freedom may be control to move from a starting point to an ending point while avoiding moving obstacles with a closed form solution through mathematical transformation. Such a system also obeys the given time constraint provided the vehicle being controlled has sufficient capabilities (e.g. Fast enough speed and has enough fuel) and the environment consider has “free space” large enough for the robot to maneuver. References [83] and [84] show how smooth trajectory generation can be planned and then re planned as situation change, e.g. More moving obstacle(s) is detected. Our original results in Chapters 3 and 4provides the starting and the ending points that are already collision free with static disks while this chapter provides collision free with moving disks.

6.1 Introduction

In [83] the authors stated that a trajectory is feasible if the boundary conditions and kinematic constraints are satisfied. The said reference employed a fourth order chained form to present and validate the study. Constraints involve with trajectory planning of the robot include boundary, kinematic, and moving obstacles. Since this chapter is depending on the special triangular form of chained form, we are introducing the chained form as follow.

A chained form is a nilpotent form that can be used as a canonical form to describe nonholonomic system. Many mechanical systems with nonholonomic constraints can be locally or globally converted to chained form through coordinate change and control mapping. In the interest of space, we refer the reader to [84] for details description and transformation of chained form from world coordinate.
Without loss of generality, let us introduce the third order chained form which represents motion dynamic or kinematic model of the two wheel robot.

\[ \dot{z}_1 = v_{c1}, \]
\[ \dot{z}_2 = v_{c2}, \]
\[ \dot{z}_3 = z_2 v_{c1}. \]

Lemma: \[ 83 \] For the kinematic model in chained form above there exist two input \( v_{c1} \) and \( v_{c2} \). To obtain a feasible trajectory \( z_3 = F(z_1) \) between two boundary conditions \( z(t_0) = z^0 = [z^0_1, z^0_2, z^0_3] \) and \( z(t_f) = z^f = [z^f_1, z^f_2, z^f_3] \), the following must hold:

\[ z^0_1 \neq z^f_1. \]

If \( z^0_1 = z^f_1 \), there must be an intermediate point \( z^m \) with \( z^m = z^0_1 = z^f_1 \) so that singularity can be avoided. The boundary conditions in chained form are

\begin{align}
z^0_1 &= x^0, z_3^0 = F(z^0_1) = y^0, \quad \frac{d z_3}{d z_1} |_{z_1 = z^0_1} = z^0_1 = \tan(\theta_0), \quad (6.1) \\
z^f_1 &= x^f, z_3^f = F(z^f_1) = y^f, \quad \frac{d z_3}{d z_1} |_{z_1 = z^f_1} = z^f_1 = \tan(\theta_f). \quad (6.2)
\end{align}

### 6.2 Criterion For Avoiding Dynamic Objects

Consider time interval \( t \in [t_0 + k T_s, t_0 + (k + 1) T_s] \) that the robot is at coordinate \((x(t), y(t))\) and the i-th obstacle is at coordinate \((x_i(t), y_i(t))\) in the figure below. The robot is moving with the vector velocity \( v_r \triangleq [\dot{x}(t) \dot{y}(t)]^T \) to be determined.
The time-varying obstacle can be modeled as a moving point at initial location \( O_i = (x_i^k, y_i^k) \) with radius \( r \), where \( x_i^k = x_i(t_0 + kT_s) \) and \( y_i^k = y_i(t_0 + kT_s) \). The point \( O_i \) is moving at a known constant velocity \( \dot{v}_i = \dot{x}_i \dot{y}_i \). The relative velocity between the robot and the \( i \)th obstacle is defined as

\[
\dot{v}_{r,i}^k \triangleq \dot{v}_r - \dot{v}_i^k \quad (6.3)
\]

Considering relative velocity concept, Figure 115 can be viewed as Figure 116 in which obstacles are “static.”
Clearly seen from Figure 116, there will be no collision for $x'_i \in \left[ \bar{x}'_i, \bar{x}'_i \right]$ if

$$\left( y'_i - y^k_i \right)^2 + \left( x'_i - x^k_i \right)^2 \geq \left( r_i + R \right)^2 \tag{6.4}$$

Where $x'_i = x^k_i - r_i - R, \bar{x}'_i = x^k_i + r_i + R, x'_i = x - v^k_{i,x} \tau, y'_i = y - v^k_{i,y} \tau$, and
\(\tau = t - (t_0 + kT_s)\) for \(t \in [t_0 + kT_s, t_0 + T]\). The canonical chained form transformed from the world coordinate [84] is

\[z_1 = x,\]
\[z_2 = \tan \theta,\]
\[z_3 = y,\]
\[u_1 = v_{c1} \sec \theta,\]
\[u_1 = v_{c1} \sec \theta.\]

In term of state transformation, the collision-free constraint between the robot and the obstacle \(i\) become

\[
(z_{3,i}' - y_i^k)^2 + (z_{1,i}' - x_i^k)^2 \geq (r_i + R)^2
\]  

\[(6.5)\]

Where \(z_{1,i}' = z_1 - v_{i,x}^k \tau\) and \(z_{3,i}' = z_3 - v_{i,y}^k \tau\).

