VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

EXPERTNÍ SYSTÉMY A POKROČILÉ ALGORITMY V OBLASTI PLÁNOVÁNÍ CEST MOBILNÍCH ROBOTŮ

EXPERT SYSTEMS AND ADVANCED ALGORITHMS IN MOBILE ROBOTS PATH PLANNING

DOKTORSKÁ PRÁCE
DOCTORAL THESIS

AUTOR PRÁCE
AUTHOR
Ing. Ahmad Abbadi

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Radomil Matoušek, Ph.D.

BRNO 2015

ABSTRACT

Motion planning is an active field in robotics domain, it is responsible for translating high-level specifications of a motion task into low-level sequences of motion commands, which respect the robot and the environments constraints.

In this work many path-planning approaches have been reviewed, mainly, the rapidly exploring random tree algorithm (RRT*)*, the cell decomposition approaches (CD), and the application of fuzzy expert system (FES) in motion planning. These approaches have been adapted to solve some of mobile robots motion-planning problems efficiently, i.e. motion planning in small and narrow areas, the global path planning in dynamic workspace, and the improvement of planning efficiency using available information about the working environments.

New planning approaches have been introduced based on exploiting and combining the advantages of cell-decomposition, and RRT, in addition to use other tools i.e. fuzzy expert system, to increase the efficiency and completeness of finding a solution.

This thesis also proposed solutions for other motion-planning problems, for example the identification of narrow area and the important regions when using sampling-based algorithms, the path shortening for RRT, and the problem of planning a safe path.

All proposed methods were implemented and simulated in Matlab to compare them with other methods, in different workspaces and under different conditions. Moreover, the results are evaluated by statistical methods using Matlab and Minitab environments.

KEYWORDS

Motion Planning, Path planning, Rapidly exploring random tree, RRT, Expert system, Fuzzy system, Cell decomposition.

ABSTRAKT

Metody plánování pohybu jsou významnou součástí robotiky, resp. mobilních robotických platforem. Technicky je realizace plánování pohybu z globální úrovně převedena do posloupnosti akcí na úrovni specifické robotické platformy a definovaného prostředí, včetně omezení.

V rámci této práce byla provedena recenze mnoha metod určených pro plánování cest, přičemž hlavním těžištěm byly metody založené na tzv. rychle rostoucích stromech (RRT), prostorovém rozkladu (CD) a využití fuzzy expertních systémů (FES). Dosažené výsledky, resp. prezentované algoritmy, využívají dostupné informace z pracovního prostoru mobilního robotu a jsou aplikovatelné na řešení globální pohybové trajektorie mobilních robotů, resp. k řešení specifických problémů plánování cest s omezením typu úzké koridory či překážky s proměnnou polohou v čase.

V práci jsou představeny nové plánovací postupy využívající výhod algoritmů RRT a CD. Navržené metody jsou navíc efektivně rozšířeny s využitím fuzzy expertního systému, který zlepšuje jejich chování.

Práce rovněž prezentuje řešení pro plánovací problémy typu identifikace úzkých koridorů, či významných oblastí prostoru řešení s využitím přístupů na bázi dekompozice prostoru. V řešeních jsou částečně zahrnuty sub-optimalizace nalezených cest založené na zkracování nalezené cesty a vyhlazování cesty, resp. nahrazení trajektorie hladkou křivkou, respektující lépe předpokládanou dynamiku mobilního zařízení.

Všechny prezentované metody byly implementovány v prostředí Matlab, které sloužilo k simulačnímu ověření efektivnosti vlastních i převzatých metod a k návrhu prostoru řešení včetně omezení (překážky). Získané výsledky byly vyhodnoceny s využitím statistických přístupů v prostředí Minitab a Matlab.

KLÍČOVÁ SLOVA

Plánování pohybu, plánování cest, RRT, rychle rostoucí stromy, expertní system, fuzzy system, rostorový rozklad.

BIBLIOGRAPHIC CITATION:

ABBADI, A. *Expert Systems and Advanced Algorithms in Mobile Robots Path Planning*. Brno: Brno University of Technology, Faculty of Mechanical Engineering, 2015. 149 pp., Supervisor of doctoral thesis: doc.Ing. Radomil Matoušek, Ph.D.

ABBADI, A. *Expertní systémy a pokročilé algoritmy v oblasti plánování cest mobilních robotů*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 149 s. Vedoucí dizertační práce doc.Ing. Radomil Matoušek, Ph.D.

Declaration

I hereby confirm that I have written the thesis independently, under the supervision of doc. Ing. Radomil Matousek, Ph.D. and using the listed references

In Brno, 2. 9. 2015

Ing. Ahmad Abbadi

Acknowledgments

I would like to express the deepest appreciation to my supervisor for his continuous support and guidance.

I would also like to express my gratitude to my parents for their advice, encouragements, and support through my entire life.

Finally, I would like to express my deepest gratitude to my best friend, my wife, Sara for her quiet patience, endless encourage and support.

Table of Contents

ABSTRACT 3

1 INTRODUCTION 9

1.1 THESIS OBJECTIVES 11
1.2 ROBOT HISTORY 12
1.3 THESIS STRUCTURE 13

2 STATE OF THE ART 15

2.1 MOTION PLANNING 15
2.2 CONFIGURATION SPACE 15
2.3 EXAMPLES OF PLANNING ALGORITHMS 16
2.3.1 BUGS ALGORITHMS 17
2.3.2 VECTOR FIELD HISTOGRAM (VFH) 18
2.3.3 ROADMAP ALGORITHMS 20
2.3.4 CELL DECOMPOSITIONS 22
2.3.5 GRID-BASED SEARCH 22
2.3.6 POTENTIAL FIELDS 23
2.3.7 SAMPLING-BASED ALGORITHMS 24
2.3.8 SUMMARY 28
2.4 SAMPLING STRATEGIES 29
2.5 NARROW PASSAGES 33

3 CELL DECOMPOSITION 39

3.1 EXACT CELL DECOMPOSITION 39
3.2 CELL DECOMPOSITION APPROXIMATION 42
3.3 CONTRIBUTIONS, TESTS AND RESULTS 43
3.3.1 SAFE PATH PLANNING USING CELL DECOMPOSITION APPROXIMATION 43
3.3.2 NARROW PASSAGE IDENTIFICATION USING CD APPROXIMATION AND MINIMUM SPANNING TREE 47

4 RAPIDLY-EXPLORING RANDOM TREE (RRT) 55

4.1 CONTRIBUTIONS, TESTS AND RESULTS 64
4.1.1 RRTS REVIEW AND OPTIONS 64
4.1.2 RRTS REVIEW AND STATISTICAL ANALYSIS 67

4.1.3 RAPIDLY-EXPLORING RANDOM TREES: 3D PLANNING 83
4.1.4 SPATIAL GUIDANCE TO RRT PLANNER USING THE CELL-DECOMPOSITION ALGORITHM 93
4.1.5 COLLIDED PATH REPLANNING IN DYNAMIC ENVIRONMENTS USING RRT AND CELL DECOMPOSITION ALGORITHMS 98

5 EXPERT SYSTEM 107

5.1 EXPERT SYSTEM STRUCTURE 108
5.1.1 KNOWLEDGE BASE 108
5.1.2 INFERENCE ENGINE 110
5.2 FUZZY EXPERT SYSTEM 110
5.3 EXPERT SYSTEM APPLICATION IN MOTION PLANNING PROBLEMS 112
5.4 CONTRIBUTION, TESTS AND RESULTS 114
5.4.1 HYBRID RULE-BASED MOTION PLANNER IN CLUTTERED WORKSPACE 114

6 CONCLUSION 127

BIBLIOGRAPHY 129

AUTHOR'S PUBLICATIONS 141

A. APPENDIX: MATLAB IMPLEMENTATION 143

PATH PLANNING IMPLEMENTATION USING MATLAB 143
SOFTWARE SNAPSHOTS 146

1 INTRODUCTION

Robots significantly affect our lives in a positive way. Their successes and desirable outcomes expanded rapidly from manufacturing and industrial application to streets, buildings, gardens, and daily tasks applications.

The major types of old robots were industrial arms and manipulators, they fixed to a base and do a specific task. However, nowadays, a big expansion is done in robotics applications; the mobile robots appear widely among us and take a part of doing human's everyday tasks, e.g. auto-pilot, autonomous car, autonomous vacuum cleaner, autonomous lawn mowers, rescue robots, and many other applications.

The autonomous mobile-robot field has been a subject of many researches last years. The high demand of autonomous robot applications motivates the researchers and scientists to increase the machine autonomy by introducing a new designs, ideas, and algorithms, especially in the applications that involve critical requirements, dangerous environment, or boring tasks.

The complexity of autonomous robots requires an efficient motion planner, which convert high-level tasks specifications into low-level descriptions of motion commands. The output of the planner is a motion plan or a path plan, which includes a sequence of actions to be executed by robots controllers and actuators.

A planner constructs a plan using planning algorithms that find a suitable control inputs given a state of the workspace. An efficient planner should rapidly and reliably computes a collision-free path, and respects the robots constraints or other kinematic and dynamic constraints.

Motion planning problems can be divided into three levels, based on the planning goals, i.e. local planning, global planning, and mission planning as shown in Figure 1-1, the motion planner module.

The local planners produce a solution locally based on sensors data. They do not require a map or any initial information except the goal location. Most of these algorithms are easy to implement and require low computational resources. The main advantage of them is the tolerance to the environment changing. However, many local planning algorithms trap in local minima, moreover, they are incomplete, and generate un-optimized paths.

In the global planning, the algorithms produce a full path from the initial position to the goal states. Usually, these algorithms require middle to high computational resources. Moreover, they require the initial and goal locations, in addition to the map of the workspace. The main advantage of these planners, that they avoid the local minima. Yet, they have less tolerance to the environment changing.

Mission planning is a high abstraction of the required tasks. The query may have multi-goals, and the planner in this level tries to find a way that satisfy all constraints and reach the goals.

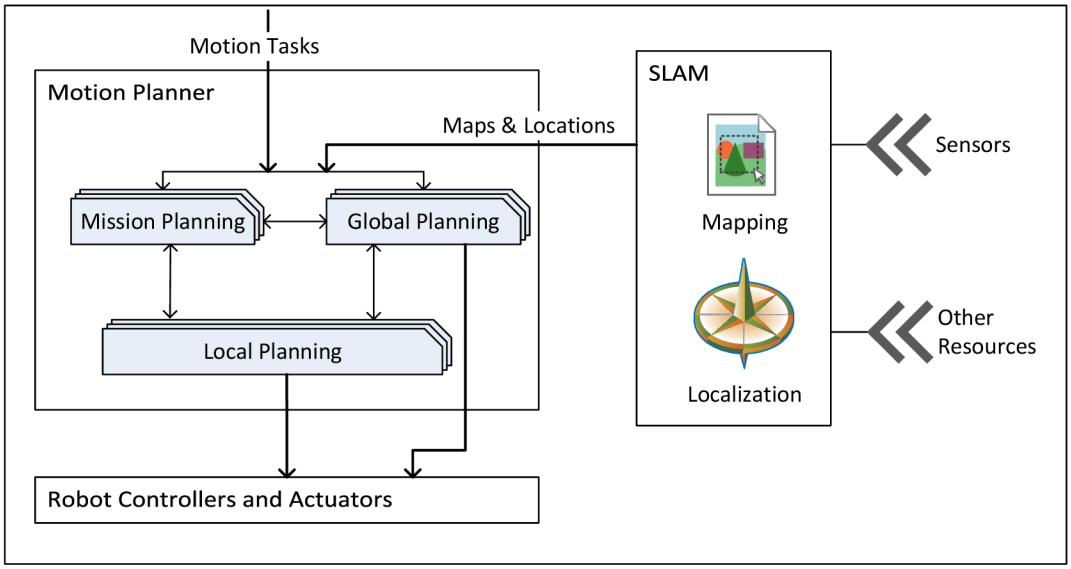


Figure 1-1: Robot navigation model.

Another complement module to the motion-planning one is the localization and mapping module. It estimates the robot location and improves awareness of the system to the surrounding environments. It is also responsible for self-localization in an unknown area, and builds a map for the explored spaces. This process is known as simultaneous localization and mapping (SLAM) (Leonard et al. 1991; Smith et al. 1986), which is out of our interest in this thesis.

The hierarchy of the motion-planning module is proposed in many researches based on the application. For example, in (Vendrell et al. 2001), the authors construct a planner using five steps, first the Mission, which deals with the highest and most abstract definition of an activity. Also, it is independent from the robot and the environment. For instance, "load part X from place Y to place Z." the second level is the Task, which corresponds to a sub-goal from the whole goal. Other levels are the Motions and Actions, they responsible for translating the plan into a set of robot orders and basic operations, which executed by the last level, the Robot Orders level. Other proposed architectures presented in (Kelouwani 2013; Knepper et al. 2010; Moore et al. 1999).

Our interest in this thesis is to develop the local and global parts of the motion-planning module for omnidirectional mobile robots.

The robot is considered as a holonomic points operates in static or dynamic workspace. The rapidly exploring random tree algorithm (RRT)[1] and its developments are reviewed,

[1] Will be discussed in chapter 4

and tested to estimate their efficiency and completeness. Then statistical studies on RRT variants have been done, in order to find alternatives to the methods that have low probabilistically completeness. We tested the RRT performance, and introduced a new method for RRT's path shortening, in addition, we utilize a smoothing-out technique to improve the generated path. The shortening algorithm reduces the number of redundant points in the path, and reduces the detours edges, in order to make the path more suitable for omnidirectional mobile robot.

We have developed new motion planners based on cell-decomposition (CD) algorithms[1]. They generate a plan that keeps a safety distance between the robot and the obstacle boundaries, and, at the same time, push the robot to perform its maneuvers in large free regions in the workspace. Moreover, new planning-algorithms were proposed and developed in order to build efficient planners. The first category of these approaches combines RRT algorithms and CD methods. It overcomes the drawbacks of RRT algorithms in narrow areas and cluttered workspaces, what is more, it overcomes the CD downsides in dynamic workspaces. Another work has been done using CD and minimum spanning tree (MST) to identify the *narrow passage* and the important regions form sampling-based algorithms point of view.

The second category of the planning algorithms uses an expert rule-based, with the aim of utilizing the collected experience, and available knowledge to generate a better solution in an efficient way. The goal of these proposed methods is to develop and improve RRTs planners for omnidirectional mobile robot, by exploiting the available knowledge in the environment.

1.1 Thesis objectives

The aims of the thesis are to improve the mobile robot strategy for path planning, by proposing new approaches to improve the completeness and efficiency of planning algorithms, which in consequence improve the robot's autonomy. Then assert the results statistically, and compare it to other methods.

The clear aims of the thesis can be summarized in the following points:

- Review of the state of the art.
- Design new approaches for path planning based on RRT and cell decomposition principles.
- Use knowledge base and expert system in the path planning methods.
- Design simulation environment that conducts simulations of the experiments, and evaluates the results statistically.

[1] Will be discussed in chapter 3

1.2 Robot History

The term "robot" was first used to denote fictional automata in the 1921 on the play “Rossum's Universal Robots” by the Czech writer Karel Čapek. He uses the word ‘robot’ to describe artificial people. The term *robot* comes from the Czech work ‘Robota’ which means forced labor that work without rest (Etymonline 2014; Slovník).

The Idea of producing autonomous machines or pre-programmable machines to serve the people or replace them in some situation was proposed frequently over the ages. Back to ancient worlds, a Chinese artificer Yan Shi (BC 1000) designed a mechanical handiwork, which was able to sing and act (Uyanik 2011). In 320 BC, Greek philosopher Aristotle wrote “If every tool, when ordered, or even of its own accord, could do the work that befits it, Then there would be no need either of apprentices for the master workers or of slaves for the lords”. After that, one of the oldest known automaton was made by ancient Egyptians (250 BC) “Clepsydra” which is a clock propelled by water (Mathia 2010; Wikipedia 2014a; sciencekids 2014).

In golden Islamic age, the polymath “Al-Jazari” which known as the creator of the first programmable humanoid robot (Uyanik 2011), wrote a book in (1206) describing the design and construct of a number of automatic machines, including kitchen appliances, musical automata powered by water. In addition to the first programmable humanoid robot which was a programmable drum machine consisting of four automatic musicians in a boat floating in a lake (Uyanik 2011; Wikipedia 2014a; Al-Jazari-Wikipedia 2014).

In 1495, Leonardo Da Vinci designed a humanoid automaton that does human-like movements. Then, around 1700, many automatons were built. Jacques de Vaucanson (1737) made many automatons like flute player, tambourine player, and his most famous work, "The digesting duck.” The Japanese craftsman Hisashige Tanaka created an array of extremely complex mechanical toys, some of which were capable of serving tea, firing arrows drawn from a quiver, or even painting a Japanese kanji character (Wikipedia 2014a).

In the recent centuries, the automation takes a place in 1913, when Henry Ford installs the world’s first moving conveyor belt-based assembly line in his car factory, which make assembling time for Model T fell from 12 hours and 30 min to 93 minutes. Then, many modern robots start to appear in different applications (Uyanik 2011; Wikipedia 2014a; sciencekids 2014).

The first digitally operated and programmable arm robot was invented by George Devol in 1954. It is known as “Unimate.” It became the first industrial robot, completing dangerous and repetitive tasks in an assembly line at General Motors (1962), and laid the foundations of the modern robotics industry.

In 1950, Alan Turing proposes a test to determine if a machine truly has the power to think for itself. To pass the test a machine must be indistinguishable from a human during

conversation. It has become known as the ‘Turing Test’ of intelligent behavior. Then in 1980 John Searle shows, that the test of intelligence is not so easy. He proposes the paradox with name 'Chinese Room'. But it is another story of the beginning of the artificial intelligence.

1.3 Thesis structure

The thesis is divided into six chapters. The second one, the state of the art, contains an overview of the famous methods in the motion-planning domain, and some approaches, which are adapted and used in this work.

The third, fourth and fifth chapters describe the used algorithms, each of these chapters is divided into theoretical parts in addition to our contribution part. The contribution section is divided into subsection based on our publications. Each one of the subchapters contains a description of our methodologies to solve a specific problem. In addition, it presents the testing results, and the discussion. The theoretical part of the chapter contains an introduction and related words, which are used in the corresponding publications. They combined and reviewed in a logical sequence.

The Third chapter reviews the cell decomposition algorithm (CD) and its improvements. It is started with the theoretical part, which review many researches and developments of CD approaches, while the second part contains our contribution in safe path planning using cell-decomposition approaches and narrow area identification.

The fourth chapter describes the principle of rapidly exploring random tree algorithms (RRT), and its variation. The chapter has been started with a general introduction of RRT and its principle, then a deep review of its developments and the related works. The last sections of this chapter present our contributions to develop the path shapes, the algorithms completeness, and the efficiency of the planners using the combination between RRT and other approaches.

The fifth chapter discusses the use of expert system in the path-planning problem. It describes our methods that exploit the available information in order to support the motion planning procedure. The chapter starts by describing the basic principles of expert systems, the hierarchy of ES, knowledge-based representation, fuzzy expert system, and then a revision of ES in mobile robot motion-planning problem is done. The last section of this chapter presents our contribution in this domain to build a hybrid planner using fuzzy expert system, RRT, and CD algorithms.

In the last chapter of this thesis, we conclude our work, and then, we list the references, which are used in this work. In appendix section, some snapshots of the simulation and testing application are presented.

2 STATE OF THE ART

In this section, some motion planning concepts are reviewed. We start with basic concepts of motion planning and the need for configuration space, which led to recent motion planning algorithms. Then, the original applicable ideas for motion planning are described in the examples of planning algorithms section, which is started by a survey of the exact and geometry methods and it is ended with sample-based methods.

2.1 Motion planning

Motion planning is the process of finding feasible movements in a continuous world. The feasible movements displace the robot toward the goal state and at the same time do not collide into obstacles, or violate environment's constraints. The robot‘s models and its working environments should be specified in motion planning problems. The robot’s model contains robot’s dimensions, kinematics, differential equations, and other parameters, which control or constrain robot movements. The model of a working environment contains maps, obstacles representation, and robot location.

The principle of using two models to formulate the robot and its environment causes some difficulties and complexity to solve the motion-planning problem, especially in the high dimensional workspaces.

A new principle is proposed to represent the robot and its environments in different ways. A configuration space is proposed to represent the robot as a point in the space, and convert the complexity of robot model to dimensions in the configuration space. The dimension of the configuration space corresponds to the number of degrees of freedom of the robot. The advantage of using the configuration space is the motion-planning problem will be viewed as a searching in a high-dimensional configuration space, which contains implicitly the representation of the obstacles. In consequence, the motion plan will be defined as a continuous path in the configuration space. Based on this proposal the *path-planning* term and path-planning algorithms is proposed as methods to find a continuous path over configuration space. The other term in this context is *trajectory planning*, which expresses the action of finding a continuous path over the configuration space, which respects the dynamic constraints, such as velocity, acceleration, inertia, etc., which means the plan contains a continuous path and the control input for every node of this path.

2.2 Configuration space

The configuration space (C-space) for motion planning is discrete space. It contains a set of all possible transformations that could be applied to the robot. The idea of the configuration space is introduced in (Lozano-Pérez et al. 1979).

The mapping between workspace and C-space is straightforward. A point in workspace corresponds to a set of configurations in C-space (LaValle 2006, chap. 4; de Berg et al. 2008, chap. 13).

A free configuration q is a position where the robot does not collide obstacles or itself. Each sample from workspace is classified as free or non-free configuration. A set of all free q is called the free configuration space, while the obstacle space or the forbidden region is the complement of the free space.

The degree of freedom (DOFs) of a robot is considered as dimensions in its C-space, e.g. a robot with n degree-of-freedoms is represented by n-dimensions C-space. For example, if a robot is represented as a single point (zero-sized) translates in a 2D plane (the workspace), then C-space is a plane, and configurations are represented using two parameters (x, y). If the robot translates and rotates in 2D workspace, then the C-space is 3D and the configurations are represented using three parameters (x, y, θ) where θ is the head direction. If the robot translates and rotates in 3-dimensional workspace, then the representation of any configuration requires six parameters (x, y, z) for translation, and the Euler angles (α, β, γ) for rotation. In some problem, the robot is considered as a single point by transforming the robot dimensions to the obstacles dimensions. This process uses some methods, e.g. Minkowski sum (Wikipedia 2014b; de Berg et al. 2008, pp. 290–296).

2.3 Examples of planning algorithms

In this section, some famous algorithms that used in the motion-planning domain have been reviewed and a brief information about the bases of these methods is given. In our work, we adopt and adapt some of these methods to support our proposals.

In the context of motion planning, different approaches have been developed. Some of them use geometric models, which construct a map/graph and use it for path planning, e.g. roadmap (Choset, Howie et al. 2005), visibility graph (Lulu et al. 2005; de Berg et al. 2008) , Voronoi diagram (LaValle 2006; Aurenhammer 1991; Aurenhammer et al. 2000; Garrido et al. 2011; Choset et al. 2000; Fabbri et al. 2002; Shkolnik et al. 2009; Sakahara et al. 2008; Masehian et al. 2010; de Berg et al. 2008, chap. 7) , and cell decomposition (Katevas et al. 1998; LaValle 2006; Choset, Howie et al. 2005; Milos Seda 2007; Latombe 1991; Brooks et al. 1985; Schwartz et al. 1983; de Berg et al. 2008; Sleumer et al. 1999; Bernard Chazelle 1987; Hwang et al. 2003).

Another category uses a grid over the workspace, e.g. artificial potential field (Arambula Cosío et al. 2004; Hani Alsafadi 2007; Masoud 2013; Mbede et al. 2000; McFetridge et al. 1998; Pêtrès et al. 2012; Sfeir et al. 2011; Zhang et al. 2012; Khatib 1985; Rosell et al. 2005; Hwang et al. 1992; Kim et al. 1991; Masoud 2013), and vector field histogram (Borenstein et al. 1991).

Another category includes sampling-based algorithm, which described in more detail in separate sections. In the next paragraphs, we have listed the basic idea of some algorithm, which is widely used in motion planning problems.

2.3.1 Bugs algorithms

Bug algorithms are used for local path planning with minimum sensor and computation requirements. They assume the robot as a point operates on a plane with ranging sensors (Choset, Howie et al. 2005, chap. 2; Kamon et al. 1998; Lumelsky et al. 1986). May Bug-like improvements were introduced later. In (Buniyamin et al. 2011; Ng et al. 2007) the authors reviewed many different variations. In the next section, we shortly explain the principle for the basic bug algorithm.

The Bug1 algorithm (Lumelsky et al. 1986) exhibits two behaviors: “motion-to-goal” and “boundary-following”. During “motion-to-goal,” the robot moves along the line toward the goal until it encounters either the goal or an obstacle. If the robot encounters an obstacle, the robot then circumnavigates the obstacle until it returns to the first hit point. Then, it determines the closest point to the goal on the perimeter of the obstacle and traverses to this point. This point is called the leave point. From that point the robot heads straight toward the goal again, i.e., it re-invokes the “motion-to-goal” behavior.

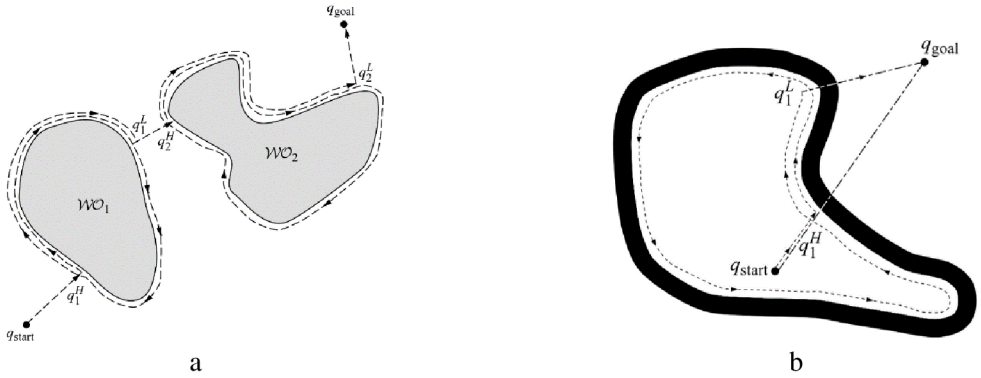


Figure 2-1: Principle of Bug1 algorithm; a: the algorithm finds a path to goal, b: no path to goal exist. Source (Lumelsky et al. 1986)

In the case, when the line, that connects the leave point to the goal one, intersects with the current obstacle, the algorithm fails and there is no path to the goal location. This case is shown in Figure 2-1-b. Otherwise, the procedure is repeated until the goal is reached, as shown in Figure 2-1-a.

Bug2 is similar to Bug1 (Lumelsky et al. 1986); it has also two behaviors: motion-to-goal and boundary-following. During motion-to-goal, the robot moves toward the goal as in Bug1. But in Bug2 the line which connects the initial point to the goal point remains fixed as shown in Figure 2-2-a. The “boundary-following” behavior is invoked when the robot encounters an obstacle. This behavior is different from Bug1, where in Bug2 the robot circumnavigates the obstacle until it reaches a new point on the fixed line. If this new

point is closer to the goal than the first intersection point, then, the robot proceeds toward the goal, Figure 2-2-a. The algorithm repeats this process if it encounters other obstacles. However, when the robot re-encounters the original departure point, in this case there is no path to the goal, as shown in Figure 2-2-b.

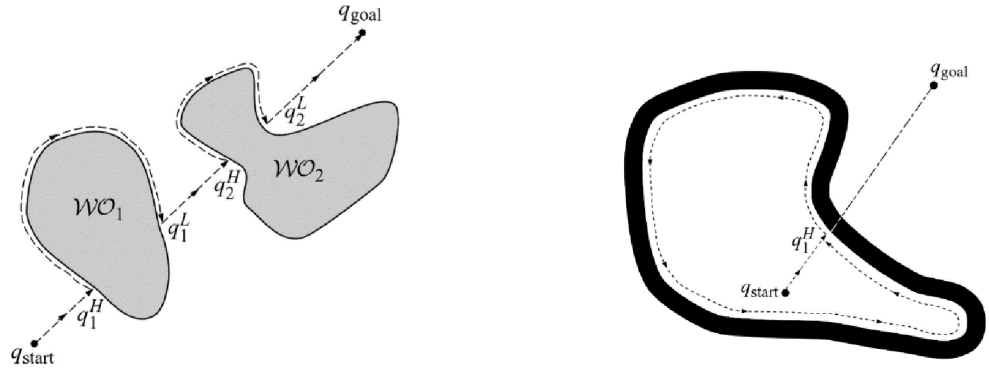


Figure 2-2 Principle of Bug2 algorithm; a: the algorithm find a path to goal, b: no path to goal exist. Source: (Lumelsky et al. 1986)

One limitation of the Bug algorithm is that the robot's behavior depends only on its most recent sensor readings. This can lead to problems where the robot's instantaneous sensor readings do not provide enough information for robust obstacle avoidance. This limitation is solved later on by the Vector field histogram (VFH) techniques by creating a local map of the environment around the robot.

2.3.2 Vector field histogram (VFH)

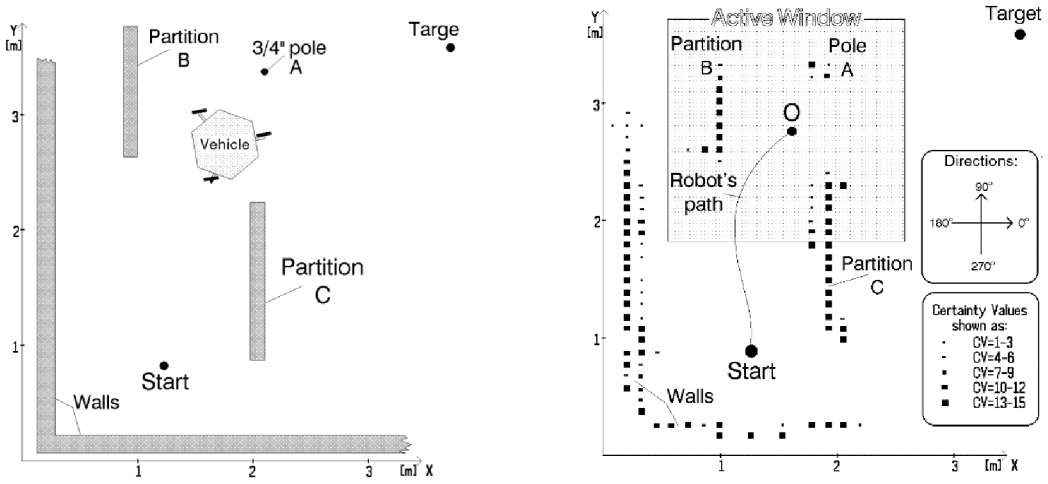
VFH is a real time motion planner. It is proposed in (Borenstein et al. 1991) as a local planner. The VFH utilizes a statistical representation of the robot's environment through a histogram grid. Therefore, it places a great emphasis on dealing with uncertainty from sensors and modeling the errors.

The VFH was developed to be computationally efficient, robust, and insensitive to misreading. VFH algorithm is fast and reliable, especially when traversing through densely populated obstacle courses.

The histogram grids in VFH represent the obstacles as shown in Figure 2-3, where an active window is used to update the cells' value on the grid. When an obstacle is detected in a cell by sensors, the algorithm increases the certainty value of this cell. This action is repeated while robot movements. This representation is well suited for inaccurate sensor data, and gives the potential for the fusion of multiple sensor readings.

The VFH algorithm contains three major behaviors: first, it constructs two-dimensional Cartesian histogram grid, as shown in Figure 2-3-b. This grid is updated continuously in real-time with range data sampled by on-board range sensors. A specific area around the robot, the active window, is chosen based on histogram grid, see Figure 2-4-a. The

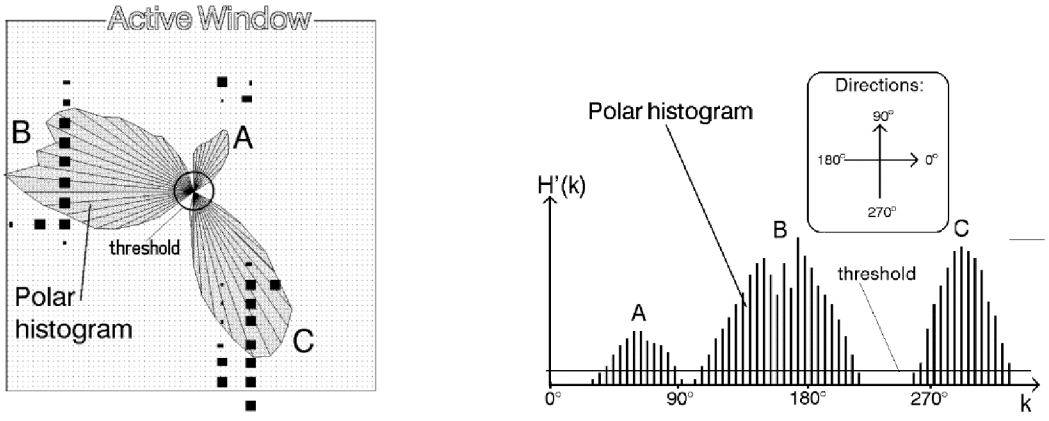
dimension of this area is set to fit the range sensors. Every cell in active window has a value representing the certainty of obstacle existence. The active window is translated when the robot translates.



a b

Figure 2-3: VFH space representation, a: is the actual space. b: the space representation by obstacle certainty value. Source: (Borenstein et al. 1991)

The second behavior constructs one-dimensional polar histogram by reducing the Cartesian histogram around the momentary location of the robot, as shown in Figure 2-4-b. This operation is done using (n) angular sectors. The sectors start from robot location, and the value of each sector is calculated by summing up the cell values in that sector. Figure 2-5 shows the sectors and certainty representation for obstacles.



a b

Figure 2-4: a: Representation of active window and polar sectors, b: one-dimension polar histogram. Source: (Borenstein et al. 1991)

The third behavior chooses the candidate valleys, which are consecutive sectors on polar histogram below a specified threshold; a smoothed polar histogram typically has peaks, and valleys. The peaks represent sectors, which have high polar obstacle density, while the valleys represent sectors that have low polar obstacle density. Any valley has a polar obstacle-density below a certain threshold, is called a candidate valley.

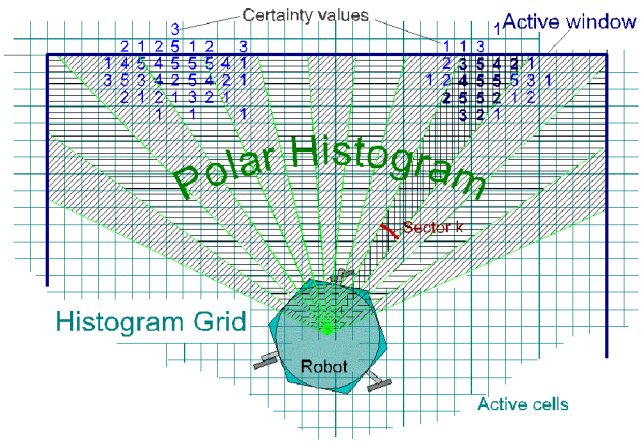


Figure 2-5: Histogram grid and certainty value representation in VFH. Source: (Johann Borenstein 1990)

Once the direction of the selected candidate is determined, the orientation of the robot is steered to match this direction. Sometime a cost function is applied; it takes into account the target direction, wheel orientation, and previous direction, then steer the robot based on the value of this function.

The main limitations of VFH are the navigation through narrow areas, the local minimum, and the number of variables that need an optimization for every workspace. In addition, this algorithm is not complete which means, it cannot guarantee to reach the goal.

2.3.3 Roadmap algorithms

The idea behind the roadmap approaches is to build a map of static workspace and use it for repeated tasks. It would be more efficient to construct a data structure once and reuse it to plan subsequent paths. This data structure is often called a roadmap. The roadmap approaches try to construct a set of one-dimensional curves, which connect two nodes from the free areas in the workspace, and reuse them for further path planning query (Choset, Howie et al. 2005).

The visibility graph is an example of roadmap. It tries to connect the initial and the goal locations with nodes from the map. Then, it searches for a continuous path between these locations, Figure 2-6 shows a visibility graph, the shaded areas represent obstacles, and the solid lines are roadmap curves set (Lulu et al. 2005).

The visibility maps method is applied in workspaces that have polygonal obstacles; however, other obstacles shapes can be approximated. The map's nodes are vertices of the polygons, while the edges are connections between two nodes which are located within line of sight (de Berg et al. 2008, chap. 15). The main shortcomings of visibility maps are 1- the visibility graph works well in two dimensions, but not three or more. 2- The Shortest paths pass through vertices, which consider unsecure in reality because it passes very near to obstacles.

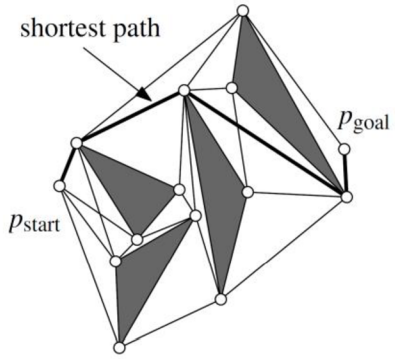


Figure 2-6: Visibility graph – Roadmap. Source: (de Berg et al. 2008)

Another example of the roadmap algorithms is the Voronoi diagram. This method tries to divide the workspace into sub-regions. Each edge of the diagram is constructed using equidistant points from the nearest two points on the obstacles boundaries. Figure 2-7 shows methods to generate the edges of the Voronoi diagram.

In navigation case, the equidistant curves from at least two obstacles are created, and then the path is generated. The algorithm tries to connect the initial location and the goal locations to the curves and then find a path along these curves.

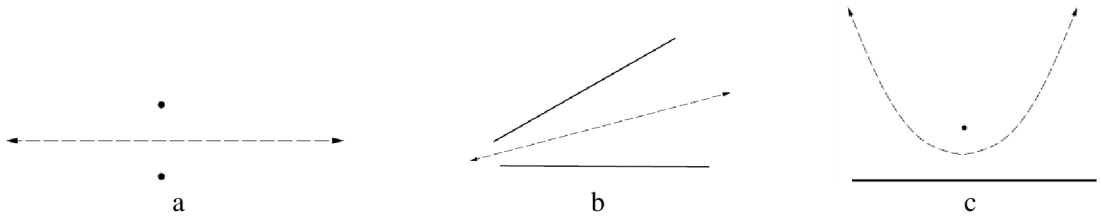


Figure 2-7: Voronoi diagram generation. a: edge between two vertices, b: edge between two lines, c: edge between vertex and line. Source: (LaValle 2006)

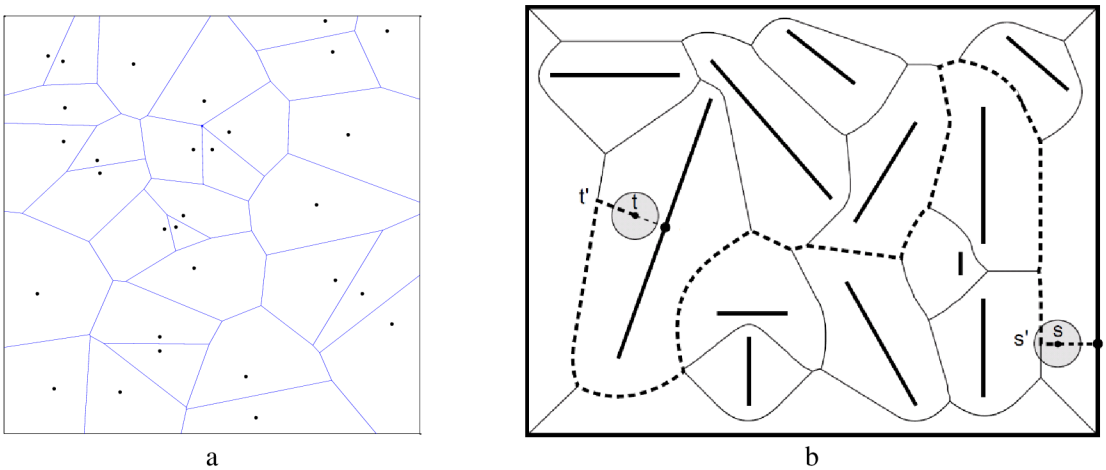


Figure 2-8: a: Voronoi diagram in points-like obstacles, b: disc robot and Voronoi diagram in line shape obstacles. Source: (Aurenhammer et al. 2000)

Figure 2-8-a shows a Voronoi diagram in obstacles-like points, Figure 2-8-b shows a disc-like robot and its path on Voronoi diagram; the robot start from *s* and finds the nearest

point *s'* on Voronoi diagram then plan a path to *t'* which is the nearest point to the target location *t*.

The Voronoi methods are used in motion planning frequently. The advance of these methods is to keep the robot far away from the obstacles (LaValle 2006; Aurenhammer 1991; Aurenhammer et al. 2000; Garrido et al. 2011; Choset et al. 2000; Fabbri et al. 2002; Shkolnik et al. 2009; Sakahara et al. 2008; Masehian et al. 2010; de Berg et al. 2008, chap. 7).

2.3.4 Cell decompositions (CD)

The basic idea behind CD is to decompose the workspace into manageable regions and determine the free ones. The free regions or what-called free cells represent the areas not occupied by the obstacles. The algorithm builds a graph of adjacency for the adjacent free cells and convert the motion planning problem to a graph search problem (LaValle 2006, chap. 6; Choset, Howie et al. 2005, chap. 6; Seda 2007). The cell decomposition approaches and its developments are discussed in more detail in a separate chapter.

2.3.5 Grid-based search

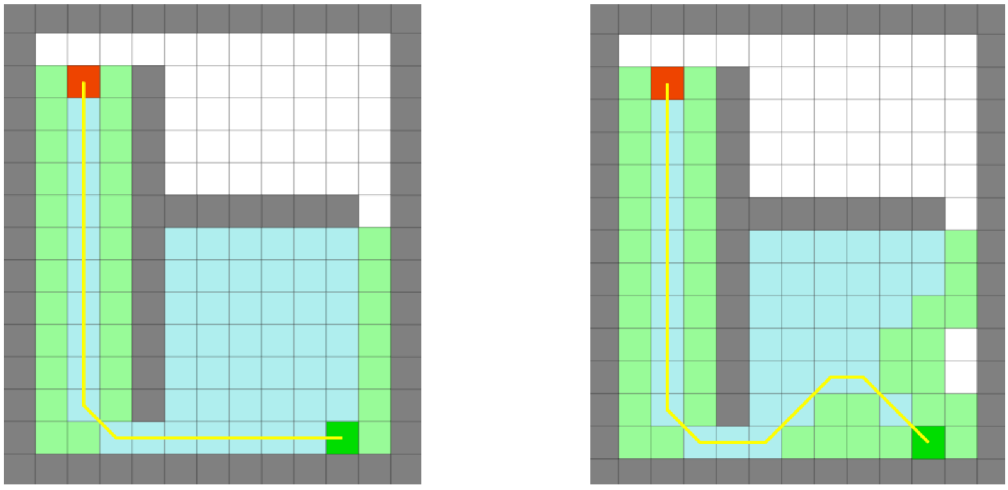
Using cell decomposition techniques or others techniques, the workspace can be represented as a grid of free and occupied areas. This representation can be easily transferred to a graph representation.

Some algorithms, known as grid-based search algorithm used the graph for path finding problems, for example A*, breadth-first search, Dijkstra's algorithm, and greedy best-first search, etc.

The breadth-first search starts from one node and explores the neighbor nodes first, and then it moves to the next level of neighbors if they not explored yet. The Dijkstra's algorithm uses the same principle, but the algorithm revisits the neighbors if they have better path to the start point. These two algorithms do not take into account the cost to the goal, they explore the graph until they find the goal.

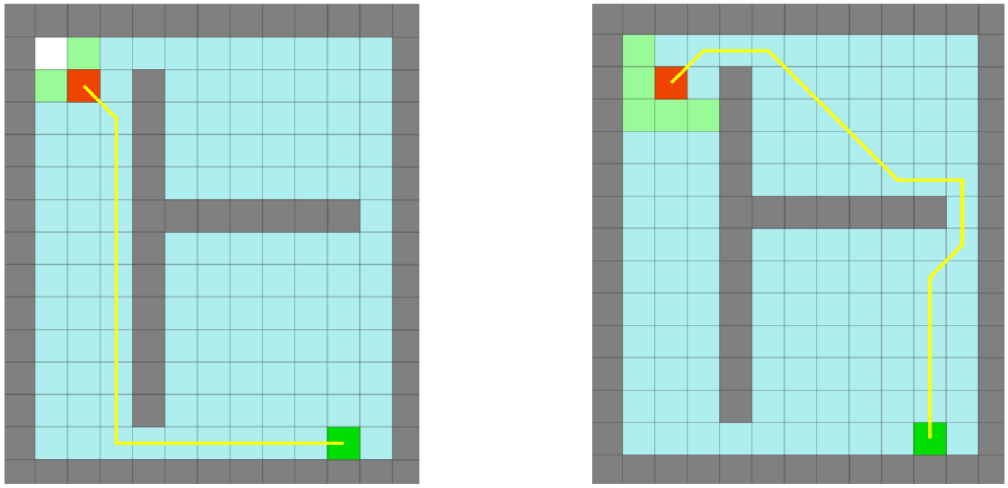
On the second hand, the greedy best-first search algorithm starts exploring the nodes that have the smallest cost to the goal. This principle makes the algorithm faster than the previous algorithms. A* combines the Dijkstra's algorithm and the greedy best first search to build the path (Amit Patel 2014). It uses the actual cost in addition to the estimated cost to the goal, and explores the most promising nodes. The estimated cost to the goal is calculated using a heuristic function that is vary depend on the problem.

Figure 2-9 and Figure 2-10, show visualization for these algorithms using online library1, is shown in, where generated paths, explored area, and in-queue regions are presented as colored boxes.



(a) (b)

Figure 2-9: (a) A* algorithm, (b) best-first search algorithm, the dark boxes represent the obstacles, the blue boxes represent explored nodes, the green boxes represent in-queue nodes



(a) (b)

Figure 2-10: (a) Dijkstra's algorithm, (b) breadth-first search, the dark boxes represent the obstacles, the blue boxes represent explored nodes, the green boxes represent in-queue nodes

2.3.6 Potential fields

Potential field is a local planner method. It is introduced in (Khatib 1985). This method involves modeling the robot as a particle moving under the influence of potential fields. These fields are determined by set of obstacles and the target destination (Arambula Cosío et al. 2004; Hani Alsafadi 2007; Masoud 2013; Mbede et al. 2000; McFetridge et al. 1998; Pêtrès et al. 2012; Sfeir et al. 2011; Zhang et al. 2012). The potential field algorithm is efficient and could be applied in real-time. Since, the motion of a robot, at any moment, is

[1] http://qiao.github.io/PathFinding.js/visual/

determined by the location and the potential fields. It is also a powerful method because it easily extensible, for the reason that, the potential fields are additive, a new obstacle can be added to the workspace by summing up the influence field of this obstacle to the old fields.

This method has a major drawback, which is the local minimum. Because the potential field approach is a local rather than a global method. This problem is overcame by coupling the potential field method with other techniques to escape the local minima (Rosell et al. 2005; Hwang et al. 1992), or constructing potential field functions, which do not contain a local minimum (Siddhartha Srinivasa 2013). The harmonic potential fields is used also to escape the local minima and it has good results (Kim et al. 1991; Masoud 2013).

2.3.7 Sampling-based algorithms

The main drawback of the former methods is the low efficiency in high dimensional problems, which makes the search space extremely large. Sampling based algorithms appeared to confront this problem. The general approach is to approximate the space instead of dealing with it exactly (Lin 2006).

In recent years, a number of sampling-based algorithms for motion planning have been introduced. They have had remarkable success in solving challenging motion planning problems. The fundamental distinction between sampling-based motion planners and earlier planners is the representation of obstacles in the workspace. The earlier planners construct explicit representations of obstacles, which has several disadvantages; e.g. time complexity and PSPACE-hardness (Lindemann et al. 2003).

Sampling-based motion planning has emerged as a way to avoid explicit constructions of obstacles. The sampling-based motion planner restricts the modeling of configuration space. This restriction eliminates many of the problems encountered in the methods that constructed an explicit representation of obstacles. Since there is no explicit model of obstacles, it is not needed to characterize all possible conditions for particular classes of problems.

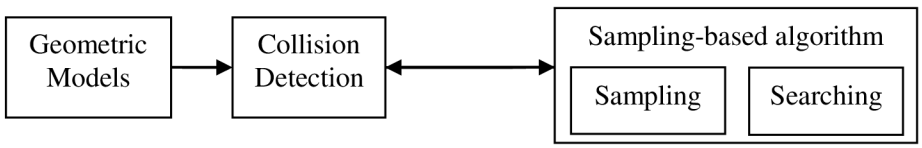


Figure 2-11: The sampling-based philosophy

Sampling-based motion planners can be applied to a broad class of motion planning problems because they treat the collision detection function as a separate module. For these reasons, these kinds of algorithms often seem strikingly simple in comparison to combinatorial motion planners. The simplicity and generality of these planners are the

significant factor contributing to their success and applicability to high-DOF (Lindemann et al. 2003). Figure 2-11 shows the principle of sampling-based approaches, which consider collision detection as a "black box," and separates the algorithms from geometric and kinematic models.

Randomized sampling-based algorithms are a powerful and practically important class of motion planning methods, i.e. randomized path planner (RPP), probabilistic roadmap planners (PRMs), Ariadne's Clew method, and rapidly exploring random Trees (RRTs), etc. Their appeal lies in their ability to address large and complex problems in an incremental fashion (LaValle et al. 2004). However, the price of this incremental approach is a reduction in completeness. Most computational geometry algorithms are algorithmically complete, meaning that they are guaranteed to find a solution to a problem if one exists, or report that none is exist. They are also guaranteed to terminate in finite time. Randomized methods sacrifice algorithmic completeness for weaker probabilistic completeness (Cheng et al. 2002). That is means if a solution is exist, the probability to find this solution is approaches to one, as the number of iterations approaches infinity (Esposito 2013).

In next sections some of randomize samples-based algorithms have been briefly reviewed.

The randomized path planner (RPP) has been proposed in (Barraquand et al. 1991, 1990), it operates as follows: first, the planner defines several potential fields over a grid imposed on the workspace, where the potential value is defined by non-negative, real-valued function. Second, the planner descends the gradient of the potential field, until a local minimum is reached. If the minimum is the global minimum, the goal state has been attained, otherwise, the planner executes a series of random walks with the aim of escaping the local minimum. After this, the planner again descends the potential field gradient, continuing this process until the goal state is reached or the user-time-limit elapsed. This latter condition is necessary because unlike combinatorial planners, sampling-based planners are typically unable to recognize that a problem has no solution; in such a situation, they will never terminate.

The performance of this planner is affected by the good construction of potential fields and a good arbitration function, which decide when to execute a random movement.

Another random planner introduced in (Glavina 1990), it known as the ZZ-method. The algorithm attempts to connect the initial and the goal locations using a straight local planner. If this fails, then a new configuration is chosen as a sub-goal. The planner attempts to connect the new sub-goal to the initial and goal configurations using the same local planner. If this fails, new sub-goals are added and attempts are made to connect them with previously existing sub-goals, as well as the initial and goal configurations. Edges between sub-goals are checked for collisions. A primitive collision detection method has been used which prevents this algorithm from applying in challenging high-DOF problems.

This was remedied in some extensions (Baginski 1996). The ZZ-method contains many elements, which have become common in algorithms that are more recent.

