VĚDECKÉ SPISY VYSOKÉHO UČENÍ TECHNICKÉHO V BRNĚ

Edice PhD Thesis, sv. 802

ISSN 1213-4198

Ing. Ahmad Abbadi

Expert Systems and Advanced Algorithms in Mobile Robots Path Planning

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

Ing. Ahmad Abbadi

EXPERT SYSTEMS AND ADVANCED ALGORITHMS IN MOBILE ROBOTS PATH PLANNING

EXPERTNÍ SYSTÉMY A POKROČILÉ ALGORITMY V OBLASTI PLÁNOVÁNÍ CEST MOBILNÍCH ROBOTŮ

Zkrácená verze Ph.D. Thesis

Obor:	Konstrukční a procesní inženýrství
Vedoucí práce:	doc. Ing. Radomil Matoušek, Ph.D.
Oponenti:	prof. RNDr. Ing. Miloš Šeda, Ph.D. doc. RNDr. PaedDr. Eva Volná, Ph.D.
Datum obhajoby:	11. března 2016

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

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

Místo uložení:
Vysoké učení technické v Brně
Fakulta strojního inženýrství
Technická 2896/2
616 69 Brno

ISBN 978-80-214-5339-5
ISSN 1213-4198

Table of contents

1 INTRODUCTION

The focus in this thesis is to develop 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) and its developments are reviewed, 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 probabilistic 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. 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 from 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.

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 the state of the art.
- Design new approaches for path planning based on RRT and cell decomposition.
- 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.

2 STATE OF THE ART

In this section, the used motion planning concepts are reviewed concisely. We start with basic concepts of cell decomposition approaches, then the rapidly-exploring random tree principle is described, and in the final part, a summary of expert system and fuzzy rule-based is presented.

2.1 CELL DECOMPOSITION

Cell decomposition algorithms (CD) extracts the obstacles and the free regions, and build a graph of adjacency for the free ones [1, Ch. 6], [2]. 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 [3].

The first category uses geometric algorithms to determine the free areas and build free cells explicitly [4], [5] . 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., [6], [7].

a b

Figure 2-1: Path planning using trapezoidal cell decomposition. a: The generated free cells, b: The graph of adjacency which corresponding to paths between cells, the shaded boxes represent the path between initial and goal cells

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.

2.2 RAPIDLY-EXPLORING RANDOM TREE (RRT)

Rapidly-exploring random tree is a probabilistic algorithm introduced in [8]. The algorithm builds a space-filling tree that is constructed incrementally using samples drawn randomly from the search space, as shown in Figure 2-2. 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. 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.

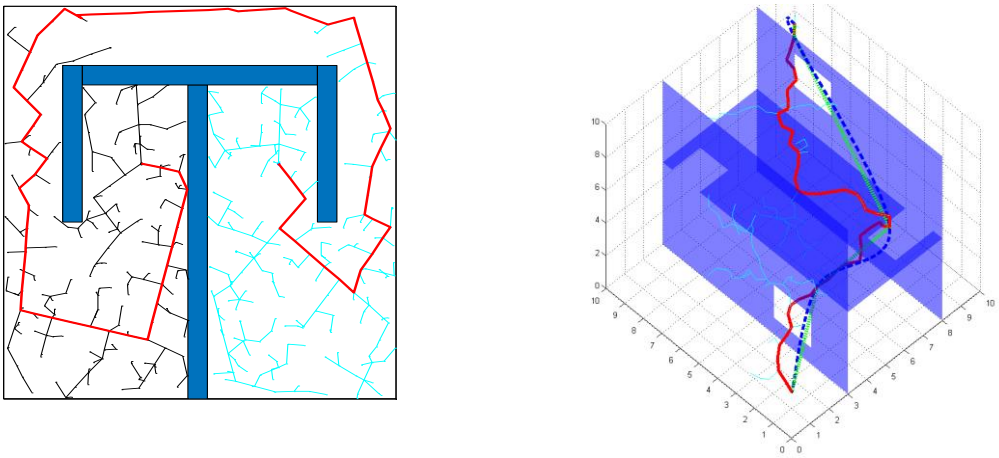


Figure 2-2: RRT expansion in 2D and 3D workspace

RRT algorithm proofed to be probabilistically complete [9], and resolution complete [10].

The algorithm, 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. 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 2-3. 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. 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 increamental 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.