### 6.3 A Feasible Collision-Free Trajectory Parameterization

The time when a robot first start to move can assume to be 0. So for the first sampling instant, the robot move from 0 to \(kT_s\). Using the first equation of the kinematic model, \(\dot{z}_1 = v_{c1}\), where \(dz_1 = v_{c1} dt\), \(z_1\) can be found by integrating both sides:

\[
\int_0^{kT_s} dz_1 = \int_0^{kT_s} v_{c1} dt
\]

Letting \(z_i^k = z_i(t_0 + kT_s)\) and \(z_i^{k-1} = z_i(t_0 + (k - 1)T_s)\), the equation above become

\[z_1^k = z_1^0 + k \frac{z_1^f - z_1^0}{k},\]
\[ z_1(t) = z_1^k + \frac{z_1^f - z_1^0}{T_s}(t - t_0 - kT_s) \quad \forall t \in [t_0 + kT_s, t_f]. \]

From the second equation in kinematic model, \( \dot{z}_2 = v_c z_1 \),

\[ \int_{t_0+(k-1)T_s}^{t_0+kT_s} d\dot{z}_2 = \int_{t_0+(k-1)T_s}^{t_0+kT_s} v_c z_2 \, dt, \]

\[ z_2^k - z_2^{k-1} = \int_{t_0+(k-1)T_s}^{t_0+kT_s} v_c z_2 \, dt, \]

\[ z_2^k = z_2^{k-1} + \int_{t_0+(k-1)T_s}^{t_0+kT_s} v_c z_2 \, dt. \]

From the third equation in kinematic model, \( \dot{z}_3 = z_2 v_c \),

\[ \frac{d\dot{z}_3}{dt} = z_2 v_c, \]

\[ z_3^k = z_3^{k-1} + v_c^1 \frac{\int_{t_0+(k-1)T_s}^{t_0+kT_s} z_2^k \, ds}{t_0+(k-1)T_s}, \quad \text{(6.6)} \]

Replacing \( v_c^1 \) and \( z_2^k \) from above,

\[ z_3^k = z_3^{k-1} + v_c^1 z_2^{k-1} (t_0 + kT_s - (t_0 + (k-1)T_s)) + v_c^1 \frac{\int_{t_0+(k-1)T_s}^{t_0+kT_s} \int_{t_0+(k-1)T_s}^{t_0+kT_s} v_c z_2 \, ds \, dt}{t_0+(k-1)T_s}. \]

\[ z_3^k = z_3^{k-1} + v_c^1 z_2^{k-1} T_s + v_c^1 \frac{\int_{t_0+(k-1)T_s}^{t_0+kT_s} \int_{t_0+(k-1)T_s}^{t_0+kT_s} v_c z_2 \, ds \, dt}{t_0+(k-1)T_s} \quad \text{(6.7)} \]

Following the same proposition as in [83], a class of feasible and collision-free trajectories can be parameterized in polynomial and matrix form as

\[ z_3^k(z_1) = a_0^k + a_1^k z_1 + a_2^k z_1^2 + a_3^k z_1^3 + a_4^k z_1^4, \quad \text{(6.8)} \]

\[ z_3(z_1) = F(z_1) = a^k f(z_1) \quad \text{(6.9)} \]
Respectively, where $a^k$ is a constant vector to be determined, and $f(z_1)$ is a vector composed of basis function $z_1(t)$:

$$a^k = [a_0^k, a_1^k, a_2^k, a_3^k, a_4^k],$$

$$f(z_1) = \left[ 1, z_1(t), (z_1(t))^2, (z_1(t))^3, (z_1(t))^4 \right]^T.$$

Dividing $\dot{z}_3 = z_2 v_{c1}$ with $\dot{z}_1 = v_{c1}$ yields

$$\frac{dz_3^k}{dz_1^k} = a_1^k + 2a_2^k z_1^k + 3a_3^k (z_1^k)^2 + 4a_4^k (z_1^k)^3 \quad (6.10)$$

Where $dz_3^k/dz_1^k$ is

$$\frac{dz_3^k}{dz_1^k} = z_2^k,$$

$$z_2^k = a_1^k + 2a_2^k z_1^k + 3a_3^k (z_1^k)^2 + 4a_4^k (z_1^k)^3 \quad (6.11)$$

