TRAJECTORY PLANNING AND OBSTACLE AVOIDNACE FOR OMNIDIRECTINONAL ROBOTS

MOHAMMED RABEEA HASHIM AL-DAHHAN

FEBRUARY 2020

TRAJECTORY PLANNING AND OBSTACLE AVOIDNACE FOR OMNIDIRECTINONAL ROBOTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF
ÇANKAYA UNIVERSITY

BY
MOHAMMED RABEEA HASHIM AL-DAHHAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF
ELECTRONIC AND COMMUNICATION ENGINEERING

FEBRUARY 2020

Title of the Thesis: Trajectory Planning and Obstacle Avoidance for Omnidirectional Robots
Submitted by **Mohammed Rabeea Hashim Al-Dahhan**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

Prof. Dr. Can ÇOĞUN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Sıtkı Kemal İDER
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Hüsnü Deniz BAŞDEMİR
Supervisor

Prof. Dr. Klaus Werner SCHMIDT
Co-Supervisor

Examination Date: 04.02.2020
Examining Committee Members

Assoc. Prof. Dr. Afşar Saranlı	(METU)	
Assoc. Prof. Dr. Hüsnü Deniz BAŞDEMİR	(Çankaya Univ.)	
Assist. Prof. Dr. Özgür ERGÜL	(Atilim Univ.)	
Assist. Prof. Dr. Ulaş BELDEK	(Çankaya Univ.)	
Assist. Prof. Dr. Selma ÖZAYDIN	(Çankaya Univ.)	

STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Mohammed Rabeea Hashim Al-Dahhan
Signature : M,R,H
Date : 04.02.2020

ABSTRACT

Trajectory Planning and Obstacle Avoidance for Omnidirectional Robots

Al-Dahhan, Mohammed Rabeea Hashim

Ph.D., Department of Electronic and Communication Engineering

Supervisor: Assoc. Prof. Dr. Hüsnü Deniz BAŞDEMİR

Co-Supervisor: Prof. Dr. Klaus Werner SCHMIDT

February 2020, 120 pages

Path planning algorithms for mobile robots are concerned with finding a feasible path between a start and goal location in a given environment without hitting obstacles. In the existing literature, important performance metrics for path planning algorithms are the path length, computation time and path safety, which is quantified by the minimum distance of a path from obstacles.

The subject of this thesis is the development of path planning algorithms that consist of straight-line segments. Such paths are suitable for omni-directional robots and can as well be used as initial solution paths for applying smoothing. As the main contribution of the thesis, we develop three new planning methodologies that address all of the stated performance metrics.

The original idea of the first approach is the pre-processing of the environment map by increasing the obstacle region. That is, when applying sampling-based path planning algorithms such as PRM* (probabilistic roadmap), RRT* (rapidly exploring random tree) or FMT (fast marching tree), node samples in irrelevant regions of the environment are avoided. This measure speeds up the path computation and increases path safety.

The second approach proposes the computation of a modified environment map that confines solution paths to the vicinity of the Voronoi boundary of the given environment. Using this modified environment map, we adapt the sampling strategy of the popular path planning algorithms PRM, PRM* and FMT. As a result, we are able to generate solution paths with a reduced computation time and increased path safety.

Different from the first two approaches, the third approach uses information about the topology of the environment from the generalized Voronoi diagram of the environment. Specifically, initial solution paths that follow Voronoi edges are iteratively refined by introduce shortcuts and by adding new waypoints to remove corners in the path.

The thesis performs comprehensive computational experiments to illustrate the advantages of the proposed approaches. In particular, the third approach proves to be most promising since it addresses the properties of environments for mobile robots.

Keywords: Path planning, omni-directional robots, sampling-based algorithms, Voronoi diagram, safety, fast computation, shortest path.

ÖZ

HER YÖNDE HAREKET EDEBİLEN ROBOTLAR İÇİN YOL PLANLAMA VE ENGELDEN KAÇINMA

Al-Dahhan, Mohammed Rabeea Hashim
Doktora., Elektronik ve Haberleşme Mühendisliği Anabilim Dalı
Tez Yöneticisi: Assoc. Prof. Dr. Hüsnü Deniz BAŞDEMİR
İkincil Tez Yöneticisi: Prof. Dr. Klaus Werner SCHMIDT
Şubat 2020, 120 sayfa

Mobil robotlar için yol planlama algoritmaları, belirli bir ortamda engellere çarpmadan başlangıç noktası ile hedef konum arasında uygun bir yol bulmak ile ilgilidir. Mevcut literatürde, yol planlama algoritmaları için önemli performans ölçütleri; yol uzunluğu, hesaplama zamanı ve bir yolun engellere olan azami mesafesi ile ölçülen yol güvenliği olarak belirtilmiştir.

Bu tezin konusu, düz doğru parçalarından oluşan yol planlama algoritmalarını geliştirmektir. Bu tür yollar, her yönde hareket edebilen robotlar için uygundur ve düzleştirmeyi uygulamak için başlangıç çözüm yolları olarak da kullanabilirler Tezimizi asıl amacı, belirtilen tüm performans ölçütlerini ele alan üç yeni planlama yöntemi geliştirmeye katkıda bulunmaktır.

İlk yaklaşımın altında yatan görüş, engel bölgesini arttırarak çevre haritasının önceden işlenmesidir. Yani, PRM* (olasılıksal yol haritası), RRT* (hızlı keşfeden rastgele ağaç) veya FMT (hızlı yürüyen ağaç) gibi örnekleme tabanlı yol planlama algoritmaları uygulanırken ortamın alakasız bölgelerindeki düğüm örneklerinden kaçınılmaktadır. Bu ölçüm, yol hesaplamasını hızlandırır ve yol güvenliğini ise arttırır.

İkinci yaklaşım, çözüm yollarını verilen ortamın Voronoi sınırının çevresine sınırlayan değiştirilmiş bir çevre haritasının hesaplanmasını ileri sürmektedir. Değiştirilmiş bir çevre haritasını kullanarak yaygın yol planlama algoritmaları olan PRM, PRM* ve FMT örneklem stratejilerini uyarlamaktayız. Sonuç olarak daha az hesaplama süresi ve artan yol güvenliği ile çözüm yolları üretebilmekteyiz.

İlk iki yaklaşımdan farklı olarak üçüncü bir yaklaşım ise çevrenin genelleştirilmiş Voronoi diyagramından çevrenin topolojisi hakkındaki bilgileri kullanmaktır. Özellikle, Voronoi kenarlarını takip eden ilk çözüm yolları; kısa yollar ortaya koyarak ve yoldaki köşeleri ortadan kaldırmak üzere yeni ara noktalar ekleyerek tekrarlanarak düzeltilmektedir.

Tezde, önerilen yaklaşımların faydalarını göstermek için kapsamlı hesaplama deneyleri uygulanmıştır. Bilhassa üçüncü yaklaşımın, mobil robotların çevre özelliklerini ele aldığı için daha çok umut vaat edici olduğu görülmüştür.

Anahtar Kelimeler: Yol planlama, her yönde hareket edebilen robotlar, örnekleme tabanlı algoritmalar, Voronoi diyagramı, güvenlik, hızlı hesaplama, en kısa yol.

ACKNOWLEDGEMENTS

It is a pleasure to thank the many people who made this thesis possible. I would like to take this opportunity to express my appreciation and respect to my Co-Supervisor Prof. Dr. Klaus Werner SCHMIDT who assisted me with his countless guidance, suggestions and encouragements throughout this project. He enthusiastically shared his vision, knowledge and expertise, and generously spent his time to help me to generate ideas and research concepts .

I would like to express my profound gratitude to my supervisor Asso. Prof. Dr. Hüsnü Deniz BAŞDEMİR.

It is my honor to express my thankfulness to the man who I proudly carry his name, my dear father Rabeea.

To the woman whose prayers are the secret of my success, my dear mother Annam.

To my brothers who always encourage me, Dr. Omer and Eng. Yassen.

To my sisters who always support me with their love.

To my friends whose their pictures and voices represent the most beautiful moments and days of my life.

TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM. iii
ABSTRACT. iv
ÖZ. vi
ACKNOWLEDGEMENTS. viii
TABLE OF CONTENTS. ix
LIST OF FIGURES. xii

CHAPTERS:

1 INTRODUCTION. 1
  1.1 Background. 1
  1.2 Path Planning for Mobile Robots. 1
  1.3 Minimizing the Path Length. 2
  1.4 Computation of Safe Robot Paths. 3
  1.5 Fast Computation of Robot Paths. 5
  1.6 Discussion and Motivation. 5
  1.7 Contributions of this Thesis. 6
  1.8 Thesis Organization. 7

2 BACKGROUNDS. 8
  2.1 Basic Definitions 8
    2.1.1 Notation 8
    2.1.2 Generalized Voronoi Diagram 10
    2.1.3 Voronoi Diagram and Dijkstra’s Algorithm 11
  2.2 Completeness and Optimality. 12

3 PRE-PROCESSING ENVIRONMENTS FOR EFFICIENT SAMPLING. 13
  3.1 Comparison of Existing Methods. 13
    3.1.1 Methods for Minimizing the Path Length. 13
    3.1.2 Methods for Path Safety. 15
    3.1.3 Methods for Fast Computation. 16
    3.1.4 Evaluation. 17
  3.2 Methodology for Safe Path Planning. 21
    3.2.1 Extended Generalized Voronoi Diagram. 21
    3.2.2 Safety Guarantees and Proposed Methodology. 23
    3.2.3 Evaluation. 25

3.2.4 Discussion. 29

4 MODIFYING THE SAMPLING STRATEGY FOR FAST CLASSI-CAL ALGORITHMS. 31

4.1 Background on Classical Methods 31
- 4.1.1 Relevant Functions for Sampling-based Methods. 31
- 4.1.2 Probabilistiy Roadmap (PRM). 32
- 4.1.3 PRM*. 33
- 4.1.4 Fast Marching Tree (FMT). 33
- 4.1.5 Confidence Random Tree (CRT) 33

4.2 Proposed Method. 36
- 4.2.1 Inflated Path. 36
- 4.2.2 Inflated-path PRM (IPRM) and Inflated-path PRM* (IPRM*). 38
- 4.2.3 Improved FMT. 38

4.3 Evaluation. 39
- 4.3.1 Experimental Setup and Maps. 39
- 4.3.2 Polygon Map 40
- 4.3.3 Maze Map with Straight Lines 42
- 4.3.4 U-Map 43
- 4.3.5 Maze Map with Spiral 45
- 4.3.6 Discussion 47

5 EFFICIENT PATH PLANNING METHOD BASED ON THE VORONOIBOUNDARY. 49

5.1 Voronoi Boundary Visibility for Efficient Path Planning 49
- 5.1.1 Connection of the Start and Goal Position. 49
- 5.1.2 Voronoi Boundary Visibility Algorithm. 53
- 5.1.3 VV with Steiner Points. 56
- 5.1.4 Overall Path Computation. 58
- 5.1.5 Inflated Obstacle Region. 61

5.2 Evaluation. 62
- 5.2.1 Properties of the VV Algorithm. 63
- 5.2.2 Comparison with State-of-the-Art Algorithms. 71

6 CONCLUSIONS AND FUTURE WORK 80

APPENDIX. 90

A Comparison of the Solution Paths for VV-ST-R. 91
- A.1 DS = 0. 91
- A.2 DS = 5. 94
- A.3 DS = 10. 95

B Comparison of the Solution Paths with the State-of-the-Art Methods. 98
- B.1 DS = 0. 98

B.2 DS = 5. 99
B.3 DS = DCRT + 1. 113

YK

LIST OF FIGURES

Figure 2.1 Example robot environment. 9
Figure 2.2 Example environment with GVD. 11

Figure 3.1 Environments used for the evaluation: (a) Polygon map (224 × 400); (b) Regular maze map (401 × 400)); (c) T map (293 × 291)); (d) Irregular maze map (544 × 512). 18
Figure 3.2 Comparison of the classical methods. 19
Figure 3.3 Computation of the EGVD: (a) Obstacle map with p_s and p_g; (b) GVD for the obstacle map; (c) pruned GVD; (d) GVD with filled start and goal regions. . 22
Figure 3.4 Desired subsets of $\mathscr{C}_\mathrm{free}$: (a) $\mathscr{C}_\mathrm{S}$; (b) $\mathscr{C}_\mathrm{D}$. 23
Figure 3.5 Pre-processed polygon map with $D_\mathrm{EGVD} = 8$, $D_\mathrm{S} = 6$, $D_\mathrm{D} = 4$: (a) using $\mathscr{C}_\mathrm{S}$; (b) using $\mathscr{C}_\mathrm{D}$.. 25
Figure 3.6 Pre-processed regular maze map with $D_\mathrm{EGVD} = 7$, $D_\mathrm{S} = 4$, $D_\mathrm{D} = 4$: (a) using $\mathscr{C}_\mathrm{S}$; (b) using $\mathscr{C}_\mathrm{D}$. 25
Figure 3.7 Pre-processed T map with $D_\mathrm{EGVD} = 14$, $D_\mathrm{S} = 9$, $D_\mathrm{D} = 7$: (a) using $\mathscr{C}_\mathrm{S}$; (b) using $\mathscr{C}_\mathrm{D}$. 26
Figure 3.8 Pre-processed irregular maze map with $D_\mathrm{EGVD} = 3$, $D_\mathrm{S} = 2$, $D_\mathrm{D} = 2$: (a) using $\mathscr{C}_\mathrm{S}$; (b) using $\mathscr{C}_\mathrm{D}$. 26
Figure 3.9 Comparison of the PRM methods.. 27
Figure 3.10 Comparison of the RRT-based methods. 28
Figure 3.11 Comparison of the most suitable methods. 29
Figure 3.12 Comparison of the standard deviation of the most suitable methods.. . 30

Figure 4.1 Example solution paths: (a) PRM; (b) PRM*; (c) FMT; (d) CRT. . . 35
Figure 4.2 Example environment with different path width: (a) D = (6 pixels) (b) D = (12 pixels) 37
Figure 4.3 Environments used for the evaluation. 39
Figure 4.4 Polygon map: extended GVD and inflated Voronoi boundary. 40
Figure 4.5 Comparison of the performance metrics for the polygon map. 41
Figure 4.6 Solution paths for the polygon map. 41
Figure 4.7 Maze map: extended GVD and inflated Voronoi boundary. 42

Figure 4.8 Comparison of the performance metrics for the maze map with straight lines. 43
Figure 4.9 Solution paths for the maze map with straight lines. 43
Figure 4.10 U-map: extended Voronoi diagram and inflated Voronoi boundary. 44
Figure 4.11 Comparison of the performance metrics for the U-map.. 45
Figure 4.12 Solution paths for the U-map. 45
Figure 4.13 Maze Map with Spiral: extended GVD and inflated Voronoi boundary. 46
Figure 4.14 Comparison of the performance metrics for the map in Fig. 4.3 (d).. 47
Figure 4.15 Solution paths for the maze map with spiral. 47

Figure 5.1 Connection of p_s and p_g to $\mathscr{V}$: (a) GVD; (B) BPs; (c) Connection to BPs on $\mathscr{V}_\mathrm{s}$ and $\mathscr{V}_\mathrm{g}$; (d) Resulting $\mathscr{V}$. 51
Figure 5.2 (a) Original GVD with BPs; (b) Artificial BP b_5. 52
Figure 5.3 Graph for the example environment. 52
Figure 5.4 VV-path illustration for: (a) P_{W_1} with $L(P_\mathrm{VV}) = 657.1$; (b) P_{W_2} with $L(P_\mathrm{VV}) = 677.5$; (c) P_{W_3} with $L(P_\mathrm{VV}) = 755.0$; (d) P_{W_5} with $L(P_\mathrm{VV}) = 840.1$. 55
Figure 5.5 (a) P_VV with $L(P_\mathrm{VV}) = 657.1$; (b) P_RR with $L(P_\mathrm{RR}) = 666.2$. 56
Figure 5.6 (a) Algorithm 6 for P_{W_1} with $L(P_\mathrm{VV}) = 657.1$; (b) Algorithm 6 and 7 for P_{W_1} with $L(P_\mathrm{S}) = 637.0$; (c) Algorithm 6 for P_{W_2} with $L(P_\mathrm{VV}) = 677.5$; (d) Algorithm 6 and 7 for P_{W_2} with $L(P_\mathrm{S}) = 645.4$.. 58
Figure 5.7 (a) Algorithm 9 for P_{W_1} with $L(P_\mathrm{S}) = 635.6$; (b) Algorithm 8 for P_{W_1} with $L(P_\mathrm{S}) = 633.9$; (c) Algorithm 9 for P_{W_2} with $L(P_\mathrm{S}) = 644.6$; (d) Algorithm 8 for P_{W_2} with $L(P_\mathrm{S}) = 631.6$. 61
Figure 5.8 Inflated obstacle region for the example environment: (a) $\mathscr{C}_\mathrm{obs}(5)$ with $L(P_\mathrm{S}) = 636.9$; (b) $\mathscr{C}_\mathrm{obs}(10)$ with $L(P_\mathrm{S}) = 642.4$.. 62
Figure 5.9 Environment maps and corresponding graphs: (a) VV1; (b) VV2. 64
Figure 5.10 Environment maps and corresponding graphs: (a) VV3; (b) VV4 64
Figure 5.11 Path length comparison for VV1. 65
Figure 5.12 Path length comparison for VV2. 65
Figure 5.13 Path length comparison for VV3. 66
Figure 5.14 Path length comparison for VV4. 66
Figure 5.15 Computation time comparison for VV1. 67
Figure 5.16 Computation time comparison for VV2. 67
Figure 5.17 Computation time comparison for VV3. 68
Figure 5.18 Computation time comparison for VV4. 68
Figure 5.19 Solution paths for VV1 and $D_\mathrm{S} = 5$. 69
Figure 5.20 Solution paths for VV2 and $D_\mathrm{S} = 10$. 69
Figure 5.21 Solution paths for VV3 and $D_\mathrm{S} = 5$. 70
Figure 5.22 Solution paths for VV4 and $D_\mathrm{S} = 0$. 70

Figure 5.23 Maps for the comparative evaluation. 72
Figure 5.24 Several interesting cases of solution paths for VV-ST-R, PRM*-L and RR-ST. 76
Figure 5.25 Solution paths of VV-ST-R for different maps. 77
Figure 5.26 Comparison of safe solution paths. 79

Figure A.1 VV1, $D_S = 0$. 91
Figure A.2 VV2, $D_S = 0$. 92
Figure A.3 VV3, $D_S = 0$. 92
Figure A.4 VV4, $D_S = 0$. 93
Figure A.5 VV1, $D_S = 5$. 93
Figure A.6 VV2, $D_S = 5$. 94
Figure A.7 VV3, $D_S = 5$. 94
Figure A.8 VV4, $D_S = 5$. 95
Figure A.9 VV1, $D_S = 10$. 95
Figure A.10 VV2, $D_S = 10$. 96
Figure A.11 VV3, $D_S = 10$. 96
Figure A.12 VV4, $D_S = 10$. 97

Figure B.1 Z1, $D_S = 0$. 98
Figure B.2 Z2, $D_S = 0$ 98
Figure B.3 Z3, $D_S = 0$ 99
Figure B.4 Z4, $D_S = 0$ 99
Figure B.5 Z5, $D_S = 0$ 100
Figure B.6 Z6, $D_S = 0$ 100
Figure B.7 Z7, $D_S = 0$ 101
Figure B.8 Z8, $D_S = 0$ 101
Figure B.9 Z9, $D_S = 0$ 102
Figure B.10 Z10, $D_S = 0$ 102
Figure B.11 Z11, $D_S = 0$ 103
Figure B.12 Z12, $D_S = 0$ 103
Figure B.13 Z13, $D_S = 0$ 104
Figure B.14 Z14, $D_S = 0$ 104
Figure B.15 Z15, $D_S = 0$ 105
Figure B.16 Z1, $D_S = 5$. 105
Figure B.17 Z2, $D_S = 5$. 106
Figure B.18 Z3, $D_S = 5$. 106
Figure B.19 Z4, $D_S = 5$. 107
Figure B.20 Z5, $D_S = 5$. 107

Figure B.21 Z6, $D_\mathrm{S} = 5$. 108
Figure B.22 Z7, $D_\mathrm{S} = 5$. 108
Figure B.23 Z8, $D_\mathrm{S} = 5$. 109
Figure B.24 Z9, $D_\mathrm{S} = 5$. 109
Figure B.25 Z10, $D_\mathrm{S} = 5$. 110
Figure B.26 Z11, $D_\mathrm{S} = 5$. 110
Figure B.27 Z12, $D_\mathrm{S} = 5$. 111
Figure B.28 Z13, $D_\mathrm{S} = 5$. 111
Figure B.29 Z14, $D_\mathrm{S} = 5$. 112
Figure B.30 Z15, $D_\mathrm{S} = 5$. 112
Figure B.31 Z1, $D_\mathrm{S} = D_\mathrm{CRT}$. 113
Figure B.32 Z2, $D_\mathrm{S} = D_\mathrm{CRT}$. 113
Figure B.33 Z3, $D_\mathrm{S} = D_\mathrm{CRT}$. 114
Figure B.34 Z4, $D_\mathrm{S} = D_\mathrm{CRT}$. 114
Figure B.35 Z5, $D_\mathrm{S} = D_\mathrm{CRT}$. 115
Figure B.36 Z6, $D_\mathrm{S} = D_\mathrm{CRT}$. 115
Figure B.37 Z7, $D_\mathrm{S} = D_\mathrm{CRT}$. 116
Figure B.38 Z8, $D_\mathrm{S} = D_\mathrm{CRT}$. 116
Figure B.39 Z9, $D_\mathrm{S} = D_\mathrm{CRT}$. 117
Figure B.40 Z10, $D_\mathrm{S} = D_\mathrm{CRT}$. 117
Figure B.41 Z11, $D_\mathrm{S} = D_\mathrm{CRT}$. 118
Figure B.42 Z12, $D_\mathrm{S} = D_\mathrm{CRT}$. 118
Figure B.43 Z13, $D_\mathrm{S} = D_\mathrm{CRT}$. 119
Figure B.44 Z14, $D_\mathrm{S} = D_\mathrm{CRT}$. 119
Figure B.45 Z15, $D_\mathrm{S} = D_\mathrm{CRT}$. 120

CHAPTER 1

INTRODUCTION

1.1 Background

Nowadays, with the vast developments in the technologies, autonomous and mobile robots [1] are employed in many application fields such as autonomous driving [2, 3, 4], navigation in complex environments [5, 6, 7, 8], vehicle routing [9], buildings [10], industrial automation [11] and Unmanned Air Vehicles (UAVs) [12, 13, 14]. Depending on the application and the task to be solved, robots have to navigate in different types of environments without colliding with obstacles. In static environments, the location of obstacles is known [15, 16, 17, 18, 19, 20, 21, 22, 23, 24], whereas obstacles can change their location in dynamic environments [25, 26, 27, 28, 29]. When applying methods of computational geometry, maps represent objects such as obstacles in the form of geometrical objects such as lines, polygons or circular shapes [30, 31]. On the other hand it is possible to employ binary images, where the free space and obstacles are represented by white and black pixels, respectively [32, 33, 34].

1.2 Path Planning for Mobile Robots

As an important application, path planning for mobile robots has attracted much attention in the recent years [35, 36, 37, 38, 15]. Path planning is concerned with finding a feasible robot path between a start and goal location, while avoiding obstacles in the robot environment [39, 40]. Hereby, the most common performance metrics to validate the quality of solution paths are the path length, the minimum distance to obstacles, which quantifies path safety and the computation time [30, 20].

There are different possible scenarios for robotic path planning depending on the availability of information about the environment [16, 41, 17, 18, 19], the type of obstacles (static or dynamic) [7, 25, 29] and the robot type [40, 37, 42]. In this thesis, we focus on the path planning in static environments, where robot paths are represented by straight-line segments. Such paths are for example suitable for omni-directional robots [42] or can be used as initial solutions when applying path smoothing [43].

1.3 Minimizing the Path Length

The most important performance metric in robotic path planning is the path length. That is, it is highly desired to obtain short paths in order to save travel time and resources. In the recent literature, sampling-based algorithms are most popular for path planning in known environments with static obstacles [15, 16, 17, 18, 19, 20, 21, 22, 23, 24]. A large majority of such algorithms is based on probabilistic roadmaps (PRM) or rapidly exploring random trees (RRT). On the one hand, PRMs generate random sample nodes and introduce connections between close nodes in the obstacle-free region to determine a solution path [16]. On the other hand, RRTs are based on the idea of growing a tree in the obstacle-free region from the start location to the goal location [18]. Moreover, there are various extensions of these algorithms. The PRM* and RRT* algorithms ensure convergence to an optimal path [17] and the FMT algorithm in [19] combines features of PRMs and RRTs to determine shorter solution paths.

In addition, the recent literature provides several methods and comparative studies involving sampling-based methods [21, 44, 45, 46, 47]. Path planning optimization based on the PRM algorithm is presented in [21]. This work proposes new strategies for distributing nodes in order to produce short paths that are comparable with the path produced by PRM. A comparison of different path planning algorithms is performed in [44]. In this work the environment representation is obtained by a SLAM algorithm. Then, different algorithms are applied to find the optimal path such as A*, RRT, bRRT, GA and PRM. The result indicates that A* search could find the shortest path but in needed a long time to find it. An enhanced PRM algorithm used to solve the path planning problem in a static environment was presented in [45]. This work uses the piece-wise Cubic Hermite Interpolation (PCHIP) technique with PRM to enhance the performance of the algorithm. PCHIP takes the points of the path generated by PRM and re-draws the graph to minimize the length of the path. The result shows that the enhanced PRM can find shorter paths than the classic PRM path. A heigh quality solution for Rapidly-exploring Random Trees (RRT) was presented in [46]. Here, the sampling strategy is modified by controlling the sampling space by using a Gaussian mixture model. The performance of this algorithm was compared to the classic RRT, RRT* and informed RRT*. Reference test maps were introduced to examine the performance of PRM and A* algorithms in [47]. The results show that the computation time of A* is long in maze maps and comparable to PRM in polygon maps. Nevertheless, A* is able to produce shorter paths in maze maps and longer paths in polygon maps.

Different from sampling-based methods, [29] uses the generalized Voronoi diagram (GVD) to create a static roadmap and [48] employs the ray-casting and tracking

(RCT) method to solve the problem of global path planning. A comparison with the Visibility graph method, A* and RRT was done to show the efficiency of the algorithm. The aim of the study in [49] is to demonstrate the efficacy of two approaches in path planning, specifically, probabilistic roadmap (PRM) and genetic algorithm (GA). The algorithm executes a test on both simple and complex environments respectively. This work indicates that PRM is faster than genetic algorithm in finding the path while the genetic algorithm has the ability to produce smooth paths for navigation.

1.4 Computation of Safe Robot Paths

A second important performance metric in robotic path planning is path safety. Here, it is desired to keep a sufficient distance from obstacles in order to avoid collisions and account for possible uncertainties during path following. In addition paths with a large clearance to obstacles are beneficial in dynamic environments, where it might be necessary to re-plan paths depending on moving objects [27, 29]. Methods that take into account path safety are frequently based on the generalized Voronoi diagram (GVD) [50, 51, 52, 53], which partitions an environment in Voronoi regions of points that are closest to an obstacle. Then, the Voronoi boundary represents the border of the Voronoi regions such that each point on the Voronoi boundary has the same distance to its closest obstacles. Using the GVD, methods such as [51, 53] guide the search for a solution path along the GVD while applying sampling-based methods in narrow passages [51] or satisfying differential constraints [53]. The method in [52] combines visibility graphs, GVDs and potential fields to obtain short paths. Although these methods make use of the GVD, they do not specifically address path safety. [50] first determines a graph from the GVD and then finds the shortest path in that graph using Dijkstra's algorithm [54]. In this case, the solution path has the largest possible distance to obstacles but can be unnecessarily long. As a remedy, [30] suggests to first refine the shortest path in the GVD by removing unnecessary turns and then introduces additional points in order to shorten the solution path. Differently, [55] applies the fast marching method on an inflated Voronoi boundary.

Artificial Bee Colony ”ABC” with Probabilistic roadmap ”PRM” was presented as a hybrid algorithm in [56]. Whereas PRM is three times faster in finding a path than the ABC algorithm, the hybrid algorithm has the ability to find a fast and safe path. The reason is that PRM generally generates the path very close to obstacles, while ABC can enhance the path by make it smooth and safe. Path planning in static environment for mobile robot was presented in [57]. The work combines three algorithms which are biogeography-based optimization (BBO), particle swarm optimization (PSO) and approximate Voronoi boundary network (AVBN). Based on the paths produced by AVBN the position will be adapted by PSO to increase the diversity of the population in BBO. Since the algorithm searches the path around the branching points of the AVBN diagram it stays away from the obstacles. Tangential escape with A* algorithm is introduced in [58] to solve the navigation problem for mobile robot. The benefit of adding Tangential escape is to avoid obstacles on the robot path by making deviations while the benefit from using A* is to find the shortest path between the two configurations. The global path planning in a multi robot environment is presented in [59]. The GVD with electric circuit based path planning ”ECPP” was presented to solve this problem. Here, the benefit of using the GVD is to obtain a basic roadmap, while the benefit from using ”ECPP” is used for multiple robot path planning. According to the clearance of the path, the robot will chose the way in order to reach the goal. However, this work indicates that this algorithm is not good for single mobile robot because the major idea of this work is how to choose the widest path. A combination of potential fields with the A* algorithm was introduced in [60] too solve the real-time path planning problem for nonholonomic unmanned ground vehicles (UGV). The algorithm is capable of finding a collision free path between the starting and ending point in real-time using the idea of distributing nodes around all the obstacles. A modified version of RRT was presented in [61]. The algorithm is able to reach the goal safely in static environments. The idea of this work is to applying gradient descent on the samples generated by RRT, it will move these samples to the free space. As a result the algorithm provides a collision free path. The Confidence random tree algorithm to find safe paths was presented in [20]. In each iteration, the algorithms determines the clearance of samples from obstacles and computes a confidence value. Based on this confidence. IN addition, the algorithm provides a rejection method for removing unnecessary samples. Path planning in partially known environment using optimized rapidly exploring random tree (ORRT-A*) was presented in [62]. Speeding up the generation of the is tree done by introducing additional step size based RRT, while A* is used to find the shortest path between the starting and ending points. Cubic spline interpolation was used to optimize the path while morphological operations help providing safety. Simulation result indicate that the proposed algorithm succeeds in finding a path comparable with RRT-based and RRT-A* algorithms. A* based on human experience was presented in [63] to solve the path planning problem. The simulation results indicate higher efficiency in finding safe and smooth path between two configurations, comparable with other algorithms like RRT.

1.5 Fast Computation of Robot Paths

Computing robot past quickly is of high importance in the case of real-time path planning and dynamic path planning. Here, it is required to find a solution path within a few seconds after determining the map of the robot environment. In this context, the Quick RRT* (Q-RRT*) algorithm in [22] promises faster convergence to the optimal path as an extension to the classical RRT* algorithm. The idea in this work is to remove intermediate vertexes, i.e. small turns in the path, to reduce the path length. In addition, Q-RRT* combines informed-RRT* to make the tree explore narrow passages. Similarly, the synchronized biased-greedy RRT algorithms in [24] grows trees towards the goal location. A GVD with path optimization was presented in [64]. The article shows the efficiency in finding the path locally and globally by adjusting the weight of the path. Then, Dijkstra's algorithm is used to find the path and the is path optimized using the visibility of the graph. PRM and Spline algorithm was introduced by [45] to find paths for mobile robot in uncertain static environments. The benefit from using PRM is to find the shortest path between two configurations, while the benefit from using Spline curve is to optimize the path which produced by PRM. The result indicate that the hybrid algorithm will produced shorter path with a smaller computation time than the classic PRM. In [65]. the PRM algorithm with a new sampling technique was studied to decrease the computation time in finding two robots paths in the same homotopic roadmap. Sampling based on the boundaries of the obstacle was used to ensure that there are nodes even in narrow corridors to cover all the environment, while PRM is used to find the path between the starting and ending point. Self-localization based on the GVD and Dijkstra's algorithm was used to solve the path planning problem in indoor environments in [66]. The real time result shows that the robot is capable of reaching the goal safely. The comparison of different path planning algorithms in [44] indicates that PRM is the best of the studied algorithms in finding short paths for real-time path planning.

1.6 Discussion and Motivation

The main motivation of this thesis is the observation that there is a large variety of robot path planning methods that focus on different performance metrics as discussed in the previous sections. In this context, it has to be noted that the stated performance metrics are in principle conflicting. That is, obtaining shorter path generally lead to unsafe paths and a longer computation time, whereas solution methods with small computation times will produce longer robot paths. Apart from this fact, it has to be pointed out that the path planning problem for mobile robots is performed either in 3D or 2D space. In particular, when considering nonholonomic mobile robots (that cannot arbitrarily move in any direction), it is important to take into account the robot position (x- and y-direction) as well as the heading angle when planning robot paths. However, when planning paths for robots that can move in any direction (such as omni-directional robots), straight-line paths that only depend on waypoints in 2D-space are suitable. More importantly, such straight-line paths can as well be used as initial solutions for generating smooth paths for nonholonomic mobile robots.

On the one hand, path planning methods for higher-dimensional spaces such as PRM* or RRT* are not as beneficial for the low-dimensional spaces of mobile robots. Although these methods are probabilistically complete (that is, the probability that the planner fails to return a solution decays to zero as the number of samples approaches infinity) and asymptotically optimal (the solution path length converges almost surely to the shortest path length) [17], they do not guarantee finding a short path when putting restrictions on the computation time. Specifically, these methods do not make use of the topology of the robot environment.

On the other hand, the GVD captures the topology of the robot environment in the sense that it provides Voronoi edges that represent points in the robot environment with a maximum distance from obstacles. In particular, it is possible to follow such Voronoi edges from the start position to a desired goal position of a mobile robot. That is, the existence of a solution path is guaranteed if there is a connection along the Voronoi edges. Nevertheless, such solution path is generally unnecessarily long since it includes many turns to keep a maximum distance from obstacles [30].

1.7 Contributions of this Thesis

In view of the above discussion, the main motivation of this thesis is the efficient usage of the topology of the robot environment based on the GVD in order to generate short and safe paths for mobile robots with a small computation time. To this end, the thesis suggests three different approaches. The first approach is based on the idea of pre-processing the environment map using information from the GVD. The aim of pre-processing is to add obstacle points in regions of the environment where solution paths are undesired. Then, we apply existing sampling-based methods to the modified environment map. Since the free space for solution paths is reduced, solution paths are expected to be safer and less samples are needed in order to obtain short paths. Since the computation time of the sampling-based methods depends on the number of samples, this further reduces the computation time. The second approach develops a new method for generating samples in a pre-processed environment map. Different from the first approach, knowledge about the GVD is used to directly place samples only in regions that are considered as safe. This approach leads to further savings in the computation time. At this point, we note that the performance of the previous approaches still depends on the placement of random samples. Differently, the third approach does not use sampling-based methods. Instead, a new iterative refinement of initial solutions paths along Voronoi edges of the GVD is proposed. As the first step, the refinement removes waypoints in case it is possible to introduce an obstacle-free shortcut. As the second step, additional waypoints are introduced in order to cut corners that are generated by the edges between waypoints. This procedure is then iteratively applied in order to gradually cut corners and introduce shortcuts. As an important feature, the third approach is able to find a solution path whenever it exists.

In the scope of this thesis, all proposed approaches are evaluated based on a comprehensive comparison with existing state-of-the-art methods. In these comparative experiments, it is shown that all the methods are able to generate short and safe paths with a small computation time and improvements compared to the existing algorithms.

1.8 Thesis Organization

This Thesis is organized as follows. Chapter 2 introduces the necessary background for the methods developed in this thesis. Chapter 3 proposes a new method for sampling-based path planning based on pre-processed environment maps. Chapter 4 develops an improved sampling strategy for fast and safe path planning and Chapter 5 introduces an iterative method for refining paths obtained from the generalized Voronoi diagram. Conclusions and directions for future work are presented in Chapter 6.

CHAPTER 2

BACKGROUND

This chapter gives the necessary background information for the methods proposed in this thesis. First, basic definitions regarding the robot environment and the representation of solution paths are given in Section 2.1. In addition, Section 2.2 recalls important formal properties of robot paths.

2.1 Basic Definitions

We next consider the basic definitions regarding to path planning.

2.1.1 Notation

The subject of this work is the path planning for mobile robots in two-dimensional (2D) static environments with obstacles. Hereby, we focus on the generation of paths that consist of straight-line segments. Such paths can for example be followed by omni-directional robots, which are able to turn on the spot [42] or can be used as a starting point for generating smooth robot paths [43]. Formally, the *configuration space* is defined as $\mathscr{C} \in \mathbb{R}^2$ and obstacles in $\mathscr{C}$ are represented by the *obstacle region* $\mathscr{C}_{\text{obs}} \subseteq \mathscr{C}$. Accordingly, the *obstacle-free region* that is available for the robot motion is determined as $\mathscr{C}_{\text{free}} = \mathscr{C} \setminus \mathscr{C}_{\text{obs}}$. Fig. 2.1 shows an illustration of the previously defined regions with an obstacle region that consists of three circular obstacles that should not be hit by the mobile robot.

Any point $p \in \mathscr{C} \subseteq \mathbb{R}^2$ can be represented by its coordinates x and y. That is, we write $p = (x, y)$ as shown in Fig.2.1. Considering that we are interested in straight-line paths in this thesis, a *robot path P* is defined by sequence of n points $p_1, p_2, \dots, p_n$ in $\mathscr{C}$. Hence, we write $P = (p_1, p_2, \dots, p_n)$, whereby $p_i \in \mathscr{C}$ for $i = 1, \dots, n$. Defining the start and goal position of a mobile robot as p_{s} and p_{g}, respectively, it must hold that $p_1 = p_{\text{s}}$ and $p_n = p_{\text{g}}$ for any suitable robot path. The actual robot path is then determined by connecting subsequent points p_i and p_{i+1} of a path P by straight lines $l_{p_i, p_{i+1}}$ for each $i = 1, \dots, n-1$. Then, the set of points traversed by the robot from p_{s} to p_{g} is given by $\mathscr{P}_P \subseteq \mathscr{C}$, whereby $\mathscr{P}_P$ consists of all the points that are covered by the line segments $l_{p_1, p_2}, \dots, l_{p_{n-1}, p_n}$. Accordingly, we denote a path P as *collision-free* if

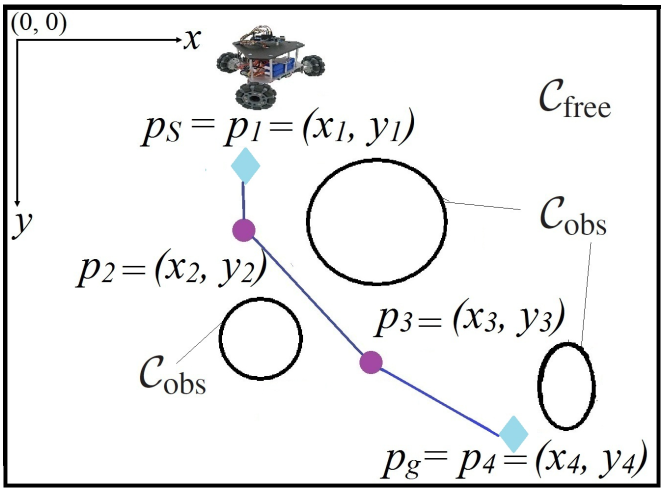


Figure 2.1: Example robot environment.

$\mathscr{P}_P \cap \mathscr{C}_{\text{obs}} = \emptyset$, that is, there is no intersection of the points covered by the path and the obstacle region. For later use in different algorithms, we also introduce the function

$$\texttt{CollisionFree}(p, \hat{p}, \mathscr{C}_{\text{obs}}) = \begin{cases} \textbf{true} & \text{if } l_{p,\hat{p}} \cap \mathscr{C}_{\text{obs}} = \emptyset \\ \textbf{false} & \text{otherwise} \end{cases}$$

that determines if the straight-line connection $l_{p,\hat{p}}$ between two points p and $\hat{p}$ intersects the obstacle region $\mathscr{C}_{\text{obs}}$.

We write

$$\mathscr{A} = \{P | \mathscr{P}_P \subseteq \mathscr{C}_{\text{free}}\} \tag{2.1}$$

for the set of obstacle-free paths. We further introduce the distance between two points $p_i = (x_i, y_i), p_j = (x_j, y_j) \in \mathscr{C}$ as

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{2.2}$$

the minimum distance between a point $p \in \mathscr{C}$ and a subset $\mathscr{C}' \subseteq \mathscr{C}$

$$d(p, \mathscr{C}') = \min_{p' \in \mathscr{C}'} d(p, p') \tag{2.3}$$

and the minimum distance between two subsets $\mathscr{C}', \mathscr{C}'' \subseteq \mathscr{C}$

$$d(\mathscr{C}', \mathscr{C}'') = \min_{p' \in \mathscr{C}', p'' \in \mathscr{C}''} d(p', p''). \tag{2.4}$$

Using (2.4), the minimum distance of a path P from the obstacle region can be written as $d(\mathscr{P}_P, \mathscr{C}_{\text{obs}})$. Finally, we compute the *path length* of P as

$$L_P = \sum_{i=1}^{n-1} d(p_i, p_{i+1}). \tag{2.5}$$

Using the notation introduced above, the main aim of this thesis is the computation of suitable obstacle-free robot paths between a given start position $p_{\text{S}} \in \mathscr{C}_{\text{free}}$ and goal position $p_{\text{G}} \in \mathscr{C}_{\text{free}}$. In this context, we characterize suitability of robot paths by performance metrics such as the path length (finding the shortest path), path safety (finding a path distant from obstacles) and the computation time.

2.1.2 Generalized Voronoi Diagram

The results presented in this thesis are based on the usage of the generalized Voronoi diagram (GVD), which is a basic data structure in robotic path planning [30, 25, 51, 52, 53]. In order to formalize the related terminology, we consider a configuration space $\mathscr{C}$ that contains a set of geometric objects $\mathscr{O}_1, \mathscr{O}_2, \ldots, \mathscr{O}_m$ such that $\mathscr{O}_i \subseteq \mathscr{C}$ for $i = 1, \ldots, m$ as illustrated in Fig. 2.2.

Each object $\mathscr{O}_i$ is associated to a Voronoi region $\mathscr{V}_i$. Specifically, $\mathscr{V}_i$ defines the set of all points $p \in \mathscr{C}$ that are closer to $\mathscr{O}_i$ than to any other object $\mathscr{O}_j$ with $i \neq j$ [51, 52, 53]. Formally, we define

$$\mathscr{V}_i = \{p \in \mathscr{C} | d(p, \mathscr{O}_i) \leq d(p, \mathscr{O}_j), \forall j \neq i\}. \tag{2.6}$$

Using (2.6), the collection of all regions $\mathscr{V}_1, \ldots, \mathscr{V}_m$ is denoted as the *generalized Voronoi diagram* (GVD). As can be seen in Fig. 2.2, adjacent Voronoi regions share a border that consists of all points with an equal distance to at least two objects. Accordingly, we write $\mathscr{V}$ for the set of all such points and we call $\mathscr{V}$ the *Voronoi boundary* (VB). Formally,

$$\mathscr{V} = \{p \in \mathscr{C} | \exists i \neq j, d(p, \mathscr{O}_i) = d(p, \mathscr{O}_j)\}. \tag{2.7}$$

That is, for each point $p \in \mathscr{V}$, there are at least two different objects $\mathscr{O}_i, \mathscr{O}_j$ with $i \neq j$ with an equal distance to p. In particular, any robot path following the VB keeps a maximum distance from the objects $\mathscr{O}_i$, $i = 1, \ldots, m$, which can be identified as obstacles. In addition, we denote points, where multiple borders of the VB meet as *branching points* (BPs). To this end, we define the set of BPs $\mathscr{B}$ as the set of all points on the boundary of at least 3 Voronoi regions.

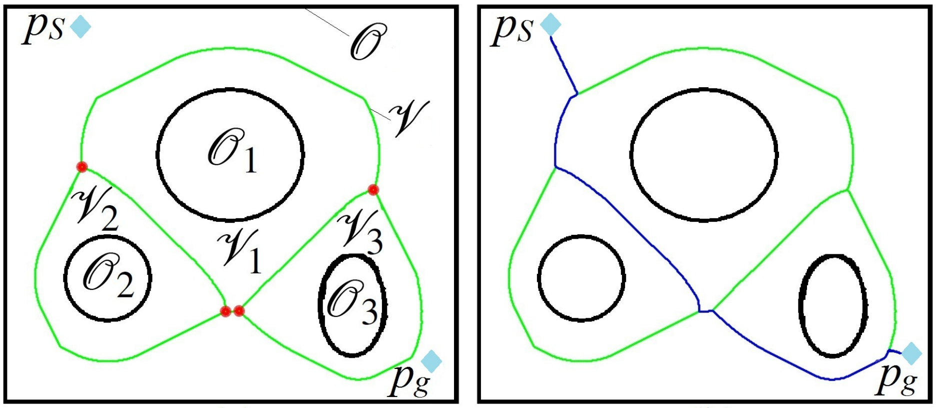
$$\begin{aligned} \mathscr{B} = \{&p \in \mathscr{V} | \exists i, j, k, i \neq j \neq k, \\ &d(p, \mathscr{O}_i) = d(p, \mathscr{O}_j) = d(p, \mathscr{O}_k)\}. \end{aligned} \tag{2.8}$$

Finally, we denote parts of the VB between BPs as *segments* of the VB. Considering two BPs $b_i, b_j \in \mathscr{B}$, we write $\mathscr{P}_{b_i,b_j} \subseteq \mathscr{V}$ for the points covered by the segment between b_i and b_j and consider $|\mathscr{P}_{b_i,b_j}|$ as the length of the segment.