Ariadne's Clew is a single-query algorithm (Mazer et al. 1998); it is designed to find paths in high-dimensional continuous spaces. It is applied to robots with many degrees of freedom in static, as well as dynamic environments. The Ariadne's clew algorithm comprises two sub-algorithms, called SEARCH and EXPLORE, applied in an interleaved manner. EXPLORE builds a representation of the accessible space while SEARCH looks for the target; It grows a tree from the initial configuration toward the goal configuration. At each step, it searches for a new "landmark," reachable from a current landmark by a Manhattan path, which is maximum distant from a point to a set of all landmarks. They use genetic algorithms to search for a solution to this optimization problem. Once a new landmark has been added to the tree, the planner attempts to connect this new landmark to the goal.

Probabilistic Roadmap method (PRM) is one of widely used methods in motion planning. It introduced in (Kavraki et al. 1996) as a planner for holonomic systems. This method proceeds in two phases, a learning phase, and a query phase. In the learning phase, Figure 2-12, a probabilistic roadmap is constructed and stored as a graph. The graph's nodes correspond to collision-free configurations, and the graph's edges correspond to feasible paths between these configurations. These paths are computed using a simple and fast local planner. In the query phase, any given start and goal configurations of the robot are connected to two nodes of the roadmap using the local planner. The roadmap, then searches for a path joining these two nodes using graph searching methods. This approach is general and easy to implement. It can be applied to virtually any type of holonomic robot.

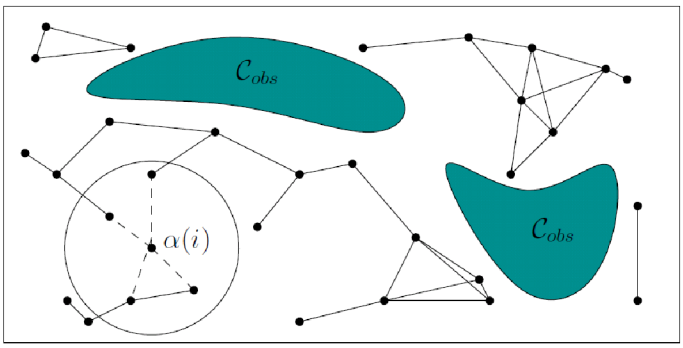


Figure 2-12 : PRM, learning phase, where the planner tries to connect a random sample to nearby vertices in the roadmap. Source: (LaValle 2006)

Rapidly-Exploring Random Tree (RRT) is another very wide used planner in motion planning problems. It is originally proposed for non-holonomic system with dynamic constraint. The RRT algorithm is probabilistic algorithm. It is introduced in (Lavalle 1998) as a planning algorithm to explore quickly high-dimensional spaces. It can handle

holonomic movements and nonholonomic constraints by randomly building a space-filling tree, see Figure 2-13.

The tree is constructed incrementally from samples drawn randomly from the searching space. It is designed for efficient searching in nonconvex obstacle environments. This algorithm has the ability to work under algebraic and differential constraints, and that due to its incremental behavior. The key idea of the RRT is to bias the exploration toward unexplored portions of the space by sampling points from them and "pulling" the search tree toward this regions. The RRT method and related research and developments have been reviewed in more detail in a separate section.

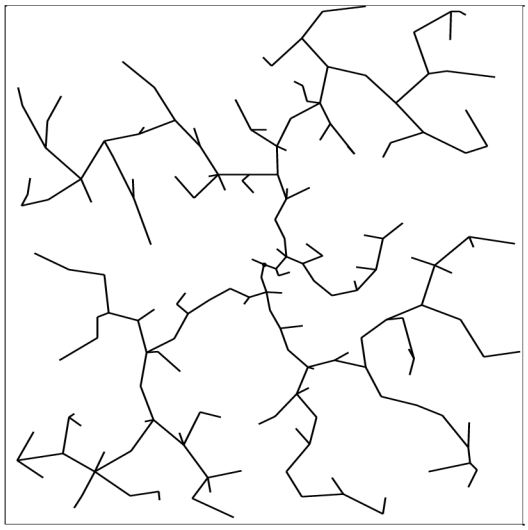


Figure 2-13: Rapidly exploring random tree

The high demand for more efficient and general planners comes out with new methods based on adaptive sampling-based planners. There is no method outperforms all others for all kinds of problems. Rather, each technique has different strengths and weaknesses, which make it best suited for certain types of problems. Some research utilize this idea by building a library of planning methods and use the suitable one based on the workspace characteristics.

For example, in (Morales et al. 2005) the authors proposed an automated framework for feature-sensitive motion planning. Their framework creates a library of roadmap methods. Then, a machine learning approach is used to characterize and partition the C-space into regions, which are well suited to one method of roadmap in the library. After the best-suited method is applied in each region, the resulting roadmaps of every region are combined to form the full roadmap for the entire planning space.

Another strategy, based on unsupervised-learning methods, is proposed to adapt the sampler (Tapia et al. 2009). This strategy models the topology of the problem in a reasonable and efficient manner, and adapts the sampler depending on characteristics of

the problem. The advantage of their method, that, it can be expanded to accept new samplers.

An adaptive RRTs method is proposed to overcome the limitations of RRTs when exploring heterogeneous environments (Denny et al. 2013). The adaptive RRT uses two levels of algorithms to expand the tree. At the first level, groups of expansion methods are expanded, according to the visibility of the node. Second, the cost-sensitive learning approach is used to select a sampler. In addition, the authors proposed a visibility for RRT nodes, which can be computed in an online manner. It is used by adaptive RRT to select an appropriate expansion method.

In the next section, some sampling methods are reviewed. The sampling methodologies have a significant effect on the sampling-based algorithm.

2.3.8 Summary

Based on the state of the art review and our opinion a concise table, Table 2-1, shows the summary of the advantages and disadvantages of some motion-planning algorithm.

Table 2-1: Comparison of motion planning algorithms

Algorithm	Optimal	Complete	advantages	disadvantages
BUG	No	Yes	Super-fast, real-time	Used in 2D, response to sensor noise.
VFH	No	No	Fast, real-time	Navigation through narrow areas, local minimum, 2D.
Visibility graph	Yes	Yes	Repeatable queries	The speed depends on dimension and number of obstacles, path close to obstacles, (2D-3D).
CD	Yes	Yes	Fast	Used for low dimension problems (2D-3D).
Grid-based	Yes	resolution	Fast in 2D	Very slow in high dimension, memory consuming.
Potential field	No	No	Super-fast(2D)	Local minimum, (2D-3D)
PRM	No	Probabilistically	Used in high dimension (2-100’s)	Slow in high dimension.

RRT	No	Probabilistically	Used in high dimension (2-100's)	Medium to fast speed in high dimension

2.4 Sampling strategies

Sampling based planners use sampling strategies to discretize continuous spaces. The sampling methods have a big impact on the efficiency, and the completeness of the planners. In general, sample based planners use uniform or non-uniform distribution.

Uniform distribution samplers choose samples randomly based on a statistically uniform distribution, e.g. axes based grid, pseudorandom number, Poisson disc, jittered grid, Halton sequence, Hammersley sequence, Lattices, Sukharev, and others (LaValle et al. 2004; *Supersampling* 2015). Figure 2-14 shows examples of the generated samples based on some sampling methods. The Voronoi diagram is plotted to increase the visual awareness.

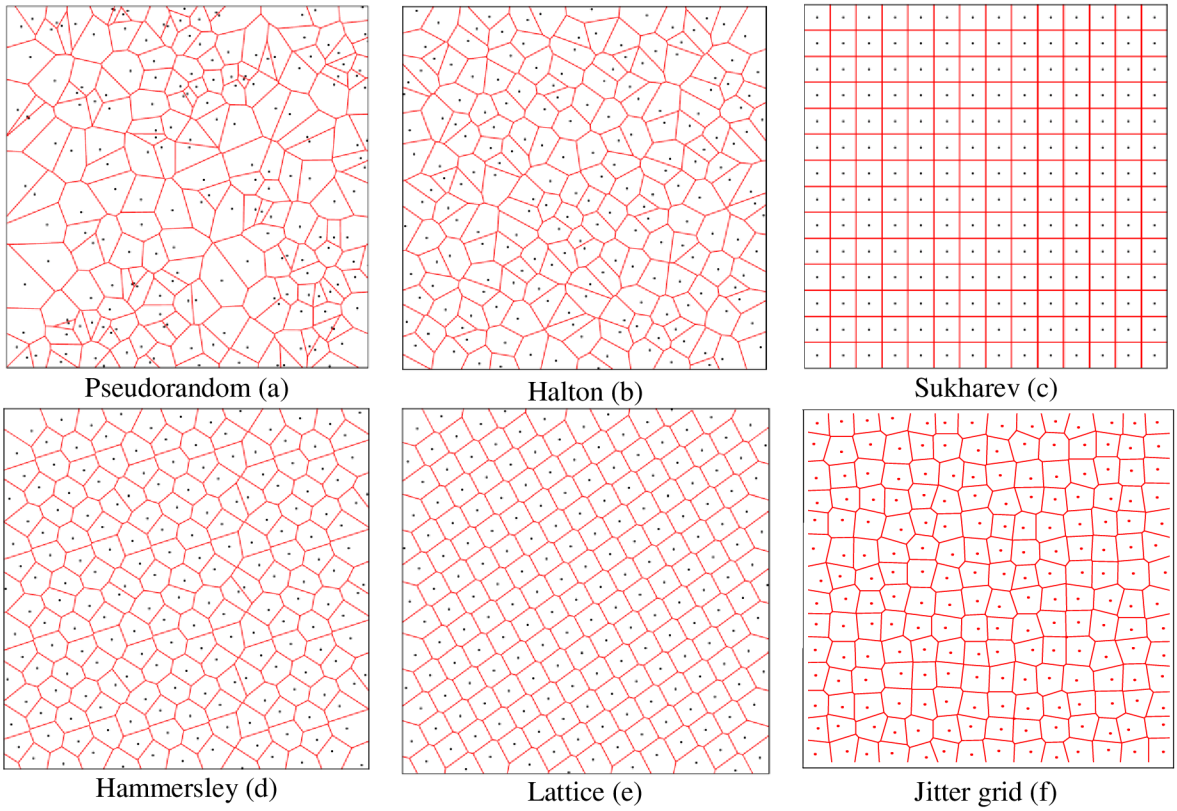


Figure 2-14: Uniform distributions, and Voronoi region for its samples. Source: (a-e) (LaValle et al. 2004)

Examples of uniform methods are the regular structures, and the infinite sequence. The regular structures, e.g. grids have an implicit neighborhood structure; some of these grids have hierarchical or multi-resolution representations, which is preferred feature in motion planning. However, regular structures have a drawback, which is, the necessary numbers of samples to solve the problem is not determined in prior. That is because they are point sets, not point sequences; which mean point sets of a fixed size.

The second type of uniform methods is infinite sequence such as Halton points and uniform pseud-random samples generator. It constructs infinite sequences based on regular structures. This approach enhances the resolution incrementally. Sequences of this type can be considered as point sets periodically, they gradually fill in the gaps between one resolution level and the next one. The generated sequences have incremental quality, which means, after every sample the sequence should be as uniform as possible.

The second category of the sampling strategies uses non-uniform sampling methods. The motivation for this type is to have a higher density in certain regions. The more sampling in important regions helps the planner to be more efficient, (Lindemann et al. 2003; LaValle et al. 2004; Liu et al. 2013; Rodriguez et al. 2006). For example, generating samples around narrow corridors (Lin 2006); sampling about the boundary of obstacles (Amato et al. 1998; Rodriguez et al. 2006); medial-axis sampling (Masehian et al. 2003), in which samples are taken from the medial axis of free configuration space; and Gaussian sampling, in which sampling is biased to be near the C-space obstacles (Lin 2006; Boor et al. 1999). Figure 2-15 shows an ideal classification of importance for the regions of the configuration space.

In general, there are two approaches to non-uniform sampling: the importance sampling methods and the adaptive sampling approaches (LaValle et al. 2004). Importance sampling methods are based on the prior evaluation and assumption about certain areas of C-space. The major drawback of these methods appears when the information about C-space is limited. The other category of non-uniform sampling is adaptive sampling approaches. In these techniques, the sampling strategy is adapted based on the available information, which gained from previous samples.

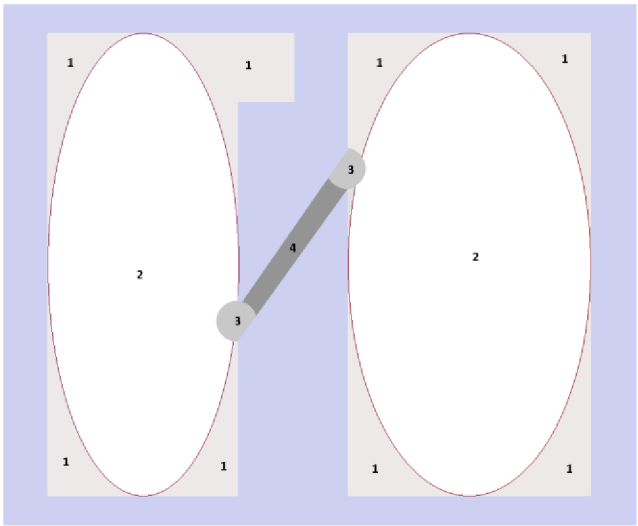
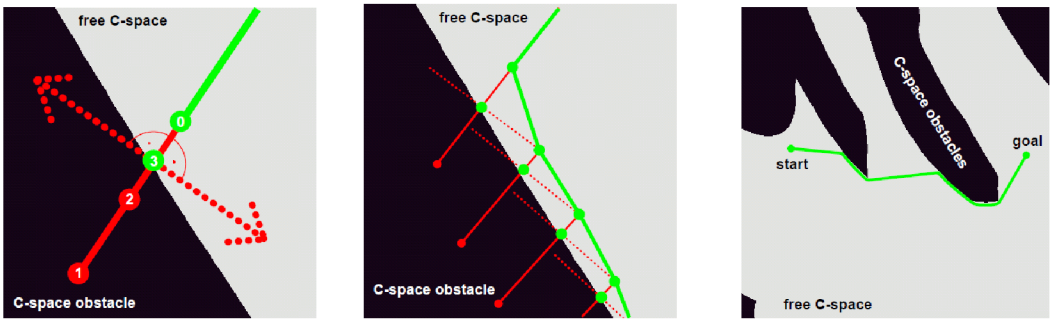


Figure 2-15: The ideal sample distribution, 1: the lowest density of samples in corners and hollows, 2: lower density in free regions, 3: higher density in opening of narrow passage, 4: the highest density in narrow passages.

An example of adaptive sampling is the visibility PRM (Bu et al. 2005; Nissoux et al. 1999). This algorithm adapts its sampler based on the visibility between the connected

regions of the roadmap. Another example is Z^3, which shown in Figure 2-16. If a collision is detected, then it updates the sample location, by translating the sample to free configuration space (Baginski 1996).

In general the most problematic and important regions for sampling based algorithm are the small and narrow ones which known in literatures as a narrow passage problem. Many importance-sampling methods are developed to solve this problem. This problem is discussed later on in a separate section.

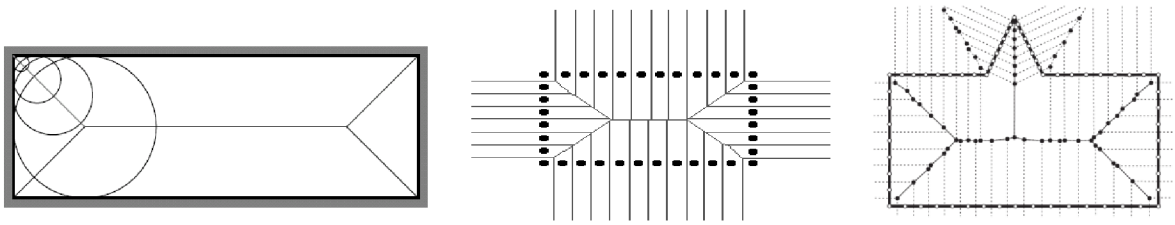


a b c

Figure 2-16: Z3 method for adapting and translation the samples to the free configuration space. Source (Baginski 1996)

An example of importance sampling methods is the medial axis sampling. It tries to retract the samples onto the medial axis of the free workspace and force them to be as far from the boundaries as possible (Masehian et al. 2003; Fabbri et al. 2002; LaValle 2006; Smogavec et al. 2012; Wilmarth et al. 1999). Medial axis sampler gives a slightly higher probability of picking samples from small free areas that is preferred feature in motion planning. An example of a medial axis creation is shown in Figure 2-17-a. It uses a geometric method to build the medial curve for rectangular shape.

On the other hand, the approximation methods are usually used in implicit representation of the free configuration-space. Figure 2-17-b,c shows the medial axis approximation curve. The principle of approximation methods is to generate a configuration randomly (valid or invalid), then it is pushed towards the medial axis of the free space.

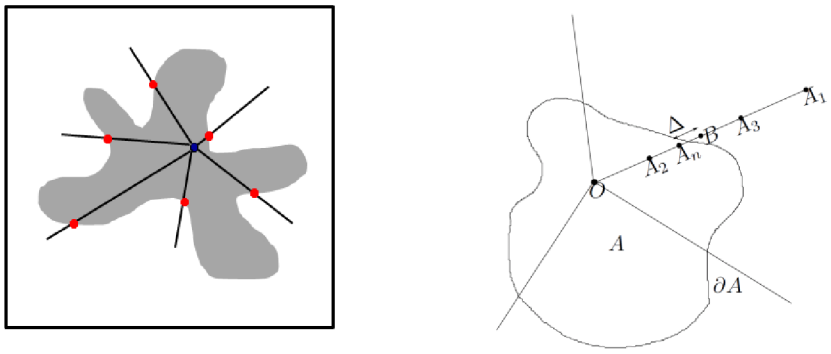


a b c

Figure 2-17: Medial axis and its approximation, a: medial axis methods generated using geometric cycles, b-c: approximation of Medial axis. Source: (Fabbri et al. 2002; Masehian et al. 2003)

Another approach of importance sampling methods is the obstacle boundaries sampling. Rather than waste samples in large areas of free configuration space, it focuses on the

obstacle boundary as important regions (LaValle 2006; Rodriguez et al. 2006; Denny et al. 2011; Yeh et al. 2012). For example OBPRM (Amato et al. 1998), It generates configurations close to the obstacle surfaces, as shown in Figure 2-18-a. First, it finds a configuration in obstacle configuration space. Second, it pushed that configuration out of the obstacles. This pushing is implemented by creating a ray from this collided configuration toward a randomly chosen free configuration. After that a bisection search is performed, it is terminated when a free and near to boundary configuration is found. Figure 2-18-b shows the bisection for randomly chosen ray. The boundary points are retained as nodes in the roadmap.



a b

Figure 2-18: OBPRM: generating samples near obstacles boundaries. Source: a (Yeh et al. 2012), b (Titas Bera et al. 2014)

The Gaussian sampling strategy is another method similar to sampling on obstacle boundaries. The goal of this method is to obtain points near to obstacle configuration space by using a Gaussian distribution. It biases the samples to be closer to free configuration space (Lin 2006; Boor et al. 1999). Figure 2-19 shows samples generated by Gaussian method.

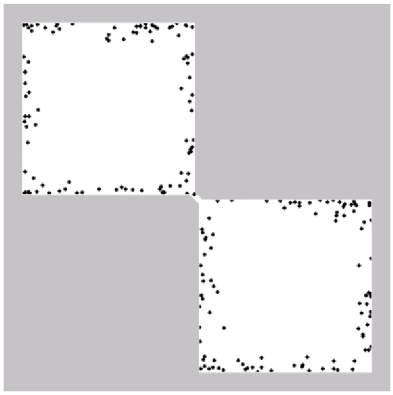


Figure 2-19: The samples generated by the Gaussian sampler. Source: (Hsu et al. 2003)

The principle of this approach is to choose a free configuration randomly. This free sample is treated as the mean (μ) of a Gaussian distribution, and then it is generated another sample with a variance specified by the user. If one configuration in free space and the other is in obstacle space, then the free one is saved as a milestone, otherwise both

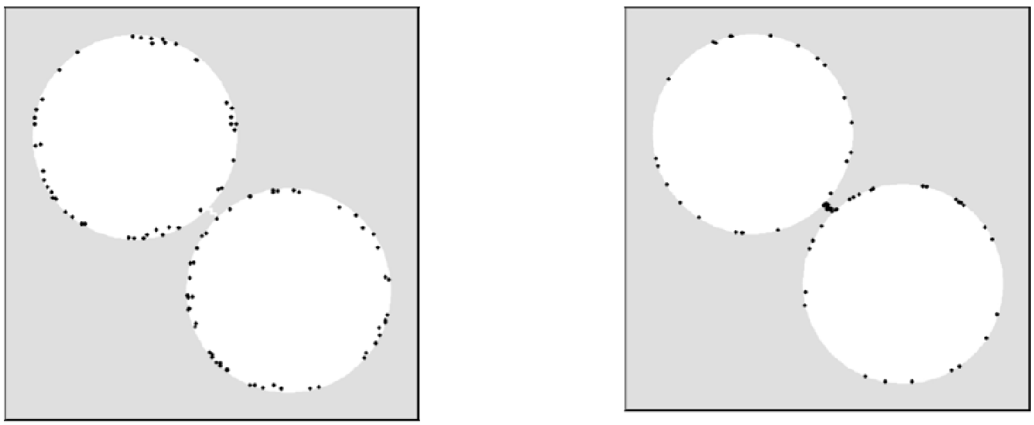
configurations are discarded. Gaussian sampling is not efficient, because it is not easy to get nodes in different spaces, and many attempts are required to generate samples. In addition, the variance optimization is important, where, if the variance is too small, the configurations will be too close to the obstacles, in the opposite, if the variance is large, the configurations will be far away from the obstacles.

2.5 Narrow Passages

Narrow passage problem is a common problem for probabilistic planning algorithms. It occurs when a uniform distribution is used, due to the small volumes of narrow passages areas. Since, the small volume reduces the sampling probability. The uniform distribution does not work well when the dispersion of the samples is higher than the narrow passage volumes. The problem has a bigger impact on RRT planners than in the other methods. In RRT case, the algorithm throws away the valuable sample if the active tree could not connect with it. While in PRM planner case, the algorithm saves the rare and valuable samples, which fall inside a narrow passage. When the number of samples and graph connectivity are increased, these samples soon will be connected to the PRM graph.

Many researches focus on narrow areas identification in order to enhance the efficiency of sample-based planners in narrow workspaces. They try to increase the samplers' ability to take samples from these important areas. In the next paragraphs, some of these approaches are discussed.

The Gaussian Sampler locates points in the neighbor of obstacles surface. This helps to obtain substantial points in narrow space, based on the idea that the narrow passage exists between obstacles. Generally, this method improves the efficiency of planners (Boor et al. 1999; Lin 2006). But is still has some difficulties to plan a path through narrow passage, where many points near the obstacle boundaries lie far away from narrow passages. Figure 2-20-a, shows this limitation.



(a) (b)

Figure 2-20: (a) The Gaussian sampler, vs (b) the randomized bridge builder. Source: (Sun et al. 2005)

Another non-uniform sampling method has been proposed in (Hsu et al. 2003), to increase samples in narrow areas. The Bridge-Test method or randomized bridge builder (RBB) assumes that the narrow areas appear between obstacles; therefore, it randomly takes two configurations in obstacle space and tests the middle points between them. If a middle point is located in the free space, the algorithm keeps it. Figure 2-21 shows the principle of this method. This process attempts to bridge the gap and generate configurations in a narrow passage. After that, the algorithm takes these middle points as milestones to build PRM. Most of milestones by this method distribute within the narrow space, as shown in Figure 2-20-b. Nevertheless, there is still recognized present of these samples lying in the corners and hollows of the obstacles (Lin 2006; Sun et al. 2005; Jiandong et al. 2011). In addition, this method requires a long time to cover the narrow areas. It may fail many times before successfully bridging a gap. Because it needs, a sequence of three nodes such that the endpoints are in obstacle space and the midpoint is in free space.

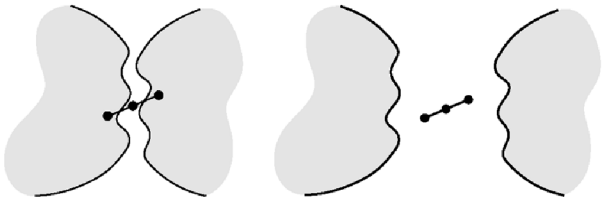


Figure 2-21: Bridge-test, the points that lie in narrow passage can pass the bridge test. Source: (Hsu et al. 2003)

Later on, a hybrid sampling strategy using RBB and uniform Sampler with a certain ratio is presented in (Sun et al. 2005) to spread some samples in free large regions. The bridge test and uniform sampling complement each other. Bridge test reduces samples density in unimportant parts of a configuration space, and increased sample density in narrow passages, While uniform sampler take sample form large free space. The two sampling strategies are combined to construct the hybrid sampling strategy spread samples in important regions.

In similar principle, a hybrid sampling strategy, which uses uniform sampler and randomized star builder (RSB), is presented in (Jiandong et al. 2011). The RSB is used to identify narrow passages in the workspace and to increase the samples density in these areas.

The Random star builder (RSB) is an improved version of RBB. It depends on more than two points lie in obstacles to build the bridge. It is designed to boost the sample density in narrow passage regions, while avoiding sampling milestones in blind corners and dead ends of obstacles, Figure 2-22 shows RSB in 2D workspace.

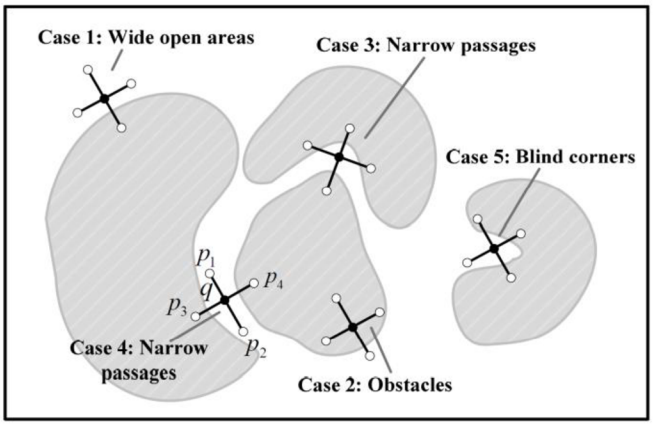


Figure 2-22: Randomized star builder in two-dimensional environment. Source: (Li et al. 2012)

Improved bridge test algorithm was employed in (Wang et al. 2010) to identify the important milestones in narrow passages. The algorithm establishes multiple trees from these samples to explore the sub-regions, which are difficult to reach. The probabilities of selecting a tree for expansion are updated on-line by learning algorithm based on the historic results of the exploration.

Quad decomposition approximation is used in (van den Berg 2005) to find an important area in the workspace, each cell is given a label and weight, depending on the size and its neighbors' size. The weight reflects the sampling probability in that cell.

The adaptive watershed algorithm is used to distinguish between the cells in narrow passages and the cells near a boundary in an open area. The watershed is originally a method for image segmentation from the image-processing field. It separates the open regions from each other by watersheds, see Figure 2-23-a,b. The watershed regions in their work represent the narrow passages.

a b c

Figure 2-23: Identify narrow passage using cell decomposition and watershed algorithm, a: narrow passage identified by adaptive watershed, b: watershed algorithm the dark area represents a free area in the sense, c: grow watershed algorithm enhance narrow passage representation. Source: (van den Berg 2005)

For a better representation of the narrow passage in the corridor and long narrow area, the grow watershed algorithm is presented to grow a watershed to cover the narrow area, as shown in Figure 2-23-c.

Triple-RRT algorithm is proposed in (Zhong et al. 2012). It uses a random star builder (RSB) to identify the configuration in the narrow passage, and then expands RRT tree. This method improves the local sampling density in the narrow passage. The triple-RRTs approach employs two trees as bi-directional expansion, one tree from the initial position and the other from the goal position, and third one is grown in the narrow passage. The three trees have the same expansion chance, which ensures that this method will find a solution quickly between start and goal location, no matter if the path should pass narrow passage or not.

Small-step retraction method presented in (Saha et al. 2005) to help PRM planners find paths through narrow passages, they suggest to fattening the robot's free space by minimizing the shapes of the obstacles. Then build a PRM and repair colliding portions of this roadmap by retracting them out of collision.

Toggle PRM methodology is introduced in (Denny et al. 2011). It simultaneously builds a graph structure for both free and obstacle spaces. These graphs use the information about collision to generate samples, which are used later to generate other samples within the narrow passage. Figure 2-24 shows the principle of this technique. If a sample s chosen randomly from free space, it is added to free graph's node. The PRM tries to connect this free sample to the nearest nodes in the graph. If a collision with obstacles happened during this connection, the collision points are stored as nodes in the obstacle graph. Later on, the graphs are toggled, and PRM tries to connect the nearest obstacle graph's nodes to these new collision points (x_1, x_2 in Figure 2-24). During this connection, another point could be generated in the free area because of collision with free space, and repeatedly, this point will generate other samples and so on.

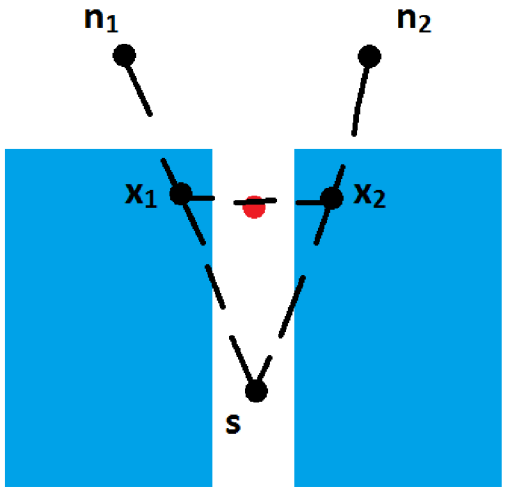


Figure 2-24: Toggle PRM principle. Source: (Denny et al. 2011)

Importance sampling method is introduced in (Rosell et al. 2011). It uses principal component analysis (PCA) to focalize the region where to sample in order to increase the probability of finding collision-free configurations. The proposal is illustrated in 2D

configuration space with a narrow passage as shown in Figure 2-25. The PCA is a statistical technique used to process a set of vectorial samples. It analyses a samples set and returns an hyper-box centered at the mean value of these samples, and the length of each side equal to two times of the data deviation in the corresponding axis. In each iteration the algorithm, chooses a number of samples randomly form the workspace, and identifies the free ones, in addition, it chooses number of samples form hyper-box, and applies the PCA to the new samples set to find the free samples trend.

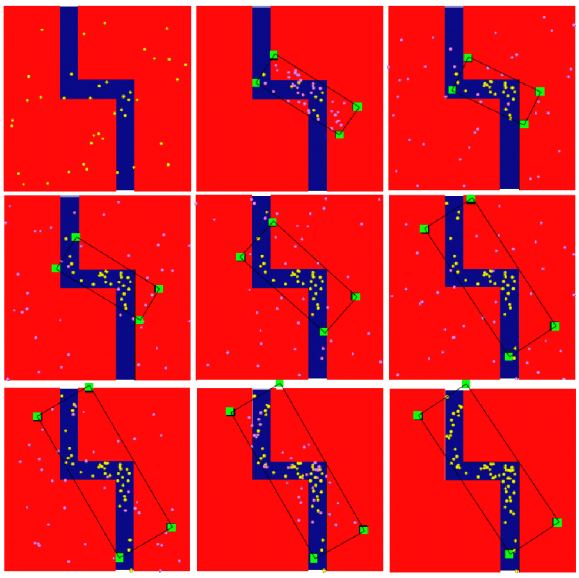


Figure 2-25: PCA sampling procedure. Source: (Rosell et al. 2011)

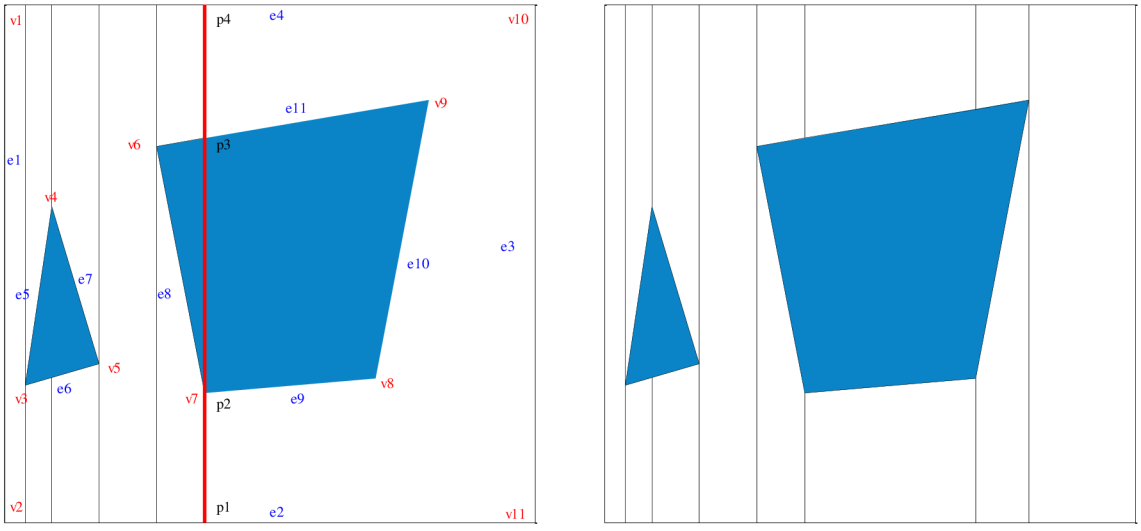
In the next chapter, the cell-decomposition is reviewed and we introduce our contributions based on this approach.

3 CELL DECOMPOSITION

Cell decomposition algorithms (CD) extracts the obstacles and the free regions, and build a graph of adjacency for the free ones (LaValle 2006, chap. 6; Milos Seda 2007). The idea of dividing the space into manageable sections is presented in many researches. In general, the cell decomposition algorithms are classified into two categories; the exact cell decomposition methods and the approximation methods (Latombe 1991).

The first category uses geometric algorithms to determine the free areas and build free cells explicitly (Brooks et al. 1985; Schwartz et al. 1983) . The union of all generated cells is exactly equal to the free space. However, finding exact free cells is not an easy task, especially in higher dimension spaces, that leads to the second category, which uses the approximation techniques to divide the workspace, e.g. quad-tree, octree division, and voxel grid, etc., (Sleumer et al. 1999; de Berg et al. 2008).

In motion planning applications, The CD is utilized by dividing the free robot's workspace into smaller regions called cells. Then it builds a connectivity graph according to the adjacency relationships between the free cells. The graph's nodes represent the cells, while the graph's edges represent the adjacency relations between these cells. From this connectivity graph, a continuous path can be found by following the adjacent free cells.



(a) (b)

Figure 3-1: Trapezoidal cell decomposition principle. a: The sweep line technique. b: Trapezoidal cell decomposition algorithm results, the bold-red line represents the sweeping line, *v* represent the obstacles vertices, *e* represents the obstacles edges, *p* represent the intersecting points.

3.1 Exact cell decomposition

An example of exact cell decomposition is the trapezoidal decomposition method or vertical cell decomposition. It decomposes the free space into trapezoidal and triangular cells. This method draws parallel vertical segments from each polygon's vertex in the workspace to the exterior boundary of the workspace. The regions, which are surrounded

by these segments and the boundaries of obstacles, construct the cells. These cells form the nodes of connectivity graph. The adjacent regions in the workspace are linked together by the graph's edges in the connectivity graph (Abbadi, Matousek 2014; Abbadi, Matousek, et al. 2014). The path in this graph corresponds to the sequence of striped free cells. Figure 3-1, shows the principle of this method.

To model this process in 2D workspace, the workspace X is divided into a free space X_{free}, and an obstacle space X_{obst}.

A set of all vertices (V) are ordered based on the x-axis. It contains the obstacles vertices in addition to the workspace boundaries vertices. Obstacles boundaries (segments) and the outer workspace contour are grouped in obstacle segment set E.

$V = \{v_1, v_2, \ldots, v_i, \ldots\} \rightarrow R^2 : v_i(x) \leq v_{i+1}(x), i \in N^+$

$E = \{e_1, e_2, \ldots, e_i, \ldots\} \rightarrow V^2 : , i \in N^+$. Where v_i represents a vertex has the index i in the points set. e_i represents a segment has an index i in the segments set.

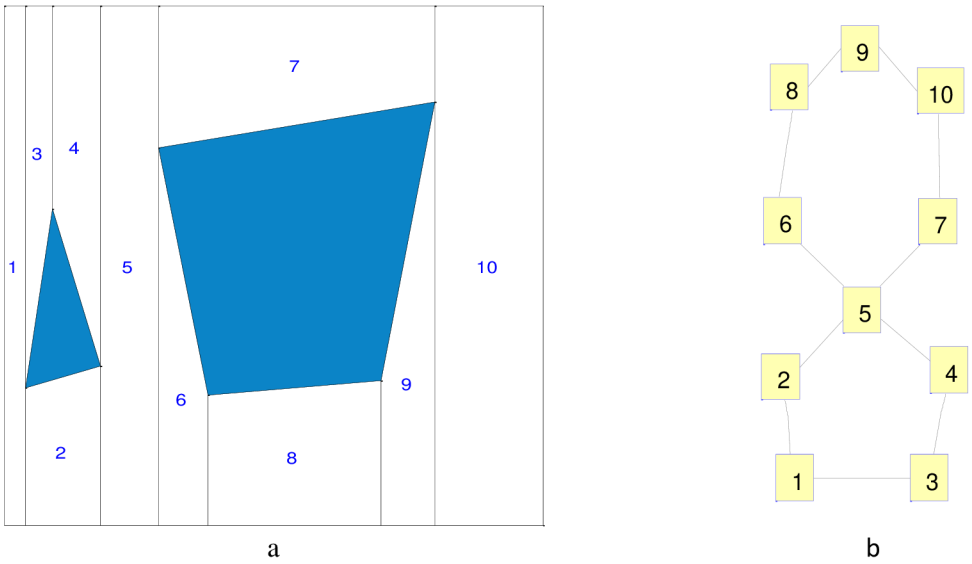
```
Algorithm: Trapezoidal cell decomposition
Input: V   Vertices set of obstacles and workspace.
       E   Edges set of obstacles and workspace.
       Xobst Obstacles workspace.
output:CD  Adjacency graph
  1.   Pvisited = ∅
  2.   V = sortX(V)
  3.   FOREACH v ∈ V BEGIN
  4.       Pintersects = VerticalIntersections( v , E )
  5.  #find vertices on the same obstacle edges.
  6.       Vneighbors = onSameObstEdge(Pvisited, Pintersects)
  7.       cells = constructCells(Vneighbors, Pintersects)
  8.       IF (cells ∉ Xobst) BEGIN
  9.           CD.addNode(cells)
  10.          CD.findAdjacency(cells)
  11.      END
  12.      Pvisited.remove(Vneighbors)
  13.      Pvisited.add(Pintersects)
  14.  END
```

Figure 3-2: Trapezoidal cell decomposition algorithm

The trapezoidal cell decomposition algorithm is shown in Figure 3-2, where $P_{visited}$ is a set of all intersection points and vertices, which are visited before. $P_{intersect}$ is a set of intersection points with the current sweeping line that is established from the vertex v.

Figure 3-1-a, shows the sweep line and the intersection point, which is denoted as p_i. $V_{neighbor}$ variable is a set of vertices in $P_{visited}$, which fall on the same segment e with one element of the $P_{intersect}$. The *Cells* variable represents the new generated cells, and *CD* is the output graph, Figure 3-1-b shows the generated regions.

For more refinement, a post-process function is executed to merge the adjacent cells, which has edges parallel to each other. Figure 3-3 shows the result of this post-process function and the corresponding graph.



a b

Figure 3-3: Trapezoidal cell decomposition. a: Post-process to merge cells, b: the generated adjacency graph

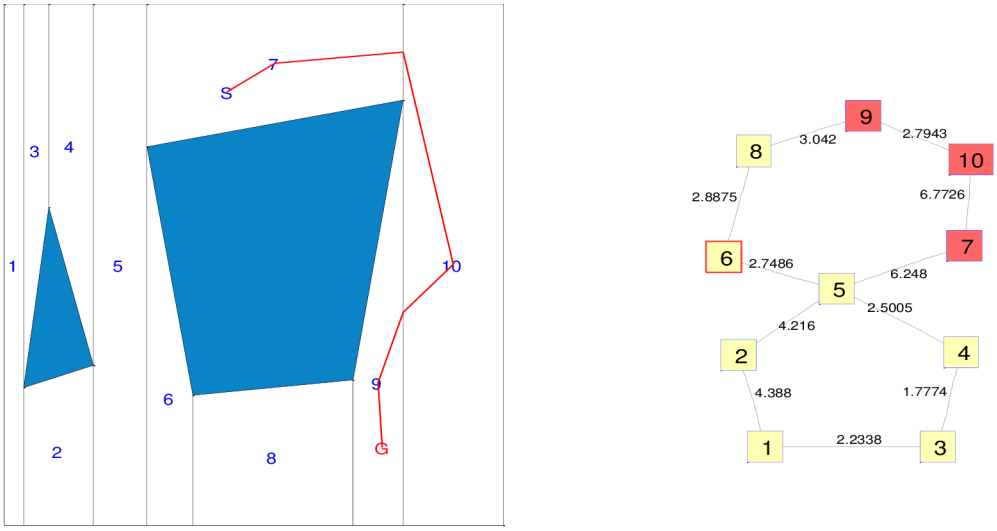
The algorithm's output is a graph, which represents free areas. This method converts the problem of navigation and path planning into a graph search problem. For example, when a plan is required between two positions, the cells that contain these locations are determined, and then a graph search is executed in order to find a path.

The transformation of motion planning problems in spatial environments into a graph search problem gives many advantages. An efficient methods can be used to find a path, e.g. A*, Dijkstra, etc. The spatial information about the cells is exploited to optimize the generated path, e.g. the shortest path can be found based on the distance between the cells.

Figure 3-4-a, shows the principle of cell decomposition planner, where the line represents the path through the free cells. Figure 3-4-b shows the graph of adjacency and the edges' weight. The shaded nodes correspond to the path's nodes from the start cell to the goal cell. In this example, the weighs correspond to the distance between cells' central and barriers' midpoints between the cells.

When a planning query is established, the planner finds the start and the goal cells, then it searches for a path between these two cells, if a path is found, the planner connects the start and the goal locations through the free cells on that path.

Another example of exact cell decomposition is the decomposition based on obstacles edges. This method considers each edge like a line. Then, it finds the intersections with other edges or cells, and builds the free cells in the free space based on these intersections (Sleumer et al. 1999).



a b

Figure 3-4: Path planning using trapezoidal cell decomposition. a: The generated vertical free cells. b: The graph of adjacency which corresponding to paths between cells

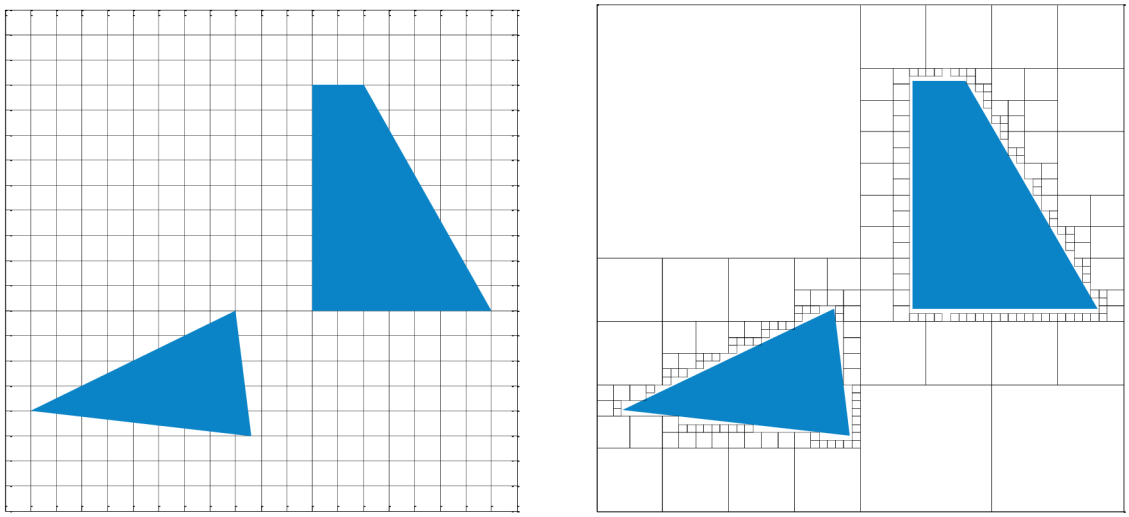
3.2 Cell decomposition approximation

Due to geometric calculations, the high computational demand of previous approaches, and the hard to implement for high dimension workspaces, the approximation methods to the CD were proposed. The most forward method for approximation is the voxel grid, as shown in Figure 3-5-a. It excludes the cells on the obstacle areas and builds a graph of adjacency for the cells in the free area. This method is efficient for low dimensional spaces. However, it generates a large number of cells. This method is resolution complete; which means the algorithm completeness depends on the grid's fine (Sleumer et al. 1999; de Berg et al. 2008).

Quad-tree decomposition is another approach for cell decomposition approximation. This approach uses a recursive method to subdivide the cells until one of the following conditions is satisfied. 1- Each cell lies completely either in a free space or in an obstacle region. 2- or, an arbitrary limit of a resolution is reached.

Once a cell fulfills one of these criteria, it stops decomposing. After decomposition steps, the free path is found by following the adjacent free cells (Katevas et al. 1998). This method is used in 2D (de Berg et al. 2008, chap. 14). Figure 3-5-b, shows the generated cells of this method. In a similar way, the Octree method approximates the decomposition in 3D spaces (Choi et al. 2011).

The quad-tree and octree methods are resolution complete. They can work efficiently for low dimensions workspaces, three or less (van den Berg 2005).



a b

Figure 3-5: Cell decomposition approximation. a: Voxel approximation methods. b: Quad-tree approximation methods

3.3 Contributions, Tests and Results

Our contributions using cell decomposition approximation are described in this section. These methods are exploited to solve the problem of safe paths planning in stationary workspace. In addition, they are used with minimum spanning tree to identify the important regions in the workspace. The other work, which combined the exact cell composition approaches with other motion planner, is described later in the next chapters.

This section is divided into two sub-chapters, each one present the problem formulation and our methodology to solve it.

3.3.1 Safe path planning using cell decomposition approximation

In this work (Abbadi, Prenosil 2015a), The cell-decomposition approximation is used to find a safe path in static workspace, for omnidirectional robot. The quad-tree approximation algorithm divides the workspace into manageable free areas, and builds a graph of adjacency between them.

New methods have been proposed to keep the robot far away from the obstacle boundaries by a minimum safety-distance. They utilize the size of free cells to generate the desire path, i.e. they give a lower cost to the graph's connection between big free cells, and a higher cost to the connections between the smaller cells. After that, the planner searches for a path that has the lowest cost.

The shortest path is not our focus in this work, however a tradeoff between the shortest path cost and the safe path cost is considered when choosing the weight values.

Proposed Methods

In this work, the path safety problem in static workspace is studied. The path is considered as safe if 1- It passes through obstacles without colliding with them. 2- It navigates and keeps a safety distance far from obstacles boundaries. 3- It follows the large open areas in the workspace when it is possible.

We utilize the cell-decomposition approximation algorithm (ACD) to find an approximation of the free areas, and exploit the resolution feature to satisfy the minimum distance condition. The resolution of ACD corresponds to the smallest cell's edge. We proposed that the robot passes through the center of the cell when it executes the path; based on that assumption the resolution is chosen to be (*2 * safety distance*).

Three versions have been proposed to plan a safe path. These methods are based on the manipulating of the weights, which assign to the graph edges, in order to make the planner choose the largest cells when translating toward the goal position.

The first approach uses equal weights for translating from one cell to another. The idea behind this proposal is to minimize the total number of cells in the path, which in consequence directs the planner to use bigger cells, when searching for a lower path cost.

The Second method introduces a penalty for translation between different cells size. This penalty is added to the edge's weight, and it is disproportional to the cells size, which means the weight of translating between the larger cells is smaller than the weight of translating between the small cells, while the weight of translating between the same size cells is kept fixed. This proposal guides the planner to do the translating in large cells when it is possible and at the same time keeping some trade-off between making the translation in large cells, and planning a path closer in length to the shortest path.

The last proposed method is very similar to the second approach in spite of it introduces disproportional penalty not only with different cells size, but also with cells that have the same size. The benefit of these methods is to push the path toward large cells when it is possible by adding more penalties when translating between small cells, in addition to the benefits of the second approach.

The proposed methods, lead the planners to use the large cells more than small cells for planning a path, at the same time they keep the safe distance far from obstacles.

Result and Discussion

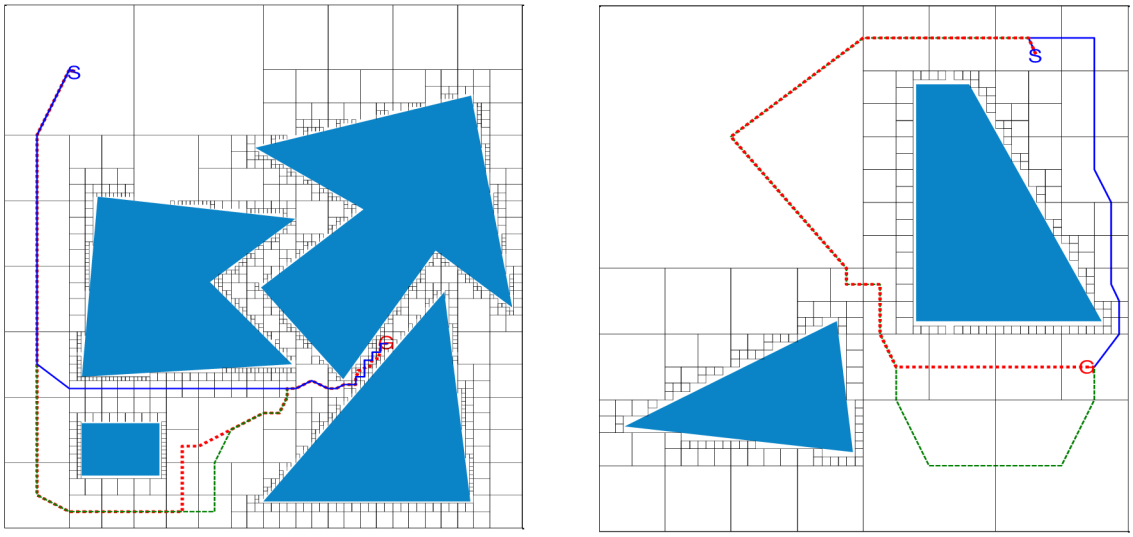
In the first proposed method, the weights of the graph's edges are uniformed to the cost of (1) unit, which corresponding to the cost of translating from one cell to another one, regardless of the cells' size.

In the second proposed method, we associate to each cell of the free cells a level. This level is disproportional to cell size. The level is used when manipulating the weights of

graph edges. The edges' weight between two cells is set to be equal to the biggest level value between these cells, i.e. if cell1 has a level of (2), and cell2 is smaller and has the level of (4), The edge's weight between them has the value of *max*(2,4) which is (4). The translation between cells that have the same level is fixed to the weight of (1).

The weights in the third proposed method are calculated in the same way as in the second method, but here the transition between same cells size is varied also based on cell's level. For example, the translation's weight between the cells that have levels of (3) will take the value of (3).

The Dijkstra's algorithm is used as a graph search algorithm to find the path over the graph. The Dijkstra's algorithm finds the minimal cost of the path efficiently. The tests are done in two workspaces using three values of safety distance {0.1, 0.3, and 0.75}. The results are shown in Figure 3-6, Figure 3-7, and Figure 3-8, respectively.