Finally, the initial and the final boundary conditions are given as follow

$$z_3^k = a_0^k + a_1^k z_1^k + a_2^k (z_1^k)^2 + a_3^k (z_1^k)^3 + a_4^k (z_1^k)^4,$$

$$z_2^k = a_1^k + 2a_2^k z_1^k + 3a_3^k (z_1^k)^2 + 4a_4^k (z_1^k)^3,$$

$$z_3^f = a_0^k + a_1^k z_1^f + a_2^k (z_1^f)^2 + a_3^k (z_1^f)^3 + a_4^k (z_1^f)^4 \quad (6.12)$$

$$z_2^f = a_1^k + 2a_2^k z_1^f + 3a_3^k (z_1^f)^2 + 4a_4^k (z_1^f)^3.$$
When $a_4^k$ is determined, the remaining coefficients can be found from the four equations four unknown variables in (6.12). Matrically,

$$Y^k = (B^k)[a_0^k, a_1^k, a_2^k, a_3^k]^T + A^k a_4^k$$  \hspace{1cm} (6.13)

Where

$$Y^k = \begin{bmatrix} z_3^k \\ z_2^k \\ z_3^f \\ z_2^f \end{bmatrix}, \quad A^k = \begin{bmatrix} (z_1^k)^4 \\ 4(z_1^k)^3 \\ (z_1^f)^4 \\ 4(z_1^f)^3 \end{bmatrix}, \quad B^k = \begin{bmatrix} 1 & z_1^k & (z_1^k)^2 & (z_1^k)^3 \\ 0 & 1 & 2z_1^k & 3(z_1^k)^2 \\ 1 & z_1^f & (z_1^f)^2 & (z_1^f)^3 \\ 0 & 1 & 2z_1^f & 3(z_1^f)^2 \end{bmatrix}.$$

Solving for $[a_0^k, a_1^k, a_2^k, a_3^k]^T$ in (6.13), we have

$$[a_0^k, a_1^k, a_2^k, a_3^k]^T = (B^k)^{-1}(Y^k - A^k a_4^k).$$ \hspace{1cm} (6.14)

### 6.4 Solution to Steering Velocity

The key to simplify (6.5)

$$(z_3 - v_{i,y}^k \tau - y_i^k)^2 + (z_1 - v_{i,x}^k \tau' - x_i^k)^2 \geq (r_i + R)^2,$$  \hspace{1cm} (6.15)

is by simplifying $z_3$. Rewriting (6.8) in matrix form, we have

$$z_3^k(z_1) = \begin{bmatrix} 1 & z_1 & z_1^2 & z_1^3 \end{bmatrix} \begin{bmatrix} a_0^k \\ a_1^k \\ a_2^k \\ a_3^k \end{bmatrix} + a_4^k z_1^4.$$

Replacing (6.14) in the above equation, we have

$$z_3^k(z_1) = f(z_1)(B^k)^{-1}(Y^k - A^k a_4^k) + a_4^k z_1^4.$$
Or

\[ z_3^k(z_1) = f(z_1)(B^k)^{-1}Y^k + \left( z_1^4 - f(z_1)(B^k)^{-1}A^k \right) a_4^k, \]  

(6.17)

where \( f(z_1) = [1 \ z_1 \ z_1^2 \ z_1^3] \). Replacing (6.17) in (6.15) yields a second order polynomial

\[
\min_{t \in [t_i, t_f]} \left[ g_2(z_1(t), k)(a_4^k)^2 + g_{1,i}(z_1(t), k, \tau) a_4^k + g_{0,i}(z_1(t), k, \tau) \right]_{t=t_0-kT_s} \geq 0, 
\]

(6.18)

where

\[
g_2(z_1(t), k) = \left[ (z_1(t))^4 - f(z_1(t))(B^k)^{-1}A^k \right]^2, 
\]

\[
g_{1,i}(z_1(t), k, \tau) = 2 \left[ (z_1(t))^4 - f(z_1(t))(B^k)^{-1}A^k \right] \left[ f(z_1(t))(B^k)^{-1}Y^k - y_i^k - v_{i,y}^k \right], 
\]

\[
g_{0,i}(z_1(t), k, \tau) = \left[ f(z_1(t))(B^k)^{-1}Y^k - y_i^k - v_{i,y}^k \right]^2 + (z_1(t) - x_i^k - v_{i,x}^k \tau)^2 - (r_i + R)^2. 
\]

The next step is to determine the steering control input \( v_{c2} \). \( v_{c1}^k = C \) is assumed to be a known constant, although it can be made varying as long as it does not violate integration rule. Let \( v_{c2} = C_0^k + C_1^k(t - t_0 - kT_s) + C_2^k(t - t_0 - kT_s)^2 \), where \( C_0^k, C_1^k, \) and \( C_2^k \). Directly integrating the kinematic model, equation of motion \( \dot{z}_1 = v_{c1}, \dot{z}_2 = v_{c2}, \) and \( \dot{z}_3 = z_2 v_{c1} \), yields equations that will lead to solution for steering velocity:

\[
z_1(t) = z_1^k + C(t - t_0 - kT_s), 
\]

\[
z_2(t) = z_2^k + C_0(t - t_0 - kT_s) + C_1^k \frac{t - t_0 - kT_s}{2} + C_2^k \frac{t - t_0 - kT_s}{3}, 
\]