There are various studies for the computation of the GVD in the existing literature [67, 68, 69, 30, 51, 52, 70]. The underlying assumption in this thesis is that the robot environment is represented by a binary image. That is, the relevant regions $\mathscr{C}_{\text{free}}$ and $\mathscr{C}_{\text{obs}}$ are not identified by geometric objects but by the pixel color such that obstacle pixels are black and the free space is characterized by white pixels as demonstrated in Fig. 2.2 (a). We note that the focus of this thesis is not the computation of GVDs. Hence, we employ the fact that GVDs can for example be obtained based on the medial axis transform [71, 72, 73]. Accordingly, we use the morphological operation of ”skeletonization” as in [74, 75, 76] to determine an approximation of the VB that consists of the image pixels on the medial axis with an equal minimum distance to obstacle pixels (see (2.7)). Then, BPs are determined as image pixels that are connected to at least three segments of the VB.

2.1.3 Voronoi Diagram and Dijkstra’s Algorithm

The VB $\mathscr{V}$ of the GVD can be used to determine robot paths with a maximum distance/clearance from the obstacles [70, 51, 52, 55, 77, 78, 30, 50, 25, 29]. Hereby, given start and goal points p_{s} and p_{g}, it is first required to connect these points to the VB in order to obtain a connection between p_{s} and p_{g} via $\mathscr{V}$. Generally, such connection is achieved by computing the shortest collision-free path from p_{s} and p_{g} to $\mathscr{V}$ [50, 30, 25, 53, 79] and extending the GVD by these paths. Then, Dijkstra’s algorithm [54, 80, 50] can be applied to determine the shortest path between p_{s} and p_{g} in this extended GVD as depicted in Fig. 2.2 (b).



(a) (b)

Figure 2.2: Example environment with GVD.

2.2 Completeness and Optimality

The literature provides a multitude of algorithms for robotic path planning. These algorithms are generally classified according to the properties of their solution paths [81]. We next summarize the most relevant properties for later use in the thesis.

An algorithm is said to be *complete* if it terminates in a finite time with a valid solution if such solution exists or with failure if there is no solution to the path planning problem. Moreover, an algorithm is called *optimal* if any returned solution is optimal with respect to a given cost function such as the path length [17], [19]. In this thesis, we focus on *sampling-based algorithms* such as PRM [16], PRM* [17], RRT [18], RRT* [17] or FMT [19] due to their computational efficiency. Although sampling-based algorithms are neither complete nor optimal, they can give completness and optimality guarantees in a probabilistic sense. The related notions are *probabilistic completeness* and *asymptotic optimality*. An algorithm is probabilistically complete if the probability that the planner fails to return a solution decays to zero as the number of samples approaches infinity in case a solution exists. In addition, an algorithm is asymptotically optimal if the cost of the returned solution converges almost surely to the optimal cost. Since a formal analysis of these notions is not required in the scope of this thesis, we refer to formal definitions of these notions in [17].

CHAPTER 3

PRE-PROCESSING ENVIRONMENTS FOR EFFICIENT SAMPLING

This chapter develops the first original approach of this thesis. In order to put this approach into perspective Section 3.1 first performs a comparative analysis of various state-of-the-art methods for robotic path planning. Then, Section 3.2 introduces our methodology and compares its performance to the existing methods.

3.1 Comparison of Existing Methods

The literature provides a vast number of algorithms for path planning with different objectives such as minimizing the path length, obtaining safe paths or finding a path with minimum computation time. Since the main aim of this thesis is the development of a methodology for safe and fast path planning, we first determine important features and weaknesses of existing algorithms. To this end, Section 3.1.1 to 3.1.3 give a brief description of the relevant methods. Section 3.1.4 performs a comprehensive comparative evaluation of these methods using different environments and discusses their performance regarding path length, safety and computation time. We note that the description of the relevant algorithms rather focuses on the basic features of the algorithms. We refer to the referenced literature for a full formal description of these algorithms.

3.1.1 Methods for Minimizing the Path Length

We next consider the most popular algorithms for computing short paths.

Probabilistiy Roadmap (PRM)

The PRM algorithm is one of the most popular algorithms for robotic path planning [41, 16], Although the method is in principle developed for multi-query applications, it is as well beneficial for single-query applications [82]. The PRM algorithm has two phases. In the first (learning) phase, it creates a graph by successively generating random nodes in $\mathscr{C}_{\text{free}}$ and tries to connect theses nodes to existing nodes by straight lines. Connections are attempted to nodes within a certain connection radius and without intersections with $\mathscr{C}_{\text{obs}}$. The learning phase is complete when a pre-defined number of N

nodes has been processed. In the second (query) phase, the algorithm tries to connect the given start point p_{s} and goal point p_{g} to the roadmap from the first phase and then returns the shortest path in the resulting graph as the solution path.

The PRM algorithm is reported to be very fast and is able to generate short paths [17]. In addition, the PRM algorithm is proven to be probabilistically complete [83] without being asymptotically optimal. In addition, the PRM algorithm does not include any measure for path safety.

PRM*

The PRM* algorithm was proposed as a modified Version of PRM in [17]. It is different from the classical PRM algorithm in the first phase by decreasing the connection radius between nodes with an increasing number of nodes. As a result, the PRM* algorithm is both probabilistically complete and asymptotically optimal. The PRM* algorithm does not address path safety and has a slightly increased computational effort compared to the PRM algorithm.

Rapidly Exploring Random Tree (RRT)

The rapidly exploring random tree (RRT) algorithm was developed in [18] for efficient single-query applications in high dimensional environments. The RRT algorithm grows a tree starting from the given start position p_{s} towards the unvisited part of the free region. In each iteration of the algorithm, a random node p_{rand} is generated. This node is connected to the closest existing node p_{near} if the connection is collision free. Otherwise, a new node x_{new} with a collision-free connection is computed based on p_{rand}. The RRT algorithm terminates if a path between p_{s} and the goal position p_{g} has been found or a pre-defined maximum number of samples N_{RRT} is exceeded.

The RRT algorithm is probabilistically complete [18] but not asymptotically optimal. Moreover, the RRT algorithm does not address safety.

RRT*

The RRT* algorithm was proposed in [17] as a modified version of the RRT algorithm. Different from the RRT algorithm, edges for a new node p_{new} are not only inserted to the nearest node p_{near} but to all nodes inside a ball, whose radius decreases with the number of explored nodes. In addition, the RRT* algorithms removes redundant edges that cannot lead to a shortest path.

The RRT* algorithm is both probabilistically complete and asymptotically optimal [17]. The RRT* algorithm does not address path safety and has an increased computational effort compared to the RRT algorithm.

Fast Marching Tree (FMT)

The FMT algorithm was proposed in [19] to combine the advantages of RRT* and PRM*. It keeps sets of closed, open and unvisited nodes, whereby each node in the open set is associated to the cost of traveling to that node. In each iteration, the FMT algorithm selects the node with the lowest cost in the open set and finds all of its neighbors in the unvisited set. Similar to PRM* and RRT*, neighbors are defined based on a connection distance that decreases with the number of iterations. The neighbors found are then connected to the closest nodes in the open set. The selected node is removed from the open set and added to the closed set. The algorithm terminates successfully if a connection to the goal node is established and is unsuccessful if a pre-defined number of iterations is reached (or the unvisited set becomes empty).

According to [19], the FMT algorithm is asymptotically optimal but it does not address path safety.

3.1.2 Methods for Path Safety

Path safety is generally addressed by increasing the minimum distance of a planned path to the obstacle region, while avoiding a considerable increase in the path length [20]. Path safety is of utmost importance in the case of uncertainties in the environment or the path following capabilities of the mobile robot. In addition, path safety contributes to the applicability of path planning algorithms in dynamic environments [29].

Voronoi Diagram and Dijkstra's Algorithm

GVDs can be used for path planning in order to obtain a path with a maximum distance from the obstacle region [70, 51, 52, 55, 30]. Hereby, the existing methods generally proceed as follows. First, the GVD of the environment is generated as explained in Section 2.1.2 and is extended by the shortest obstacle-free connection of the start and end position to the GVD [25, 53, 79]. Then, a graph is extracted from this extended GVD such that the start position, goal position and BPs correspond to vertexes. Edges are introduced between vertexes that are connected by the Voronoi boundary and are labeled with the connection distance. Finally, Dijkstra's algorithm [54] is applied to this graph to determine the shortest path in the extended GVD. The described algorithms are complete but generally produce long paths.

Medial Axis PRM (MAP)

A possible shortcoming of the classical PRM algorithm is that it places nodes randomly in $\mathscr{C}_{\text{free}}$. As a result, it is unlikely that a sufficient number of nodes is placed in narrow passages, which makes it difficult to find solution paths in such environments. In addition, nodes might be generated very close to $\mathscr{C}_{\text{obs}}$, affecting path safety. As a remedy, medial axis PRM (MAPRM) is proposed in [84]. The main idea of MAPRM is to generate nodes in the close vicinity of the Voronoi boundary, denoted as medial axis (MA) of $\mathscr{C}_{\text{free}}$. The MAPRM algorithm first generates random nodes similar to PRM. In a second step, these nodes are retracted towards the MA in an iterative procedure without explicitly computing the GVD. Although the MAPRM algorithm enables passing narrow passages, it has two possible disadvantages. First, the iterative procedure for retracting nodes to the MA increases the computational effort. Second, while it is ensured that nodes are generated far from $\mathscr{C}_{\text{obs}}$, the connecting lines can still be close to obstacles, which decreases path safety.

Medial Axis RRT (MAR)

Following the same line of argument as in the previous section (for MAPRM), MARRT is introduced as an improvement to RRT in order to grow the tree in the vicinity of the MA [85]. The MARRT algorithm iteratively generates a random node and then expands the tree from the nearest node towards the random node along the MA. Different from MAPRM, the MARRT algorithm is probabilistically complete under mild assumptions. Otherwise, the MARRT algorithm has the same advantages and disadvantages as the MAPRM algorithm.

Confidence Random Tree (CRT)

The confidence random tree (CRT) algorithm is specifically designed for path safety [20]. Instead of generating random nodes, the CRT algorithm expands a tree by generating new nodes at a pre-computed distance from previously accepted nodes starting from the start node. Hereby, the distance is adjusted based on the confidence of the selected node, which depends on its distance from $\mathscr{C}_{\text{obs}}$. In order to limit the number of nodes, the CRT algorithm includes a node rejection method to avoid generating nodes in previously explored areas. Due to the consideration of node confidence, the CRT algorithm generates a safe path at the expense of an increased path length. A shortcoming of the CRT algorithm is the possibility of rejecting necessary nodes, which can lead to longer paths or failure to find a solution.

3.1.3 Methods for Fast Computation

Fast computation of solution paths is of particular importance in real-time applications [86]. Accordingly, several algorithms aim at the fast computation of solutions with a possible loss of completeness and optimality. We next summarize two such extensions of the RRT algorithm.

Obstacle Directed RRT (ObsRRT)

The ObsRRT algorithm is introduced as an extension of RRT for high-dimensional environments in [87]. Since this thesis considers 2D-environments, our explanation focuses on this case. Similar to RRT, the ObsRRT algorithm expands a tree from a nearest node towards a randomly generated node if the nearest node is far away from $\mathscr{C}_{\text{obs}}$. On the other hand, if the nearest node is close to $\mathscr{C}_{\text{obs}}$, ObsRRT tries to expand the tree parallel to the obstacle boundary. This measure leads to a reduced computation time especially in environments with narrow passages. However, ObsRRT does not provide any guarantees regarding path safety, path length, completeness and optimality.

Goal-biased RRT (GBRRT)

The goal-biased RRT (GBRRT) algorithm is a modification of RRT with the aim of growing the tree towards the goal fast [88]. The GBRRT algorithm defines a goal-bias probability p_{GB} and draws a random number before generating a new sample. If the random number is below p_{GB}, a normally distributed random sample around the goal is generated. Otherwise, uniform sampling is used as in the original RRT algorithm [18]. The main advantage of the GBRRT algorithm is the fast computation of a solution path, whereby it has to be noted that the path is generally longer than the minimum path, path safety is not taken into account and completeness and optimality are not addressed.

3.1.4 Evaluation

We next perform a comparison of the described algorithms based on different environments. All algorithms were implemented in Matlab using the same functions for common tasks of the different algorithms such as computing the distance of points from $\mathscr{C}_{\text{obs}}$. The experiments were run on a personal computer with Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz and 8.00 GB RAM. Here, we note that this section does not present novel results regarding the stated algorithms. The aim of this section is to compare the algorithms regarding their path length, safety and computation time. This comparison is then the basis for the development of the proposed methodology in Section 3.2.

Environments and Parameters

We consider the environments in Fig. 4.3 which are given as binary images, where pixels in $\mathscr{C}_{\text{free}}$ are white and pixels in $\mathscr{C}_{\text{obs}}$ are black. The start position and the goal position are shown by a green circle and red diamond, respectively.

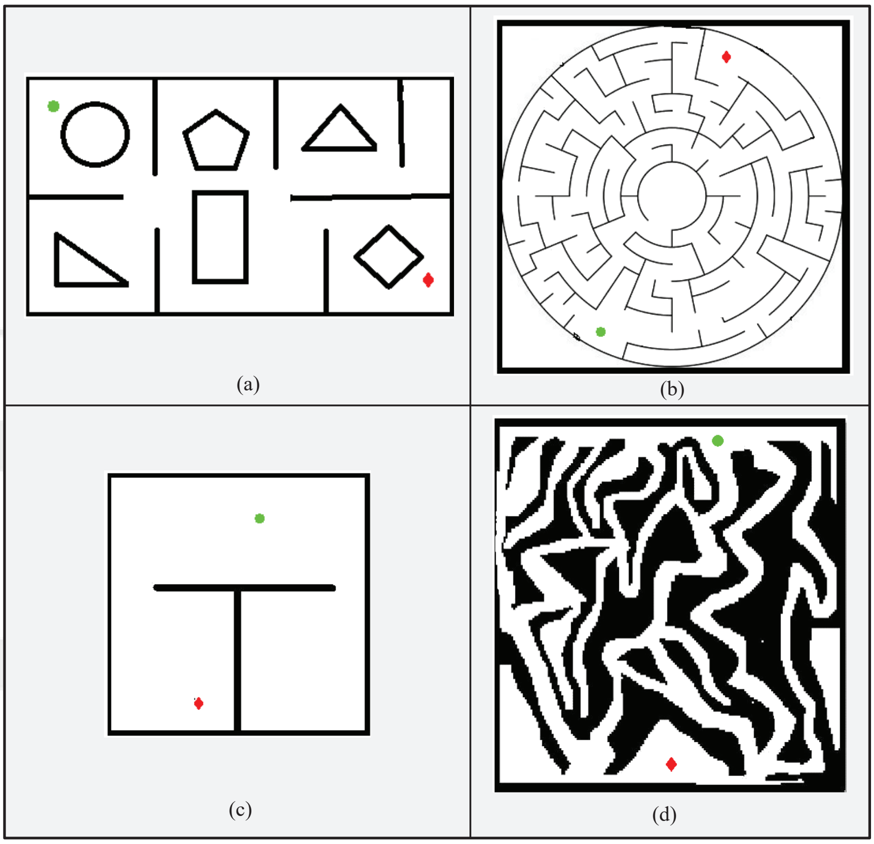

(a)
(b)
(c)
(d)

Figure 3.1: Environments used for the evaluation: (a) Polygon map (224 × 400); (b) Regular maze map (401 × 400)); (c) T map (293 × 291)); (d) Irregular maze map (544 × 512).

The maps are selected according to their different properties as follows. The polygon map in Fig. 3.1 (a) has different obstacles that are represented by polygon shapes and that leave sufficient free space. There are multiple alternative routes between the start and goal position. In the regular maze map in Fig. 3.1 (b), obstacles are represented by straight lines or circular lines, whereby the distance between obstacles is small. Although there is a unique route between the start and end position, there are many potential routes to be explored by sampling-based algorithms. The T map in Fig. 3.1 (c) offers free space both towards the goal position and away from the goal position. In addition, there is a comparatively narrow passage when traveling on the unique route from the start to the goal position. The irregular maze map in Fig. 3.1 (d) provides a multitude of narrow and irregular routes between the start and goal position.

Computational Experiments

We next evaluate the computational experiments for the different environments in Fig. 3.1 and the described algorithms. Since the considered algorithms are based on random sampling, we take the average of 100 trials for each of the algorithms for each parameter. In order to quantify the performance of the solution paths P for each algorithm, we determine the path length L_P, the minimum distance to the obstacle $d(\mathscr{P}_P, \mathscr{C}_{\text{obs}})$ and the computation time T. The experimental results are shown in Fig. 3.2. We again recall that all algorithms are applied for the case of a 2D configuration space which is suitable for omnidirectional robots.

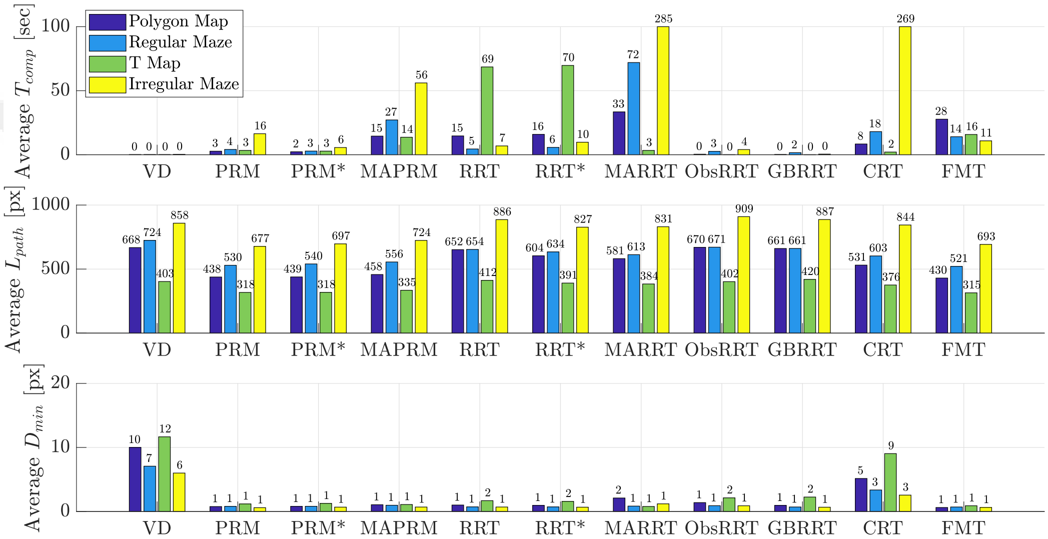


Figure 3.2: Comparison of the classical methods.

Regarding the computation time, it can be observed that VD, ObsRRT and GBRRT generate solution paths in the shortest time for all environments. The PRM-based algorithms are generally faster than the RRT-based algorithms with the exception of the irregular maze map. MA-based algorithms mostly show the largest computation times, which is expected due to the computationally expensive retraction of nodes to the MA. Although CRT has a faster computation time than the RRT-based algorithms for the environments with free spaces (polygon map and T map), it leads to long computation times in maze environments, since too many nodes are generated along alternative routes.

When analyzing the path length, it can be seen that the shortest paths are achieved by FMT and the PRM-based algorithms. The RRT-based algorithms including CRT generally lead to longer solution paths. In particular, algorithms such as ObsRRT and GBRRT generate the longest path due to their fast termination. Solution paths are as well long for VD since they strictly follow the Voronoi boundary.

Figure 3.2 clearly indicates that path safety is only taken into account by VD and

CRT. This is expected since both algorithm explicitly consider the obstacle distance during the path computation. The remaining algorithms provide poor results regarding path safety. Even MAPRM and MARRT try to place nodes close to the MA, these algorithms cannot ensure a sufficient distance of the solution paths to the obstacle region.

In addition to the above parameters, Table 3.1 shows the number of nodes and the success rate of finding a solution path for each algorithm. Hereby, we note that the number of nodes for each algorithm was adjusted so as to obtain a meaningful success rate without requiring an extensive computation time. Accordingly, the table shows that most of the algorithms provide a high success rate, whereby the required number of nodes is closely related to the resulting computation time. Only the MA-based algorithms show a large computation due to the computationally expensive retraction of nodes to the MA. It is further interesting to inspect the results for the regular maze map. Here, algorithms such as PRM*, MAP and CRT show a low success rate, which is due to the required exploration of unnecessary parts of the configuration space.

Table 3.1: Success rate and number of nodes for the classical methods on the polygon map.

	VD	PRM	PRM*	MAP	RRT	RRT*	MAR	ObsR	GBR	CRT	FMT
	Polygon Map										
R_{suc}	100	100	100	100	100	100	100	100	100	100	92
N_{nodes}	589	1000	1000	500	2910	2900	357	804	537	1200	3000
	Regular Maze Map										
R_{suc}	100	100	79	60	96	100	100	100	97	82	95
N_{nodes}	602	1500	1500	500	2580	2650	343	2900	839	2020	3000
	T Map										
R_{suc}	100	100	100	100	100	100	100	100	100	99	97
N_{nodes}	342	1000	1000	500	2810	2810	624	583	343	396	3000
	Irregular Maze Map										
R_{suc}	100	100	98	97	100	100	100	100	100	98	98
N_{nodes}	728	2500	2500	500	6080	6050	373	1350	692	4050	3000

Discussion of Existing Methods

Together, the comparison in the previous section indicates that

- FMT leads to the shortest paths but without any guarantee of safety and with a possibly large computation time,
- PRM-based methods perform better than RRT-based methods regarding computation time and path length,

- path safety is not taken into account by any of the methods except for VD and CRT. Hereby, VD generates long paths and CRT leads to large computation times and can have a low success rate especially in difficult environments,
- MAPRM and MARRT that generated nodes around the MA do not contribute to path safety since the connections between node can come close to the obstacle region and they have large computation times,
- the only method giving formal safety guarantees is VD. Consider a solution path P computed by the VD algorithm. Then, it holds that its minimum distance from $\mathscr{C}_{\text{obs}}$ is lower bounded by the minimum distance of the Voronoi boundary from the obstacle region:

  $$d(\mathscr{P}_P, \mathscr{C}_{\text{obs}}) \geq d(\mathscr{V}, \mathscr{C}_{\text{obs}}). \tag{3.1}$$

  Differently, it is the case that, although the CRT algorithm achieves a safer path than the other methods, there is no formal guarantee about the minimum distance.

In summary, none of the existing methods can achieve short paths with a small computation time and provide formal guarantees regarding path safety. Accordingly, the main aim of this thesis is the development of a methodology that combines the advantages of the existing methods in order to compute solution paths with the described properties. The next section proposes such methodology based on the idea of pre-processing the given environment map.

3.2 Methodology for Safe Path Planning

This section develops a general methodology for pre-processing environment maps that are suitable for the fast computation of short and safe solution paths. As a common feature, these pre-processed environment maps enable the application of any of the sampling-based path planning algorithms. As the first novel idea of this thesis, Section 3.2.1 describes a modification of the GVD that is the basis for the pre-processing methodology. Then, Section 3.2.2 formalizes the desired safety guarantees and describes the computation of the pre-processed environment map. Finally, Section 3.2.3 evaluates the performance of different sampling-based algorithms on the pre-processed maps.

3.2.1 Extended Generalized Voronoi Diagram

In general, the start and goal position (p_{s} and p_{g}) are not located on the Voronoi boundary. As described in Section 3.1.2, the existing methods suggest to connect these points to the Voronoi boundary with a shortest obstacle-free path. In this section, we propose a different modification of the Voronoi boundary that will serve as the first preprocessing step of our methodology. To this end, we propose the following procedure to determine an extended GVD (EGVD), which is also illustrated in Fig. 3.3.

1. Extend $\mathscr{C}_{\text{obs}}$ by the start and goal position: $\mathscr{C}_{\text{obs}} = \mathscr{C}_{\text{obs}} \cup \{p_{\text{s}}, p_{\text{g}}\}$ (Fig. 3.3 (a)),
2. Compute the GVD for the resulting $\mathscr{C}_{\text{obs}}$ (Fig. 3.3 (b)),
3. Prune the GVD by removing parts that cannot be part of a solution path (Fig. 3.3 (c)),
4. Mark the regions around the start and goal position as free space (Fig. 3.3 (d)).

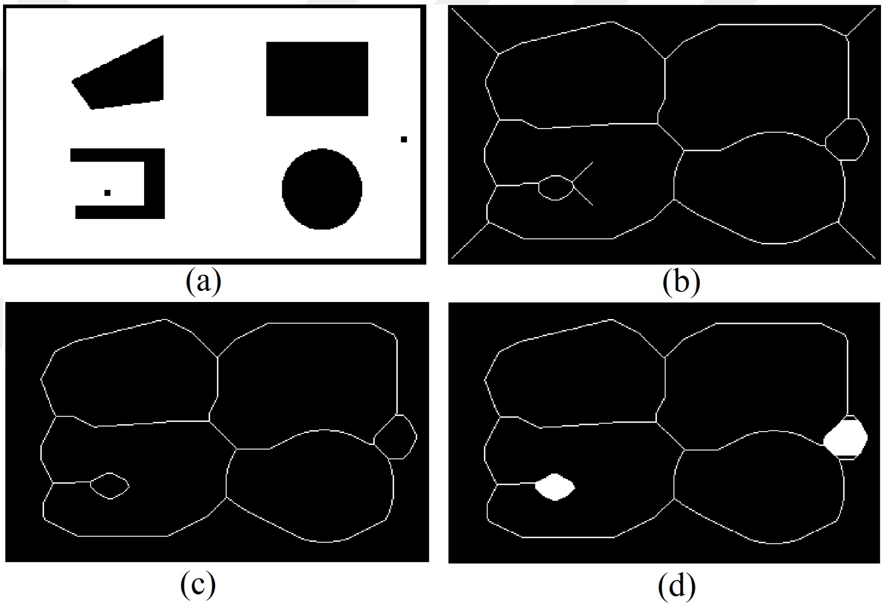

(a) (b) (c) (d)

Figure 3.3: Computation of the EGVD: (a) Obstacle map with p_{s} and p_{g}; (b) GVD for the obstacle map; (c) pruned GVD; (d) GVD with filled start and goal regions.

We next discuss the steps of the proposed procedure. Since p_{s} and p_{g} lie in the obstacle-free region and are marked as $\mathscr{C}_{\text{obs}}$ (step 1), the GVD will encircle these points, whereby all points inside the circle must belong to $\mathscr{C}_{\text{free}}$ (step 2). That is, any solution path must start within the circle around p_{s} and end in the circle around p_{g}. Accordingly, all parts of the GVD that terminate at open ends cannot be part of a solution path and hence can be pruned as suggested in [52] (step 3). Furthermore, marking the regions around p_{s} and p_{g} as $\mathscr{C}_{\text{free}}$ enables reaching p_{s} and p_{g} from the remaining part of the GVD (step 4).

3.2.2 Safety Guarantees and Proposed Methodology

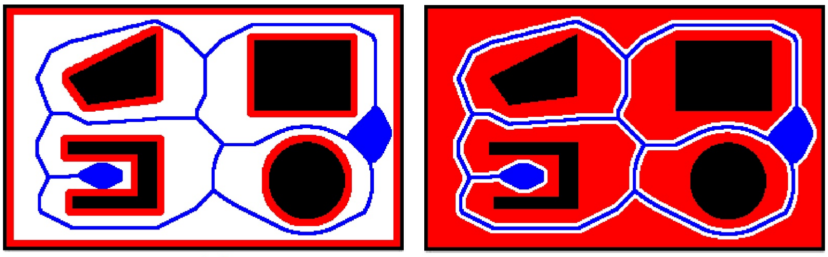
We next formulate subsets of $\mathscr{C}_{\text{free}}$ that we consider as desirable for solution paths of the robotic path planning problem when considering both path length and safety. Writing $\mathscr{V}_{\text{EGVD}}$ for the Voronoi boundary of the EGVD, we define the minimum distance of the EGVD from $\mathscr{C}_{\text{obs}}$ as

$$D_{\text{EGVD}} = d(\mathscr{V}_{\text{EGVD}}, \mathscr{C}_{\text{obs}}). \tag{3.2}$$

Based on D_{EGVD} and assuming that a solution path for the path planning problem exists in the EGVD[1], we identify two interesting subsets of any given environment. First, given a safety distance $D_{\text{S}} < D_{\text{EGVD}}$, the set $\mathscr{C}_{\text{S}}$ consists of all points, whose distance from $\mathscr{C}_{\text{obs}}$ is larger than D_{S}:

$$\mathscr{C}_{\text{S}} = \{p \in \mathscr{C}_{\text{free}} | d(p, \mathscr{C}_{\text{obs}}) > D_{\text{S}}\}. \tag{3.3}$$

The set $\mathscr{C}_{\text{S}}$ is illustrated in Fig. 3.4 (a), whereby the set $\mathscr{C}_{\text{S}}$ is shown in white, the original obstacles are black, the inflated obstacles are red and the EGVD is blue. As an important property, the set $\mathscr{C}_{\text{S}}$ fully contains the Voronoi boundary $\mathscr{V}_{\text{EGVD}}$ of the EGVD since $D_{\text{S}} < D_{\text{EGVD}}$. That is, assuming that a solution path exists in $\mathscr{V}_{\text{EGVD}}$, it is ensured that a solution path exists in $\mathscr{C}_{\text{S}}$. Moreover, according to (5.4), any path in $\mathscr{C}_{\text{S}}$ has a distance greater than D_{S} from $\mathscr{C}_{\text{obs}}$ and any path that does not belong to $\mathscr{C}_{\text{S}}$ has a distance closer than D_{S} to $\mathscr{C}_{\text{obs}}$.



(a) (b)

Figure 3.4: Desired subsets of $\mathscr{C}_{\text{free}}$: (a) $\mathscr{C}_{\text{S}}$; (b) $\mathscr{C}_{\text{D}}$.

Second, given a distance $D_{\text{D}} < D_{\text{EGVD}}$, the set $\mathscr{C}_{\text{D}}$ consists of all points, whose distance from $\mathscr{V}_{\text{EGVD}}$ is smaller than D_{D}:

$$\mathscr{C}_{\text{D}} = \{p \in \mathscr{C}_{\text{free}} | d(p, \mathscr{V}_{\text{EGVD}}) \leq D_{\text{D}}\}. \tag{3.4}$$

---

[1] We note that there is no solution of the path planning problem if there is no solution path in the EGVD.

The set $\mathscr{C}_{\mathrm{D}}$ is illustrated in Fig. 3.4 (b). Similar to $\mathscr{C}_{\mathrm{S}}$, the set $\mathscr{C}_{\mathrm{D}}$ fully contains the Voronoi boundary $\mathscr{V}_{\mathrm{EGVD}}$ of the EGVD since $D_{\mathrm{D}} < D_{\mathrm{EGVD}}$. That is, assuming that a solution path exists in $\mathscr{V}_{\mathrm{EGVD}}$, it is ensured that a solution path exists in $\mathscr{C}_{\mathrm{D}}$. Moreover, according to (3.4), paths are directed towards the goal along $\mathscr{V}_{\mathrm{EGVD}}$ since $\mathscr{C}_{\mathrm{D}}$ does not deviate from $\mathscr{V}_{\mathrm{EGVD}}$ by more than D_{D} and the minimum distance to $\mathscr{C}_{\mathrm{obs}}$ is larger than $D_{\mathrm{EGVD}} - D_{\mathrm{D}}$.

The methodology proposed in this thesis is based on the sets $\mathscr{C}_{\mathrm{S}}$ and $\mathscr{C}_{\mathrm{D}}$. In particular, we point out that any of the sampling-based methods can be applied to the environments with the modified free space $\mathscr{C}_{\mathrm{S}}$ or $\mathscr{C}_{\mathrm{D}}$.

Consequently, we introduce the methods S-PRM, S-PRM*, S-RRT, S-RRT*, S-FMT, S-GBRRT, which apply the respective sampling-based method on the pre-processed map with $\mathscr{C}_{\mathrm{free}} = \mathscr{C}_{\mathrm{S}}$. For all of the proposed methods, path safety is directly taken into account since a lower bound on the distance from $\mathscr{C}_{\mathrm{obs}}$ is ensured by the pre-processed environment according to (5.4). Formally, the stated methods address the following safe path planning problem.

$$\arg \min_{P, \mathscr{P}_P \in \mathscr{C}_{\mathrm{S}}} L_P. \tag{3.5}$$

That is, the proposed methods attempt to find the shortest possible path with the minimum given distance D_{S} to the obstacle region. As an interesting feature, each of the above algorithms has the same properties as the corresponding classical algorithm regarding probabilistic completeness and asymptotic optimality for the problem in (3.5).

We note that the idea of using an inflated obstacle region was previously used in [34] in combination with the application of goal-biased RRT. Different from our methodology, the work in [34] does not quantify how much obstacles should be inflated, does not formalize the properties of the computed paths and only applies GBRRT on the modified map with inflated obstacles.

As a possible shortcoming of using $\mathscr{C}_{\mathrm{S}}$, it has to be considered that $\mathscr{C}_{\mathrm{S}}$ contains the entire part of the free space with a safety distance of D_{S} to the obstacles. Hence, it is expected to contain regions that need not be explored during path planning as can be observed in Fig. 3.4 (a). Accordingly, we further define the methods D-PRM, D-PRM*, D-RRT, D-RRT*, D-FMT, D-GBRRT for the pre-processed map with $\mathscr{C}_{\mathrm{free}} = \mathscr{C}_{\mathrm{D}}$. These methods address the directed path planning problem

$$\arg \min_{P, \mathscr{P}_P \in \mathscr{C}_{\mathrm{D}}} L_P. \tag{3.6}$$

Since $\mathscr{C}_{\mathrm{D}}$ always stays close to the EGVD according to (3.4), it does not contain unnecessary regions of the free space. Furthermore, path safety is ensured by the lower bound $D_{\mathrm{EGVD}} - D_{\mathrm{D}}$ on the distance to $\mathscr{C}_{\mathrm{obs}}$. Again, each of the above algorithms has the same properties as the corresponding classical algorithm regarding probabilistic completeness and asymptotic optimality for the problem in (3.6). As a possible shortcoming of this method, regions that could lead to a shorter path might be missed depending on D_{D}.

3.2.3 Evaluation

We next evaluate the proposed methodology regarding the resulting path length, safety distance and computation time as well as the comparison to the existing methods in Section 3.1. To this end, we apply the proposed methodology to the environments in Section 3.1.4. The resulting pre-processed maps for the different environments with their parameters D_{EGVD}, D_{S}, D_{D} are shown in Fig. 3.5 to 3.8. Following the procedure in Section 3.2.1, we determine the GVD from a digital image in the jpeg format using morphological operations and the pruning algorithm in [52]. Similarly, the sets $\mathscr{C}_{\mathrm{S}}$ and $\mathscr{C}_{\mathrm{D}}$ are obtained by inflating $\mathscr{C}_{\mathrm{obs}}$ and $\mathscr{V}_{\mathrm{EGVD}}$, respectively, using the morphological operation dilation.

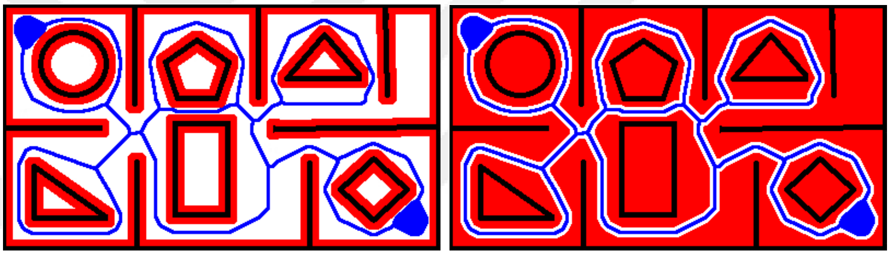


Figure 3.5: Pre-processed polygon map with $D_{\mathrm{EGVD}} = 8$, $D_{\mathrm{S}} = 6$, $D_{\mathrm{D}} = 4$: (a) using $\mathscr{C}_{\mathrm{S}}$; (b) using $\mathscr{C}_{\mathrm{D}}$.

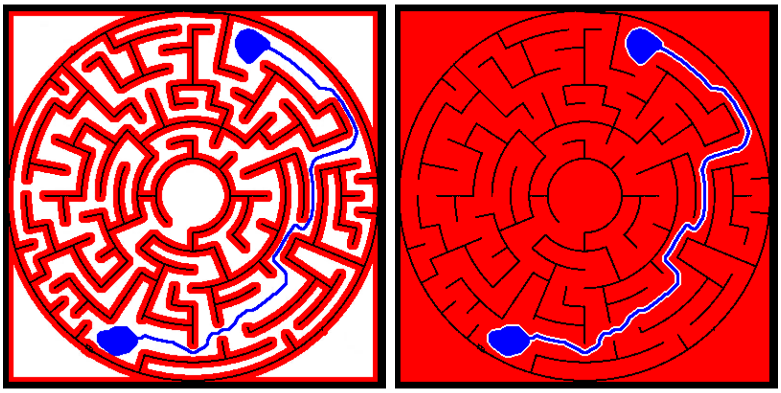


Figure 3.6: Pre-processed regular maze map with $D_{\mathrm{EGVD}} = 7$, $D_{\mathrm{S}} = 4$, $D_{\mathrm{D}} = 4$: (a) using $\mathscr{C}_{\mathrm{S}}$; (b) using $\mathscr{C}_{\mathrm{D}}$.

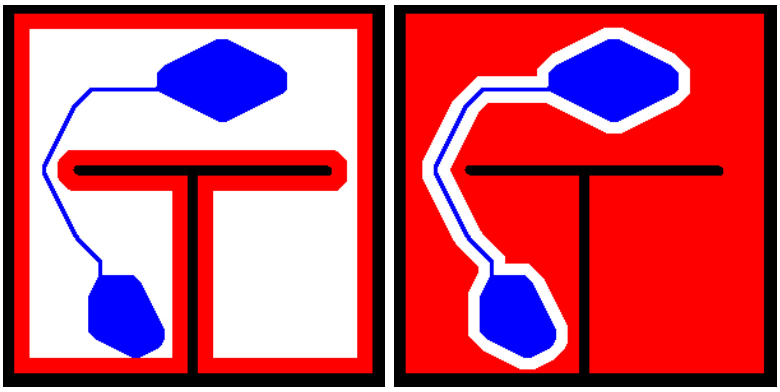


Figure 3.7: Pre-processed T map with $D_{\mathrm{EGVD}} = 14$, $D_{\mathrm{S}} = 9$, $D_{\mathrm{D}} = 7$: (a) using $\mathscr{C}_{\mathrm{S}}$; (b) using $\mathscr{C}_{\mathrm{D}}$.

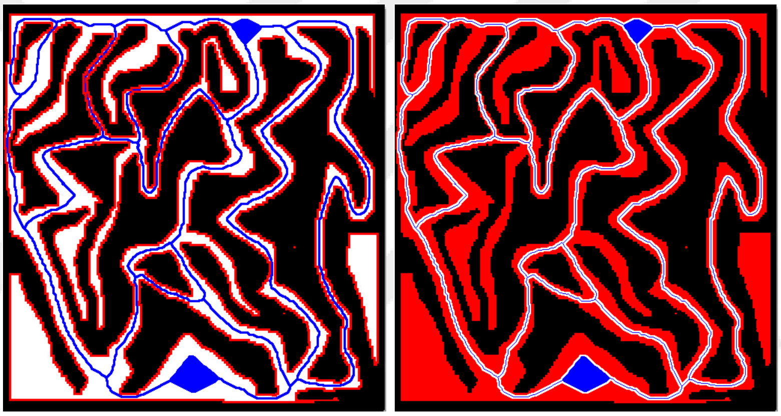


Figure 3.8: Pre-processed irregular maze map with $D_{\mathrm{EGVD}} = 3$, $D_{\mathrm{S}} = 2$, $D_{\mathrm{D}} = 2$: (a) using $\mathscr{C}_{\mathrm{S}}$; (b) using $\mathscr{C}_{\mathrm{D}}$.

The next sections perform a comparison of the PRM-based methods (Section 3.2.3) and RRT-based methods (Section 3.2.3) on the pre-processed environment maps. Based on this comparison, the most suitable methods for finding short and safe solution paths with a short computation time are determined in Section 3.2.4.

PRM-based Methods

We first investigate the performance of the PRM-based methods in comparison to the safety-oriented methods in Section 3.1.2 (Voronoi-Dijkstra – VD) and Section 4.1.5 (CRT). The results regarding computation time, path length and path safety for the different environments are shown in Fig. 3.9.

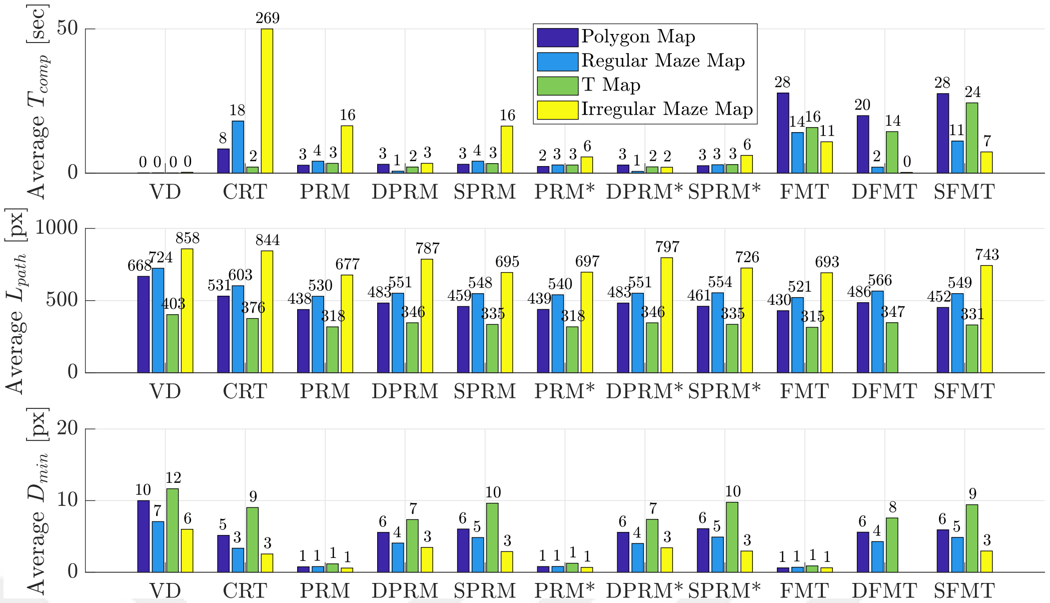


Figure 3.9: Comparison of the PRM methods.

Fig. 3.9 shows that VD generally provides the fastest and safest solution but with a much longer solution path. It can further be seen that the proposed methods (D-PRM, S-PRM, D-PRM*, S-PRM*, D-FMT, S-FMT) significantly increase path safety compared to the classical algorithms PRM, PRM* and FMT. Hereby, S-PRM, S-PRM* and S-FMT achieve shorter paths with a similar or better safety distance compared to D-PRM, D-PRM* and D-FMT. However, the computation time of S-PRM, S-PRM* and S-FMT is higher due to the larger free space to be explored. It is important to note that the proposed methods are superior to CRT regarding path length, path safety and computation time. In particular, CRT does not perform well for maze environments and environments with multiple candidate routes, whereas the algorithms based on $\mathscr{C}_\mathrm{D}$ can find safe paths within a short time. Comparing among the proposed algorithms, the algorithms D-PRM* and S-PRM* are preferable to achieve a short computation time.

A further important parameter is the success rate R_suc of each algorithm, which quantifies the percentage of all trials where a solution path could be found. The values for the PRM-based algorithms are summarized in Table 3.2. Here, the interesting observation is that the success rate of safe algorithms such as S-PRM, S-PRM* and CRT is clearly below 100% for the regular maze map. The reason is that the original environment in Fig. 3.1 as well as the pre-processed map in Fig. 3.6 (a) has tight passages and large irrelevant portions that need to be explored. For such type of environments, the methods D-PRM and D-PRM* that are directed along $\mathscr{V}_\mathrm{EGVD}$ provide a much higher success rate.

Table 3.2: Success rate for the PRM-based methods.

	VD	PRM	DPRM	SPRM	PRM*	DPRM*	SPRM*	CRT	FMT	DFMT	SFMT
	Polygon Map										
R_{suc}	100	100	100	100	100	100	100	100	92	97	94
	Regular Maze Map										
R_{suc}	100	100	100	92	79	100	65	82	95	56	86
	T Map										
R_{suc}	100	100	100	100	100	100	100	99	97	96	98
	Irregular Maze Map										
R_{suc}	100	100	100	100	99	99	98	98	98	0	93

RRT-based Methods

We next perform a comparison of the RRT-based methods. The computational results regarding path length, path safety and computation time are shown in Fig. 3.10.

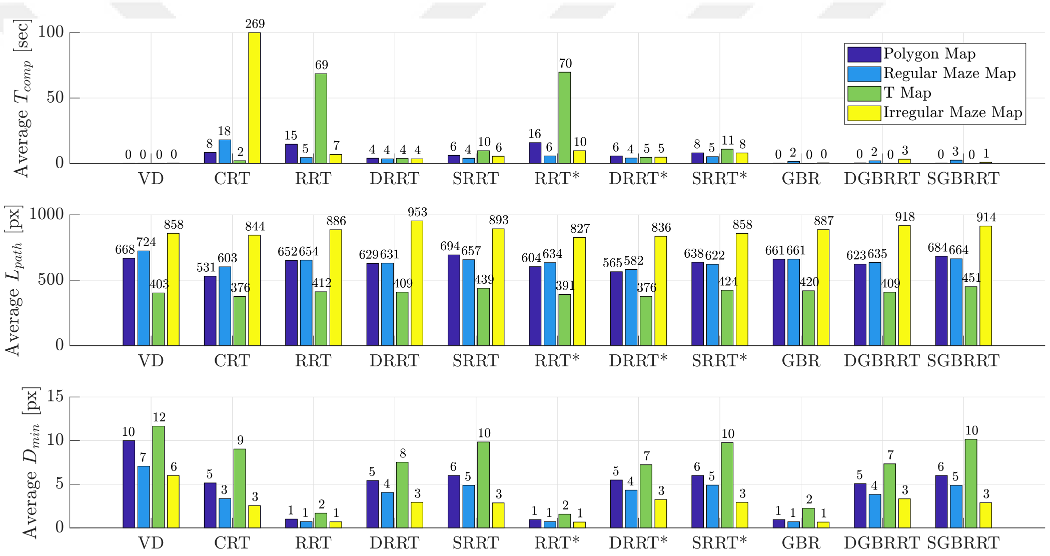


Figure 3.10: Comparison of the RRT-based methods.