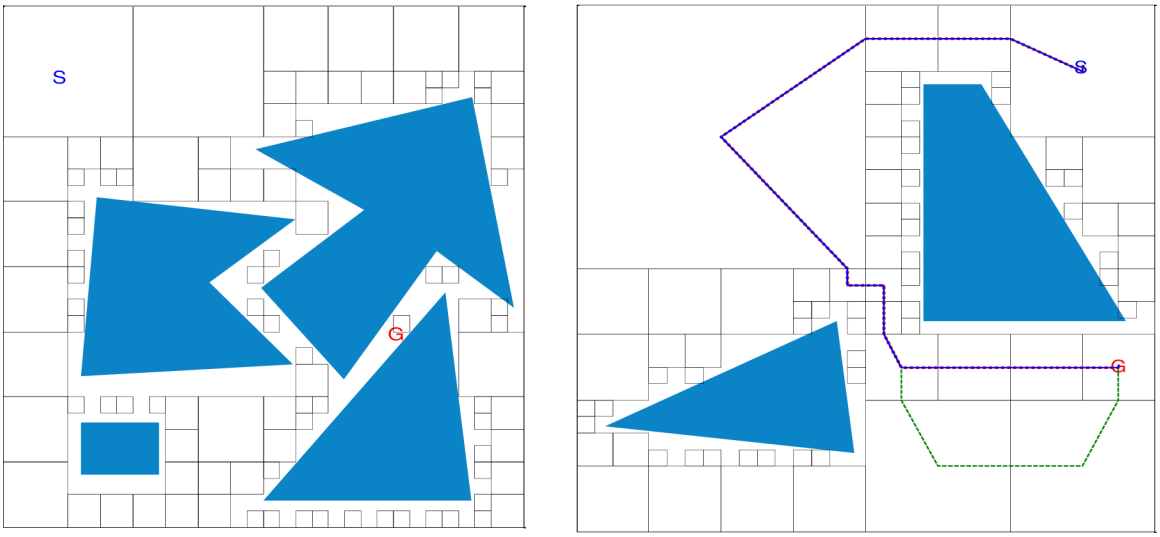
a b

Figure 3-6: Safe paths planning, the safety distance is equal to (0.1), a, b: the testing workspaces. The solid blue line represents the equal weights of translation method, the doted-red line represents the disproportional penalty to translating between different cells size, and the dashed-green line represents the disproportional penalty to the size of the cells method. *S* and *G* is the initial and the goal positions

We can infer from the results that the proposed methods generate a path that respects the safety distance condition. The first method (the solid blue line) tries to minimize the number of cells as shown in Figure 3-6-a, b. The path keeps the safety distance, but it does not follow the large areas. The second method (the dotted red line) is better in this criteria; it forces the planner to plan in the large cells. However, it follows the large cells, but not if smaller cells are adjacent to each other; in that case the algorithm plan through these adjacent cells. The last approaches solve this drawback (the dashed green line), and it plans in large open regions when it is possible.

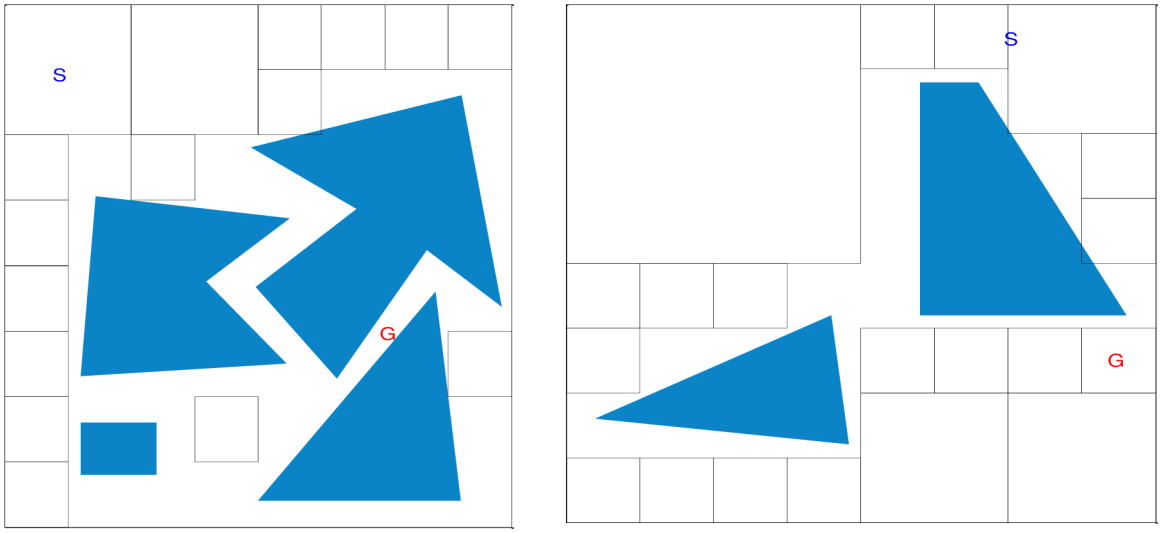
Figure 3-7-a, shows the unreachable path because of the safety distance. The same case in Figure 3-8-a,b. That because the algorithm excludes the collided cells in obstacles,

which break the continuity of the graph's edges. When a path searching is executed, the search algorithm cannot find a route between the initial cell and the goal cell.



(a) (b)

Figure 3-7: Safe path planning, (a, b) are the testing workspaces. Safety distance is equal to (0.3), the goal in (a) is unreachable. The first (-) and second(…) methods in b plan the same paths



(a) (b)

Figure 3-8: Safe path planning, (a, b) are the testing workspaces, the safety distance is equal to (0.75), the goal is unreachable in both workspaces

Summary

In this work the ACD planner is used to find safe path for the robot; the quad-tree approximation algorithm divides the workspace into manageable free areas, and builds a graph of adjacency between them. Three approaches have been proposed to plan a safe path. These methods manipulate the edges' weights in order to make the planners choose the largest cells when translates toward the goal position. And at the same time keeps the robot far from obstacles by a safety distance. The proposed methods show the ability to plan the desire path.

3.3.2 Narrow passage identification using CD approximation and minimum spanning tree

Narrow passage problem is a problematic issue facing sampling-based motion planners. In this work (Abbadi, Matousek, et al. 2015), a new approach for narrow areas identification is proposed. The quad-tree cell-decomposition approximation is used to divide the free workspace into smaller cells, and build a graph of adjacency for them. The proposed method follows the graph edges and finds a sequence of cells, which have the same size, preceded and followed by a bigger cell size. The sequence, which has the pattern "bigger-smaller-bigger" cells size, is more likely to be located in a narrow area. The minimum spanning tree algorithm is used, to linearize adjacency graph. Many methods have been proposed to manipulate the edges cost in the graph, in order to make the generated spanning tree traverse through narrow passages in detectable ways. Five methods have been proposed, some of them give bad results, and the others give better one in simulations.

Proposed methods

Narrow passage problem faces most of sampling based approaches. The problem occurs when a uniform distribution is used to take samples form the workspace, because the small and narrow areas have low probability to get samples within their space.

We exploit the information about the cells size to find the narrow area. Our proposal based on the idea of following the adjacent cells size. If the translation is done from a big cell to others smaller ones, which have the same size, then followed by a translation to another bigger cell, then this sequence of the small same-size cells is most likely to be a narrow passage or important area from motion planning point of view. Figure 3-9 shows an example, where the shaded cells represent the most important region in this workspace.

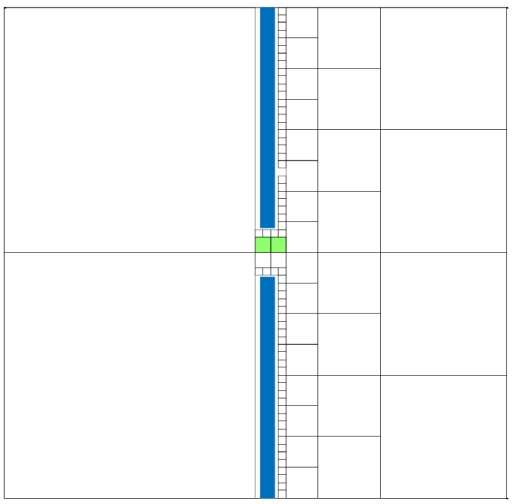


Figure 3-9: Example of the narrow passage identification (green-shaded boxes)

To implement the proposed method, a preprocessing step should be applied to the adjacency graph. Since, the graph of adjacency has many loops and cycled connections between the nodes, for that, a linearization of the graph should be done before the narrow passage identification method is applied. Based on that, the minimum spanning tree (MST) approach is used to build a new liner graph.

The MST tries to build a spanning tree that has the lowest cost, and contains all nodes visited one time. This principle causes another problem, where the tree is planned in unpredictable regions in the workspace based on the edges costs. In order to solve this problem, the edges' weight, which effect the spanning tree construction process, is updated and adapted. The weights are manipulated, in order to give a low cost for edges that placed within narrow and small areas, and at the same time, prevent the MST method of constructing the tree structure near to the obstacles boundaries. Many ideas for weights manipulation are tested to generate the desire spanning-tree. We propose and test five methods. The first method uses the real distance between cells.

The second one uses the uniform cost for translating from one cell to another one. This method based on the idea that, the generated tree should minimize the path cost by using the minimal number of translations; in consequence, it uses the bigger cells when it is possible.

The third proposed method, the bias toward different cells size, updates the edges' weight in such a way that it makes the cost of translation between different cells' size lower than translation between cells that have the same size. This method makes the span tree uses the smaller cells as leaves for the tree, while it uses the bigger cells as roots.

The fourth method, the bias toward equal cells size, suggests giving the lowest cost to the translation between the same size cells. It is the opposite of the previous method, the idea behind this proposal is to make the cells that have the same size, as a sequence does not satisfy the narrow passage condition "bigger-smaller-bigger," instead it will has the pattern "bigger-smaller". The MST in this case constructs narrow passage pattern just when it is necessary.

The last proposed method, the disproportional cost to the distance, gives the edges a cost based on the cells size, the smallest cost is given when translating between the bigger cells. We realize this proposal by finding the longest distance between cells then subtracts all translation distances from that distance. The result is given as a weight of the graph edges. This method gives the translation between the largest cells, which have the longest distance, the lowest cost, while the translation between smaller cells will have higher costs.

Results and discussion

The proposed methods are simulated and tested in two workspaces. The first one is an office-like workspace, where there is one route to connect any two rooms. The second

workspace is generated in such a way that the connections between the free regions have multi-routes.

The result is shown graphically using grading colors, where each color represents a narrow passage sequence. The size of the shaded sequence represents the size of the corresponding cells.

The results of the first and second methods show that the algorithm finds many narrow passages. But the results are considered failed because it generates many sequences near the obstacles and far away from the narrow passage.

The first method that uses the real distance as a cost, makes the MST constructs the tree near the obstacle and follows the smaller cells as shown in Figure 3-10. Where the left figures (a,c) show the ACD and MST graph graphically while the right figures (b,d) show the identified narrow passages, which are denoted using a color for each passage. As seen from the figures the extracted narrow passages using this methods is not accurate.

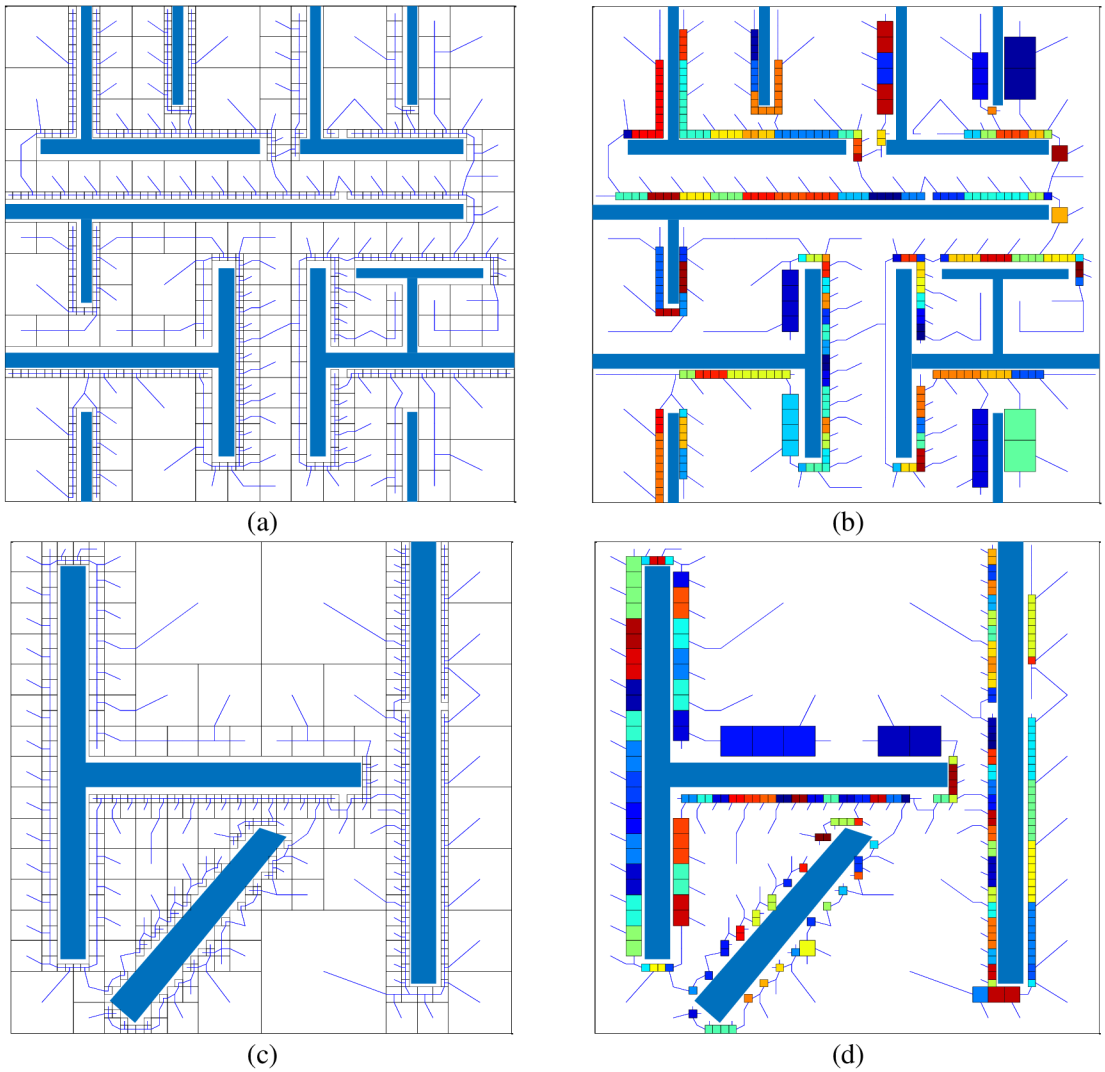


Figure 3-10: Real distance cost method, (a,c) represent ACD and MST tree graphically, (b,d) show the identified narrow passages, each color represents one passage, this approach failed in detecting the correct passages

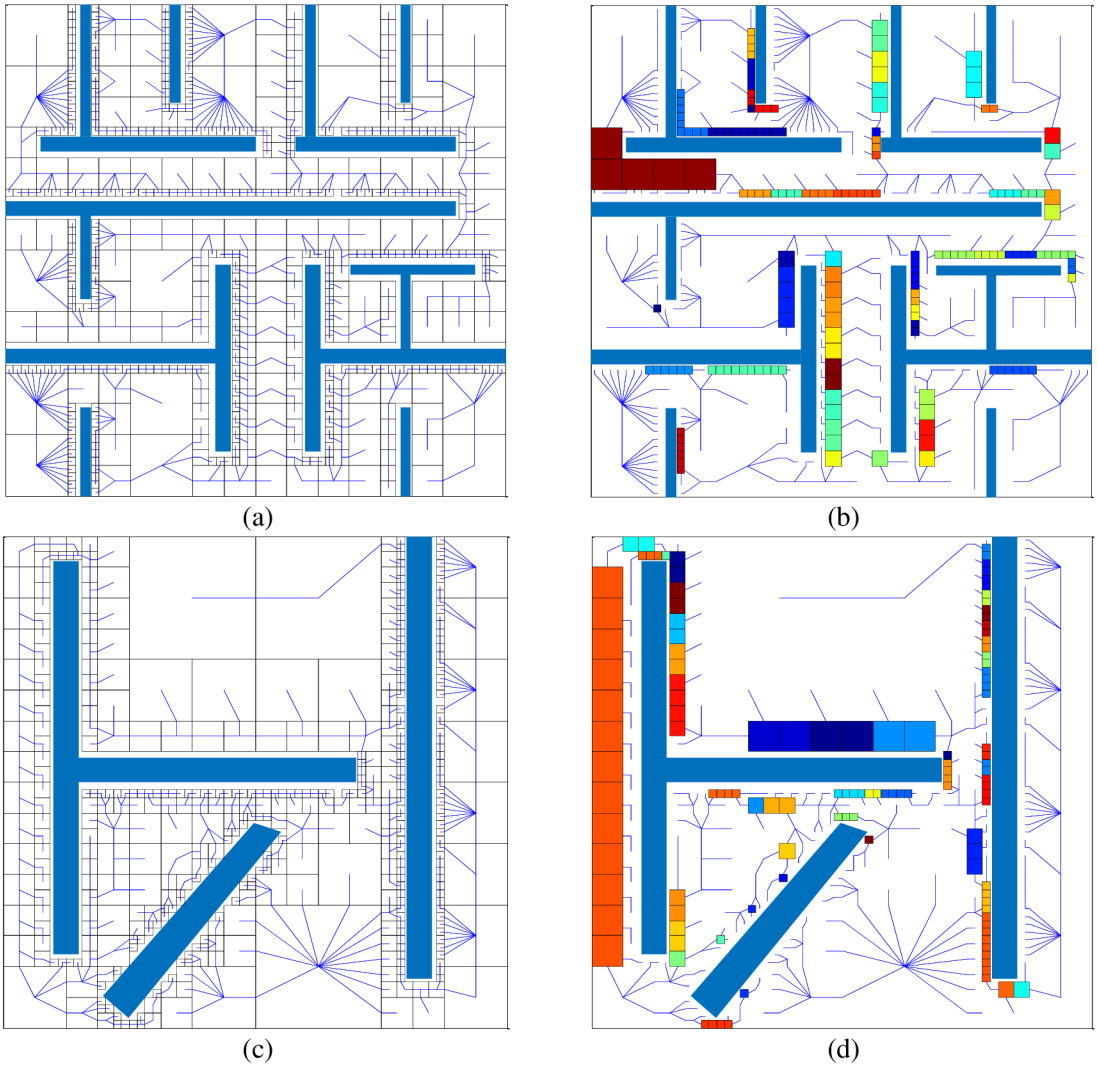


Figure 3-11: Uniform translation cost method, (a,c) represent ACD and MST tree graphically, (b,d) show the identified narrow passages, each color represents one narrow passage, this approach failed in detecting the correct passages

The uniform cost method generates a tree structure which uses more bigger cells as expected, and it generates a better solution, however the result still not good and unreliable. Figure 3-11-a,c show the generated spanning tree in both workspaces, while the narrow passages sequences are shown in Figure 3-11-b,d.

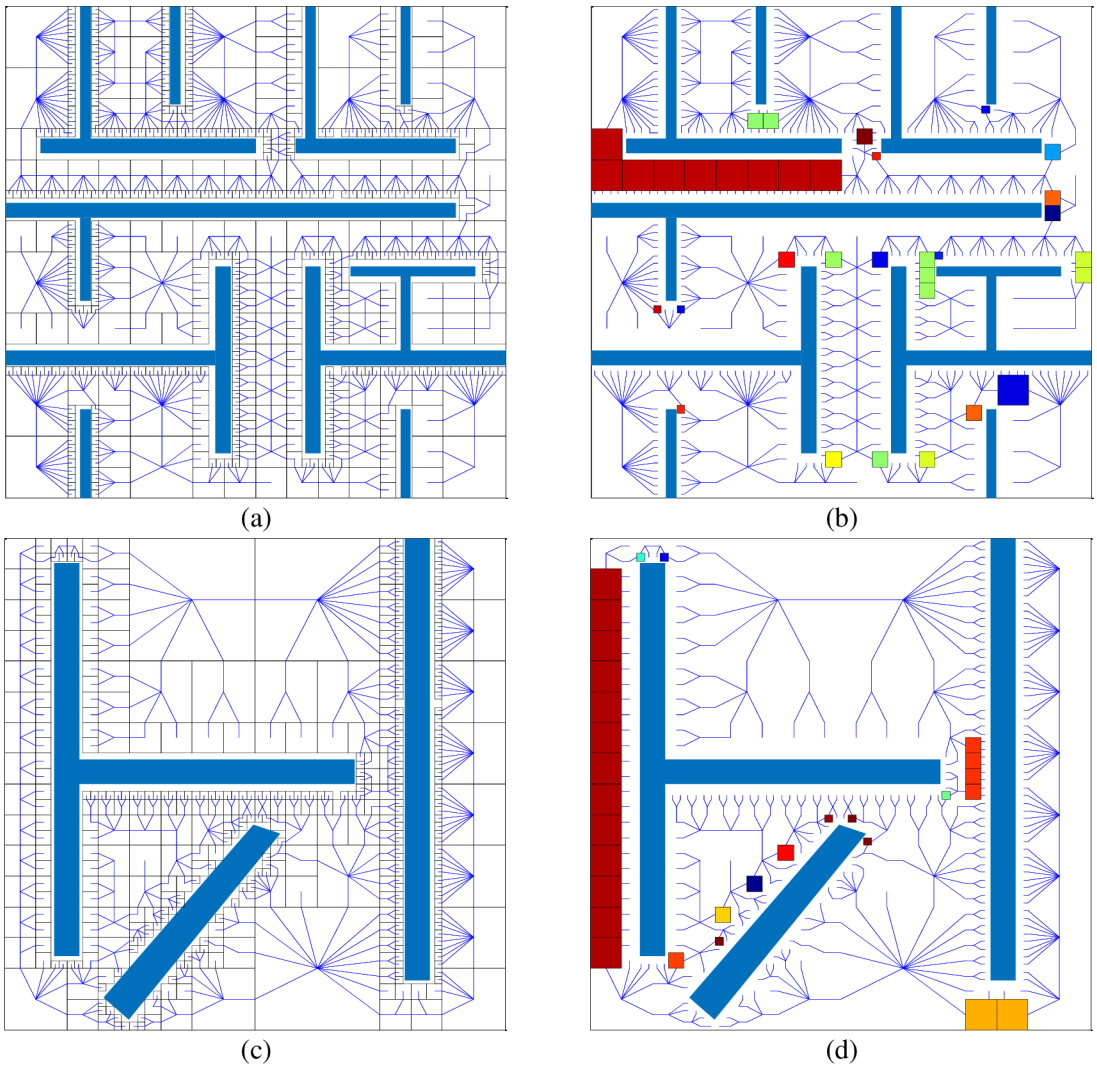


Figure 3-12: Bias to different cells size method, (a,c) represent ACD and MST graph graphically, (b,d) show the identified narrow passages, each color represents one narrow passage

The third method directs the MST algorithm to use different cell size. The generated trees translate between cells that have different size more than the translation between the cells that have the same size, Figure 3-12-a,c show the spanning trees.

This method generates a better solution as shown in Figure 3-12-b,d. However, it also generates long sequences and undesired sequences, especially in the second workspace, which has un-alignment obstacle to the axis, as shown in Figure 3-12-d.

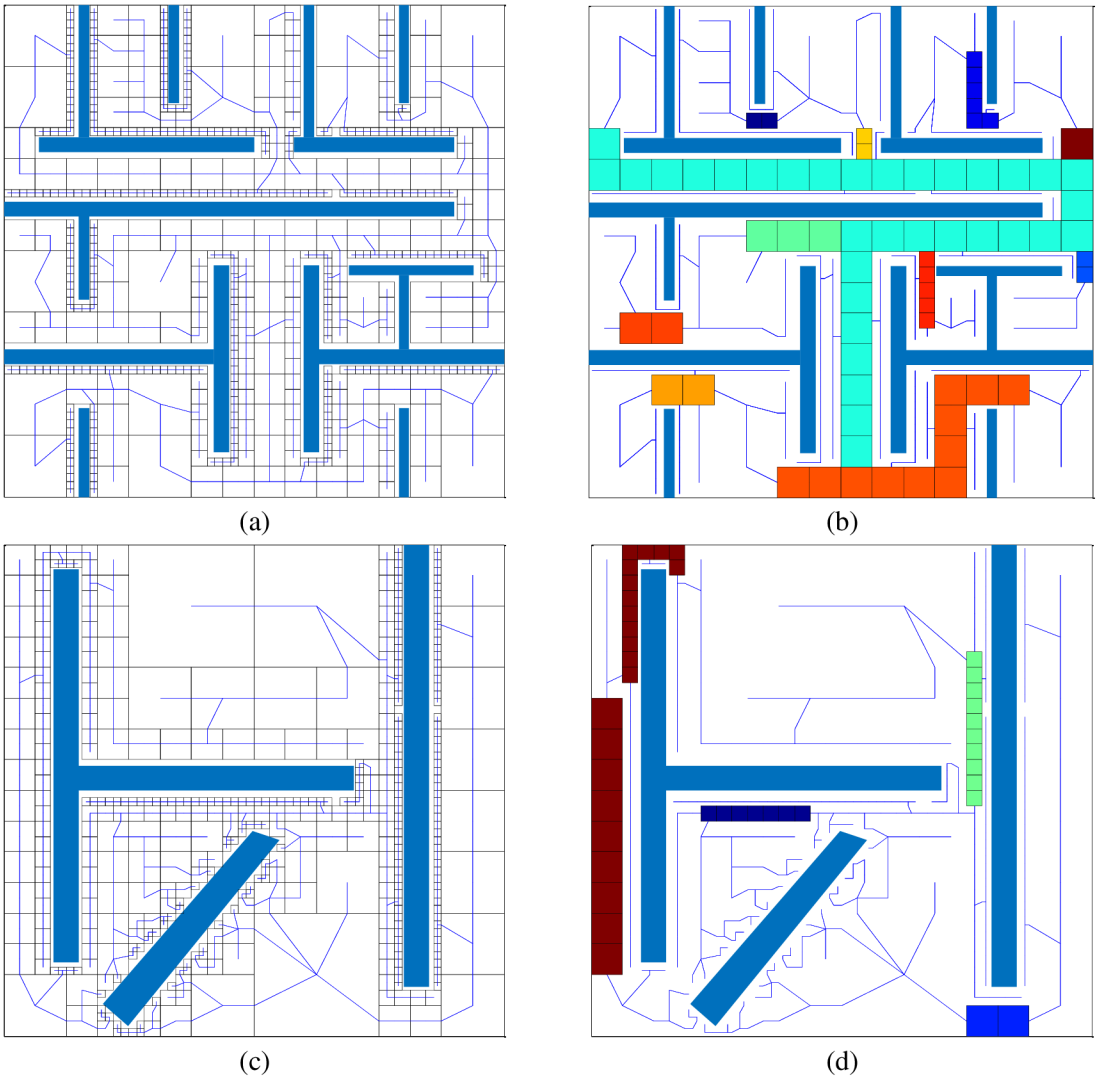


Figure 3-13: Bias to equal cells size method, (a,c) represent ACD and MST graph graphically, (b,d) show the identified narrow passages, each color represents one narrow passage

The fourth method, which gives lower cost to the translation between equal cells size as shown in Figure 3-13-a,c, generates better results, it has the ability to find all narrow passage. But, it generates very long sequences as shown in Figure 3-13-b,d.

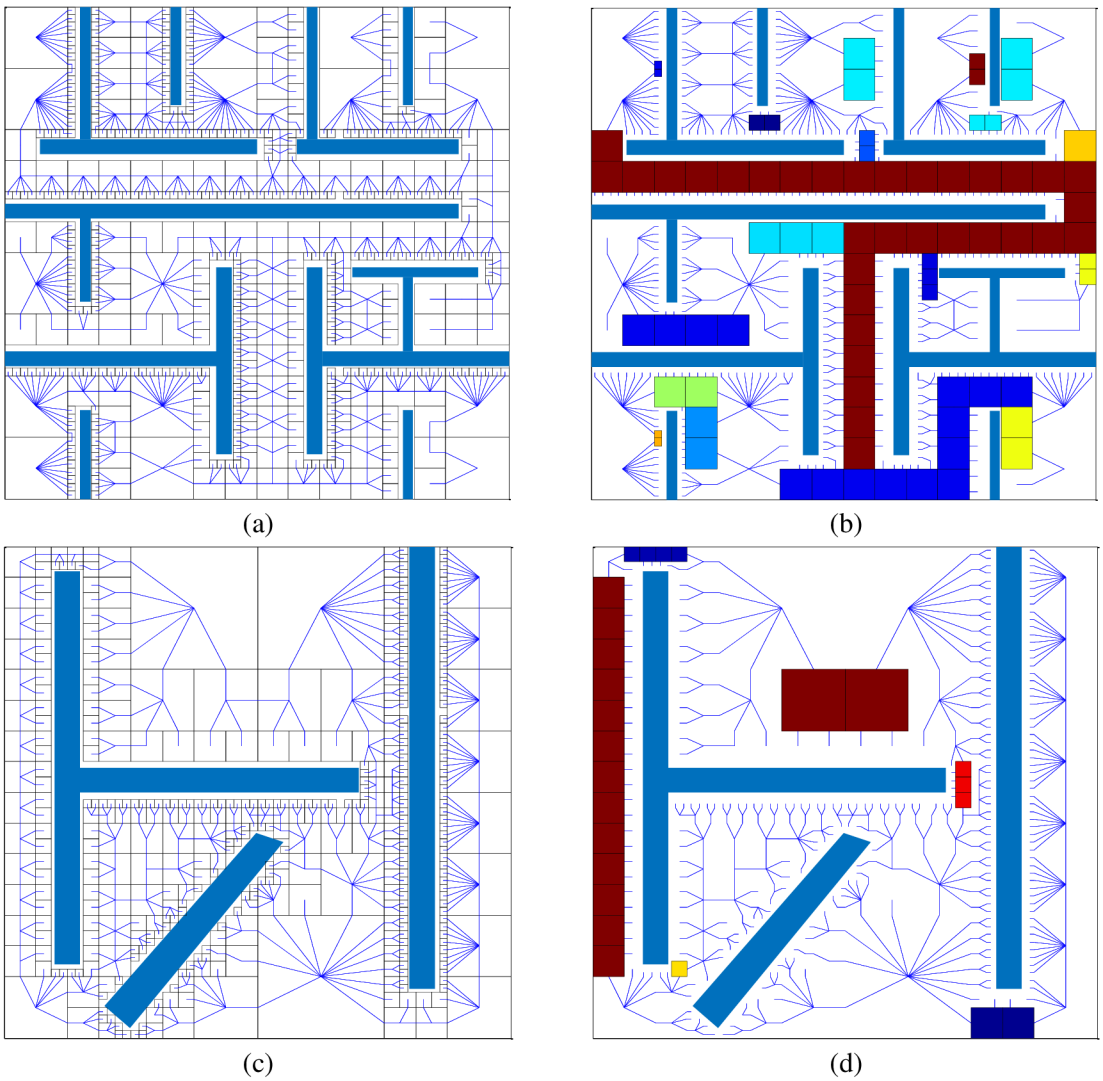


Figure 3-14: Disproportional cost to the distance method, (a,c) represent ACD and MST graph graphically, (b,d) show the identified narrow passages, each color represents one narrow passage

The last proposed method, which has disproportional cost to the distance, generates spanning trees as shown in Figure 3-14-a,c. It produces a relatively good solution. However, it is still has a problem with sequences generation, since it has some faults to find the correct narrow passages, in addition the generated sequences are long, and sometime they merge many narrow passages together as shown in Figure 3-14-b,d.

Summary

In this work, the narrow passage identification problem is discussed. Narrow areas are a problematic issue facing sampling-based motion planner. The cell-decomposition approximation algorithm is utilized to find the free regions and build a graph of adjacency for them based on the adjacency information.

The proposed method to identify the narrow passage, finds a sequence of cells along the graph edges that have the same size, preceded and followed by a bigger cells size. This smaller sequence is more likely to be located in the narrow passage.

Because of the graph of adjacency characteristic, which has many loop connections between the adjacent cells the minimum spanning tree algorithm is used to linearize this graph. Many methods have been proposed to manipulate the edges cost in the graph, in order to make the generated spanning tree traverse through narrow passages in a detectable way, which means following the pattern of the narrow area “bigger-smaller-bigger” sequence of cells. Five methods are proposed, some of them give bad results, and the others give better results as shown in the simulation.

We noticed that the first two methods which gave a bad results (real distance cost, uniform cost), can be updated to find obstacles boundaries cells, based on that, the non-uniform distribution can be introduced to be used in the motion planning samplers, which improve the performance.

We also notice that the minimum spanning tree has a drawback in this algorithm, where some routes are lost. That is happened when the workspace has multi-routes between free areas, where the MST does not distinguish between the loop around obstacle and the loop between cells.

More studies and analysis to the cost manipulation process should be reviewed in the future work.

In the next chapter, the rapidly exploring random tree is reviewed and our contributions using this method combined with exact cell decomposition are presented.

4 RAPIDLY-EXPLORING RANDOM TREE (RRT)

Rapidly-exploring random tree is a probabilistic algorithm introduced in (Lavalle 1998). It is proposed originally for non-holonomic systems, which contain dynamic constraints. The algorithm builds a space-filling tree that is constructed incrementally using samples drawn randomly from the search space, as shown in Figure 4-1. RRT is designed for efficient search in environments that have nonconvex obstacles. In addition, it works directly with a set of admissible inputs. This feature makes the algorithm applicable to complex and dynamic systems. This algorithm also, has the ability to use holonomic or non-holonomic movement, and respect algebraic and differential constraints, and that due to its incremental behaviors. The key idea of the RRT is to bias the exploration toward unexplored regions of the space, where the sampler takes points from these regions, and incrementally pulling the search tree outward of the initial position.

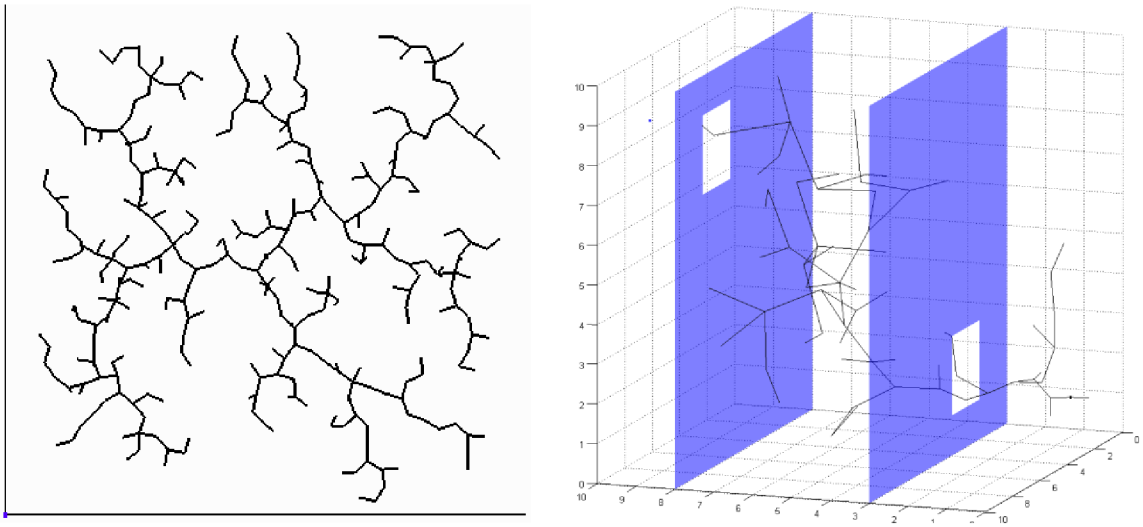


Figure 4-1: RRT expansion in 2D and 3D workspace

RRT algorithm proofed to be probabilistically complete (LaValle et al. 2001), and resolution complete (Cheng et al. 2002).

The algorithm, which shown in Figure 4-3, takes as inputs the initial and the goal locations, along with termination parameters, e.g. the maximum number of iterations to grow a branch, time limit, or other parameters based on the application. RRT algorithm is an incremental approach, where the incremental step is passed to the algorithm as an input parameter (in the basic RRT algorithm). The output of the algorithms is a tree structure, where the nodes represent free samples of the workspace, and the edges represent feasible connections between these vertices.

The principle of the basic RRT algorithm is shown in Figure 4-2. The algorithm places the tree's root at the initial location. Then it takes a random sample from the configuration space, and finds the nearest tree's vertex to this sample (*nearestPnt*). A new point is created on the segments between the random point and the nearest point, it is located far

from the nearest point by *e* distance, where *e* is the incremental step. If no collision is detected with the segment between the nearest and the new points, then the algorithm adds the new point as a vertex to the tree and the segments is added as an edge to the tree structure. These steps are repeated until a termination condition is satisfied or a path between the initial and the goal locations is found.

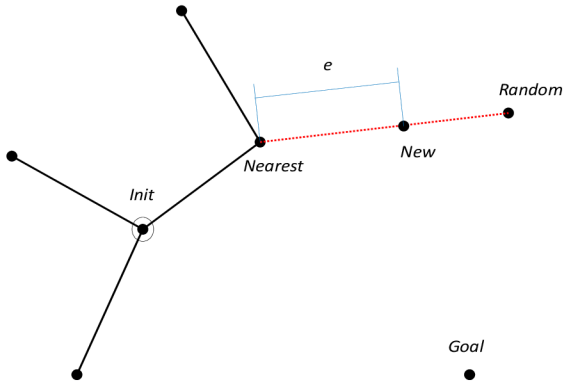


Figure 4-2: RRT algorithm principle

```
Algorithm: RRT
Input:   Initial, Goal, Max Iteration I,
         and the incremental step e.
Output:  The tree graph G.
1.  G.init(Initial)
2.  FOR (i = 1 TO I) BEGIN
3.    randomPnt = randConfiguration()
4.    nearestPnt = G.nearestVertex(randomPnt)
5.    newPnt = NewConfiguration(nearestPnt,randomPnt,e)
6.    IF NOT isCollided(nearestPnt,newPnt) BEGIN
7.       G.addVertex(newPnt)
8.       G.addEdge(nearestPnt,newPnt)
9.       IF G.checkGoal(Goal)BEGIN
10.        RETURN G.success()
11.      END
12.   END
13. END
14. RETURN G.fail()
```

Figure 4-3: Basic concept of RRT algorithm

The quality of RRT solutions is proofed as asymptotically optimal when applying special variations of RRT, e.g. RRT* (Karaman et al. 2012, 2011) , LQR-RRT*, and others (Perez et al. 2012; Nasir et al. 2013; Li et al. 2014).

The drawbacks of the basic RRT algorithm can be summaries as follows:

1. The basic RRT algorithm does not take the path cost into account, which generates non-optimal path.

2. Large numbers of redundant points are generated, when exploring the space to find a path between two locations.

3. It has some difficulty when planning and exploring small areas and narrow passages, because the probability to choose a configuration in these areas is small. Moreover, the probability to connect configurations from these regions to other tree vertices without collision is also small.

4. The generated paths are usually tortuous, and have abrupt changes in the curve.

Researchers try to overcome the downsides of the original RRT. They proposed many ideas to improve the performance and efficiency of this randomize technique. Some of their work based on new ideas, and the other based on improving the algorithm itself, where RRT algorithm can be divided into sub-functions, i.e. 1- Initialize the tree. 2- Choose the next configuration in order to pull the tree toward it. 3- Find the nearest vertex of the tree to the chosen point. 4- Expand a new tree branch. 5- Check the collision. These sub-functions were studied and reformulated to be more efficient.

The first sub-function is developed in various methods, i.e. instead of using one tree, bi-directional trees or multi-trees can be used (Kuffner et al. 2011; Lavalle et al. 2000; Militão et al. 2010; Strandberg 2004), and that lead to other researches on choosing the root of these trees (Wang et al. 2012).

The second category of RRT improvements enhances the sampling strategies. For example, the bias toward goal configuration, or toward hull around the goal (Lavalle et al. 2000). In another work the authors introduce a bias toward previous success (Bruce et al. 2002). Other researchers adapt the choosing of a next point, based on the environmental aspects, e.g. large Voronoi regions (Lindemann et al. 2004; Sakahara et al. 2008), narrow passage identification (Jiandong et al. 2011; Zhong et al. 2012), and collision information (Peng Cheng 2001; Cheng et al. 2001; Jaillet et al. 2011).

The third category of improvement optimizes the nearest-point searching in the tree structure, using spatial indexing techniques, e.g. kd-tree (Urmson et al. 2003; Yershova et al. 2007; Atramentov et al. 2002).

In the fourth category, some researches try to develop the way of extending the branch (Militão et al. 2010) or to introduce a new branching method to fit kinematic and dynamic constraints (LaValle 2006; Jaillet et al. 2011; LaValle et al. 2001).

The fifth category improve the efficiency by manipulating the collisions checking methods, e.g. the use of lazy approach, which postpone the collision checking until it's needed (Vahrenkamp et al. 2007).

Many RRT variants try to solve the disadvantages of basic RRT algorithm. A survey of some RRT variations were reviewed and published in (Abbadi, Matousek 2012; Abbadi et al. 2011). In the next paragraphs, some of these methods are discussed in more details.

Bidirectional and multidirectional planners are examples of the RRT variants that try to speed up the exploration by controlling the root location and the tree number. In

bidirectional planners, two trees are expanded. The first one rooted on the initial position and the second tree rooted on the goal location. The two trees branch until they meet each other, and then the algorithm merges them and find the path from the start to the goal locations. Figure 4-4, Shows dual RRT trees in the T-trap workspace.

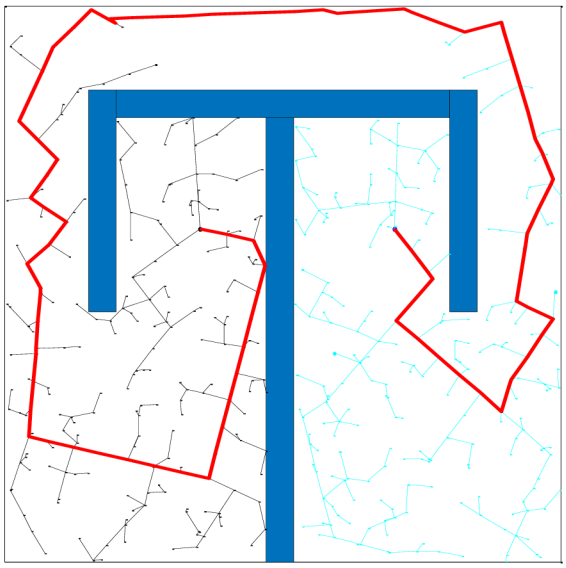


Figure 4-4: Dual RRT trees

Another improvement, in the same course, uses multi RRT planners. The trees work simultaneously, and try to connect to each other in order to find a solution. The choosing of trees roots is done uniformly, or based on heuristic concepts.

Another approach uses augmented local trees beside the bidirectional trees. This method proposed to explore the hard to reach regions (Strandberg 2004). It is based on the idea that RRT algorithm needs to take better care of samples which fall into crucial, but hard to reach, regions. The algorithm spawns a new local tree, which grows until it reaches outside of the hard to reach the region, and merges with another tree. However, a new issue based on this method rises up, where the quantity and percent of growing for these trees to main trees should be optimized. The author in (Strandberg 2004) suggests using a limited number of the local trees, during the planning phase to prevent this method from acting like RPM. Another suggestion is to use a probability parameter to make a tuning between the growth of the two main trees and the local trees. Another idea suggests the use of volume of the box bounding the tree in the configuration space in order to reduce the cost of trees' connection checking. Each time the bounding box of a tree grows, the new node will be used for possible connection with other trees, since it was this node that caused the growth of the bounding box (Strandberg 2004).

This methods increase the probability to find a path, because, instead of testing the reachability to the goal point, they test if any points of the tree are near to other points in different trees, which increases the probability to find a path.

The drawbacks of using multi-trees can be summarized as follows.

1- The generated path is very tortuous and contains many sharp angles between edges.

2- Many redundant points are generated in each tree.

3- When the tree uses kinematic equations to generate the branches, the connection between two trees could be inapplicable; that happens when the connection between two nodes of the trees cannot adapt to satisfy the constraints.

4- The other drawback come out when the goal is not a specific configuration (Bruce et al. 2002), it could be desire state or set of configurations, which means the bidirectional or multidirectional search are not used, because they decrease the generality of the goal state specification.

Another development of the RRT algorithm improve the expanding methods, in order to pull the growth of the trees outward of the root rapidly. For example, bias the tree expansion toward the large Voronoi regions. Originally, the basic RRT algorithm uses the randomize approach to approximate the bias toward large Voronoi areas, and that because the correlation between the probability of selecting a random point and the volume of its Voronoi region, where the larger regions implied a higher probability of selecting points from them.

Some techniques of RRT construct a Voronoi diagram explicitly. Then update it incrementally, while the algorithm grows the tree. RRT uses this information to choose a node of the tree, which has the largest Voronoi region for the next expansion. The direction of expansion pointed to somewhere in the region, e.g. the center of Voronoi region or to the farthest Voronoi vertex from tree node. In (Lindemann et al. 2004) the authors proposed two methods to improve the bias to larger Voronoi regions without explicitly calculating the Voronoi diagram. The first one is the Volume-based Voronoi-biased RRT (VB-RRT). It directs the exploration to the approximate center of the region. The second method is dispersion-reducing Voronoi-biased (DR-RRT). It directs the exploration to the farthest vertex of the approximate vertices that bounded the largest Voronoi region. This strategy for dispersion reduction based on the idea that, connecting nearest neighbor with farthest vertex will reduce the dispersion, since this distance is considered as the largest empty distance in configuration space and eliminating this distance from dispersion calculation will reduce it.

The VB-RRT method based on selecting K samples from the space instead of selecting only one as in original RRT. Then, a node is chosen from the tree, if it is the nearest one to most of selected samples. The average of the samples approximates the center of their region. The cost of doing K nearest-neighbor queries in every iteration for all tree's node is highly expensive. To reduce this cost, the K samples are kept for further reuse in next iterations. If at some point during the search the initial K samples are insufficient, more

samples are added. The drawback of VB-RRT strategy is, it can easily trap in local minima when the search tree encounters obstacles in the large Voronoi regions.

The DR-RRT (Dispersion-reducing Voronoi-biased RRT) method proceeds similarly to the previous approach. It is based on the choosing of K random samples and creates ordered samples-set S; it sorts them based on the distance to their nearest neighbors in the search tree. Then it chooses the farthest sample and grows a branch toward it. If this fails, the algorithm takes the next farthest sample and repeats the process. When it fails for all samples, more samples are added and the algorithm continues until it achieves its goal.

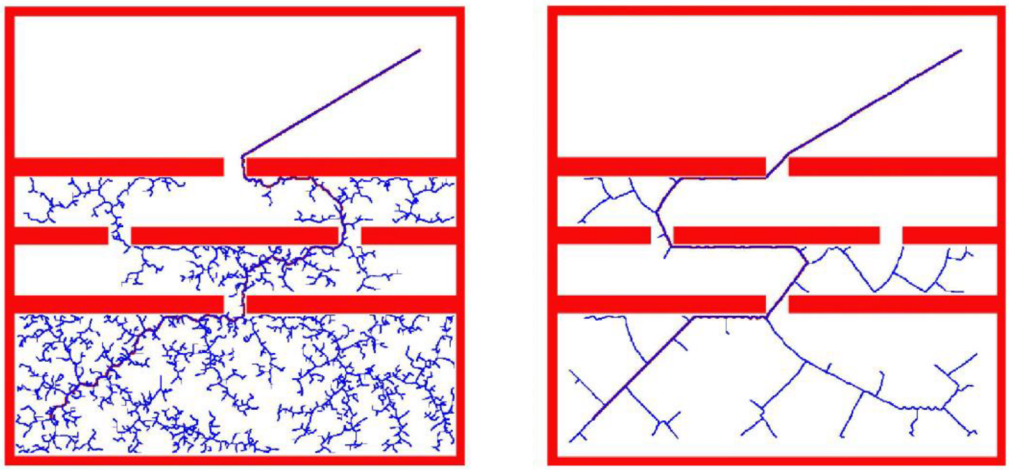


Figure 4-5: Standard RRT vs dispersion-reducing RRT using initial samples number = 1000. Source: (Lindemann et al. 2004)

The distance and nearest neighbor are computed one time when a sample is added to S initially or later on when it is needed. This step adds more cost to the algorithm, but the benefit is the fast exploration. Figure 4-5 shows the difference between original RRT and DR-RRT from exploration point of view.

Another expansion-based improvement to RRT called RRT-Connect. It uses a greedy approach to growing up the tree. It is based on the iterative growth of the tree's branches as long as they can; Instead of attempting to extend an RRT branch by a single step e, it iterates branching in the same direction until it reaches the random point or an obstacle is collided. This greedy approach frequently performs better since any relatively open and unobstructed regions are traversed in a single iteration. However, the RRT-Connect planner was designed for path planning problems that do not involve differential constraints. In this case, the need for incremental motions is less important. This approach proved to be probabilistically complete (Kuffner et al. 2000).

RRT-Blossom is another variant of RRT that behaves in the same way like the basic RRT. However, it adapts the branching function of RRT. It adds a new point to the tree if the distance between the new point and the other tree's nodes is more than a specific distance *BLOOSM_DIS*. The benefit of that is to reduce redundant points, which are added to the tree, and make the tree spreads in the free space faster by preventing the tree from

growing from inside. Figure 4-6, shows the principle of this method. Generally, reducing the number of nodes in RRT trees has a significant effect on the performance of RRT algorithm when checking the nearest neighbor (Maciej Kalisiak et al. 2006; Almahairi 2010).

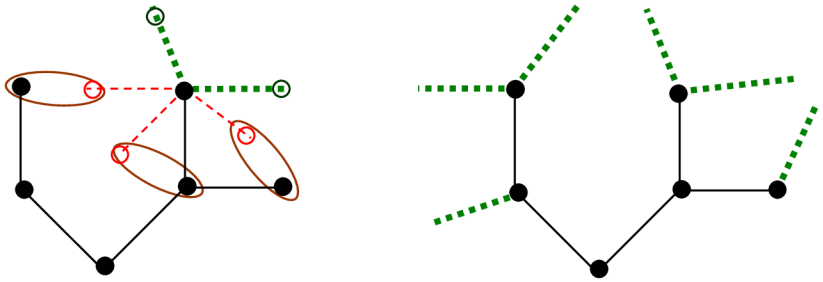


Figure 4-6: RRT-BLOSSOM principle. Source: (Maciej Kalisiak et al. 2006)

Variable Length RRT (VLRRT) is proposed by (Militão et al. 2010). It uses the information that is collected throughout the exploration to adapt the length of the tree's branches. The tree will cover the less obstructed regions faster, while maintaining the ability to navigate through more obstructed areas. This proposal suggests changing the extension lengths as follows. The extension lengths of the branches become longer in open areas and shorter in cluttered ones. For implementing this idea, each node of the tree has an extension factor associated with it. Whenever an extension from a node fails, the extension factor is decreased. Else, the extension factor is increased. The new node inherits the extension factor from its parent.

The way of increasing or decreasing extension factor realized in many ways, e.g. multiplying the original extension-length by a fixed value, or adding a constant value to the extension length. In decreasing the extension factor, the same principle is used in addition to another option, which reset the extension factor to the original extension length.

Another update to the previous method takes into account the direction of obstacle. In the directional variable length (DVLRRT) approach (Militão et al. 2010), the successful or unsuccessful branching provides an information about the presence or the absence of obstacles in a particular direction, not in all directions. This method is realized by storing a directional map of the extension factor in each node of the tree. The extension value is chosen based on the direction of the obstacle.

In similar way, as in VLRRT, the success or fail affect the extension factor, but here in obstacle direction. The new node also inherits the map from its parent. Figure 4-7 shows an example of this method in simulation.