(6.19)

\[
z_3(t) = z_3^k + C_0 z_2^k(t - t_0 - kT_s) + C_0 C_1^k \frac{t - t_0 - kT_s}{2} + C_1^k C_2^k \frac{t - t_0 - kT_s}{6} + C_2^k \frac{t - t_0 - kT_s}{12}. 
\]
for \( t \in (t_0 + kT_s, t_0 + (k + 1)T_s] \). Substituting \( z_1(t) = z_1^k + C(t - t_0 - kT_s) \) into

\[
z_3(z_1) = a^k f(z_1) \] yields

\[
z_3(t) = b_0 + b_1(t - t_0 - kT_s) + b_2(t - t_0 - kT_s)^2 + b_3(t - t_0 - kT_s)^3 + b_4(t - t_0 - kT_s)^4
\]

Where

\[
b_0 = a^k_0 + a^k_1 z_1^k + a^k_2 (z_1^k)^2 + a^k_3 (z_1^k)^3 + a^k_4 (z_1^k)^4,
\]

\[
b_1 = a^k_1 C + 2a^k_2 z_1^k C + 3a^k_3 (z_1^k)^2 C + 4a^k_4 (z_1^k)^3 C,
\]

\[
b_2 = a^k_2 C^2 + 3a^k_3 z_1^k C^2 + 6a^k_4 (z_1^k)^2 C^2,
\]

\[
b_3 = 4a^k_4 z_1^k C^3 + a^k_3 C^3,
\]

\[
b_4 = a^k_4 C^4. \quad (6.20)
\]

Comparing (6.19) and (6.20), the followings are obtained:

\[
C_0^k = 2a^k_2 C + 6a^k_3 z_1^k C + 12a^k_4 (z_1^k)^2 C,
\]

\[
C_1^k = 6a^k_3 C^2 + 24a^k_4 z_1^k C^2,
\]

\[
C_2^k = 12a^k_4 C^3. \quad (6.21)
\]

Equations above result in steering inputs to achieve obstacle avoidance path \((x,y)\). For \( t \in t_0 + kT_s, (k + 1)T_s] \),

\[
u_{c1}^k(t) = \frac{z_1^f - z_1^0}{\tau} \quad (6.22)
\]
\[ v_{c2}^k(t) = \left( 2a_2^k + 6a_3^k z_1^k + 12a_4^k (z_1^k)^2 \right) \mathcal{C} + \left( 6a_3^k + 24a_4^k z_1^k \right) (t - t_0 - kT_s) \mathcal{C}^2 + 12a_4^k (t - t_0 - kT_s)^2 \mathcal{C}^3 \]  

(6.23)

The results represent by equations (6.22) and (6.23) provide the real and steering velocities of \( u_1 \) and \( u_2 \) respectively. Based on observation with the two wheels robot, we can use \( u_1 \) and \( u_2 \) to find the speed of left and right wheels of the robot.

\[ \dot{\theta} = \frac{(V_R - V_L)}{L} \]

Since the guide point is at the center of the robot,

\[ u_1 = \frac{(V_R + V_L)}{2}. \]

### 6.5 Simulation

The second order inequality polynomial that dictates the solution of \( a_4 \) has three different scenarios: both \( a_{4_{min}} \) and \( a_{4_{max}} \) are positive, both \( a_{4_{min}} \) and \( a_{4_{max}} \) are negative, and both are of different signs. When \( a_{4_{min}} \) and \( a_{4_{max}} \) are positive or both are negative, \( a_4(k) \) can be chosen to be zero. When both have different signs, \( a_4(k) \) is the one with the smaller magnitude. See reference [103] for code and implementation.
Figure 117. Robot and moving obstacles’ trajectories

The trajectory of the robot is denoted by a solid blue line. The trajectory of the first obstacle is denoted by a dotted light-blue line. The trajectory of the second obstacle is denoted by a dash red line. The trajectory of the third obstacle is denoted by dash light-blue line. Along each trajectory, there exist 10 circles representing the robot’s or obstacle’s position during that sampling time. Placement of obstacles and robot was designed so that different interesting cases are shown.
Figure 118 shows the modified trajectory which is different from the initial trajectory planning due to obstacles being detected. As obstacle continues to be detected, trajectory modification continues to occur. This happen while the robot is continuously moving.
Robot’s trajectory for the last sampling instant is shown in Figure 119. All potential collisions are voided due to the guidance control published in [82]. The control inputs that drive the robot from specified initial location to desired location collision free are shown below.
Figure 120 shows that speed is very smooth from the beginning to the final position. Figure 121 shows that steering is also smooth. Both the speed and the steering are smooth due to canonical control provides by chained form and parameterized and analytical approach in [82].
Figure 121. Steering control (inch per second)