First, it can be observed that, although the algorithms based on GBRRT provide fast results, they generally lead to even longer paths than the VD algorithm. Regarding the other proposed methods, it is evident that D-RRT, S-RRT, D-RRT* and S-RRT* considerably increase path safety and decrease the computation time compared to RRT and RRT* at a slightly increased path length. Furthermore, especially D-RRT and D-RRT* run faster than CRT in most of the cases, whereby it has to be noted that these algorithms show a low success rate in maze maps as can be seen in Table 3.3. S-RRT and S-RRT* require a slightly larger computation time but offer a higher success rate. Nevertheless, it has to be recognized that CRT provides similar results to the RRT-based algorithms regarding path length and safety. That is, different from the PRM-based algorithms, the proposed RRT-based algorithms do not offer considerable improvements in the studied 2D environments.

Table 3.3: Success rate and number of nodes for the RRT-based methods.

	VD	RRT	DRRT	SRRT	RRT*	DRRT*	SRRT*	GBRRT	DGBRRT	SGBRRT
	Polygon Map									
R_{suc}	100	100	100	100	100	100	100	100	100	100
	Regular Maze Map									
R_{suc}	100	96	68	86	100	61	84	97	64	81
	T Map									
R_{suc}	100	100	100	100	100	100	100	100	100	100
	Irregular Maze Map									
R_{suc}	100	100	28	100	100	18	100	100	15	100

3.2.4 Discussion

We next discuss the main outcomes of our evaluation. For convenience, Fig. 3.11 displays the results of the methods that were found most suitable in Section 3.2.3. Recalling that the focus was on the average values of 100 computational experiments, we also provide the standard deviation (STD) of these experiments in Fig. 3.12.

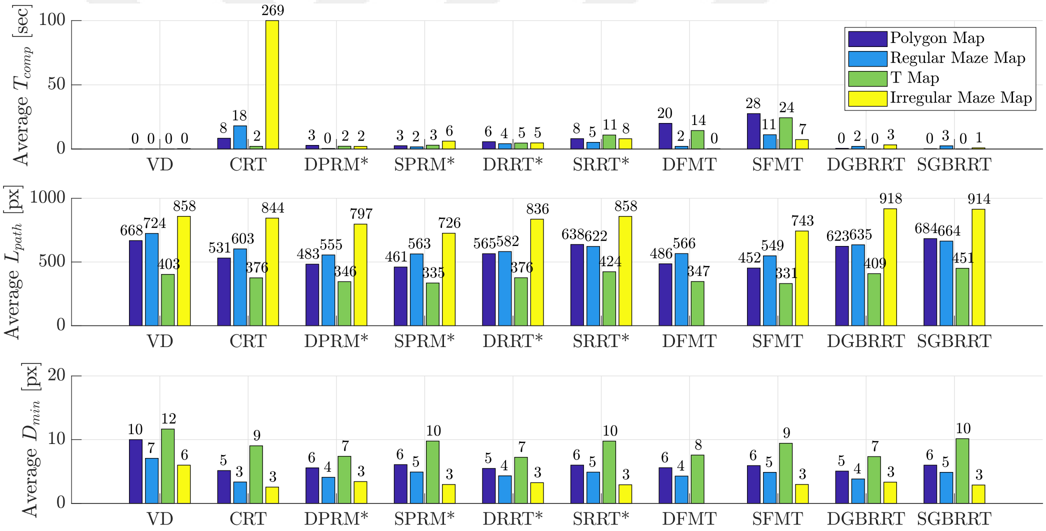


Figure 3.11: Comparison of the most suitable methods.

Inspecting Fig. 3.11, it is evident that the PRM-based algorithms provide the best results when taking into account the average values of path length, path safety and computation time. CRT generally leads to longer paths, a smaller safety distance, a larger computation time and a smaller success rate compared to D-PRM* and S-PRM*. In addition, different from these algorithms, CRT does not provide any guarantee regarding probabilistic completeness and asymptotic optimality. The RRT-based algorithms show an increased path length and a possibly reduced success rate compared to the PRM-based algorithms. Although D-FMT and S-FMT can find short paths, the computation time and success rate are worse than for the PRM-based algorithms.

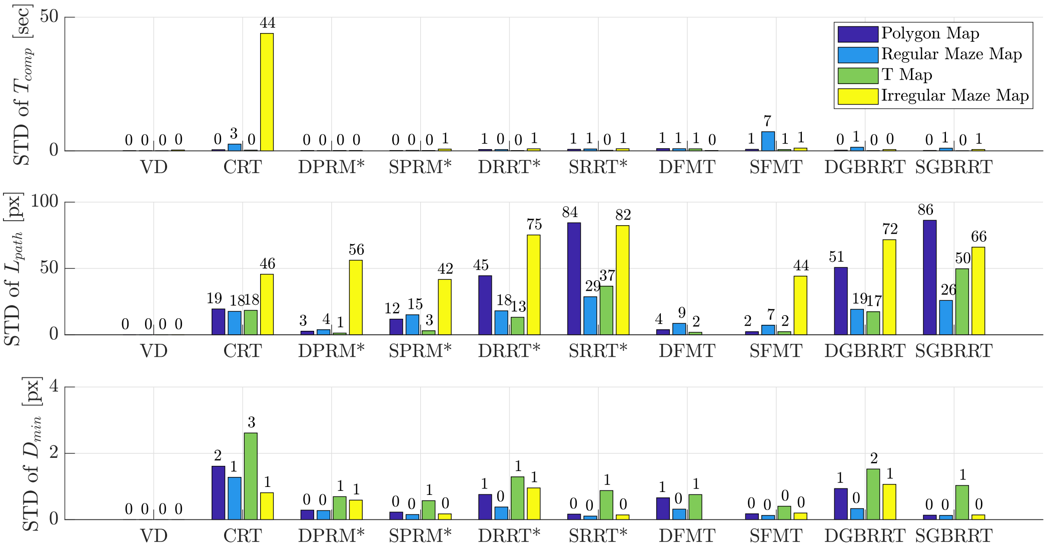


Figure 3.12: Comparison of the standard deviation of the most suitable methods.

Looking at Fig. 3.12, it can be seen that D-PRM* is superior to the other methods regarding the STD of the different performance metrics. In particular, confining solution paths to the vicinity of the EGVD ensures that there are small deviations in the lengths of different solution paths for most of the environments. The only exception is the irregular maze map, which offers a multitude of routes between the start and goal position as was discussed in Section 3.1.4. Accordingly, similar to the other algorithms, the D-PRM* algorithm finds solution paths along different routes, leading to larger differences in the path lengths.

Overall, we conclude that the proposed methodology outperforms existing algorithms that both minimize path length and take into account path safety such as CRT [20] and SGBRRT [34]. Especially the PRM-based algorithms D-PRM* and S-PRM* compute solution paths with a short path length, guaranteed safety and a small computation time. Hereby, D-PRM* provides better results regarding the success rate, the computation time and the STD of the performance parameters such that different solution paths show only small differences. On the other hand, S-PRM* is able to find shorter paths due to the availability of more free space (see also Fig. 3.5 to 3.8). As an important property, the proposed methods provide formal guarantees regarding path safety. While the minimum distance D_{min} of the CRT algorithm is a result of applying the algorithm, the minimum distance obtained from D-PRM* and S-PRM* is determined by the given parameters D_{EGVD}, D_{S} and D_{D} as is confirmed by the observed STD in Fig. 3.12.

CHAPTER 4

MODIFYING THE SAMPLING STRATEGY FOR FAST CLASSICAL ALGORITHMS

The methodology proposed in the previous chapter suggest to use pre-processed environment maps in order to restrict the free space available for robot paths. However, the original sampling-based methods are applied on these modified environment maps. In particular, samples are generated everywhere on the map and rejected if they happen to lie in the obstacle region. Since the sample generation and rejection adds to the computation time, this chapter focuses on reducing the number of samples to be generated. That is, instead of applying the original sampling-based methods, we next propose efficient modifications of the classical sampling-based algorithms. To this end, Section 4.1 highlights the relevant functions of classical sampling-based methods. Section 4.2 explains the proposed method for reducing the number of required samples and Section 4.3 performs a comprehensive comparison based on different environment maps.

4.1 Background on Classical Methods

This section summarizes the required background information for the prposed method.

4.1.1 Relevant Functions for Sampling-based Methods

Sampling-based algorithms for robotic path planning are of high interest due to their fast computation of reasonable solution paths. We next outline common functions of these sampling-based algorithms.

All sampling-based methods are based on the generation of random samples in $\mathscr{C}_{\text{free}}$. We denote the function that generates N samples in $\mathscr{C}_{\text{free}}$ that are drawn from a uniform distribution as

$$X_{\text{rand}} = \texttt{SampleFree}(\mathscr{C}_{\text{free}}, N) \subseteq \mathscr{C}_{\text{free}}.$$

Hereby, each $x \in X_{\text{rand}}$ represents a random sample in $\mathscr{C}_{\text{free}}$. In addition, we introduce the function

$$X_{\text{new}} = \texttt{SampleRad}(x, r, N, \mathscr{C}_{\text{free}}) \subseteq \mathscr{C}_{\text{free}}$$

that generates N random samples X_{new} in $\mathscr{C}_{\text{free}}$ on a circle with radius r around a point x. Further,

$$X_{\text{near}} = \texttt{Near}(V, x, r) = \{\hat{x} \in V | d(x, \hat{x}) \leq r\}$$

determines all points in the set V that lie on a disk with radius r around a point x. Finally,

$$\texttt{CollisionFree}(x, \hat{x}, \mathscr{C}_{\text{obs}}) = \begin{cases} \textbf{true} & \text{if } l_{x,\hat{x}} \cap \mathscr{C}_{\text{obs}} \neq \emptyset \\ \textbf{false} & \text{otherwise} \end{cases}$$

returns **true** if the straight line $l_{x,\hat{x}}$ between the points x and $\hat{x}$ intersects $\mathscr{C}_{\text{obs}}$ and **false** otherwise.

4.1.2 Probabilistiy Roadmap (PRM)

The PRM algorithm in Algorithm 1 is one of the most popular algorithms for robotic path planning [16, 41] that has initially been introduced for multi-query applications. However, the PRM algorithm is as well suitable for single-query applications [82]. In this case, the PRM algorithm first generates a set of N_{PRM} random nodes in $\mathscr{C}_{\text{free}}$ and creates a graph $G = (V, E)$. Initially, the vertexes consist of X_{rand}, p_{S} and p_{G} and there are no edges. Then, the PRM algorithm iteratively picks one of the sample nodes (line 4) and determines all neighbor nodes X_{near} of x_{rand} within a radius r_{PRM} (line 4). Edges from nodes $x_{\text{near}} \in X_{\text{near}}$ to x_{rand} are introduced if x_{rand} and x_{near} do not belong to the same connected component in G and if they can be connected by a collision free line (line 5 to 8). Each edge $(x_{\text{rand}}, x), (x, x_{\text{rand}})$ is labeled by its cost $c((x_{\text{rand}}, x)) = c((x, x_{\text{rand}}))$, which is represented by the distance between the related nodes (line 9). Finally, the algorithm returns the shortest (minimum cost) path P from p_{s} to p_{g} in the resulting graph G.

Algorithm 1 PRM algorithm (for fixed value of r_{PRM}) and PRM* algorithm (for $r_{\text{PRM}} = \gamma_{\text{PRM}} \cdot (\log(n)/n)^2)$).

```
 1: Input: $P = \mathit{PRM}(p_S, p_G, N_{\text{PRM}}, \mathscr{C}, \mathscr{C}_{\text{obs}}, \mathscr{C}_{\text{free}})$
 2: Initialize: $X_{\text{rand}} = \texttt{SampleFree}(\mathscr{C}_{\text{free}}, N_{\text{PRM}})$; $V = \{p_S, p_G\} \cup X_{\text{rand}}$; $E = \emptyset$
 3: for $i = 1, \ldots, N_{\text{PRM}}$ do
 4:     Pick $x_{\text{rand}} \in X_{\text{rand}}$; $X_{\text{rand}} = X_{\text{rand}} \setminus \{x_{\text{rand}}\}$  $X_{\text{near}} = \texttt{Near}(V, x_{\text{rand}}, r_{\text{PRM}})$
 5:     for $x \in X_{\text{near}}$ in order of increasing $d(x, x_{\text{rand}})$ do
 6:         if $x_{\text{rand}}$ and $x$ are not in the same connected component of $G$ then
 7:             if $\mathtt{CollisionFree}(x_{\text{rand}}, x, \mathscr{C}_{\text{obs}})$ then
 8:                 $E = E \cup \{(x_{\text{rand}}, x), (x, x_{\text{rand}})\}$
 9:                 $c((x_{\text{rand}}, x)) = c((x, x_{\text{rand}})) = d(x_{\text{rand}}, x), (x, x_{\text{rand}})$
10: return shortest path $P$ from $p_S$ to $p_G$ in $G$.
```

4.1.3 PRM*

The PRM* algorithm was proposed as a modified version of PRM in [17]. Its only difference to the classical PRM algorithm in Algorithm 1 is that the connection radius r_{PRM} between nodes in line 4 decreases with an increasing number of nodes in the form $r_{\text{PRM}} = \gamma_{\text{PRM}} \cdot \sqrt{\log(n)/n}$. Hereby, γ_{PRM} is a constant that has to be chosen larger than $2 \cdot \sqrt{1.5} \cdot \mu(\mathscr{C}_{\text{free}})/\pi$ for 2D-environments ($\mu(\mathscr{C}_{\text{free}})$ denotes the area of $\mathscr{C}_{\text{free}}$). As a result, the PRM* algorithm is expected to produce shorter solution paths than the PRM algorithm with a possibly increased computational effort.

4.1.4 Fast Marching Tree (FMT)

The FMT algorithm was proposed in [19]. Its set of nodes V is generated in the same way as for the PRM algorithm. Moreover, it keeps sets of closed (V_{closed}), open (V_{open}) and unvisited (V_{un}) nodes that are initialized in line 1. Hereby, $c(x)$ represents the cost (distance) of traveling from p_{S} to x along the graph $G = (V, E)$. In each iteration, the FMT algorithm selects the node with the lowest cost in the open set (line 15) and finds all of its neighbors X_{near} in the unvisited set (line 3). In analogy to PRM*, neighbors are defined based on a connection distance $r_{\text{FMT}} = \gamma_{\text{FMT}} \cdot \sqrt{\log(n)/n}$ that decreases with the number of iterations. Each neighbor $x \in X_{\text{near}}$ is then connected to the closest neighbor y_{min} in the open set Y_{near} if the connection is collision free and the relevant sets are updated (line 5 to 13). The algorithm terminates without success if no more nodes are left to be processed ($V_{\text{open}} = \emptyset$ in line 14). Otherwise, a solution path is returned if p_{G} is found to be the unprocessed node with the lowest cost.

4.1.5 Confidence Random Tree (CRT)

The previously discussed algorithms are designed for finding short paths with a small computation time. Including path safety as a performance metric, the confidence random tree (CRT) algorithm tries to generate solution paths that stay away from obstacles [20]. To this end, the CRT algorithm introduces the notion of *confidence* of a node $x \in \mathscr{C}_{\text{free}}$ as

$$\texttt{Conf}(x, \mathscr{C}_{\text{obs}}) = \min\{d(x, \mathscr{C}_{\text{obs}})/c_{\max}, 1\}.$$

Hereby, $c_{\max}$ is a maximum distance parameter and the confidence gives an indication of the distance of x to the obstacle region. Then, the CRT algorithm expands a tree $G = (V, E)$ starting from p_{S} (line 2). In each iteration, a set X_{new} of new nodes is determined at a distance $d = \texttt{Conf}(x, \mathscr{C}_{\text{obs}}) \cdot c_{\max}$ from each node x in the current open set X_{open} (line 5 to 9). Hereby, each element of X_{new} stores the generated node x_{new} and its parent node x (line 9). In order to limit the number of nodes, the CRT algorithm includes a node rejection method to avoid generating nodes in previously explored

Algorithm 2 FMT algorithm.

```
Input: $P = \mathit{FMT}(p_S, p_G, N_{FMT}, \mathscr{C}, \mathscr{C}_{obs}, \mathscr{C}_{free})$ Initialize: $V = \{p_S, p_G\} \cup \texttt{SampleFree}(\mathscr{C}_{free}, N_{FMT})$; $E = \emptyset$; $V_{un} = V \setminus p_S$; $V_{open} = \{p_S\}$; $V_{closed} = \emptyset$; $c(p_S) = 0$; $z = p_S$
while $z \neq p_G$ do
    $\hat{V}_{open} = \emptyset \quad X_{near} = V_{un} \cap \texttt{Near}(V \setminus \{z\}, z, r_{FMT})$
    for $x \in X_{near}$ do
        $Y_{near} = V_{open} \cap \texttt{Near}(V \setminus \{x\}, x, r_{FMT})$
        $y_{min} = \arg\min_{y \in Y_{near}} \{c(y) + d(y, x)\}$
        if $\mathit{CollisionFree}(y_{min}, x, \mathscr{C}_{obs})$ then
            $E = E \cup \{(y_{min}, x)\}$
            $\hat{V}_{open} = \hat{V}_{open} \cup \{x\}$
            $V_{un} = V_{un} \setminus \{x\}$
            $c(x) = c(y_{min}) + d(y_{min}, x)$
    $V_{open} = (V_{open} \cup \hat{V}_{open}) \setminus \{z\}$
    $V_{closed} = V_{closed} \cup \{z\}$
    if $V_{open} = \emptyset$ then
        return no path found
    $z = \arg\min_{x \in V_{open}} \{c(x)\}$
return shortest path $P$ from $p_S$ to $p_G$ in $G$.
```

areas (line 10 to 22). Here, nodes are rejected if they are too close to previously explored nodes in X_{closed} (line 14 to 16) or if they are too close to an accepted node x_{new} with a higher confidence (line 20 to line 22). Accepted nodes x_{new} are added to X_{open} for processing and an edge to the parent node is introduced in G (line 18 and 19). The CRT algorithm terminates without a solution path if X_{open} is empty or with a solution path P if a connection to p_G is found (line 25).

Due to the consideration of node confidence, the CRT algorithm generates safe solution paths at the expense of an increased path length.

Algorithm 3 CRT algorithm.

```
**Input:** $P = \mathtt{CRT}(p_S, p_G, c_{\max}, c_{\min}, \mathscr{C}, \mathscr{C}_{\text{obs}}, \mathscr{C}_{\text{free}})$
**Initialize:** $X_{\text{open}} = \{p_S\}$; $X_{\text{closed}} = \emptyset$; $V = \{p_S\}$, $E = \emptyset$
**while** $X_{\text{open}} \neq \emptyset$ **do**
    $X_{\text{closed}} = X_{\text{closed}} \cup X_{\text{open}}$; $X_{\text{new}} = \emptyset$
    **for** $x \in X_{\text{open}}$ **do**
        $X_{\text{rand}} = \mathtt{SampleRad}(x, \mathtt{Conf}(x, \mathscr{C}_{\text{obs}}) \cdot c_{\max}, n_{\text{CRT}}, \mathscr{C}_{\text{free}})$
        **for** $x_{\text{new}} \in X_{\text{rand}}$ **do**
            **if** $\mathtt{Conf}(x_{\text{new}}, \mathscr{C}_{\text{obs}}) \geq c_{\min}$ **then**
                $X_{\text{new}} = X_{\text{new}} \cup \{(x_{\text{new}}, x)\}$
    Sort $X_{\text{new}}$ with decreasing confidence
    $X_{\text{open}} = \emptyset$
    **while** $X_{\text{new}} \neq \emptyset$ **do**
        Take first element $(x_{\text{new}}, x)$ from $X_{\text{new}}$; $X_{\text{new}} = X_{\text{new}} \setminus \{(x_{\text{new}}, x)\}$; $f_{\text{accept}} = 1$
        **for** $\hat{x} \in X_{\text{closed}}$ **do**
            **if** $\mathtt{Conf}(\hat{x}) \cdot c_{\max} > d(x_{\text{new}}, \hat{x})$ **then**
                $f_{\text{accept}} = 0$ **break**
        **if** $f_{\text{accept}} \neq 0$ **then**
            $X_{\text{open}} = X_{\text{open}} \cup \{x_{\text{new}}\}$
            $V = V \cup \{x_{\text{new}}\}$; $E = E \cup \{(x, x_{\text{new}})\}$
            **for** $(\hat{x}_{\text{new}}, \hat{x}) \in X_{\text{new}}$ **do**
                **if** $\mathtt{Conf}(x_{\text{new}}) \cdot c_{\max} > d(x_{\text{new}}, \hat{x}_{\text{new}})$ **then**
                    $X_{\text{new}} = X_{\text{new}} \setminus \{(\hat{x}_{\text{new}}, \hat{x})\}$
    $x_{\text{near}} = \arg\min_{x \in X_{\text{open}}} d(x, p_G)$
    **if** $\text{Conf}(x_{\text{near}}) \cdot c_{\max} > d(x_{\text{near}}, p_G)$ **then**
        **return** path $P$ from $p_S$ to $p_G$ in $G = (V, E)$
**return** $P = \emptyset$
```

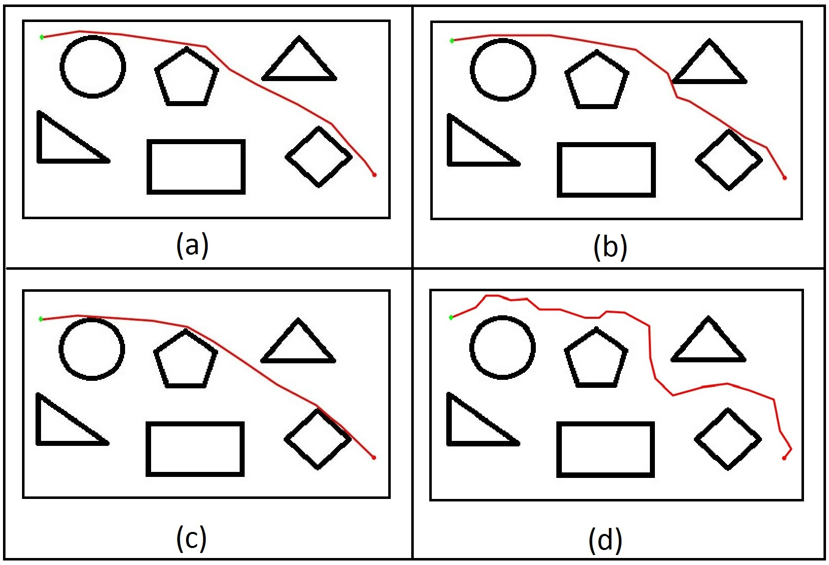


Figure 4.1: Example solution paths: (a) PRM; (b) PRM*; (c) FMT; (d) CRT.

For illustration, Fig. 4.1 shows solution paths for the described algorithms using an example map. It is readily observed that the paths obtained for PRM, PRM* and FMT come very close to the obstacles. This is expected since these algorithms try to minimize the path length and do not account for path safety. On the other hand, the example path for CRT in Fig. 4.1 (d) stays away from the obstacles but leads to an increased path length. The main focus of this thesis is the adaptation of algorithms such as PRM, PRM* and FMT in order to address path safety without a significant increase in path length.

4.2 Proposed Method

This section develops the method for safe and fast path planning based on the knowledge of the extended GVD and the corresponding Voronoi boundary $\mathscr{V}$ in Section 3.1.2. Section 5.1.5 describes the general methodology and Section 4.2.2 and 4.2.3 combine the proposed methodology with the classical path planning algorithms described in Section 4.1.

4.2.1 Inflated Path

We assume that the Voronoi boundary is available in the form of a set of points $\mathscr{V} \subseteq \mathscr{C}_{\text{free}}$ and the start and goal point p_{S} and p_{G} are elements of $\mathscr{V}$ in analogy to Section 3.1.2. Specifically, we consider the case where a solution path exists in the extended GVD.[1] As the first step of our algorithm, we suggest to prune the extended GVD in order to remove parts of $\mathscr{V}$ that cannot lie on a solution path from p_{S} to p_{G} similar to [52].

Next, we define a distance D_{I} and we inflate $\mathscr{V}$ by the width D_{I}. Considering that the environment is given in the form of a digital image (such as JPEG), whereby $\mathscr{V}$ is represented by pixels in this image, the inflated Voronoi boundary $\mathscr{V}_{\text{I}}$ can be computed by a morphological dilation operation:

$$\mathscr{V}_{\text{I}} = \mathscr{V} \oplus B_{D_{\text{I}}} = \bigcup_{b \in B_{D_{\text{I}}}} \mathscr{V}_b, \tag{4.1}$$

whereby $B_{D_{\text{I}}}$ represents a disk with radius D_{I}, $\oplus$ represents the dilation operation and $\mathscr{V}_b$ is the translation of $\mathscr{V}$ by $b \in B_{D_{\text{I}}}$. The resulting map for the environment in Fig. 4.1 with the inflated pruned Voronoi boundary $\mathscr{V}_{\text{I}}$ for $D_{\text{I}} = 6$ and $D_{\text{I}} = 12$ is shown in Fig. 4.2.

---

[1] We note that this is not a restriction of the general case. If there is no solution in the extended GVD, there is generally no solution of the path planning problem.

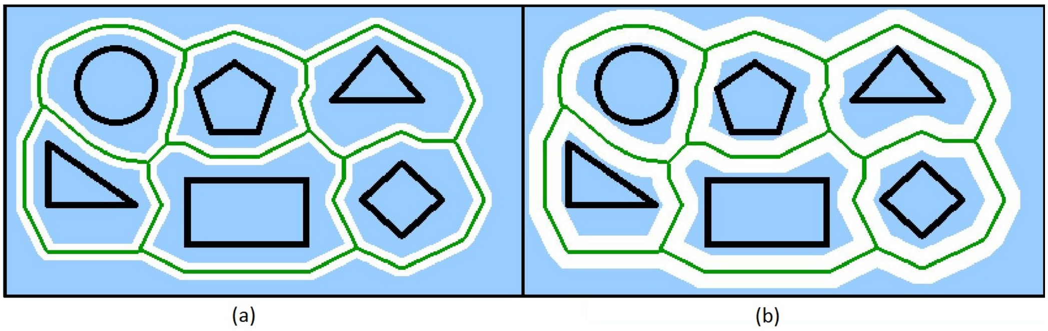

(a) (b)

Figure 4.2: Example environment with different path width: (a) D = (6 pixels) (b) D = (12 pixels)

Using $\mathscr{V}_{\mathrm{I}}$, the main idea of this thesis is a modification of the sampling method `SampleFree` used in the sampling-based path planning algorithms in Section 4.1. Instead of generating random samples in the free space $\mathscr{C}_{\mathrm{free}}$, we suggest to generate samples only in $\mathscr{V}_{\mathrm{I}}$. To this end, we next both develop an efficient method for generating such samples and provide a formula for deciding on the number of required node samples.

In order to efficiently generate samples in $\mathscr{V}_{\mathrm{I}}$, we first observe that all such samples should have a maximum distance of D_{I} from the original Voronoi boundary $\mathscr{V}$. That is, we first select a number of N_{V} random points $P_{\mathscr{V}}$ from $\mathscr{V}$. For each point $p \in P_{\mathscr{V}}$, we generate random values $d \in [0, D_{\mathrm{I}}]$ and $\theta \in [0, 2, \pi]$ and determine the sample

$$v = p + d \cdot \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}. \tag{4.2}$$

That is, each node sample is represented by a point, whose distance to $\mathscr{V}$ is bounded by D_{I}. The proposed procedure is summarized in Algorithm 4.
Writing $|\mathscr{V}|$ for the overall sum of all path lengths on $\mathscr{V}$, we observe that the number of required node samples increases with $|\mathscr{V}|$ and decreases with D_{I} since a larger value of D_{I} leaves more free space for obstacle-free connections (compare Fig. 4.2 (a) and (b)). Hence, we suggest to compute the number of node samples as

$$N_{\mathscr{V}} = \gamma_{\mathscr{V}} \cdot \frac{|\mathscr{V}|}{D_{\mathrm{I}}}, \tag{4.3}$$

whereby $\gamma_{\mathscr{V}}$ is a safety coefficient that can be adjusted depending on the specific environment. We next point out the proposed modifications of the algorithms (PRM, PRM', FMT) in Section 4.1.

Algorithm 4 Computation of samples close to $\mathscr{V}$.

```
$S = \mathtt{SampleInflated}(N_\mathrm{V}, \mathscr{V}, D_\mathrm{I})$
Initialize: $S = \emptyset$
Select $N_\mathrm{V}$ random points $P_\mathrm{V}$ from $\mathscr{V}$
for $p \in P_\mathrm{V}$ do
    Generate random $d \in [0, D_\mathrm{I}]$
    Generate random $\theta \in [0, 2, \pi]$
    Generate new sample $v$ according to (4.2)
    $S = S \cup \{v\}$
return $S$
```

4.2.2 Inflated-path PRM (IPRM) and Inflated-path PRM* (IPRM*)

The original PRM algorithm (Section 4.1.2, Algorithm 1) computes node samples in the overall free space using $\mathtt{SampleFree}(\mathscr{C}, \mathscr{C}_\mathrm{obs}, N_\mathrm{PRM})$ and checks if connections between nodes are collision-free using $\mathtt{CollisionFree}(x_\mathrm{rand}, x, \mathscr{C}_\mathrm{obs})$ with the obstacle region $\mathscr{C}_\mathrm{obs}$. The proposed algorithms inflated-path PRM (IPRM) and inflated-path PRM* (IPRM*) generate node samples as described in Section 5.1.5 and check collision-freeness using the modified obstacle region $\hat{\mathscr{C}}_\mathrm{obs} = \mathscr{C} \setminus \mathscr{V}_\mathrm{I}$. That is, line 2 in Algorithm 1 is replaced by

$$X_\mathrm{rand} = \mathtt{SampleInflated}(N_\mathscr{V}, \mathscr{V}, D_\mathrm{I}); V = \{p_\mathrm{S}, p_\mathrm{G}\}; E = \emptyset$$

and line 7 in Algorithm 1 is replaced by

$$\textbf{if } \mathtt{CollisionFree}(x_\mathrm{rand}, x, \hat{\mathscr{C}}_\mathrm{obs}) \textbf{ then}.$$

4.2.3 Improved FMT

Similar to the modification of PRM and PRM*, we suggest to change the sampling method and the obstacle region of the FMT algorithm in Section 4.1.4. To this end, we replace line 1 in Algorithm 2 by

$$V = \{p_\mathrm{S}, p_\mathrm{G}\} \cup \mathtt{SampleInflated}(N_\mathscr{V}, \mathscr{V}, D_\mathrm{I}); E = \emptyset; V_\mathrm{un} = V \setminus p_\mathrm{S}; V_\mathrm{open} = \{p_\mathrm{S}\};$$

In addition, we replace line 7 in Algorithm 2 by

$$\textbf{if } \mathtt{CollisionFree}(y_\mathrm{min}, x, \hat{\mathscr{C}}_\mathrm{obs}) \textbf{ then}.$$

4.3 Evaluation

We next perform a comparison of the proposed methods and the existing methods in Section 4.1 regarding the resulting path length, safety distance and computation time. Section 4.3.1 explains the setup of the computational experiments and Section 4.3.2 to 4.3.5 evaluate the considered algorithms for different environments. A discussion of the obtained results is given in Section 4.3.6.

4.3.1 Experimental Setup and Maps

We apply the described algorithms to the environments in Figure 3.1 which are given as binary images, where pixels in $\mathscr{C}_{\text{free}}$ are white and pixels in $\mathscr{C}_{\text{obs}}$ are black. The start position and the goal position are shown by a green diamond and red circle, respectively.

The maps are selected according to their different properties as follows. The polygon map in Fig. 4.3 (a) has different obstacles that are represented by polygon shapes and that leave sufficient free space for multiple routes between p_{S} and p_{G}. In the maze map in Fig. 4.3 (b), obstacles are represented by straight lines and there are multiple routes between p_{S} and p_{G}. The U-map in Fig. 4.3 (c) offers U-shaped obstacles, where candidate paths can be trapped. The maze map in Fig. 4.3 (d) provides a single long and narrow circular solution route.

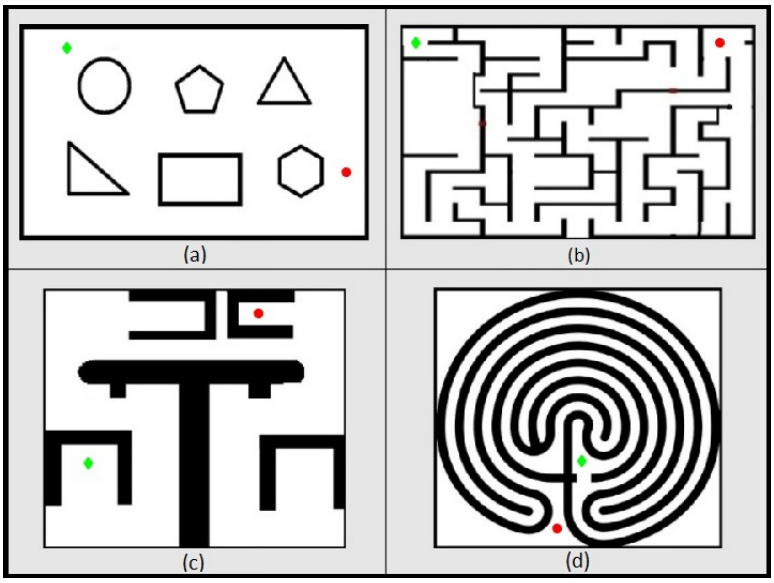


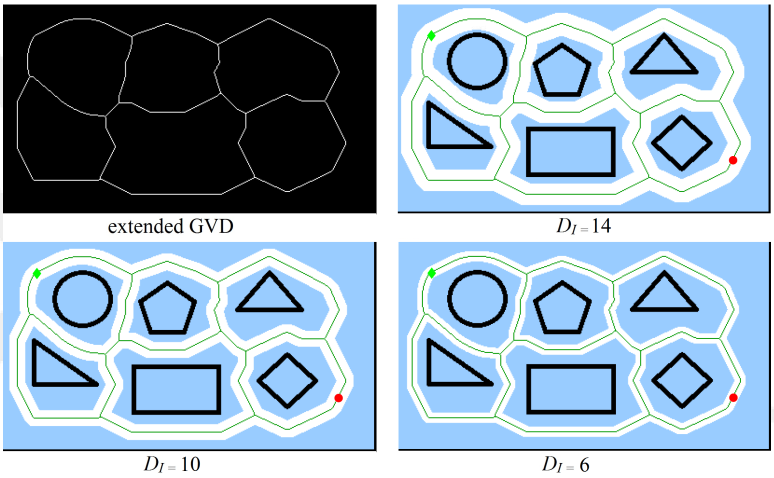
Figure 4.3: Environments used for the evaluation.

In order to perform a fair evaluation, all the algorithms were implemented in Matlab using the same functions for common tasks of the different algorithms as indicated in Section 4.1.1. The experiments were run on a personal computer with Intel(R)

Core(TM) i5-6500 CPU @ 3.20GHz and 8.00 GB RAM. For each environment, 100 test runs of each algorithm were performed.

4.3.2 Polygon Map

We first consider the polygon map in Fig. 4.3 (a). Following the procedure in Section 2.1.2 and 3.2.1, we determine the extended GVD and the inflated Voronoi boundary as shown in Fig. 4.4 for different values of $D_{\mathrm{I}} = 14$, $D_{\mathrm{I}} = 10$ and $D_{\mathrm{I}} = 6$. Here, $\mathscr{V}_{\mathrm{I}}$ is shown in white, $\mathscr{V}$ is shown in green and $\mathscr{C}_{\mathrm{obs}}$ is shown in black. The light blue region represents the part of $\mathscr{C}_{\mathrm{free}}$ that is not close enough to $\mathscr{V}$ and hence is not considered for solution paths.



extended GVD

$D_I = 14$

$D_I = 10$

$D_I = 6$

Figure 4.4: Polygon map: extended GVD and inflated Voronoi boundary.

The computational results for the polygon map are shown in Fig. 4.5. For each method the average values of computation time ($\bar{T}_{\mathrm{comp}}$), path length ($\bar{L}_{\mathrm{path}}$), minimum obstacle distance ($\bar{D}_{\mathrm{min}}$) and number of nodes ($\bar{N}_{\mathrm{nodes}}$) for 100 runs (represented by a bar) are displayed. In addition, the error bars show the maximum and minimum value among the 100 test runs. We point out the following main observations from the figure.

- Regarding the computation time, it is observed that the proposed algorithms are always faster than the related classical algorithm, whereby the computation time increases for a narrower inflated path. This change in the computation time is directly related to the number of nodes $\bar{N}_{\mathrm{nodes}}$ as computed with (4.3).

- Although it is the case that PRM, PRM* and FMT produce short solution paths, these algorithms do not account for path safety, that is, $\bar{D}_{\mathrm{min}}$ is small. All the proposed algorithms achieve increased safety depending on the value of D_{I}. Most interestingly, path safety is comparable to the results of the CRT algorithm if D_{I}

is chosen small enough, whereas the computation time, the path length and the variation among solutions are smaller for the proposed algorithms.

- Only the VD algorithm can achieve safer paths than the proposed algorithms but with a significantly increased path length.

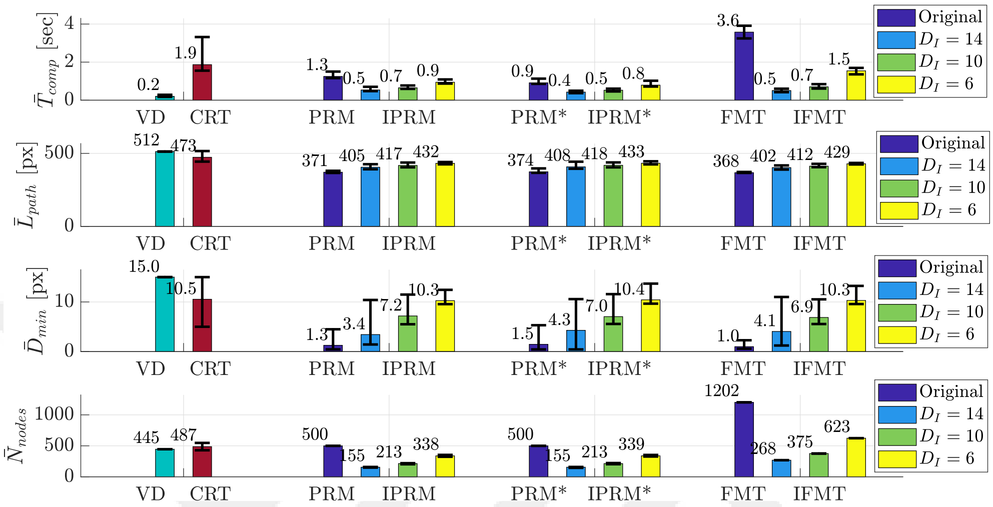
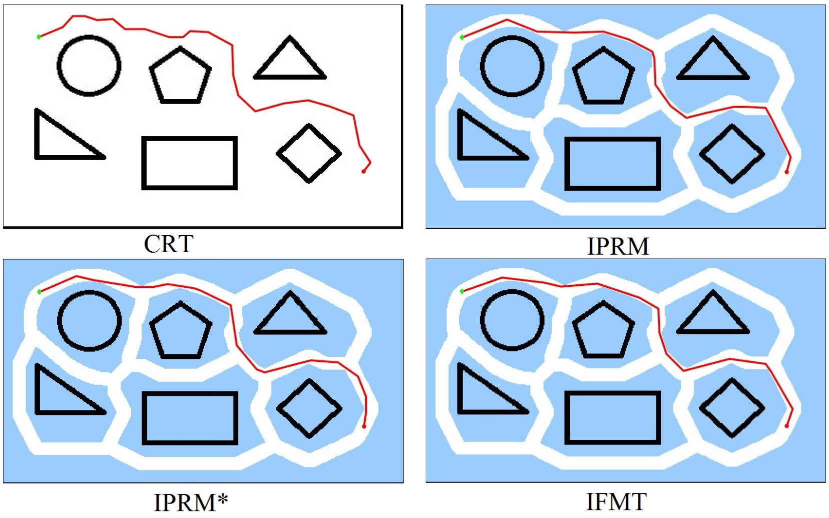


Figure 4.5: Comparison of the performance metrics for the polygon map.

In summary, the proposed algorithms clearly outperform the existing algorithms for the polygon map when taking into account computation time, path length and safety. For illustration, Fig. 4.6 compares solution path examples for the different methods ($D_\text{i} = 6$). It can be seen that the path for CRT has unnecessary turns, which extend the path compared to the paths generated by IPRM, IPRM* and IFMT, which attempt to find the shortest path within the inflated Voronoi boundary.

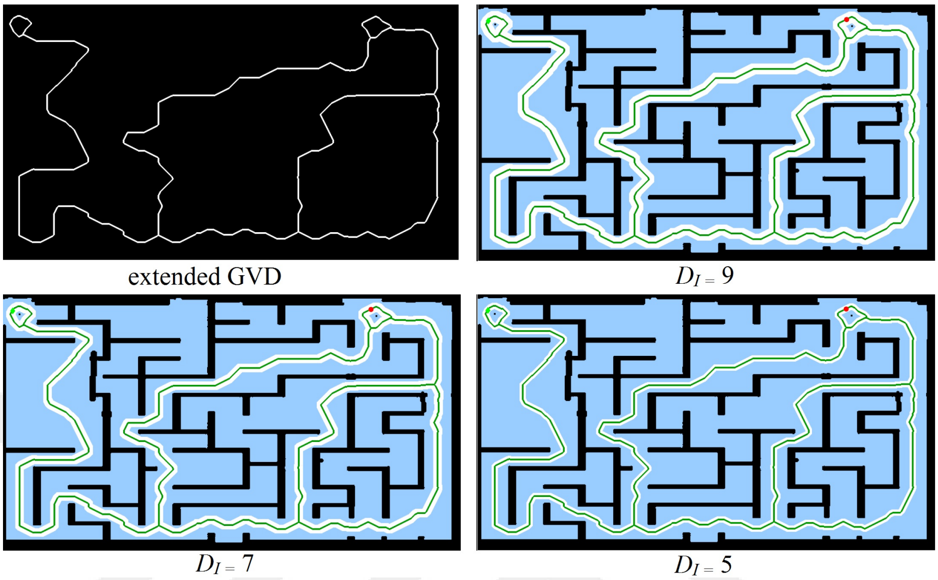

CRT IPRM

IPRM* IFMT

Figure 4.6: Solution paths for the polygon map.

4.3.3 Maze Map with Straight Lines

We next consider the maze map in Fig. 4.3 (b). The extended GVD and the inflated Voronoi boundary for this map are shown in Fig. 4.7 for $D_{\mathrm{I}} = 9$, $D_{\mathrm{I}} = 7$ and $D_{\mathrm{I}} = 5$, whereas Fig. 4.8 depicts the computational results for this map.


extended GVD

$D_I = 9$

$D_I = 7$

$D_I = 5$

Figure 4.7: Maze map: extended GVD and inflated Voronoi boundary.

We point out the following main observations from this experiment.

- Regarding the computation time, it can again be seen that the proposed algorithms are generally faster than the related classical algorithm, whereby the IPRM* algorithm is fastest.
- Path safety can be significantly increased compared to the classical algorithms when using the proposed algorithms. Moreover, an increase in path safety compared to the CRT algorithm is possible at a significantly reduced computation time, path length and variation of the obtained solutions. Here, the main reason for the increased computation time of the CRT algorithm is the generation of node samples in parts of the map that are not relevant for finding a solution path. The proposed algorithms only generate node samples along the possible routes along the inflated Voronoi boundary.

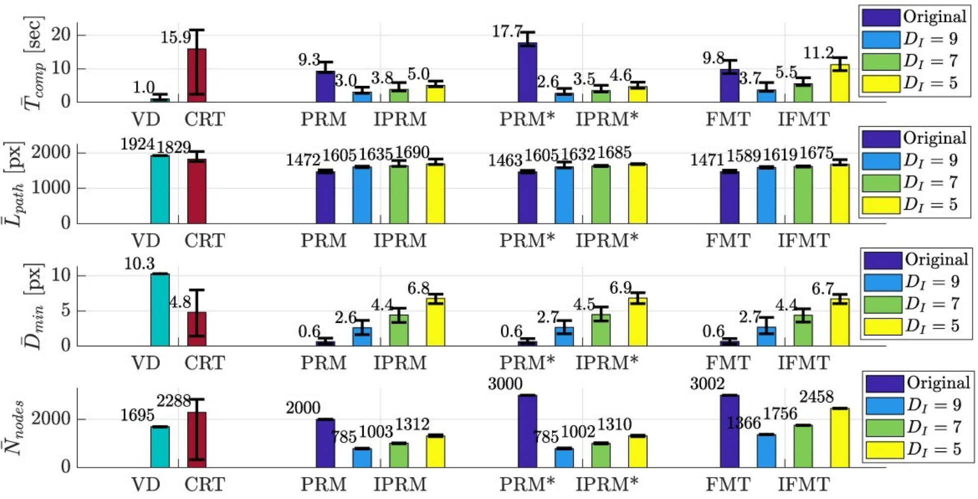


Figure 4.8: Comparison of the performance metrics for the maze map with straight lines.