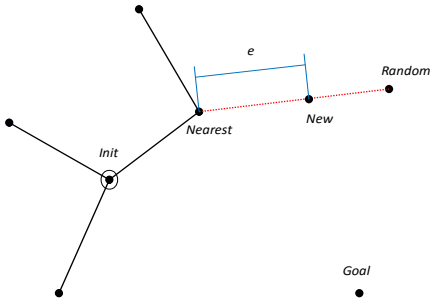


Figure 2-3: RRT algorithm principle

2.3 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 [11], [12].

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 system contains two main components, the knowledge base module, and the inference engine module. Other parts can be added based on the application [13]. For example, user interface, knowledge engineer interface, and explanation facilities, etc.

The knowledge is a theoretical or practical understanding of a specific area. And the knowledge base in expert system contains the “domain-specific knowledge”, which is required 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. In if-then rules case, the rule consists of two parts: the “IF” part, called the condition, which is evaluated based on what is currently known about the problem; and, the “THEN” part, which is called the action. For example,

IF *pathExecTime* **is** *High*, **and** *pathFailure* **is** *High* **THEN** *collideTendency* **is** *High*

The variables in this example have symbolic value. In this case, uncertainty in variables’ values, can be handled using fuzzy expert system.

The inference engine is another 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 [11]. 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.

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 [12].

2.3.1 Fuzzy Expert System

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.

Many attempts introduced to improve the robotic motion planner using the previous experience [14]–[20]. 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 [21]–[28].

3 CONTRIBUTIONS, TESTS AND RESULTS

In this section, the contributions to the motion-planning problem using RRT algorithm, CD, and the fuzzy expert system are presented. We re-implement the algorithms to fit the applications of omnidirectional mobile robot. This section is divided into sub-chapters. In the first two ones, 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 path. In the third and fourth parts, new methods to enhance the RRT navigation in small and narrow areas are presented. Then the fifth part presents our proposal using the combination of RRT, CD, and the fuzzy expert system. The last two sections present the use of CD approaches on safe path planning, and the narrow passage identification.

3.1 RRTS REVIEW AND STATISTICAL ANALYSIS

In this work [29] statistical tests were done, to make a better decision for using a variant of RRT. This work is based on the previous results in [30], 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.

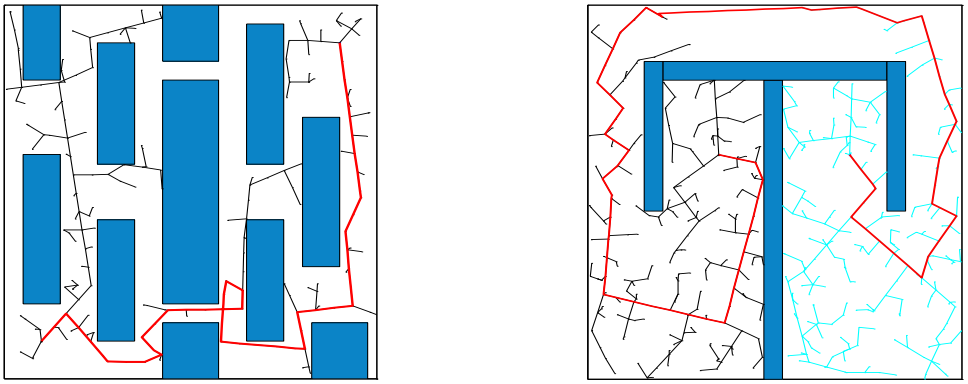
3.1.1 Test and Results

We made the tests for 13 RRT variations in four workspaces, i.e. low-density of obstacles, T-trap, high obstacles density, and the doors workspace. Figure 3-1 shows the high obstacle density and T-trap workspaces

The test is applied on 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. 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.

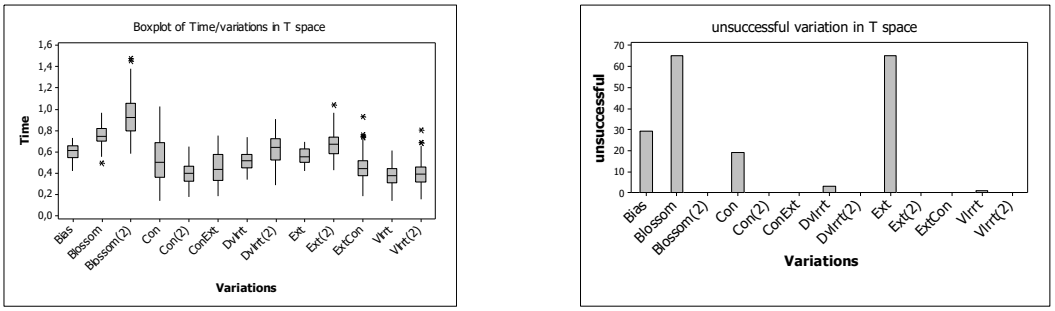
Execution Time, and probabilistic completeness results

The tests results show that the best variation in T obstacle workspace is Vlrrt, it 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 3-1. And Figure 3-2 shows the boxplot representation of the execution time results (a), and the completeness results (b).