Figure 122. Turning angle in degree with respect to time
To summarize the result of Chapter 6, we presented the patrolling algorithm which works very well in both dynamic and static environment. A differential robot is the simplest representation of nonholonomic system that operates on land or in the water. Its simplest representation is given by

\[
\begin{align*}
\dot{x} &= u_1 \cos(\theta) \\
\dot{y} &= u_1 \sin(\theta) \\
\dot{\theta} &= u_2
\end{align*}
\] (6.24)

\[
\begin{align*}
\dot{z}_1 &= v_1 \\
\dot{z}_2 &= v_2 \\
\dot{z}_3 &= z_2 v_1
\end{align*}
\] (6.25)

The coordinate and input transformations to transform (6.24) into (6.25) were derived in Section 6.1 and they are:

\[
\begin{align*}
\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} &= \begin{pmatrix} x \\ \tan(\theta) \\ y \end{pmatrix}, \\
(u_1) &= \begin{pmatrix} v_1 \\ \sec(\theta) \\ \frac{v_2 \cos^2(\theta)}{v_1} \end{pmatrix}.
\end{align*}
\] (6.26)

Based on the results derived earlier in Chapter 6 and the previous result published in [81], our patrolling algorithm is as follows:

(i) Select coordinates (x, y), e.g. The observer points, and orientation of the robot such that \( \theta \neq \frac{\pi}{2} \), apply state and input transformations (6.26) and (6.27), and determine the corresponding boundary conditions \( z^0 = [z_1^0, z_2^0, z_3^0] \), \( z^f = [z_1^f, z_2^f, z_3^f] \) to obtain the chained form dynamics (6.25).

(ii) Let \( T_j \) be the time for the mobile robot to complete its maneuver between the adjacent pair of points and \( T_s^j \) be the sampling period such that \( \bar{k} = \frac{T_j}{T_s^j} \) is an integer. The center of the moving disks \( O_i \) moving at constant velocities \( v_i^k \) for \( t \in [t_i^0 + kT_s^j, t_i^0 + (k + 1)T_s^j] \) are located at \( (x_i^k, y_i^k) \) during the time when, \( t = t_0 + kT_s^j \). For time \( k = 0, \ldots, \bar{k} - 1 \), recursively determine constants \( a_i^k \) from inequality (6.28) and equations (6.29)-(6.35).

\[
\begin{align*}
\min_{t \in [t_i^0, t_i^f]} & \quad g_2(z_1(t), k)(a_i^k)^2 + g_1(z_1(t), k, \tau)a_i^k + g_{0,i}(z_1(t), k, \tau) \geq 0, \\
\tau &= t - t_0 - kT_s
\end{align*}
\] (6.28)
\[ g_2(z_1(t), k) = \left[ (z_1(t))^4 - f(z_1(t))(B^k)^{-1}A^k \right]^2, \quad (6.30) \]
\[ g_{1,i}(z_1(t), k, \tau) = 2 \left[ (z_1(t))^4 - f(z_1(t))(B^k)^{-1}A^k \right] \left[ f'(z_1(t))(B^k)^{-1}y^k - y^k_i - v_{i,j}^k \tau \right], \quad (6.31) \]
\[ g_{0,i}(z_1(t), k, \tau) = \left[ f'(z_1(t))(B^k)^{-1}y^k - y^k_i - v_{i,j}^k \tau \right]^2 + (z_1(t) - x_i^k - v_{i,x}^k \tau)^2 - (R_i + R_r)^2. \quad (6.32) \]

\[ B^k = \begin{bmatrix}
1 & z_1^k & (z_1^k)^2 & (z_1^k)^3 \\
0 & 1 & 2z_1^k & 3(z_1^k)^2 \\
1 & z_1^f & (z_1^f)^2 & (z_1^f)^3 \\
0 & 1 & 2z_1^f & 3(z_1^f)^2 \\
\end{bmatrix} \quad (6.33) \]

\[ Y^k = \begin{bmatrix}
z_3^k \\
z_2^k \\
z_3^f \\
z_2^f \\
\end{bmatrix}, \quad A^k = \begin{bmatrix}
z_1^{k4} \\
4(z_1^f)^3 \\
z_1^f \\
4(z_1^f)^3 \\
\end{bmatrix} \quad (6.34) \]

\[ f(z_1) = [1 \quad z_1 \quad z_1^2 \quad z_1^3] \quad (6.35) \]