Fig. 4.9 compares solutions paths of the different methods. Similar to previous section, CRT generates longer paths due to unnecessary turns when following straight passages. On the contrary, the solution paths of IPRM, IPRM* and IFMT use straight line connections in such passages.

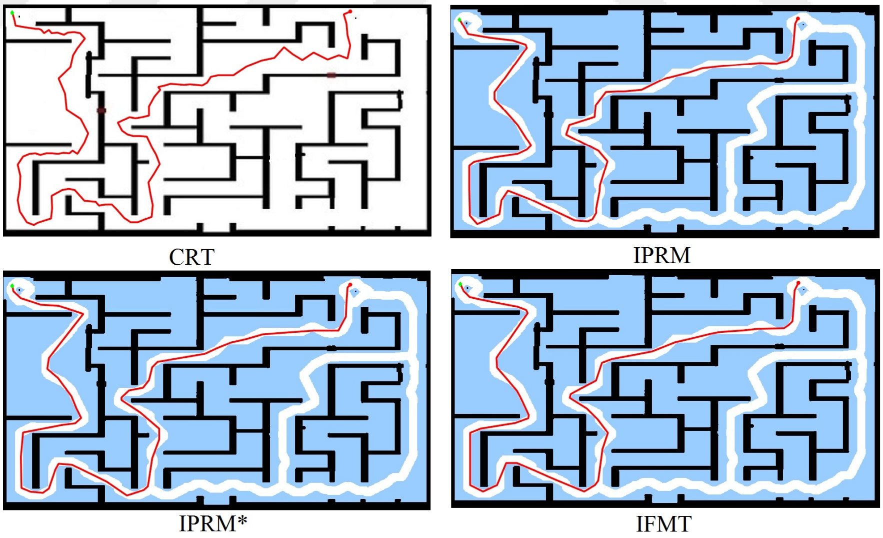
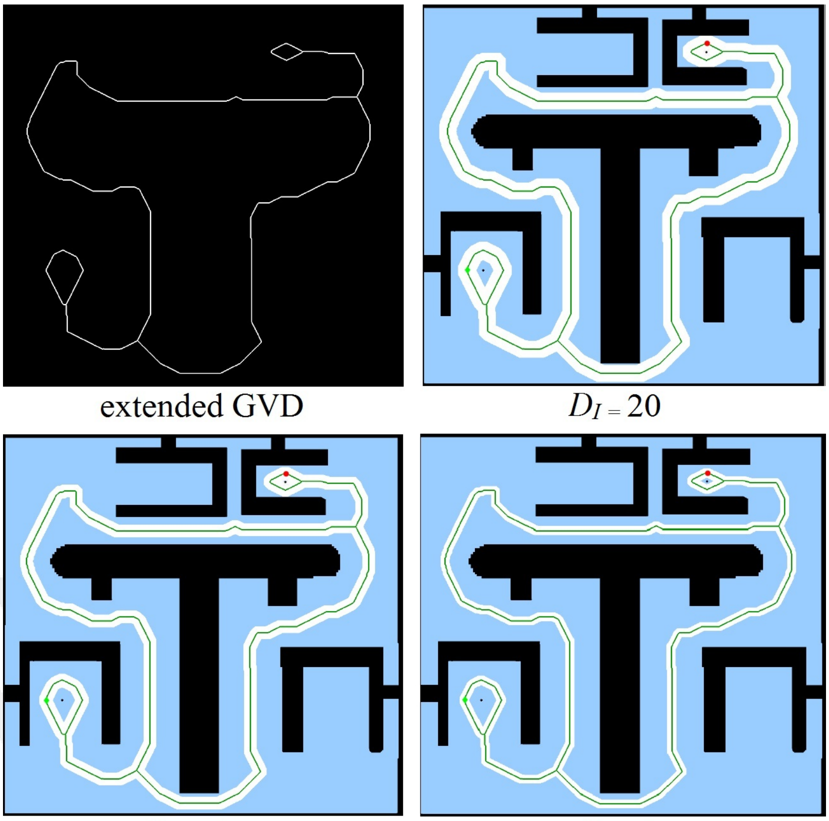


Figure 4.9: Solution paths for the maze map with straight lines.

4.3.4 U-Map

We further study the U-map in 4.3 (c). The extended GVD and the inflated Voronoi boundary for this map are shown in Fig. 4.10 for $D_\mathrm{I} = 20$, $D_\mathrm{I} = 15$ and $D_\mathrm{I} = 10$, whereas Fig. 4.8 depicts the computational results for this map.



extended GVD

$D_I = 20$

$D_I = 15$

$D_I = 10$

Figure 4.10: U-map: extended Voronoi diagram and inflated Voronoi boundary.

The main observations from this experiment are summarized as follows.

- The proposed algorithms lead to a reduced computation time compared to the existing algorithms except for the VD algorithm, which generates a very long path.

- The proposed algorithms allow adjusting path safety by selecting an appropriate value of D_{I}. For this environment, it has to be mentioned that the CRT algorithm generates safe paths with a similar path length as the proposed algorithms. Nevertheless, the CRT algorithm still leads to a larger computation time and significant variations in the minimum distance from the obstacle region. The main reason is that the CRT algorithm generates random samples that can be more or less close to the obstacle region depending on the found confidence values. On the other hand, the node samples of the proposed algorithms are restricted to the inflated Voronoi boundary such that the lower bound for $D_{\min}$ is well-defined.

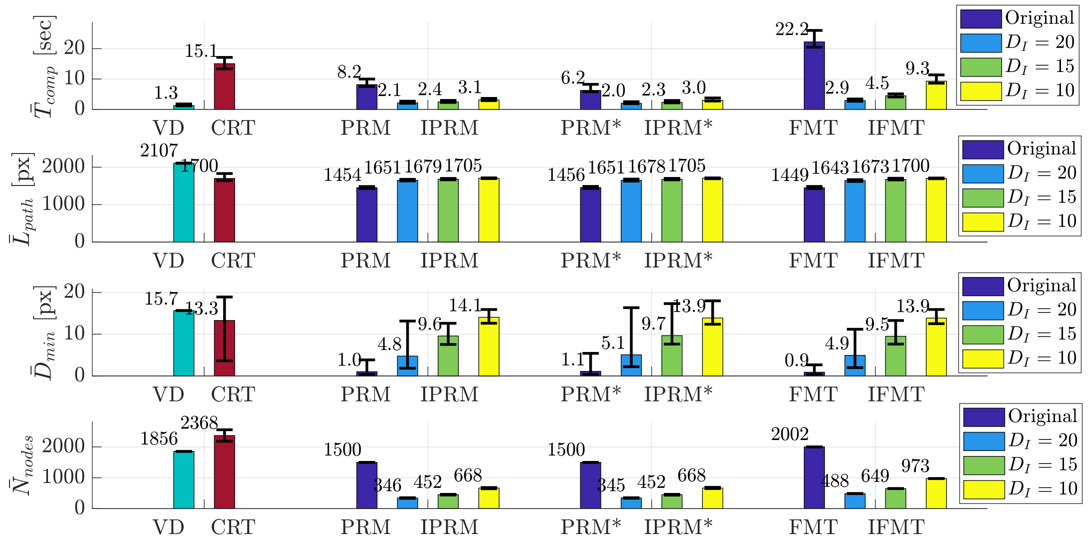


Figure 4.11: Comparison of the performance metrics for the U-map.

We further note that the solution paths in Fig. 4.12 confirm the observations from the previous sections and solution paths of CRT might come close to obstacles.

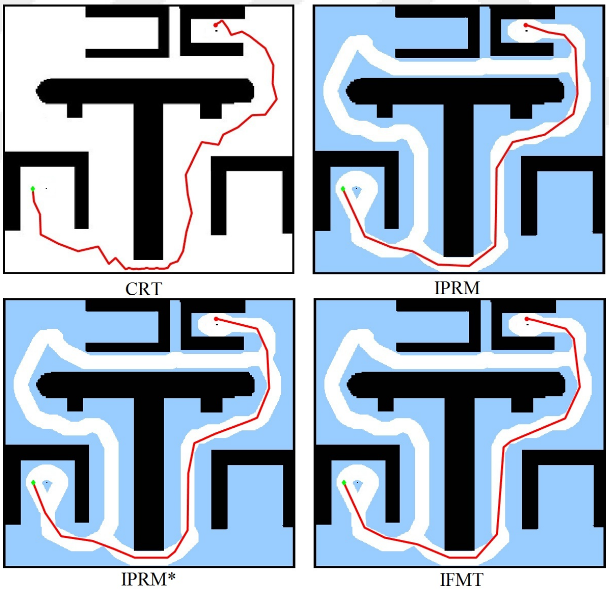
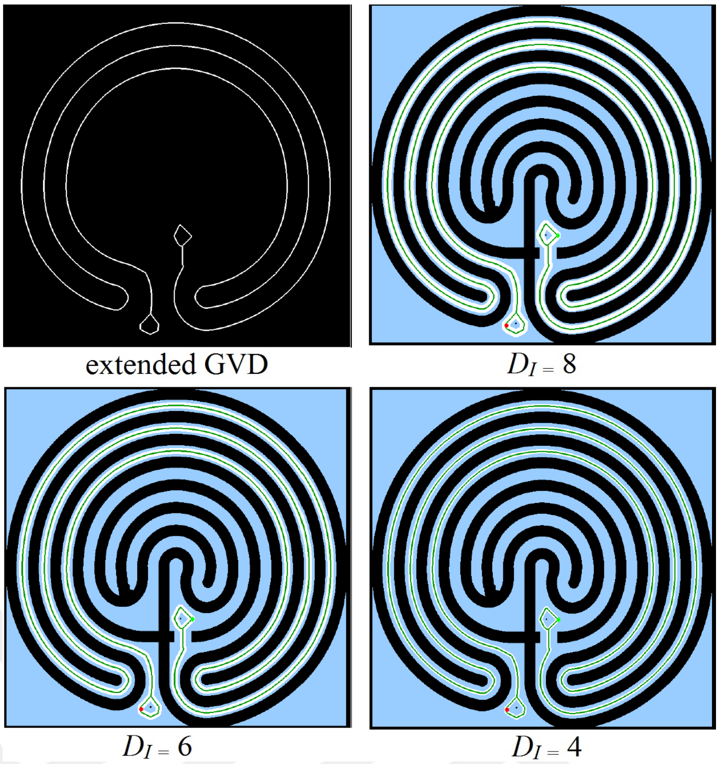


Figure 4.12: Solution paths for the U-map.

4.3.5 Maze Map with Spiral

We finally investigate the maze map in 4.3 (d) that has a very long solution path. The extended GVD and the inflated Voronoi boundary for this map are shown in Fig. 4.13 for $D_\mathrm{I} = 8$, $D_\mathrm{I} = 6$ and $D_\mathrm{I} = 4$, whereas Fig. 4.14 depicts the computational results for this map.



extended GVD

$D_I = 8$

$D_I = 6$

$D_I = 4$

Figure 4.13: Maze Map with Spiral: extended GVD and inflated Voronoi boundary.

We next describe the main observations from this experiment.

- The proposed algorithms mostly lead to significantly smaller computation times compared to the existing algorithms. It can only be observed that a very small value of D_I should be avoided due to the increase in the required number of nodes.

- For this map with a narrow space between obstacles, the proposed algorithms enable a significant increase of path safety without much increase in the path length compared to the classical methods. In addition, the proposed algorithms outperform the CRT algorithm in all performance metrics.

Finally, the solution paths in Fig. 4.15 again support the superiority of the proposed methods compared to CRT.

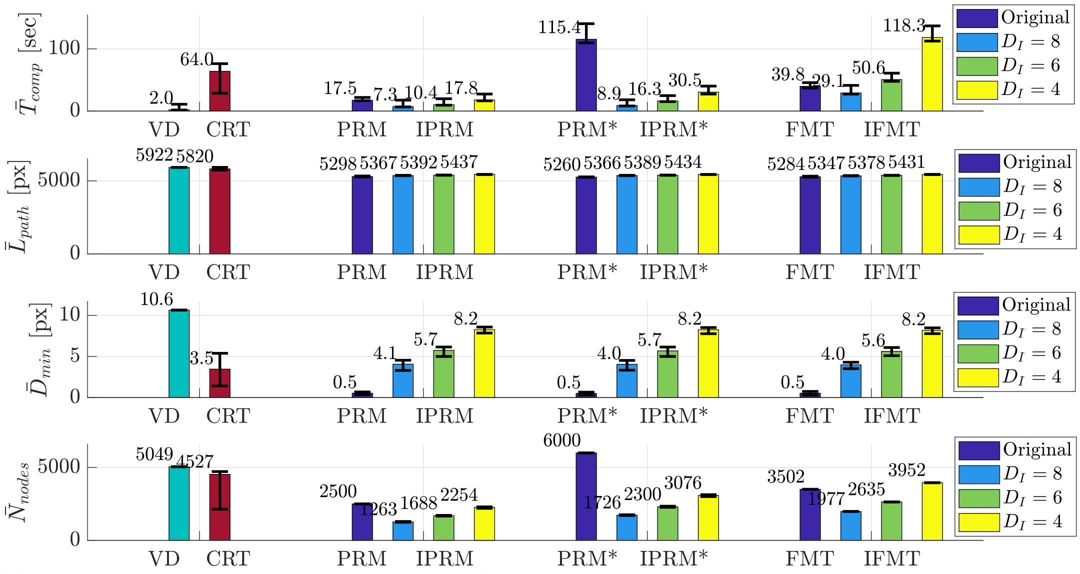


Figure 4.14: Comparison of the performance metrics for the map in Fig. 4.3 (d).

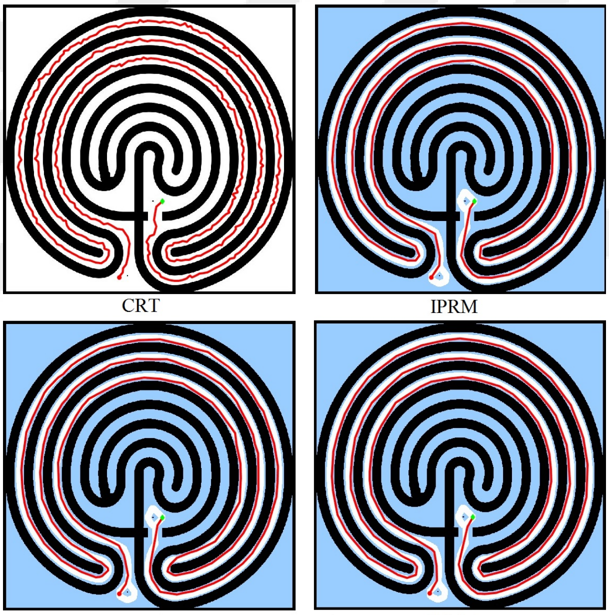


Figure 4.15: Solution paths for the maze map with spiral.

4.3.6 Discussion

Overall, the computational experiments for different environments indicate that the proposed methods are superior to both existing sampling-based methods such as PRM [41], PRM* [17] and FMT [19] as well as recent methods for path safety such as CRT [20]. In particular, the proposed IPRM algorithm outperforms the original PRM algorithm, the proposed IPRM* algorithm outperforms the original PRM* algorithm and the proposed IFMT algorithm outperforms the original FMT algorithm regarding both computation time and path safety while accepting a slight increase in path length.

Moreover, all the proposed algorithms (IPRM, IPRM* and FMT*) lead to a reduced computation time and path length while providing comparable path safety as the CRT algorithm. When comparing the proposed algorithms IPRM, IPRM* and IFMT, it can be observed that all of these algorithms provide similar results regarding computation time, path length and path safety. It is only the case that the IFMT algorithm leads to an increased computation time in case of a small width of the inflated Voronoi boundary and for environments with very long solution paths.

In this context, it has to be noted that the proposed algorithms benefit from confining the generated node samples to the inflated Voronoi boundary. This helps avoiding the exploration of irrelevant regions of $\mathscr{C}_{\text{free}}$. Moreover, this ensures the generation of reliable solution paths with small variations in the path length, minimum distance from obstacles and computation time.

CHAPTER 5

EFFICIENT PATH PLANNING METHOD BASED ON THE VORONOI BOUNDARY

The approaches proposed in the previous chapters use sampling-based methods in order to determine short and safe robot paths. A possible disadvantage of these methods is that they do not always find a solution path due to the random generation of samples. In addition, the approach in Chapter 3 does not use information about the topology of the robot environment, whereas the approach in Chapter 4 only uses the idea of an inflated Voronoi path without exploring the possible routes from the start to the goal position. Differently, the approach in this chapter precisely determines the possible routes in the environment map by refining paths along the edges of the Voronoi diagram. Section 5.1 describes our iterative refinement method step by step. Then, Section 5.2 performs a comprehensive evaluation of the proposed method based on state-of-the-art methods.

5.1 Voronoi Boundary Visibility for Efficient Path Planning

In this section, we develop our proposed method for the fast computation of short paths using information from the GVD as described in Section 2.1.2. Section 5.1.1 introduces a new method for connecting the start/goal point to the GVD and then defines a graph that captures the connectivity of the VB $\mathscr{V}$ using the BPs in $\mathscr{B}$. Then, Section 5.1.2 and 5.1.3 introduce the new Voronoi-Visibility (VV) algorithm and a method for its iterative application for computing short paths starting from the VB. Finally, Section 5.1.5 discusses how this method can be extended in case of additional safety requirements on robot paths.

5.1.1 Connection of the Start and Goal Position

As described in Section 2.1.3, one way of connecting p_{s} and p_{g} to the VB is finding the shortest collision-free path from $p_{\text{s}}/p_{\text{g}}$ to $\mathscr{V}$. In this section, we propose an alternative method that leads to shorter solution paths. In particular, for p_{s}, we determine all points on the VB that have an equal distance to p_{s} and to the obstacle region $\mathscr{C}_{\text{obs}}$. Formally, we determine the set $\mathscr{V}_{\text{s}} \subseteq \mathscr{V}$ such that

$$\forall p_j \in \mathscr{V}_{\text{s}}, d(p_j, p_{\text{s}}) = d(p_j, \mathscr{C}_{\text{obs}}). \tag{5.1}$$

Similarly, for p_{g}, we compute the set $\mathscr{V}_{\text{g}} \subseteq \mathscr{V}$ such that

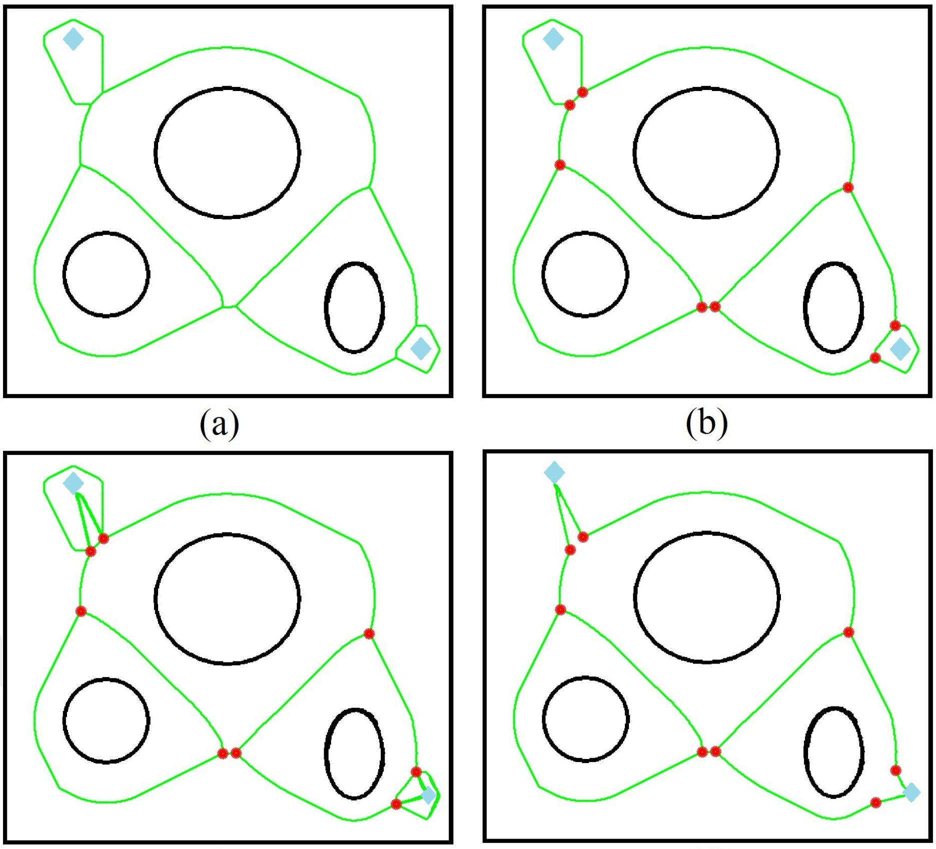
$$\forall p_j \in \mathscr{V}_{\text{g}}, d(p_j, p_{\text{g}}) = d(p_j, \mathscr{C}_{\text{obs}}). \tag{5.2}$$

These points are guaranteed to have a collision-free connection to p_{s} and p_{g}, respectively. Moreover, these points can be computed efficiently based on an image with a modified obstacle region $\hat{\mathscr{C}}_{\text{obs}} = \mathscr{C}_{\text{obs}} \cup \{p_{\text{s}}, p_{\text{g}}\}$. To this end, we follow the procedure described in Algorithm 5.

Algorithm 5 Connection of the start and goal point

```
1: Input: $\mathscr{C}$, $\mathscr{C}_{\text{obs}}$, $p_{\text{s}}$, $p_{\text{g}}$
2: Output: $\mathscr{V}$, $\mathscr{B}$
3: Initialize: $\hat{\mathscr{C}}_{\text{obs}} = \mathscr{C}_{\text{obs}} \cup \{p_{\text{s}}, p_{\text{g}}\}$
4: Determine the VB $\mathscr{V}$ for $\mathscr{C}$ and $\hat{\mathscr{C}}_{\text{obs}}$
5: Determine all BPs $\mathscr{B}$ on the VB
6: Determine $\mathscr{V}_{\text{s}} \subseteq \mathscr{B}$ as the BPs on the circle around $p_{\text{s}}$
7: Determine $\mathscr{V}_{\text{g}} \subseteq \mathscr{B}$ as the BPs on the circle around $p_{\text{g}}$
8: Connect $p_{\text{s}}/p_{\text{g}}$ to the respective BPs in $\mathscr{V}_{\text{s}}/\mathscr{V}_{\text{g}}$
9: Remove all segments of $\mathscr{V}$ on the circles around $p_{\text{s}}/p_{\text{g}}$
```

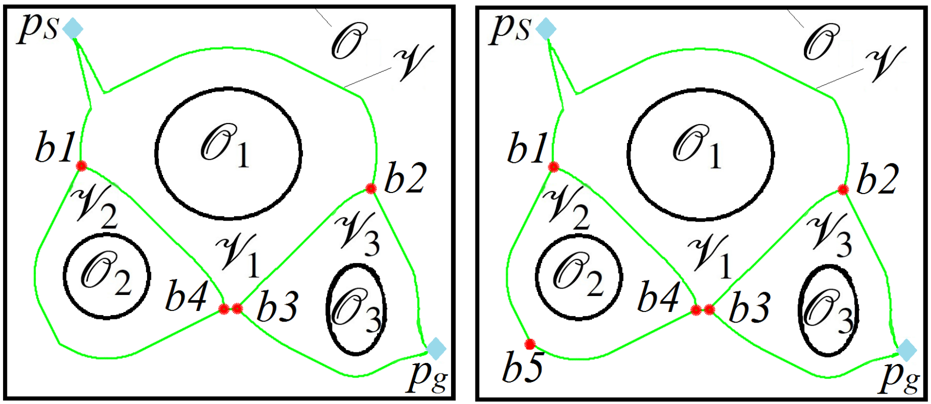
That is, the proposed algorithm first determines the GVD for image with the modified obstacle region $\hat{\mathscr{C}}_{\text{obs}} = \mathscr{C}_{\text{obs}} \cup \{p_{\text{s}}, p_{\text{g}}\}$ (line 3) with the VB $\mathscr{V}$. This GVD has the property that the start point p_{s} and the goal point p_{g} are encircled by the VB as illustrated in Fig. 5.1 (a). Then, the algorithm determines all BPs on $\mathscr{V}$ (line 5). The BPs on the circle around p_{s} are identified as $\mathscr{V}_{\text{s}}$ (line 6), whereas the BPs on the circle around p_{g} belong to $\mathscr{V}_g$ (line 7) as can be seen in Fig. 5.1 (b). Then, the points $p_{\text{s}}/p_{\text{g}}$ are connected to the corresponding BPs in $\mathscr{V}_{\text{s}}/\mathscr{V}_{\text{g}}$ (line 8). This procedure is depicted in Fig. 5.1 (c). Finally, the unnecessary parts of the circles around p_{s} and p_{g} are removed (line 9) to obtain the resulting VB $\mathscr{V}$ in Fig. 5.1 (d).



(a) (b) (c) (d)

Figure 5.1: Connection of p_{s} and p_{g} to $\mathscr{V}$: (a) GVD; (B) BPs; (c) Connection to BPs on $\mathscr{V}_{\mathrm{s}}$ and $\mathscr{V}_{\mathrm{g}}$; (d) Resulting $\mathscr{V}$.

Using the outputs $\mathscr{V}$, $\mathscr{B}$ of Algorithm 5, we next compute a graph $G = (V,E)$ that characterizes the connectivity of the VB. That is, the vertexes of G are defined as $V = \mathscr{B}$ and the edges E contain all (unordered) pairs $\{b_i, b_j\}$ with $b_i, b_j \in V$ such that there is a direct connection between b_i and b_j on $\mathscr{V}$. We further take into account the special case, where different connections between a pair of vertexes b_i, b_j exist on $\mathscr{V}$. This case can for example be seen in Fig. 5.2. Here, there are two connections between the BPs b_1 and b_4. In order to resolve this ambiguity, we simply insert an artificial BP on one of the connections (for example the longer one) and update the edges accordingly.



(a) (b)

Figure 5.2: (a) Original GVD with BPs; (b) Artificial BP b_5.

The graph for the example environment in Fig. 5.2 is given by the vertexes $V = \{p_s, p_g, b_1, b_2, b_3, b_4, b_5\}$ and the edges $E = \{\{p_s, b_1\}, \{p_s, b_2\}, \{b_1, b_4\}, \{b_1, b_5\}, \{b_2, b_3\}, \{b_2, p_g\}, \{b_3, p_g\}, \{b_3, b_4\}, \{b_4, b_5\}\}$. We finally label each edge $\{b_i, b_j\} \in E$ by the path length $|\mathscr{P}_{b_i, b_j}|$ on $\mathscr{V}$ between the BPs b_i and b_j. The labeled graph for the example environment is shown in Fig. 5.3.

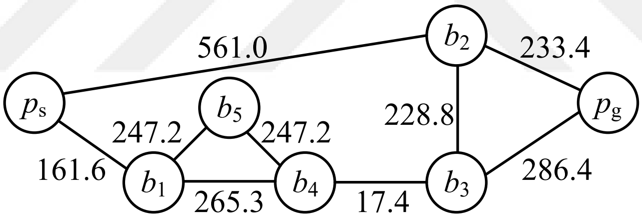


Figure 5.3: Graph for the example environment.

For the graph $G = (V, E)$, we define a walk in G by a finite sequence of vertexes $(v_1, v_2, \ldots, v_n)$ with $v_1, \ldots, v_n \in V$ such that each edge $\{v_i, v_{i+1}\} \in E$ for $i = 1, \ldots, n-1$. In particular, we are interested in walks from p_s to p_g, that is, $v_1 = p_s$ and $v_n = p_g$. In order to avoid confusion with the notion of a ”path” in the robot environment, we use the notion of a start-goal walk for a walk from p_s to p_g in G. Using G, it is possible to determine the shortest start-goal walk along $\mathscr{V}$ using Dijkstra’s algorithm [54]. Moreover, it is possible to compute the k shortest (cycle-free) start-goal walks using Yen’s algorithm [89]. Accordingly, we introduce the notation $W_1, \ldots, W_k$ to denote the k shortest (collision-free) start-goal walks from p_s to p_g along $\mathscr{V}$. We further note that the maximum number of different shortest start-goal walks depends on the topology of each environment and can also be determined by Yen’s algorithm. For the example environment, there are 6 different start-goal walks, which are given in Table 5.1.

Table 5.1: $k = 6$ shortest start-goal walks for the example

Path	Length [px]
$W_1 = (p_\mathrm{s}, b_1, b_4, b_3, p_\mathrm{g})$	730.2
$W_2 = (p_\mathrm{s}, b_2, p_\mathrm{g})$	794.4
$W_3 = (p_\mathrm{s}, b_1, b_4, b_3, b_2, p_\mathrm{g})$	906.0
$W_4 = (p_\mathrm{s}, b_1, b_5, b_4, b_3, p_\mathrm{g})$	959.3
$W_5 = (p_\mathrm{s}, b_2, b_3, p_\mathrm{g})$	1076.2
$W_6 = (p_\mathrm{s}, b_1, b_5, b_4, b_3, b_2, p_\mathrm{g})$	1135.1

For each edge $\{v_i, v_{i+1}\} \in E$, we write $\mathscr{P}_{v_i,v_{i+1}} \subseteq \mathscr{V}$ for the corresponding path between the points $v_i, v_{i+1} \in V$ (recall that $V = \mathscr{B}$). Then, the path $\mathscr{P}_{W_j} \subseteq \mathscr{V}$ that corresponds to a start-goal walk $W_j = (v_1, \ldots, v_n)$, is given by the concatenation of the paths $\mathscr{P}_{v_1,v_2}, \ldots, \mathscr{P}_{v_{n-1},v_n}$ (note that $v_1 = p_\mathrm{s}$ and $v_n = p_\mathrm{g}$). Accordingly, any path $\mathscr{P}_{W_j}$, $j = 1, \ldots, k$, constitutes a solution of the path planning problem and there is no solution of the path planning problem if $k = 0$.

5.1.2 Voronoi Boundary Visibility Algorithm

It is a well-known fact that robot paths following the VB $\mathscr{V}$ take unnecessary turns and are hence comparably long [90, 51, 52, 55, 30, 50]. Accordingly, we suggest to first employ a shortcut heuristic to reduce the path length. Our method uses the information obtained from the VB $\mathscr{V}$ and the corresponding graph G in Section 5.1.1 to determine a collision-free path $P_\mathrm{VV} = (p_1, \ldots, p_{|P_\mathrm{VV}|})$ such that $\{p_1, \ldots, p_{|P_\mathrm{VV}|}\} \subseteq \mathscr{V}$.

Since our algorithm is based on the visibility of points on $\mathscr{V}$, we denote it as Voronoi Boundary Visibility (VV). We next explain the VV algorithm following the pseudo-code given in Algorithm 6. It is based on an initial solution path $P = (p_1, \ldots, p_{|P|})$ that is for example obtained from a start-goal walk W_j as in Section 5.1.1 and the obstacle region $\mathscr{C}_\mathrm{obs}$ (line 1). The algorithm then determines two different sequences of waypoints to be followed. In the first case ($l = 1$ in line 13), the algorithm follows the given path P from p_s to p_g. In the second case ($l = 2$), the waypoints of P are ordered in the reverse direction from p_g to p_s (line 5). P_VV is initialized with p_1, which is equal to p_s for $l = 1$ and to p_g for $l =$ (line 5). Moreover, the points p_cur and p_last keep track of the current point to be explored and the last collision-free connection point, respectively. The algorithm loops over all points in P (line 6). It is then checked if the straight-line connection from p_cur to p_i is collision-free (flag = **true** in line 7). If flag is **true**, p_last is updated since there is no collision on the straight-line connection from p_cur to p_i (line 9). If flag is **false**, it holds that the straight-line connection from p_cur to p_i intersects with the obstacle region $\mathscr{C}_\mathrm{obs}$ (line 10). Nevertheless, we know from the previous iteration (where flag must have been **true** that the straight-line connection from p_{cur} to p_{last} is collision-free. Hence, we accept p_{last} on the solution path (line 11) and restart the search for a collision-free connection from p_{last} (line 12 and 13). The search for new points terminates if a connection to the goal position p_{g} is found (line 14). The algorithm finally compares the two solution paths P_1 (for $l = 1$) and P_2 (for $l = 2$) and returns the shorter one as the result. In the sequel, we will denote the path P_{VV} resulting from Algorithm 6 as a VV-path.

Algorithm 6 `ComputeVV`$(P, \mathscr{C}_{\text{obs}})$

```
 1: Input: $P$, $\mathscr{C}_{\text{obs}}$
 2: Output: $P_{\text{VV}}$
 3: for $l = 1,2$ do
 4:     if $l = 2$ then
 5:         $\forall i = 1,\ldots,|P|$: $p_i = p_{|P|-i+1}$
        Initialize: $P_l = (p_1)$; $p_{\text{cur}} = p_1$; $p_{\text{last}} = p_1$
 6:     for $k = 2,\ldots,|P|1$ do
 7:         flag $=$ CollisionFree$(p_{\text{cur}}, p_i, \mathscr{C}_{\text{obs}})$
 8:         if flag $=$ true then
 9:             $p_{\text{last}} = p_i$
10:         else
11:             $P_l = (P_l, p_{\text{last}})$
12:             $p_{\text{cur}} = p_{\text{last}}$
13:             $l = l - 1$
14:         if $p_{\text{last}} = p_{|P|}$ then
15:             break
16: return $\arg\min_{P_1,P_2}\{L(P_1), L(P_2)\}$
```

For illustration, we compute the VV-paths for W_1, W_2, W_3 and W_5 in Table 5.1. That is, for $i = 1,2,3,5$, we apply Algorithm 6 with the initial path P_{W_i}. The resulting VV-paths are shown in Fig. 5.4.

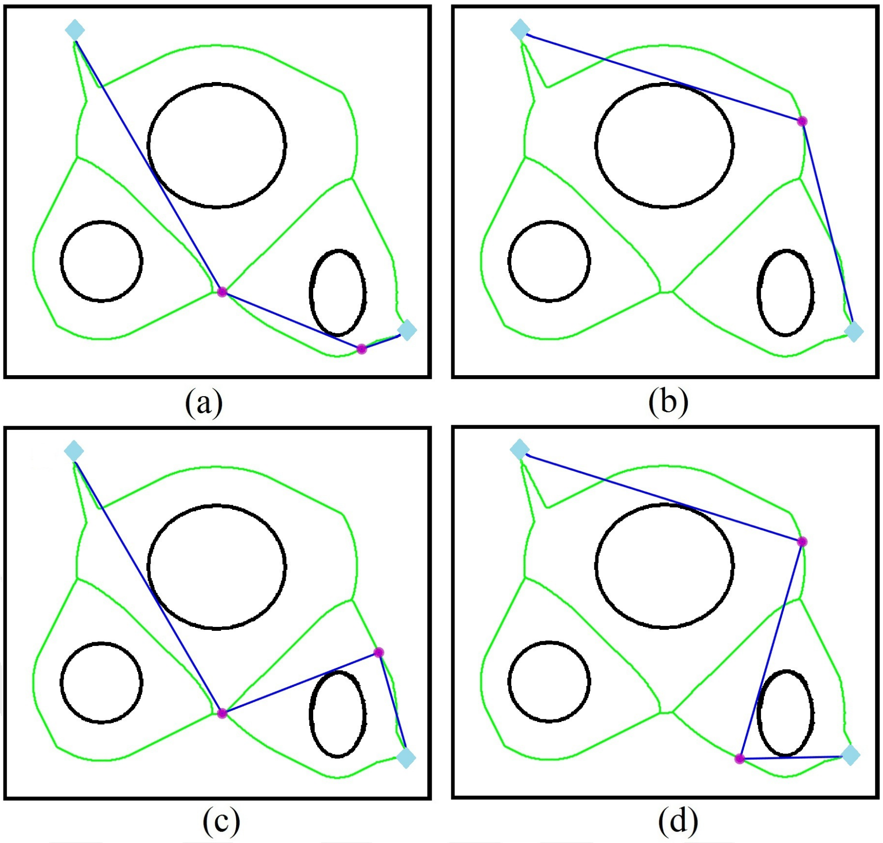


Figure 5.4: VV-path illustration for: (a) P_{W_1} with $L(P_{\mathrm{VV}}) = 657.1$; (b) P_{W_2} with $L(P_{\mathrm{VV}}) = 677.5$; (c) P_{W_3} with $L(P_{\mathrm{VV}}) = 755.0$; (d) P_{W_5} with $L(P_{\mathrm{VV}}) = 840.1$.

Remark 1. *We note that shortcut heuristics with the same objective of reducing the number of waypoints and removing unnecessary turns were introduced in [90, 78, 91, 92, 30]. Different from our algorithm, the algorithm in [91, 92, 30] suggests to iteratively remove points from a given path P if the connection between its adjacent points is collision-free. Although this method is also able to reduce the path length, we will show in Section 5.2 that our algorithm generally leads to shorter paths. For later usage, we refer to the algorithm in [30] (denoted as "Remove Redundancy") as*

$$P_{RR} = \mathtt{RemoveRedundancy}(P, \mathscr{C}_{obs}).$$

The algorithm in [78] also uses the idea of checking connections to waypoints until a collision is detected. Differently, that algorithm only evaluates the path from p_s to p_g but omits the path from p_g to p_s. It is obtained from Algorithm 6 by iterating only for $l = 1$ in line 13.

For illustration, we show two paths that are obtained for the initial path P_{W_1} of the example environment. Fig. 5.5 (a) and (b) depict the solution paths P_{VV} and P_{RR} obtained from `ComputeVV` and `RemoveRedundancy`, respectively. It can be seen that both paths select waypoints on $\mathscr{P}_{W_1}$. Hereby, P_{VV} defines a shorter connection since `computeVV` looks ahead more.

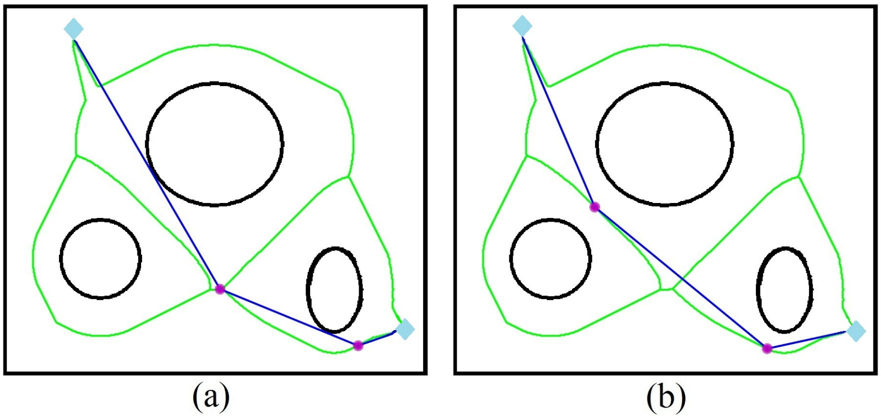

(a) (b)

Figure 5.5: (a) P_{VV} with $L(P_{\mathrm{VV}}) = 657.1$; (b) P_{RR} with $L(P_{\mathrm{RR}}) = 666.2$.

5.1.3 VV with Steiner Points

The VV-paths computed by Algorithm 6 are able to avoid unnecessary turns when following the VB as can be seen in Fig. 5.4. Nevertheless, it is still the case that VV-paths have unnecessary corners that increase the path length. Accordingly, we suggest to apply the technique of adding Steiner points in order to cut corners. We note that this technique was first introduced in [91, 92, 30]. Nevertheless, in this work we propose a particular combination with Algorithm 6 and the graph G that both reduces the computation time and enables the reduction of the path length.

We first formalize the technique in [91, 92, 30] in Algorithm 7 for later usage in our improved algorithms. Algorithm 7 computes a solution path P_{S} (line 2) based on a given feasible path P that is for example obtained from Algorithm 6, the obstacle region $\mathscr{C}_{\mathrm{obs}}$ and a distance value Δ (line 1). The solution path is initialized with the given path (line 3) and the algorithm terminates if P_{S} only contains the start and goal position (line 5). Otherwise, the algorithm repeats the following iteration (line 6 to 22): The algorithm tries to reduce the path length by looking at 3 consecutive points p_{L}, p, p_{R} (line 7) starting from $p_{\mathrm{s}} = P_{\mathrm{S}}[1]$. Hereby, the notation $P_{\mathrm{S}}[i]$ denotes the i-th waypoint in P_{S} and the main idea is to remove a potential corner with p when moving from p_{L} to p_{R}. Then, the algorithm tries to put new points p_{SL} and p_{SR} between p, p_{L} and p, p_{R}. These points are generated at increasing distances $k \cdot \Delta$ from p (line 12). p_{SL} and p_{SR} are accepted as new waypoints if their straight-line connection is collision-free (line 15). No more new points can be generated if there is a collision (line 17) or the generated points do no longer lie between p, p_{L} and p, p_{R} (line 11). After completing the generation of new points, the algorithm checks if any new points could be found (line 18). If there are new points, the original point p is removed from P_{S} and replaced by $\hat{p}_{\mathrm{SL}}$ and $\hat{p}_{\mathrm{SR}}$ (line 19). Then, the algorithm continues from the next unvisited waypoint (line 20 or 22). If the goal position p_{g} is reached (line 9), the inner loop (line 9) terminates. If no more improvement in the solution path P_{S} can be achieved (line 6), P_{S} is returned.

Algorithm 7 `ComputeST`$(P, \mathscr{C}_{\mathrm{obs}}, \Delta)$

```
 1: Input: Collision-free path $P = (p_1, \dots, p_{|P|})$ such that $p_1 = p_{\mathrm{s}}$ and $p_{|P|} = p_{\mathrm{g}}$; $\mathscr{C}_{\mathrm{obs}}$; $\Delta$
 2: Output: Solution path $P_{\mathrm{S}}$
 3: Initialize: $P_{\mathrm{S}} = P$; $n_{\mathrm{old}} = \infty$
 4: if $|P_{\mathrm{S}}| = 2$ then
 5:     return $P_{\mathrm{S}}$
 6: while $|P_{\mathrm{S}}| \neq n_{\mathrm{old}}$ do
 7:     $n_{\mathrm{old}} = |P_{\mathrm{S}}|$; $p = P_{\mathrm{S}}[2]$; $p_{\mathrm{L}} = P_{\mathrm{S}}[1]$; $p_{\mathrm{R}} = P_{\mathrm{S}}[3]$
 8:     $d_{\mathrm{L}} = d(p_{\mathrm{L}}, p)$; $d_{\mathrm{R}} = d(p_{\mathrm{R}}, p)$; $u_{\mathrm{L}} = \frac{p_{\mathrm{L}} - p}{d_{\mathrm{L}}}$; $u_{\mathrm{R}} = \frac{p_{\mathrm{R}} - p}{d_{\mathrm{R}}}$
 9:     while $p \neq p_{\mathrm{g}}$ do
10:         $\hat{p}_{\mathrm{SL}} = p$; $\hat{p}_{\mathrm{SR}} = p$; $k = 1$
11:         while $k \cdot \Delta < d_{\mathrm{L}}$ and $k \cdot \Delta < d_{\mathrm{R}}$ do
12:             $p_{\mathrm{SL}} = p + u_{\mathrm{L}} \cdot k \cdot \Delta$; $p_{\mathrm{SR}} = p + u_{\mathrm{R}} \cdot k \cdot \Delta$
13:             flag = CollisionFree$(p_{\mathrm{L}}, p_{\mathrm{R}}, \mathscr{C}_{\mathrm{obs}})$
14:             if flag = true then
15:                 $\hat{p}_{\mathrm{SL}} = p_{\mathrm{SL}}$; $\hat{p}_{\mathrm{SR}} = p_{\mathrm{SR}}$
16:             else
17:                 break
18:         if $\hat{p}_{\mathrm{SL}} \neq p$ then
19:             $P_{\mathrm{S}} = (p_{\mathrm{s}}, \dots, p_{\mathrm{L}}, \hat{p}_{\mathrm{SL}}, \hat{p}_{\mathrm{SR}}, p_{\mathrm{R}}, \dots, p_{\mathrm{g}})$
20:             $p = \hat{p}_{\mathrm{SL}}$
21:         else
22:             $p = p_{\mathrm{R}}$
23: return $P_{\mathrm{S}}$
```

Fig. 5.6 illustrates the successive application of `computeVV` in Algorithm 6 and Algorithm 7 for the initial paths P_{W_1} and P_{W_2} in Fig. 5.4. In both cases, it can be seem that additional waypoints are introduced in order to remove unnecessary corners in the VV-paths.

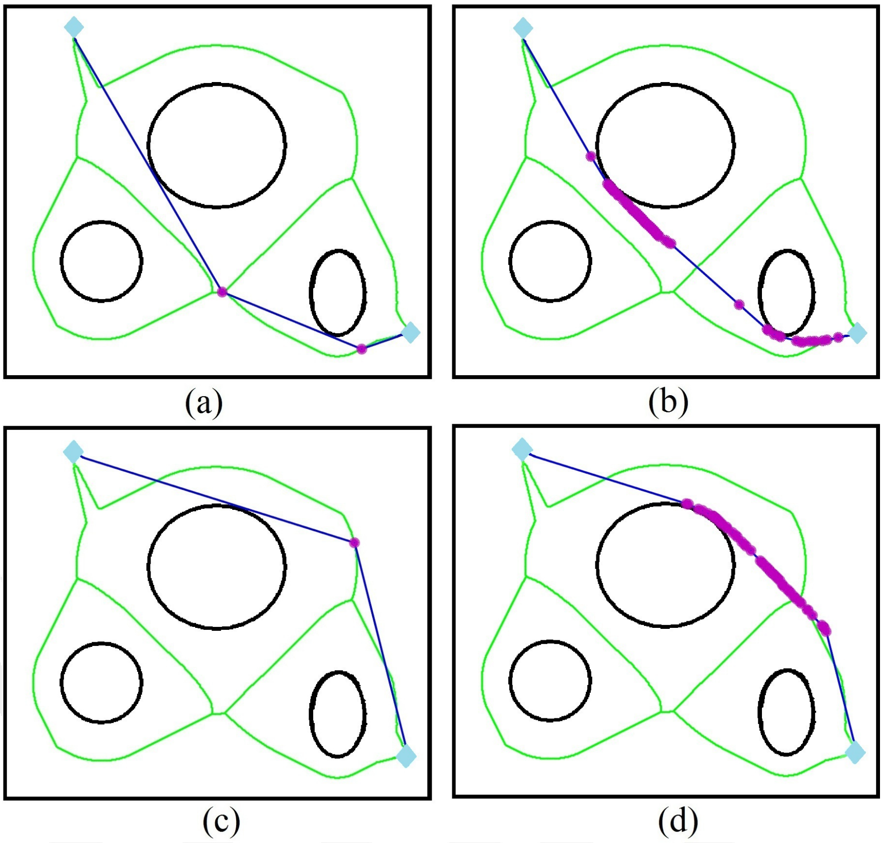

(a) (b) (c) (d)

Figure 5.6: (a) Algorithm 6 for P_{W_1} with $L(P_{\mathrm{VV}}) = 657.1$; (b) Algorithm 6 and 7 for P_{W_1} with $L(P_{\mathrm{S}}) = 637.0$; (c) Algorithm 6 for P_{W_2} with $L(P_{\mathrm{VV}}) = 677.5$; (d) Algorithm 6 and 7 for P_{W_2} with $L(P_{\mathrm{S}}) = 645.4$.