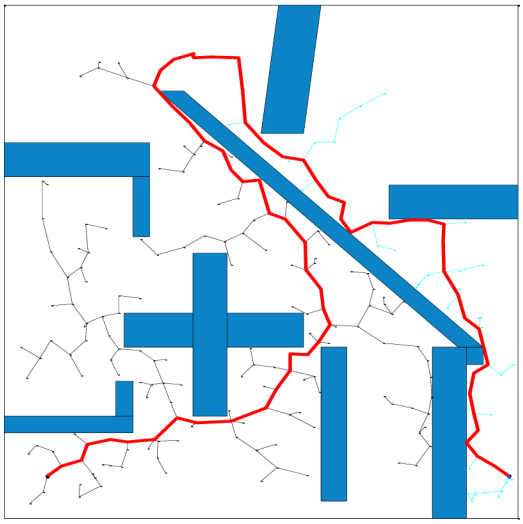


Figure 4-7: The generated path using DVLRRT

In the next paragraphs, we review some RRT improvements that based on trees-growth directing. These strategies improve the original RRT by reformulating the random-point choosing procedure.

RRT-GoalBias method chooses the goal point as the chosen point using a probability of *p,* instead of choosing a random point randomly (Lavalle et al. 2000). Usually converges to the goal would be much faster than the basic RRT. However, in this method, a trade off should be considered when choosing the *p* value. If *p* were a big value, the planner would behave like the randomized potential field method, which is trapped in a local minimum. In general, the bias value is chosen to be small, because, even a small value of the bias forces the planner to reach the goal faster.

The RRT-GoalZoom method is an improvement to GoalBias method (Lavalle et al. 2000). It uses a *p* probability to choose the goal instead of a random sample, and uses a *q* probability to choose a sample from the hull around the goal. The nearest RRT vertex at any iteration controls the size of the region around the goal. The more close the RRT to the goal, the smaller the size of the region around the goal. The author claim, that, this planner has performed well in practice, but still some possibility that the performance is degraded due to the local minima.

The waypoint cache RRT (ERRT) method is proposed for real-time multi-robot systems (Bruce et al. 2002). The key idea of this method is to keep the successful plans in the previous queries, and reuse them as guidance to the RRT growth. The waypoint cache was implemented by keeping an array of constant size of states. Whenever a plan was found, all the states in the plan were placed into the cache. This stores the knowledge of where a plan might again be found in the near future, where the space does not change too much. The results of re-planning using ERRT are more efficient than the basic RRT planner. The algorithm starts with an initial state as the root of the tree, and then it iterates. It uses a probability of *p* as a bias value toward the goal, and a probability of *q* as a bias toward the

old path points, in addition to a probability of *1-q–p* to pick a random point uniformly from the space. This technique can be used to speeds up the path finding in moving obstacles spaces, where first, it finds the path regardless of these moving obstacles, and then biases toward the path's points when a new plan is required.

Another methods was proposed in (Urmson et al. 2003) for guiding the tree growth. The heuristically guided RRT (hRRT) method guides the growth of the randomized tree, based on two aspects. 1- The size of Voronoi region for tree nodes. 2- The quality of the path to that node. Using these aspects, it estimates the quality of tree regions and expands branches from the qualified ones, which means, the regions of the tree are chosen for expansion rather than particular nodes.

RRT with the collision tendency method (RRT-CT) was proposed to improve the planner under kinematic and dynamic constraints (Peng Cheng 2001; Cheng et al. 2001). The key idea is to keep track of the unsuccessful edge expansions, and exploiting this information. The authors proposed two methods to improve the original RRT; the first one depends on excluding control-input from re-execution, if it is already applied to a specific node. The second improvement is done by reducing the probability of choosing nodes that have a high collision tendency. They call this factor the constraint violation frequency (CVF). Each node in the tree has a CVF, which calculated over the route from the initial state to this node. It represents the number of collisions when applying control-inputs, divided by the number of all branching possibility. The advantage of this method is to prevent further expansion attempts, which have high probability to fail. In addition, it uses the available information to bias the selection of next node to the nodes with lower collision tendency. Figure 4-8, shows the CVF value, where the darker points represent node with high CVF value.

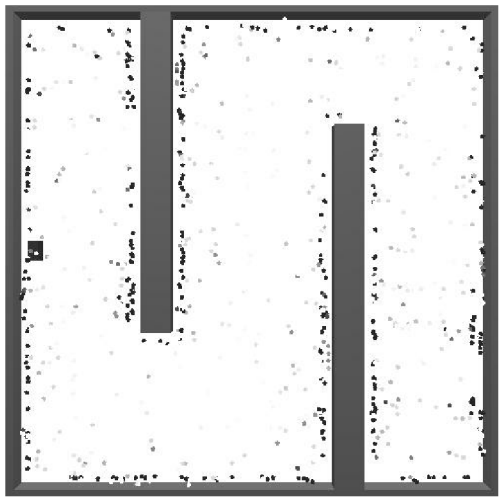


Figure 4-8: CVF; the darker points the higher value. Source: (Cheng et al. 2001)

The conditional density growth (CDG) model is proposed in (Esposito 2013) as an idealized model of RRT growth. It is primarily suited to holonomic systems operating in

expansive configuration spaces. Using this model various statistical properties of the RRT's configuration space could be derived such as the expected value, variance, and distribution properties.

In the next section, our contributions to develop the RRT algorithm are presented, in addition to the simulation results.

4.1 Contributions, Tests and Results

In this section, we present our contributions to the motion-planning problem using RRT algorithm. We re-implement the RRT algorithm to fit the applications of omnidirectional mobile robot, and propose some advance methods to enhance the RRT performance and overcome the drawbacks.

This section is divided into five parts. In the first three, we review many RRT developments and made an evaluation of them, in addition to statistical analysis. We also proposed a new algorithm to shorten the RRT's path. In the last two parts, new methods to enhance the RRT navigation in small and narrow areas are presented. All contributions in this domain are published, and the title of each section is taken from the publication name.

4.1.1 RRTs Review and Options

The path planning is an important issue in the mobile robot field. It allows the robot to move from point A to point B safely. Many methods have been proposed in this domain, which are differed in efficiency and time complexity. One of the advanced path planning methods is the rapidly exploring Random Trees (RRT). In this work (Abbadi et al. 2011), several variations of RRTs are reviewed, and an evaluation of their performance was tested in different environments.

Experiments and Results

Many RRTs options are tested in four different workspaces, which are, the free workspace, the low density of obstacles, the high density of obstacles, and trap workspace as shown in Figure 4-9.

The parameters of the experiments are set as follows. The maximum number of RRT iteration is limited to 2000; the iteration means the number of RRT attempts to grow a branch of the tree.

The results are obtained statistically using 100 tests for each method. The outputs of the tests are the average value of node number in RRT tree, the average value of path's nodes number, the average value of the execution time and the success rate to find a path.

The result of each attempt is considered in the average calculation, if the RRT find a path, else, the results of failed tries are ignored.

The average of tree nodes number comparing to the number of path nodes gives an idea about redundant points in each method. In addition the average of path nodes corresponds to the number of curves in that path; the higher the number the more torturous the path.

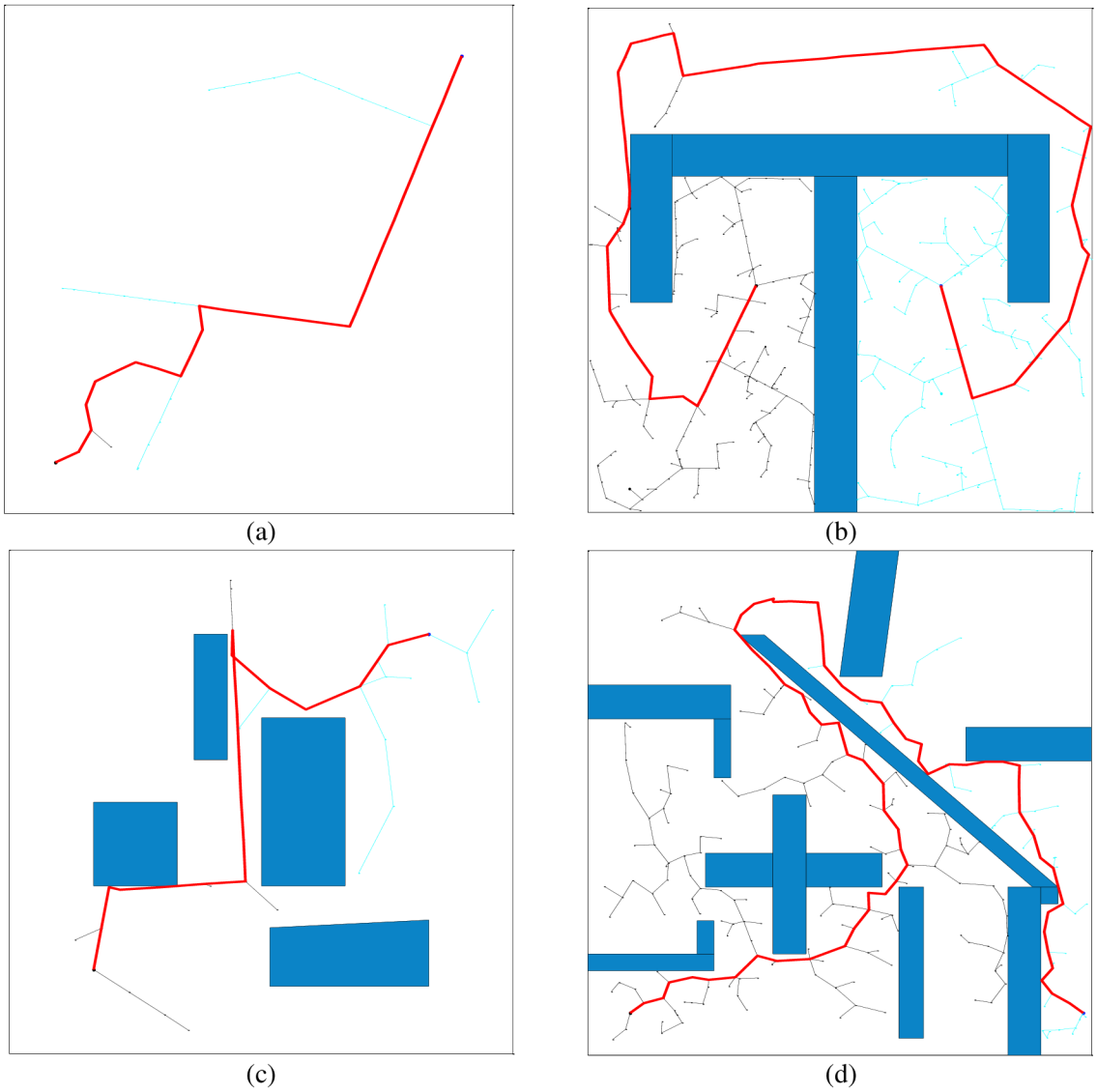


Figure 4-9: RRT testing workspaces, (a) the free space workspace, (b) the trap obstacle workspace, (c) the low density of obstacles workspace, (d) the high density of obstacles workspace

The simulation results of the free workspace are listed in Table 4-1, where the bidirectional-connect method has the best results in terms of time efficiency. Generally, the bidirectional methods were able to find the solution faster than the unidirectional approaches.

The trap workspace results are shown in Table 4-2. The bidirectional-VLRRT method has the best result in terms of time efficiency. Moreover, the bidirectional methods show better results in completeness aspect. They were able to find a solution 100% (except the ExtExt; the bidirectional basic RRT).

Table 4-1: The results of free workspace. The (BI) denote the using of bidirectional trees, and the bold numbers indicate the best results

	Time [s]	Tree node	Path node	Success [%]
Basic RRT	0.1240	384.27	38.83	100
Con RRT	0.0835	409.12	7.51	100
Bias RRT	0.0421	124.52	36.63	100
ConCon (BI)	**0.0026**	36.06	**3.00**	100
ConExt (BI)	0.0094	87.13	10.79	100
ExtCon (BI)	0.0106	93.53	11.74	100
ExtExt (BI)	0.0208	75.71	37.54	100
RRT-BLOSSOM	0.1559	352.99	38.89	100
BLOBLO (BI)	0.0251	76.85	37.97	100
VLRRT	0.0177	51.41	16.85	100
VLRRT(BI)	0.0114	**35.36**	21.25	100
DVLRRT	0.0263	71.76	23.71	100
DVLRRT (BI)	0.0163	47.30	26.60	100

Table 4-2: The results of trap workspace. The (BI) denote the using of bidirectional trees, and the bold numbers indicate the best results

	Time [s]	Tree node	Path node	Success [%]
Basic RRT	0.6236	815.58	89.40	11
Bias RRT	0.6722	697.66	87.06	31
RRT Con	0.3591	708.89	**26.06**	99
ConCon (BI)	0.3253	645.14	26.64	100
ConExt (BI)	0.4362	723.97	39.27	100
ExtCon (BI)	0.4102	696.12	38.44	100
ExtExt (BI)	0.7815	1019.40	85.14	96
RRT-BLOSSOM	0.6708	591.36	55.51	74
BLOBLO (BI)	0.4665	464.40	55.16	100
VLRRT	0.2750	**272.63**	27.81	100
VLRRT (BI)	**0.2406**	279.77	40.23	100
DVLRRT	0.4030	368.91	36.77	98
DVLRRT (BI)	0.3039	326.09	42.57	100

The results of the last two tests in low-density and high-density of obstacles are shown in Table 4-3, and Table 4-4, respectively. The ConCon (bidirectional connect RRT method) has the best results regarding to the execution time. Also, it is notable, that all bidirectional methods are probabilistically complete in these tests.

Summary

The aim of this work was to review and test the performance of the reviewed algorithms. The results show that the dual tree variants are more completeness in all workspaces. The most successful strategy regarding to the time of execution is the bidirectional-VLRRT in

trap obstacles. Also, the bidirectional-ConCon strategy gives the best results in the low and the high density of obstacles. However, the result cannot be generalized for all environments.

Table 4-3: The results of low-density workspace. The (BI) denote the using of bidirectional trees, and the bold numbers indicate the best results

	Time [s]	Tree node	Path node	Success [%]
Basic RRT	0.1650	336.49	43.03	100
Bias RRT	0.0841	147.97	41.55	100
RRT Con	0.1232	375.27	11.24	100
ConCon (BI)	**0.0270**	127.57	**7.56**	100
ConExt (BI)	0.0393	144.56	16.51	100
ExtCon (BI)	0.0362	140.71	16.11	100
ExtExt (BI)	0.0512	111.32	41.42	100
RRT-BLOSSOM	0.1620	259.36	43.73	100
BLOBLO (BI)	0.0556	104.24	40.80	100
VLRRT	0.0519	87.94	25.10	100
VLRRT (BI)	0.0366	**68.31**	27.34	100
DVLRRT	0.0634	102.05	30.28	100
DVLRRT (BI)	0.0434	76.24	31.49	100

Table 4-4: The results of high-density workspace. The (BI) denote the using of bidirectional trees, and the bold numbers indicate the best results

	Time [s]	Tree node	Path node	Success [%]
Basic RRT	1.0550	0	0	0
Bias RRT	1.1255	0	0	0
RRT Con	1.2552	1552.15	33.29	07
ConCon (BI)	**0.3984**	518.46	**25.27**	100
ConExt (BI)	0.4906	550.77	53.83	100
ExtExt (BI)	0.4900	406.33	87.37	100
RRT-BLOSSOM	0.8902	0	0	0
BLOBLO (BI)	0.5133	**328.38**	87.11	100
VLRRT	1.1311	0	0	0
VLRRT (BI)	0.5444	380.29	66.37	100
DVLRRT	1.1497	0	0	0
DVLRRT (BI)	0.5422	365.49	70.68	100

4.1.2 RRTs Review and Statistical Analysis

Many ideas have been proposed to solve the path-planning problem. One of them is the rapidly exploring random Tree (RRT). This method is not optimal, but it reduces the required time to obtain a solution. The result of the RRT is a tortuous path, which has many useless vertices.

In this work (Abbadi, Matousek 2012) statistical tests were done, to make a better decision about using a variant of RRT. This work is based on the previous results in (Abbadi et al. 2011), where the tested methods give a variety of results, some of them are very close and some are very diverse. For that, a statistical analysis is done to build some confidence of using one RRT variation instead of another one in some situations.

In addition to statistical tests, we propose a method to reduce the degree of tortuous, and make the path shorter by omitting the useless points.

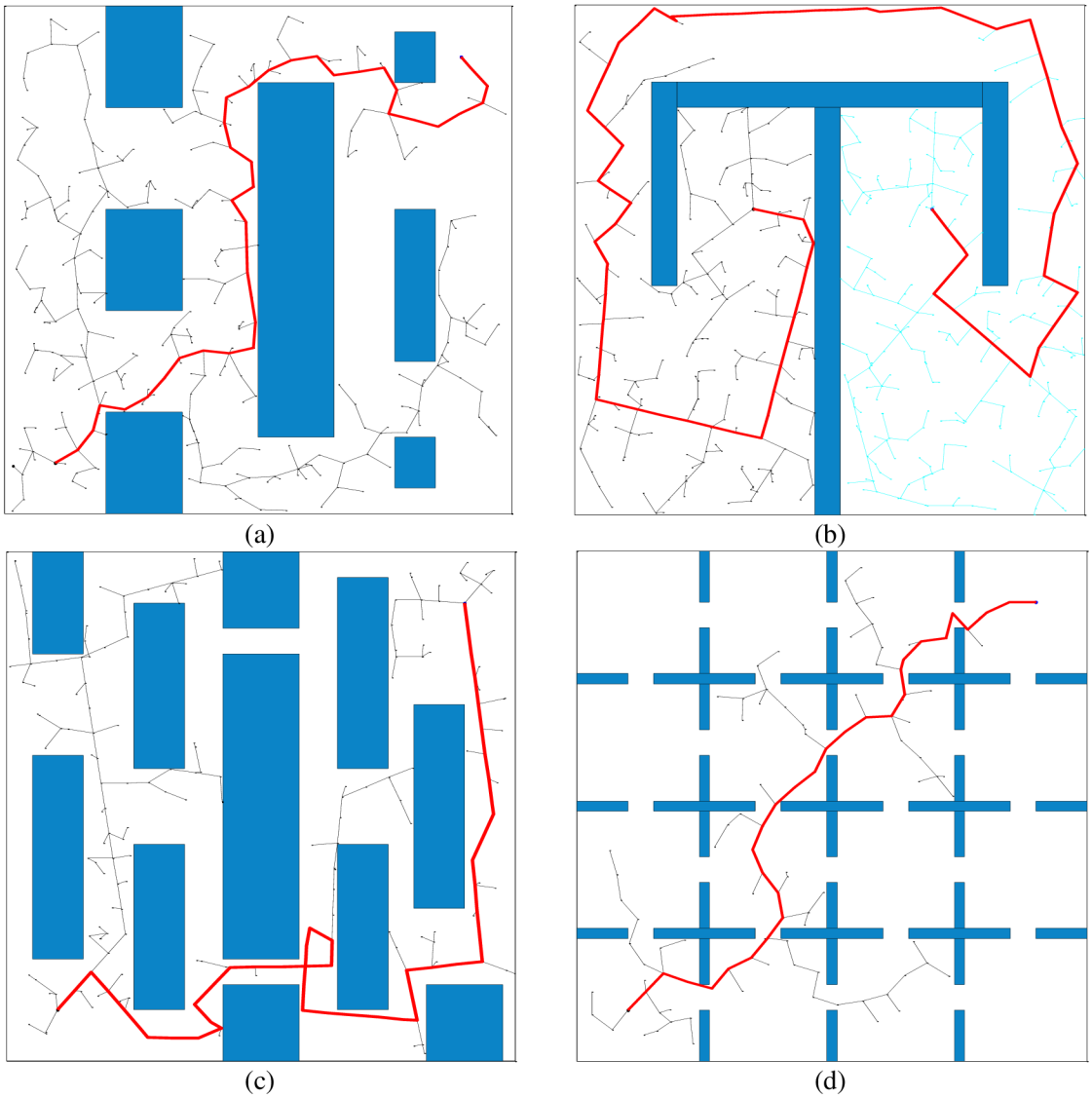


Figure 4-10: The testing workspaces, (a) low obstacles density, (b) T-trap workspace, (c) high obstacles density, (d) doors workspace

Test and Results

We made the tests for 13 RRT variations in four spaces as shown in Figure 4-10. The first workspace has low-density of obstacles (a). The second one has T-trap obstacle (b); the high density of obstacle shown in (c) and the last one is the doors workspace (d).

The test is applied in every workspace separately; we test 13 variants of RRT, 100 times. The fails occurs when RRT variation attempted to extend a branch 2000 times without reaching the goal. We used PC equipped with 2.5 GHz Core2Duo CPU, 2 GB RAM.

The implementation of RRT variations is developed in Matlab and the statistical results are done using Minitab. The comparison between the tests results is done based on the time of execution, the success rate of reaching the goal and the path length.

Execution Time results

The tests results show that the best variation in Low obstacles space is the Vlrrt(2) method, where the average of the time to reach the goal is 0.0467 second and the median is 0.0418, the second best variation is Dvlrrt(2), it has the mean value of 0.0484 second and median value equal to 0.0407. Table 4-5 shows the numerical result of the tests in low obstacle space and the Figure 4-11 shows the boxplots representation of these results.

Table 4-5: Tests results of low density of obstacles. The bold numbers correspond to the best two results, the best results marked by (*), the (2) indicate a bidirectional method

Method	Mean	StDev	Variance	Median	Success
BIAS	0.1035	0.0484	0.0023	0.0890	100
BLOSSOM	0.3552	0.2584	0.0668	0.2714	94
BLOSSOM (2)	0.0615	0.0255	0.0007	0.0564	100
CON	0.3434	0.2546	0.0648	0.2526	93
CON(2)	0.0578	**0.0198**	**0.0004**	0.0559	100
ConExt(2)	0.0617	0.0202	**0.0004**	0.0585	100
EXT	0.2806	0.1991	0.0396	0.2380	95
EXT(2)	0.0516	0.0249	0.0006	0.0444	100
ExtCon(2)	0.0637	0.0234	0.0006	0.0621	100
DVLRRT	0.0893	0.0493	0.0024	0.0734	100
DVLRRT(2)	**0.0484**	0.0259	0.0007	**0.0407**	100
VLRRT	0.0840	0.0436	0.0019	0.0698	100
VLRRT(2)	***0.0467**	***0.01754**	***0.0003**	***0.0418**	100

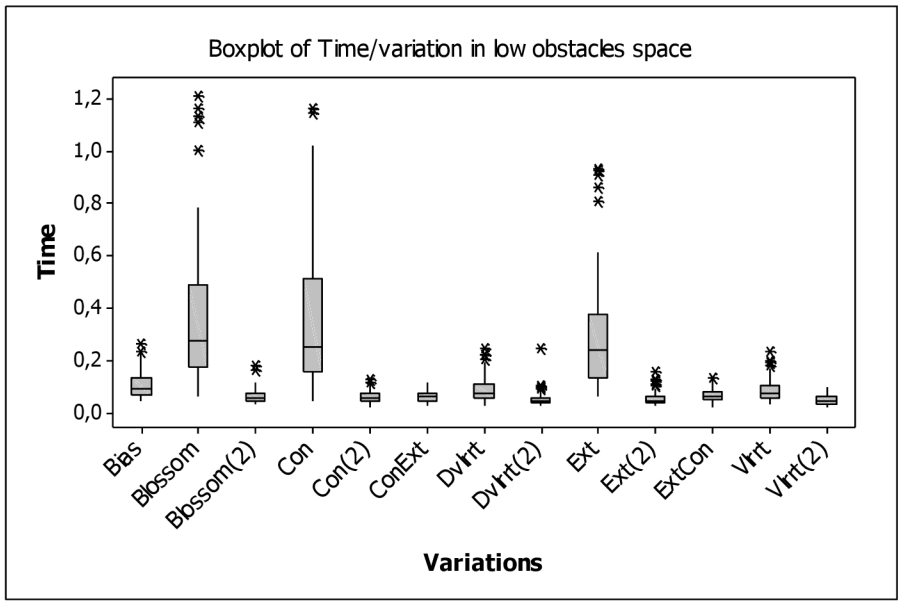


Figure 4-11: Boxplots representation of the results of in the low-density of obstacles

In the T obstacle workspace, the Vlrrt has the best result regarding to the time of execution, however, it also has one fail of reaching the goal. The time average is 0.3740 and the median is 0.3713. The second result achieved by the bidirectional-Vlrrt(2) which has the average time of 0.3984 and median of 0.3849, and without any fail. The numerical results are presented in Table 4-6. And Figure 4-12 shows the boxplot representation of execution time results.

A statistical test was done on Vlrrt and Vlrrt(2), which give the best results. The aim of this test is to validate the hypothesis of using the second best method instead of the first one. Which means, if we use the second best option Vlrrt(2) without any fail, it will give the same result in confidence level of 95%.

This hypothesis implies that we can replace the method that has more probabilistically completeness, with the method that has a less completeness ratio; Figure 4-13 shows the tested hypothesis.

Based on the P-Value, which is >5%, the hypothesis of "Vlrrt and Vlrrt(2) are not equal" is rejected, which means, there is no sufficient difference between the two variations, and the Vlrrt(2) variant can be used instead of Vlrrt, using the confidence level of 95%.

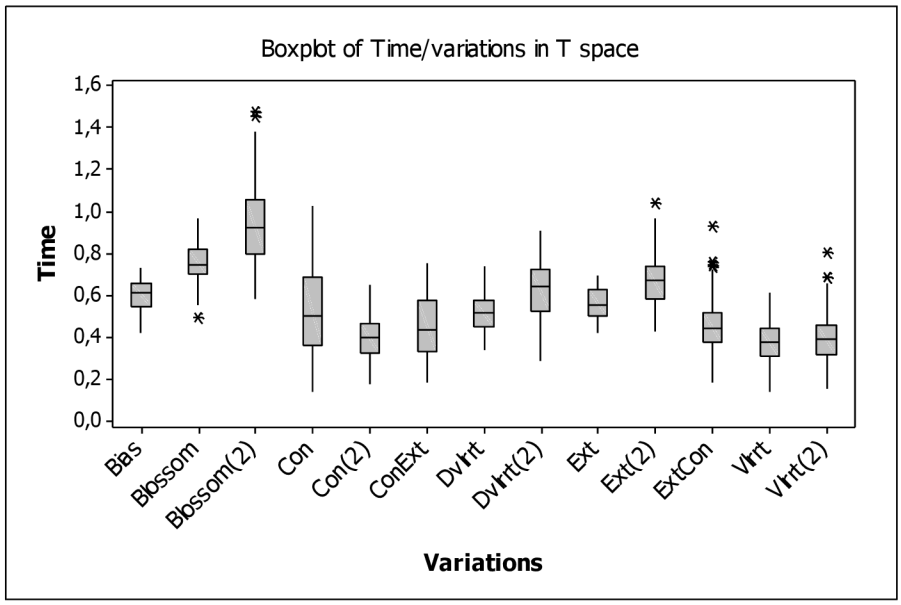


Figure 4-12: Boxplots representation of results in T obstacle

Table 4-6: Tests results of T-trap obstacle. The bold numbers correspond to the best two results, the best results marked by (*), the (2) indicate a bidirectional method

Method	Mean	StDev	Variance	Median	Success
BIAS	0,5968	**0,0736**	**0,0054**	0,6121	71
BLOSSOM	0,7482	0,1068	0,0114	0,7476	35
BLOSSOM (2)	0,9371	0,1852	0,0343	0,9198	100
CON	0,5320	0,2062	0,0425	0,5017	81
CON(2)	0,3996	0,1024	0,0105	0,3948	100
ConExt(2)	0,4484	0,1433	0,0205	0,4326	100
EXT	0,5592	***0,0721**	***0,0052**	0,5521	97
EXT(2)	0,6696	0,1211	0,0147	0,6712	100
ExtCon(2)	0,4502	0,1303	0,0170	0,4388	35
DVLRRT	0,5188	0,0887	0,0079	0,5109	100
DVLRRT(2)	0,6250	0,1235	0,0153	0,6369	100
VLRRT	***0,3740**	0,0984	0,0097	***0,3713**	99
VLRRT(2)	**0,3984**	0,1224	0,0150	**0,3849**	100

In the high obstacle workspace, the Con(2) method gives the best time average, where the mean is 0.1871 and the median is 0.1844. The numerical results are presented in Table 4-7 and the boxplot representation is shown in Figure 4-14.

A statistical analysis is conducted to figure out if the Vlrrt(2) can be used generally based on the confidence level of 95%. The T-test result gives *P-Value* > 5%, as shown in Figure 4-15, which indicate that there is no sufficient difference between the use of Con(2) the best method, and the Vlrrt(2) method, the third best one, in confidence level of 95%.

Two-sample T for Vlrrt vs Vlrrt(2)

	N	Mean	StDev	SE Mean
Vlrrt	99	0.3740	0.0984	0.0099
Vlrrt(2)	100	0.398	0.122	0.012

Difference = mu (Vlrrt) - mu (Vlrrt(2))

Estimate for difference: -0.0244

95% CI for difference: (-0.0554; 0.0067)

T-Test of difference = 0

vs not =): T-Value = -1.55

P-Value = 0.123 DF = 189

Figure 4-13: T-test for the hypothesis "Vlrrt and Vlrrt(2) not equal" in T

Table 4-7: Tests results of high-density of obstacles. The best results marked by (*), the (2) indicate a bidirectional method

Method	Mean	StDev	Variance	Median	Success
BIAS	0.3790	0.1316	0.0173	0.3662	100
BLOSSOM	0.5642	0.2259	0.0510	0.5559	82
BLOSSOM (2)	0.2665	0.1148	0.0132	0.2485	100
CON	0.4397	0.2582	0.0667	0.3640	83
CON(2)	***0.1871**	***0.0712**	***0.0051**	***0.1844**	100
ConExt(2)	0.2033	0.0945	0.0089	0.1902	100
EXT	0.4738	0.1968	0.0387	0.4070	80
EXT(2)	0.2024	0.0738	0.0055	0.1981	100
ExtCon(2)	0.2053	0.0843	0.0071	0.1960	100
DVLRRT	0.3700	0.1510	0.0228	0.3506	99
DVLRRT(2)	0.2175	0.0746	0.0056	0.2033	100
VLRRT	0.3370	0.1216	0.0148	0.3324	99
VLRRT(2)	0.2072	0.0837	0.0070	0.1905	100

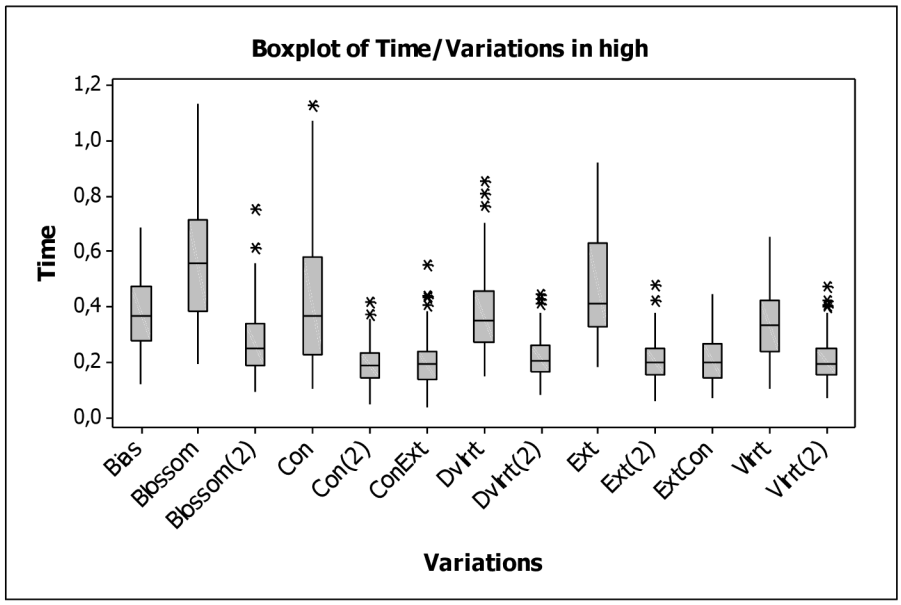


Figure 4-14: Boxplots representation of results in high-density of obstacles

	N	Mean	StDev	SE Mean
Con(2)	100	0.1871	0.0712	0.0071
Vlrrt(2)	100	0.2072	0.0837	0.0084

Difference = mu (Con(2)) - mu (Vlrrt(2)

Estimate for difference: -0.0201

95% CI for difference: (-0.0418; 0.0015)

T-Test of difference = 0

(vs not =): T-Value = -1.83

P-Value = 0.068 DF = 193

Figure 4-15: T-test for the hypothesis "Con(2) and Vlrrt(2) not equal" in high density obstacles

In the doors obstacles workspace, the best variant is Dvlrrt(2) which has the time average equal to 0.2961 and the median equal to 0.2623. The numerical results are shown in Table 4-8, and the boxplot representation is shown in Figure 4-16 for all tested variations.

In the same manner, we test if the Vlrrt(2) can replace the best method in this workspace. The T-test hypothesis assumes that there is a difference between the best variant Dvlrrt(2) and the second best one Vlrrt(2) as shown in Figure 4-17. Based on this test we reject the hypothesis, because of the value of *P-Value* is greater than 0.05, which means there is no sufficient difference between the two best variations in the confidence level of 95%.

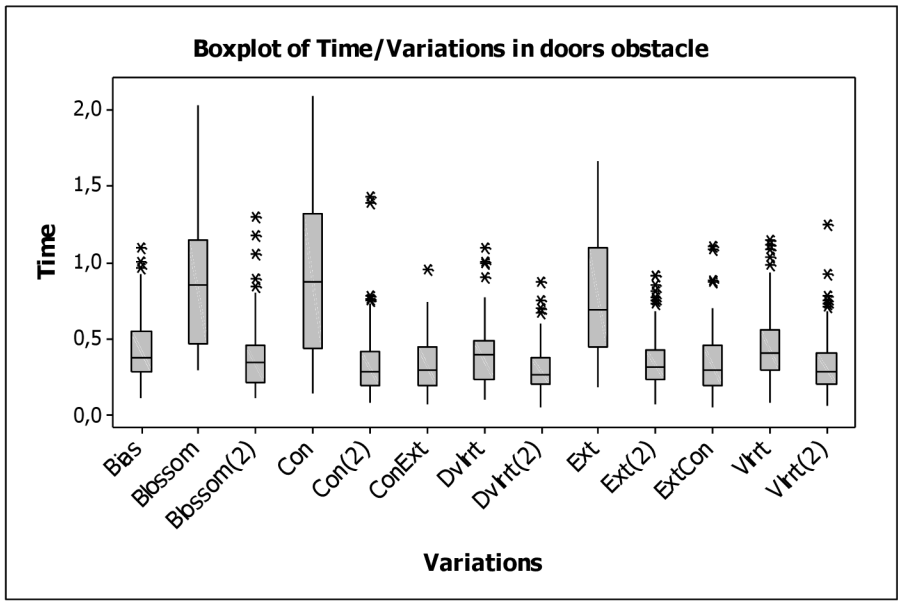


Figure 4-16: Boxplots representation of results in doors obstacles

	N	Mean	StDev	SE Mean
Dvlrrt(2)	100	0.296	0.148	0.015
Vlrrt(2)	100	0.317	0.198	0.020

Difference = mu (Dvlrrt(2)) - mu (Vlrrt(2))

Estimate for difference: -0.0213

95% CI for difference: (-0.0702; 0.0275)

T-Test of difference = 0

(vs not =): T-Value = -0.86

P-Value = 0.390 DF = 183

Figure 4-17: T-test for the hypothesis "Dvlrrt(2) and Vlrrt(2) not equal" in doors obstacle

Table 4-8: Tests results of doors obstacles. The best results marked by (*), the (2) indicate a bidirectional method

Method	Mean	StDev	Variance	Median	Success
BIAS	0.4232	0.2040	0.0416	0.3690	100
BLOSSOM	0.8529	0.4155	0.1727	0.8450	82
BLOSSOM (2)	0.3830	0.2202	0.0485	0.3433	100
CON	0.8834	0.5052	0.2552	0.8670	84
CON(2)	0.3316	0.2315	0.0536	0.2757	100
ConExt(2)	0.3320	0.1854	0.0344	0.2883	100
EXT	0.7535	0.3801	0.1444	0.6839	84
EXT(2)	0.3511	0.1794	0.0322	0.3065	100
ExtCon(2)	0.3427	0.2057	0.0423	0.2896	100
DVLRRT	0.3884	0.1987	0.0395	0.3915	100
DVLRRT(2)	***0.2961**	***0.1479**	***0.0219**	***0.2623**	100
VLRRT	0.4522	0.2389	0.0571	0.4003	99
VLRRT(2)	0.3174	0.1984	0.0394	0.2775	100

The last result and statistical analysis indicate that the Vlrrt(2) can be used in all spaces without sufficient difference between it and the best variants in all space, based on the confidence level of 95%.

Successful rate results

The tests show some variations have a tendency to fail of reaching the goal location, mainly the unidirectional methods. Table 4-9 shows the successful rate of planning process between the initial and the goal locations. The test repeated 100 times, in the four workspaces. In each iterations, the RRT tree tries 2000 times to grow a branch and the test is considered failed if the tree did not reach the goal within this limit.

Table 4-9: Successful rate of RRT methods, the number (2) after the method names, indicates a bidirectional method

	Low	T	High	Doors
BIAS	100	71	100	100
BLOSSOM	94	35	82	82
BLOSSOM (2)	100	100	100	100
CON	93	81	83	84
CON(2)	100	100	100	100
ConExt(2)	100	100	100	100
EXT	95	35	80	84
EXT(2)	100	100	100	100
ExtCon(2)	100	100	100	100
DVLRRT	100	97	99	100
DVLRRT(2)	100	100	100	100
VLRRT	100	99	99	99
VLRRT(2)	100	100	100	100

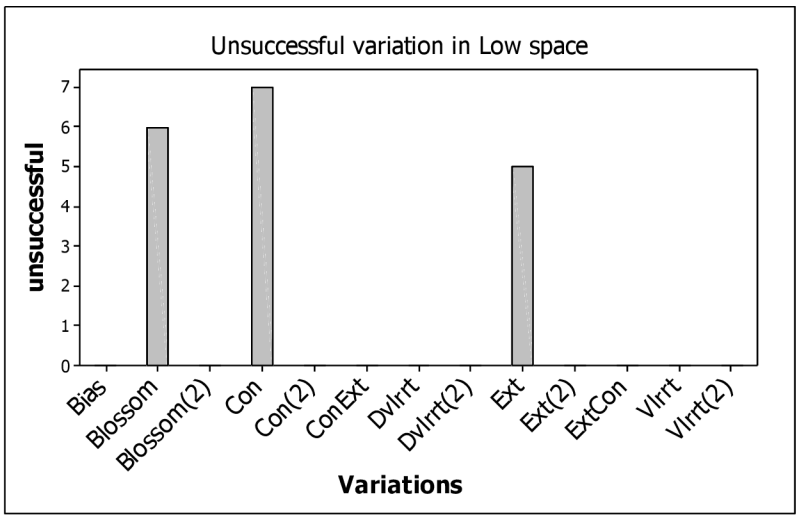


Figure 4-18: Unsuccessful results in low-density obstacles workspace

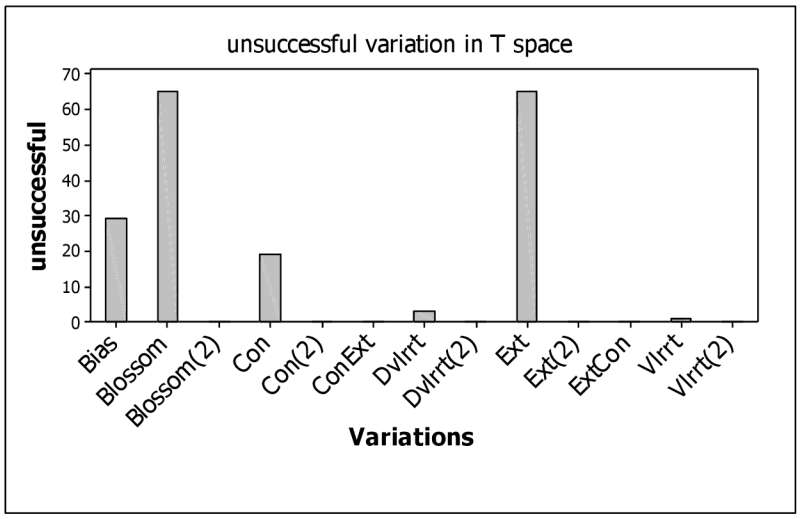


Figure 4-19: Unsuccessful results in T obstacle workspace

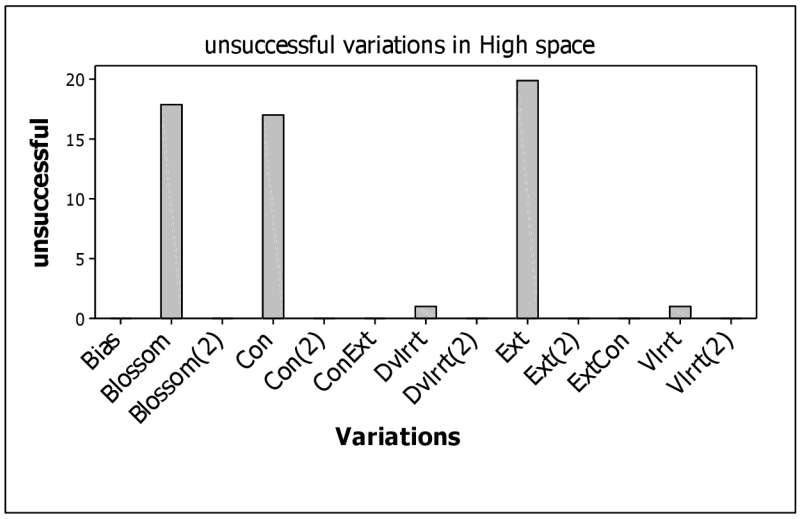


Figure 4-20: Unsuccessful results in high-density obstacles workspace

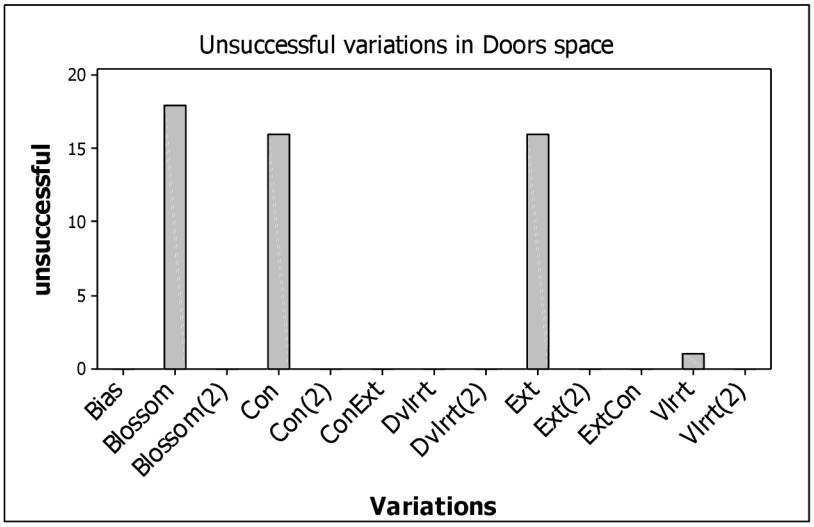


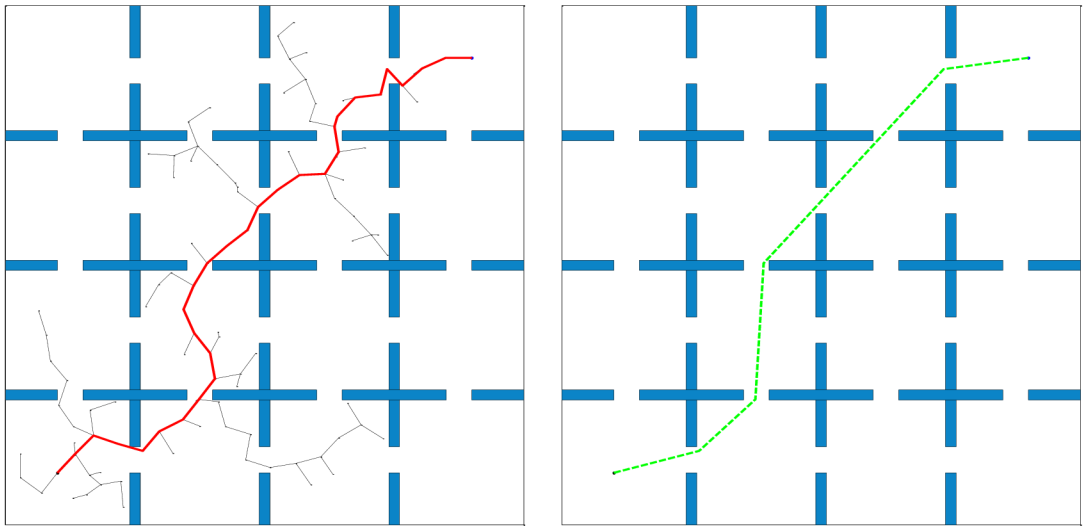
Figure 4-21: Unsuccessful results in doors obstacles workspace

The results show that unidirectional algorithms have more tendencies to fail, more than the bidirectional versions. Figure 4-18, Figure 4-19, Figure 4-20, and Figure 4-21 show graphical representations of the unsuccessful rate in low-density, t-trap, high-density and doors workspaces, respectively.

Path length and short path tests

In this section, the path length is tested for all variations in low, T, High and Doors workspaces. And a method for shortening the generated RRT path is proposed.

The generated path of RRT usually a tortuous path and has many points and sharp curves. The proposed algorithm makes the path shorter in the length by omitting the useless points. It tries to replace multi-segments by one straight segment when it is possible. The generated path is not the optimal, neither the shortest one, but, it has fewer vertices and much more straightforward. Figure 4-22 shows the original path generated by RRT (a) and the shortened path (b).



(a) (b)

Figure 4-22: The shortening path algorithm, (a) the original RRT path (38 point, length =18.13), (b) the shortened path (6 point, length= 14.2)

The algorithm pseudo code is shown in Figure 4-23. The algorithm tests the connection between the first points of the path with the last points directly, if a connection exists without a collision, it deletes the midpoints between these two locations. In case of failure the algorithm tries to connect the next vertex of the path (*testing point*) to the last one (*tested points*). It repeats this process until the testing point is reached the last vertex in the path, in this case, the algorithm starts again from the first point and tries to connect to the previous vertex of the last one. The algorithm stops when the tested points reach the first vertex.

The *collisionCheck* function is used to check the collisions between the obstacles and the segment from *pnt1* to *pnt2* location.

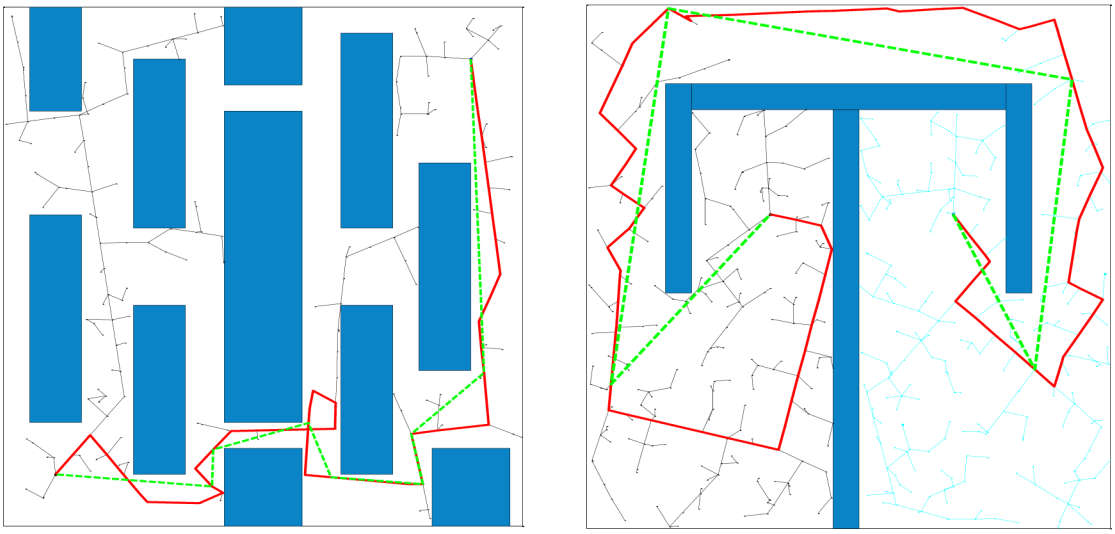
The generated path is a path has fewer vertices and segments. It is not the optimal one, because, it is based on the original path, which generated by RRT.

Figure 4-24 shows the short path in two different workspaces. The thick line represents the generated RRT path, while the dashed one represents the shortened path.

```
Shortening RRT Path Algorithm.
Input: The RRT's path.
Output the shorten path.
```

```
LastTestedPntInd =index of last point in the path;
WHILE (LastTestedPntInd ~= 2 )
    pnt2 = path.get(LastTestedPntInd);
    FOR (floatPntInd=1;floatPntInd<LastTestedPntInd-1;floatPntInd++)
       pnt1 = path.get(floatPntInd);
       IF  ~collisionCheck(pnt1,pnt2)
          path = path.remove(floatPntInd, LastTestedPntInd);
          LastTestedPntInd=updateInd(LastTestedPntInd);
          LastTestedPntInd = LastTestedPntInd-1;
          BREAK;
       END
    END
END
```

Figure 4-23: Shortening path algorithm.



(a) (b)

Figure 4-24: The original RRT path (thick-red), and the shortened path (dashed-green)

The path length tests are conducted in the four workspaces. In the first workspace, the low-density obstacle workspace, all variations are tested in order to estimate the difference between these variations. The numerical results are listed in Table 4-10, and a graphical representation of them is shown as boxplots in Figure 4-25. The results show that the best method based on path length is the unidirectional Bias RRT.