(iii) A feasible and collision-free path in the transformed state can be parameterized in polynomial and matrix form as:

\[ z_3(z_1) = F(z_1) = a^k f(z_1) \quad (6.36) \]
where:
\[ a^k = [a_0^k, a_1^k, a_2^k, a_3^k, a_4^k], \quad (6.37) \]
\[ f(z_1) = \begin{bmatrix} 1, z_1(t), (z_1(t))^2, (z_1(t))^3, (z_1(t))^4 \end{bmatrix}^T, \quad (6.38) \]
\[ [a^k]^T = (B^k)^{-1}(Y^k - A^k a^k). \quad (6.39) \]

(iv) The steering inputs to achieve path (6.36) which is a subset of s(t) in (1.1) are determined to be:
\[ v_1(t) = C = \frac{z_1^f - z_0^f}{\tau} \]
\[ v_2(t) = \left( 2a_2^k + 6a_3^k z_1^k + 12a_4^k (z_1^k)^2 \right) C + \left( 6a_3^k + 24a_4^k z_1^k \right) (t - t_0 - kT_2) C^2 + 12a_4^k (t - t_0 - kT_3)^2 C^3 \]
6.6 Conclusions

This Chapter shows that trajectory planning in dynamic environment for a third order robot, in fact for any order nonholonomic robot [82], can move the robot from initial point to final point. Merging this dynamic part of the algorithm with the static coverage path planning, an efficient coverage path planning control can be performed in real time. From the static coverage paths designed in Chapter 3 and Chapter 4, the dynamic obstacle avoidance algorithm can be executed if dynamic obstacles are encountered. The dynamic algorithm can done by using two known points in world coordinate which are then transformed into chained coordinate and finally applied motion planning algorithm to avoid the obstacles. Once obstacle avoidance is performed, the robot can switch back to the static CP that has been planned.
CHAPTER 7 CONCLUSIONS AND FUTURE WORK

7.1 Summary of Main Contributions

Delaunay triangulation is a stable technique to partition the Target Region into several simple cells known as regular triangulations (RT). The concept of RT is different from a triangulation due to weighted disks, all edges of the triangle are chopped off with circles of different radii. Every time the Delaunay algorithm is ran, a unique output is obtained which may contain several nonconvex RT. With the RT, we developed 5 novel algorithms which together form our first Local Coverage Path Planning approach along with several existing algorithms. Our first novel algorithm, the Visible Polygon algorithm, transformed this nonconvex cell into a convex region for observer placement purpose. From the triangulation’s result, we were able to globally partition our target region into several simple cells and then locally plan a sufficient number of observers with circular sensing range which guaranteed complete coverage for each cell with the second algorithm that we developed, the Row Based Observer Placement (RBOP) algorithm. The Local Coverage Path Planning approach, approach 1, is shown in red in Figure 123 for the sample target region. A Local Coverage Path Planning approach enjoys plenty of overlap which may be desirable in certain application. After initial placements of observer for each RT in all RTs of the TR, all observers are connected with existing Nearest Neighbor algorithm to form a closed path also known as a tour. At this point, collision between initial coverage path segment that connected any observer and the static disk may occur. This collision is repaired with the third algorithm that we developed known as the Static Collision Avoidance (SCA) algorithm. The CP outputs by the SCA algorithm remain discontinuous so the fourth algorithm, Adaptive Circle (AC) algorithm, and the fifth algorithm, Discontinuous to Continuous (DC) algorithm are developed to remove discontinuity. An AC algorithm finds the
programmable curvature or user specified curvature to transform an otherwise discontinuous line segment into a smooth and continuous segment. This desirable property is achieved with the help of the DC algorithm which continuously connected the curvatures found by the AC algorithm and connected them with their respective line segments in the CP. Our LCPP approach achieve complete coverage path planning with continuous CP for all sensing ranges. LCPP is really sensitive to the initial placement of the observer with respect to the RT. This sensitivity is caused by several algorithms being used to place and connect the observers: RBOP and TSP’s nearest neighbor algorithm. As it is, the result the CP obtained with a LCPP may yield ripple CP and not efficient when the sensing getting really big. The downside of RBOP is that when sensing range getting larger, the coverage path may be formed by many ripples. To get a better result, we developed the PNWCC algorithm and incorporated it in the improved Local Coverage Path Planning approach with the result shown in blue in Figure 123.