5.1.4 Overall Path Computation

As can be seen in Fig. 5.6, the length of solution paths can be reduced when applying Algorithm 6 and 7 consecutively. Nevertheless, it also has to be noted that the solution path P_{S} of Algorithm 7 can have an increased number of waypoints since it repeatedly adds new waypoints. That is, shorter connections between some of the added waypoints might be possible. Accordingly, we next define two particular combinations of Algorithm 6 and 7 for the efficient computation of short solution paths with a small number of waypoints. The first method is stated in Algorithm 8. Here, the input parameters Δ_{init} and Δ_{min} represent the initial and minimum distance value for the refinement according to Algorithm 7. The algorithm first computes the VB $\mathscr{B}$ and the set of BPs $\mathscr{B}$ according to Algorithm 5 (line 3) and then determines the k shortest paths $P_{W_1}, \ldots, P_{W_k}$ from p_{s} to p_{g} along the VB following the graph construction in Section 5.1.1 (line 4). Then, the shortest path is initialized with the shortest path P_{W_1} along the VB and the algorithm tries to reduce all the paths P_{W_k}, $i = 1, \ldots, k$ (line 6 to 14). In each iteration, the solution candidate P is initialized with P_{W_i}. The algorithm first applies the VV algorithm (line 8) and then repeatedly inserts points in P in order to remove corners (line 10). Hereby, Δ is decreased until the minimum resolution given by Δ_{min} is reached (line 9 and 12). Moreover, `ComputeVV` is repeatedly applied (line 11) in order to keep the number of points in P_{S} small. At the end of each iteration, the solution candidate is updated if the current path P is shorter than the previously found paths (line 14). The algorithm returns the shortest path P_{S} found among all the candidates (line 15).

```
Algorithm 8 VV-ST-R($\mathscr{C}, \mathscr{C}_{\text{obs}}, p_{\text{s}}, p_{\text{g}}, k, \Delta_{\text{init}}, \Delta_{\text{min}}$)
 1: Input: $\mathscr{C}$; $\mathscr{C}_{\text{obs}}$; $p_{\text{s}}$; $p_{\text{g}}$;$\Delta_{\text{init}}$; $\Delta_{\text{min}}$.
 2: Output: Solution path $P_{\text{S}}$
 3: Compute $\mathscr{V}$ and $\mathscr{B}$ using Algorithm 5
 4: Compute the $k$ shortest paths $P_{W_1}, \ldots, P_{W_k}$ as described in Section 5.1.1
 5: Initialize: $P_{\text{S}} = P_{W_1}$
 6: for $i = 1, \ldots, k$ do
 7:     $P = P_{W_1}$; $\Delta = \Delta_{\text{init}}$
 8:     $P =$ ComputeVV($P, \mathscr{C}_{\text{obs}}$)
 9:     while $\Delta \geq \Delta_{\text{min}}$ do
10:         $P =$ ComputeST($P, \mathscr{C}_{\text{obs}}, \Delta$)
11:         $P =$ ComputeVV($P, \mathscr{C}_{\text{obs}}$)
12:         $\Delta = \Delta/2$
13:     if $L(P) < L(P_{\text{S}})$ then
14:         $P_{\text{S}} = P$
15: return $P_{\text{S}}$
```

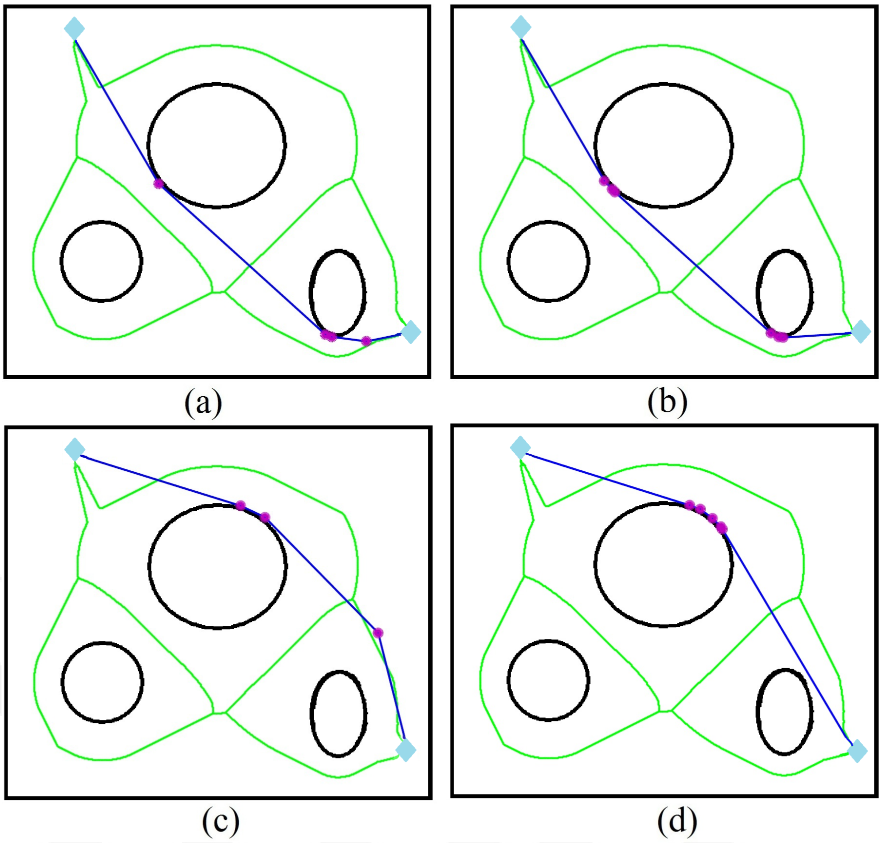
The second method is a modification of Algorithm 8 that is stated for comparison with the work in [30]. The main difference of Algorithm 8 and 9 is the usage of `ComputeVV`. While this function is used in each iteration of the loop in Algorithm 8, it is only used at the end of Algorithm 9 (line 12).

Algorithm 9 `VV-ST`($\mathscr{C},\mathscr{C}_{\text{obs}},p_{\text{s}},p_{\text{g}},k,\Delta_{\text{init}},\Delta_{\text{min}}$)

```
 1: Input: $\mathscr{C}$; $\mathscr{C}_{\text{obs}}$; $p_{\text{s}}$; $p_{\text{g}}$;$\Delta_{\text{init}}$; $\Delta_{\text{min}}$.
 2: Output: Solution path $P_{\text{S}}$
 3: Compute $\mathscr{V}$ and $\mathscr{B}$ using Algorithm 5
 4: Compute the $k$ shortest paths $P_{W_1},\ldots,P_{W_k}$ as described in Section 5.1.1
 5: Initialize: $P_{\text{S}} = P_{W_1}$
 6: for $i = 1,\ldots,k$ do
 7:     $P = P_{W_1}$; $\Delta = \Delta_{\text{init}}$
 8:     $P =$ ComputeVV($P,\mathscr{C}_{\text{obs}}$)
 9:     while $\Delta \geq \Delta_{\text{min}}$ do
10:         $P =$ ComputeST($P,\mathscr{C}_{\text{obs}},\Delta$)
11:         $\Delta = \Delta/2$
12:     $P =$ ComputeVV($P,\mathscr{C}_{\text{obs}}$)
13:     if $L(P) < L(P_{\text{S}})$ then
14:         $P_{\text{S}} = P$
15: return $P_{\text{S}}$
```

Remark 2. *We note that Algorithm 9 is shown in this work in order to evaluate the improvements of our method compared to [91, 92, 30]. In particular, using* $k = 1$ *and replacing "*`ComputeVV`*" by "*`RemoveRedundany`*" in line 8 and 12 leads to the algorithm in [30]. The comprehensive evaluation in Section 5.2 will reveal that both Algorithm 8 and 9 lead to considerable reductions in the path length without increasing the computation time. We further note that we evaluated different methods for creating additional waypoints. For example, [90] uses the idea of Bisection in order to generate Steiner points on neighboring edges of a solution path. Nevertheless, there was no considerable effect on the final result (regarding path length and computation time) when using different methods.*

For illustration, we apply Algorithm 8 and 9 to the paths P_{W_1} and P_{W_2} of the example environment as shown in Fig. 5.7. Due to the repeated application of the VV algorithm in Algorithm 8, it is the case that shorter solution paths are found by this algorithm. Specifically, we get $L(P_{\text{S}}) = 633.9$, $L(P_{\text{S}}) = 635.6$, $L(P_{\text{S}}) = 631.6$ and $L(P_{\text{S}}) = 644.6$ for the solution paths in Fig. 5.7 (a), (b), (c) and (d), respectively. It is further interesting to note that the solution path of Algorithm 8 for the second-shortest start-goal walk W_2 along VB is shorter than the solution path for the shortest start-goal walk W_1 along VB. This demonstrates the advantage of taking into account several short start-goal walks instead of only the shortest start-goal walk.



(a) (b)

(c) (d)

Figure 5.7: (a) Algorithm 9 for P_{W_1} with $L(P_\mathrm{S}) = 635.6$; (b) Algorithm 8 for P_{W_1} with $L(P_\mathrm{S}) = 633.9$; (c) Algorithm 9 for P_{W_2} with $L(P_\mathrm{S}) = 644.6$; (d) Algorithm 8 for P_{W_2} with $L(P_\mathrm{S}) = 631.6$.

5.1.5 Inflated Obstacle Region

In addition to finding a shortest path in a given environment, it is frequently required to determine a path that keeps a specified safety distance D_S from the obstacle region $\mathscr{C}_\mathrm{obs}$ [30, 20, 29, 34]. That is, it is desired for the solution path P_s that

$$d(\mathscr{P}_{P_\mathrm{S}}, \mathscr{C}_\mathrm{obs}) > D_\mathrm{S}. \tag{5.3}$$

In order to address this issue within the proposed methodology, we define the inflated obstacle region $\mathscr{C}_\mathrm{obs}(D_\mathrm{S})$ that contains all points in $\mathscr{C}$, whose distance from $\mathscr{C}_\mathrm{obs}$ is less or equal to D_S:

$$\mathscr{C}_\mathrm{obs}(D_\mathrm{S}) = \{p \in \mathscr{C} | d(p, \mathscr{C}_\mathrm{obs}) \leq D_\mathrm{S}\}. \tag{5.4}$$

In particular, it holds that $\mathscr{C}_\mathrm{obs} = \mathscr{C}_\mathrm{obs}(0)$. A viable method for determining $\mathscr{C}_\mathrm{obs}(D_\mathrm{S})$ in a digital map is to inflate $\mathscr{C}_\mathrm{obs}$ using the morphological dilation operation

$$\mathscr{C}_\mathrm{obs}(D_\mathrm{S}) = \mathscr{C}_\mathrm{obs} \oplus B_{D_\mathrm{S}} = \bigcup_{b \in B_{D_\mathrm{S}}} \mathscr{V}_{\mathrm{obs},b}, \tag{5.5}$$

whereby B_{D_S} represents a disk with radius D_S, $\oplus$ represents the dilation operation and $\mathscr{C}_{\text{obs},b}$ is the translation of $\mathscr{C}_{\text{obs}}$ by $b \in B_{D_S}$.

Using $\mathscr{C}_{\text{obs}}(D_S)$ instead of $\mathscr{C}_{\text{obs}}$, all the algorithms in the previous sections (Algorithm 5 to 9) can be applied in order to find short solution paths that keep a distance of at least D_S to the obstacle region $\mathscr{C}_{\text{obs}}$. In this case, we first compute $\mathscr{C}_{\text{obs}}(D_s)$ using (5.5) and determine the VB for the resulting map with inflated obstacle region using Algorithm 5. After that, we apply Algorithm 8 using $\mathscr{C}_{\text{obs}}(D_S)$ to obtain a solution path P_{VV}.

In order to illustrate the proposed method, we apply the proposed method to the example environment. The inflated obstacle region and the resulting safe solution paths for $D_I = 5$ and $D_I = 10$ are shown in Fig. 5.8. As can be seen from Fig. 5.7 (d) and Fig. 5.8, the solution paths become longer as the safety distance increases. This is expected since the available free space for the robot motion is reduced for larger values of D_S.

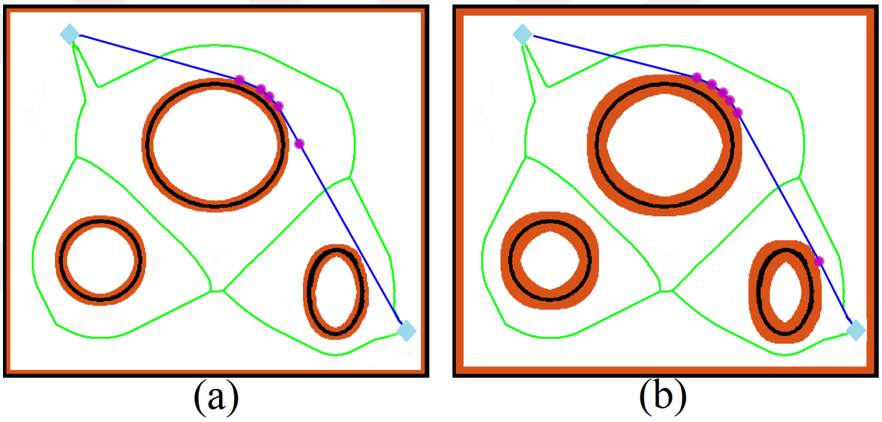

(a) (b)

Figure 5.8: Inflated obstacle region for the example environment: (a) $\mathscr{C}_{\text{obs}}(5)$ with $L(P_S) = 636.9$; (b) $\mathscr{C}_{\text{obs}}(10)$ with $L(P_S) = 642.4$.

5.2 Evaluation

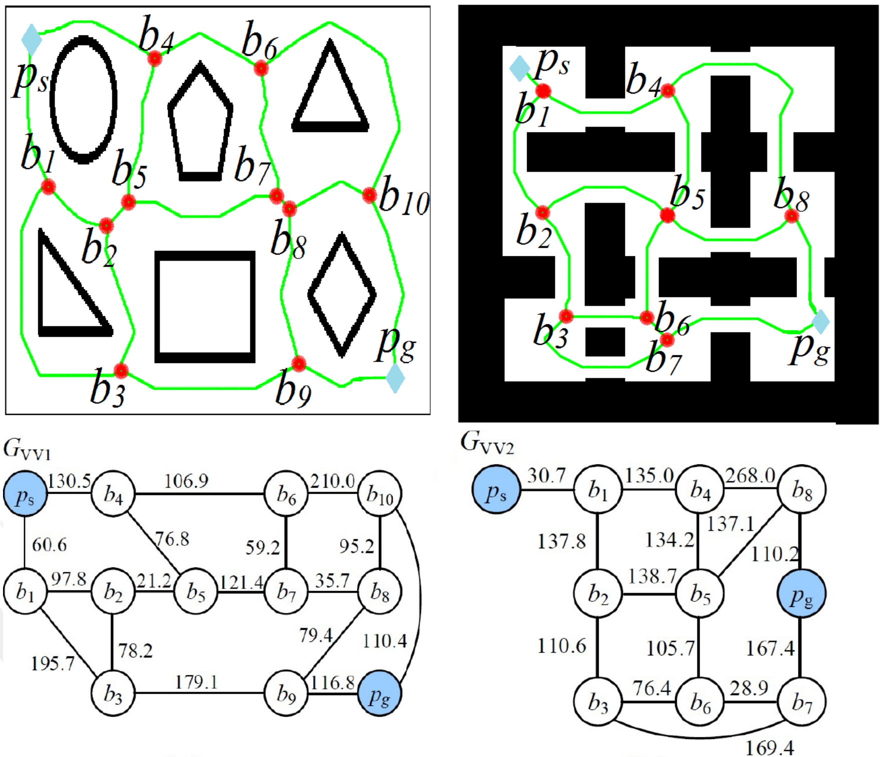
In this section, we evaluate the solution paths P_{VV} of the proposed algorithms regarding the obtained path length $L(P_{VV})$ and the computation time T_{VV} for obtaining the solution path. To this end, we compare the obtained solutions to several state-of-the-art methods [30, 17, 18, 19, 20] for a large variety of environment maps. All the algorithms are implemented in Matlab [93] and run under the same conditions on a Laptop with Intel(R) Core(TM) i7-4510 CPU @ 2.60Ghz processor and 6GB RAM. Section 5.2.1 investigates the properties of the VV algorithm for maps with different properties and Section 5.2.2 provides a comprehensive comparison of different algorithms.

5.2.1 Properties of the VV Algorithm

We first analyze the properties of the proposed path planning algorithm depending on its constituent components. That is, we evaluate the improvements achieved when applying Algorithm 8 compared to Algorithm 6 and 9. For comparison, we also show the results when using the Voronoi-Dijkstra (VD) algorithm in Section 2.1.3, the RR algorithm and the RR-ST algorithm according to [30] and the PRM* algorithm in [17]. Hereby, the latter algorithm is applied with a large number of 15 000 samples and serves as a reference due to its proven convergence to a minimum path.

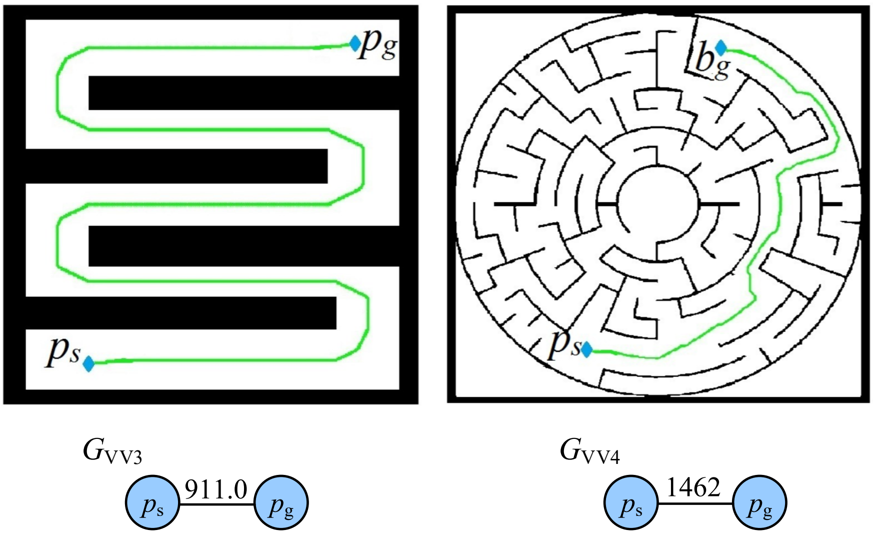
Our evaluation in this section is based on four maps which feature different properties and that are shown together with their graphs in Fig. 5.9 and 5.10. Since this section focuses on the evaluation of the VV algorithm, we denote these maps as VV1, VV2, VV3, VV4. The maps VV1 and VV2 in Fig. 5.9 allow for multiple start-goal walks with similar length in their respective graphs G_{VV1} and G_{VV2}. Hereby, VV1 provides scattered circular and polygonal shapes that leave sufficient space for the robot motion, whereas the maze map VV2 offers tight passages as well as regions with free space. The maps VV3 and VV4 in Fig. 5.10 have a single start-goal walk. Although there is sufficient space for the robot motion in VV3, this maps requires sharp turns of the robot. Differently, the map VV4 is a maze map with curved obstacle boundaries and tight passages. For each map, we consider three scenarios with safety distances of $D_{\mathrm{S}} = 0$ (original map), $D_{\mathrm{S}} = 5$ and $D_{\mathrm{S}} = 10$ in order to validate the performance of our algorithm in the case of additional safety requirements as specified in Section 5.1.5.

We first compare the different methods regarding the path length. The obtained results for the different maps are shown in Fig. 5.11 to 5.14. It is readily observed that VV-ST-R in Algorithm 8 produces the shortest paths among the VB-based algorithms for all the maps and for all the considered safety distances D_{S}. It can also be seen that replacing the function `RemoveRedundancy` in [30] by the proposed VV-algorithm (Algorithm 6) already leads to a considerable improvement. That is, VV improves on RR and VV-ST improves on RR-ST. Moreover, an additional improvement is achieved by VV-ST-R since it iteratively removes corners by adding Steiner points and then reduces the number of waypoints using the VV-algorithm. Hereby, it is interesting to note that VV-ST-R produces path lengths that are either very close to or shorter than the ones from the close-to-optimal PRM* paths.



(a) (b)

Figure 5.9: Environment maps and corresponding graphs: (a) VV1; (b) VV2.



(a) (b)

Figure 5.10: Environment maps and corresponding graphs: (a) VV3; (b) VV4

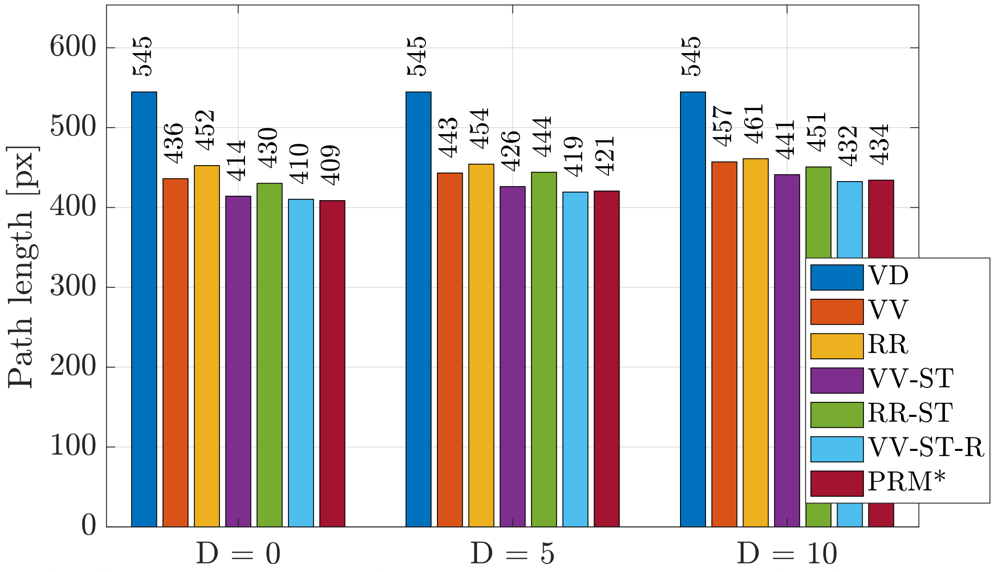


Figure 5.11: Path length comparison for VV1.

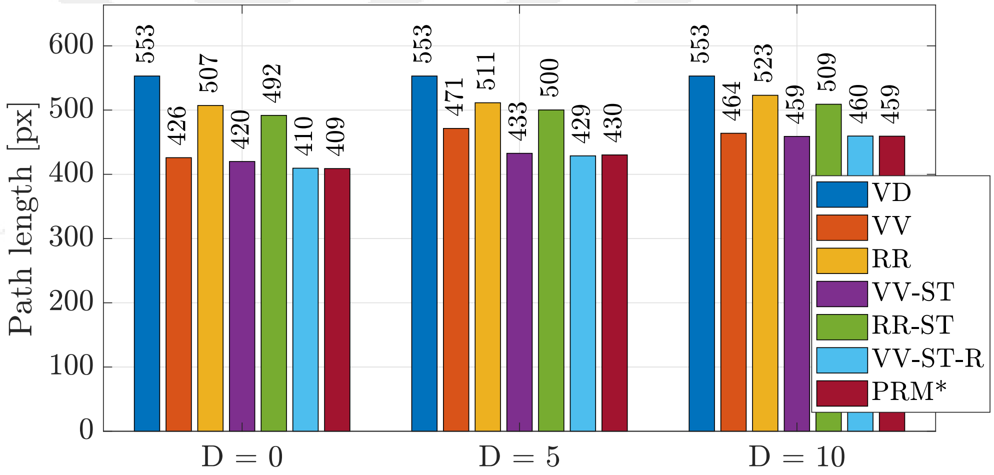


Figure 5.12: Path length comparison for VV2.

In addition, our results regarding the computation time are depicted in Fig. 5.15 to 5.18. Here, we note that the computation time for PRM* is not shown in these figures since it exceeds 10 min and the computation time for all the algorithms includes the computation of the GVD, which is the prerequisite of all the VB-based algorithms. It holds that all algorithms show a similar computation time with slight variations depending on the chosen map. Hereby, it is interesting to point out that applying the VV-algorithm repeatedly (VV-ST-R) as suggested in Algorithm 8 reduces the path lengths compared to Algorithm 9 and the algorithm in [30] without increasing the computation time.

In addition to the computational results, Fig. 5.19 to Fig. 5.22 show a selection of solution paths for different maps and methods (all related figures are shown in the

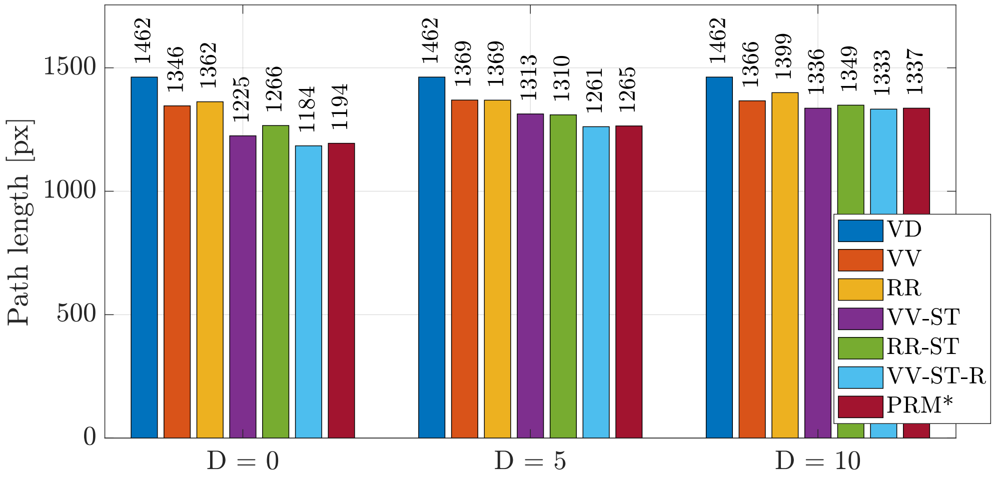


Figure 5.13: Path length comparison for VV3.

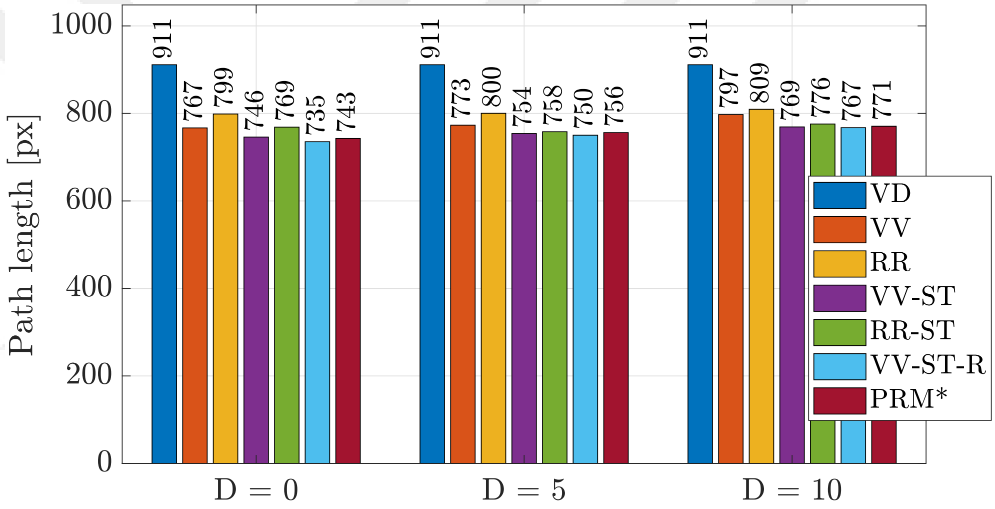


Figure 5.14: Path length comparison for VV4.

supplementary material). Fig. 5.19 and 5.20 illustrate that the solution paths of Algorithm 8 do not necessarily follow the shortest start-goal walk, which is different from the algorithm in [30]. Furthermore, Fig. 5.21 and 5.22 highlight that the paths generated by Algorithm 8 are able to tightly follow the shape of obstacles, while favoring straight-line connections in free space. As a result, these paths are even shorter than the paths obtained from applying the PRM*-algorithm with a large number of nodes.

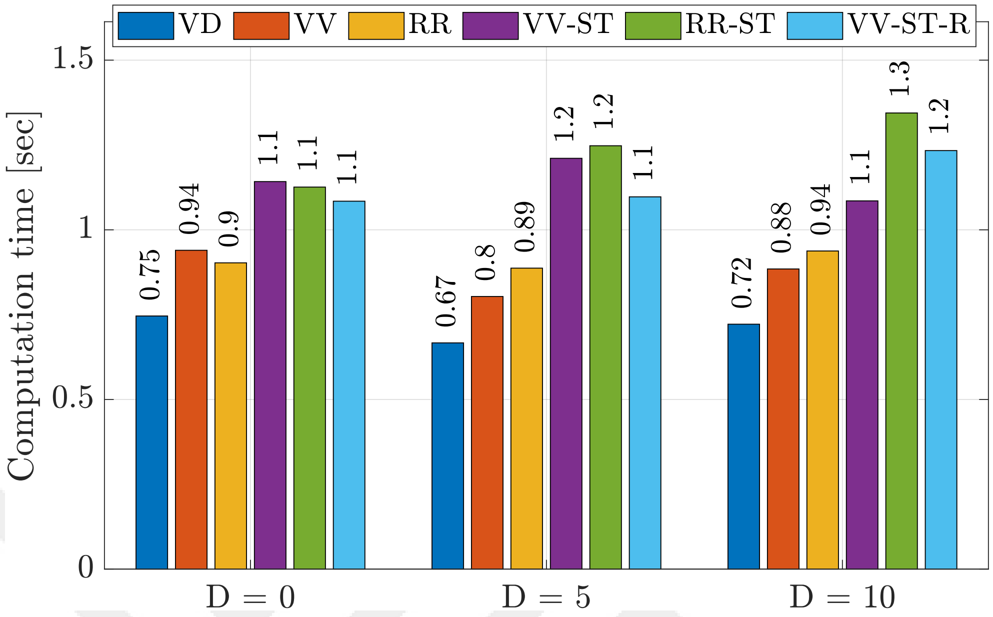


Figure 5.15: Computation time comparison for VV1.

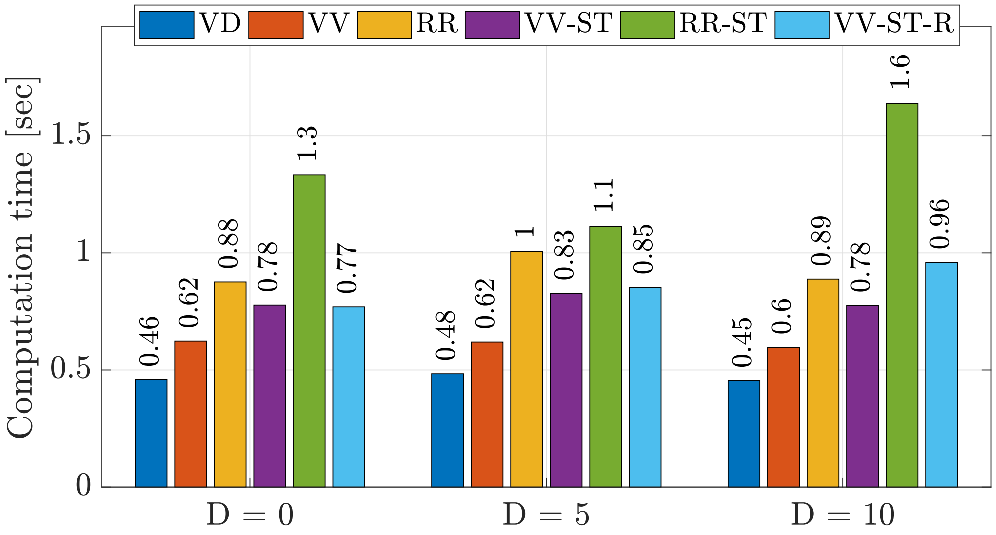


Figure 5.16: Computation time comparison for VV2.

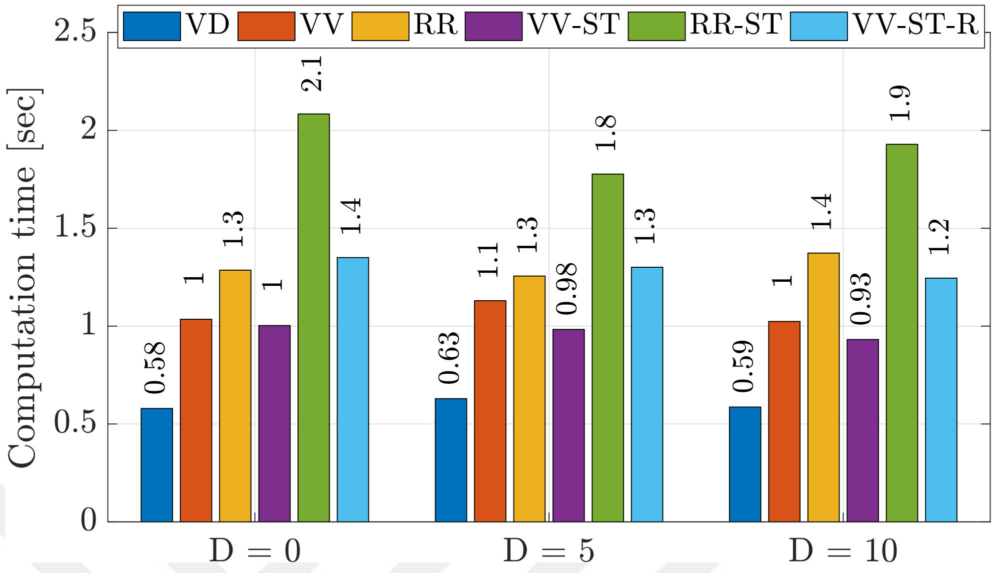


Figure 5.17: Computation time comparison for VV3.

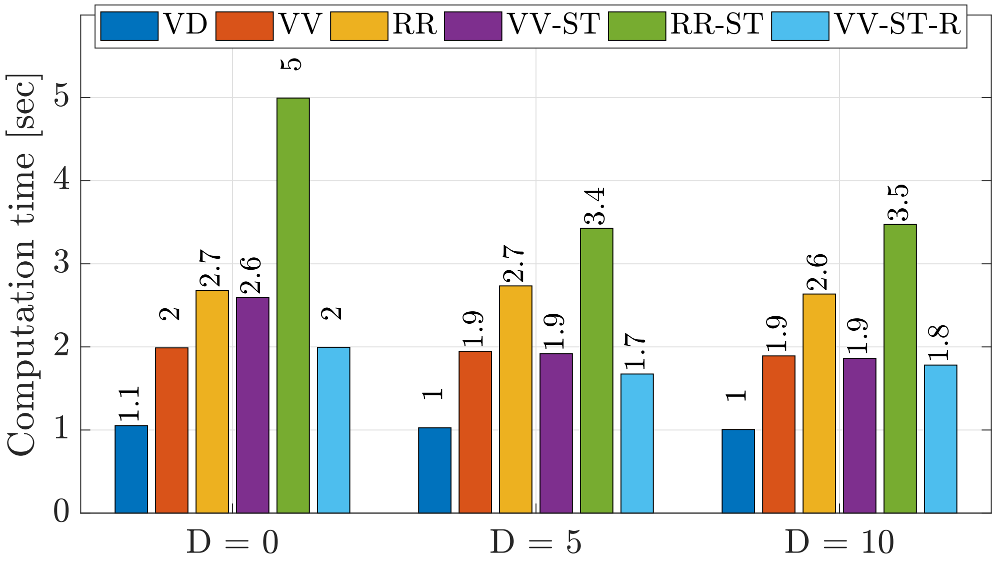
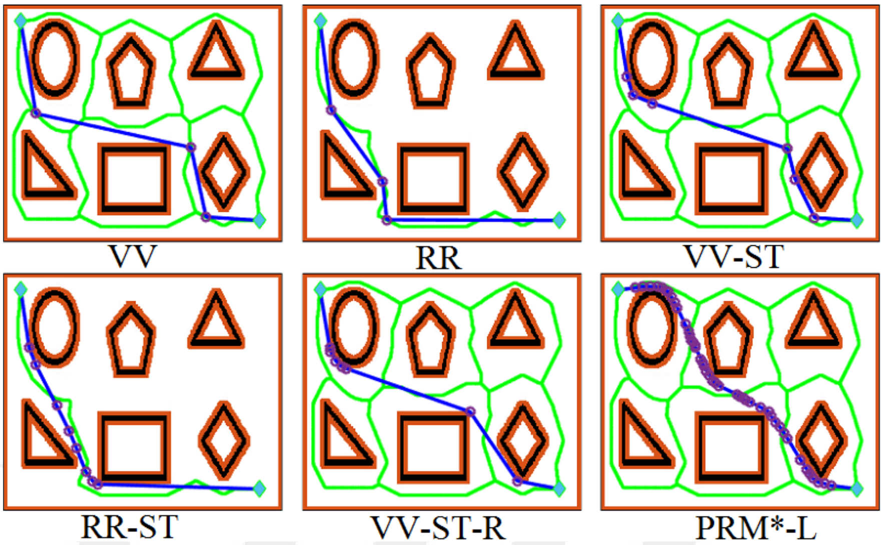


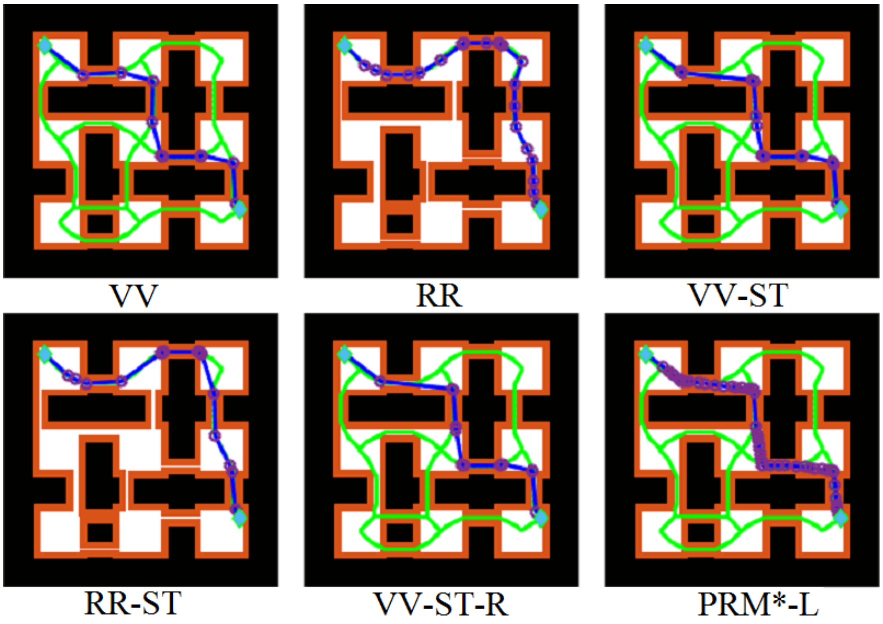
Figure 5.18: Computation time comparison for VV4.



VV RR VV-ST

RR-ST VV-ST-R PRM*-L

Figure 5.19: Solution paths for VV1 and $D_S = 5$.



VV RR VV-ST

RR-ST VV-ST-R PRM*-L

Figure 5.20: Solution paths for VV2 and $D_S = 10$.

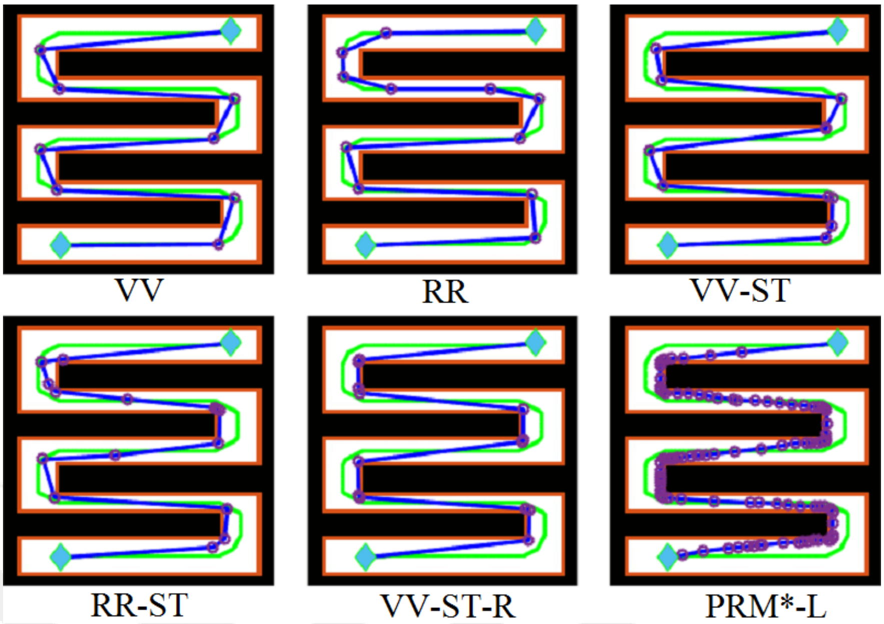

VV RR VV-ST
RR-ST VV-ST-R PRM*-L

Figure 5.21: Solution paths for VV3 and $D_S = 5$.

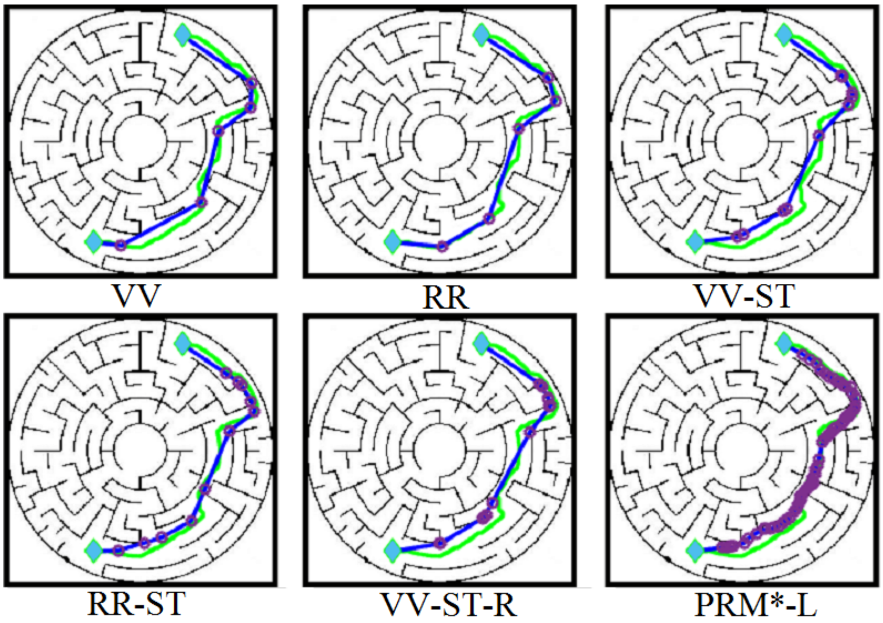

VV RR VV-ST
RR-ST VV-ST-R PRM*-L

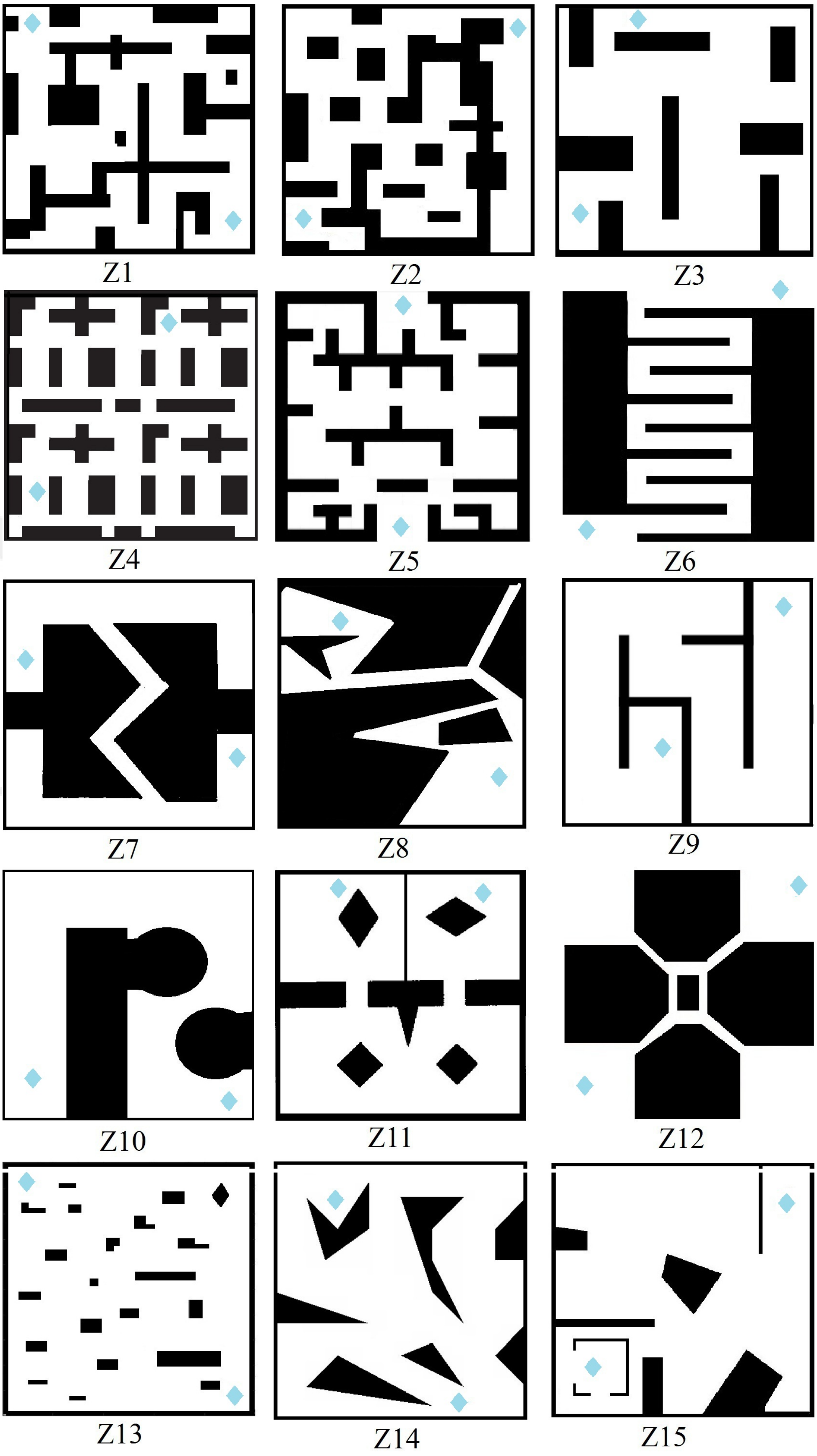
Figure 5.22: Solution paths for VV4 and $D_S = 0$.