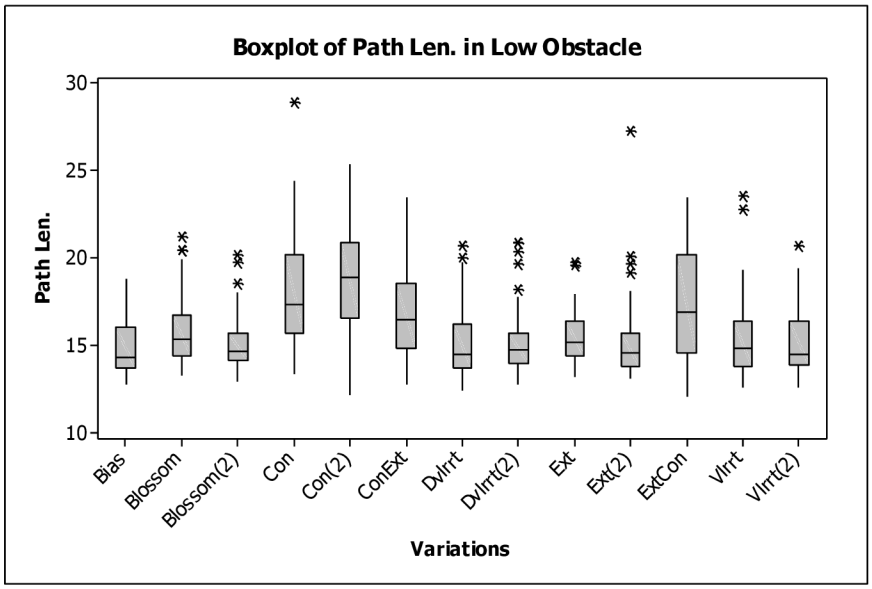


Figure 4-25: Path length boxplots in low density of obstacle workspace

Table 4-10: Path length results in low density of obstacle workspace. The best results marked by (*), the (2) indicate a bidirectional method

	Path Median	Path Min	S-path Median	S-Path Min	Rate %
BIAS	***14.336**	12.770	11.856	11.478	17.30
BLOSSOM	15.373	13.316	11.817	***11.471**	23.13
BLOSSOM (2)	14.644	12.947	11.787	11.490	19.51
CON	17.359	13.366	14.085	11.553	18.86
CON(2)	18.880	12.210	14.195	11.532	24.81
ConExt(2)	16.534	12.796	12.010	11.556	27.36
EXT	15.189	13.235	11.831	11.473	22.11
EXT(2)	14.604	13.155	***11.810**	11.496	19.13
ExtCon(2)	16.929	***12.062**	12.058	11.530	***28.77**
DVLRRT	14.540	12.444	11.834	11.504	18.61
DVLRRT(2)	14.773	12.808	11.862	11.499	19.70
VLRRT	14.846	12.565	11.946	11.565	19.53
VLRRT(2)	14.545	12.629	11.846	11.476	18.56

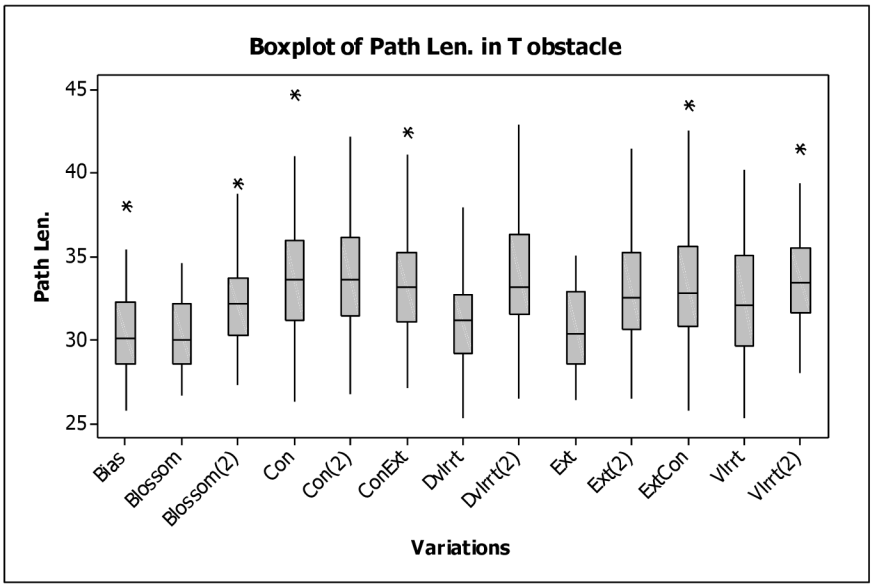


Figure 4-26: Path length boxplot in T obstacle workspace

Table 4-11: Path length in T obstacle workspace. The best results marked by (*), the (2) indicate a bidirectional method

	Path Median	Path Min	S-Path Median	S-Path Min	Rate %
BIAS	30.124	25.814	***24.089**	22.309	20.03
BLOSSOM	***29.976**	26.635	24.488	22.191	18.31
BLOSSOM (2)	32.166	27.269	24.498	22.643	***23.84**
CON	33.658	26.281	25.934	22.277	22.95
CON(2)	33.618	26.742	25.831	22.462	23.16
ConExt(2)	33.170	27.120	25.427	***21.740**	23.34
EXT	30.385	26.432	23.908	22.491	21.32
EXT(2)	32.562	26.460	25.091	22.127	22.94
ExtCon(2)	32.815	25.764	25.378	22.050	22.66
DVLRRT	31.148	25.324	25.104	22.522	19.40
DVLRRT(2)	33.177	26.489	25.831	22.367	22.14
VLRRT	32.041	***25.318**	25.906	22.534	19.15
VLRRT(2)	33.436	28.006	26.165	23.233	21.75

In the T-obstacle workspace, the best result is recorded by blossom method based on the median value of the path length. As shown in Table 4-11. The boxplot of these results is plotted in Figure 4-26.

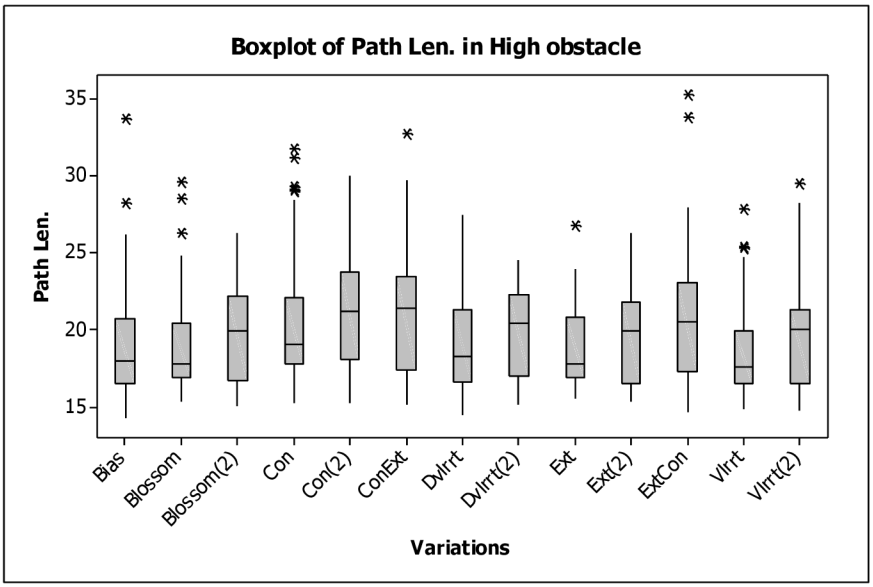


Figure 4-27: Path length boxplot in high density of obstacle workspace

Table 4-12: Path length of high density of obstacle workspace. The best results marked by (*), the (2) indicate a bidirectional method

	Path Median	Path Min	S-Path Median	S-Path Min	Rate %
BIAS	17.911	***14.233**	15.045	13.269	16.00
BLOSSOM	17.766	15.349	***14.70**	13.353	***17.26**
BLOSSOM (2)	19.879	14.977	16.716	13.321	15.91
CON	19.004	15.260	16.697	13.252	12.14
CON(2)	21.144	15.253	17.541	13.498	17.04
ConExt(2)	21.363	15.079	18.164	***13.191**	14.97
EXT	17.752	15.542	14.702	13.286	17.18
EXT(2)	19.947	15.328	17.277	13.222	13.39
ExtCon(2)	20.535	14.613	17.109	13.393	16.68
DVLRRT	18.263	14.415	16.185	13.236	11.38
DVLRRT(2)	20.350	15.087	17.376	13.479	14.61
VLRRT	***17.528**	14.846	14.919	13.347	14.88
VLRRT(2)	20.022	14.730	16.832	13.412	15.93

In the high obstacle workspace, the RRT variations are tested and the results are listed in Table 4-12. In addition, the boxplot representations of these results are shown in Figure 4-27. The best method's result in terms of the median of the path length is achieved by Vlrrt method.

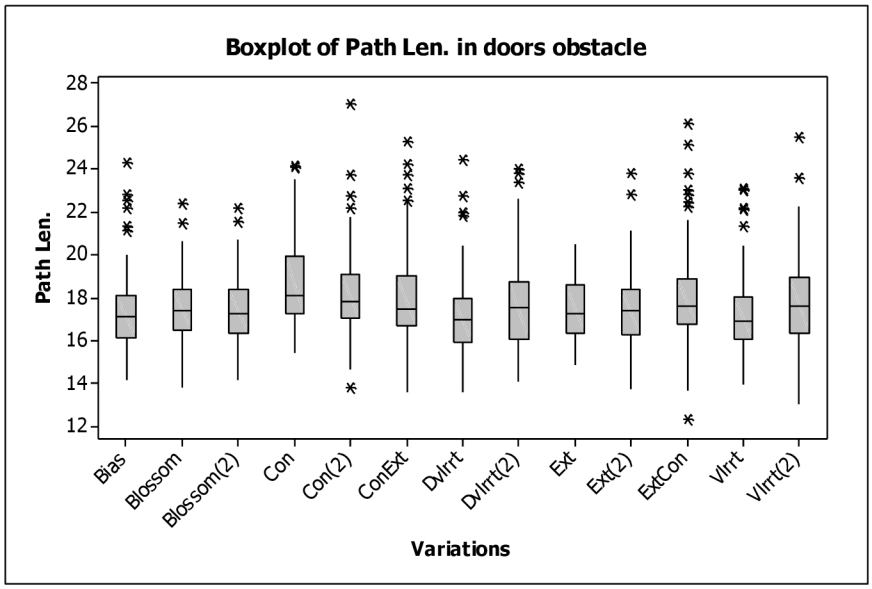


Figure 4-28: Path length boxplot in doors obstacle workspace

Table 4-13: Path length in doors obstacles workspace. The best results marked by (*), the (2) indicate a bidirectional method

	Path Median	Path Min	S-Path Median	S-Path Min	Rate %
BIAS	17.089	14.145	14.285	11.826	16.41
BLOSSOM	17.389	13.787	***13.960**	11.771	***19.72**
BLOSSOM (2)	17.256	14.153	14.127	***11.740**	18.13
CON	18.072	15.407	14.883	11.869	17.65
CON(2)	17.810	13.757	14.792	11.797	16.95
ConExt(2)	17.422	13.574	14.749	11.930	15.34
EXT	17.263	14.872	14.157	11.802	17.99
EXT(2)	17.413	13.685	14.304	11.764	17.85
ExtCon(2)	17.583	***12.278**	14.524	11.799	17.40
DVLRRT	16.934	13.604	14.250	11.850	15.85
DVLRRT(2)	17.532	14.104	14.134	11.787	19.38
VLRRT	***16.878**	13.957	14.411	11.870	14.62
VLRRT(2)	17.577	13.046	14.165	11.835	19.41

In the last workspace, the door obstacle, the tests show that Vlrrt has the lowest median of the path length. Table 4-13 shows the numerical results of the tested methods, while Figure 4-28 shows the boxplot representation for these results.

Based on the results in all testing workspaces, the unidirectional tree methods generally, give better results than the bidirectional trees do. The reason of this difference is the extending procedure of RRT, where in unidirectional tree the expansion is done from the nearest node in the tree, while in bidirectional cases, the path is composed of two paths, which make it longer than unidirectional path.

The results also show that using shortening method reduces the path length in the average of 13% – 28%, depending on the testing environment, the obstacles shape, and the methods.

Summary

In this work, many approaches of RRT were tested in four different workspaces, some statistical analyses have been done to support our decision about using one variation instead of the others. In addition, we proposed a shortening algorithm to reduce the length and the tortuous of the RRT paths.

We conclude that regarding to the time of execution, in low-density obstacles the Vlrrt(2) method gives the best result. It has the rate of 100% of success. For the T obstacle workspace, Vlrrt achieves the best result, however, it has one fail. So we choose to use the Vlrrt(2) based on the statistical result, which shows that there is no sufficient difference in these two variants with a confidence level of 95%. In high-density workspace, the best variant is Con(2) method, and in the last workspace, the best variant is Dvlrrt(2).

4.1.3 Rapidly-Exploring Random Trees: 3D planning

In this work (Abbadi, Matousek, et al. 2012) the RRT algorithm is applied in three-dimensional workspace to find a path for a holonomic system. We also developed an algorithm for path shortening. This algorithm shortens the path by omitting unnecessary points from the original path. Furthermore, we present a smoothing-out technique for real dynamic behavior.

The result of this work can be applied in many applications, e.g. the robot arms, the flying objects, CNC machine, 3D laser cutting machines, and other machines that work in 3D dimension.

Proposed methods

The generated path using RRT is a tortuous path. It has many nodes and sharpness edges. We try to shorten the RRT path and make it as smooth as possible by removing useless points. We introduce an algorithm in (Abbadi, Matousek 2012). It generates a shortened path based on the original one. A new version is shown in Figure 4-30.

The algorithm tries to connect vertices from both path's edges and delete the midpoints between them. The updated version tests the path from two directions and returns the shortest one.

The original tortuous path that is generated by RRT is shown in Figure 4-29, in addition to the first shortened path that starts from first toward the last point, and the second shortened path, which start from last toward the starting point of the original path.

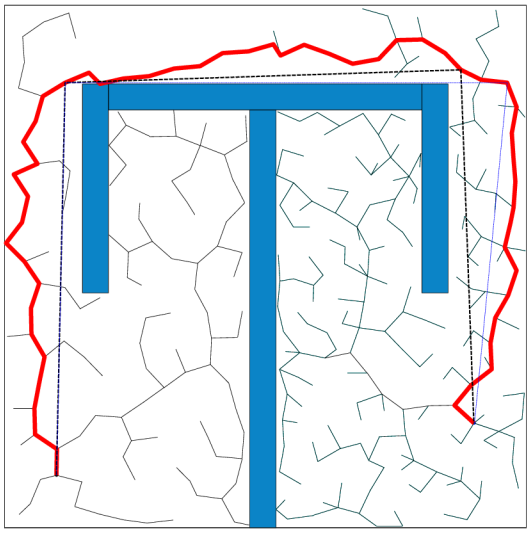


Figure 4-29: The shortening algorithms results. The solid red line represents the original RRT path, the (black - -) line represent the first shortened path, and the (blue - .) line represent the second shortened path

```
1.  EndPnt ←index of last point in the path;
2.  StartPnt ←1;
3.  tmpPath1←originalPath;
4.  WHILE (EndPnt ~= 2 )
5.     pnt2 ← tmpPath1 (EndPnt);
6.     FOR (StartPnt ←1 ; StartPnt < EndPnt -1 ; StartPnt ++ )
7.        pnt1 ← tmpPath1 (StartPnt);
8.        IF  ~collisionCheck(p1,p2)
9.           tmpPath1 ← tmpPath1 (1 to StartPnt)
10.            + tmpPath1(EndPnt to the end);
11.          Endpnt ← index of previous point to EndPnt;
12.          BREAK;
13.       END
14.    END
15. END
16. tmpPath2← originalPath;
17. startpnt←1;
18. WHILE (StartPnt < tmpPath2's size)
19.    pnt1 ← tmpPath2 (StartPnt);
20.    FOR (EndPnt ←tmpPath2's size ; EndPnt > StartPnt +1  ; EndPnt -- )
21.       pnt2 ← tmpPath2 (EndPnt);
22.       IF  ~collisionCheck(p1,p2)
23.          tmpPath2 ← tmpPath2 (1 to StartPnt)
24.            + tmpPath2(EndPnt to the end);
25.          StartPnt ← Startpnt+1;
26.          BREAK;
27.       END
28.    END
29. END
30. IF (length of tmpPath1< length of tmpPath2 )
31.   RETURN tmpPath1;
32. ELSE
33.   RETURN tmpPath2;
34. END
```

Figure 4-30: The shortening path algorithm

A smoothing-out technique is applied to the shortened path using Catmul-Rom spline (Catmull et al. 1974), as shown in Figure 4-31. Advantage of this solution is also for the future extension of the path planner. The shorter and smooth path is more convenient for a dynamic ride of the real vehicle.

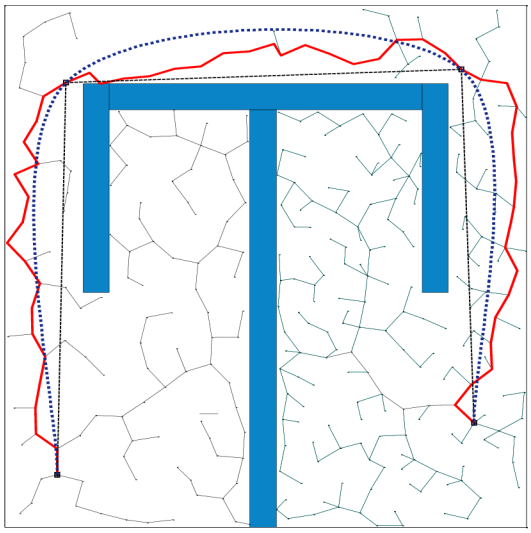


Figure 4-31: The smoothed path (bold ...) which generated based on the shortened path (- -)

There are several approaches to spline design. Catmul-Rom spline is a special kind of Hermite spline. The spline is a sequence of curves joined together to form a larger curve. These curves pass through given points smoothly and continually.

Hermite spline method calculates the curve using two points and tangents vectors in these points as shown in Figure 4-32, (Shikin et al. 1995; Salomon 2011).

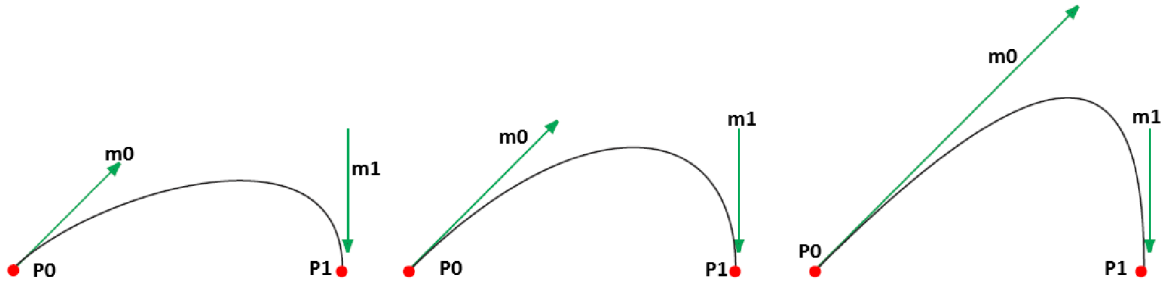


Figure 4-32: Hermite spline principle, and the effect of the tangent magnitude, p_0, p_1 are the start and the end points, m_0, m_1 are the corresponding tangents in the points, Source: (Salomon 2011)

The curve P(t) is calculated using the following equation

$$\mathrm{P}(t) = (2t^3 - 3t^2 + 1)p_0 + (t^3 - 2t^2 + t)m_0 + (-2t^3 + 3t^2)p_1 + (t^3 - t^2)m_1,$$
$$0 \leq t \leq 1$$

Where p_0, p_1 is the given points, m_0, m_1 is the tangents vectors, t is the knots parameter in the intervals [0,1].

Using the matrices notation, the previous equation is written as follows.

$$\mathrm{P}(t) = \mathrm{T}(t)\mathrm{HB} = (t^3 \quad t^2 \quad t \quad 1)\begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} p_0 \\ p_1 \\ m_0 \\ m_1 \end{pmatrix}$$

Where the H matrix is called a Hermite basis matrix.

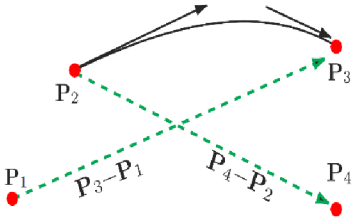


Figure 4-33: Catmull-Rom calculation method, Source: (Salomon 2011)

In Catmul-Rom case, four points are used to generate the curve on the segment P_2P_3, i.e. [P_1, P_2, P_3, P_4], where the tangent on P_2 is parallel to the P_3P_1 segment, and the tangent in P_3 is parallel to the P_4P_2 segment, as shown in Figure 4-33. Based on these settings the equations is written as follows

$$p_0 = \mathrm{P}_2, \quad p_1 = \mathrm{P}_3, \quad m_0 = s(\mathrm{P}_3 - \mathrm{P}_1), \quad m_1 = s(\mathrm{P}_4 - \mathrm{P}_2)\,, s \in R^+$$

$$\mathrm{P}(t) = (t^3 \quad t^2 \quad t \quad 1)\begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \mathrm{P}_1 \\ \mathrm{P}_2 \\ s(\mathrm{P}_3 - \mathrm{P}_1) \\ s(\mathrm{P}_4 - \mathrm{P}_2) \end{pmatrix}$$

$$= (t^3 \quad t^2 \quad t \quad 1)\begin{pmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}\begin{pmatrix} \mathrm{P}_1 \\ \mathrm{P}_2 \\ \mathrm{P}_3 \\ \mathrm{P}_4 \end{pmatrix}$$

The tension parameter *s* is used to change the magnitude of the tangent vectors. The effect of these changes is shown in Figure 4-32. In Catmull-Rom method, the *s* parameter is fixed and has the value (0.5).

For multiple points, [P_1,P_2,…,P_m], the Catmull-Rom curve is calculated for every segment P_i,P_{i+1} using four points [P_{i-1},P_i,P_{i+1},P_{i+2}]. The points' sets that generate the curves are overlapped, i.e. [P_1,P_2,P_3,P_4], [P_2,P_3,P_4,P_5], …, [P_{m-3},P_{m-2},P_{m-1},P_m] .

A problem rises up because of spline algorithm; the smoothed line sometime collides the near obstacles, and that is because the smoothing algorithm does not check the generated path if it collide or not, moreover the Catmull-Rom method generate uncontrollable curves. Because of this problem, the algorithm is re-implemented and the local-spline is proposed. It smoothes the path around the corners, which means the path will be kept straightforward, but only sharp edges will be smoothed.

To implement the local spline, two points on the path near the corner are used. These points are taken far from the corner by *d* distance, where *d* is chosen depending on the

kinematic and dynamic constraints. These points in addition to the corner point are passed to the smoothing algorithm to generate a path around the corners. Figure 4-34 shows how the normal spline collides with walls and how the new local-spline works. However, this method reduces the collided points, but it still needs more checking for collision.

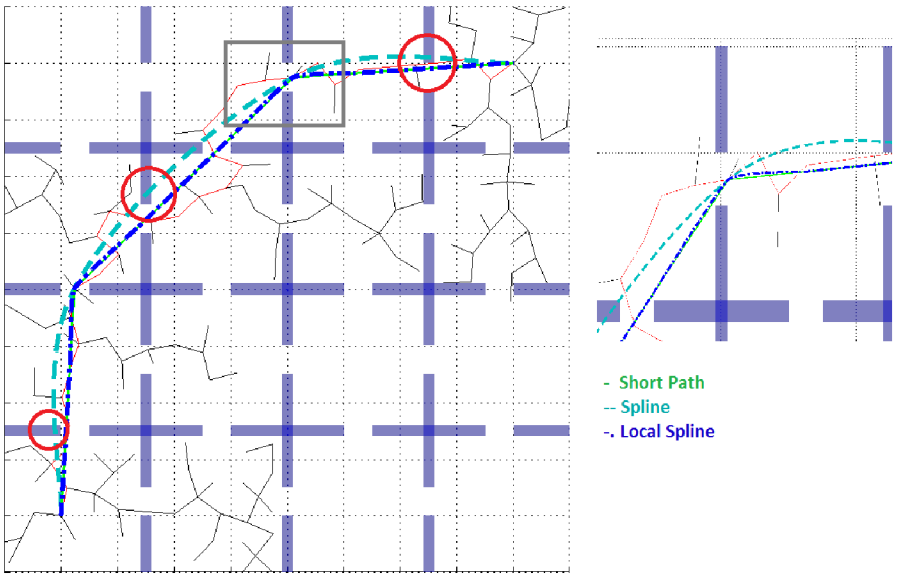


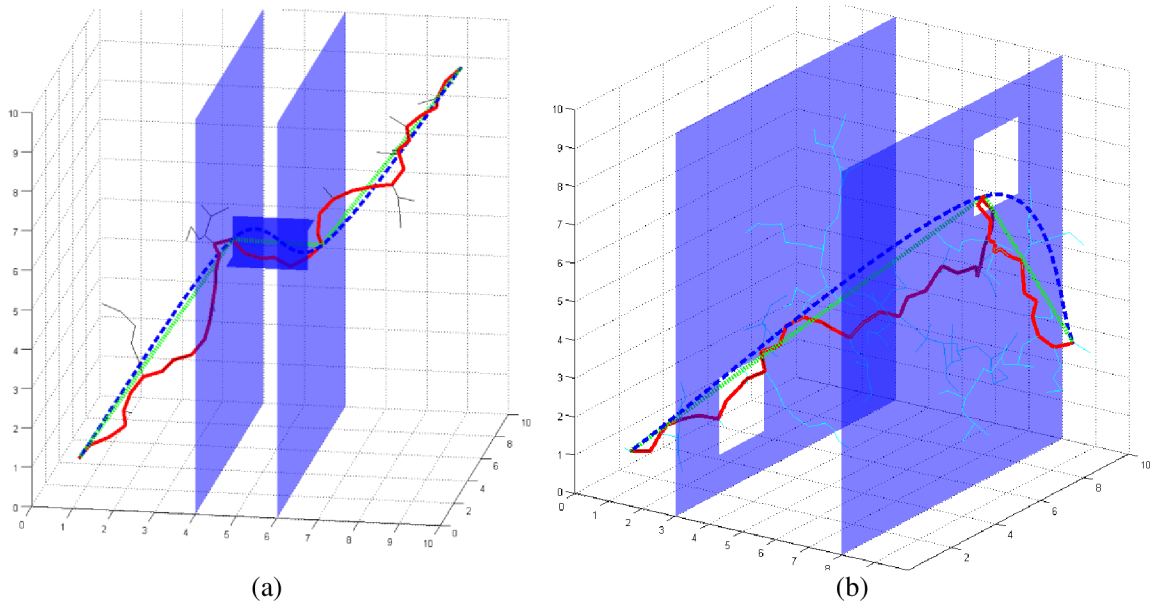
Figure 4-34: Local spline. The dashed line represents the spline path, the (.-) path represents the local-spline path

In literatures, many researches deal with the smoothing problem, for example, in (Kito et al. 2003), the authors proposed a global path generation method. It is based on the visibility graph, and it re-arranges the path to be as a sequence of sub-goals (middle points). Then it constructs a graph for the smoothed paths. Another work is presented by (Yang 2013) for smoothing non-holonomic path planning, they proposed the Spline-based Rapidly-exploring Random Tree (SRRT) algorithm. It uses the cubic Bezier splines as a local planner to connect two states, which replace the dynamic simulation of RRT by parameterization of the cubic Bezier splines. Another research on non-holonomic domain proposed a real-time method for re-planning the path and smoothing it. The smoothing step is achieved by selecting appropriate sequences of alternating trims and maneuvers from a precomputed library of motion primitives (Bottasso et al. 2008).

Testing Environments

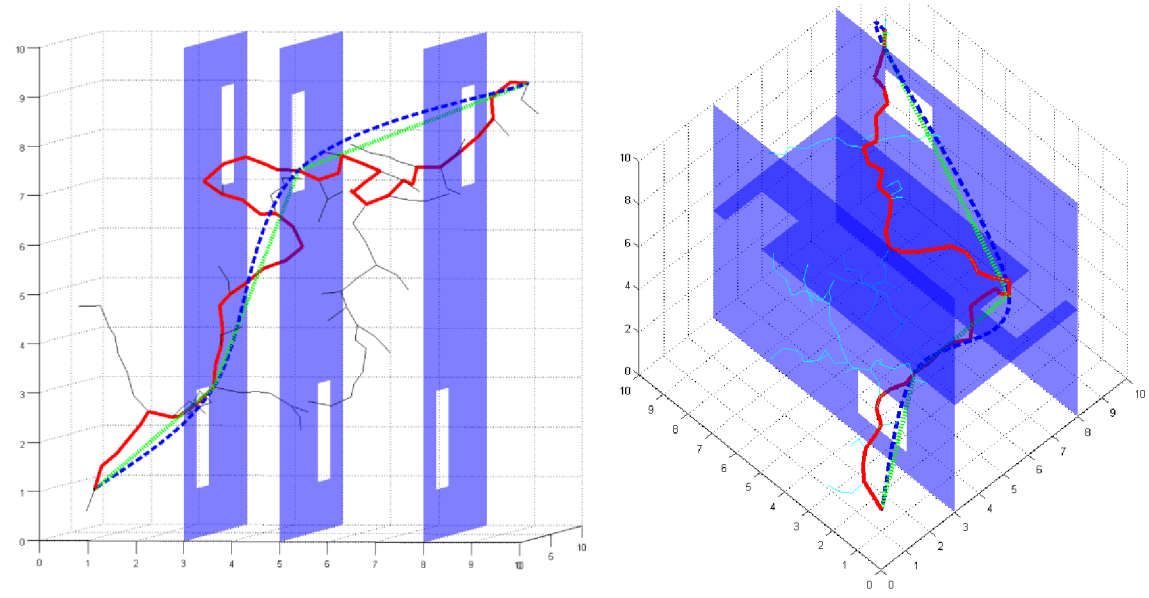
We have constructed four testing scenarios. The first one involves a wall has a passage as shown in Figure 4-35-a. This obstacle evaluates the algorithm efficiency in simple narrow passage scenario. The second workspace involves two walls where each one has a window as shown in Figure 4-35-b.

The third workspace has three walls with windows in different locations as shown in Figure 4-36-a. The last scenario has vertical and horizontal obstacles with different locations of the windows as shown in Figure 4-36-b.



(a) (b)

Figure 4-35: (a) The narrow passage workspace Wall 1, (b) the different windows location workspace Wall 2



(a) (b)

Figure 4-36: (a) Multi-walls and different windows location workspace, Wall 3, (b) the horizontal and vertical walls workspace, Wall 4

Results

We have tested six variations of RRT the basic RRT (*Ext*), *Blossom*, *Vlrrt* and the bidirectional versions of them.

The tests were executed for every method 100 times per scenario. The testing platform was as follows, a PC equipped with Intel Core2Duo CPU 2.53 GHz, and 2 GB of memory, and Windows7 64-bit is used. The algorithms have been implemented in Matlab environment. We consider the RRT failed to reach the goal after 2000 attempts to grow a branch.

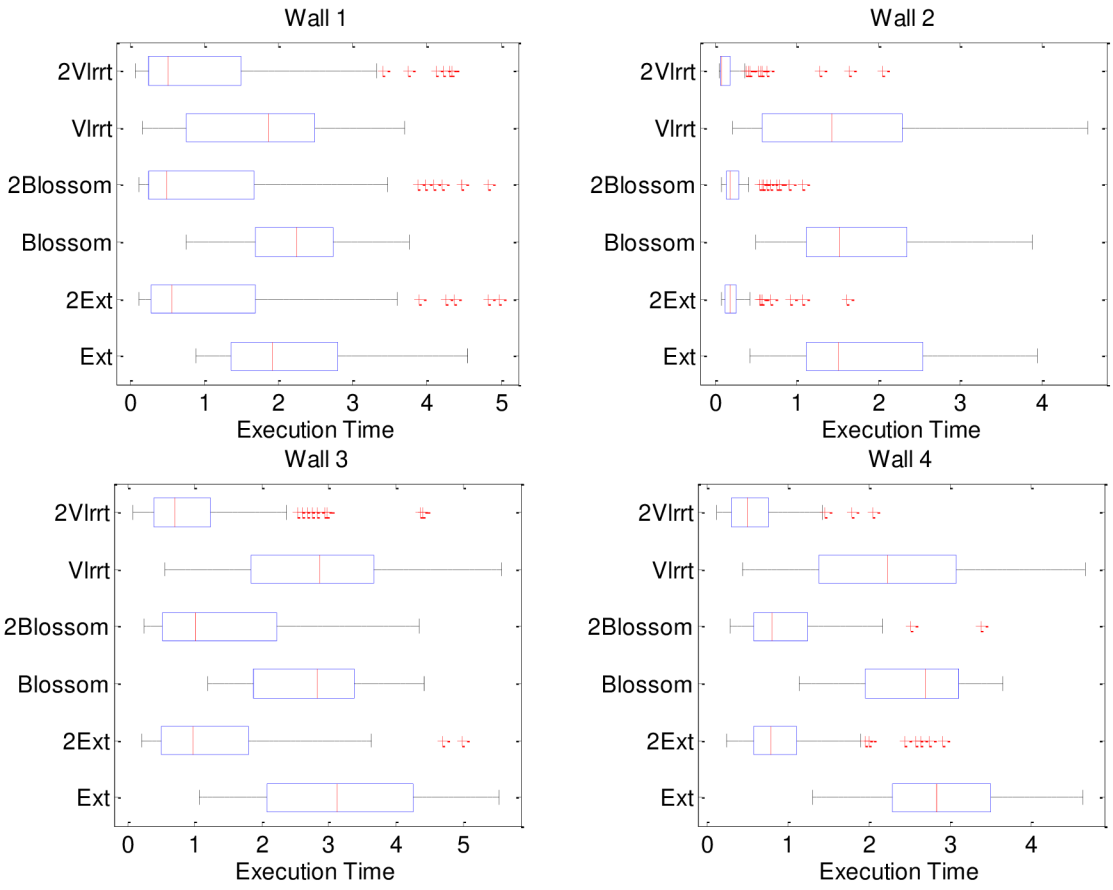


Figure 4-37: Boxplots for RRT variations based on an average time of executing

The numerical results in every testing method are shown in Table 4-14. They represent the average of execution time for successful tries to find a path. In addition, the boxplot representation is shown in Figure 4-37.

The results show that using bidirectional-trees are better than unidirectional methods where these methods has the lowest average of execution time to find a result, they also more probabilistically complete, as shown graphically in Figure 4-38, which shows the number of failed tries to find a path.

The tested algorithms have some difficulties to find a solution in a narrow passage, where even the bidirectional approaches failed to find a solution in some tests, as shown in Figure 4-38-wall1.

Table 4-14: The average execution time for the successful tries of the RRT

	Wall 1	Wall 2	Wall 3	Wall 4
Ext	2.16	1.83	3.17	2.87
2Ext	1.16	0.24	1.29	0.95
Blossom	2.16	1.74	2.76	2.56
2Blossom	1.08	0.26	1.40	0.96
Vlrrt	1.73	1.57	2.80	2.27
2Vlrrt	**1.05**	**0.20**	**0.99**	**0.57**

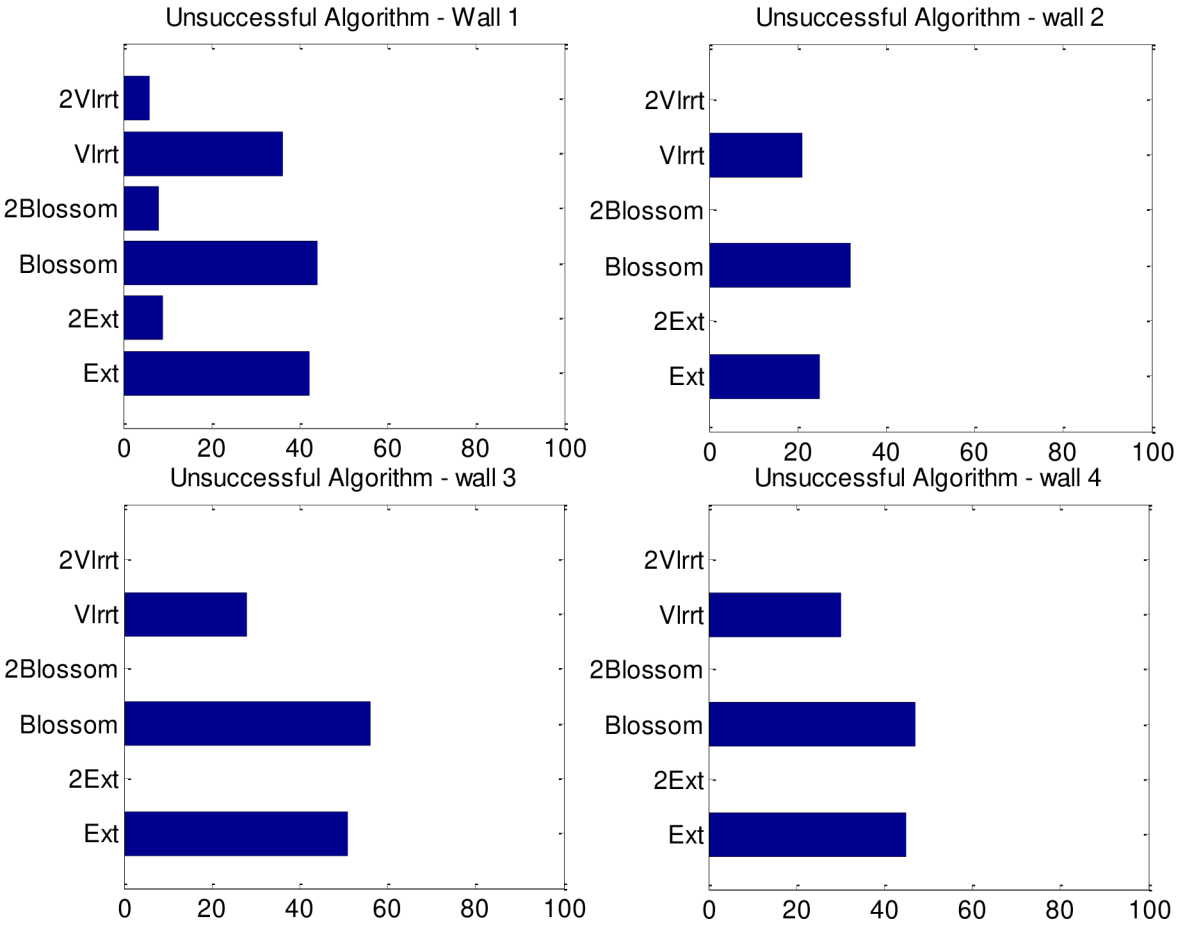


Figure 4-38: Unsuccessful attempts to find a path in 100 tests per scenario

The result shows that the bidirectional Vlrrt (*2Vlrrt*) is better than the other methods, where the successful branching increases the extension step that makes the tree spreads faster in free spaces, and performs a fine exploration around the obstacle.

The advance of *2Vlrrt* can be inferred also from the boxplot representation of the number of nodes in the generated path, see Figure 4-39. *2Vlrrt* has a small path size that means the path has less curves, which in consequence indicates that the path is more straightness and has longer branches than the other generated paths.

The Vlrrt method's performance is affected by the increase or the decrease of the branching length factors. In these tests, the incremental factor is set to add 20% of the current node's extension step. The decrement factor, in case of branching fail, is reset to be as the original extension step (e=0.5).

In the *Blossom* algorithm, an optimization of the blossom distance (*Blossom_Dis*) variable is tested to find the best performance in the given situations. The optimization is based on the effect of *Blossom_Dis* on the blossom method. If the distance is very small, the *blossom* RRT will behave like the basic RRT, in the opposite, if it set to a large value, the old branches will block the new ones, which cause a failure to navigate through the

small area, windows, and narrow passages. The used value of *Blossom_Dis* in our test is set to be equal to *e**1.25, where the *e* is the original extension distance.

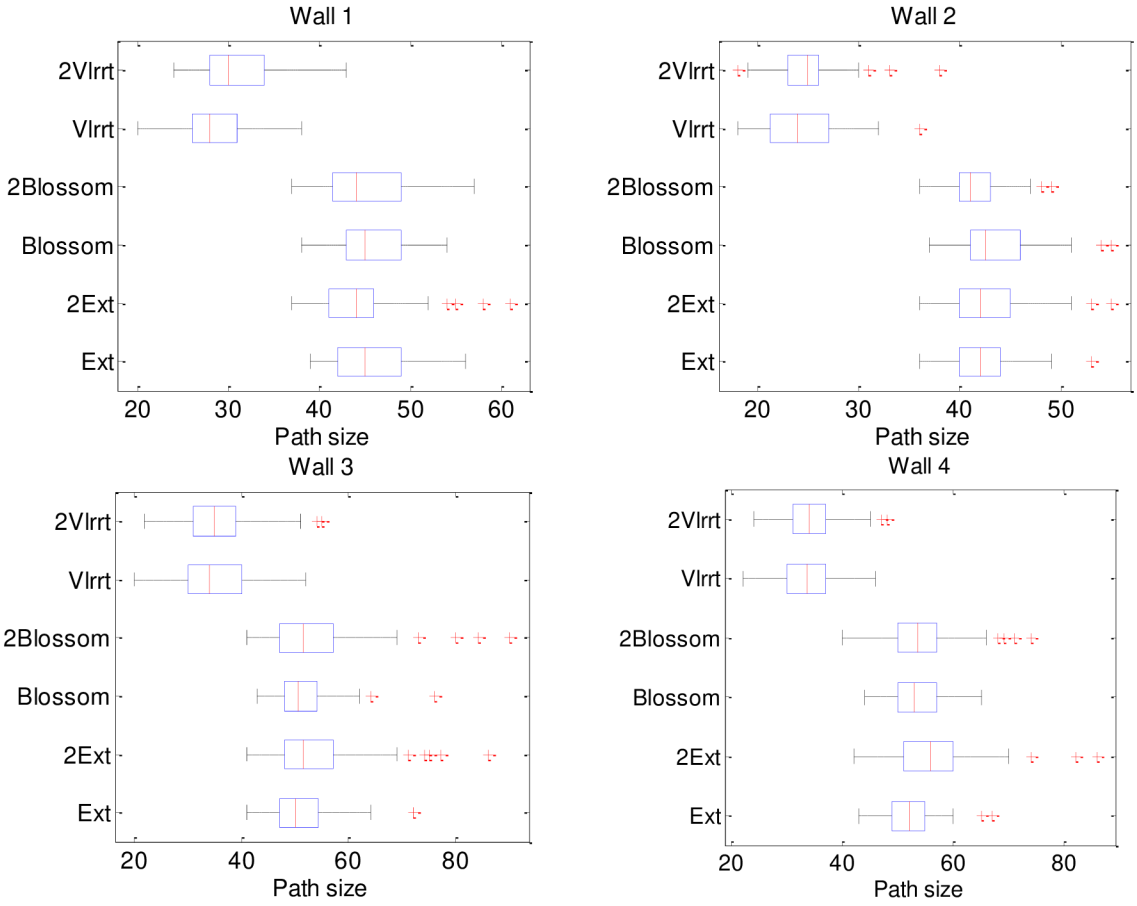


Figure 4-39: Boxplot of path size in the tested workspaces

We make the *Blossom_Dis* optimizing in the Wall2 scenario, and the result is shown in Figure 4-40, where mean, median and median filter are drawn. The median curve is smoothed by the median filter, for a better estimation. The best estimation based on the curve is between 0.2 and 1.5 and the MIN value of median is when *Blossom_Dis* = 1.1 which corresponding to execution time ~= 0.174.

In the same way, another optimization for the extension step *e* in two different workspaces is tested. The results show that the optimal value of Wall 2 scenario is between 1.25 ~ 3.25 as shown in Figure 4-41-b, and the optimal one in the Wall1 located in the range between 0.5~1, Figure 4-41-a.

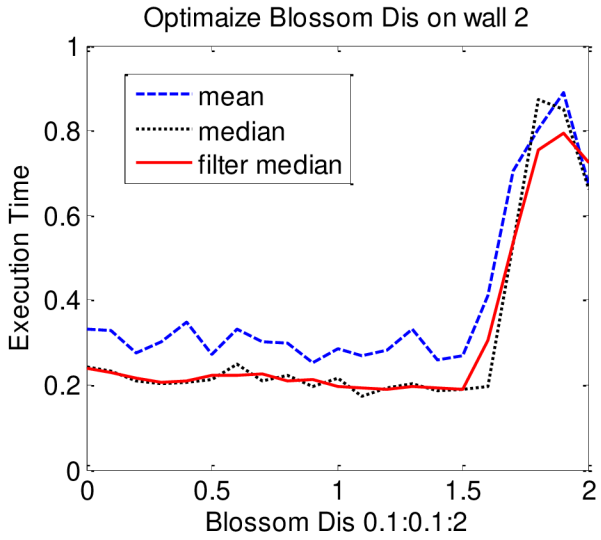
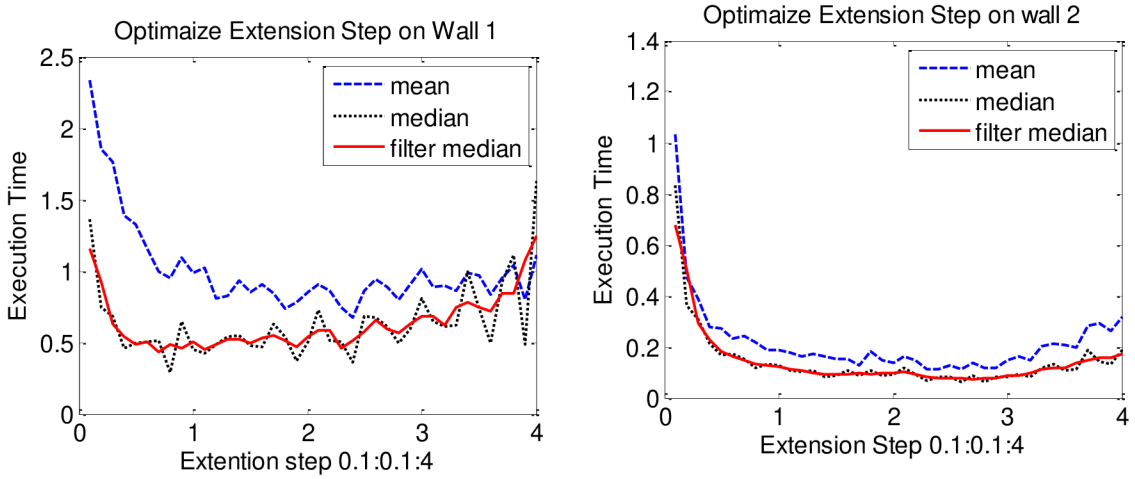


Figure 4-40: *Bloosom_Dis* optimization in Wall2 scenario



(a) (b)

Figure 4-41: Extension step optimization in the wall1 scenario (a), and in the Wall2 scenario (b)

Summary

In this work, we test RRT algorithms in three-dimensional workspaces. In addition, an algorithm for shortening the path is introduced, and a smoothing-out technique to the shortened path has been presented.

The results show that using two trees is better than using one tree in all scenarios. Several failures of finding the path have occurred in the narrow passage scenario.

The *2Vlrrt* is better than the other methods in terms of execution time to find the path and the generated path has fewer points than the others do.

We conducted several tests to optimize the parameter of *Blossom* RRT and to optimize the length of extension; the results were different depend on the environment and scenario. In the future works, narrow passage environments needs more study to find efficient methods and avoid algorithm failures.

4.1.4 Spatial Guidance to RRT Planner using the Cell-Decomposition Algorithm

In this work (Abbadi, Matousek, et al. 2014), we made a comparison between the probabilistic path-planning method, i.e. RRT and the spatial planner, i.e. exact cells-decomposition algorithm.

A new test is proposed to make some tradeoff between the efficiency of planning using CD in 2D space and the planning in dynamic space using RRT. The proposed method uses the path's points of the CD inside the RRTs planner as a spatial guidance.

Problem formulation and proposed solution

The RRTs as example of randomized algorithms, has a good performance in high dimension or continuous spaces. In general, the limitation of these algorithms is the planning in small areas. In cell-decomposition case, it is efficient in low dimensions planning; however, the building of its graph could be hard in some cases.

The available spatial information and the randomize approaches is combined to overcome the drawbacks in narrow area. The CD is used to produce a primitive path over 2D or 3D workspace and provide this path to the RRTs planner as bias-path. This approach will keep some reasonable balance between dynamic and uncertainties from one side, and optimality, efficiency in spatial planning from the other side. Moreover, the CD can guide RRTs in a small area.

In order to show the difference between these two planners we make some tests in two scenarios. The first one is a simulation of offices and corridors architecture-schema and the other is the typical issue for RRTs, which is a small area and narrow passage.

Results

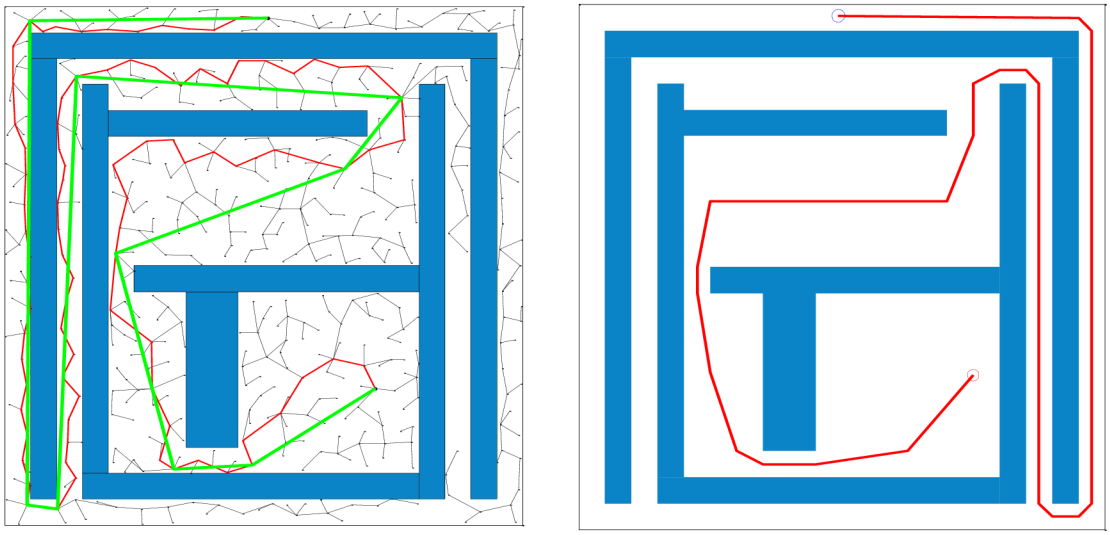
We repeat the test 100 times for RRTs in every scenario and take the mean of the results for the successful tries to reach the goal. In each run, RRTs planner is setup as follows. The extending length set to (e = 0.5). The tests are repeated based on the RRT iteration, where the RRT is considered failed to reach the goal after {3000, 5000, 10000, and 100000} tries to grow a branch. We use Intel Xeon(R) PC with CPU of 2.67 GHz, and 6 GB of memory, and Windows 7 64-bit. We implement the algorithm in MATLAB environment.

The CD planner uses the Dijkstra's algorithm for search on the graph. The Dijkstra in this case has $O(\log(N)E)$ time complexity. Where N represents the nodes number and E is the edges number in the graph.

In the first workspace (building-like scenario), a simulation was lunched for a path planning, and the results are listed in Table 4-15.

The results show the generated nodes number in both CD and RRT cases. In CD case the nodes number represents the number of graph's nodes, which is constant. While in RRTs case, the node numbers are taken as an average of the results in a 100 times of repeated tests.

The results show that the RRT algorithm is probabilistically complete when the iteration approaching 10000 in this workspace. In addition, the results show that CD algorithm is faster than RRTs in complete case by 7.6032 times.



(a) (b)

Figure 4-42: Path planning in building-like workspace using (a) RRT, (b) CD algorithm

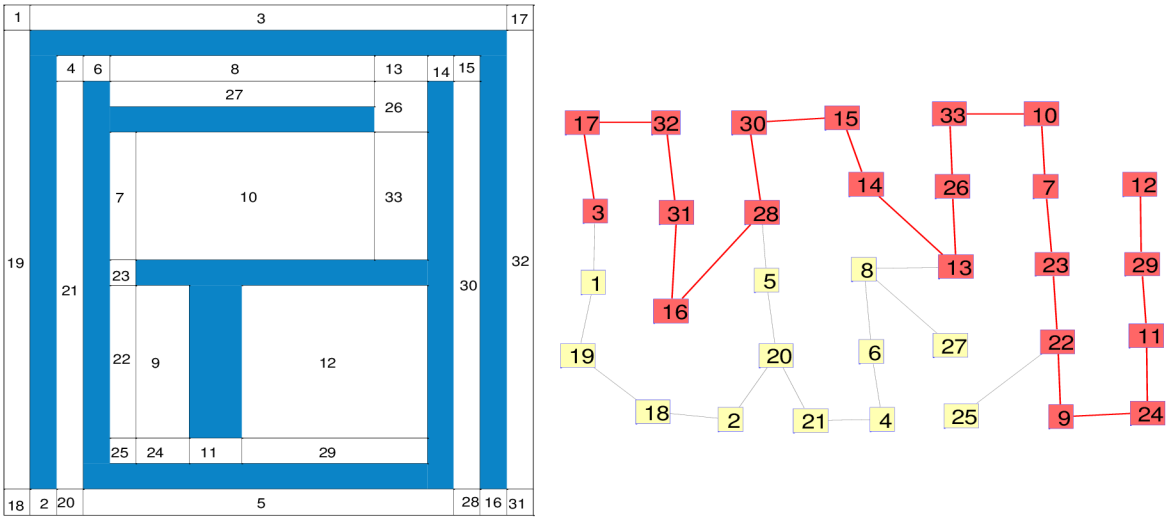
Table 4-15: Test results of building-like scenario, the numbers in time fields (), represent the percent of RRT's time comparing to CD's time

	Nodes number	Preparing Time	Planning Time	Total time when success	Path Length	Time when fail	Successful
CELL Dec.	33	0.3152	0.0056(1)	0.3208(1)	40.92	-	100 %
RRTs(3K)	478.17	0	1.85(331.03)	1.85 (5.77)	43.46	2.0520	12%
RRTs(5K)	547.17	0	2.46(493.17)	2.46(7.67)	41.77	3.2539	90%
RRTs(10K)	540.48	0	2.44(435.55)	2.44(7.60)	42.71	-	100%

The preparing time is the time required to generate the graph in CD. However, it is not required in the RRT case. We can infer based on the planning time that the CD is an efficient planner comparing to RRTs in 2D workspace for non-holonomic movements. In consequence, for repeated task, the CD can be 436 times faster than RRTs. In addition, the graph size in the CD is constant, which makes it applicable for real-time planning.