The Local Coverage Path Planning approach is modified by inserting an additional algorithm that we developed to reduce ripple in the CP. We called the modified LCPP approach as Improved Local Coverage Path Planning Approach. Figure 123 illustrates that the ILCPP approach, the blue CP, is better than the LCPP CP, the red CP. Improvement includes the shorter coverage path length, less curvature, fewer numbers of turns, and fewer observers while maintaining complete coverage of the target region. The LCPP approach has five novel algorithms while the ILCPP approach has six novel algorithms. Until the end of the third step, both approaches are identical. Steps 4 and 5 of the LCPP approach are identical to steps 5 and 6 of the ILCPP approach respectively. Step 4 of the ILCPP approach is detailed in Section 3.4 as PNWCC algorithm. PNWCC algorithm takes the coverage linear spline path which may be formed by the TSP algorithm as input and then output a new path that is shorter, contains fewer
sharp angles, and with reduces curvatures. The output of step 3 of the LCPP approach is a linear spline CP. The output coverage path produces by the PNWCC algorithm remains a linear spline CP until we perform adaptive curvature transformation to get a differentiable coverage path.

Finally, to improve our CP’s result when sufficient range is available, we developed a global coverage path planning approach, also known as the Rainbow coverage path planning approach. Fundamentally, the Rainbow coverage path planning approach begins by generating a foundational path which then transforms again and again, seven times, to find the final desirable coverage path. The result of the Rainbow coverage path for our example is shown in green. Figure 123 shows that the Rainbow CP is the shortest and the most desirable CP. In the worst case, both the Local Coverage Path Planning Approach and the Rainbow Coverage Path Planning approach obtain the final result in quadratic run time. In optimal run time scenario, the Rainbow Coverage Path Planning approach may achieve log linear run time while the Local Coverage Path Planning approach remains quadratic run time. The three approaches consist of the following sub algorithms:

**Approach 1 - A Local Coverage Path Planning (LCPP) Approach**

1. Visible Polygon (VP) – developed to compute the convex polygon inside the RT from which a sufficient range observer can be placed within the convex polygon to observe the entire RT.
2. Row Based Observers Placement (RBOP) – developed to find a set of observers within the RT that can collectively observe the entire RT. This algorithm is used when an RT is relatively larger than the sensing range, Rs.
3. Static Collision Avoidance (SCA) – developed to replace the colliding line segment of the CP with two new line segments that detour around the disk that involve in collision.
4. Adaptive Circle (AC) – developed to find smooth curvature between three waypoints and the two LSs connecting them.
5. Discontinuous to Continuous (DC) – developed to transform the linear spline CP, piecewise continuous CP, into smooth CP while maintaining the same observer location.
Approach 2 - An Improve Local Coverage Path Planning (ILCPP) Approach

1. Visible Polygon (VP) – developed to compute the convex polygon inside the RT from which a sufficient range observer can be placed within the convex polygon to observe the entire RT.
2. Row Based Observers Placement (RBOP) – developed to find a set of observers within the RT that can collectively observe the entire RT. This algorithm is used when an RT is relatively larger than the sensing range, \( R_s \).
3. Static Collision Avoidance (SCA) – developed to replace the colliding line segment of the CP with two new line segments that detour around the disk that involve in collision.
4. Previous Next Waypoint Coverage Constraint (PNWCC) – developed to reduce the path length while maintaining complete coverage. This algorithm may move observers within the TR or delete them as needed.
5. Adaptive Circle (AC) – developed to find smooth curvature between three waypoints and the two LSs connecting them.
6. Discontinuous to Continuous (DC) – developed to transform the linear spline CP, piecewise continuous CP, into smooth CP while maintaining the same observer location.

Approach 3 – A Rainbow Coverage Path Planning or Global Coverage Path Planning (GCPP) Approach

1. RedPath - Generate a foundational path based on the interior disks’ position. The result of this algorithm is a convex hull that has all interior disks’ center lie either inside the convex hull or on the convex hull.
2. OrangePath - Processes the RedPath to obtain a coverage path that wrap around all interior disks whose center is on the convex hull/RedPath. The OrangePath may intersect/collide with one or more disks.
3. YellowPath - Processes the OrangePath to obtain a coverage path that wrap around all interior disks as well as avoiding collision with all known disks.
4. GreenPath - Processes the YellowPath to obtain a coverage path that can observe all partitioned regions of the target region that the coverage path crosses. The GreenPath inherits the collision-free property of the YellowPath.
5. BluePath - Processes the GreenPath to obtain a first order differentiable coverage path. The BluePath inherits the collision-free’s and the observability’s properties of the GreenPath.
6. IndigoPath - Processes the BluePath to obtain a coverage path that can observe all partitioned regions of the target region that the BluePath does not cross. The IndigoPath is actually the BluePath with the ability to observe all regions of the target region.
7. VioletPath - Process the IndigoPath to obtain a coverage path that can observe all partitioned regions of the target region. The VioletPath is actually the IndigoPath with all observe points found to observe the entire target region.
Although our obstacles are assumed circular, they can be relaxed to that of ellipsoid without affecting our LCPP approach and GLPP approach provided that our LCPP algorithms and GCPP algorithms generate coverage observer points. In the case of polygonal obstacle, it can be enveloped by an ellipsoid which means that a differentiable coverage path may be kept, however, the visibility analysis has to be modified based on the Art Gallery Theorem [72] or level set technique. Because our foundational simple cells are determined by the Delaunay triangulation, our LCPP approach and GCPP approach can be generalized to higher dimensions. It is well known that the convex hull and the triangulation techniques can be applied in 2-dimension as well as in 3-dimension.
7.2 Future Directions and Possible Extensions