5.2.2 Comparison with State-of-the-Art Algorithms

The results in the previous section indicate that the VV-ST-R algorithm outperforms other VB-based algorithms. We next perform a comparison of this algorithm with several state-of-the-art sampling-based path planning methods. To this end, we consider the 15 different maps in Fig. 5.23 that were previously used in path planing applications.

For the comparison, we choose the PRM* (Probabilistic Roadmap) and RRT* (Rapidly Exploring Random Tree) algorithm since they are probabilistically complete and asymptotically optimal [17]. In addition, we apply the FMT (Fast Marching Tree) algorithm that is supposed to generate shorter paths than PRM* and RRT* with a reasonable computation time without being asymptotically optimal [19]. Finally, the very recent CRT (Confidence Random Tree) algorithm is tailored for finding solution paths that keep a safe distance from obstacles without computing the GVD [20]. For all these sampling-based algorithms, we take the average of 100 trials and we select a suitable number of nodes/samples in order to ensure finding a solution path in more than 98% of the trials. In addition, similar to the previous section, we perform runs of the PRM* algorithm with 15 000 nodes as a reference for close-to-optimal paths.

The results of our computational experiments are summarized in Table 5.2 and 5.3. For convenience the cells of the method with the shortest path and the shortest computation time are shaded in gray. As the first main observation, it can be seen that the VV-ST-R algorithm always generates the shortest path among the algorithms that produce a solution path in a practical time. In addition, VV-ST-R has the smallest computation time in almost all of the cases. As a further advantage, the VV-ST-R algorithm is guaranteed to find a solution path if such path exists since it starts from the connection of start and goal point along the VB. The implication of this fact can for example be be seen when inspecting the computation time for the map Z7 or Z12 for $D_{\mathrm{S}} = 5$. Here, PRM* and FMT require a large number of nodes in order to find a solution path since (i) there is only a narrow passage between obstacles and (ii) most of the sampled nodes are generated in the obstacle region or in the part of the free space that does not contribute to the solution path. In contrast, VV-ST-R is able to shorten the initial solution path, which is given by the connection of p_{s} and p_{g} along the VB.



Z1 Z2 Z3

Z4 Z5 Z6

Z7 Z8 Z9

Z10 Z11 Z12

Z13 Z14 Z15

Figure 5.23: Maps for the comparative evaluation.

Table 5.2: Path length comparison

Map	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15
	$D_S = 0$														
VV-ST-R	720	661	533	486	500	959	931	651	1234	915	651	456	533	585	551
PRM*-Long	727	671	542	490	508	958	945	655	1267	925	660.7	457	536	601	557
RR-ST	769	693	577	498	511	1027	983	658	1421	1005	700	462	541	616	564
PRM*	787	690	571	511	509	1009	1012	675	1379	981	699	462	547	617	592
RRT*	1086	856	819	655	748	1302	1350	912	2004	1316	1102	476	663	817	735
FMT	756	678	551	495	511	1046	955	674	1289	954	671	463	537	596	568
CRT	878	806	665	606	590	1209	1151	775	1484	1138	809	509	613	682	657
	$D_S = 5$														
VV-ST-R	746	686	553	503	511	1111	992	677	1277	946	677	470	539	600	575.4
PRM*-Long	757	691	564	511	514	1106	997	684	1305	955	688	471	541	607	582
RR-ST	778	720	562	540	530	1116	1053	701	1459	995	723	472	557	619	583
PRM*	765	698	638	525	531	1124	1020	691	1486	1006	723	473	559	652	603
RRT*	1148	876	850	707	823	1277	1324	853	2024	1317	1174	479	676	846	745
FMT	759	698	605	521	525	1123	1020	703	1363	986	705	475	547	627	592
CRT	884	797	667	606	590	1208	1151	777	1483	1131	808	508	613	684	658

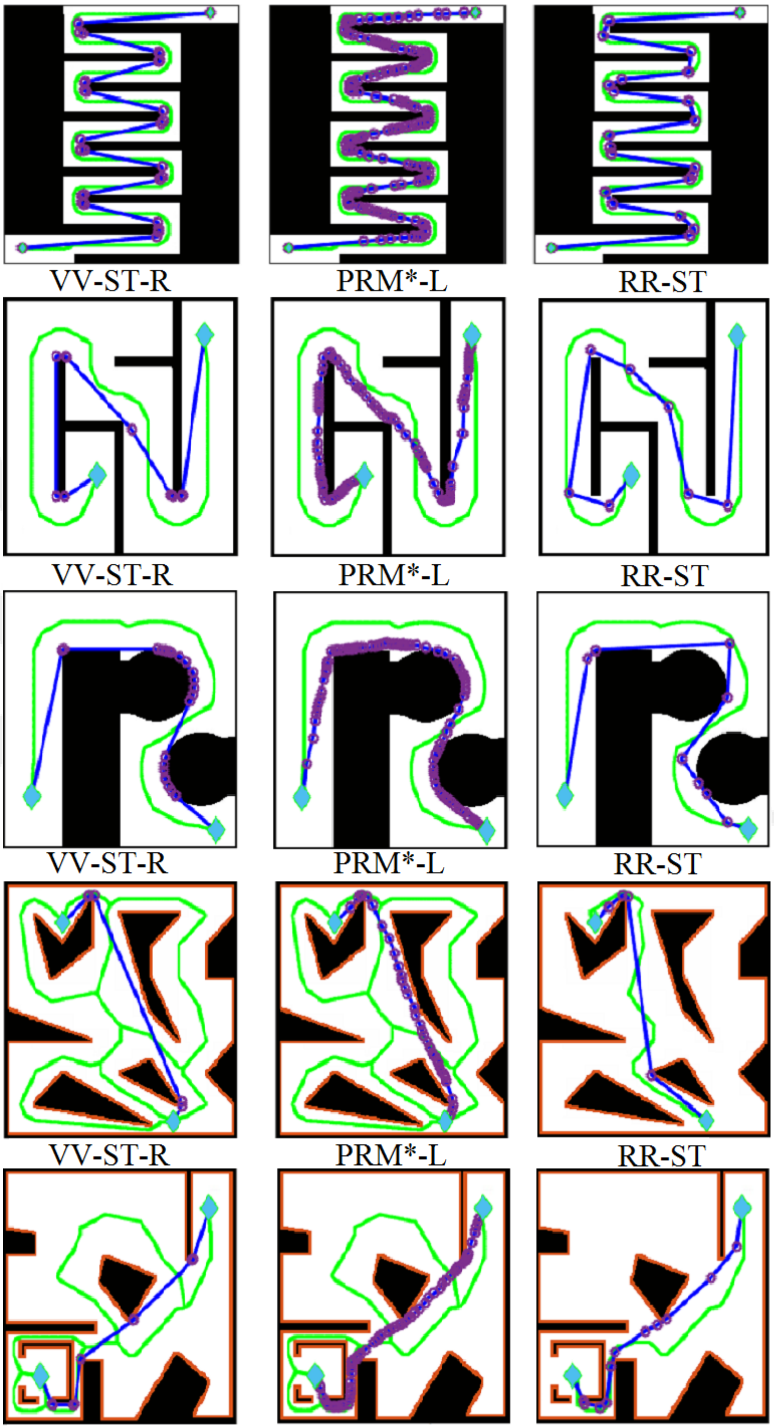
Table 5.3: Computation time comparison

Map	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15
	$D_{\mathrm{S}} = 0$														
VV-ST-R	1.7	1.4	1.7	1.3	1.2	0.49	1.1	1.2	1.5	1.2	1.5	0.9	1.4	1.4	1.5
RR-ST	1.6	1.6	2.2	2.2	1.8	0.89	2.1	2.2	2.5	1.7	1.9	1.9	2.4	2.5	2.1
PRM*	2.1	14	10	3.8	4.3	9.2	4.4	9.8	3.8	1.9	1.9	2.3	2.8	4.3	2.7
RRT*	2	9.1	9	12	1.2	3.8	4.4	1.3	3.4	1.6	1.8	1.6	4	3.5	5.5
FMT	4.1	39	28	14	7.6	4.4	5.1	2.9	5.6	3.3	9.1	3.9	9	13.2	10
CRT	13	12	8.7	12	2.4	4.8	10	2.3	14	7.7	10.5	4.3	10	15	9.1
	$D_{\mathrm{S}} = 5$														
VV-ST-R	1.7	1.4	1.6	1.3	1.2	0.49	1.1	1.2	1.5	1.3	1.6	0.9	1.5	1.6	1.6
RR-ST	2.1	2	1.9	2	1.6	0.9	1.9	1.7	3.6	2.2	2.2	1.4	2.5	1.8	2.1
PRM*	16	16	1.6	8.1	5.5	8.2	87	12	1.5	2	1.9	32.5	1.9	1.6	8.6
RRT*	3.4	3.1	1.5	7.5	3	5.8	13	3	1.3	1.3	1.3	4.5	1.5	1.4	4.9
FMT	54	70	3.1	17	12	16	29	28	3.7	3.4	5.4	20	6.6	4.3	16
CRT	5.6	5.3	8.4	15	12	7.2	12	4.5	14	7.6	9.8	5.2	10	15	9.3

We further highlight several interesting observations from the solutions obtained using the VV-ST-R algorithm for the maps Z9, Z10, Z14 and Z15. A comparison of the solution paths for VV-ST-R, PRM*-L and RR-ST is shown in Fig. 5.24. The upper plot shows Z6 for $D_\mathrm{S} = 0$. Here, it can be seen that VV-ST-R is able to generate solution paths that turn around sharp corners, while finding short connections in free space. In contrast, RR-ST cannot reduce the path length in case the generated waypoints become too close to each other. This observation is also supported by the second plot for Z9 and $D_\mathrm{S} = 0$. This plot further illustrates that the solution paths of VV-ST-R can tightly follow straight walls. This is not the case for the solution paths of the PRM*s algorithm, which depend on the locations of the randomly sampled nodes. The second plot from bottom shows solution paths for Z10 and $D_\mathrm{S} = 0$. It can be readily observed that solution paths generated by VV-ST-R can follow curved objects by placing an increased number of waypoints where necessary. Finally, the lower plot with Z14 and $D_\mathrm{S} = 5$ illustrates the advantage of computing solution paths for multiple start-goal walks along the VB. Here, VV-ST-R chooses a different start-goal walks and is hence able to generate a solution path that is considerably shorter than the paths for PRM*-L and RR-ST. For completeness, the solution paths of VV-ST-R for the remaining maps and $D_\mathrm{S} = 0$ are shown in Fig. 5.25

We next discuss the ability of the VV-ST-R algorithm to compute safe solution paths. To this end, we consider the safety distance D_CRT obtained by applying the CRT algorithm for each of the maps and select $D_\mathrm{S} = D_\mathrm{CRT} + 1$ as the safety distance for the other methods. We note that this choice of D_S is made in order to compare the idea of inflating the obstacle region with a method that generates safe paths without inflating the obstacle region. In principle, any choice of D_S would be possible as long as there is a connection between p_s and p_g along the VB. In addition to path length and computation time, we also investigate the minimum distance to the obstacle region as shown in Table 5.4.

Looking at the results for the path length, we first note that the methods based on the inflated obstacle region are able to find a shorter solution path than the CRT algorithm, which is specifically targeted for safety. Most importantly, VV-ST-R always finds the shortest safe solution paths among all the methods with a practical computation time. In the most of the cases, the solution paths of VV-ST-R are determined with the shortest computation time and are as well shorter than the ones of PRM*-L.



VV-ST-R PRM*-L RR-ST

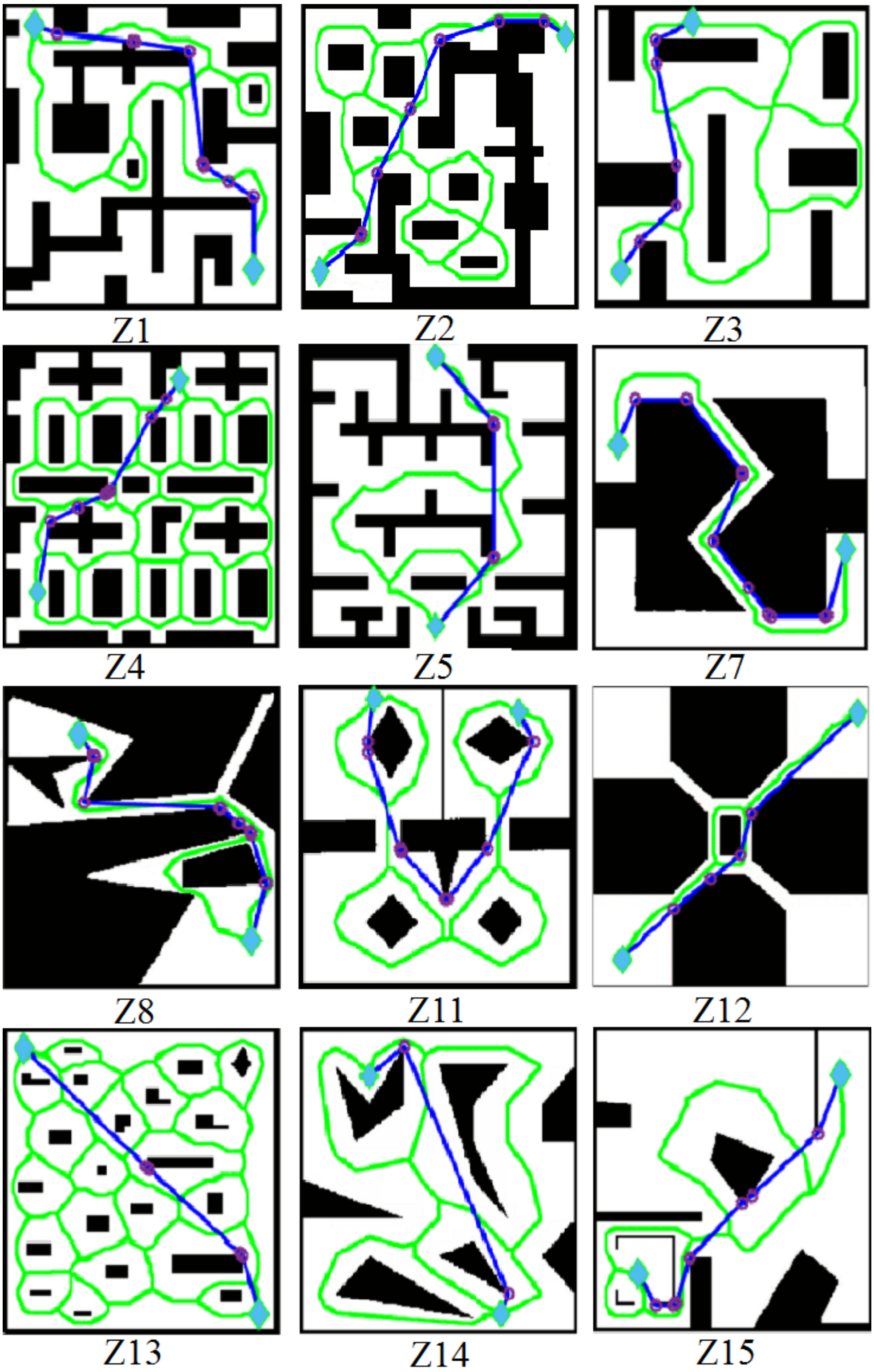
VV-ST-R PRM*-L RR-ST

VV-ST-R PRM*-L RR-ST

VV-ST-R PRM*-L RR-ST

VV-ST-R PRM*-L RR-ST

Figure 5.24: Several interesting cases of solution paths for VV-ST-R, PRM*-L and RR-ST.



Z1 Z2 Z3

Z4 Z5 Z7

Z8 Z11 Z12

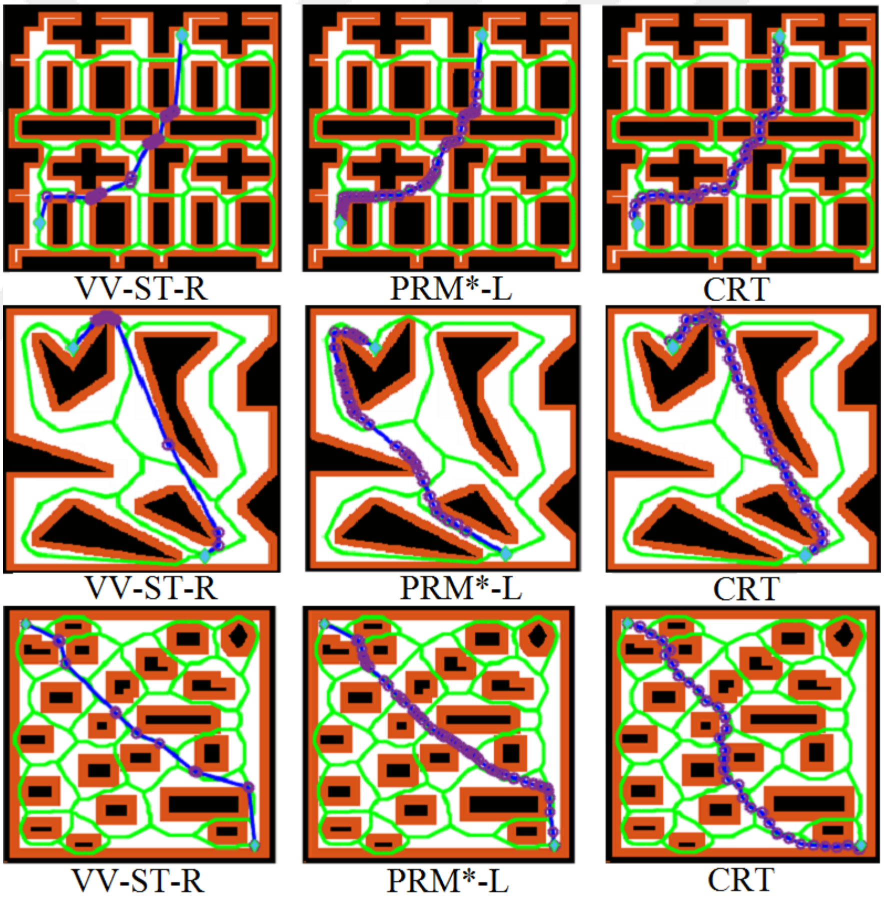
Z13 Z14 Z15

Figure 5.25: Solution paths of VV-ST-R for different maps.

Table 5.4: Performance evaluation for safe paths with $D_{\text{S}} = D_{\text{CRT}} + 1$

Map	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15
	Path length														
VV-ST-R	798	680	604	543	534	1111	1034	705	1371	1027	727	470	561	632	594
PRM*-Long	802	691	607	552	543	1106	1042	701	1384	1023	732	471	563	666	602
RR-ST	817	720	628	559	535	1116	1079	708	1435	1069	795	472	566	658	595
PRM*	819	698	618	586	681	1124	1050	707	1478	1033	743	473	575	690	610
RRT*	1176	876	903	706	826	1277	1320	830	2046	1316	1186	479	717	957	735
FMT	886	698	636	577	621	1123	1060	710	1438	1025	755	475	572	684	604
CRT	883	797	654	608	589	1208	1163	774	1485	1131	807	508	614	685	656
	Computation time														
VV-ST-R	1.7	.97	1.7	1.7	1.3	0.49	1.6	2.4	2.1	2.2	1.6	0.95	2	1.8	1.7
RR-ST	2.3	2.9	2	2.1	2.1	0.94	2.5	1.7	2.8	2.4	1.9	1.4	2	2.1	2.2
PRM*	8.3	16	9.6	23	12	8.2	150	62	4.2	36	9.4	32	13	8.9	45
RRT*	2.3	3.1	2.7	3.3	3.7	5.8	19	5.3	3.4	3.5	2.4	4.5	4.2	3.7	5.7
FMT	25	70.4	27	46	51	16	53	140.5	5.6	26	28	19.9	43	14	41
CRT	4.1	5.3	4.3	5.7	12	7.2	11	9.3	13	7.7	3.7	5.2	10	15	9.1
	Minimum distance to obstacle region														
VV-ST-R	14	6	16	11.7	12	6	11	8.5	16	15.9	14	6.4	16.8	13	10
PRM*-Long	14	6	16.1	12	13	6	11	9	16	16	14.5	6	18	15	10.5
RR-ST	14	6.1	17	11.2	12	6	12	8.5	19.1	18.2	15	6.1	16.9	13	10.2
PRM*	14.2	6.1	17.4	12	12.2	6	11.3	9.1	17.9	15.8	14.6	6.5	17.1	17.5	9.4
RRT*	14	6	16.9	12	12.1	6	11.8	9	17.3	16.5	14.4	6.4	15.8	17.4	9
FMT	14.7	6.3	17.5	12.1	12.2	6	11.8	9	17.8	15.6	14.8	6.6	17.2	17.2	9.6
CRT	10.8	5.4	13.4	9.2	9.8	4.6	10.2	7.2	13	13.7	11.4	5.7	12.9	12.2	8.6

We finally illustrate the findings of this experiment by comparing the solution paths of VV-ST-R, PRM*-L and CRT for the maps Z4, Z13 and Z14 in Fig. 5.26. Here, it can be first noted that both PRM*-L and CRT require a large number of waypoints, leading to large computation times. The upper plot shows that, even all the methods follow the same start-goal walk, the shortest path is computed by VV-ST-R due to the effective placement of a small number of waypoints. In contrast, the solution paths of CRT show unnecessary turns since this algorithm tries to find a path with the highest confidence of avoiding obstacles but without knowledge about the VB. This is true both in cases where CRT follows the correct start-goal walk (upper and center plot) and in cases where CRT follows different start-goal walk (bottom plot). Furthermore, the safety distance achieve by CRT is a result of applying the algorithm and cannot be adjusted as for VV-ST-R by setting D_S.



VV-ST-R PRM*-L CRT

VV-ST-R PRM*-L CRT

VV-ST-R PRM*-L CRT

Figure 5.26: Comparison of safe solution paths.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The subject of this thesis is the path planning problem for mobile robots in a two-dimensional configuration space with obstacles. That is, the presented work specifically addresses the case of omni-directional robots that can perform turning maneuvers on the spot. When solving the described path planning problem, different performance metrics have to be taken into account. First, it is desired to compute robot paths that are as short as possible. Second, it is beneficial for safety if solution paths have a sufficient obstacle clearance. Finally, it is very important for real-time applications to generate suitable paths with a short computation time. The thesis addresses the stated requirements by developing three new approaches for robotic path planning. The first approach focuses on the pre-processing of environment maps that can then be used for any sampling-based path planning method. Here, the main aim is to increase the obstacle region in order to generate samples only in the part of the map that is promising for safe solution paths. It is hence expected that safe solution paths can be computed in a shorter time. The second approach consists of three new path planning algorithms that are extensions of the PRM (probabilistic roadmap) algorithm, PRM* algorithm and FMT (fast marching tree) algorithm. The underlying idea for defining the new algorithms is to first compute a generalized Voronoi diagram (GVD) of the robot environment. The Voronoi boundary of this GVD is then inflated by a certain width. Each of the stated algorithms (PRM, PRM* and FMT) is adapted such that node samples are only generated on the inflated Voronoi boundary and node connections lie fully within the inflated Voronoi boundary. The resulting algorithms are denoted as IPRM (Inflated PRM), IPRM* (Inflated PRM*) and IFMT (Inflated FMT). As a particular feature, the proposed algorithms require fewer nodes when determining a solution path and ensure a minimum distance to obstacles by appropriately choosing the width of the inflated Voronoi boundary. Different from the first two approaches, which employ sampling-based strategies, the third approach uses information about the topology of the environment from the GVD. Specifically, initial solution paths that follow Voronoi edges are iteratively refined by introduce shortcuts and by adding new waypoints to remove corners in the path.

Our comprehensive evaluation of the first approach shows that safe robot paths can be obtained with small computation times for all of the proposed algorithms. Hereby, the algorithms which use PRM-based sampling provide the best results. Specifically, the obtained solution paths are better than the paths produced by algorithms such as CRT (confidence random tree), MAPRM (medial axis PRM) and MARRT (medial axis RRT) which are designed for path safety. As a particular feature of the proposed algorithms, they provide formal guarantees regarding path safety. Our second approach is evaluated by computational experiments with different environments. In these experiments, it was confirmed that the proposed methods IPRM, IPRM* and IFMT outperform the existing methods PRM, PRM* and FMT regarding computation time and path safety at a slight increase of the path length. Moreover, a comparison with the recent CRT algorithm that specifically addresses path safety was performed. This comparison indicates that the proposed algorithms are significantly faster, generate shorter paths and lead to a comparable path safety. Furthermore, large variations of these performance metrics that are observed for the CRT algorithm can be avoided for the proposed algorithms since solution paths are confined to the inflated Voronoi boundary. As an important result regarding the second approach, we conclude that it is preferable to apply proven algorithms such as PRM, PRM* or FMT on pre-processed environment maps instead of designing specific algorithms such as CRT for the original environment map. The third approach constitutes the most promising algorithm developed in this thesis. Here, we emphasize that this algorithm is guaranteed to find a solution path if such path exists. In addition, our evaluation show that the algorithm is able to produce shorter paths in a shorter time compared to other methods using the GVD. The main reason for this result is the iterative removal and addition of waypoints and the consideration of multiple start-goal connections. It can further be concluded that the third approach successfully addresses the properties of low-dimensional spaces.

There are several directions for future work. Regarding the first approach, it is the case that the pre-processing is done based on global properties such as the minimum distance of the Voronoi edges from obstacles. As an improvement, it is possible to use local properties of environment maps for pre-processing. A similar improvement can be employed for the second approach. Regarding the third approach, it has to be stated that the formulation is for static environments. That is, an extension to environments with dynamic obstacles would be beneficial. In addition, it would be interesting to apply smoothing in order to make the generated path usable for non-holonomic robots.

REFERENCES

[1] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, 2019.

[2] S. Spanogianopoulos and K. Sirlantzis, *Car-Like Mobile Robot Navigation: A Survey*, pp. 299–327. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.

[3] G. R. Bermüdez-Bohórquez, C. A. Mancipe-Bernal, and J. E. Ortiz-Velásquez, "Control of a robotic convoy through path planning and orientation strategies," *Revista cientifica*, pp. 237–251, 12 2017.

[4] S. G. Tzafestas, "Mobile robot control and navigation: A global overview," *Journal of Intelligent & Robotic Systems*, vol. 91, pp. 35–58, 2018.

[5] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, p. 463–497, 2015.

[6] I. Noreen, A. Khan, H. Ryu, N. L. Doh, and Z. Habib, "Optimal path planning in cluttered environment using rrt*-ab," *Intell. Serv. Robot.*, vol. 11, p. 41–52, Jan. 2018.

[7] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh, *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, 2019.

[8] L. G. D. O. Véras, F. L. L. Medeiros, and L. N. F. Guimaráes, "Systematic literature review of sampling process in rapidly-exploring random trees," *IEEE Access*, vol. 7, pp. 50933–50953, 2019.

[9] D. G. Macharet and M. F. M. Campos, "A survey on routing problems and robotic systems," *Robotica*, vol. 36, no. 12, p. 1781–1803, 2018.

[10] Y. Dong, E. Camci, and E. Kayacan, "Faster rrt-based nonholonomic path planning in 2d building environments using skeleton-constrained path biasing," *J. Intell. Robotics Syst.*, vol. 89, p. 387–401, Mar. 2018.

[11] S. Agnisarman, S. Lopes, K. C. Madathil, K. Piratla, and A. Gramopadhye, "A survey of automation-enabled human-in-the-loop systems for infrastructure visual inspection," *Automation in Construction*, vol. 97, pp. 52 – 76, 2019.

[12] R. Cui, Y. Li, and W. Yan, "Mutual information-based multi-auv path planning for scalar field sampling using multidimensional rrt*," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, pp. 993–1004, July 2016.

[13] P. Pandey, A. Shukla, and R. Tiwari, "Aerial path planning using meta-heuristics: A survey," in *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1–7, Feb 2017.

[14] G. Jain, G. Yadav, D. Prakash, A. Shukla, and R. Tiwari, "Mvo-based path planning scheme with coordination of uavs in 3-d environment," *Journal of Computational Science*, vol. 37, p. 101016, 2019.

[15] T. T. Mac, C. Copot, T. T. Duc, and R. D. Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[16] L. Kavraki, P. Svestka, J. claude Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, pp. 566–580, 1996.

[17] D. Scaramuzza, R. Siegwart, and A. Martinelli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 149–171, 2009.

[18] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[19] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.

[20] Y. N. Kim, D. W. Ko, and I. H. Suh, "Confidence random tree-based algorithm for mobile robot path planning considering the path length and safety," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419838179, 2019.

[21] L. Gang and J. Wang, "Prm path planning optimization algorithm research," *Wseas Transactions on Systems and control*, vol. 11, pp. 81–86, 2016.

[22] I.-B. Jeong, S.-J. Lee, and J.-H. Kim, "Quick-rrt*: Triangular inequality-based implementation of rrt* with improved initial solution and convergence rate," *Expert Systems with Applications*, vol. 123, pp. 82–90, 2019.

[23] T. Bai, Z. Fan, M. Liu, S. Zhang, and R. Zheng, "Multiple waypoints path planning for a home mobile robot," in *2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 53–58, IEEE, 2018.

[24] K. Yang, "Anytime synchronized-biased-greedy rapidly-exploring random tree path planning in two dimensional complex environments," *International Journal of Control, Automation and Systems*, vol. 9, no. 4, p. 750, 2011.

[25] D. A. L. García and F. Gomez-Bravo, "Vodec: A fast voronoi algorithm for car-like robot path planning in dynamic scenarios," *Robotica*, vol. 30, no. 7, pp. 1189–1201, 2012.

[26] D. Connell and H. M. La, "Dynamic path planning and replanning for mobile robots using RRT," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1429–1434, IEEE, 2017.

[27] N. Piccinelli, F. Vesentini, and R. Muradore, "Planning with real-time collision avoidance for cooperating agents under rigid body constraints," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1261–1264, IEEE, 2019.

[28] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171 – 185, 2018.

[29] B. B. K. Ayawli, X. Mei, M. Shen, A. Y. Appiah, and F. Kyeremeh, "Mobile robot path planning in dynamic environment using voronoi diagram and computation geometry technique," *IEEE Access*, vol. 7, pp. 86026–86040, 2019.

[30] P. Bhattacharya and M. L. Gavrilova, "Roadmap-based path planning - using the voronoi diagram for a clearance-based shortest path," *IEEE Robotics & Automation Magazine*, vol. 15, 2008.

[31] P. Loncomilla, J. R. del Solar, and L. Martínez, "Object recognition using local invariant features for robotic applications: A survey," *Pattern Recognition*, vol. 60, pp. 499 – 514, 2016.

[32] F. M. S., E. J. G., and H. M. A., "Navigable points estimation for mobile robots using binary image skeletonization," in *Eighth International Conference on Graphic and Image Processing (ICGIP 2016)* (Y. Wang, T. D. Pham, V. Vozenilek, D. Zhang, and Y. Xie, eds.), vol. 10225, pp. 19 – 23, International Society for Optics and Photonics, SPIE, 2017.

[33] R. Samaniego, J. Lopez, and F. Vazquez, "Path planning for non-circular, non-holonomic robots in highly cluttered environments," *Sensors*, vol. 17, no. 8, 2017.

[34] B. Ayawli, X. Mei, M. Shen, A. Y. Appiah, and F. Kyeremeh, "Optimized rrt-a* path planning method for mobile robots in partially known environment," *Information technology and control*, vol. 48, no. 2, pp. 179–194, 2019.

[35] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 159–185, 2018.

[36] M. M. Costa and M. F. Silva, "A survey on path planning algorithms for mobile robots," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 1–7, April 2019.

[37] A. Khan, I. Noreen, and Z. Habib, "On complete coverage path planning algorithms for non-holonomic mobile robots: Survey and challenges," *J. Inf. Sci. Eng.*, vol. 33, pp. 101–121, 2017.

[38] A. S. H. H. V. Injarapu and S. K. Gawre, “A survey of autonomous mobile robot path planning approaches,” in *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*, pp. 624–628, Oct 2017.

[39] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer Publishing Company, Incorporated, 1st ed., 2013.

[40] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 5 2005.

[41] L. Kavraki and J. . Latombe, “Randomized preprocessing of configuration for fast path planning,” in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 2138–2145 vol.3, May 1994.

[42] M. R. H. Al-Dahhan and M. M. Ali, “Path tracking control of a mobile robot using fuzzy logic,” in *2016 13th International Multi-Conference on Systems, Signals & Devices (SSD)*, pp. 82–88, IEEE, 2016.

[43] E. Magid, R. Lavrenov, M. Svinin, and A. Khasianov, “Combining voronoi graph and spline-based approaches for a mobile robot path planning,” in *International Conference on Informatics in Control, Automation and Robotics*, pp. 475–496, Springer, 2017.

[44] M. Korkmaz and A. Durdu, “Comparison of optimal path planning algorithms,” in *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, pp. 255–258, IEEE, 2018.

[45] P. Sudhakara, V. Ganapathy, and K. Sundaran, “Probabilistic roadmaps-spline based trajectory planning for wheeled mobile robot,” in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 3579–3583, IEEE, 2017.

[46] J. Wang, W. Chi, M. Shao, and M. Q.-H. Meng, “Finding a high-quality initial solution for the rrts algorithms in 2d environments,” *Robotica*, vol. 37, no. 10, pp. 1677–1694, 2019.

[47] J. Kim and S. H. A. Woo, “Reference test maps for path planning algorithm test,” *International Journal of Control, Automation and Systems*, vol. 16, no. 1, pp. 397–401, 2018.

[48] I.-S. Kim, W.-K. Lee, and Y.-D. Hong, “Simple global path planning algorithm using a ray-casting and tracking method,” *Journal of Intelligent & Robotic Systems*, vol. 90, no. 1-2, pp. 101–111, 2018.

[49] R. M. C. Santiago, A. L. De Ocampo, A. T. Ubando, A. A. Bandala, and E. P. Dadios, “Path planning for mobile robots using genetic algorithm and probabilistic roadmap,” in *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pp. 1–5, IEEE, 2017.

[50] H. Dong, W. Li, J. Zhu, and S. Duan, “The path planning for mobile robot based on voronoi diagram,” in *2010 Third International Conference on Intelligent Networks and Intelligent Systems*, pp. 446–449, IEEE, 2010.

[51] M. Foskey, M. Garber, M. C. Lin, and D. Manocha, “A voronoi-based hybrid motion planner,” in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1, pp. 55–60, IEEE, 2001.

[52] E. Masehian and M. Amin-Naseri, “A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning,” *Journal of Robotic Systems*, vol. 21, no. 6, pp. 275–300, 2004.

[53] Q. Wang, M. Wulfmeier, and B. Wagner, “Voronoi-based heuristic for nonholonomic search-based path planning,” in *Intelligent Autonomous Systems 13*, pp. 445–458, Springer, 2016.

[54] E. W. Dijkstra *et al.*, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[55] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, “Path planning for mobile robot navigation using voronoi diagram and fast marching,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2376–2381, IEEE, 2006.

[56] N. Alpkiray, Y. Torun, and O. KAYNAR, “Probabilistic roadmap and artificial bee colony algorithm cooperation for path planning,” in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, pp. 1–6, IEEE, 2018.

[57] H. Mo and L. Xu, “Research of biogeography particle swarm optimization for robot path planning,” *Neurocomputing*, vol. 148, pp. 91–99, 2015.

[58] G. C. de Oliveira, K. B. de Carvalho, and A. S. Brandão, “A hybrid strategy for robot navigation in semi-structured environments,” in *2018 IEEE International Conference on Industrial Technology (ICIT)*, pp. 23–28, IEEE, 2018.

[59] Q. Wang, M. Langerwisch, and B. Wagner, “Wide range global path planning for a large number of networked mobile robots based on generalized voronoi diagrams,” *IFAC Proceedings Volumes*, vol. 46, no. 29, pp. 107–112, 2013.

[60] M. A. A. Hemmat, Z. Liu, and Y. Zhang, “Real-time path planning and following for nonholonomic unmanned ground vehicles,” in *2017 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pp. 202–207, IEEE, 2017.

[61] R. Acharya and D. Jena, “Gradient descent in sample-based single-query path planning algorithm,” in *2018 IEEMA Engineer Infinite Conference (eTechNxT)*, pp. 1–6, IEEE, 2018.

[62] B. B. K. Ayawli, X. Mei, M. Shen, A. Y. Appiah, and F. Kyeremeh, “Optimized rrt-a* path planning method for mobile robots in partially known environment,” *Information Technology and Control*, vol. 48, no. 2, pp. 179–194, 2019.

[63] W. Gong, X. Xie, and Y.-J. Liu, “Human experience–inspired path planning for robots,” *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, p. 1729881418757046, 2018.

[64] E. Magid, R. Lavrenov, and I. Afanasyev, “Voronoi-based trajectory optimization for ugv path planning,” in *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*, pp. 383–387, IEEE, 2017.

[65] R. Kala, “Homotopic roadmap generation for robot motion planning,” *Journal of Intelligent & Robotic Systems*, vol. 82, no. 3-4, pp. 555–575, 2016.

[66] F. Carreira, J. Calado, C. Cardeira, and P. Oliveira, “Navigation system for mobile robots using pca-based localization from ceiling depth images: Experimental validation,” in *2018 13th APCA International Conference on Automatic Control and Soft Computing (CONTROLO)*, pp. 159–164, IEEE, 2018.

[67] D. A. L. García and F. Gomez-Bravo, “Vodec: A fast voronoi algorithm for car-like robot path planning in dynamic scenarios,” *Robotica*, vol. 30, no. 7, pp. 1189–1201, 2012.

[68] Q. Wang, M. Wulfmeier, and B. Wagner, “Voronoi-based heuristic for non-holonomic search-based path planning,” in *Intelligent Autonomous Systems 13*, pp. 445–458, Springer, 2016.

[69] K. Sugihara, “Approximation of generalized voronoi diagrams by ordinary voronoi diagrams,” *CVGIP: Graphical Models and Image Processing*, vol. 55, no. 6, pp. 522–531, 1993.

[70] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver, “Fast computation of generalized voronoi diagrams using graphics hardware,” in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 277–286, 1999.

[71] D. T. Lee, “Medial axis transformation of a planar shape,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, pp. 363–369, July 1982.

[72] M. Couprie, D. Coeurjolly, and R. Zrour, “Discrete bisector function and euclidean skeleton in 2d and 3d,” *Image and Vision Computing*, vol. 25, no. 10, pp. 1543 – 1556, 2007. Discrete Geometry for Computer Imagery 2005.

[73] S. Biasotti, D. Attali, J.-D. Boissonnat, H. Edelsbrunner, G. Elber, M. Mortara, G. S. di Baja, M. Spagnuolo, M. Tanase, and R. Veltkamp, *Skeletal Structures*, pp. 145–183. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[74] R. Ogniewicz and M. Ilg, “Voronoi skeletons: theory and applications,” in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 63–69, June 1992.

[75] R. Tam and W. Heidrich, “Shape simplification based on the medial axis transform,” in *IEEE Visualization, 2003. VIS 2003.*, pp. 481–488, Oct 2003.

[76] L. Mestetskiy and A. Semenov, “Binary image skeleton - continuous approach,” in *International Conference on Computer Vision Theory and Applications*, 2008.

[77] R. Wein, J. P. van den Berg, and D. Halperin, “The visibility–voronoi complex and its applications,” *Computational Geometry*, vol. 36, no. 1, pp. 66–87, 2007. Special Issue on the 21st European Workshop on Computational Geometry.

[78] M. Hasan, M. L. Gavrilova, and J. G. Rokne, “A geometric approach to clearance based path optimization,” in *Computational Science and Its Applications – ICCSA 2007* (O. Gervasi and M. L. Gavrilova, eds.), (Berlin, Heidelberg), pp. 136–150, Springer Berlin Heidelberg, 2007.

[79] D. Ji, J. Cheng, and B. Wang, “Path planning for mobile robots in complex environment via laser sensor,” in *2018 Chinese Control And Decision Conference (CCDC)*, pp. 2715–2719, IEEE, 2018.

[80] M. L. Wager, “Making roadmaps using voronoi diagrams,” 2000.

[81] A. D. D. H. K. E. B. Rahul Shome, Kiril Solovey, “Scalable and informed asymptotically-optimal multi-robot motion planning,” *Autonomous Robots*, 2019.

[82] C. Martínez Alandes, “A comparison among different sampling-based planning techniques,” 2015.

[83] L. E. Kavraki, M. N. Kolountzakis, and J. . Latombe, “Analysis of probabilistic roadmaps for path planning,” *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 166–171, Feb 1998.

[84] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, “Maprm: a probabilistic roadmap planner with sampling on the medial axis of the free space,” in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, pp. 1024–1031 vol.2, May 1999.

[85] J. Denny, E. Greco, S. Thomas, and N. M. Amato, “Marrt: Medial axis biased rapidly-exploring random trees,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 90–97, May 2014.

[86] S. Chen, M. Song, and S. Sahni, “Two techniques for fast computation of constrained shortest paths,” *IEEE/ACM Transactions on Networking*, vol. 16, pp. 105–115, Feb 2008.

[87] Rodriguez, Xinyu Tang, Jyh-Ming Lien, and N. M. Amato, “An obstacle-based rapidly-exploring random tree,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pp. 895–900, May 2006.

[88] R. Hess, F. Kempf, and K. Schilling, “Trajectory planning for car-like robots using rapidly exploring random trees*,” *IFAC Proceedings Volumes*, vol. 46, no. 29, pp. 44 – 49, 2013. 3rd IFAC Symposium on Telematics Applications.

[89] J. Y. Yen, “Finding the k shortest loopless paths in a network,” *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[90] D. Hsu, J. . Latcombe, and S. Sorkin, "Placing a robot manipulator amid obstacles for optimized execution," in *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning (ISATP'99) (Cat. No.99TH8470)*, pp. 280–285, July 1999.

[91] P. Bhattacharya and M. L. Gavrilova, "Geometric algorithms for clearance based optimal path computation," in *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, GIS '07, (New York, NY, USA), Association for Computing Machinery, 2007.

[92] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," in *Proceedings of the 4th International Symposium on Voronoi Diagrams in Science and Engineering*, ISVD '07, (USA), p. 38–47, IEEE Computer Society, 2007.

[93] MATLAB, *version 9.6.0.1072779 (R2019a)*. Natick, Massachusetts: The MathWorks Inc., 2019.

Appendices

APPENDIX A

Comparison of the Solution Paths for VV-ST-R.

Several examples for the comparison of the solution paths for the methods VV, RR, VV-ST, RR-ST, VV-ST-R and PRM*-L are given in Section IV-A of the main document. This section provides the complete set of solution paths for the maps VV1, VV2 ,VV3 and VV4 as well as the safety distances $D_S = 0$, $D_S = 5$ and $D_S = 10$.

A.1 $D_S = 0$

Figure A.1 to A.4 show the solution paths for the different maps and $D_S = 0$.