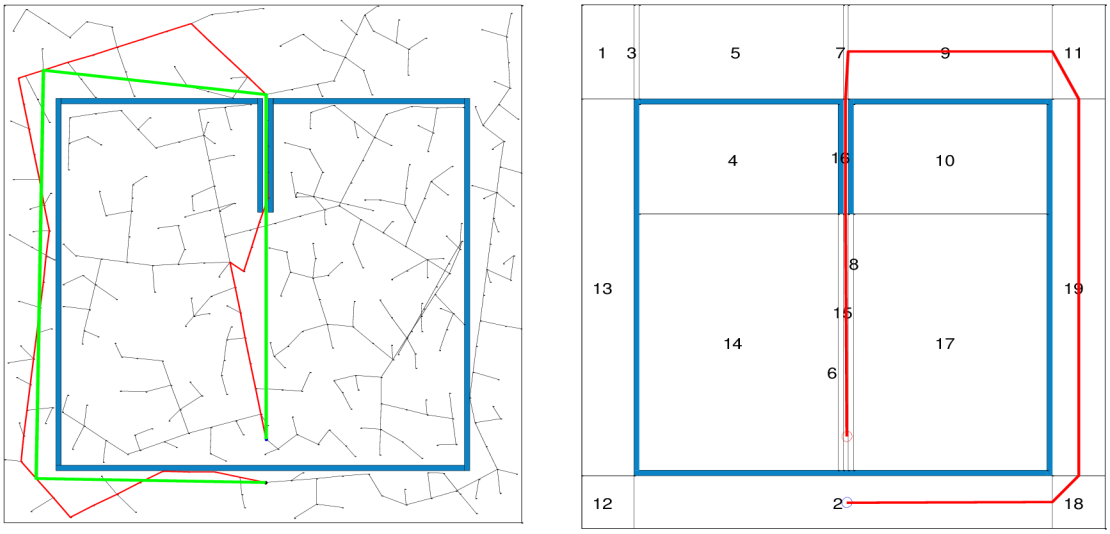
Figure 4-42 shows the testing workspace, which consists of rooms and corridors. The first part Figure 4-42-a shows the solution founded by RRTs planner. While Figure 4-42-b

uses the CD planner to find a path. The generated cells are shown graphically in Figure 4-43-a, and the corresponding CD's graph is also represented graphically in Figure 4-43-b.



(a) (b)

Figure 4-43: (a) The generated calls using CD, (b) the CD's graph of adjacency in building-like workspace. The shaded nodes represent an example of a path between cell 12 and cell 3



(a) (b)

Figure 4-44: Path planning in narrow passage workspace using (a) RRT, (b) CD, and the generated cells

In the second scenario, the narrow passage problem is simulated. The result is shown in Table 4-16. It indicates that the RRT records some failures, even when the iteration-limit set to 100000 iterations.

This workspace is presented in Figure 4-44, where the first figure (a) uses the RRT planner to find a solution. In (b) the generated cells using CD algorithm are shown, in addition to the path between the initial and goal positions, which lay on the cells (2 and 15). The generated graph is shown in Figure 4-45, where the shaded cells represent the path between cells 2 to 15.

Table 4-16: Test results in the narrow passage scenario, the numbers in time fields (), represent the percent of RRT's time comparing to CD's time

	Nodes	Preparing Time	Planning Time	Total time when success	Path Length	Time when fail	Successful
CELL Dec.	19	0.2	0.003(1)	0.203(1)	24.33	-	100%
RRTs(3000)	554.65	0	1.19(441.48)	1.19(5.88)	22.14	2.13	31%
RRTs(5K)	629.66	0	1.59(587.22)	1.59(7.82)	23.17	3.24	41%
RRTs(10K)	644.9	0	1.80(668.11)	1.80(8.90)	23.08	5.82	44%
RRTs(100K)	729.07	0	6.39(2368.40)	6.39(43.90)	23.00	50.44	67%

In order to enhance the RRT planner, a new method was proposed. It tries to exploit the spatial information that provided by CD and guide the RRT growth toward the possible path. The CD's path points are considered as bias points to the RRT trees as shown in Figure 4-46, where the dots represent these points.

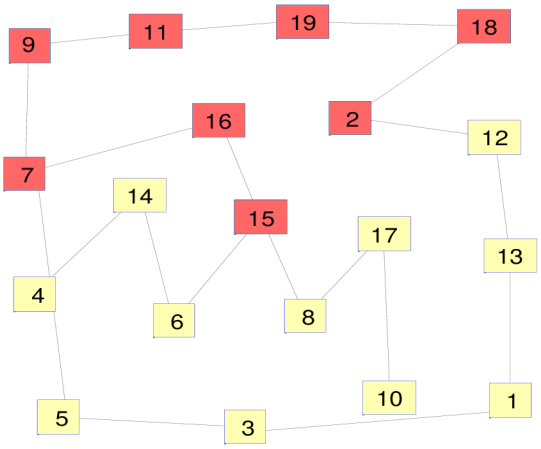


Figure 4-45: The generated graph by CD algorithm in the narrow passage workspace. The shaded nodes represent the corresponding path in the graph between initial and goal cells

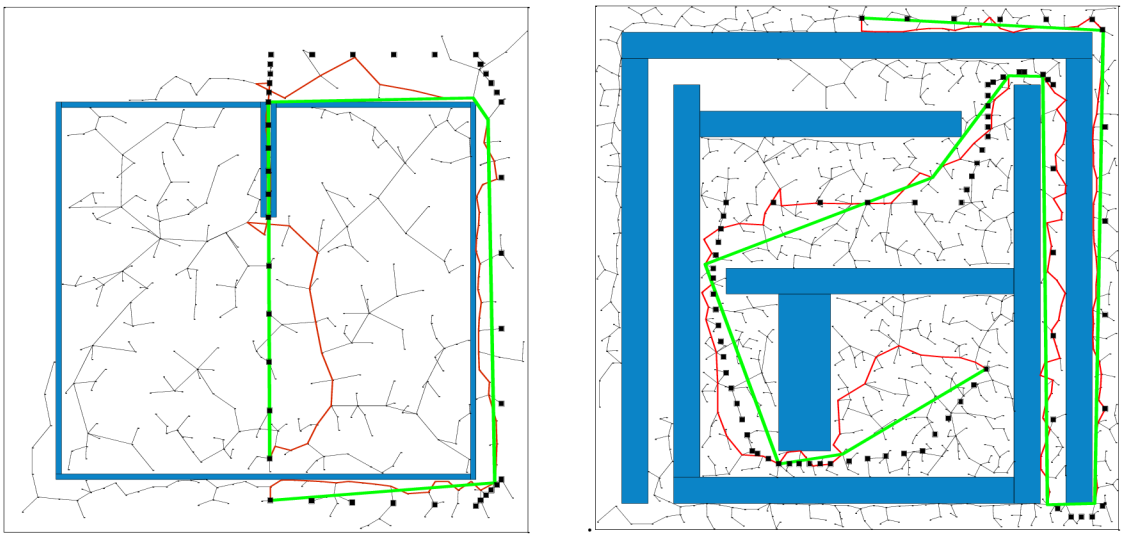


Figure 4-46: Path planning using RRTs with a bias toward CD path points, the dots represent points on CD path

We set the RRTs planner to bias toward these points in the probability of (0.2). The results are listed in Table 4-17, and Table 4-18 for both building-like and narrow passage workspaces, respectively.

Table 4-17: Test results in the building-like workspace, the bias to CD-path's points is equal to 20%, and the numbers in () in planning time fields represent the time reduction percent using the bias

1st scenario	Nodes Num.	Planning Time (without bias)	Planning Time	Successful	Successful (without bias)
RRTs(3000)	428.58	1.8538	1.8189 (-1.9%)	64%	12%
RRTs(5000)	447.1	2.4594	2.0266 (-17.5%)	100%	90%
RRTs(10000)	463.31	2.4391	2.1459 (-12%)	100%	100%

Table 4-18: Test results in narrow passage workspace, the bias to CD-path's points is equal to 20%, and the numbers in (), in planning time fields represent the time reduction percent using the bias.

1st scenario	Nodes Num.	Planning Time (without bias)	Planning Time	Successful	Successful (without bias)
RRTs(3000)	297.10	1.1920	0.5676(-52.4%)	86%	31%
RRTs(5000)	292.72	1.5855	0.5512(-65.2%)	86%	41%
RRTs(10000)	299.96	1.8039	0.6178(-65.8%)	86%	44%

In comparison with the previous results, the bias enhances the RRT algorithm's completeness significantly in all cases. Also in the narrow passage scenario, the time of planning decreases about 52% in worth case while the completeness increases. The success of the planner in a narrow passage workspace using spatial guidance remained at 86%. It is because of another drawback of RRTs, which is the branch blocking. That means the tree's nodes are located near to the narrow area's gate and they take some position where the new branch cannot pass to the passage without colliding with obstacles. The solution for this case can be made by choosing a smaller extension distance. However, that generate a larger number of nodes to construct the tree, which means it increases the computation and memory cost.

Summary

In this work, we test CD algorithm, which construct an adjacency graph of the free workspace cells, in addition, to the RRT planner. The results show that the CD planner finds a path efficiently in static and known environments. The CD is faster than the RRT planner in preparing and planning a path in simple workspace. We test the idea of introducing the spatial information to the RRT planner and it gives a good result. It improves the completeness and the planning time.

This work was a first step to build a hybrid planner, which works efficiently in continuous, high dimension space using the available knowledge and spatial information,

and overcome the drawback of randomized sampling-base algorithms. The future work will focus on using available information to speed up the complex motion planning for robots in uncertainty and dynamic environments.

4.1.5 Collided path replanning in dynamic environments using RRT and Cell decomposition algorithms

In this work, the cell decomposition algorithm is used to find a spatial path in preliminary static workspaces, and then the RRT is used to validate this path in the actual workspace (Abbadi, Prenosil 2015b). Two methods are proposed to enhance the omnidirectional robot's navigation in partially changed workspace. First, the planner creates RRT tree and biases its growth toward the path's points in ordered form. The planner reduces the probability of choosing the next point if a collision is detected, which increases the RRT expansion in the free space. Second method uses a straight planner to connect the CD-path's points. If a collision is detected, the planner places RRT trees in the both sides of collided segment. The proposed methods are compared with others approaches. The simulation shows that the proposed methods have better results in terms of efficiency and completeness.

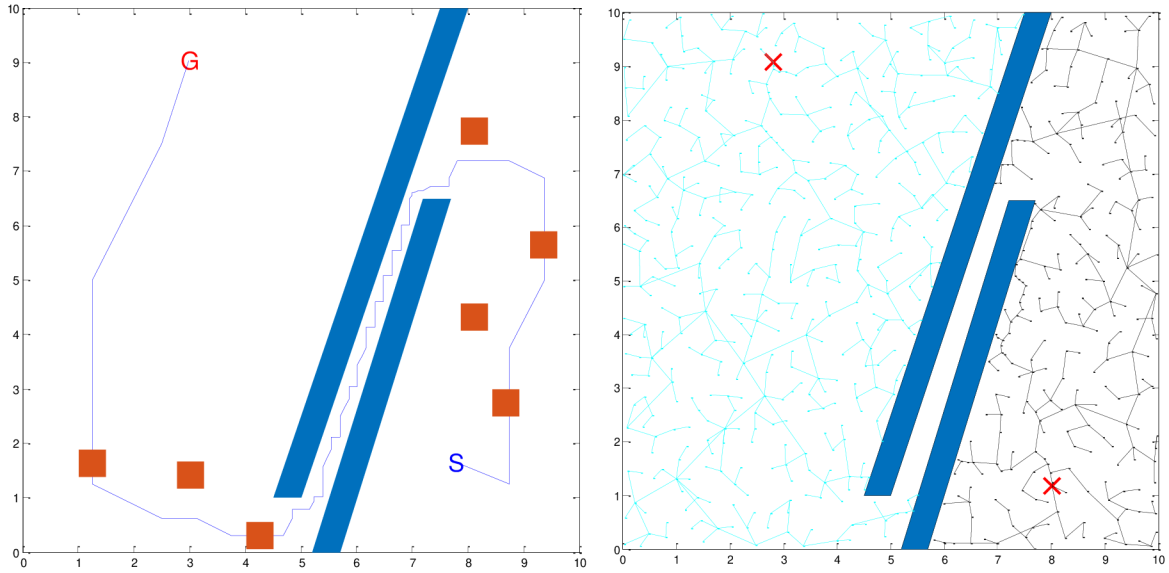


Figure 4-47: (a) The drawback of ACD in dynamic environments, (b) the drawback of RRT in narrow passage and small regions

Proposed Methods

In this work, the RRT and approximation cell decomposition (ACD) algorithms are combined together in order to exploit the advantages of each of them. The new planners try to overcome the drawbacks, which effect the performance of the navigation process significantly, by complementing these two approaches.

The RRT planner has relatively high tolerance to obstacles shapes and workspace changes. This feature is missing in the ACD planner as shown in Figure 4-47-a. However, The RRT is not efficient in small areas and narrow passage as shown in Figure 4-47-b, unlike the ACD planner, which does not face this problem. Based on that, the efficient spatial planner, ACD, is used to plan a primary path in stationary workspace. Then, this path is used to guide the RRT growth.

The RRT planner validates the ACD's path when a query is established in the actual workspace. If a collision is detected due to the change in the workspace, the planner re-plans the path locally through the changed regions. Figure 4-48 shows the generated path using this principle.

Two approaches have been proposed to benefit from this combination. These planners focus on the enhancement of navigation problem for omnidirectional robots in partially dynamic workspace. In next sections these two proposed methods is discussed in more details.

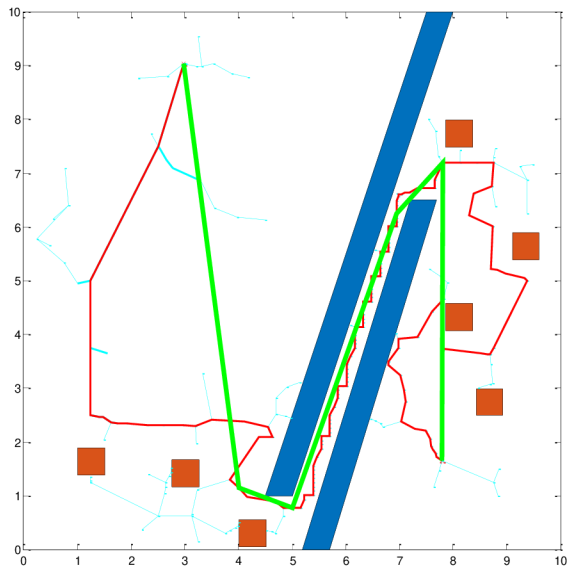


Figure 4-48: The generated path using the combination between ACD and RRT

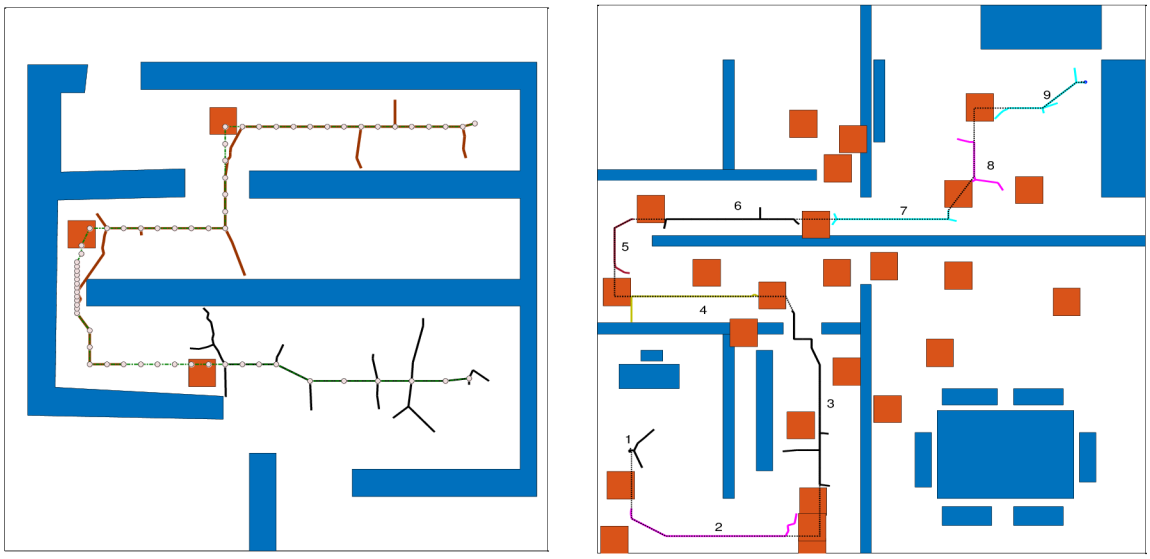
a. RRT Validator Planner

The RRT validator uses ACD's path as a guidance to the RRT tree's growth. It considers the CD-path's points as an ordered set, and directs the bias toward these vertices. The RRT trees branching toward these set in the same order, point by point. In the initial state, the probability of choosing the next point of the path is set to the value of 100%. If a collision is detected, then this probability is reduced in order to allow the RRT explores the free space and attempts to reconnect to any point of the ordered set.

If it reconnects, then the probability to choose the next point is reset again to the value of 100% to force the planner follows the original ACD's path once again.

This strategy forces the planner to follow the guiding path when it is possible, and at the same time, it gives the planner a freedom to find an alternative local path to the collided segments.

In our tests, two RRT validators are used to validate the path. The first one rooted at the initial position and the second one rooted at the goal position. They try to follow the ACD path, or find an alternative local path. The RRT trees are shown in Figure 4-49-a, where they try to follow the ACD' path (the dotted line).



(a) (b)

Figure 4-49: Examples of the proposed methods. The dotted line represents the ACD path in stationary workspace. (a) The RRT validator method, which creates two RRT trees from the initial and the goal location. (b) The local RRTs method, which creates nine local RRT trees

b. Local RRT Planners

The second proposed planner uses simple straight-line planner to connect the ACD path's points and test the collision. The planner tracks the valid points of the path and creates sequences of these points. In case that all points are valid, then the planner returns these points as a solution. In the other case when the workspace is changed, and a collision happened, the planner breaks the original path sequence in the collided locations and rebuilds sequences of the continuance valid points. It also excludes the points, which locate in the obstacle areas.

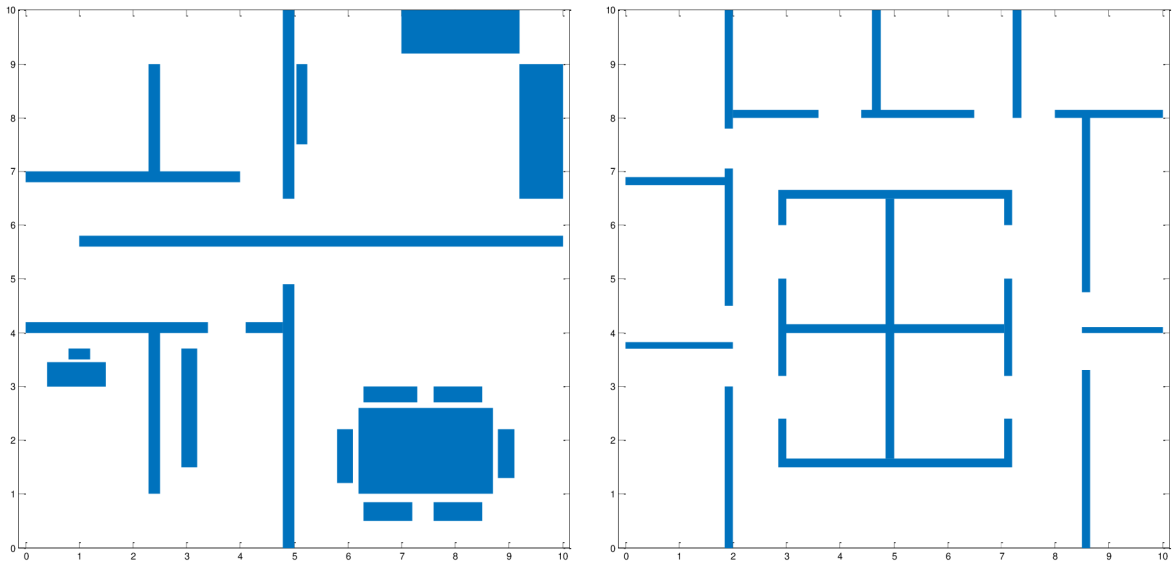
Each of these sequences is associated with RRT tree. The trees later on explore the space freely with small bias toward the other tree's nodes. If two trees are near to each other, they are merged to form one tree. When all trees are merged, they form a single tree, which include the initial, and goal locations.

In this planner, our strategy is to generate augmented local RRTs, in order to navigate around the new obstacles locally. Figure 4-49-b shows the local RRTs planners method in

simulation. In this example, it creates nine local RRT trees based on the original path, which is generated in the stationary workspace.

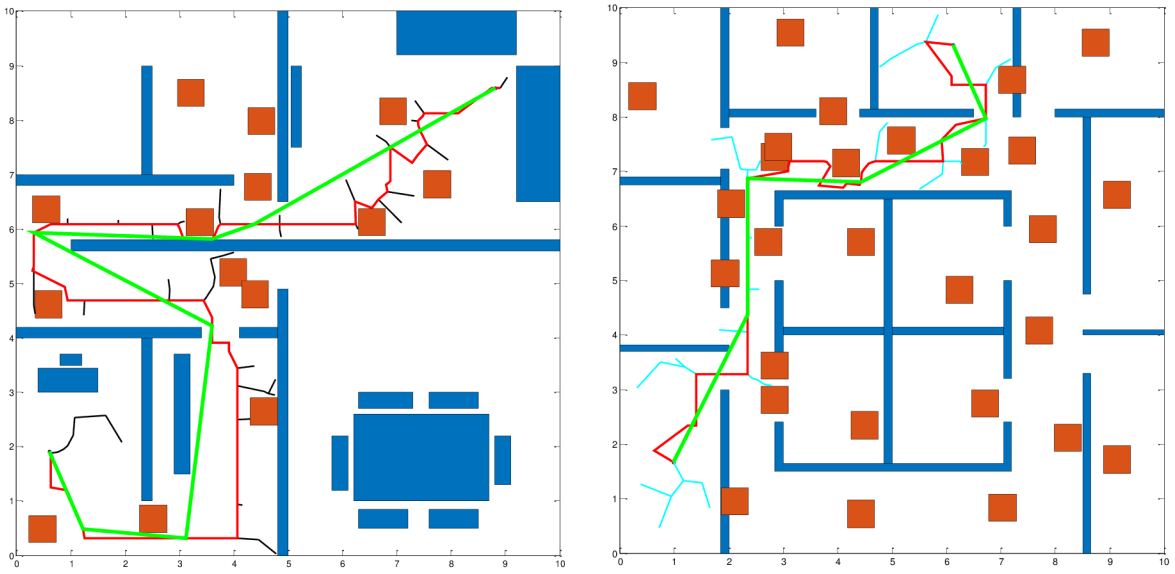
Tests and Results

The proposed approaches are tested in two different workspaces as shown in Figure 4-50. The first one represents an office with one route between the rooms, and the second one represents offices, which have two possible routes between them.



(a) (b)

Figure 4-50: Testing workspaces, (a) one route office between rooms (WS1), (b) multi routes between offices (WS2)



(a) (b)

Figure 4-51: Example of RRT path using (a) Local RRTs method, (b) RRTs validator method, in partially changed workspaces. The bold-green lines represent the shortened RRT path in both tested workspaces. The boxes represent the new obstacles

The robot in this work is considered as a holonomic point translates in the workspace. The results of the proposed methods are compared to the other methods, i.e. the basic RRT algorithm, Goal Bias RRT, and the bias toward the other trees. Figure 4-51 shows an example of RRT path for the proposed methods, the local RRTs method (a), and RRTs validator method (b) in the testing workspaces.

a. Testing Parameters

The bias values, which are given to the compared methods, are set as shown in Table 4-19, where the basic RRT chooses a random point without any bias. The goal-bias RRT directs the growth of the tree toward the goal location by selecting this location in probability of 10%. In the tree's nodes bias, the RRT chooses a point of the others trees by the probability of 30% that force the trees to merge more quickly.

Table 4-19: The probability of choosing next points (bias value)

Method	Bias Value
RRT	0
Goal Bias	0.1
Tree Node Bias	0.3
RRTs validator (valid point)	1
RRTs validator (Collison)	[0.2,0.1,0.7]
Local RRTs	0.3

In our proposed methods, the bias value of the validator RRTs is set to 100% when no collision occurred. Else, the bias value is set as follows, it has the value of 20% toward the next valid point in the ordered set, in addition, to the value of 10% bias toward any other points in those points set. The planner in this case has the probability of 70% to explore the workspace freely and biases the growth toward a randomly chosen sample.

The last method, the local RRTs approach, uses the bias toward the other trees by the value of 30%.

The simulation repeated 100 times and the average of the successful attempts are taken for results comparison. The results include the execution time, the number of RRT iterations, which is corresponding to the number of RRT branching attempts, and the number of successful attempts to find a path.

The probabilistically completeness is estimated using the successful attempts result. While the efficiency is estimated using the time of execution and iterations results. The time of execution could be varied significantly based on the hardware and code optimization, while RRT iteration is independent of HW and the programmers skillful.

The ACD resolution is set to be 0.2 unit. Moreover, the ACD's path points are generated in ordered form, from the initial to the goal locations. They are constructed using the initial and the goal points, the free cells' centers, and the barriers' midpoint between the consequence cells. We use the Dijkstra's algorithm to search in the ACD's graph. RRT

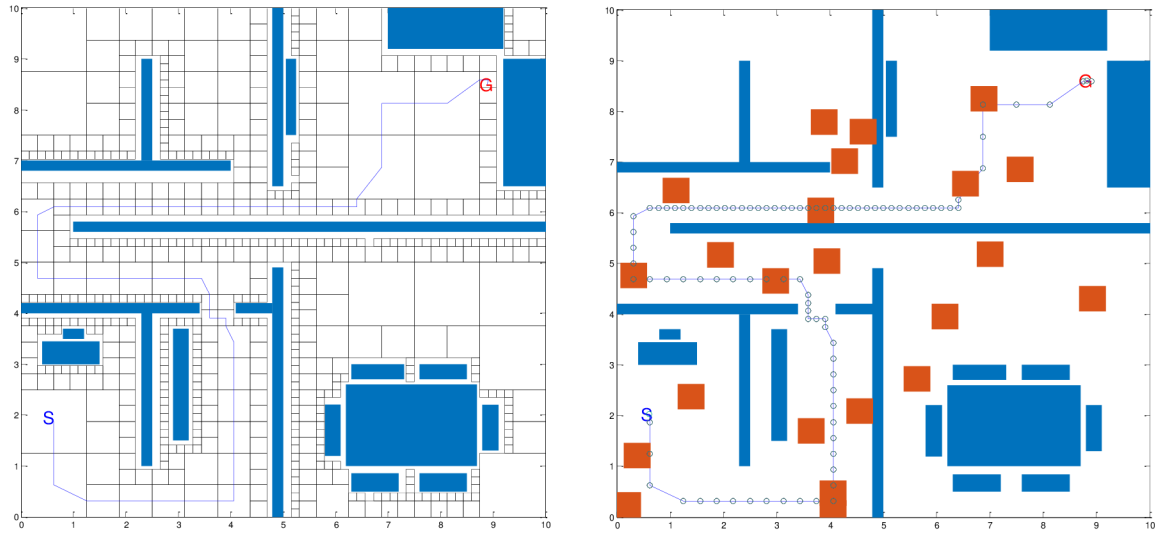
parameters are set as follows; the extension step is equal to 0.3 unit. And, the bias probability is fixed at 100% for next path' points in case of no collision is detected.

The reduced bias is divided into three values when the path is collided within obstacles. 1- The bias toward the next valid point is set to a value of 20%. 2- The bias toward other path's points is given the value of 10%. 3- The rest of the bias is relaxed to allow the planner chooses random samples freely. The RRT planner considered as failed if it cannot find a path after 2000 tries of branching.

b. Results and Discussions

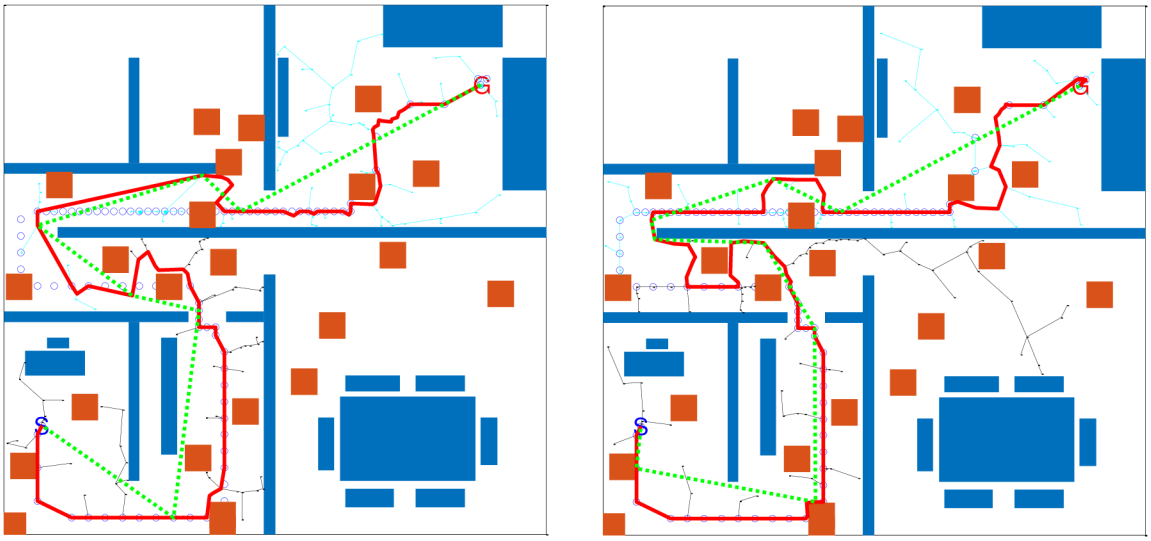
In the first workspace, new obstacles are scattered in the original workspace. They are positioned to collide within the ACD path and add more difficulty to navigation process through the changed workspace. The workspace changes are shown in Figure 4-52-b, where the boxes represent the new obstacles. The ACD path is shown as a solid line between the initial and the goal locations. The cycle markers represent the bias points. ACD algorithm approximates the free cells as shown graphically in Figure 4-52-a, the path in this case is produced using the Dijkstra searching method in the ACD adjacency graph. Figure 4-53, shows the generated path using RRT validator (a), and the local RRT (b) methods.

The numerical results are shown in Table 4-20, where the proposed methods show more probabilistically completeness than the other methods do.



(a) (b)

Figure 4-52: Office-like workspace (WS1); (a) cell decomposition approximation, (b) new obstacles, ACD path represented by the solid line, and the bias points represented by cycle markers



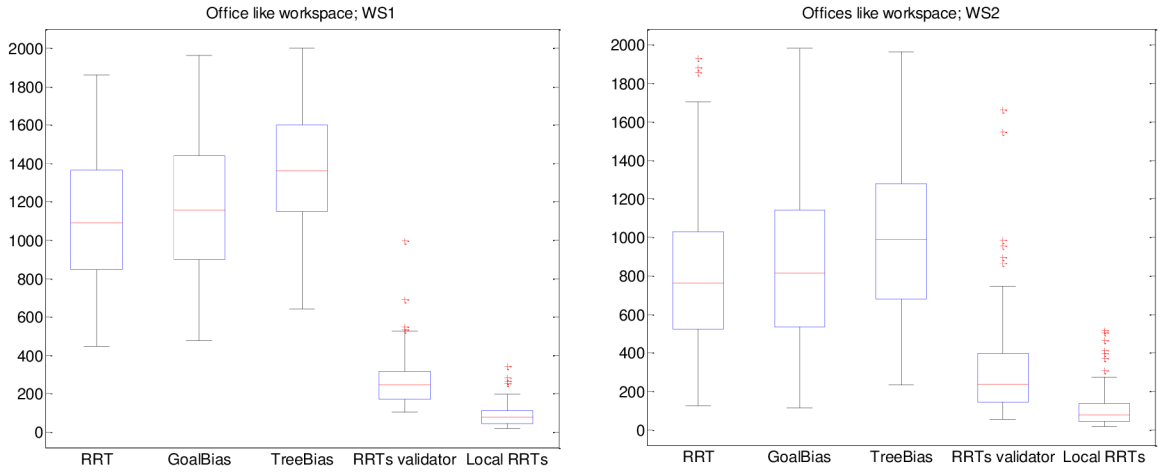
(a) (b)

Figure 4-53: (a) The generated RRT path using RRT validator method, (b) the generated RRT path using local RRTs method in WS1 workspace, the red-bold line represents the original RRT path, the green-dotted line represents the shortened path, the bias points represented as (o) markers

Table 4-20: The result of the tested methods in WS1

Method	Mean Time	Mean Iteration	Success
RRT	1.03	1137.11	96
Goal Bias	1.12	1180.57	87
Tree Node Bias	1.23	1365.34	80
RRTs validator	0.45	270.19	**100**
Local RRTs	**0.19**	**95.20**	**100**

The Local RRT trees method gives the best results; it has the lowest execution time, and the lowest iteration to find a path. Moreover, the RRT validator method gives better results than the other competitor does. Figure 4-54-a sums up the iteration results for the first workspace WS1 using the boxplot representation.



(a) (b)

Figure 4-54: RRT iteration boxplots for WS1 (a) and WS2 (b)

In the second workspace, the partially changes are introduced by scattering new obstacles in the stationary workspace. The obstacles collide the ACD path and produce more narrow passages. Figure 4-55-b shows the changes in the workspace, where the obstacles are represented by the boxes. The ACD path is shown in the figures as a solid line between initial and goal locations, and the bias points, which are generated on this path, are shown in the figures as cycle-markers. The approximation of the free workspace using ACD is shown in Figure 4-55-a. And the generated path using RRT validator, and the local RRT methods are shown in Figure 4-56.

Table 4-21: The result of the tested methods in WS2

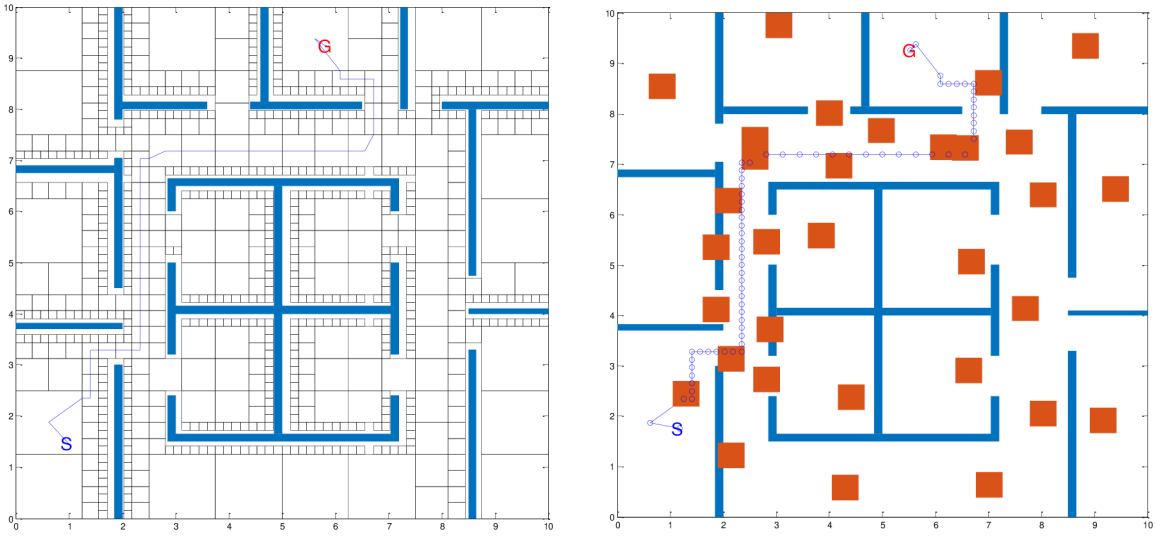
Method	Mean Time	Mean Iteration	Success
RRT	0.92	817.13	96
Goal Bias	0.98	871.06	94
Tree Node Bias	1.076	1005.10	86
RRTs validator	0.62	332.07	**100**
Local RRTs	**0.24**	**117.17**	**100**

The numerical results are presented in Table 4-21. As shown in this table the proposed methods give the best results; they are probabilistically complete as it is inferred from the success rate result. The Local RRT trees method gives the best results in terms of efficiency; it has the lowest execution time, and the lowest iteration average. Figure 4-54-b condenses the iteration results for WS2 using the boxplot.

Summary

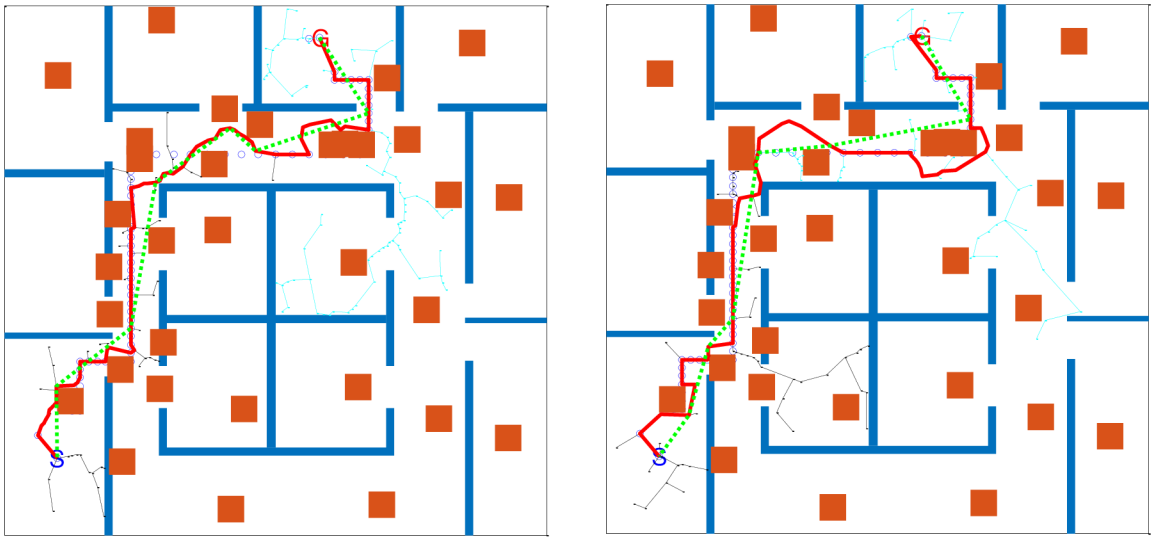
In this work, the approximation cell-decomposition algorithm (ACD) is combined with the RRT planner in order to enhance the omnidirectional robot navigation in partially changed workspace. The ACD finds a spatial path in preliminary and stationary workspaces, and then the RRT is used to validate this path in the actual workspace.

Two methods have been proposed in this work. First, the planner creates instances of RRT which bias toward the path's points in ordered form. It updates its bias value based on the collision detection information. The Second method uses a straight-line planner to connect path's points and creates local RRT trees on both sides of the collided segment of the path. The proposed methods compared with other approaches and the simulation shows that they give the best results in terms completeness, while the local RRTs method gives the best result in terms of efficiency in both testing workspaces.



(a) (b)

Figure 4-55: Offices-like workspace (WS2); (a) approximation cell decomposition, (b) new obstacles, ACD' path represented by the solid line, and the bias points represented by cycle markers



(a) (b)

Figure 4-56: (a) The generated RRT path using RRT validator method, (b) the generated RRT path using local RRTs method in WS2 workspace, the red-bold line represents the original RRT path, the green-dotted line represents the shortened path, the bias points represented as (o) markers

5 EXPERT SYSTEM

Expert system (ES) is “An intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant expertise” [prof. Feigenbaum].

Expert systems are designed to solve complex problems by reasoning about knowledge like an expert, they do not follow the procedure way as in conventional programming case, rather they act as an expert to solve a problem in a particular domain. ES processes the information in symbolic form, copes with errors in data, and with imperfect rules of reasoning. In addition, ES can answer why and how questions reasonably well, and explains how it arrived at a particular (Sasikumar et al. 2007; Negnevitsky 2005).

The terms of expert system and knowledge-based system are used interchangeably, even though there are small differences between them. These differences are based on the inference methods, data storage, and knowledge collecting methods.

The expert systems are used in many applications, e.g. interactive application, faults diagnostic, medical decision support, educational application, knowledge management, resource planning, controls, and many other fields.

ESs have many advantages, they are used to capture the scarce expertise, increased productivity, and quality, decreased decision-making time, reduced downtime via diagnosis, knowledge transfer, integration of several experts' opinions, and working with uncertain information.

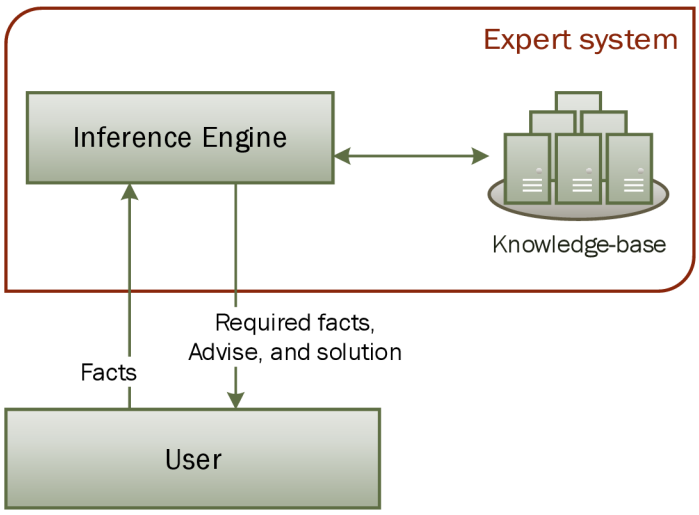


Figure 5-1: Expert system components

5.1 Expert system structure

The expert system contains two main components, the knowledge base module, and the inference engine module. Other parts can be added based on the application (Sajja et al. 2010). For example, user interface, knowledge engineer interface, and explanation facilities, etc. Figure 5-1 shows the basic expert system components.

5.1.1 Knowledge base

The knowledge is a theoretical or practical understanding of a specific area. The knowledge base in expert system contains the "domain-specific knowledge," which is required knowledge to solve a problem.

The knowledge can be represented in many ways, e.g. production rules (if-then rules), clausal logic, Object-Attribute-Value Triples, semantic networks, and frame representations.

Rule-based representation

If-then rules are one of the most common forms of knowledge representation that used in expert systems. Most experts are capable to express their knowledge in the form of rules.

Any rule consists of two parts: the IF part, called the condition (premise or antecedent), which is evaluated based on what is currently known about the problem; and, the THEN part, which is called the action (conclusion or consequent). For example,

IF *pathFailure* **>** 3 **THEN** *pathWeight* **is** 0
IF *pathExecTime* **is** *High*, **and** *pathFailure* **is** *High* **THEN** *collideTendency* **is** *High*

In the first example, the variables in IF-THEN statements are crisp variables (constants); in this case, if the knowledge about these variables is not certain, then a degree of certainty is attached to each rule. These degrees of certainty are called certainty factors.

In the second example, the variables have symbolic value. In this case, uncertainty in variables' values, beside the certainty factor method, can be handled using fuzzy expert system. The fuzzy expert system is discussed in a separate section.

Clausal Logic statements representation

The clausal logic statements are similar to rules-based structure. However, the expressive power of logic based knowledge representation languages is much better than that in if-then rules.

A clause is formed by combining a number of literals using *connectives*. The permissible connectives are *(← implication*), (*and*), and (*or*). A clause begins with a consequent-part, followed by an implication and then an antecedent (Sasikumar et al. 2007).

Object-Attribute-Value Triples representation

The 3-tuple representation is a very simple form. It consists of three parts, for example

- Path Length 10
- Path nVertices 23
- RRT execTime 0.5sec

The drawbacks of these structures are it repeats the name of the object many times, in addition, this representation does not reflect the priority between these statements, and the data structure.

Semantic networks representation

The basic idea behind semantic networks is to link the related concepts together. A semantic network consists of nodes representing concepts. The concepts, which are semantically close to each other will be closer to each other in the network, while concepts that do not have direct connection will be far apart. The path in this network between two concepts represents how they related, and the length of the path indicates how these concepts are close to each other.

A semantic network differs from a graph in that there is meaning (semantics) associated with each link. This semantics comes usually from the fact that the links are labelled with English words (Sasikumar et al. 2007).

Frames representation

The frame is a data structure; it represents a cluster of facts and properties that describe an object in detail. A knowledge representation using frames can be thought as a network of nodes and relations, where each node represents a frame.

Frames provide a natural way for the structured and concise representation of knowledge. The frame combines all necessary knowledge about a particular object or concept in a single entity. They are an application of the object-oriented programming in the expert systems domain (Sasikumar et al. 2007).

In the rule-based system, a set of rules representing the knowledge is used for problem solving. Each rule captures some heuristic of the problem, and each new rule adds some new knowledge and thus makes the system smarter. The rule-based system can easily be modified by changing, adding, or subtracting rules.

In a frame-based system, the problem is viewed in a different manner, where the overall hierarchical structure of the knowledge is decided first. The classes and their attributes are identified, and hierarchical relationships between frames are established. The architecture of a frame-based system should provide a natural description of the problem (Negnevitsky 2005).

5.1.2 Inference Engine

The inference engine is a part of ES, it tries to derive new information about a given problem using the knowledge base. In rule-based systems, it is used to decide which rules should be executed based on the satisfaction of the antecedents and priorities of the rules. Inference engines in the rule-based systems use different strategies to derive the goal. The most common strategies are the forward chaining, and the backward chaining (Sasikumar et al. 2007). The expert systems can use either one of these strategies or a combination of them.

Forward chaining is a data driven reasoning. It starts from antecedent parts of the rules, and evaluates these rules based on the available facts, until the goal is reached or the inference process requires other facts to find a goal. Generally, this method is used in applications such as monitoring, controlling, and prognosticating problem.

Backward chaining is the goal-driven reasoning, where, the expert system has the goal (a hypothetical solution) and the inference engine attempts to find the evidence to prove it (Negnevitsky 2005). This method is good for problems like diagnosis problems.

5.2 Fuzzy Expert System

Fuzzy or multi-valued logic was introduced in 1930s by Jan Lukasiewicz. He studied the mathematical representation of fuzziness, and introduced a logic that extended the range of truth-values to all real numbers in the interval between 0, and 1. He used a number in this interval to represent the possibility that a given statement was true or false. Then, in 1937, Max Black, a philosopher, published a paper called 'Vagueness: an exercise in logical analysis' where, he defined the first simple fuzzy set and outlined the basic ideas of fuzzy set operations. In 1965 Lotfi Zadeh, published his paper 'Fuzzy sets', where he rediscovered the fuzziness. Zadeh extended the work on possibility theory into a formal system of mathematical logic, and introduced a new concept for applying natural language terms. This new logic for representing and manipulating fuzzy terms was called fuzzy logic (Negnevitsky 2005).

Fuzzy-logic deals with approximate reasoning rather than fixed and exact one. Fuzzy-logic handles the concept of partial truth, where the truth-value may range between completely true and completely false.

In fuzzy rules-based inference system, e.g. Mamdani method, the input data are converted into fuzzy values using fuzzification procedure. Then, the fuzzy rules are evaluated, these rules determine the inputs-outputs relations and the system behavior.

The output of each rule is a fuzzy set. In order to obtain a precise solution, not a fuzzy one, the outputs of all rules are aggregated into a single fuzzy output, and then it is defuzzified into a single number. Figure 5-2, shows the structure of Mamdani inference method.

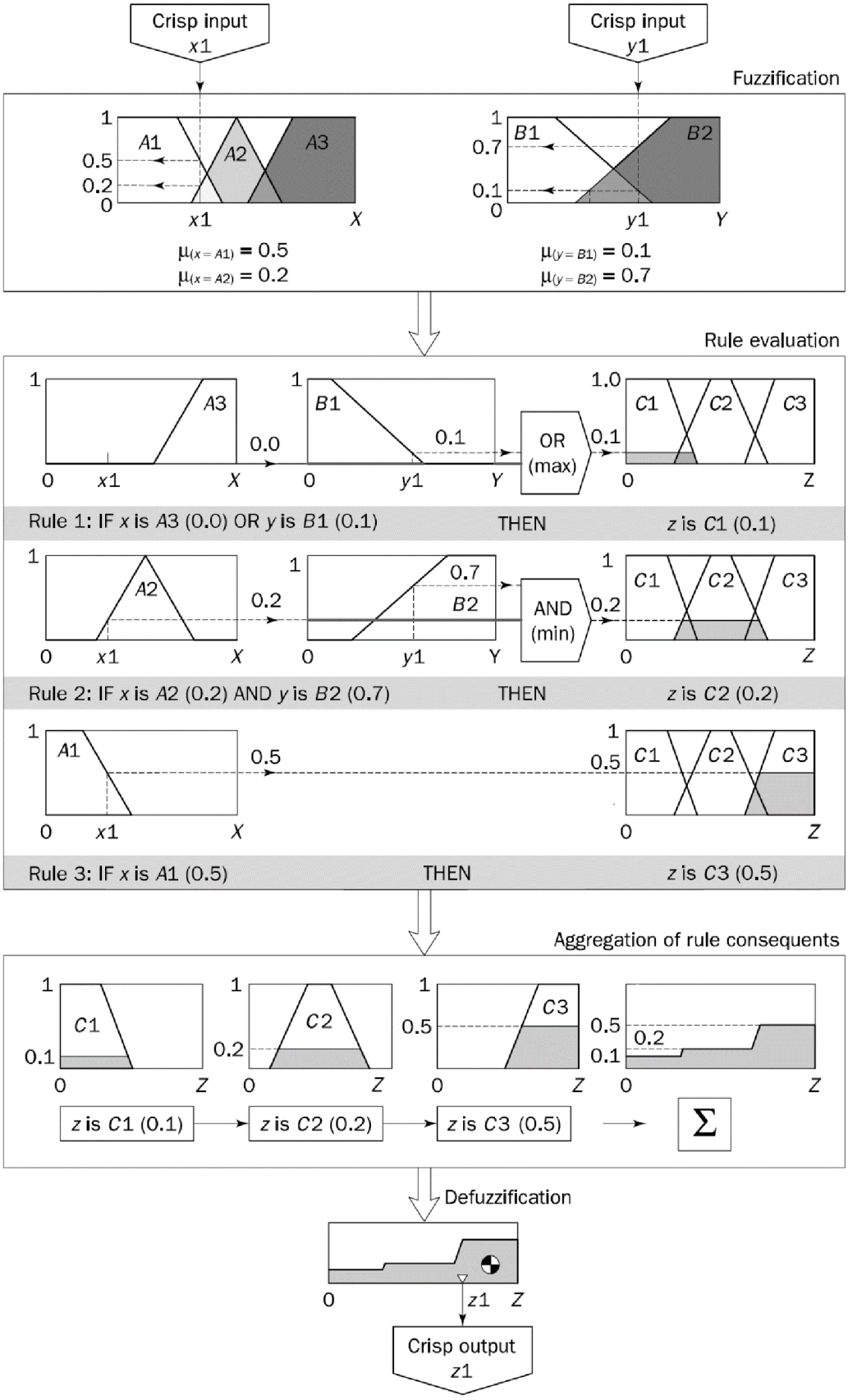


Figure 5-2: The basic structure of Mamdani fuzzy inference, source (Negnevitsky 2005)

The linguistic variables are used to represent the input and output fuzzy-sets. The range of possible values of a linguistic variable represents the universe of discourse of that variable. For example, in the following rules

IF *RegionState* **is** *Changed* **AND** *CollisionRate* **is** *Low* **THEN** *BiasToReg* **is** *High*
IF *RegionState* **is** *Changed* **AND** *CollisionRate* **is** *High* **THEN** *BiasToReg* **is** *Low*

The universe of discourse of the linguistic variable might have a value such as {*Changed*, *Unchanged*, …. }, in the *RegionState* variable case. It may include fuzzy subsets as *Low*, *Medium, and High as in CollisionRate* variable.