(a) (b)

Figure 3-1: The testing workspaces, a: High obstacles density workspace, b: T-trap workspace



(a) (b)

Figure 3-2: T obstacle workspace tests, (a) Boxplots representation of the execution time results, (b) unsuccessful attempts to find a path

Table 3-1: 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

The tests show that unidirectional methods have more tendencies to fail, more than the bidirectional versions.

A statistical test was done on Vlrrt and Vlrrt(2). 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 probabilistic completeness, with the method that has a less completeness ratio; Figure 3-3 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%. The others performance and statistical tests, in addition to the graphical and numerical results are presented in the full version of this thesis.

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 3-3: T-test for the hypothesis "H_1: Vlrrt and Vlrrt(2) not equal" in T workspace

3.2 RAPIDLY-EXPLORING RANDOM TREES: 3D PLANNING

In this work [31] 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

We try to shorten the RRT path and make it as smooth as possible by removing useless points. We introduce an algorithm in [29], it generates a shortened path based on the original one. A new version is proposed in this work.

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

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.

Figure 3-4-a shows the original tortuous path that is generated by RRT, 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.

A smoothing-out technique is applied to the shortened path using Catmul-Rom spline [32], as shown in Figure 3-4-b.

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.

(a) (b)

Figure 3-4: (a) The shorten algorithms results, the (black - -) line represent the first shortened path, and the (blue - .) line represent the second shortened path, (b) the smoothed path (bold ...), the solid red line represents the original RRT path,

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 3-5 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.

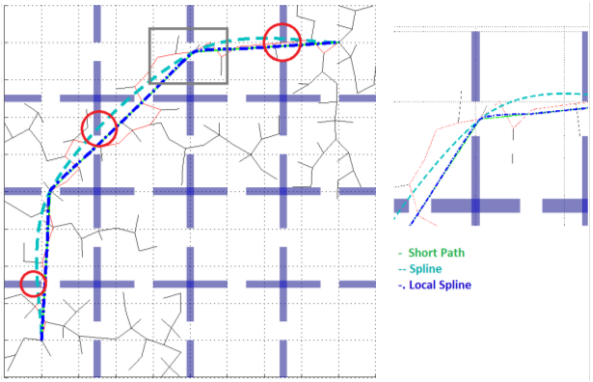
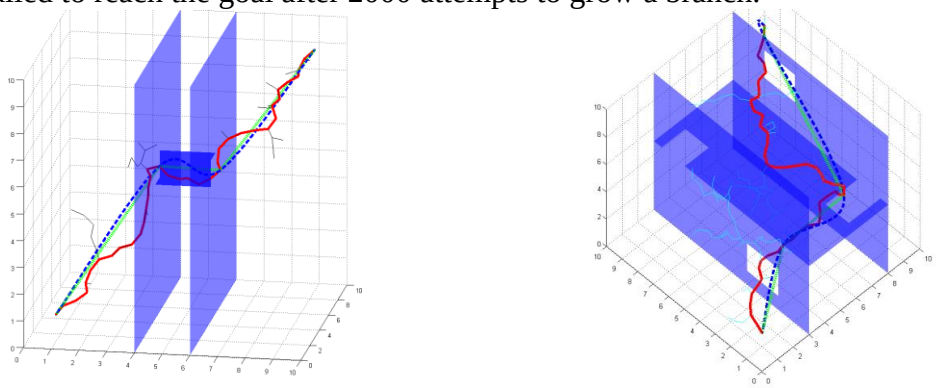


Figure 3-5: Local spline. The dashed line represents the spline path, the (.-) line represents the local-spline path

Testing Environments

We have constructed four testing scenarios for simulating the RRT performance in 3D workspaces. The first one involves a wall has a passage as shown in Figure 3-6-a. This obstacle evaluates the RRT efficiency in simple narrow passage scenario. The second workspace involves two walls where each one has a window. The third workspace has three walls with windows in different. The last scenario has vertical and horizontal obstacles with different locations of the windows as shown in Figure 3-6-b.

We have tested six variations of RRT, i.e. the basic RRT (*Ext*), *Blossom*, *Vlrrt* and the bidirectional versions of them. The tests were executed for every method 100 times per scenario. The algorithms have been implemented in Matlab environment. We consider the RRT failed to reach the goal after 2000 attempts to grow a branch.