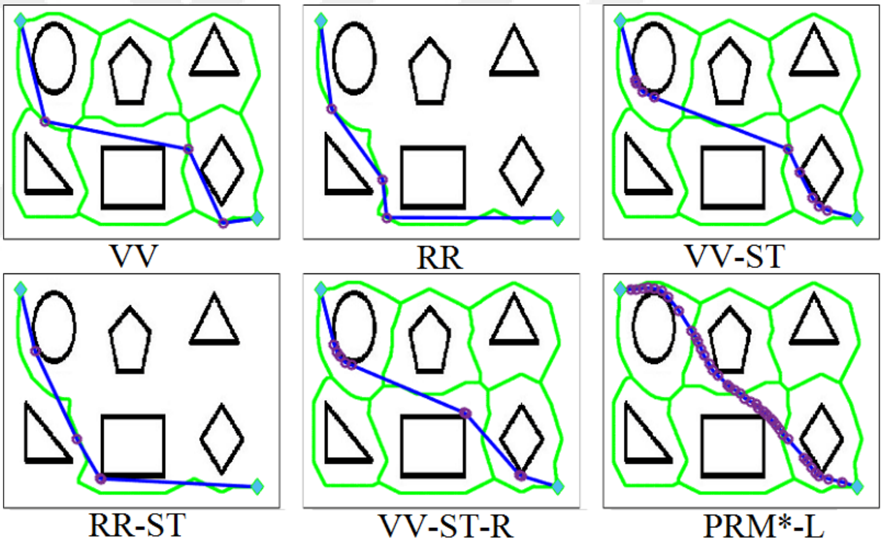
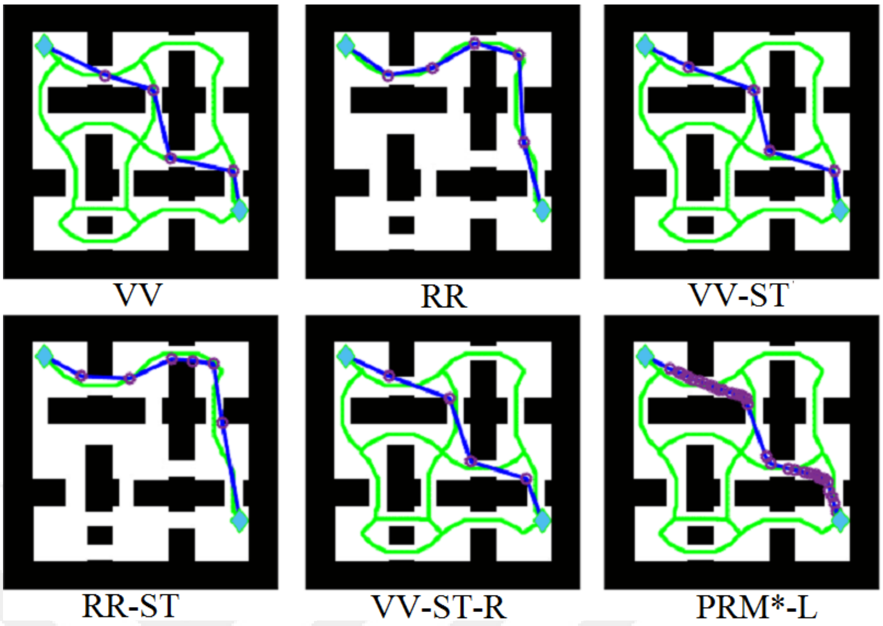


Figure A.1: VV1, $D_S = 0$.

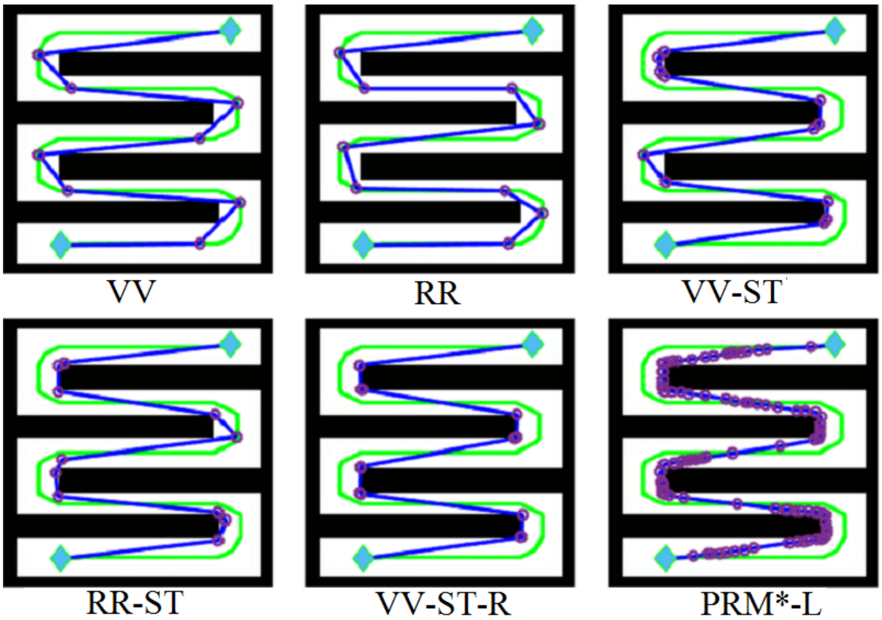
A.2 $D_S = 5$

Figure A.5 to A.8 show the solution paths for the different maps and $D_S = 5$.


VV RR VV-ST

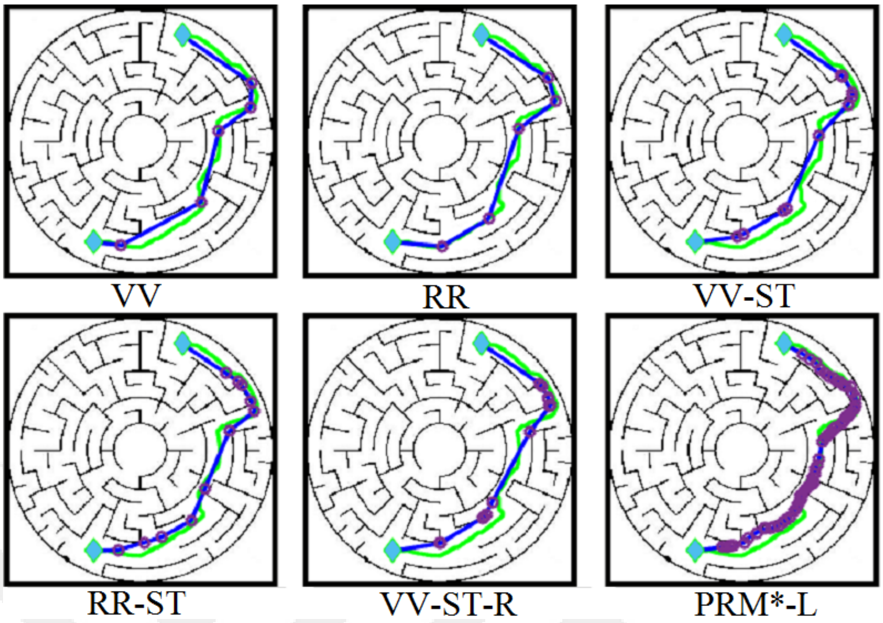
RR-ST VV-ST-R PRM*-L

Figure A.2: VV2, $D_{\mathrm{S}} = 0$.


VV RR VV-ST

RR-ST VV-ST-R PRM*-L

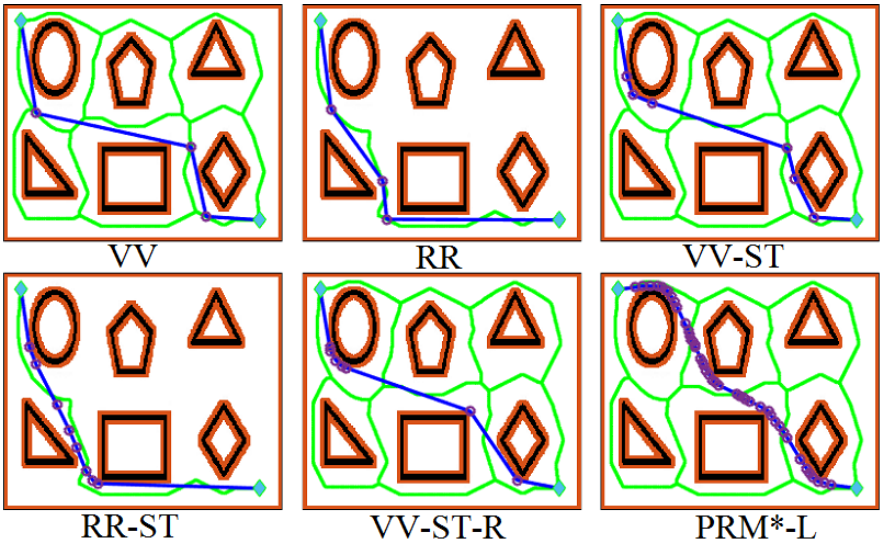
Figure A.3: VV3, $D_{\mathrm{S}} = 0$.



VV RR VV-ST

RR-ST VV-ST-R PRM*-L

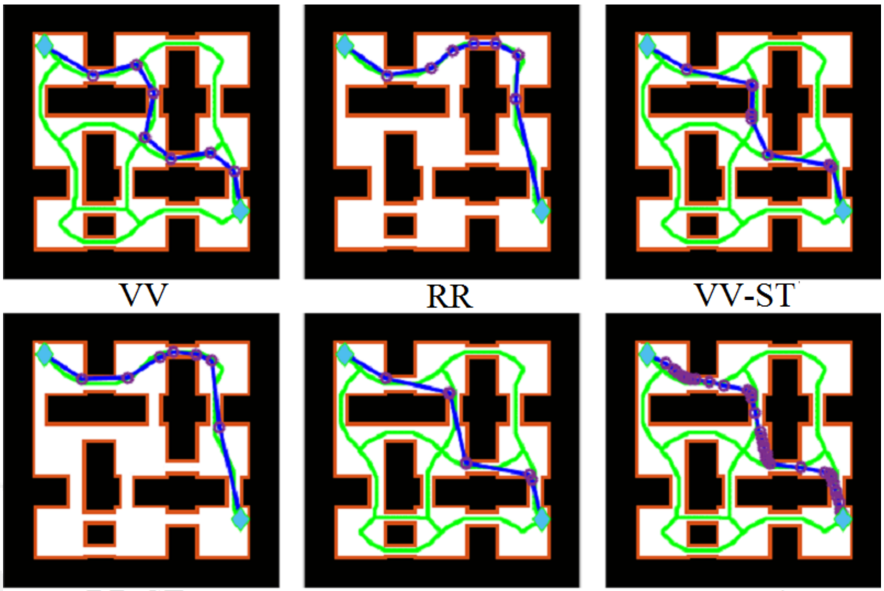
Figure A.4: VV4, $D_S = 0$.



VV RR VV-ST

RR-ST VV-ST-R PRM*-L

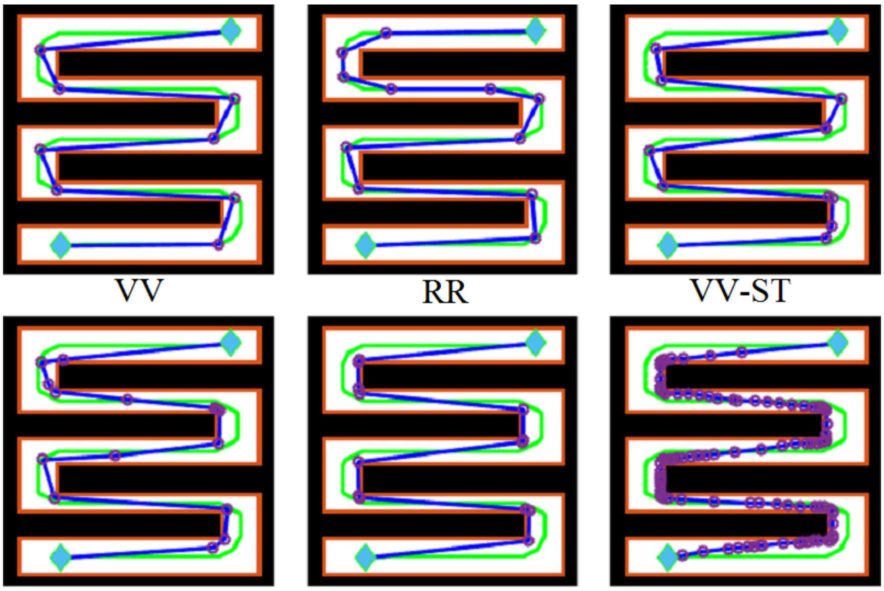
Figure A.5: VV1, $D_S = 5$.



VV RR VV-ST

RR-ST VV-ST-R PRM*-L

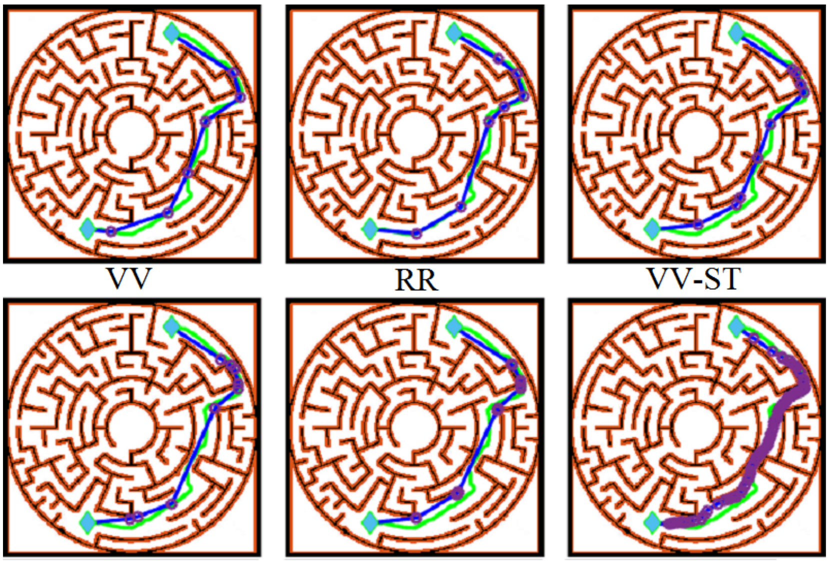
Figure A.6: VV2, $D_S = 5$.



VV RR VV-ST

RR-ST VV-ST-R PRM*-L

Figure A.7: VV3, $D_S = 5$.

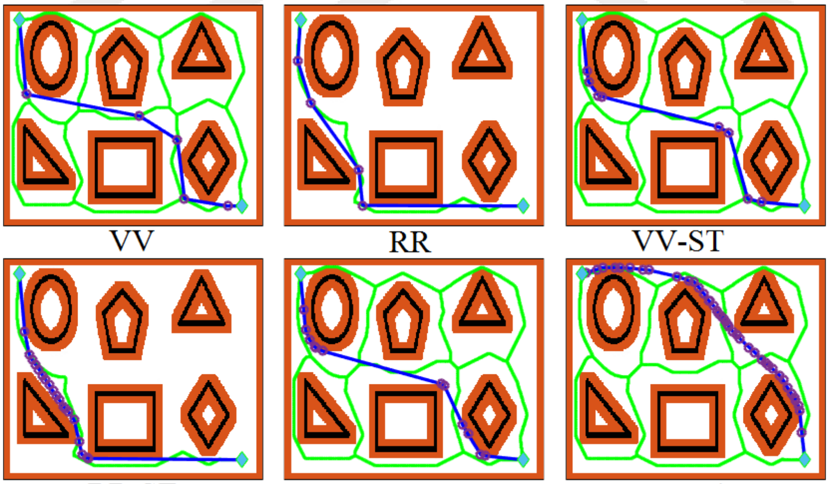

VV RR VV-ST

RR-ST VV-ST-R PRM*-L

Figure A.8: VV4, $D_S = 5$.

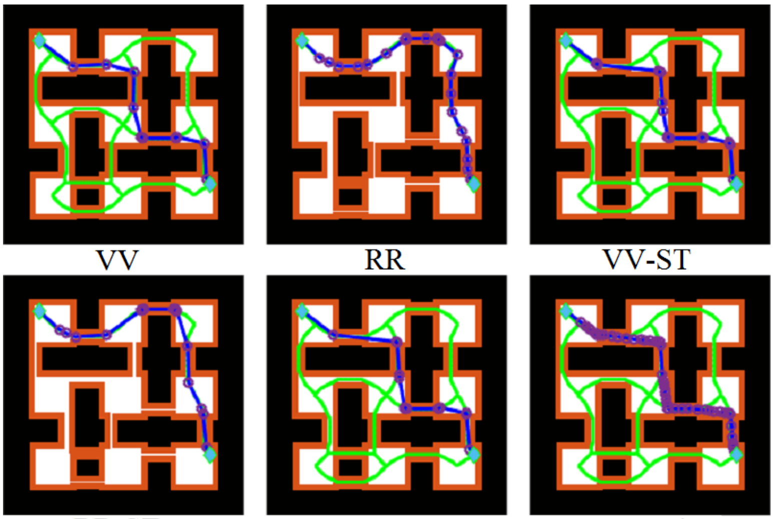
A.3 $D_S = 10$

Figure A.9 to A.12 show the solution paths for the different maps and $D_S = 10$.


VV RR VV-ST

RR-ST VV-ST-R PRM*-L

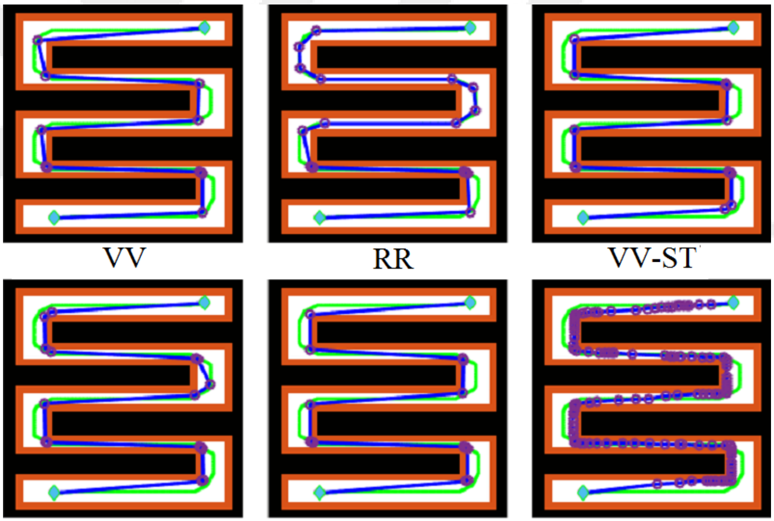
Figure A.9: VV1, $D_S = 10$.



VV RR VV-ST

RR-ST VV-ST-R PRM*-L

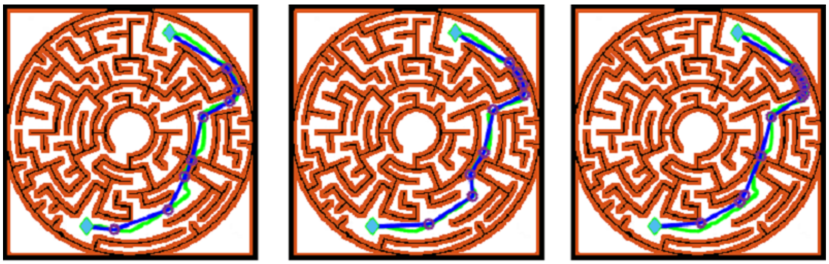
Figure A.10: VV2, $D_S = 10$.



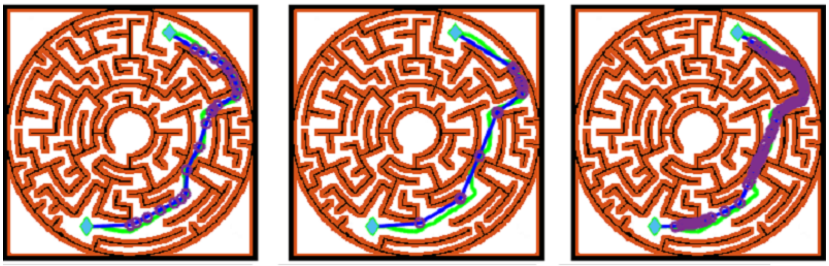
VV RR VV-ST

RR-ST VV-ST-R PRM*-L

Figure A.11: VV3, $D_S = 10$.



VV RR VV-ST



RR-ST VV-ST-R PRM*-L

Figure A.12: VV4, $D_S = 10$.

APPENDIX B

Comparison of the Solution Paths with the State-of-the-Art Methods

This part shows a comparison of the solution paths for the proposed VV-ST-R algorithm and the most relevant state-of-the-art algorithms for the maps Z1 to Z15.

B.1 $D_S = 0$

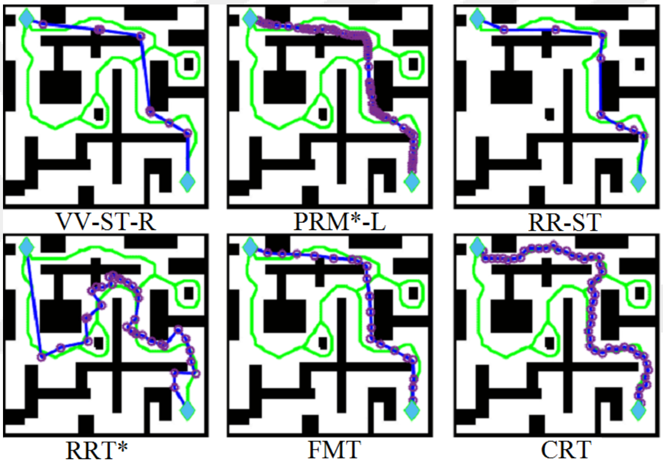


Figure B.1: Z1, $D_S = 0$.

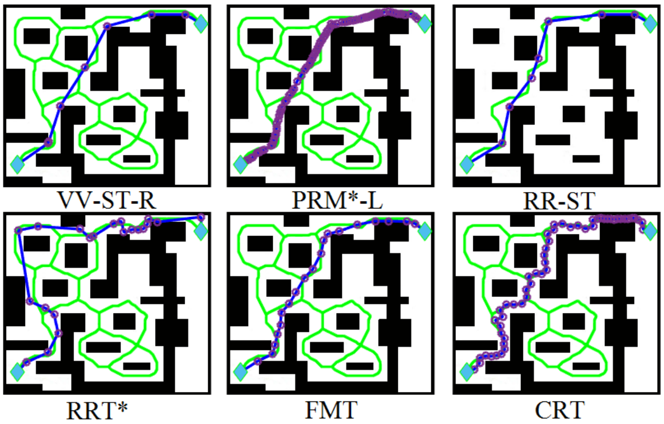
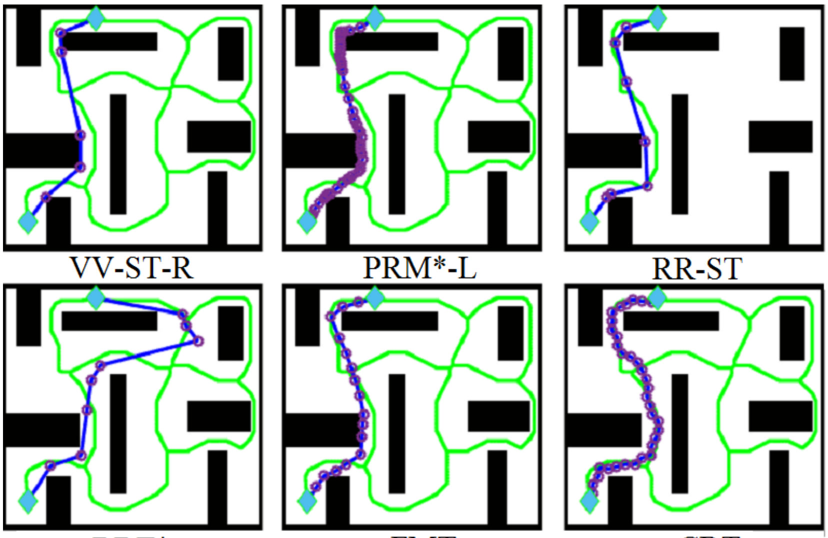
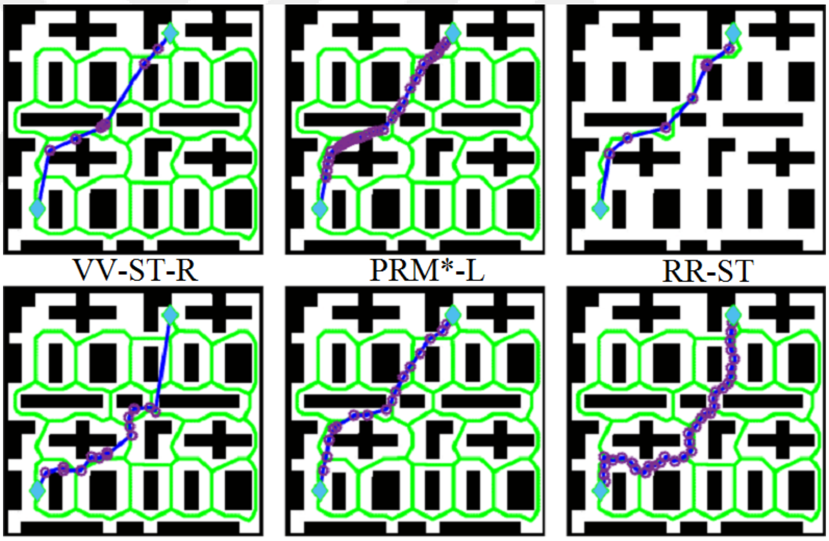


Figure B.2: Z2, $D_S = 0$


VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

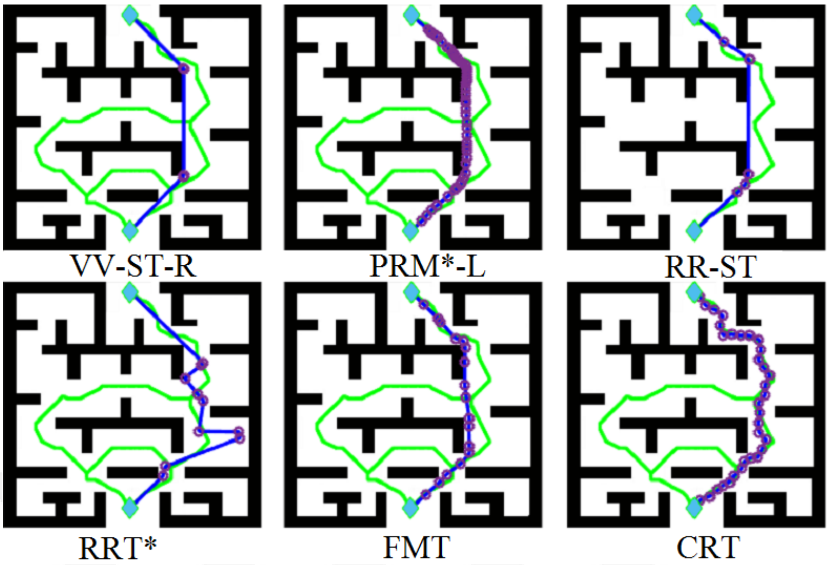
Figure B.3: $Z3$, $D_{\mathrm{S}} = 0$


VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.4: $Z4$, $D_{\mathrm{S}} = 0$

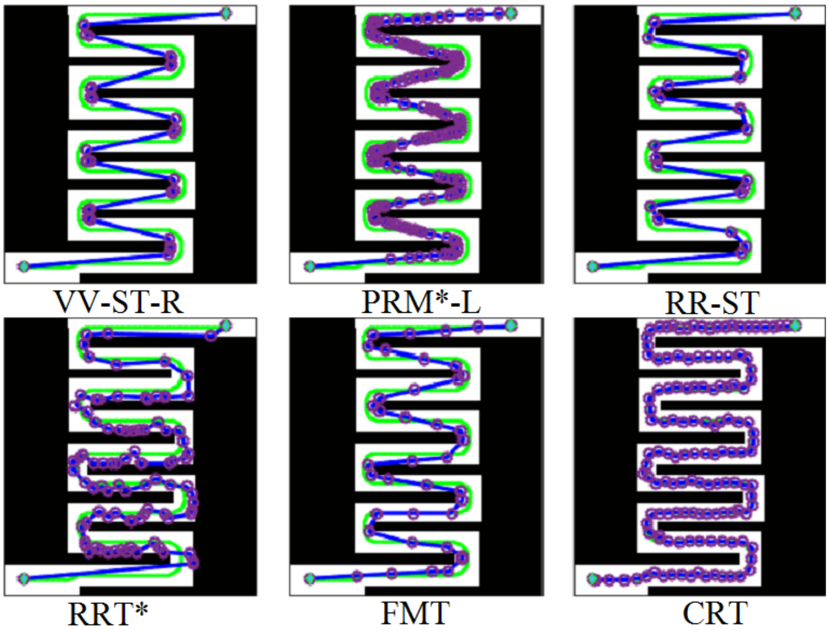
B.2 $D_{\mathrm{S}} = 5$



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

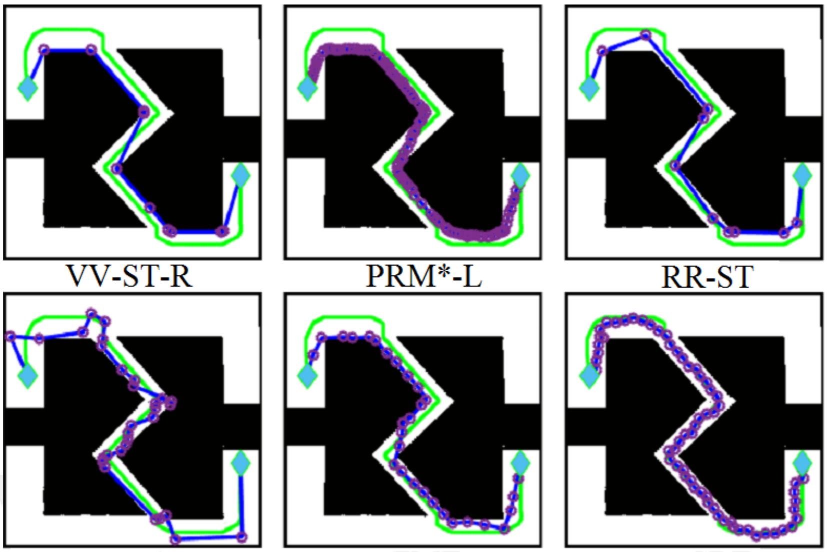
Figure B.5: Z5, $D_S = 0$



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

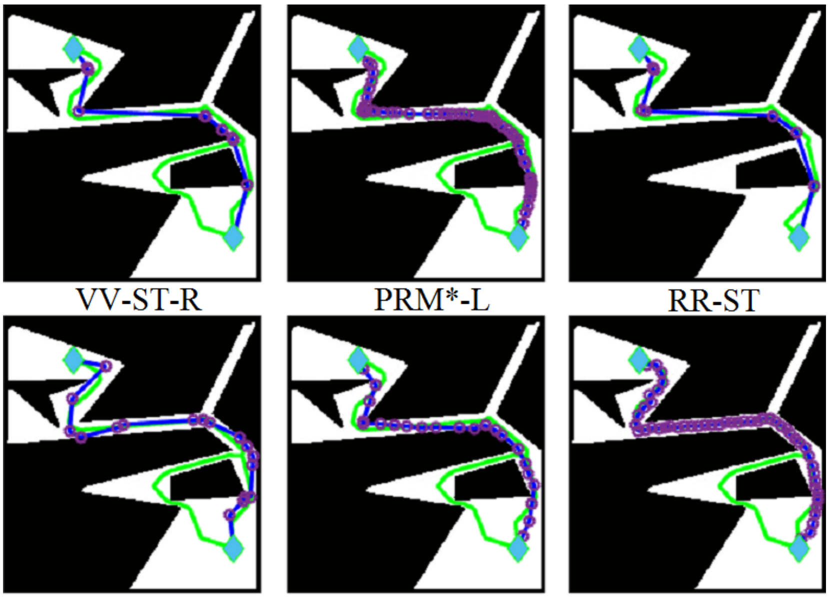
Figure B.6: Z6, $D_S = 0$



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

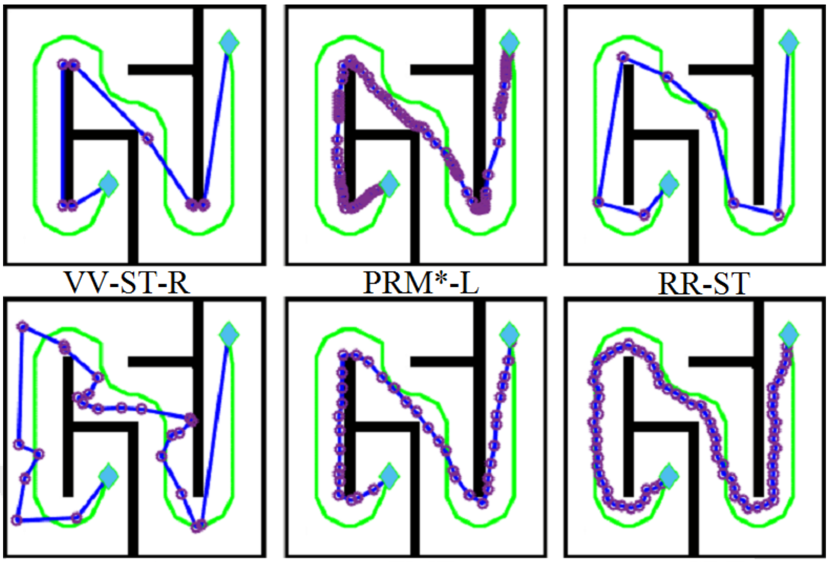
Figure B.7: Z7, $D_S = 0$



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

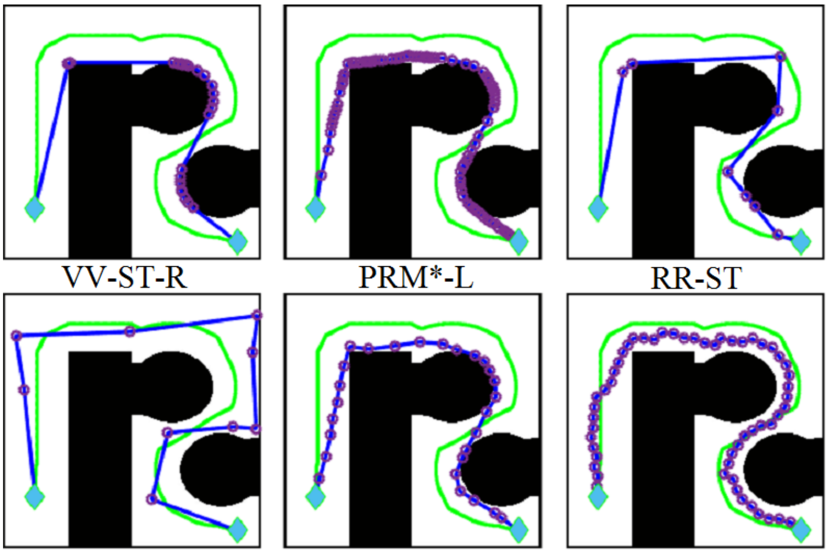
Figure B.8: Z8, $D_S = 0$



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.9: Z9, $D_S = 0$



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.10: Z10, $D_S = 0$

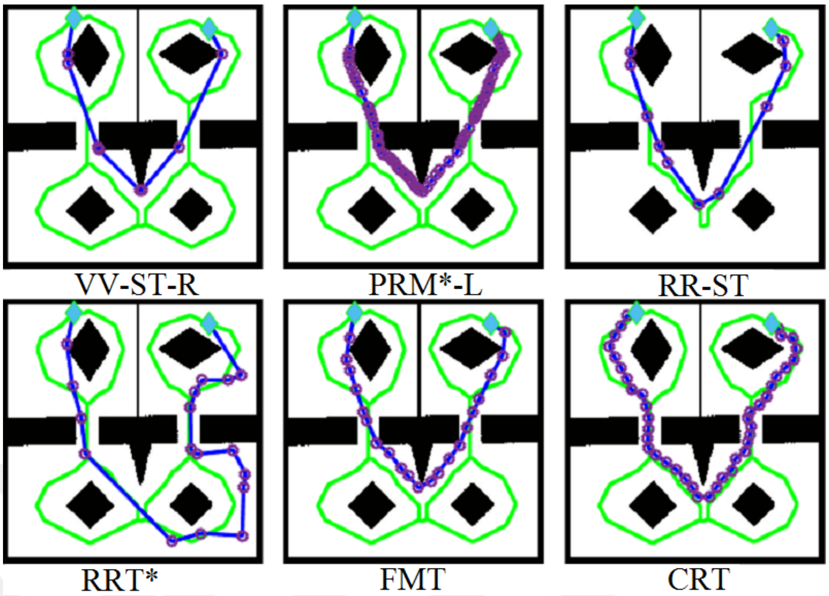


Figure B.11: Z11, $D_S = 0$

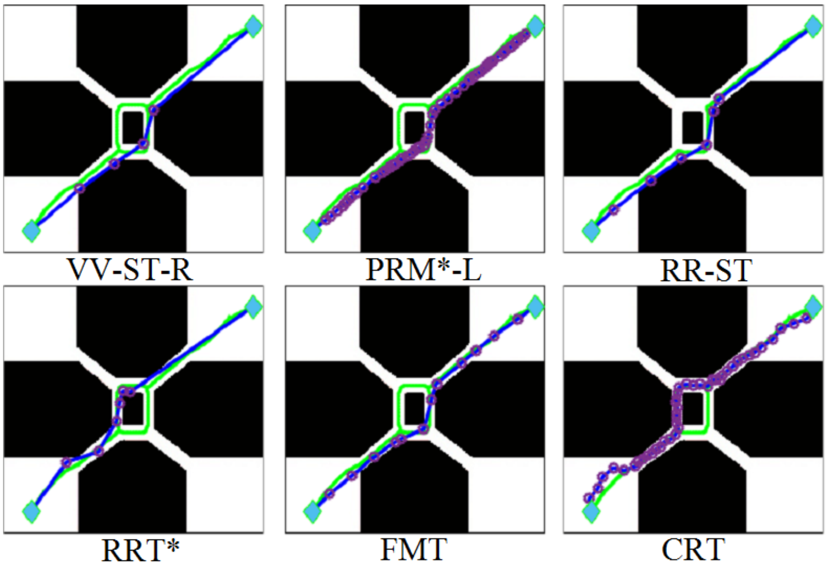


Figure B.12: Z12, $D_S = 0$

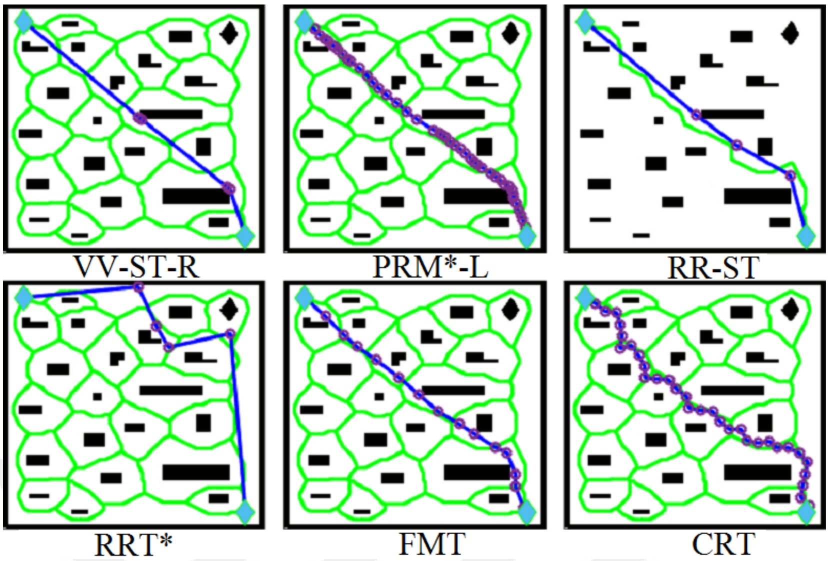


Figure B.13: Z13, $D_S = 0$

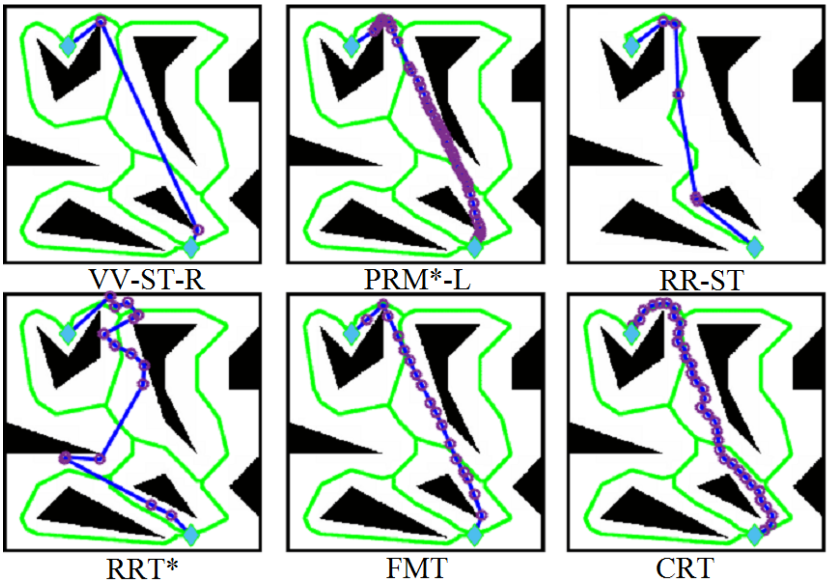
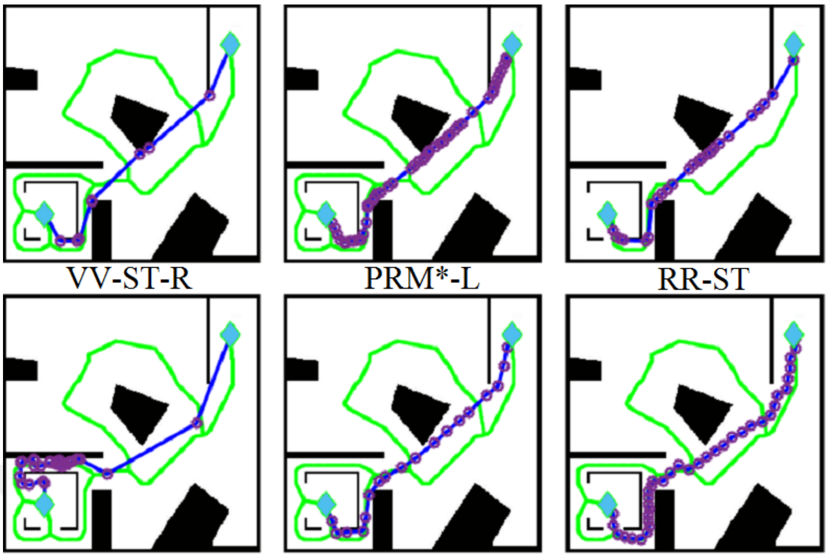


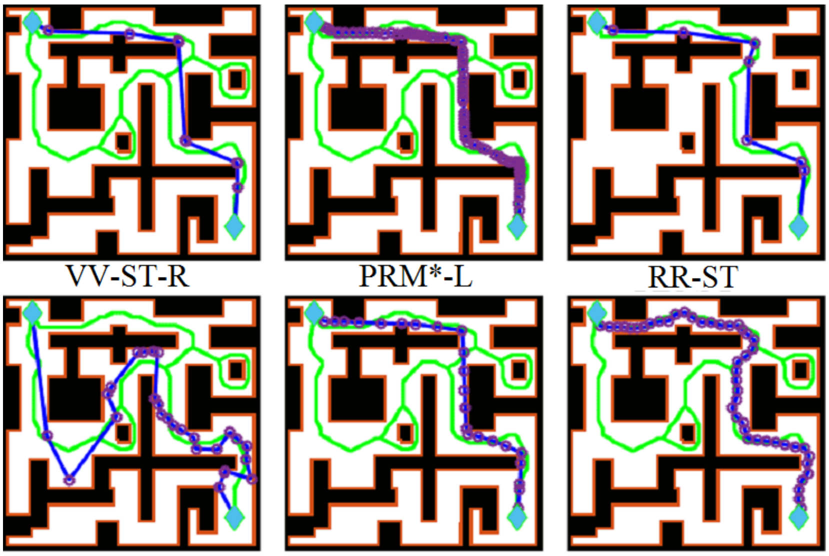
Figure B.14: Z14, $D_S = 0$



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

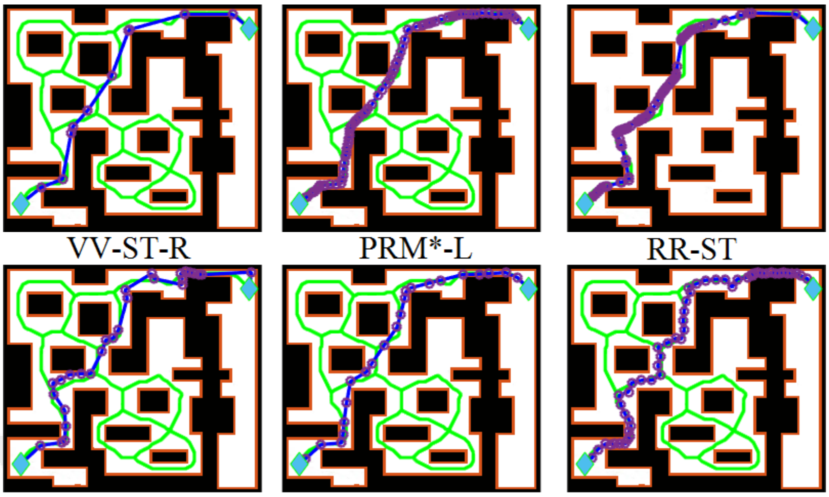
Figure B.15: Z15, $D_S = 0$



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

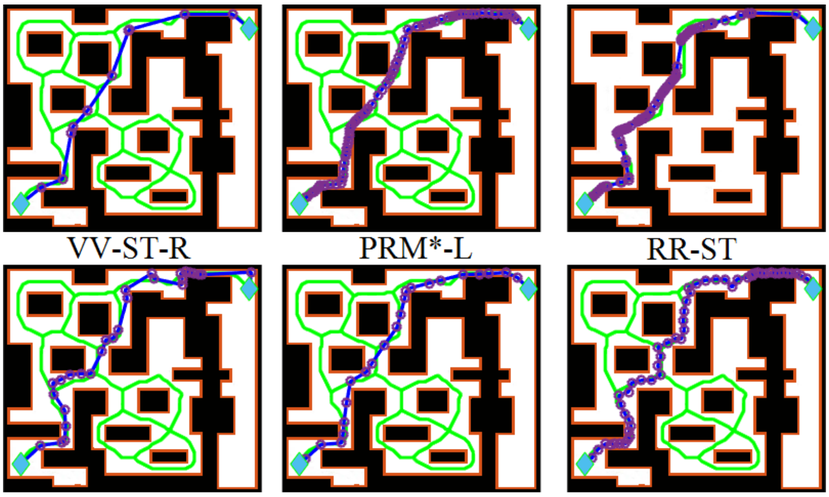
Figure B.16: Z1, $D_S = 5$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

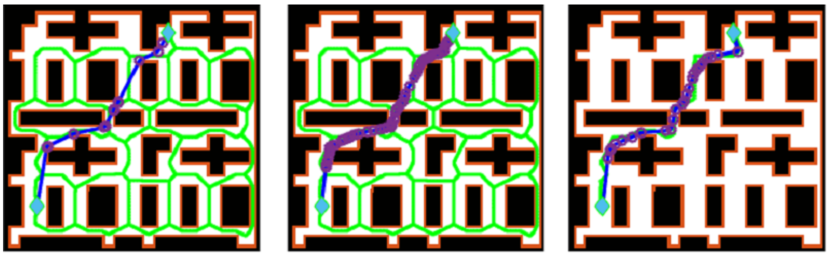
Figure B.17: Z2, $D_S = 5$.