The fuzzy reasoning, in general, includes two parts: evaluating the rule antecedent, and applying the result to the consequent of the rules. In classical rule-based systems, if the rule antecedent is true, then the consequent is also true. However, in fuzzy systems, where the antecedent is a fuzzy statement, all rules are evaluated and the uncertainty is expressed using the fuzzy sets. A typical process to develop the fuzzy expert system incorporates the following steps (Negnevitsky 2005):

1. Specify the problem and define linguistic variables.
2. Determine fuzzy sets.
3. Elicit and construct fuzzy rules.
4. Encode the fuzzy sets, fuzzy rules, and procedures to perform fuzzy inference into the expert system.
5. Evaluate and tune the system.

5.3 Expert System application in motion planning problems

The experience of robots when they move from one location to another one can be stored and then used by ES to guide the planner.

Many attempts introduced to improve the robotic motion planner using the previous experience (Berenson et al. 2012; Lien et al. 2009; Martin et al. 2007; Zucker et al. 2007, 2008 ; Atkeson et al. 2003; Stolle et al. 2006).

Figure 5-3 shows a framework called “Lightning framework” which utilizes this idea. It uses the old success path, when a new query is established, both modules retrieve-repair (RR) and planning from scratch (PFS) are started simultaneously, and the first path produced by either module is executed on the robot while the other module is stopped. After generating a new path, a library manager decides whether to store that path or not, based on the computation times of the two modules and the generated path’s similarity to the retrieved one (Berenson et al. 2012).

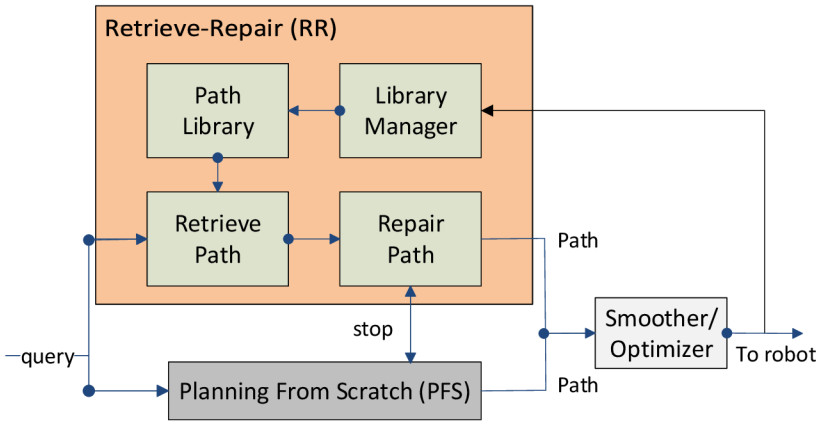


Figure 5-3: Diagram of the lightning framework. Source: (Berenson et al. 2012)

Fuzzy expert system is used in robotics applications frequently as a fuzzy controller to steer robots based on sensor data, also in motion-planning, navigations problems, and in location estimation (Aguirre et al. 2000; Sharef et al. 2010; Petr Krcek et al. 2004; Montaner et al. 1998; Driankov et al. 2001). An important problem in autonomous navigation is the need to cope with the large amount of uncertainty that is inherent of natural environments, which is one of fuzzy systems' strong side.

The author in (Saffiotti 1997), uses fuzzy logic to make an adequate tool to address the problem of uncertainty. They focus on designing robust modules and coordinate the activity between them. They use data from several sensors, and integrate the high-level reasoning with the low-level of execution.

Some researches combined the fuzzy system with the other motion planning techniques, for example the in (Jaradat et al. 2012), the fuzzy-based potential field method is presented to autonomous mobile robot motion planning. They used Mamdani and TSK methods to develop the total attractive and repulsive forces acting on the mobile-robot's workspace. These methods use a fuzzy logic expert system to provide the robot with the most appropriate heading toward a stationary or moving target. The attractive force modeled using expert if–then rules based on the position and the velocity of the robot with respect to the target.

A new perspective, which utilizes a knowledge-driven approach for path planning, is studied in (Chen et al. 2014). The concept of relative state tree (RST) is proposed to develop an incremental learning method based on a path planning knowledge base. The knowledge library established by offline or online learnings techniques. As the robot plans online, its movement is guided by the optimal decision that is retrieved from the library based on the information that matches mostly the current environment.

ES in our work is adapted and utilized to evaluate the free regions in the workspace and guide the planner to possible routes in the workspace. It uses the workspace map, and the experimental results analysis, e.g. collision tendency, for reasoning about these regions.

5.4 Contribution, Tests and Results

In the next section, we use the fuzzy expert system to bias the sampler module in the motion planner. The sampler drew samples from free regions in a different density, based on the region evaluation. The evaluation of the free region is calculated using fuzzy rule-based expert system (Abbadi et al. 2015).

5.4.1 Hybrid rule-based motion planner in cluttered workspace

In this work, two new planners have been proposed. They depend on rules-based adviser. Each of these hybrid planners is composed of two-layers to enhance motion planning in heterogeneous, cluttered, and dynamic workspace. The first layer uses the exact cell decomposition algorithm, in order to find the free regions and the graph of adjacency in simple, static, and 2D workspace. Then, the second layer utilizes the rapidly exploring random trees approach, to find a path in cluttered and dynamic workspace. The information about free regions from the first layer and the exploration information from the second layer are combined to guide the growth of RRT trees. The combination is done using expert rules-based adviser that classifies the free regions and update their bias-weights.

The adviser of the first planner biases and pulls the trees growth toward the boundary areas between explored and unexplored regions. While the adviser of the second planner uses the collision information, and fuzzy rule-based set, to bias the trees growth toward low collision areas around the boundaries of the explored regions.

These planners exploit and combine the advantages of the exact cell decomposition in simple, and low dimensional workspace, and the advantages of RRTs, which have a relatively higher tolerance to the changes in the environments.

The planners are tested in stationary workspaces, minor changes, and major changes scenarios. The proposed methods have been compared to other approaches, and the simulations results show that the proposed methods have better results, in terms of completeness and efficiency.

Proposed methods

The planner consists of two layers, the first one uses the trapezoidal cell decomposition method in static workspace to find the adjacency graph of free cells, while the second layer uses RRTs algorithm to find a path in the same workspace, but after new cluttered and dynamic obstacles are added. In order to enhance the RRTs ability to find a path, rules-based advisers have been proposed also. The function of this adviser is to update the weights of free regions in order to pull the trees growth toward the most important regions in the workspace.

The rules-based adviser in first planner uses the adjacency graph information and RRTs nodes' location to update the regions' weight. The rules-based adviser in second planner uses in addition to former information the collision information in the workspace regions. These resources of information are combined to bias the exploration toward the most important and low collision areas.

The adjacency graph contains information about the free regions and the relations between them, while the information that comes from RRTs contains the location of trees' nodes in the free areas and the difficulty to reach these regions.

To formulate this procedure the region state variable ($state_r$) is defined to take one of these four values [*boundary*, *neighbor*, *expanded*, and *far*]. The value of this variable depends on the existence of any valid RRT node inside the corresponding region *r,* or in its neighbors. The formulation of this proposal is described as follows.

R is the set of all free regions in the workspace.

S_r is a set of all samples in region r.

N_r is a set of all regions neighbor to region r.

RRTree is a set of all samples, which are considered as valid node in RRT trees.

For any region r the variable $state_r$ takes the value of *far* when the region and its neighbors do not contain any sample belongs to RRT. It takes the value of *neighbor* when at least one sample of RRT is located in r's neighbor regions but not in r itself. The $state_r$ takes the value of *boundary* when at least one sample of RRT is located in region r and there is still at least one neighbor not explored yet. Lastly, the $state_r$ takes the value of *expanded* when at least one sample of RRT is located in region r and all neighbors are explored; i.e. their state is *expanded* or *boundary*. The formulation of these values and conditions are listed in Figure 5-4.

$$\forall r \in R: state_r = \begin{cases} far, & (RRTree \cap S_r = \emptyset) \wedge (\forall t \in N_r: RRTree \cap S_t = \emptyset) \\ neighbor, & (RRTree \cap S_r = \emptyset) \wedge (\exists t \in N_r: RRTree \cap S_t \neq \emptyset) \\ boundary, & (RRTree \cap S_r \neq \emptyset) \wedge (\exists t \in N_r: RRTree \cap S_t = \emptyset) \\ expanded, & (RRTree \cap S_r \neq \emptyset) \wedge (\forall t \in N_r: RRTree \cap S_t \neq \emptyset) \end{cases}$$

Figure 5-4: The $state_r$ variable values and their Conditions

Adviser's rules in planner 1	
IF $state_r$ **is** *far*	**THEN** *weight* **is** *veryLow*
IF $state_r$ **is** *expanded*	**THEN** *weight* **is** *low*
IF $state_r$ **is** *boundary*	**THEN** *weight* **is** *high*
IF $state_r$ **is** *neighbor*	**THEN** *weight* **is** *veryHigh*

Figure 5-5: The adviser's rules of "bias toward boundaries" planner

Based on these values, the regions' *weight* are updated. Figure 5-5 shows the rules-based adviser in the first planner. After each iteration of RRTs, the regions' *weight* are

updated to identify the most important ones. The *weight* variable could take one of these values [*veryLow*, *low*, *high*, *veryHigh*]. These values are translated into RRT bias. The RRTs is directed to grow trees to the boundaries of explored areas, by making the *neighbor* regions having the highest weights, and the *boundary* regions have less or equal importance. Figure 5-6 shows the RRT growth and the regions classifications.

In explored areas, the algorithm blocks RRT trees from branching or selecting a new node inside them. However, a small amount of bias toward these regions is kept to avoid the situation where the planner works in small regions and block itself.

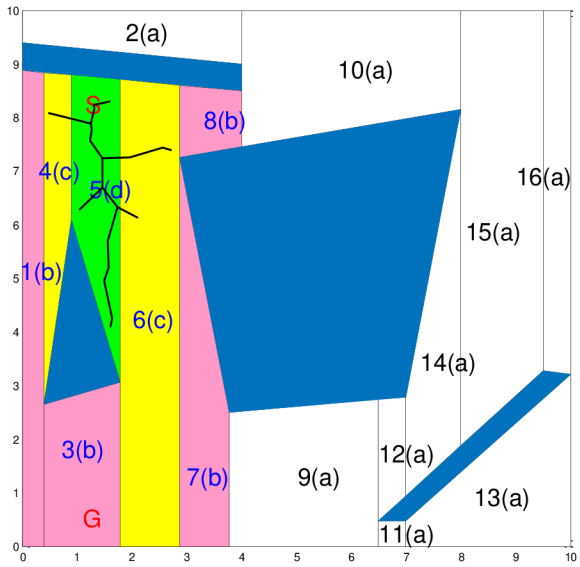


Figure 5-6: RRT growth and rules-based classification of free Regions; a: *far* regions, b: *neighbor* regions, c: *boundary* regions, d: *expanded* regions, *S* represents the initial position, *G* represents the goal position, and the blue regions represent the obstacles

The trees grow and follow the free areas, and do more work to navigate through local workspace instead of the whole workspace. If a region is obstacle-free, then the planner passes through it rapidly, if not the RRTs tries to navigate around the local obstacles.

Adviser's rules in planner 2				
IF	$state_r$ **is** *far*		**THEN** *weight* **is** *veryLow*
IF	$state_r$ **is** *expanded*		**THEN** *weight* **is** *low*
IF	$state_r$ **is** *boundary*	**AND** *collisionRate* **is** *low*	**THEN** *weight* **is** *high+*
IF	$state_r$ **is** *boundary*	**AND** *collisionRate* **is** *high*	**THEN** *weight* **is** *high-*
IF	$state_r$ **is** *neighbor*	**AND** *collisionRate* **is** *low*	**THEN** *weight* **is** *veryHigh+*
IF	$state_r$ **is** *neighbor*	**AND** *collisionRate* **is** *high*	**THEN** *weight* **is** *veryHigh-*

Figure 5-7: The adviser's rules of fuzzy bias planner

The second proposed method uses fuzzy rules-based to update the weights as in previous version, in addition, the collision information is considered. The new fuzzy variable *collisionRate* is defined. This variable takes the values of [*low*, *high*]. The information about the collision is collected during the execution.

The influence of collision rate is restricted to the most important areas. The *weight* variable in this case takes a value of [*veryLow*, *low*, *high-*, *high+*, *veryHigh-*, *veryHigh+*].

For a high value of collision rate, the weight of the *boundary* and *Neighbor* regions is reduced and the exploration is pulled toward more relax regions. Figure 5-7 shows the Rules-based for this fuzzy planner.

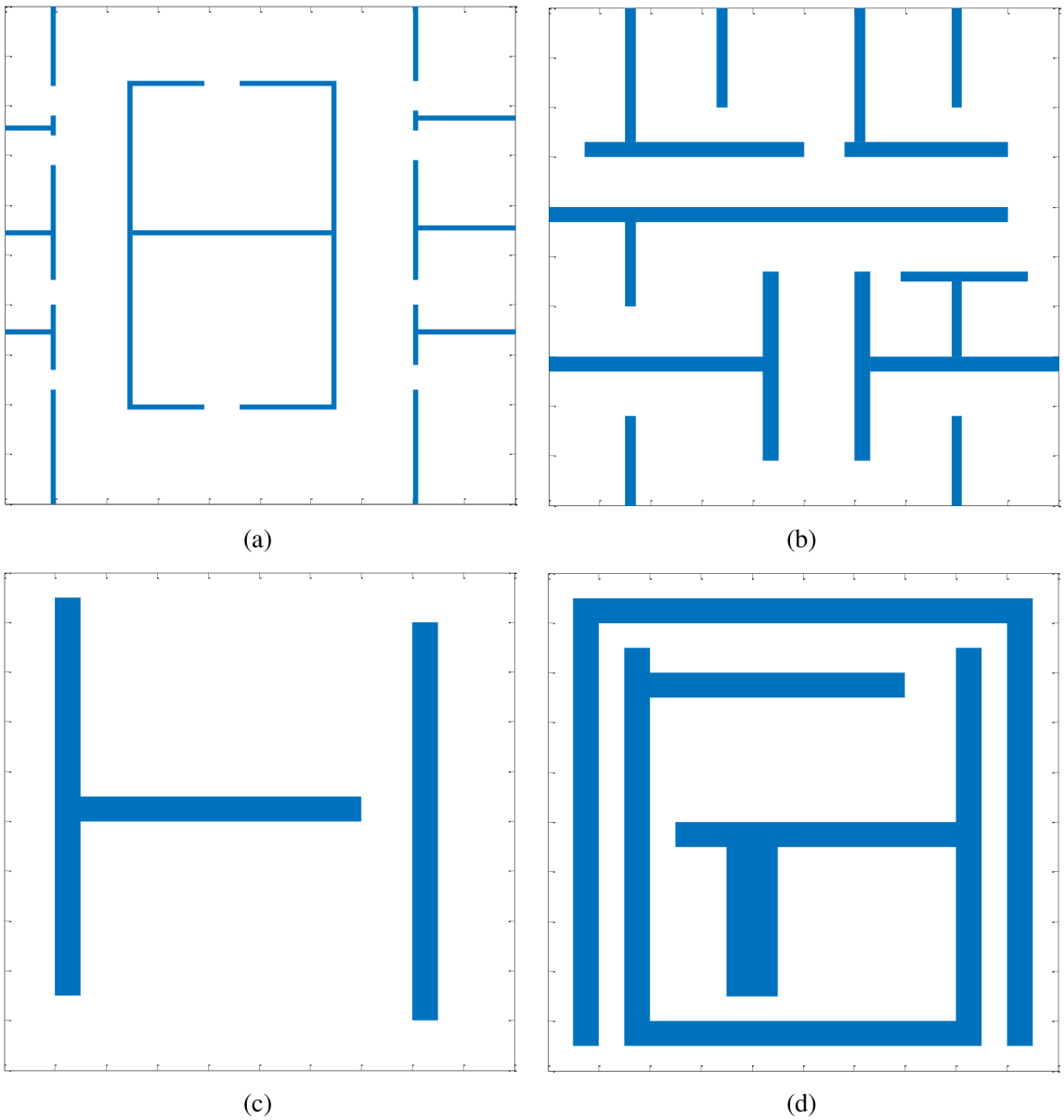


Figure 5-8: The testing scenarios. (a) Office-like workspace (WS1), (b) WS2, (c) WS3, (d) WS4

Simulations and Results

The tests are made in four workspaces to simulate the holonomic robots movements in offices and cluttered or crowded areas. The workspaces are shown in Figure 5-8. Every workspace is tested in three levels of changes. The first level is for stationary workspace. The second level includes workspace with minor changes, and the last one has major

changes in the workspace. The major change means close some routes or cluttered obstacles in high density.

The results of the first proposed planner "biasTowardBoundaries" and the second proposed one "FuzzyBias" are compared with other methods, i.e. RRTs without bias; RRTs with a bias toward the goal; RRTs with a bias toward others RRTs' nodes; RRTs with a bias toward the path that is generated by the cell decomposition algorithm.

The trapezoidal cell decomposition planner is used in these tests. It uses the Dijkstra's algorithm for searching the graph. In this case the Dijkstra has $O(\log(N)E)$ time complexity, where N is the number of nodes in the graph, and E is the number of edges.

Our focus in this work is to improve the completeness and efficiency in cluttered workspace.

The results are organized in two tables for every scenario. The first table lists the completeness value of each planner on the three levels of changes, while the second table contains data about the RRTs iterations. The RRTs iterations mean the number of required steps to find the goal. The smaller the iteration, the efficient the planner is.

Testing parameters

The tests are repeated 100 times, in every workspace. The completeness comparison uses the percent of successful tries to reach the goal, while, the average of RRT iteration is used for efficiency comparison.

The RRTs planner has extending-length (e = 0.3). The RRTs planning result is considered as failed, if it fails to reach the goal after 2000 tries of growing a branch.

The simulator implemented in Matlab and it uses a PC equipped with Intel Xeon (R) CPU 2.67 GHz, 6 GB of memory, and Windows 7 64-bit.

The bias value of every method is shown in Table 5-1. These values represent the probability of choosing the bias points. The complementary probability represents the choosing of a random sample from the workspace using a pseudo-random number generator.

Table 5-1: Bias values in the testing methods

Goal	Other Trees	CD path	Fuzzy	Boundaries
0.1	0.3	0.5	1	1

Results

In the first scenario, the path-planning problem in the "WS1" workspace is simulated. Figure 5-9 shows the original workspace, while the Figure 5-10 shows the minor changes, and the major changes in the workspace. The thin line represents the generated path of cell

decomposition, and the bold one is the shortened path of the original CD path. G and S points represent the goal and the initial locations, respectively.

The probabilistically completeness results are presented in Table 5-2, while, the iterations values are shown in Table 5-3. In this scenario, the office-like workspace is simulated. The major changes test simulates the situation where the shortest path is closed and the robot should find an alternative route to the goal, and avoid the cluttered obstacles.

Table 5-2: Number of successful attempts to reach the goal in WS1 workspace

Methods/ workspace	Without change	Minor change	Major change
No bias	98	94	45
Goal bias	96	90	47
Other Trees bias	97	90	25
CD path bias	99	95	24
Fuzzy bias	**100**	**100**	**100**
Boundaries bias	**100**	99	95

Table 5-3: The average of RRTs branching attempts to reach the goal in WS1 workspace

Methods/ workspace	Without change	Minor change	Major change
No bias	439	693	1253
Goal bias	470	780	1302
Other Trees bias	461	821	116
CD path bias	208	647	1404
Fuzzy bias	79	**397**	**590**
Boundaries bias	**77**	428	669

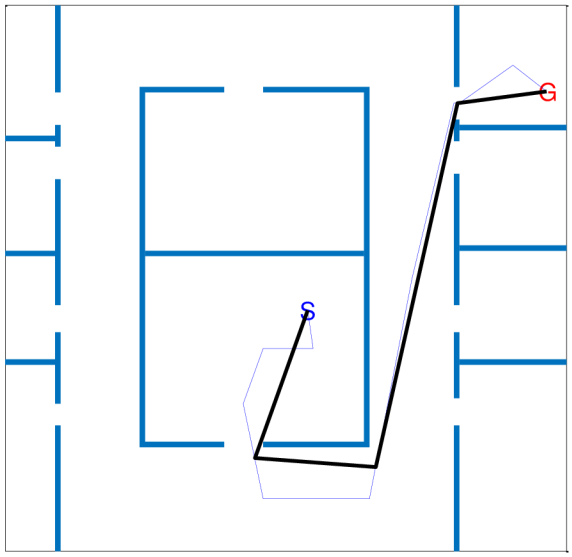


Figure 5-9: The basic workspace WS1. The thin line represents the CD path, and the bold line represents the shortened path. *G* and *S* represent the goal and the initial locations, respectively

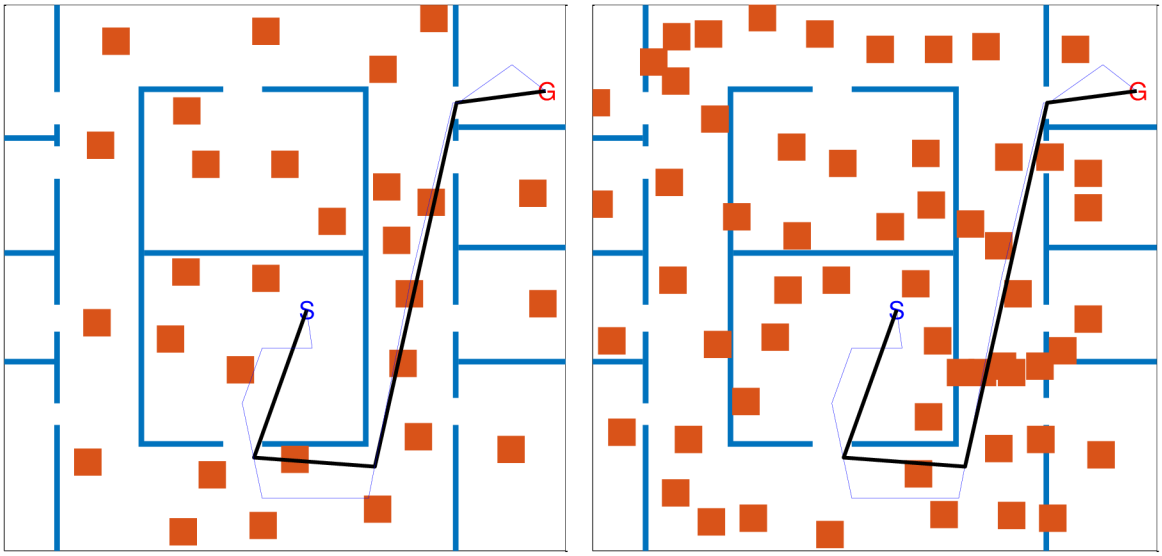


Figure 5-10: (a) The minor changes in WS1, (b) the major changes in WS1. The thin line represents the CD path, and the bold line represents the shortened path. *G* and *S* represent the goal and the initial locations, respectively

In the second scenario, the path-planning problem in "WS2" workspace is simulated. Figure 5-11 shows the original workspace. The minor changes and the major changes in workspace are shown in Figure 5-12. In these figures, the thin line represents the generated path using cell decomposition, and the bold line represents the shortened path of the original CD path. G and S represent the goal and the initial locations, respectively. The probabilistically completeness results are presented in Table 5-4, and the iterations values are shown in Table 5-5. In this scenario, the major changes test simulates the highly cluttered obstacles situation where the robot should pass through very small regions.

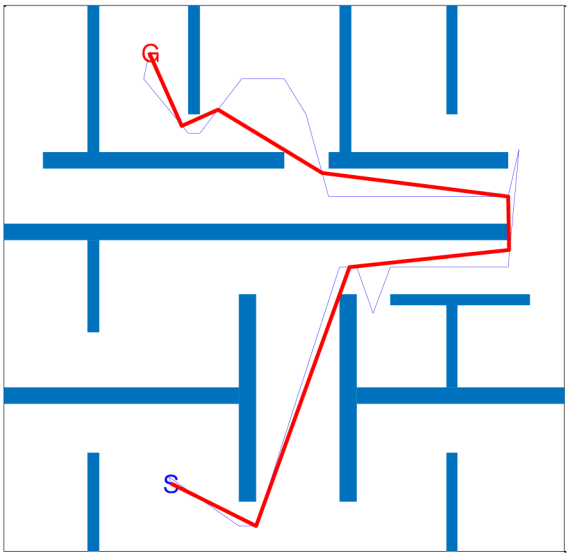


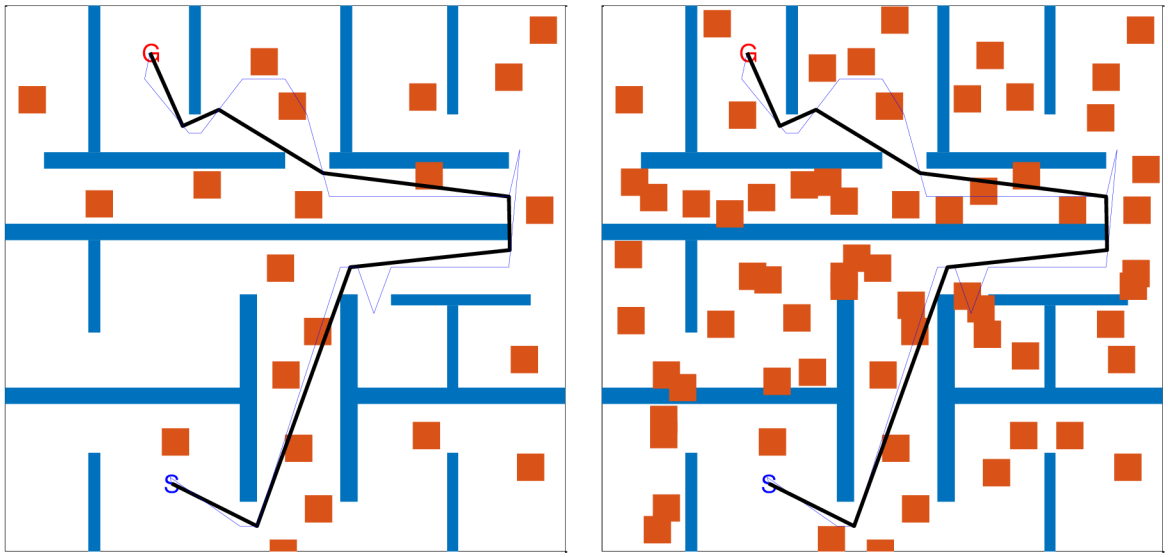
Figure 5-11: The basic workspace WS2. The thin line represents the CD path. Bold line represents the shortened path. *G* and *S* represent the goal and the initial locations, respectively

Table 5-4: Number of successful attempts to reach the goal in WS2 workspace

Methods/ workspace	Without change	Minor change	Major change
No bias	97	88	51
Goal bias	97	75	53
Other Trees bias	91	66	27
CD path bias	**100**	88	**71**
Fuzzy bias	**100**	**100**	62
Boundaries bias	**100**	99	66

Table 5-5: The average of RRTs branching attempts to reach the goal in WS2 workspace

Methods/ workspace	Without change	Minor change	Major change
No bias	792	1104	1235
Goal bias	864	1153	1269
Other Trees bias	1012	1227	1295
CD path bias	372	701	**798**
Fuzzy bias	**164**	**472**	1017
Boundaries bias	186	476	1027



(a) (b)

Figure 5-12: (a) The minor change in WS2, (b) the major change in WS2. The thin line represents the CD path, and bold line represents the shortened path. *G* and *S* represent the goal and the initial locations, respectively

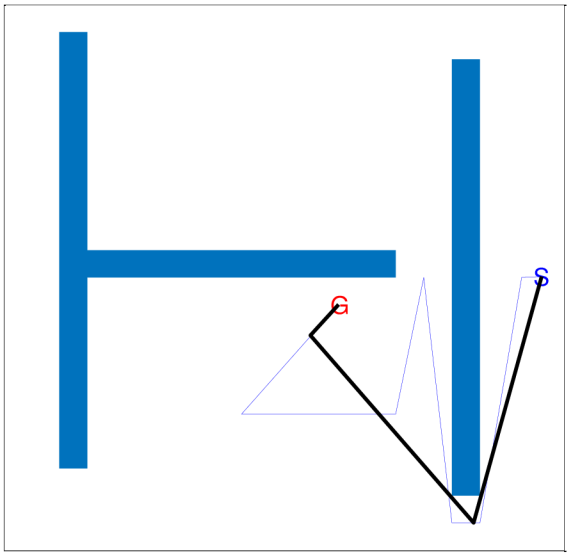


Figure 5-13: Basic workspace, WS3. The thin line represents the CD path, and bold line represents the shortened path. *G* and *S* represent the goal and the initial locations, respectively

In the third scenario, the "WS3" workspace is shown in Figure 5-13, while the minor changes and the major changes are shown in Figure 5-14. The thin line represents the generated path using the cell decomposition approach, and the bold line represents the shortened path of this original CD path. G and S represent the goal and the initial locations, respectively. The probabilistically completeness results are listed in Table 5-6. The iterations values are shown in Table 5-7.

In major changes test, we simulate the situation where some paths are closed and the robot should find an alternative route and avoid the cluttered obstacles.

Table 5-6: Number of successful attempts to reach the goal in WS3 workspace

Methods/ workspace	Without change	Minor change	Major change
No bias	100	100	33
Goal bias	100	100	23
Other Trees bias	100	100	9
CD path bias	100	98	0
Fuzzy bias	100	100	73
Boundaries bias	100	100	**98**

Table 5-7: The average of RRTs branching attempts to reach the goal in WS3 workspace

Methods/ workspace	Without change	Minor change	Major change
No bias	206	462.8	1627.8
Goal bias	217	482	1728.6
Other Trees bias	284	609	1735.6
CD path bias	139	301	-
Fuzzy bias	50	214	823
Boundaries bias	**49**	**200**	**794**

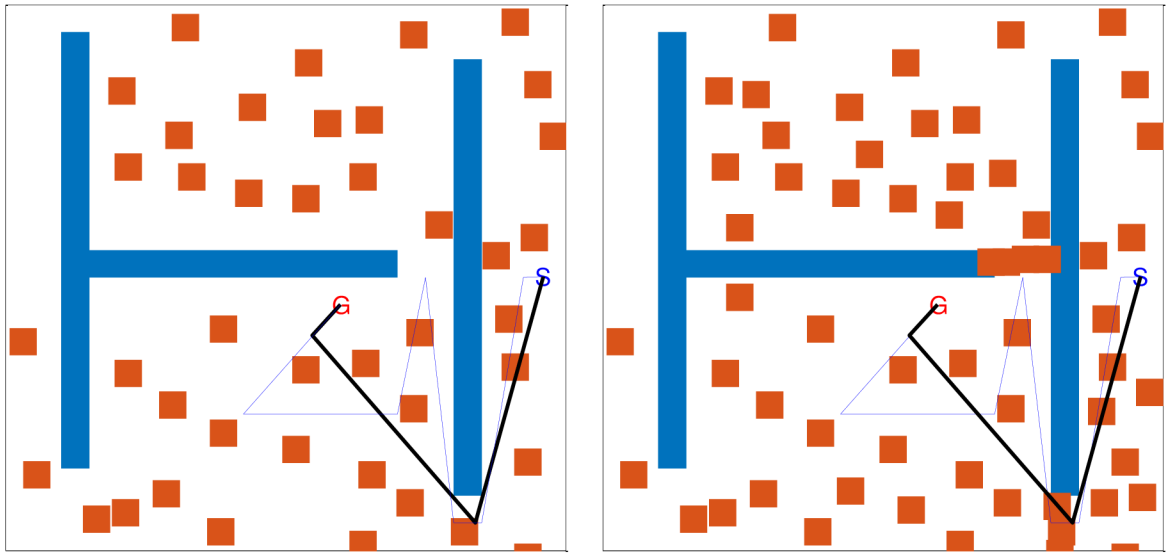


Figure 5-14: (a) The minor change in WS3, (b) the major change in WS3. The thin line represents the CD path, and bold line represents the shortened path. *G* and *S* represent the goal and the initial locations, respectively

In the fourth scenario, the path-planning problem is simulated in the "WS4" workspace, which is shown in Figure 5-15. The minor changes and the major changes are presented in Figure 5-16. The thin line represents the generated path using cell decomposition, and the bold line represents the shortened path of the original CD path. G and S represent the goal and the initial locations, respectively. The probabilistically completeness results are presented in Table 5-8. The iterations values are shown in Table 5-9.

In this test, we simulate the narrow passage and narrow area problems. The robot should pass through narrow and long corridors, which contains cluttered obstacles, and narrow connection between free regions.

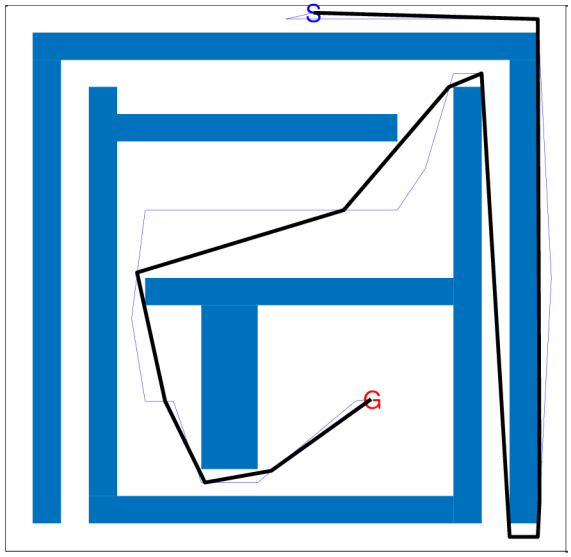
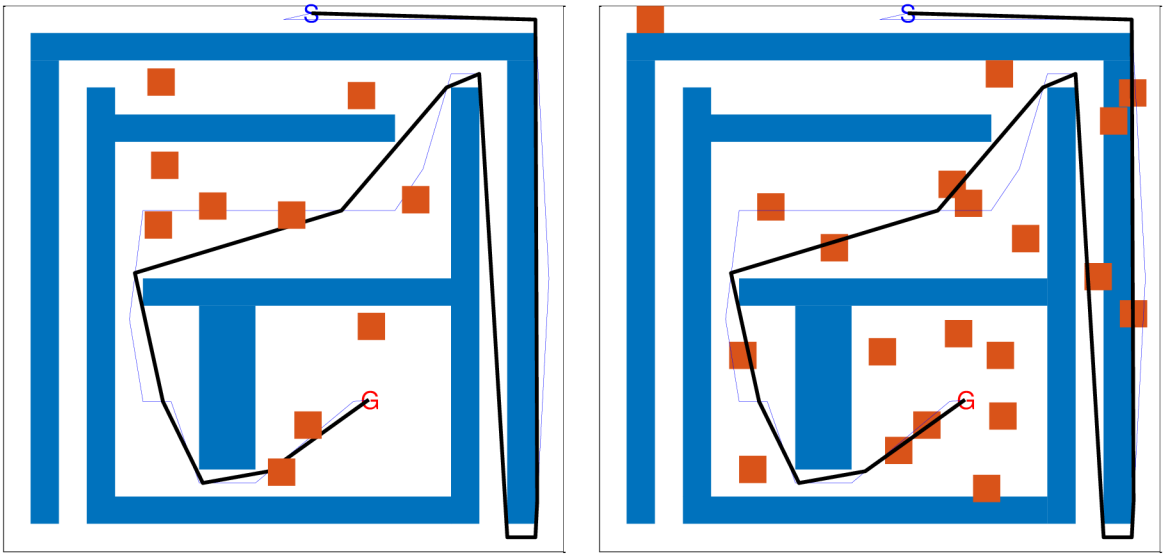


Figure 5-15: Basic workspace, WS4. The thin line represents the CD path, and bold line represents the shortened path. *G* and *S* represent the goal and the initial locations, respectively



(a) (b)

Figure 5-16: (a) The minor change in WS4, (b) the major change in WS4. The thin line represents the CD path, and bold line represents the shortened path. *G* and *S* represent the goal and the initial locations, respectively

Table 5-8: Number of successful attempts to reach the goal in WS4 workspace

Methods/ workspace	Without change	Minor change	Major change
No bias	0	0	0
Goal bias	0	0	0
Other Trees bias	0	0	0
CD path bias	0	0	0
Fuzzy bias	**100**	**98**	11
Boundaries bias	**100**	**98**	**12**

Table 5-9: The average of RRTs branching attempts to reach the goal in WS4 workspace

Methods/ workspace	Without change	Minor change	Major change
No bias	-	-	-
Goal bias	-	-	-
Other Trees bias	-	-	-
CD path bias	-	-	-
Fuzzy bias	326.9	422	748.7
Boundaries bias	**255.7**	**407.5**	**747.8**

Discussions

The results show that, the proposed planners work more efficiently than the other planners do in cluttered workspaces except in WS2 (the major change test). In all scenarios, the probabilistically completeness results, for both proposed planners, have a higher value in comparison to the other methods. Our planners navigate through all problems and find a path where the others competitors could not i.e. in WS4 tests.

The time of execution is not discussed here, because the execution time varies based on implementation platform and code optimization. Instead, the average of required iterations to find a solution is discussed.

During the simulation, the high impact of the sampling strategy is noticed on the results. In this work, the pseudo-random number generator is used to generate samples inside regions. The sampling strategies need more review and research as future work.

Summary

In this work a new hybrid planners have been proposed. The planners use rules-based adviser as a guidance toward the most important region in the space.

Each planner has two layers; the first one utilizes trapezoidal cell-decomposition algorithm to find a feasible path in the workspace. The second layer utilizes RRTs to find path in the configuration space. The information about the free regions, which is obtained from the first layer, is combined with the exploration information that is inferred from the second layer. The combination is done using rule-based adviser, which classifies the free regions and updates their weights.

These planners enhance the efficiency and completeness of the motion-planning problem in heterogeneous, cluttered, and dynamic workspaces. The planners exploit and combine the advantages of the exact cell decomposition in simple and low dimensional workspace, and the advantages of RRTs, which has a relatively higher tolerance to the changes in the environments.

The adviser of first planner biases and pulls the trees growth toward the boundary areas between explored and unexplored regions. The adviser of second planner uses the collision information and a fuzzy expert system to bias the trees growth toward low collision areas around the boundaries of explored regions.

The proposed methods are compared with other methods; the simulations results show that the proposed methods have better results, in terms of completeness and efficiency.

6 CONCLUSION

The aim of this dissertation was to improve the mobile robot path planning strategies, which, consequently, improves the robots autonomy and thus makes it more adaptable to our everyday life.

The goals of this thesis are fulfilled as many motion-planning algorithms and their applications in mobile robot path planning have been reviewed and simulated. Then, some of these algorithms were tested in 2D and 3D workspaces and the performance results were evaluated using statistical analyses. Based on these tests, the advantages and drawbacks of these methods were identified, and, new methods for path planning and path shortening were introduced to overcome the drawbacks and improve the performance.

The new motion planning methods are classified in three types. First, the cell decomposition based planners which generate a path that keeps a safety distance between the robot and the obstacle boundaries. At the same time, they perform the maneuvers through the large free regions in the workspace.

The second type uses hybrid two-layer planners which combine the advantages of RRT algorithms and CD approaches to overcome the difficulty when planning a path through narrow areas and dynamic workspaces.

The third type, the hybrid rule-based planner, utilizes the collected experience and expert knowledge base to produce better solution in an efficient way. This type of planner is constructed using multi-planning layers, i.e. the fuzzy expert system, RRT, and CD algorithms.

In this work, also new supportive methods were proposed to solve specific problems, for example the problem of navigation in a narrow area using sample-based algorithms. A combination of CD and minimum spanning tree has been proposed to identify the narrow passages and important regions in the workspaces.

The objectives of this work are met and the simulations show the ability of these planning approaches to solve different problems in the motion-planning domain. The simulation environment has been developed using Matlab to conduct the simulations and generate the numerical and graphical results, while the statistical analyses were done using Minitab and Matlab.

Naturally, the results open many new research questions. For example, determine the best sampling methods in the sampling-based algorithms. And, describe the impact of using different knowledge bases on path generating, i.e. the collision tendency, primitive local paths, etc.

BIBLIOGRAPHY

ABBADI, Ahmad and MATOUSEK, Radomil, 2012, RRTs Review and Statistical Analysis. *International journal of mathematics and computer in simulation*. 2012. Vol. 6, no. 1.

ABBADI, Ahmad and MATOUSEK, Radomil, 2014, Path Planning Implementation Using Matlab. In : *International Conference of Technical Computing Bratislava 2014*. Bratislava : Humusoft.cz. 11 April 2014. p. 1–5. ISBN 978-80-7080-898-6.

ABBADI, Ahmad and MATOUSEK, Radomil, 2015, Hybrid rule-based motion planner in cluttered workspace. *Soft Computing*. 2015.

ABBADI, Ahmad, MATOUSEK, Radomil, JANCIK, Stanislav and ROUPEC, Jan, 2012, Rapidly-exploring random trees: 3D planning. In : *Mendel*. 2012. p. 594–599.

ABBADI, Ahmad, MATOUSEK, Radomil and KNISPEL, Lukas, 2015, Narrow passage identification using cell decomposition approximation and minimum spanning tree. In : *Mendel*. 2015. p. 131–138.

ABBADI, Ahmad, MATOUSEK, Radomil, KRCEK, Petr and SOUSTEK, Petr, 2011, RRTs Review and Options. In : *computational Engineering in Systems Applications*. 2011. p. 194–199.

ABBADI, Ahmad, MATOUSEK, Radomil, OSMERA, Pavel and LUKAS KNISPEL, 2014, Spatial Guidance to RRT Planner Using Cell-decomposition Algorithm. In : *20th International Conference on Soft Computing, MENDEL 2014*. 25 June 2014. ISBN 978-80-214-4984-8.

ABBADI, Ahmad and PRENOSIL, Vaclav, 2015a, Safe Path Planning Using Cell Decomposition Approximation. In : *International Conference DISTANCE LEARNING, SIMULATION AND COMMUNICATION*. Brno : University of Defence, Brno. 2015. p. 8–14. ISBN 978-80-7231-992-3.

ABBADI, Ahmad and PRENOSIL, Vaclav, 2015b, Collided Path Replanning in Dynamic Environments Using RRT and Cell Decomposition Algorithms. In : *Modelling and Simulation for Autonomous Systems*. Cham : Springer International Publishing. p. 131–143. ISBN 978-3-319-22382-7. Available from: http://link.springer.com/10.1007/978-3-319-22383-4_9

AGUIRRE, Eugenio and GONZÁLEZ, Antonio, 2000, Fuzzy behaviors for mobile robot navigation: design, coordination and fusion. *International Journal of Approximate Reasoning*. November 2000. Vol. 25, no. 3, p. 255–289. DOI 10.1016/S0888-613X(00)00056-6.

AL-JAZARI-WIKIPEDIA, 2014, al-Jazari - Wikipedia, the free encyclopedia. . 2014. Available from: http://en.wikipedia.org/wiki/Al-Jazari

ALMAHAIRI, Amjad, 2010, Rapidly-Exploring Random Trees in Highly Constrained Environments. *McGill University, Mobile Robotics Project*. 2010.

AMATO, Nancy M., BAYAZIT, O. Burchan, DALE, Lucia K., JONES, Christopher and VALLEJO, Daniel, 1998, OBPRM: An Obstacle-Based PRM for 3D Workspaces. In : *WAFR '98 Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics : the algorithmic perspective*. 1998. p. 155–168. ISBN 1-56881-081-4.

AMIT PATEL, 2014, Introduction to A*. . 2014. Available from: http://www.redblobgames.com/pathfinding/a-star/introduction.html

ARAMBULA COSÍO, F. and PADILLA CASTAÑEDA, M. A., 2004, Autonomous robot navigation using adaptive potential fields. *Mathematical and Computer Modelling*. November 2004. Vol. 40, no. 9–10, p. 1141–1156. DOI 10.1016/j.mcm.2004.05.001.

ATKESON, Christopher G. and MORIMOTO, Jun, 2003, Nonparametric Representation of Policies and Value Functions: A Trajectory-Based Approach. In : *In NIPS 15*. MIT Press. 2003. p. 1611–1618.

ATRAMENTOV, A. and LAVALLE, S.M., 2002, Efficient nearest neighbor searching for motion planning. In : *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA '02*. 2002. p. 632–637 vol.1.

AURENHAMMER, Franz, 1991, Voronoi Diagrams-a Survey of a Fundamental Geometric Data Structure. *ACM Comput. Surv.* September 1991. Vol. 23, no. 3, p. 345–405. DOI 10.1145/116873.116880.

AURENHAMMER, Franz and KLEIN, Rolf, 2000, Voronoi diagrams. *Handbook of computational geometry*. 2000. Vol. 5, p. 201–290.

BAGINSKI, Boris, 1996, The Z^3-Method for Fast Path Planning in Dynamic Environments. In : *Proc. IASTED Conf. Applications of Control and Robotics*. 1996. p. 47–52.

BARRAQUAND, J. and LATOMBE, J.-C., 1990, A Monte-Carlo algorithm for path planning with many degrees of freedom. In : *, 1990 IEEE International Conference on Robotics and Automation, 1990. Proceedings*. May 1990. p. 1712–1717 vol.3.

BARRAQUAND, J. and LATOMBE, J.-C., 1991, Robot Motion Planning: A Distributed Representation Approach. *The International Journal of Robotics Research*. 1 December 1991. Vol. 10, no. 6, p. 628–649. DOI 10.1177/027836499101000604.

BERENSON, D., ABBEEL, P. and GOLDBERG, K., 2012, A robot path planning framework that learns from experience. In : *2012 IEEE International Conference on Robotics and Automation (ICRA)*. May 2012. p. 3671–3678.

BERNARD CHAZELLE, 1987, *Algorithmic and geometric aspects of robotics*. Hillsdale, N.J : L. Erlbaum Associates. Advances in robotics, vol. 1. ISBN 0-89859-554-1.

BOOR, V., OVERMARS, M.H. and VAN DER STAPPEN, A.F., 1999, The Gaussian sampling strategy for probabilistic roadmap planners. In : *1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings*. 1999. p. 1018–1023 vol.2.

BORENSTEIN, J. and KOREN, Y., 1991, The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*. June 1991. Vol. 7, no. 3, p. 278–288. DOI 10.1109/70.88137.

BOTTASSO, C.L., LEONELLO, D. and SAVINI, B., 2008, Path Planning for Autonomous Vehicles by Trajectory Smoothing Using Motion Primitives. *IEEE Transactions on Control Systems Technology*. November 2008. Vol. 16, no. 6, p. 1152–1168. DOI 10.1109/TCST.2008.917870.

BROOKS, R.A. and LOZANO-PEREZ, T., 1985, A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man and Cybernetics*. March 1985. Vol. SMC-15, no. 2, p. 224–233. DOI 10.1109/TSMC.1985.6313352.

BRUCE, J. and VELOSO, M., 2002, Real-time randomized path planning for robot navigation. In : *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002*. 2002. p. 2383–2388 vol.3.

BU, Tian-Ming, LI, Zhen-Jian and SUN, Zheng, 2005, Adaptive and relaxed visibility-based PRM. In : *2005 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2005. p. 174–179.

BUNIYAMIN, N., WAN NGAH, W. A. J., SARIFF, N. and MOHAMAD, Z., 2011, A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development*. 2011. Vol. 5, no. 2, p. 151–159.

CATMULL, Edwin and ROM, Raphael, 1974, A CLASS OF LOCAL INTERPOLATING SPLINES. In : *Computer Aided Geometric Design*. Elsevier. p. 317–326. ISBN 978-0-12-079050-0.

CHEN, Yang, CHENG, Lei, WU, Huaiyu, ZHAO, Xingang and HAN, Jianda, 2014, Knowledge-driven path planning for mobile robots: relative state tree. *Soft Computing*. 9 May 2014. DOI 10.1007/s00500-014-1299-4.

CHENG, Peng and LAVALLE, S.M., 2001, Reducing metric sensitivity in randomized trajectory design. In : *2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001. Proceedings*. 2001. p. 43–48 vol.1.

CHENG, Peng and LAVALLE, S.M., 2002, Resolution complete rapidly-exploring random trees. In : *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA '02*. 2002. p. 267–272 vol.1.

CHOI, Jinwoo, CHOI, Minyong, NAM, Sang Yep and CHUNG, Wan Kyun, 2011, Autonomous topological modeling of a home environment and topological localization using a sonar grid map. *Autonomous Robots*. 1 May 2011. Vol. 30, no. 4, p. 351–368. DOI 10.1007/s10514-011-9223-6.

CHOSET, Howie and BURDICK, Joel, 2000, Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research*. 2000. Vol. 19, no. 2, p. 96–125.

CHOSET, HOWIE, LYNCH, KEVIN M. and HUTCHINSON, SETH, 2005, *Principles of Robot Motion : Theory, Algorithms, and Implementation*. MIT Press. ISBN 978-0-262-03327-5.

DE BERG, Mark, CHEONG, Otfried, VAN KREVELD, Marc and OVERMARS, Mark, 2008, *Computational Geometry*. Berlin, Heidelberg : Springer Berlin Heidelberg. ISBN 978-3-540-77973-5.

DENNY, Jory and AMATO, N.M., 2011, Toggle PRM: Simultaneous mapping of C-free and C-obstacle - A study in 2D -. In : *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 2011. p. 2632–2639.

DENNY, J., MORALES, M., RODRIGUEZ, S. and AMATO, N.M., 2013, Adapting RRT growth for heterogeneous environments. In : *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. November 2013. p. 1772–1778.

DRIANKOV, Dimiter and SAFFIOTTI, Alessandro (eds.), 2001, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*. Heidelberg : Physica-Verlag HD. Studies in Fuzziness and Soft Computing. ISBN 978-3-7908-2479-7.

ESPOSITO, Joel M., 2013, Conditional Density Growth (CDG) model: a simplified model of RRT coverage for kinematic systems. *Robotica*. August 2013. Vol. 31, no. 05, p. 733–746. DOI 10.1017/S0263574712000690.