In future research, we like to extend the coverage path planning to include irregular obstacle shapes in the target region. We also like to extend our RBOP algorithm, PNWCC algorithm, and the Rainbow algorithm to 3 dimensions. In the current form of 2 dimensional settings, both of our approaches provide a great starting point to incorporate a coverage path planning management system as a function of robotic parameters such as the order of nonholonomy and speed constraints, sensing parameters such as sensing range and sensing coverage, and other environmental parameters. An interesting question to ask at this point is can each of our approach obtain (1) minimal coverage path length, (2) minimum number of observer point, and (3) fastest possible computational speed to obtain the solution? Can all three optimal performance capabilities be obtained concurrently?

The current local coverage path planning approach and global coverage path planning approach consist of several algorithms. In both approaches, Delaunay Triangulation algorithm must be implemented to partition the target region into several simple cells or RTs. In a uniform sensing range scenario with a single robot, there may be one or more RTs that are relatively smaller than the sensing range. In such scenario, only one observer is required to observe each of those RTs. What this also implies is that the sensor may be able to see partial regions of other neighboring RTs. One of the problems to be solved in order to obtain the minimum coverage path length in both approaches is to partition each of the RT in the target region with all three LosPoRT algorithms that are currently developed for the Rainbow algorithm. Then run the Rainbow algorithm with the assumption that sufficient sensing range is available to measure possible optimal CP. If the Rainbow algorithm does not have to execute the VCPM algorithm to obtain the CP, then we have the true minimum CP length or optimal CP. With this knowledge,
we can work backward to find the optimal CP which is the minimum sensing range required to keeps the CP that we just obtain also known as the minimum coverage path length. It is a subject of further research if VCPM is executed to find the CP or the sensing range available is not sufficient.

With regard to obtaining the minimum number of observer, more research needs to be done. One possible solution for both the local coverage path planning and the global coverage path planning is to partition the RTs in the target region to a very small size such that each observer found can sense all SRs with very minimal overlap with other observer. This may require finite element algorithm [102] to be incorporate into our local and global approaches along with the three LosPoRT algorithms. Of course, the VP polygon needs to be enlarged to include region outside of the RT. This will incurred additional computational resources.

An important question to ask now is “can we concurrently obtain minimal coverage path length along with minimal number of observer while maintaining the fastest possible computational speed?” In the previous paragraph, we see that an attempt to minimize the number of observer with our approaches will incurred additional computational resources due to the expansion of the VP to include region outside of the RT. While enlarging the VP to include region outside of the RT is unavoidable, the increase in computation is not large enough to change the speed outcome. A number of data structures are required to compute the running time of our local and global approaches. They include the RT’s and LS’ data structures. Chapter 5 shows the running time of our approaches in term of big O notation. Since values of \(O(N), O(2N),\) \(\text{and } 2(O(N))\) are consider the same in term of the big O’s notation, there is hope that the answer to the question posed at the beginning of this paragraph is “it is possible.” However, it is still the subject of further research which is beyond the scope of this dissertation.
LIST OF PUBLICATIONS


LIST OF REFERENCES


A. Okabe, A. Suzuki, “Location optimization problems solved through Voronoi

M. Berg, O. Cheong, M. Kreveld, M. Overmars, “Computational Geometry:

M. Lam, Y. Liu, “Heterogeneous Sensor Network Deployment Using Disk Packings,”
in the Proceedings of the IEEE International Conference on Robotics and Automation, Roma,
Italy, 2007.

Fabri, Andreas, Pierre Alliez, and Efi Fogel. "Computational Geometry Algorithms

Z. Qu, J. Wang, and C. E. Plaisted, “A New Analytical Solution to Mobile Robot
Florida Conference on Recent Advances in Robotics, May 2003, also appearing in the IEEE
Transactions on Robotics.

Vatana An, “A Third Order Differential Steering Robot and Trajectory Generation in the
Presence of Moving Disks,” Masters of Science thesis, University of Central Florida,
December 2005.

A. Lawler, J. Lenstra, A. Rinnooy Kan, D. Shmoys, “The Traveling Salesman
1995.

E. Fogel, D. Halperin, R. Wein, “CGAL Arrangements and their Applications,”

M. Summerfield, “Advanced Qt Programming: Creating Great Software with C++ and


M. Schwager, D. Rus, “Unifying Geometric, Probabilistic, and Potential Field