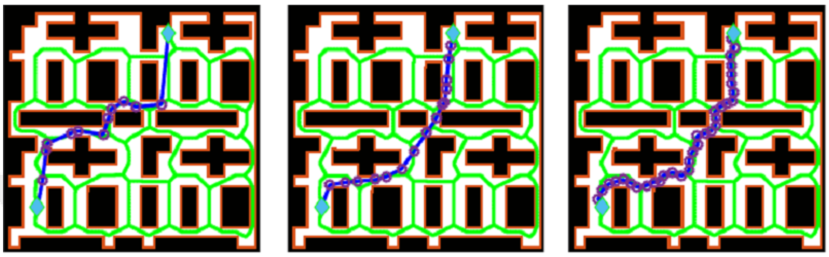
VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.18: Z3, $D_S = 5$.

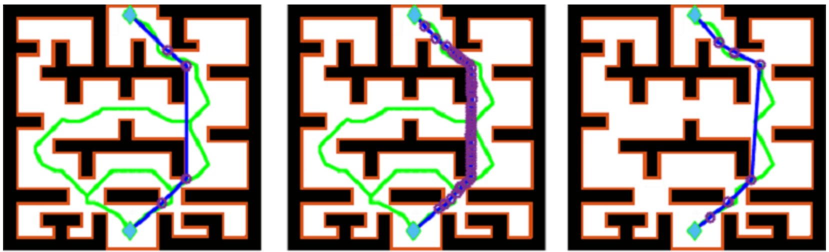


VV-ST-R PRM*-L RR-ST

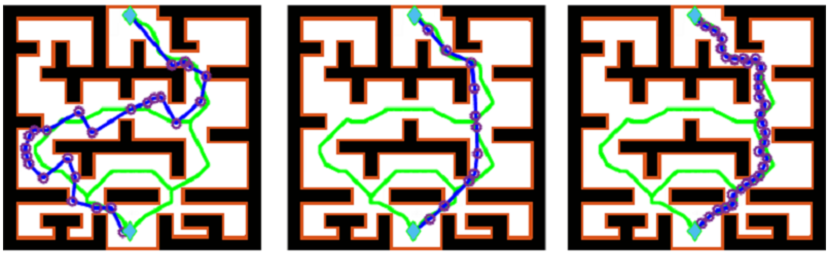


RRT* FMT CRT

Figure B.19: Z4, $D_S = 5$.

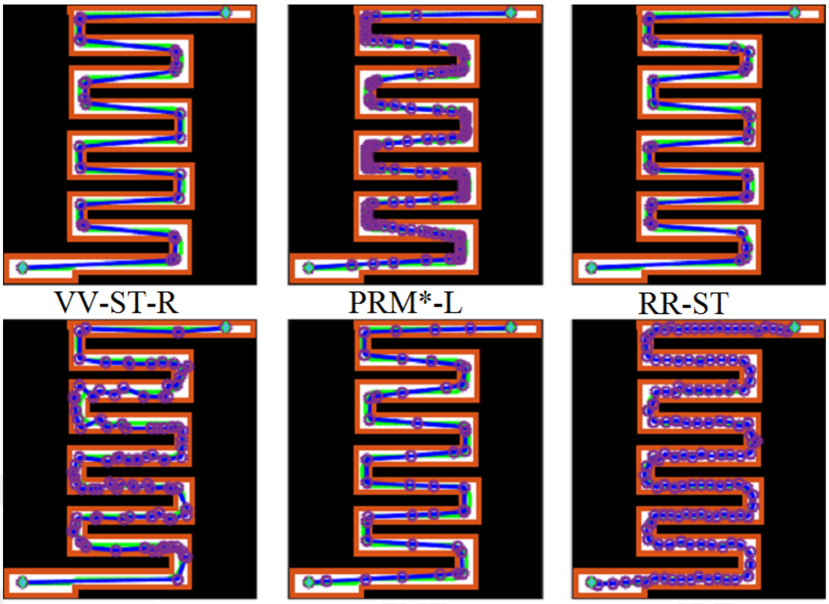


VV-ST-R PRM*-L RR-ST



RRT* FMT CRT

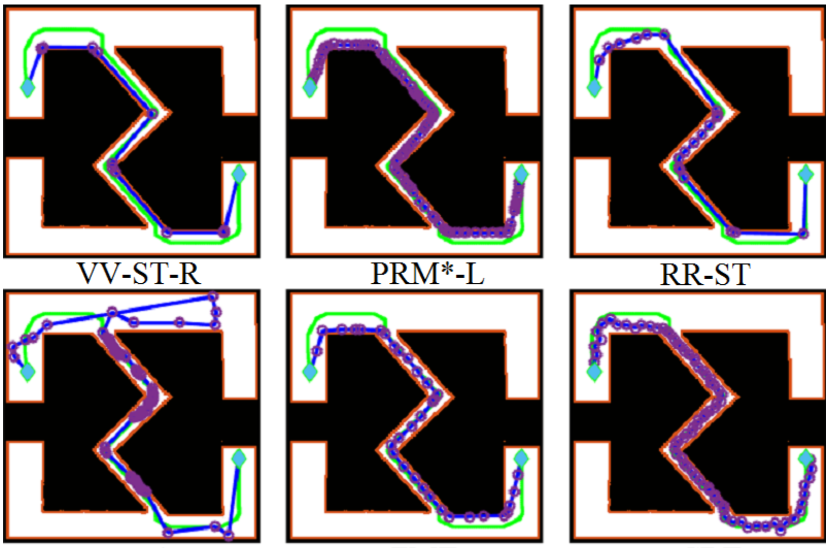
Figure B.20: Z5, $D_S = 5$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.21: Z6, $D_S = 5$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.22: Z7, $D_S = 5$.

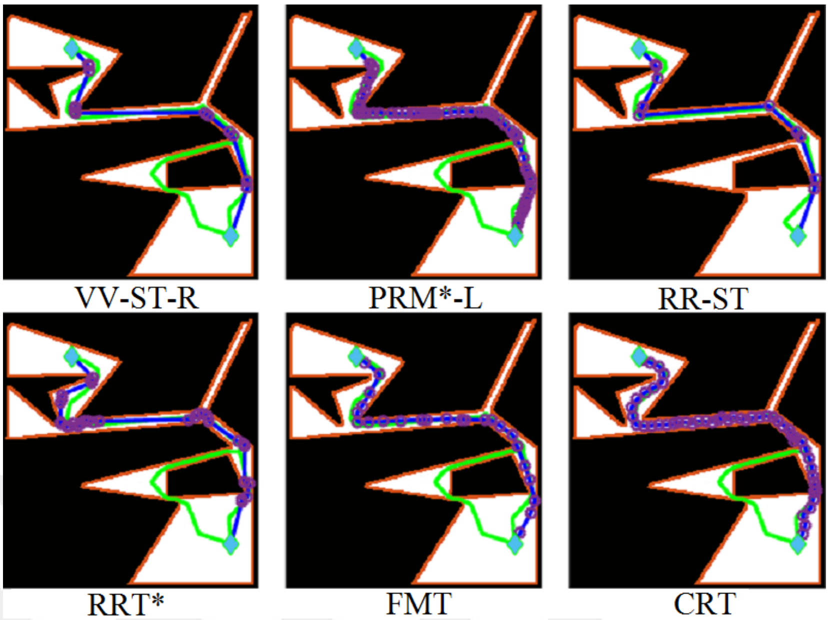


Figure B.23: Z8, $D_S = 5$.

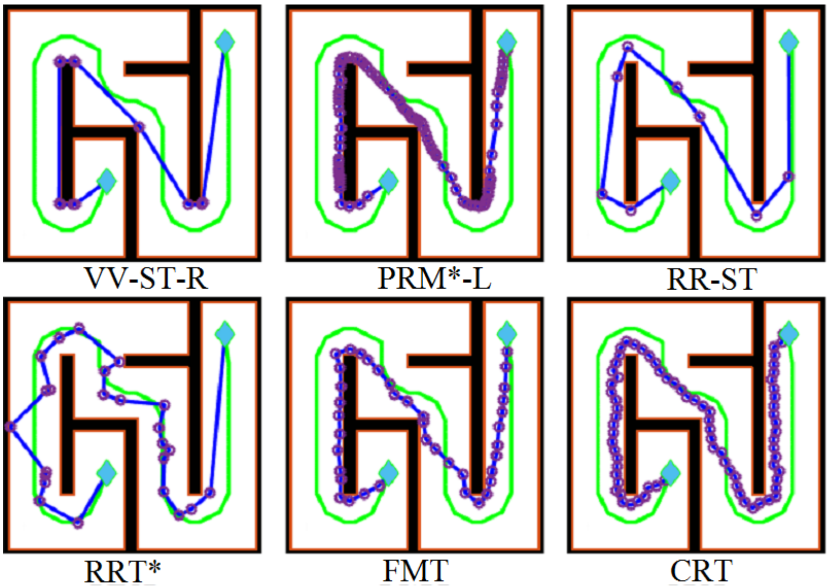


Figure B.24: Z9, $D_S = 5$.

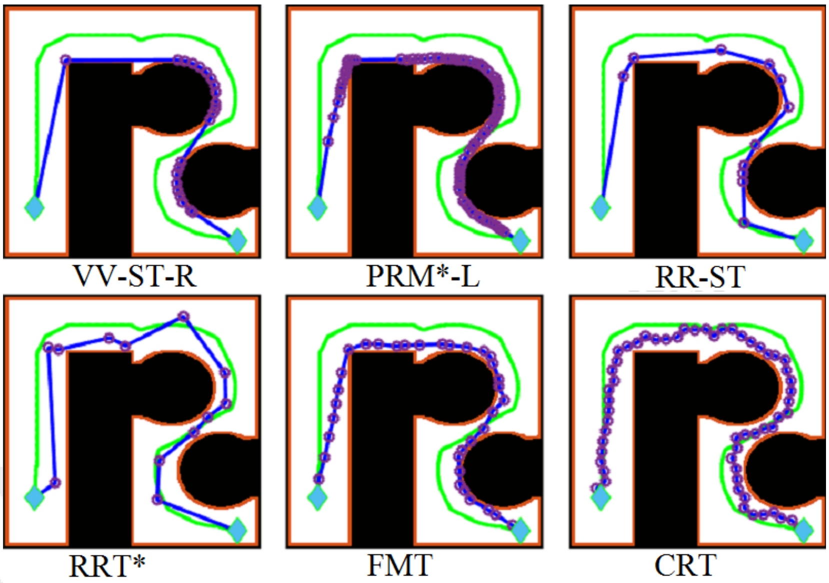

VV-ST-R PRM*-L RR-ST
RRT* FMT CRT

Figure B.25: Z10, $D_S = 5$.

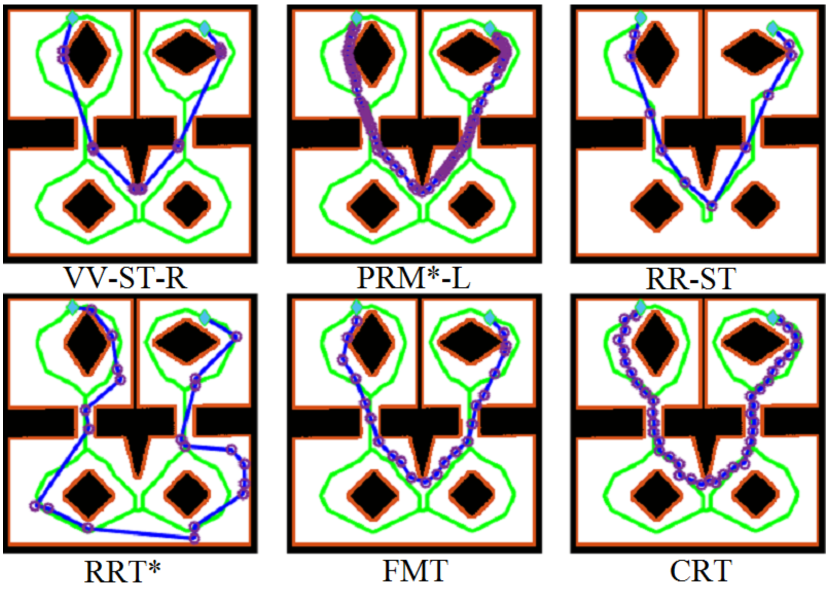
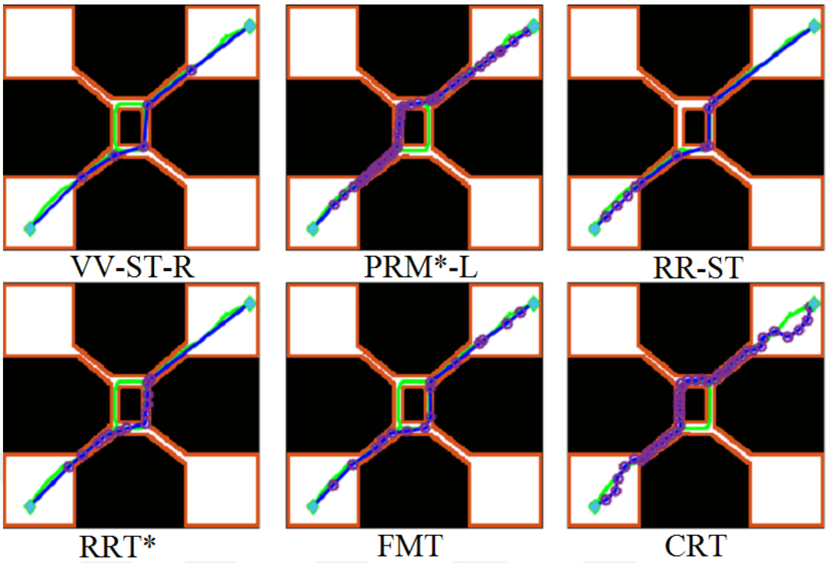

VV-ST-R PRM*-L RR-ST
RRT* FMT CRT

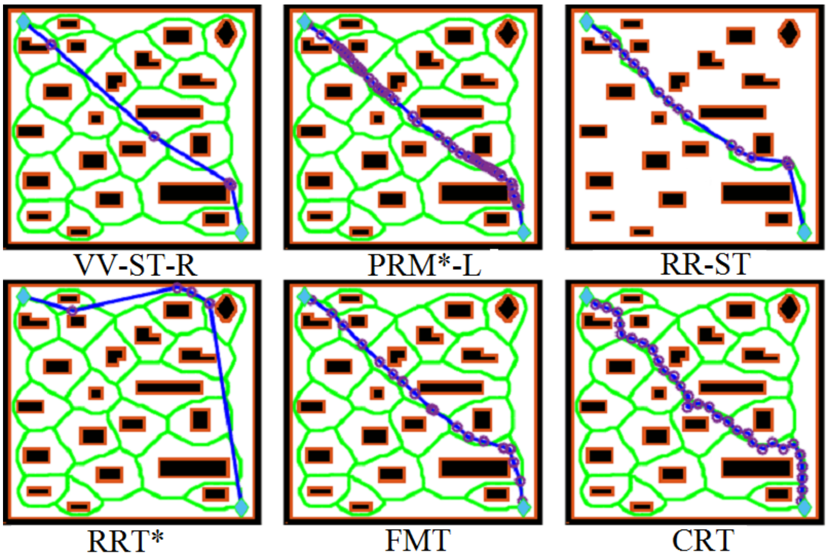
Figure B.26: Z11, $D_S = 5$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.27: Z12, $D_S = 5$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.28: Z13, $D_S = 5$.

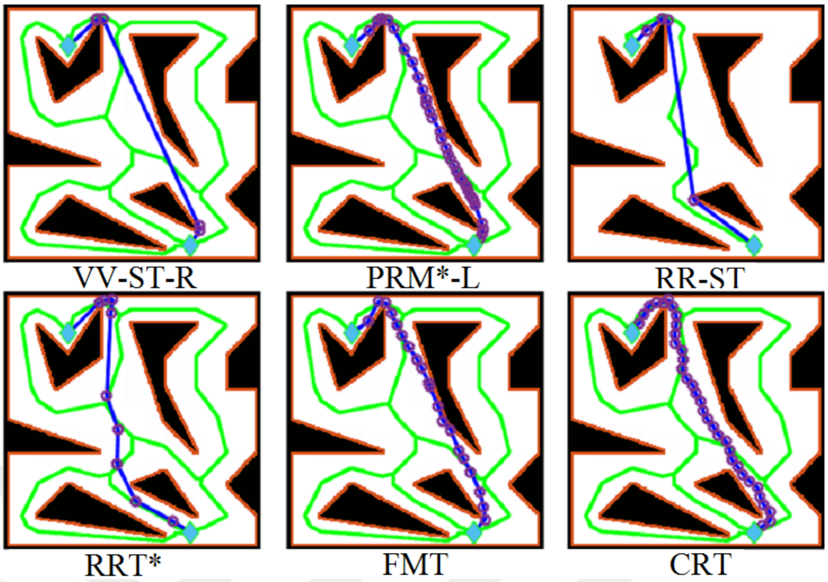


Figure B.29: Z14, $D_S = 5$.

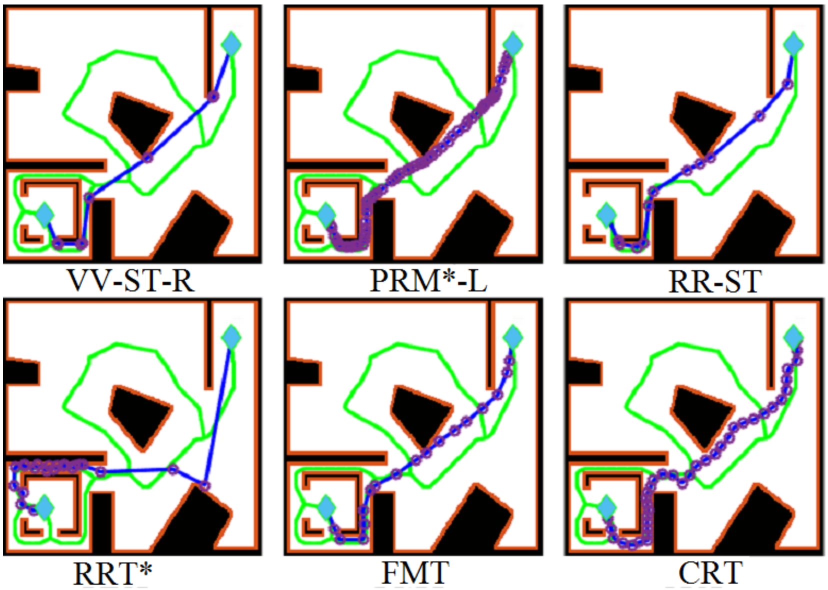
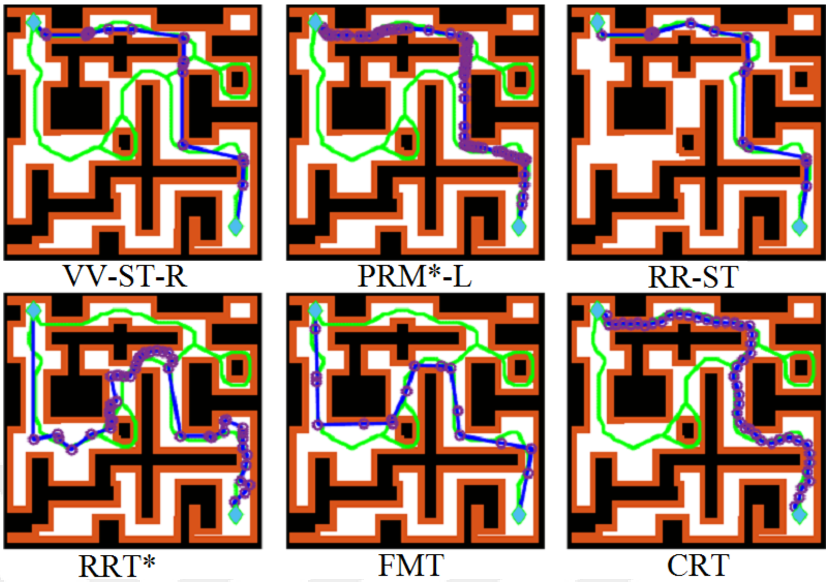


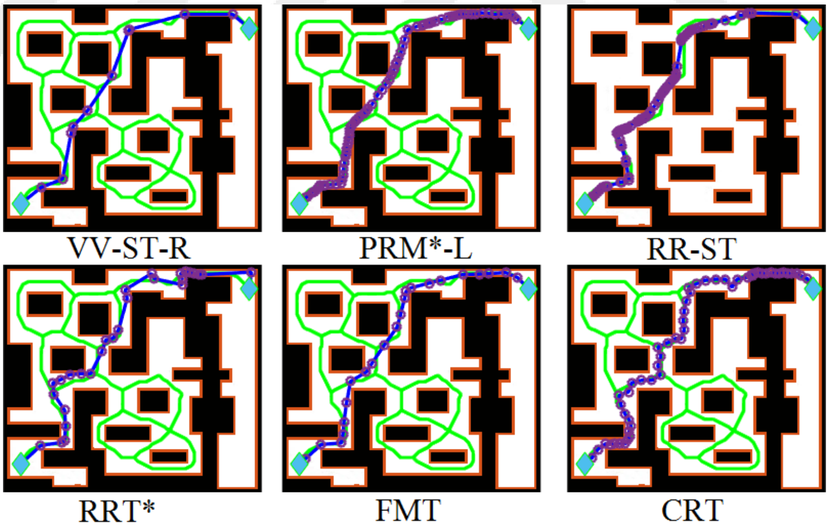
Figure B.30: Z15, $D_S = 5$.

B.3 $D_{\mathrm{S}} = D_{\mathrm{CRT}} + 1$


VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.31: Z1, $D_{\mathrm{S}} = D_{\mathrm{CRT}}$.


VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.32: Z2, $D_{\mathrm{S}} = D_{\mathrm{CRT}}$.

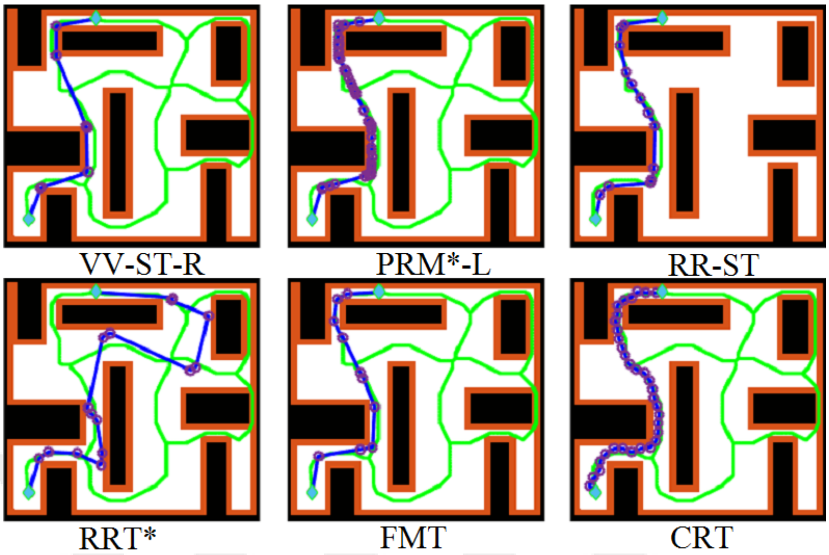


Figure B.33: Z3, $D_S = D_{CRT}$.

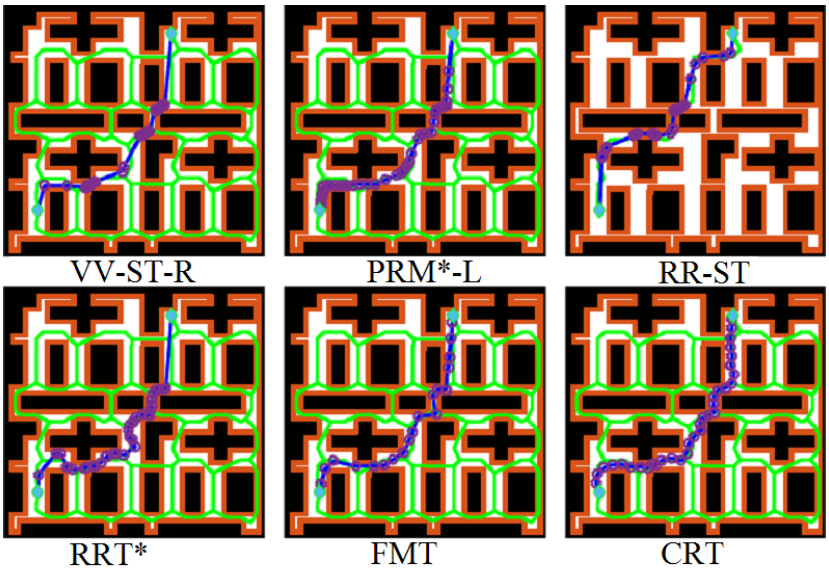


Figure B.34: Z4, $D_S = D_{CRT}$.

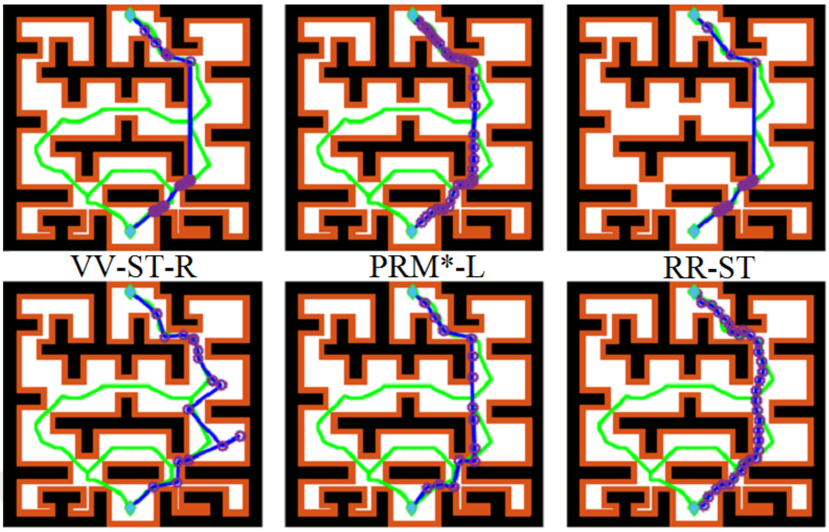


Figure B.35: Z5, $D_S = D_{CRT}$.

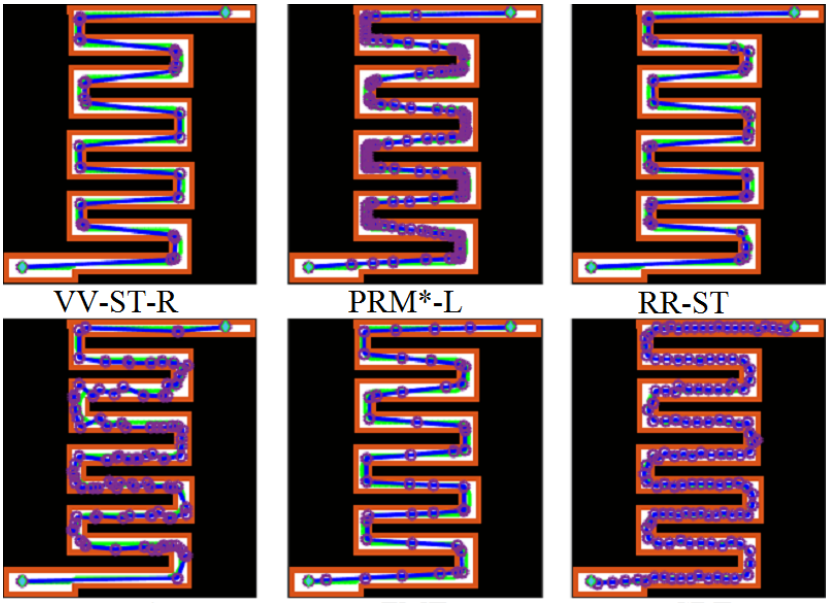
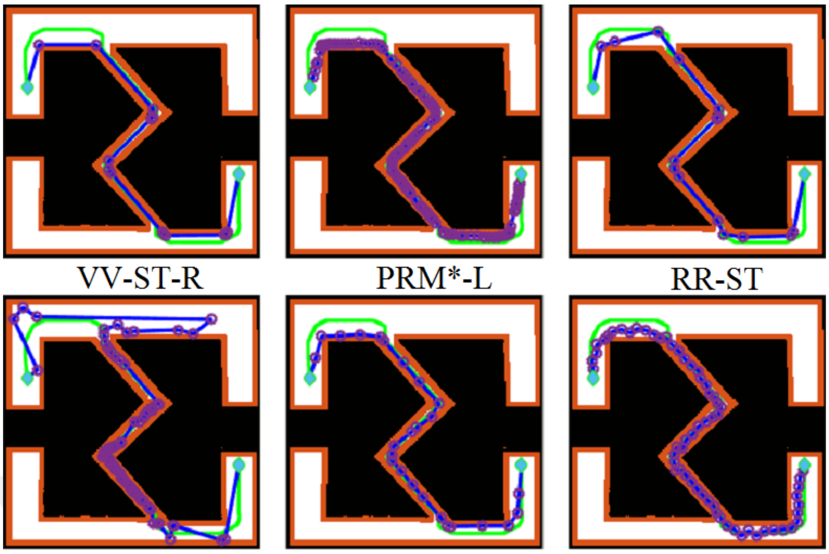


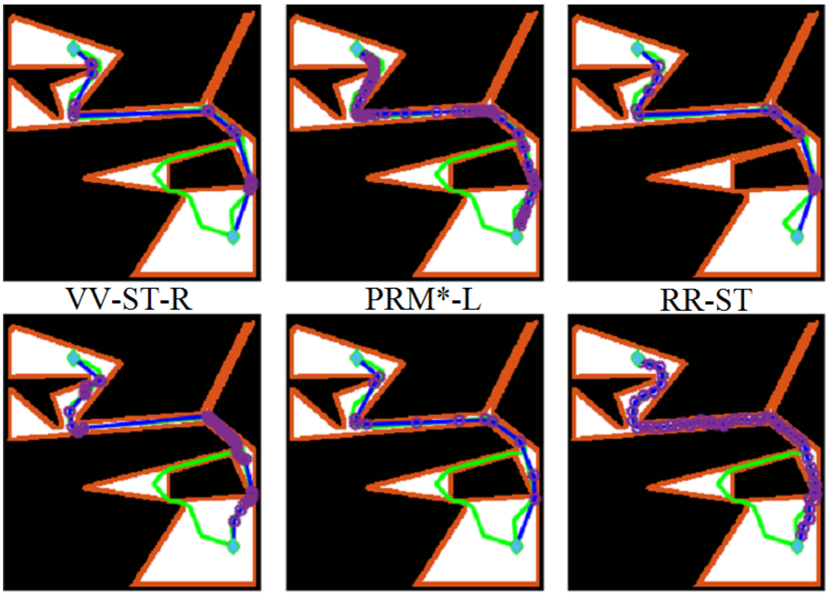
Figure B.36: Z6, $D_S = D_{CRT}$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.37: Z7, $D_{\mathrm{S}} = D_{\mathrm{CRT}}$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.38: Z8, $D_{\mathrm{S}} = D_{\mathrm{CRT}}$.

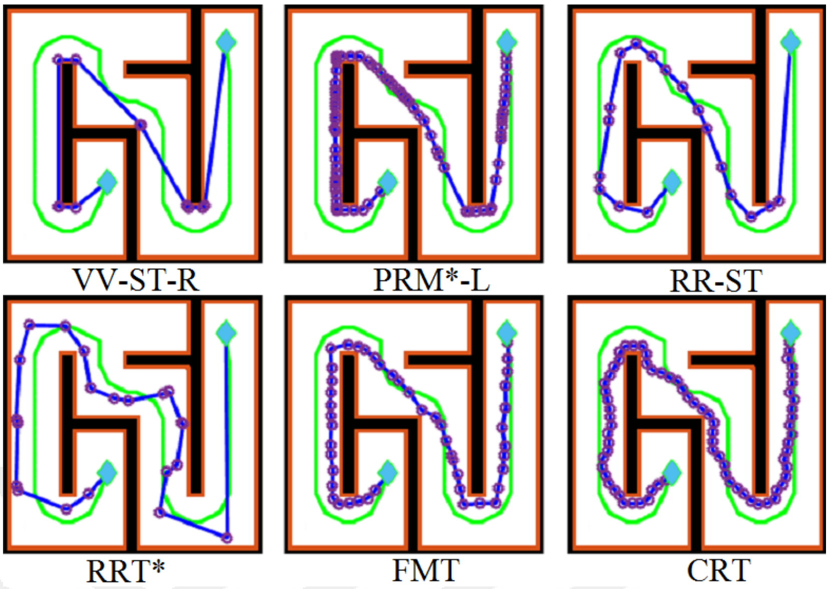


Figure B.39: Z9, $D_S = D_{CRT}$.

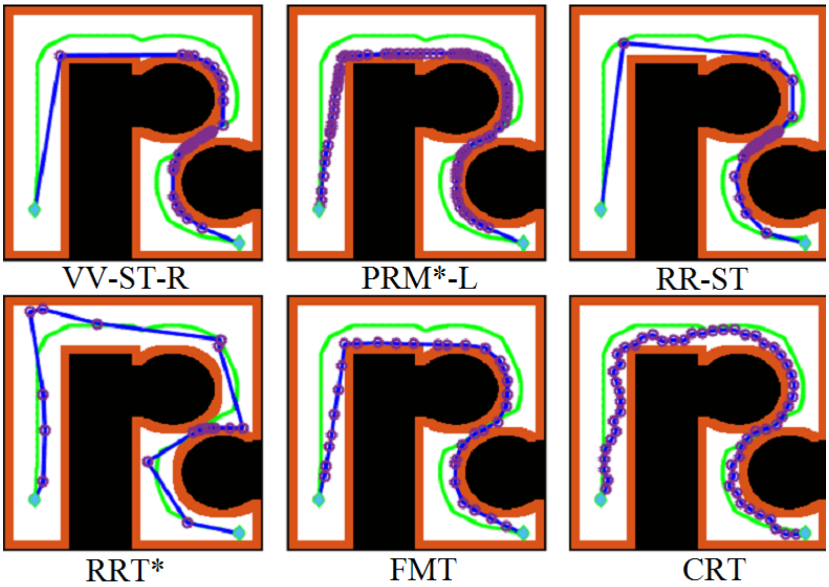
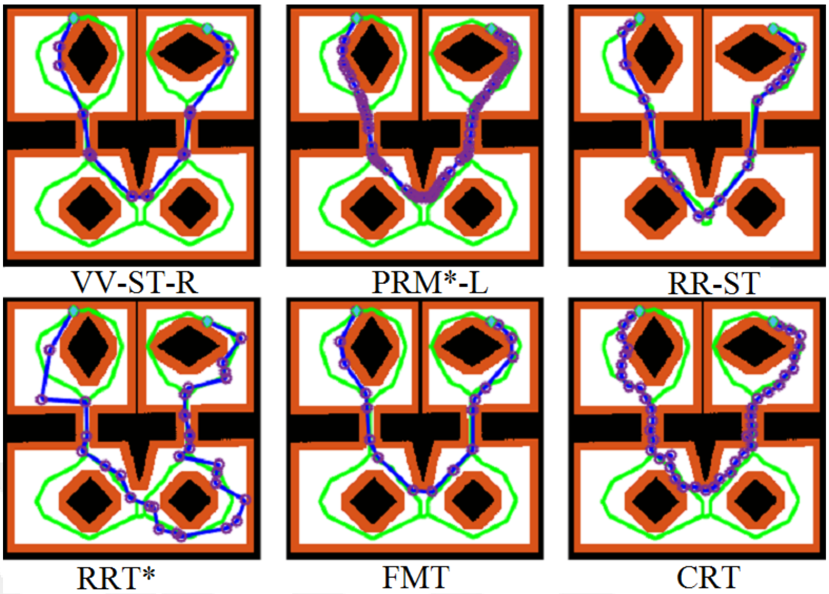


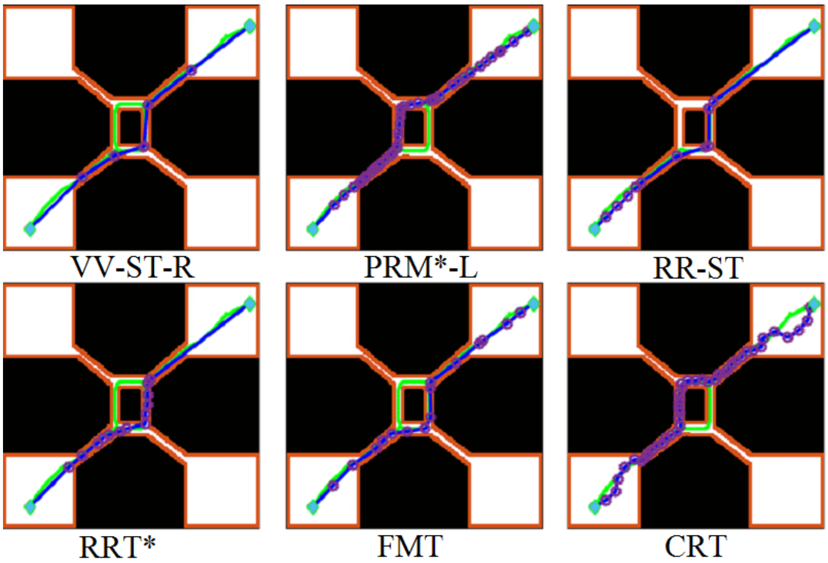
Figure B.40: Z10, $D_S = D_{CRT}$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.41: Z11, $D_S = D_{CRT}$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.42: Z12, $D_S = D_{CRT}$.

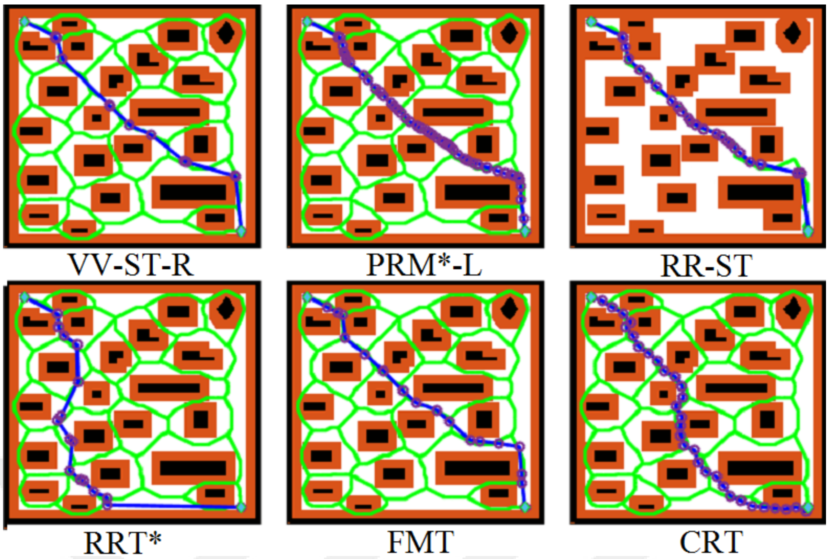

VV-ST-R PRM*-L RR-ST
RRT* FMT CRT

Figure B.43: Z13, $D_{\mathrm{S}} = D_{\mathrm{CRT}}$.

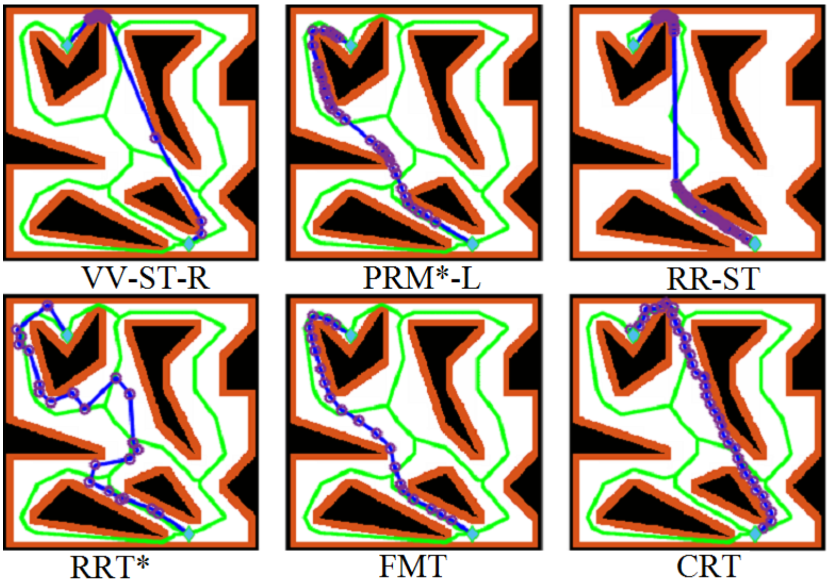
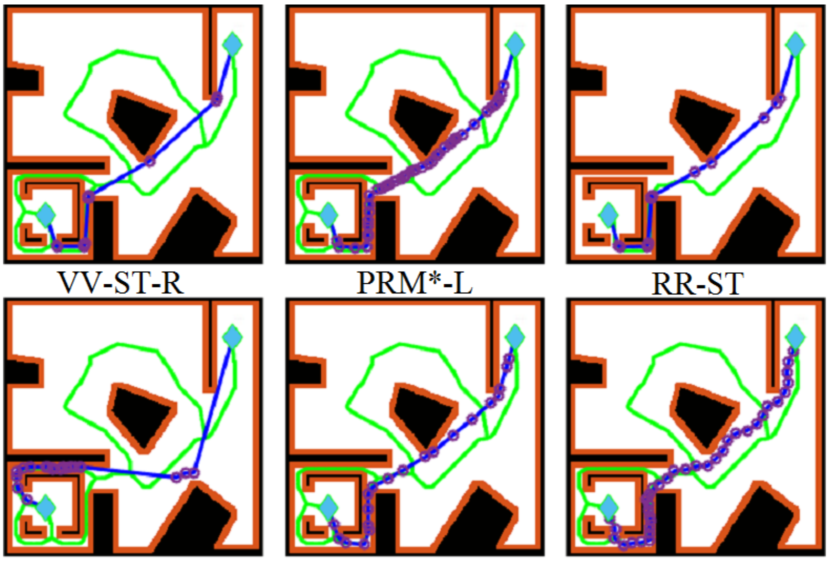

VV-ST-R PRM*-L RR-ST
RRT* FMT CRT

Figure B.44: Z14, $D_{\mathrm{S}} = D_{\mathrm{CRT}}$.



VV-ST-R PRM*-L RR-ST

RRT* FMT CRT

Figure B.45: Z15, $D_{\mathrm{S}} = D_{\mathrm{CRT}}$.

CURRICCULUM VITAE

PERSONAL INFORMATION

Surname, Name: Al-Dahhan, Mohammed Rabeea Hashim Al-Dahhan

Nationality: Iraqi (IRAQ)

Date and Place of Birth: 06/January / 1992 ,Anbar

Marital Status: Single

Phone: +9647711088666, +905383797946

Email: alanymohammed5@gmail.com

EDUCATION

Degree	Institution	Year
MS	Philadelphia university. Mechatronics Eng.	2015
B.Sc.	Al Maaref University College. Computer Eng.	2013
High School	AL-Fallujah high school	2009

FOREIGN LANGUAGES

Arabic, English

PUPLICATIONS

Al-Dahhan, M. R. H., & Schmidt, K. W. (2019). Safe and Efficient Path Planning for Omni-directional Robots using an Inflated Voronoi Boundary. *Çankaya Üniversitesi Bilim ve Mühendislik Dergisi*, *16*(2), 46-69.

Al-Dahhan, M. R. H., & Schmidt, K. W. (2019). Path Planning Based on Voronoi Diagram and PRM for Omnidirectinal Mobile *Robots. Digital Transformation & Smart Systems, Turkey.*

Al-Dahhan, M. R. H., & Ali, M. M. (2016). Path tracking control of a mobile robot using fuzzy logic. In *2016 13th International Multi-Conference on Systems, Signals & Devices (SSD)* (pp. 82-88). IEEE.

HOBBIES

Sport, Reading