ETYMONLINE, 2014, Online Etymology Dictionary. . 2014. Available from: http://www.etymonline.com/index.php?allowed_in_frame=0&search=Robot&searchmode=none

FABBRI, R., ESTROZI, L. F. and COSTA, L. F., 2002, On Voronoi diagrams and medial axes. *Journal of Mathematical Imaging and Vision*. 2002. Vol. 17, p. 27–40.

GARRIDO, Santiago, MORENO, Luis, BLANCO, Dolores and JUREWICZ, Piotr, 2011, Path planning for mobile robot navigation using voronoi diagram and fast marching. *International Journal of Robotics and Automation (IJRA)*. 2011. Vol. 2, no. 1, p. 42–64.

GLAVINA, B., 1990, Solving findpath by combination of goal-directed and randomized search. In : *, 1990 IEEE International Conference on Robotics and Automation, 1990. Proceedings*. May 1990. p. 1718–1723 vol.3.

HANI ALSAFADI, 2007, Local Path Planning Using Potential Field. . 2007. Available from: http://www.cs.mcgill.ca/~hsafad/robotics/

HSU, D., JIANG, Tingting, REIF, J. and SUN, Zheng, 2003, The bridge test for sampling narrow passages with probabilistic roadmap planners. In : *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*. September 2003. p. 4420–4426.

HWANG, Y.K. and AHUJA, N., 1992, A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*. February 1992. Vol. 8, no. 1, p. 23–32. DOI 10.1109/70.127236.

HWANG, Joo Young, KIM, Jun Song, LIM, Sang Seok and PARK, Kyu Ho, 2003, A fast path planning by path graph optimization. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*. January 2003. Vol. 33, no. 1, p. 121–129. DOI 10.1109/TSMCA.2003.812599.

JAILLET, L., HOFFMAN, J., VAN DEN BERG, J., ABBEEL, P., PORTA, J.M. and GOLDBERG, K., 2011, EG-RRT: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles. In : *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 2011. p. 2646–2652.

JARADAT, Mohammad Abdel Kareem, GARIBEH, Mohammad H. and FEILAT, Eyad A., 2012, Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field. *Soft Computing*. January 2012. Vol. 16, no. 1, p. 153–164. DOI 10.1007/s00500-011-0742-z.

JIANDONG, Zhong and JIANBO, Su, 2011, Narrow passages identification for Probabilistic Roadmap Method. In : *Control Conference (CCC), 2011 30th Chinese*. July 2011. p. 3908–3912.

JOHANN BORENSTEIN, 1990, VFF and VFH -- Fast Obstacle Avoidance for Mobile Robots. . 1990. Available from: http://www-personal.umich.edu/~johannb/vff&vfh.htm

KAMON, Ishay, RIMON, Elon and RIVLIN, Ehud, 1998, TangentBug: A Range-Sensor-Based Navigation Algorithm. *The International Journal of Robotics Research*. 1 September 1998. Vol. 17, no. 9, p. 934–953. DOI 10.1177/027836499801700903.

KARAMAN, Sertac and FRAZZOLI, Emilio, 2011, Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*. 1 June 2011. Vol. 30, no. 7, p. 846–894. DOI 10.1177/0278364911406761.

KARAMAN, Sertac and FRAZZOLI, Emilio, 2012, *Sampling-based algorithms for optimal path planning problems*. Massachusetts Institute of Technology.

KATEVAS, Nikos I., TZAFESTAS, Spyros G. and PNEVMATIKATOS, Christos G., 1998, The Approximate Cell Decomposition with Local Node Refinement Global Path Planning Method: Path Nodes Refinement and Curve Parametric Interpolation. *Journal of Intelligent and Robotic Systems*. 1 July 1998. Vol. 22, no. 3-4, p. 289–314. DOI 10.1023/A:1008034314006.

KAVRAKI, L.E., SVESTKA, P., LATOMBE, J.-C. and OVERMARS, M.H., 1996, Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*. August 1996. Vol. 12, no. 4, p. 566–580. DOI 10.1109/70.508439.

KELOUWANI, Sousso, 2013, Human-Robot Collaborative Planning for Navigation Based on Optimal Control Theory. *Open Journal of Optimization*. 2013. Vol. 02, no. 03, p. 72–79. DOI 10.4236/ojop.2013.23010.

KHATIB, O., 1985, Real-time obstacle avoidance for manipulators and mobile robots. In : *1985 IEEE International Conference on Robotics and Automation. Proceedings*. March 1985. p. 500–505.

KIM, J. and KHOSLA, P., 1991, Real-time obstacle avoidance using harmonic potential functions. In : *, 1991 IEEE International Conference on Robotics and Automation, 1991. Proceedings*. April 1991. p. 790–796 vol.1.

KITO, T., OTA, J., KATSUKI, R., MIZUTA, T., ARAI, T., UEYAMA, T. and NISHIYAMA, T., 2003, Smooth path planning by using visibility graph-like method. In : *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*. September 2003. p. 3770–3775 vol.3.

KNEPPER, R.A., SRINIVASA, S.S. and MASON, Matthew T., 2010, Hierarchical planning architectures for mobile manipulation tasks in indoor environments. In : *2010 IEEE International Conference on Robotics and Automation (ICRA)*. May 2010. p. 1985–1990.

KUFFNER, J.J. and LAVALLE, S.M., 2000, RRT-connect: An efficient approach to single-query path planning. In : *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00*. 2000. p. 995–1001 vol.2.

KUFFNER, J.J. and LAVALLE, S.M., 2011, Space-filling trees: A new perspective on incremental search for motion planning. In : *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 2011. p. 2199–2206.

LATOMBE, Jean-Claude, 1991, *Robot motion planning*. Boston : Kluwer Academic Publishers. ISBN 0-7923-9129-2.

LAVALLE, Steven M., 1998, *Rapidly-Exploring Random Trees: A New Tool for Path Planning*.

LAVALLE, Steven Michael, 2006, *Planning algorithms*. Cambridge ; New York : Cambridge University Press. ISBN 0-521-86205-1.

LAVALLE, Steven M., BRANICKY, Michael S. and LINDEMANN, Stephen R., 2004, On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research*. 2004. Vol. 23, no. 7-8, p. 673–692.

LAVALLE, Steven M. and KUFFNER, James J., 2001, Randomized Kinodynamic Planning. *The International Journal of Robotics Research*. 1 May 2001. Vol. 20, no. 5, p. 378–400. DOI 10.1177/02783640122067453.

LAVALLE, Steven M., KUFFNER, James J. and JR., 2000, Rapidly-Exploring Random Trees: Progress and Prospects. In : *Algorithmic and Computational Robotics: New Directions*. 2000. p. 293–308.

LEONARD, John J. and DURRANT-WHYTE, Hugh F., 1991, Simultaneous map building and localization for an autonomous mobile robot. In : *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*. Ieee. 1991. p. 1442–1447.

LIEN, Jyh-ming and LU, Yanyan, 2009, Planning Motion in Environments with Similar Obstacles. In : *Robotics: Science and Systems V*. Seattle, USA : MIT Press. 2009. ISBN 978-0-262-51463-7.

LI, Dachuan, LI, Qing, CHENG, Nong and SONG, Jingyan, 2012, Extended RRT-based path planning for flying robots in complex 3D environments with narrow passages. In : *2012 IEEE International Conference on Automation Science and Engineering (CASE)*. August 2012. p. 1173–1178.

LI, Jiadong, LIU, Shirong, ZHANG, Botao and ZHAO, Xiaodan, 2014, RRT-A* Motion planning algorithm for non-holonomic mobile robot. In : *SICE Annual Conference (SICE), 2014 Proceedings of the*. September 2014. p. 1833–1838.

LIN, Yu-Te, 2006, The Gaussian PRM Sampling for Dynamic Configuration Spaces. In : *9th International Conference on Control, Automation, Robotics and Vision, 2006. ICARCV '06*. December 2006. p. 1–5.

LINDEMANN, Stephen R. and LAVALLE, Steven M., 2003, Current Issues in Sampling-Based Motion Planning. . 2003. Vol. 15, p. 36–54.

LINDEMANN, S.R. and LAVALLE, S.M., 2004, Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In : *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*. April 2004. p. 3251–3257 Vol.4.

LIU, Hong, RAO, Kai and XIAO, Fang, 2013, Obstacle guided RRT path planner with region classification for changing environments. In : *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. December 2013. p. 164–171.

LOZANO-PÉREZ, Tomás and WESLEY, Michael A., 1979, An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles. *Commun. ACM*. October 1979. Vol. 22, no. 10, p. 560–570. DOI 10.1145/359156.359164.

LULU, L. and ELNAGAR, A., 2005, A comparative study between visibility-based roadmap path planning algorithms. In : *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005)*. August 2005. p. 3263–3268.

LUMELSKY, Vladimir J. and STEPANOV, A.A., 1986, Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*. November 1986. Vol. 31, no. 11, p. 1058–1063. DOI 10.1109/TAC.1986.1104175.

MACIEJ KALISIAK and VAN DE PANNE, Michiel, 2006, RRT-blossom: RRT with a local flood-fill behavior. In : . 2006.

MARTIN, S.R., WRIGHT, S.E. and SHEPPARD, J.W., 2007, Offline and Online Evolutionary Bi-Directional RRT Algorithms for Efficient Re-Planning in Dynamic Environments. In : *IEEE International Conference on Automation Science and Engineering, 2007. CASE 2007*. September 2007. p. 1131–1136.

MASEHIAN, E., AMIN-NASERI, M.R. and KHADEM, S.E., 2003, Online motion planning using incremental construction of medial axis. In : *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*. September 2003. p. 2928–2933 vol.3.

MASEHIAN, Ellips and NASERI, Amin, 2010, Mobile Robot Online Motion Planning Using Generalized Voronoi Graphs. *Journal of Industrial Engineering*. 2010. Vol. 5, p. 1–15.

MASOUD, Ahmad A., 2013, A harmonic potential field approach for joint planning and control of a rigid, separable nonholonomic, mobile robot. *Robotics and Autonomous Systems*. June 2013. Vol. 61, no. 6, p. 593–615. DOI 10.1016/j.robot.2013.02.007.

MATHIA, Karl, 2010, *Robotics for Electronics Manufacturing: Principles and Applications in Cleanroom Automation*. Cambridge University Press.

MAZER, Emmanuel, AHUACTZIN, Juan Manuel and BESSIERE, Pierre, 1998, The Ariadne's clew algorithm. *Journal of Artificial Intelligence Research*. 1998. Vol. 9, no. 1. DOI 10.1613/jair.468.

MBEDE, Jean Bosco, HUANG, Xinhan and WANG, Min, 2000, Fuzzy motion planning among dynamic obstacles using artificial potential fields for robot manipulators. *Robotics and Autonomous Systems*. 31 July 2000. Vol. 32, no. 1, p. 61–72. DOI 10.1016/S0921-8890(00)00073-7.

MCFETRIDGE, L. and YOUSEF IBRAHIM, M., 1998, New technique of mobile robot navigation using a hybrid adaptive fuzzy potential field approach. *Computers & Industrial Engineering*. December 1998. Vol. 35, no. 3–4, p. 471–474. DOI 10.1016/S0360-8352(98)00136-3.

MILITÃO, Filipe, NADEN, Karl and TONINHO, Bernardo, 2010, Improving RRT with Context Sensitivity 15-780 Grad AI. In : . 2010.

MONTANER, Marc Boumedine and RAMIREZ-SERRANO, Alejandro, 1998, Fuzzy knowledge-based controller design for autonomous robot navigation. *Expert Systems with Applications*. January 1998. Vol. 14, no. 1–2, p. 179–186. DOI 10.1016/S0957-4174(97)00059-6.

MOORE, K.L. and FLANN, N.S., 1999, Hierarchical task decomposition approach to path planning and control for an omni-directional autonomous mobile robot. In : *Proceedings of the 1999 IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics, 1999*. 1999. p. 302–307.

MORALES, Marco, TAPIA, Lydia, PEARCE, Roger, RODRIGUEZ, Samuel and AMATO, Nancy M., 2005, A Machine Learning Approach for Feature-Sensitive Motion Planning. In : *Algorithmic Foundations of Robotics VI*. Berlin, Heidelberg : Springer Berlin Heidelberg. p. 361–376. ISBN 978-3-540-25728-8. Available from: http://link.springer.com/10.1007/10991541_25

NASIR, Jauwairia, ISLAM, Fahad, MALIK, Usman, AYAZ, Yasar, HASAN, Osman, KHAN, Mushtaq and MUHAMMAD, Mannan Saeed, 2013, RRT*-SMART: A Rapid Convergence Implementation of RRT*. *International Journal of Advanced Robotic Systems*. 2013. Vol. 10.

NEGNEVITSKY, Michael, 2005, *Artificial intelligence: a guide to intelligent systems*. 2nd ed. Harlow, England ; New York : Addison-Wesley. ISBN 0-321-20466-2.

NG, James and BRÄUNL, Thomas, 2007, Performance Comparison of Bug Navigation Algorithms. *Journal of Intelligent and Robotic Systems*. 2 August 2007. Vol. 50, no. 1, p. 73–84. DOI 10.1007/s10846-007-9157-6.

NISSOUX, C., SIMEON, T. and LAUMOND, J-P, 1999, Visibility based probabilistic roadmaps. In : *1999 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999. IROS '99. Proceedings*. 1999. p. 1316–1321 vol.3.

PENG CHENG, 2001, *Reducing RRT metric sensitivity for motion planning with differential constraints*. Graduate College Iowa State University.

PEREZ, A., PLATT, R., KONIDARIS, G., KAELBLING, L. and LOZANO-PEREZ, T., 2012, LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. In : *2012 IEEE International Conference on Robotics and Automation (ICRA)*. May 2012. p. 2537–2542.

PÊTRÈS, C., ROMERO-RAMIREZ, M. -A. and PLUMET, F., 2012, A potential field approach for reactive navigation of autonomous sailboats. *Robotics and Autonomous Systems*. December 2012. Vol. 60, no. 12, p. 1520–1527. DOI 10.1016/j.robot.2012.08.004.

PETR KRCEK and JIŘÍ DVORAK, 2004, MOBILE ROBOT MOTION CONTROL BY MEANS OF FUZZY RULES. In : *Engineering Mechanics 2004*. Svratka : nstitute of Thermomechanics AS CR, v.v.i., Prague. May 2004.

RODRIGUEZ, S., TANG, Xinyu, LIEN, Jyh-Ming and AMATO, N.M., 2006, An obstacle-based rapidly-exploring random tree. In : *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. May 2006. p. 895–900.

ROSELL, J., CRUZ, L., SUAREZ, R. and PEREZ, A., 2011, Importance sampling based on adaptive principal component analysis. In : *2011 IEEE International Symposium on Assembly and Manufacturing (ISAM)*. May 2011. p. 1–6.

ROSELL, J. and INIGUEZ, P., 2005, Path planning using Harmonic Functions and Probabilistic Cell Decomposition. In : *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005*. April 2005. p. 1803–1808.

SAFFIOTTI, A., 1997, The uses of fuzzy logic in autonomous robot navigation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*. 16 December 1997. Vol. 1, no. 4, p. 180–197. DOI 10.1007/s005000050020.

SAHA, Mitul, LATOMBE, Jean-claude, CHANG, Yu-chi and PRINZ, Friedrich, 2005, Finding narrow passages with probabilistic roadmaps: The small step retraction method. In : *in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*. 2005.

SAJJA, Priti Srinivas and RAJENDRA AKERKAR (eds.), 2010, *Advanced Knowledge-Based Systems: Models, Applications and Research*. ISBN 978-81-908426-0-0.

SAKAHARA, H., MASUTANI, Y. and MIYAZAKI, F., 2008, Real-time motion planning in unknown environment: Voronoi-based StRRT (Spatiotemporal RRT). In : *SICE Annual Conference, 2008*. August 2008. p. 2326–2331.

SALOMON, David, 2011, *The Computer Graphics Manual*. London : Springer London. Texts in Computer Science. ISBN 978-0-85729-885-0.

SASIKUMAR, M., RAMANI, S., RAMAN, S. Muthu, ANJANEYULU, K. S. R. and CHANDRASEKAR, R., 2007, *A Practical Introduction to Rule Based Expert Systems*. Narosa Publishing House, New Delhi.

SCHWARTZ, Jacob T and SHARIR, Micha, 1983, On the "piano movers" problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*. September 1983. Vol. 4, no. 3, p. 298–351. DOI 10.1016/0196-8858(83)90014-3.

SCIENCEKIDS, 2014, History of Robotics - Timeline, AI, Industrial, Toy Robots, Robotic Arm, Technology. . 2014. Available from: http://www.sciencekids.co.nz/sciencefacts/technology/historyofrobotics.html

SEDA, Milos, 2007, Roadmap methods vs. cell decomposition in robot motion planning. In : *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*. World Scientific and Engineering Academy and Society (WSEAS). 2007. p. 127–132.

SFEIR, J., SAAD, M. and SALIAH-HASSANE, H., 2011, An improved Artificial Potential Field approach to real-time mobile robot path planning in an unknown environment. In : *2011 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*. September 2011. p. 208–213.

SHAREF, S.M., SA'ID, W.K. and KHOSHABA, F.S., 2010, A rule-based system for trajectory planning of an indoor mobile robot. In : *2010 7th International Multi-Conference on Systems Signals and Devices (SSD)*. 2010. p. 1–7.

SHIKIN, E. V. and PLIS, Alexander I., 1995, *Handbook on splines for the user*. Boca Raton : CRC Press. ISBN 0-8493-9404-X.

SHKOLNIK, Er and TEDRAKE, Russ, 2009, Path planning in 1000+ dimensions using a task-space voronoi bias. In : *In IEEE International Conference on Robotics and Automation*. 2009.

SIDDHARTHA SRINIVASA, 2013, *Sampling-Based Methods, Lecture 12*. 2013. [Accessed 22 December 2014]. Available from: https://personalrobotics.ri.cmu.edu/courses/16662/notes/rrt/16662_Lecture12.pdf

SLEUMER, Nora H. and TSCHICHOLD-GÜRMAN, Nadine, 1999, *Exact Cell Decomposition of Arrangements used for Path Planning in Robotics*.

SLOVNÍK, Slovník spisovného jazyka českého. . Available from: http://ssjc.ujc.cas.cz/search.php?hledej=Hledat&heslo=robot&sti=EMPTY&where=hesla&hsubstr=no

SMITH, Randall C., 1986, Development System for Flexible Assembly System. . 1986. DOI 10.1177/027836498600500404.

SMOGAVEC, G. and ŽALIK, B., 2012, A fast algorithm for constructing approximate medial axis of polygons, using Steiner points. *Advances in Engineering Software*. October 2012. Vol. 52, p. 1–9. DOI 10.1016/j.advengsoft.2012.05.006.

STOLLE, M. and ATKESON, C.G., 2006, Policies based on trajectory libraries. In : *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. May 2006. p. 3344–3349.

STRANDBERG, M., 2004, Augmenting RRT-planners with local trees. In : *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*. April 2004. p. 3258–3262 Vol.4.

SUN, Zheng, HSU, D., JIANG, Tingting, KURNIAWATI, H. and REIF, J.H., 2005, Narrow passage sampling for probabilistic roadmap planning. *IEEE Transactions on Robotics*. December 2005. Vol. 21, no. 6, p. 1105–1115. DOI 10.1109/TRO.2005.853485.

Supersampling, 2015. , Available from: http://en.wikipedia.org/wiki/Supersampling

TAPIA, Lydia, THOMAS, Shawna, BOYD, Bryan and AMATO, Nancy M., 2009, An unsupervised adaptive strategy for constructing probabilistic roadmaps. In : *in Proc. IEEE Int. Conf. Robot. Autom. (ICRA*. 2009. p. 4037–4044.

TITAS BERA, M. SEETHARAMA BHAT and DEBASISH GHOSE, 2014, Analysis of Obstacle based Probabilistic RoadMap Method using Geometric Probability. In : *3rd International Conference on Advances in Control and Optimization of Dynamical Systems*. IIT-Kanpur, Kanpur, India. 2014. p. 462–469.

URMSON, C. and SIMMONS, R., 2003, Approaches for heuristically biasing RRT growth. In : *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings*. October 2003. p. 1178–1183 vol.2.

UYANIK, Kadir Firat, 2011, Social Robot Partners: Still Sci-fi? . 2011.

VAHRENKAMP, N., ASFOUR, T. and DILLMANN, R., 2007, Efficient motion planning for humanoid robots using lazy collision checking and enlarged robot models. In : *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007*. October 2007. p. 3062–3067.

VAN DEN BERG, J. P., 2005, Using Workspace Information as a Guide to Non-uniform Sampling in Probabilistic Roadmap Planners. *The International Journal of Robotics Research*. 1 December 2005. Vol. 24, no. 12, p. 1055–1071. DOI 10.1177/0278364905060132.

VENDRELL, Eduardo, MELLADO, Martín and CRESPO, Alfons, 2001, Robot planning and re-planning using decomposition, abstraction, deduction, and prediction. *Engineering Applications of Artificial Intelligence*. August 2001. Vol. 14, no. 4, p. 505–518. DOI 10.1016/S0952-1976(01)00027-6.

WANG, Quan, WANG, Wei and LI, Yan, 2012, A multi-RRT based hierarchical path planning method. In : *2012 IEEE 14th International Conference on Communication Technology (ICCT)*. November 2012. p. 971–975.

WANG, Wei, YAN, Li, XU, Xin and YANG, Simon X., 2010, An adaptive roadmap guided Multi-RRTs strategy for single query path planning. In : *2010 IEEE International Conference on Robotics and Automation (ICRA)*. May 2010. p. 2871–2876.

WIKIPEDIA, 2014a, History of robots - Wikipedia, the free encyclopedia. . 2014. Available from: http://en.wikipedia.org/wiki/History_of_robots

WIKIPEDIA, 2014b, Minkowski addition. *Wikipedia, the free encyclopedia*. 2014. Available from: http://en.wikipedia.org/w/index.php?title=Minkowski_addition&oldid=637892103

WILMARTH, S.A., AMATO, N.M. and STILLER, P.F., 1999, MAPRM: a probabilistic roadmap planner with sampling on the medial axis of the free space. In : *1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings*. 1999. p. 1024–1031 vol.2.

YANG, Kwangjin, 2013, An efficient Spline-based RRT path planner for non-holonomic robots in cluttered environments. In : *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. May 2013. p. 288–297.

YEH, Hsin-Yi, THOMAS, Shawna, EPPSTEIN, David and AMATO, Nancy M., 2012, UOBPRM: A uniformly distributed obstacle-based PRM. In : *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012. p. 2655–2662.

YERSHOVA, A. and LAVALLE, S.M., 2007, Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching. *IEEE Transactions on Robotics*. February 2007. Vol. 23, no. 1, p. 151–157. DOI 10.1109/TRO.2006.886840.

ZHANG, Qiushi, CHEN, Dandan and CHEN, Ting, 2012, An Obstacle Avoidance Method of Soccer Robot Based on Evolutionary Artificial Potential Field. *Energy Procedia*. 2012. Vol. 16, Part C, p. 1792–1798. DOI 10.1016/j.egypro.2012.01.276.

ZHONG, Jiandong and SU, Jianbo, 2012, Triple-Rrts for robot path planning based on narrow passage identification. In : *2012 International Conference on Computer Science and Information Processing (CSIP)*. August 2012. p. 188–192.

ZUCKER, Matt, KUFFNER, James and BAGNELL, J. Andrew, 2008, Adaptive workspace biasing for sampling-based planners. In : *Robotics and Automation, 2008. ICRA 2008*. Pasadena, CA : IEEE. May 2008. ISBN 978-1-4244-1647-9.

ZUCKER, M., KUFFNER, J. and BRANICKY, M., 2007, Multipartite RRTs for Rapid Replanning in Dynamic Environments. In : *2007 IEEE International Conference on Robotics and Automation*. April 2007. p. 1603–1609.

AUTHOR'S PUBLICATIONS

[1] A. Abbadi and R. Matousek, "Hybrid rule-based motion planner in cluttered workspace," Soft Computing, 2015. (Accepted)

[2] A. Abbadi and V. Prenosil, "Collided Path Replanning in Dynamic Environments Using RRT and Cell Decomposition Algorithms," in Modelling and Simulation for Autonomous Systems, vol. 9055, J. Hodicky, Ed. Cham: Springer International Publishing, 2015, pp. 131–143.

[3] A. Abbadi and R. Matousek, "RRTs Review and Statistical Analysis," International journal of mathematics and computer in simulation, vol. 6, no. 1, 2012.

[4] A. Abbadi, R. Matousek, and L. Knispel, "Narrow passage identification using cell decomposition approximation and minimum spanning tree," presented at the Mendel, 2015, vol. 2015-January, pp. 131–138.

[5] A. Abbadi and V. Prenosil, "Safe Path Planning Using Cell Decomposition Approximation," presented at the International Conference DISTANCE LEARNING, SIMULATION AND COMMUNICATION, Brno, 2015, vol. DLSC2015, pp. 8–14.

[6] L. Knispel, R. Matousek, A. Abbadi, and J. Dvorak, "A note about pseudo 3D grid approximation of landscape for a holonomic robot path planning with naïve path optimization," presented at the Mendel, 2015, vol. 2015-January, pp. 127–130.

[7] A. Abbadi and R. Matousek, "Path Planning Implementation Using Matlab," presented at the International Conference of Technical Computing Bratislava 2014, Bratislava, 2014, pp. 1–5.

[8] A. Abbadi, R. Matousek, P. Osmera, and Lukas Knispel, "Spatial Guidance to RRT Planner Using Cell-decomposition Algorithm," presented at the 20th International Conference on Soft Computing, MENDEL 2014, 2014.

[9] A. Abbadi, R. Matousek, S. Jancik, and J. Roupec, "Rapidly-exploring random trees: 3D planning," presented at the Mendel, 2012, pp. 594–599.

[10] S. Jancik, R. Matousek, J. Dvorak, and A. Abbadi, "Local navigation techniques by means of ICPF," in Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS), IEEE, 2012, 2012, pp. 1–7.

[11] S. Jancik, R. Matousek, J. Dvorak, and A. Abbadi, "The ICP for fragment identification," presented at the Mendel, 2012, pp. 588–593.

[12] A. Abbadi, R. Matousek, P. Krcek, and P. Soustek, "RRTs Review and Options," in computational Engineering in Systems Applications, 2011, vol. 2, pp. 194–199.

A. APPENDIX: MATLAB IMPLEMENTATION

In this appendix, the main simulation software is presented. It is used for conducting tests, simulations, generating the numerical and graphical results and building workspaces map in graphical and interactive way.

This chapter is divided into two sections, the first one describes very concisely RRT and cell decomposition implementation on Matlab, while the second section lists some GUI snapshots of the software.

Some parts of the code where published online for robotics community as an open-source software[1].

Path planning implementation using Matlab

This section is based on a technical paper published in (Abbadi, Matousek 2014) for describing the implementation of the RRT planner and the cell decomposition algorithms in Matlab environment.

The basic RRT implementation is shown in Figure A-1. It shows the main structure for the RRT's class.

```
%%RRT class
CLASSDEF RRTClass<handle & cSpace
   PROPERTIES
    cords; parent; startPos;goalPos; maxIteration;
    extensionStep= E; rrtType='basic_RRT';
% bias to goal, other trees, specific points,…
    bias.enable=1;
    bias.type=['biasToGoal', 'biasToTreePoints',…];
    bias.rangeVal=[0.05,0.07,…];
%biasToGoal in range 0-0.05=5%,biasToTreePoints in range 0.05-0.07=2% the
%rest is normal random point selection
   END
   METHODS
    FUNCTION RRT=RRTClass(initialValues) …...

    FUNCTION [objective,tElapsed]=rrtPlanner(RRT,drawType)
     tRRTStart=tic;
     FOR iter=1:RRT.maxIteration
        [objective]=grawTree(RRT);

% drawType :realtime draw, draw the result, don't draw
        RRT.draw(iter ,drawType);
        IF objective
          BREAK;
        END
     END
    tElapsed=toc(tRRTStart);
    END
```

[1] https://sourceforge.net/projects/celldecopositionmotionplanning/

```
FUNCTION [objective]=grawTree(RRT)
 objective=0;
  [randomConfiguration]=
                      RRT@cSpace.getRandomPoint(rrt.bias);
  [nearestConfiguration]= RRT.getNearestPoint(randomConfiguration);
  [newConfiguration]=
              RRT.branching(nearestConfiguration,randomConfiguration);
  IsCollid=RRT@cSpace.checkCollision(nearestConfiguration,
                                                    newConfiguration);
  IF IsCollid ; RETURN ; END;
  RRT.addToTree(nearestConfiguration, newConfiguration);
   [objective]= RRT.checkGoal();
END ….
```

Figure A-1: Selected lines of RRT code in Matlab

In our implementation, either the pseudo-random number generator is used to draw a random sample from the workspace, or the bias toward a specific set using "*getRandomPoint*" function, as shown in Figure A-2.

The bias to point/points set associated with some probability that represents the percentage of choosing a point from the points set. Example of points set are the bias toward the goal point, toward other trees-points, toward points around the goal, toward points drawn from old successful path, or toward points from important regions. We implement the function "*getBiasPoint*" to give users the freedom to specify the bias methods and the probability value to these biases.

```
%%%%%%%% configurationSpace class %%%%%%%
… ...
FUNCTION newPnt= getRandomPoint(CSpace ,bias)
      newPnt=[];
      IF bias.enable
            newPnt = CSpace.getBiasPoint(bias);
      END
      IF empty(newPnt)
            range=abs(CSpace.dimensions(:,2)- CSpace.dimensions(:,1));
            FOR i=1:size(CSpace.dimensions,1)
                  newPnt (1,i)= CSpace.dimensions(i,1)+range(i)*rand;
            END
      END
END
FUNCTION newPoint=getBiasPoint(CSpace ,bias)
      randVal=rand; %bias probability
      methodIndex=find(randVal <= bias.rangeVal,1,'first');
      biasMethod= bias.type(methodIndex);
      SWITCH biasMethod
            CASE 'biasToGoal'
                  newPnt =goalPos;
            CASE 'biasToTreePoints' % bias to one point in other trees
          …..
            CASE 'biasToGivenPoints'
           …..
                  randVal=randperm(size(CSpace.biasGivenPoints,1));
                  newPnt = CSpace.biasGivenPoints(randVal(1),:); …..
      END
END …..
```

Figure A-2: Selected lines of *getRandomPoint* function

The implementation of cell decomposition in Matlab finds the graph of adjacency graph based on sweep-line technique. Then, to deal with this generated graph the Bioinformatics toolbox functions was used. An example of these functions is "*graphshortestpath*" it searches over the graph for the shortest path between the initial and the goal positions. This function could be configured to use (Bellman-Ford, BFS, Acyclic, or Dijkstra) algorithms as a searching method. In this implementation, the Dijkstra's algorithm was used. Another useful function is "*graphallshortestpaths*" which gives all available shortest paths. Moreover, for graph visualization the "*biograph*" function was used to create graph object, and then draw it using "*view*" function, Figure A-3 shows selected lines of code that search and visualize the graph of adjacency. The result of this code is seen in Figure A-4.

```
%prepare Undirected Graph
wieght=1;
DG=sparse(graph.edges(:,1),graph.edges(:,2),wieght);
UG=tril(DG+DG');
% Graph search functions in Graph Theory, Bioinformatics Toolbox
[dist,path]                                                                    =
graphshortestpath(graph,InitialPosition,GoalPosition,'directed', false);
%draw graph of adjacency
h=view(biograph(UG,cellstr(num2str([1:size(UG,1)]')),'ShowArrows','off','
ShowWeights','on'));
…..
```

Figure A-3: Search and draw graph, based on bioinformatics toolbox function

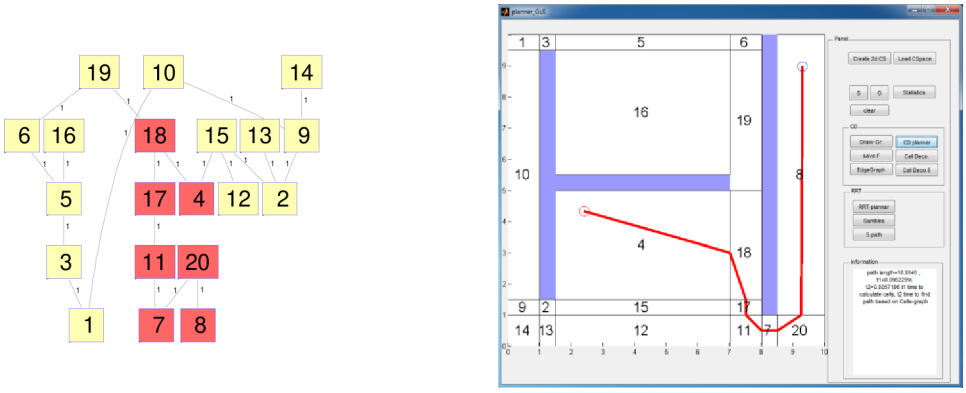
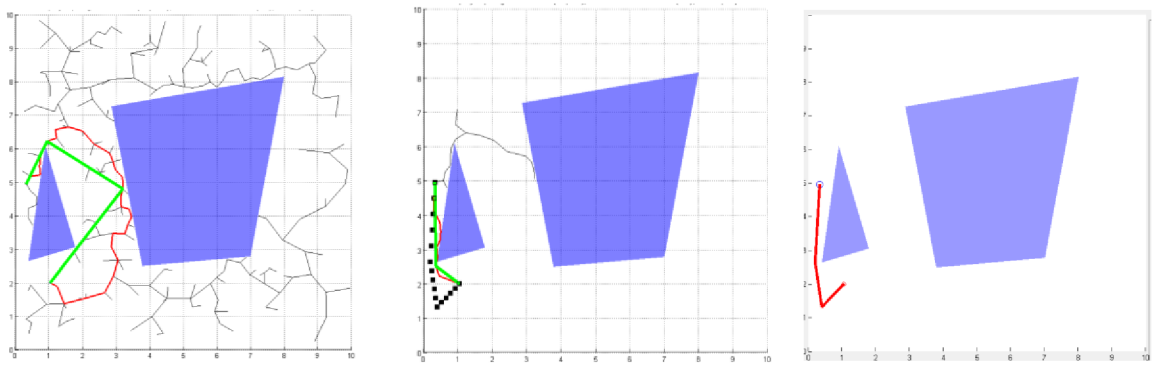


Figure A-4: Cell-decomposition planner GUI and the generated graph

The results from cell-decomposition and RRT algorithms are integrated together. Figure A-5-a, shows the RRT path without bias, and the CD path in Figure A-5-c, then the RRT path using the bias to CD path's points, as shown in Figure A-5-b.

The planner in Figure A-5-a, has to explore wide areas before it finds the goal, while the using of the bias-points increase the efficiency for RRT tree.



(a) (b) ©

Figure A-5: (a) RRT planner without bias, (b) RRT with bias toward cell-decomposition path's points, and (c) the cell-decomposition path

Software snapshots

The main window of the simulation software contains the working space window, as shown in Figure A-6-(8), and some general options, for example, place the goal and initial positions, and clear the workspace, Figure A-6-(1), load a workspace Figure A-6-(6), some statistics parameter Figure A-6-(5). It also contains some RRT parameters Figure A-6-(4), exact cell decomposition options Figure A-6-(3), cell decomposition approximation Figure A-6-(2), and the information bar Figure A-6-(7).

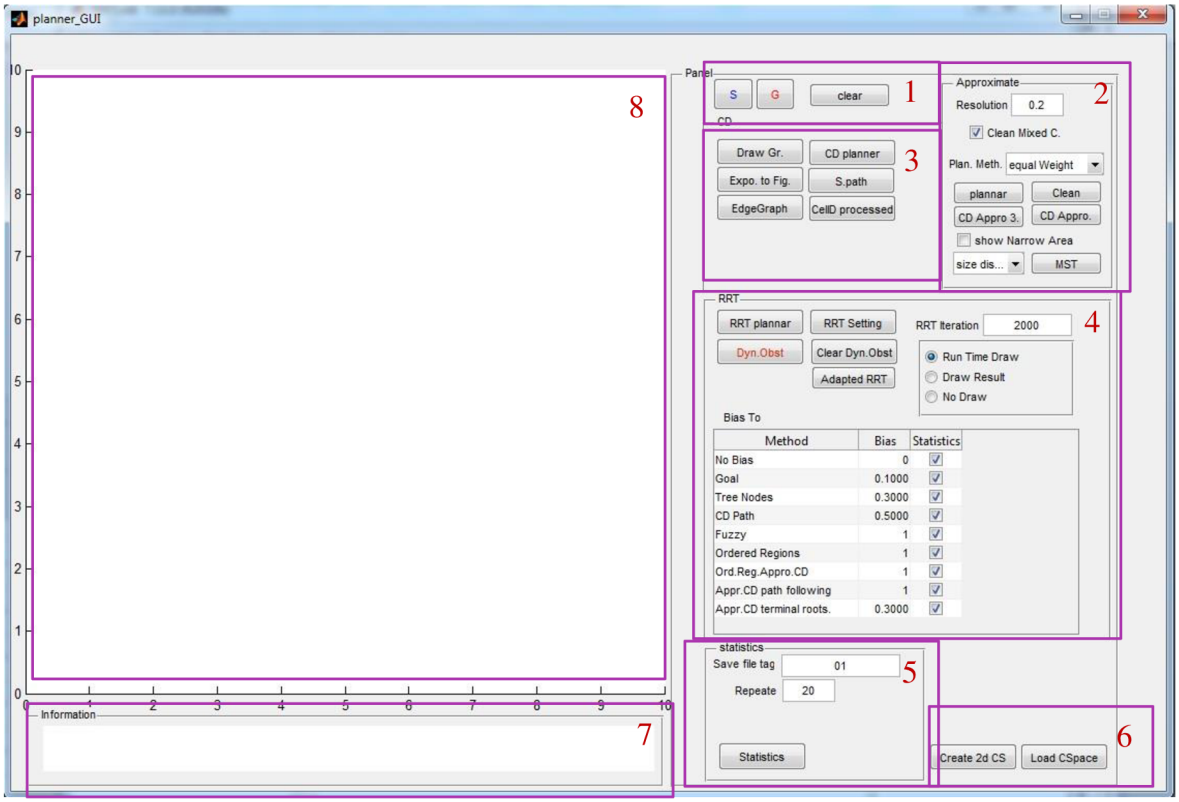


Figure A-6: Main Software window

The workspaces are drawn in separate window as shown in Figure A-7, where the user can draw polygon obstacles and modify the coordinates.

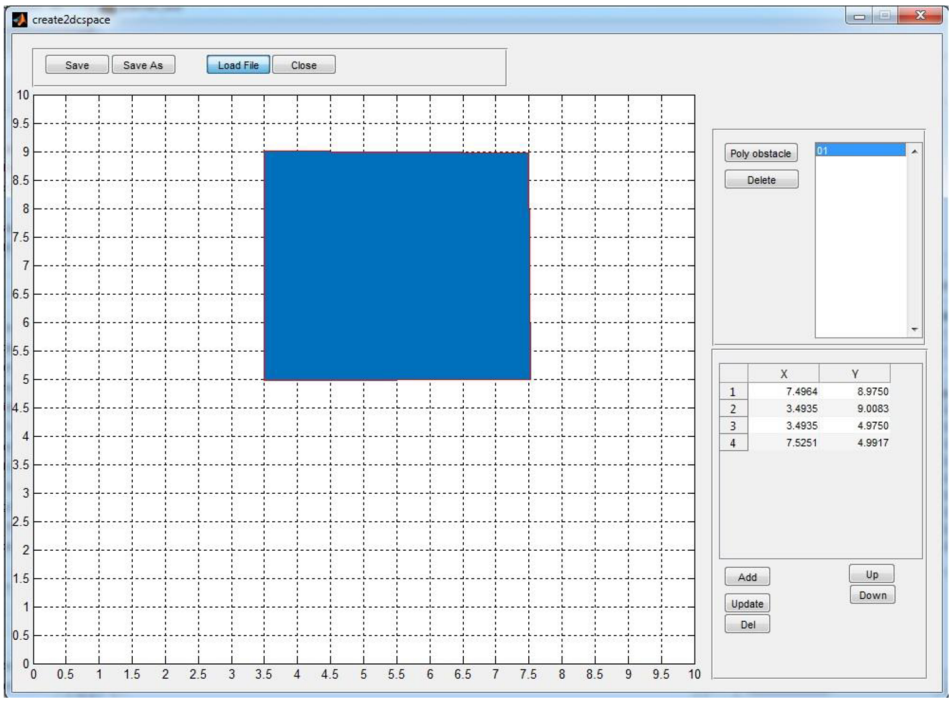


Figure A-7: Drawing workspace interface

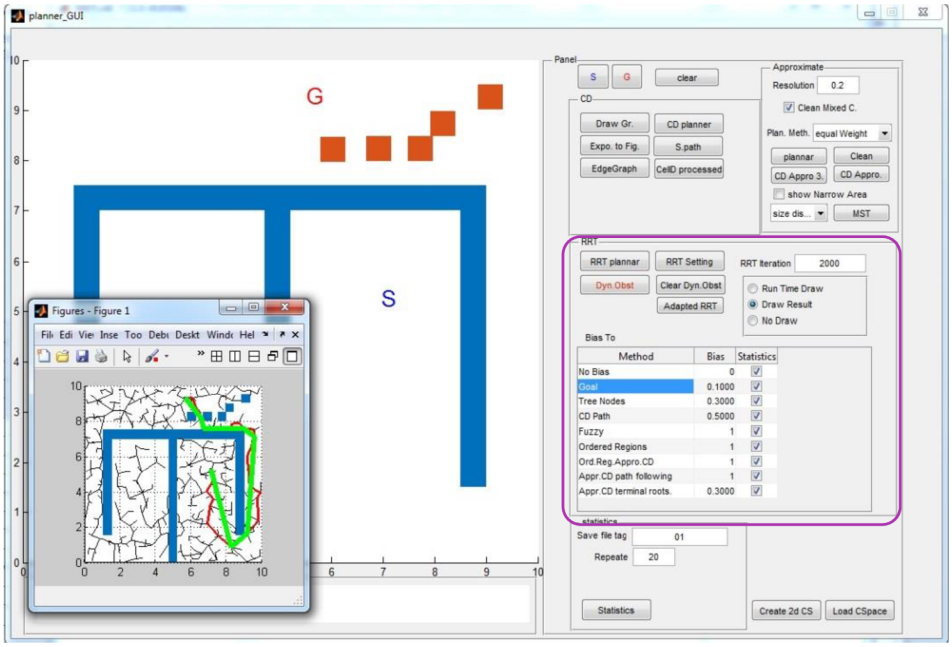


Figure A-8: RRT algorithm interfaces, and its parameters

The RRT algorithm can be set to bias to specific points for example the exact cell decomposition path[1] as shown in Figure A-9. It can be also used to simulate the dynamic workspace[2], as shown in Figure A-8.

[1] See section 4.1.4 and 5.4.1

[2] See section 4.1.5

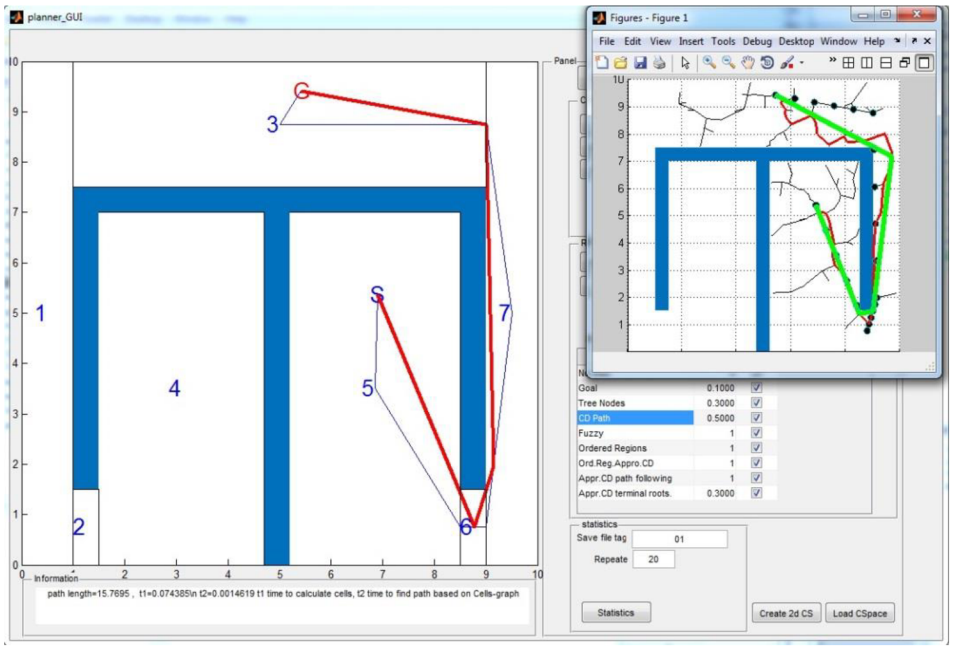


Figure A-9: RRT bias toward CD's path points

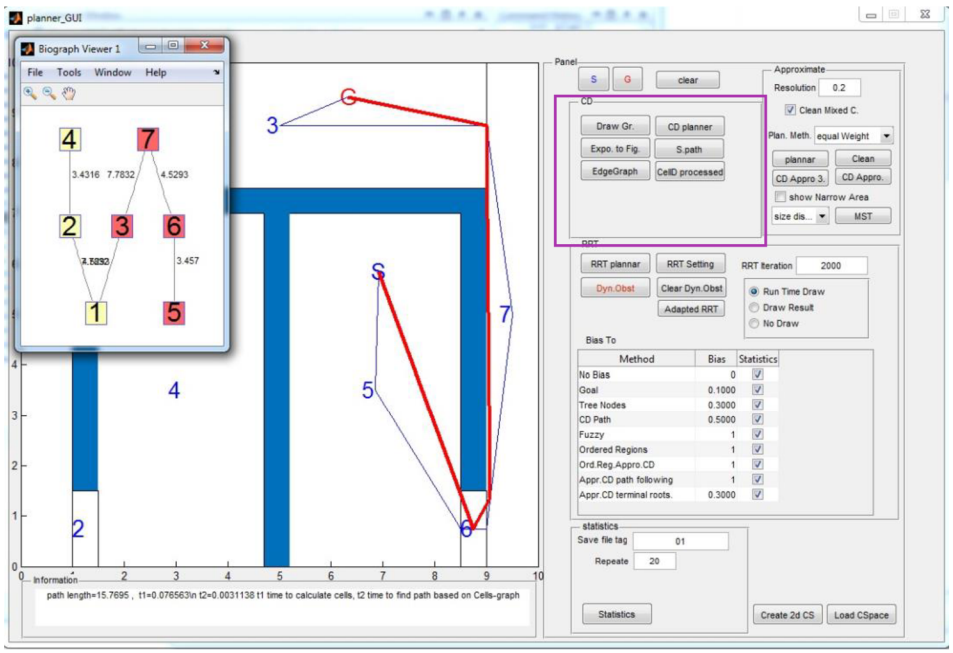


Figure A-10: Exact cell decomposition, and the generated graph

The exact cell decomposition options contain generating the cells, planning a path, shortening the path, and other visualization tool as draw the graph, as shown in Figure A-10.

The approximation cell decomposition algorithm is implemented in the way that the user can set the minimum resolution as shown in Figure A-11. In addition, the cost of graph edges can be set using four methods[1], i.e. equal translation cost, cost proportional to

[1] See chapter 3.3.1

cells size, cost proportional to translation between different cells size, and cost based on real distance between cell centers.

In Figure A-12, minimum spanning tree algorithm is implemented to identify the narrow passages using five methods[1] that set the graph edges' cost. We use "*graphMinSpanTree*" function from bioinformatics toolbox to find the required graph.

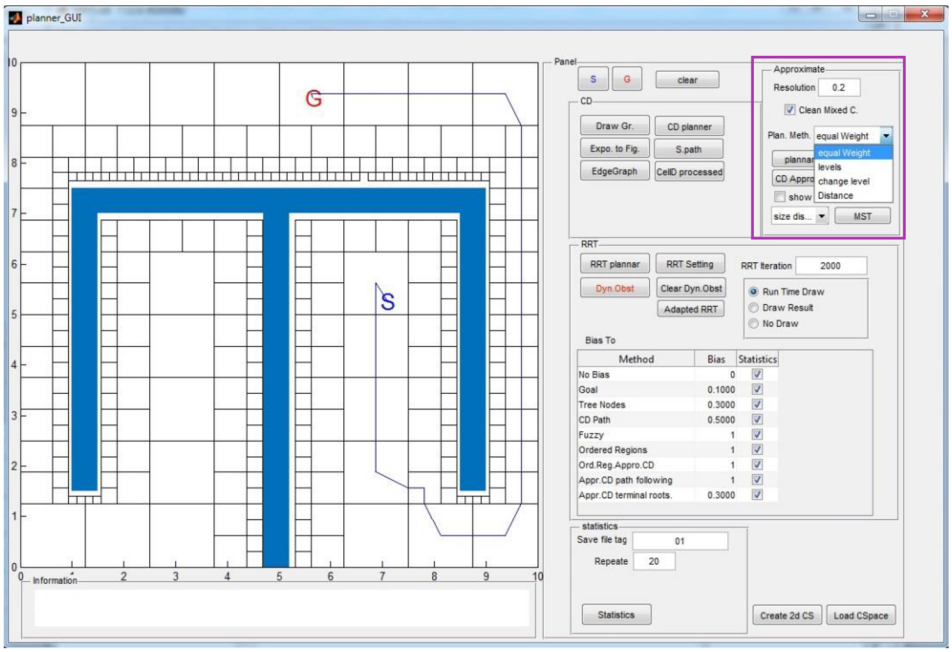


Figure A-11: Cell decomposition approximation and the planning options

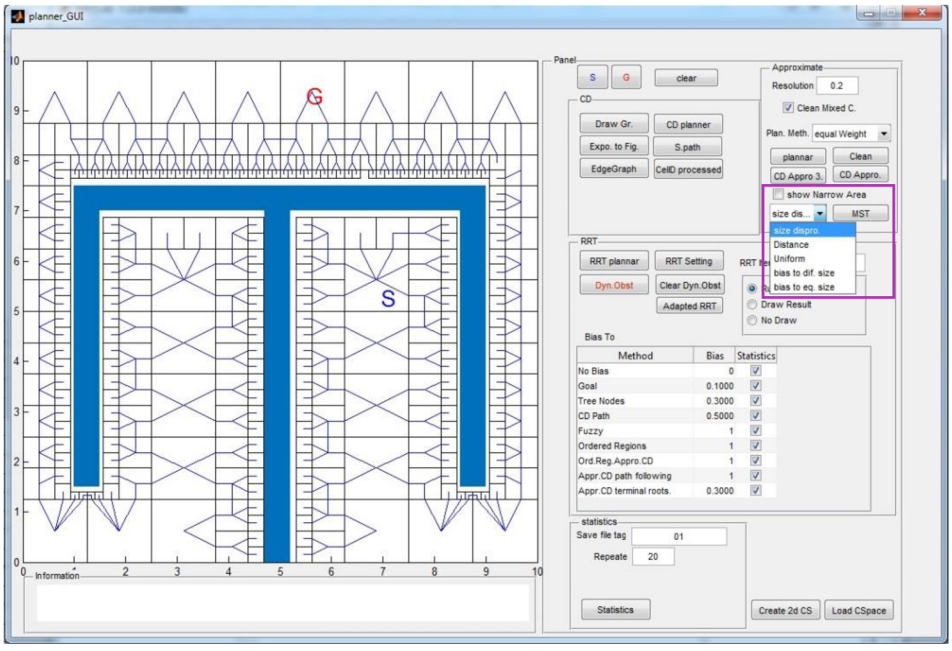


Figure A-12: Minimum spanning tree usage, over cell decomposition approximation's graph, and the narrow areas identification options

[1] See section 3.3.2