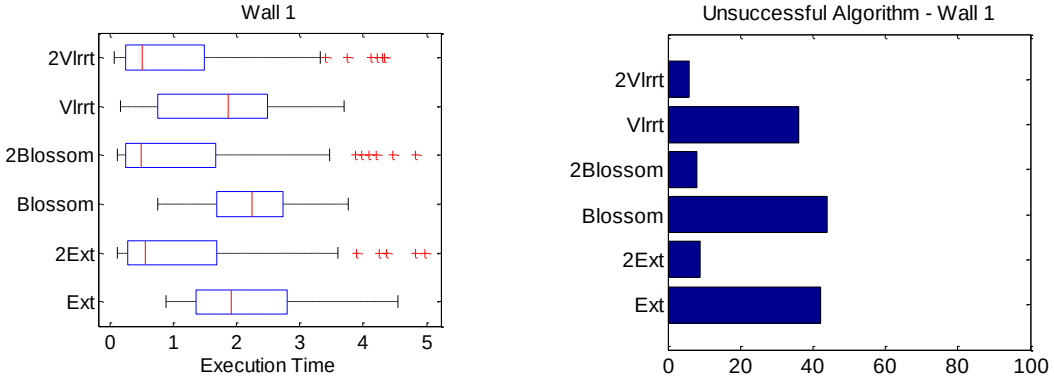


(a) (b)

Figure 3-6: (a) Narrow passage scenario (Wall 1), (b) horizontal and vertical workspace (Wall 4)

The numerical results represent the average of execution time for successful tries to find a path. The boxplot representation is shown in Figure 3-7-a for the narrow passage scenario.

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. 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 in Figure 3-7-b.



(a) (b)

Figure 3-7: Wall 1, (a) boxplot for RRT variations based on an average time of executing, (b) the unsuccessful attempts to find a path.

3.3 SPATIAL GUIDANCE TO RRT PLANNER USING THE CELL-DECOMPOSITION ALGORITHM

In this work [33], we made a comparison between the probabilistic path-planning method, i.e. RRT and the spatial planner, i.e. exact cells-decomposition algorithm (*CD*).

A new method is tested 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.

3.3.1 Problem formulation and proposed solution

The RRTs as example of randomized algorithms, has a good performance in high dimensional workspace. In general, the limitation of these algorithms is the planning in small areas. In cell-decomposition case, it is efficient in low dimensions planning, even in small areas; 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.

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.

3.3.2 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.

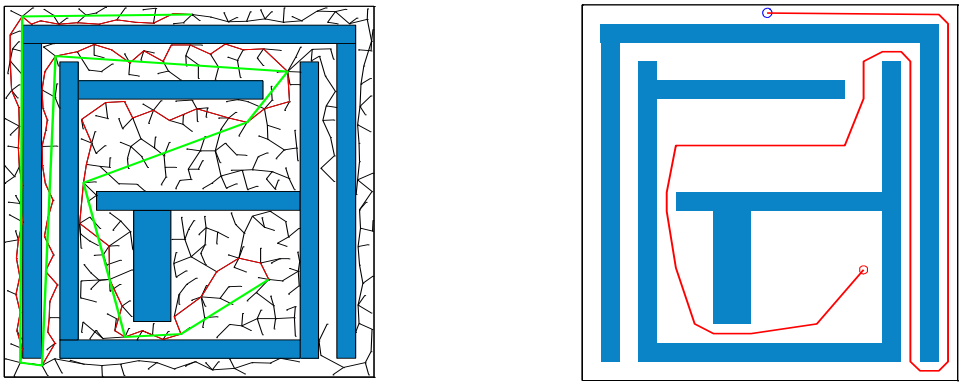
In the first workspace (building-like scenario), a simulation was lunched for a path planning, and the results are listed in Table 3-2. In CD case the nodes number represents the number of graph's nodes, which is constant. While in RRTs case, the node numbers are taken as an average of the results in a 100 times of repeated tests.

Table 3-2: Test results in 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(3000)	478.17	0	1.85(331.03)	1.85 (5.77)	43.46	2.0520	12%
RRTs(5000)	547.17	0	2.46(493.17)	2.46(7.67)	41.77	3.2539	90%
RRTs(10000)	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 3-8 shows the testing workspace, which consists of rooms and corridors. The first part Figure 3-8-a shows the solution founded by RRTs planner. While Figure 3-8-b uses the CD planner to find a path. 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 possible paths. The CD's path points are considered as bias points to the RRT's tree as shown in Figure 3-9, where the dots represent these points.



(a) (b)

Figure 3-8: Path planning in building-like workspace using (a) RRT, (b) CD algorithm

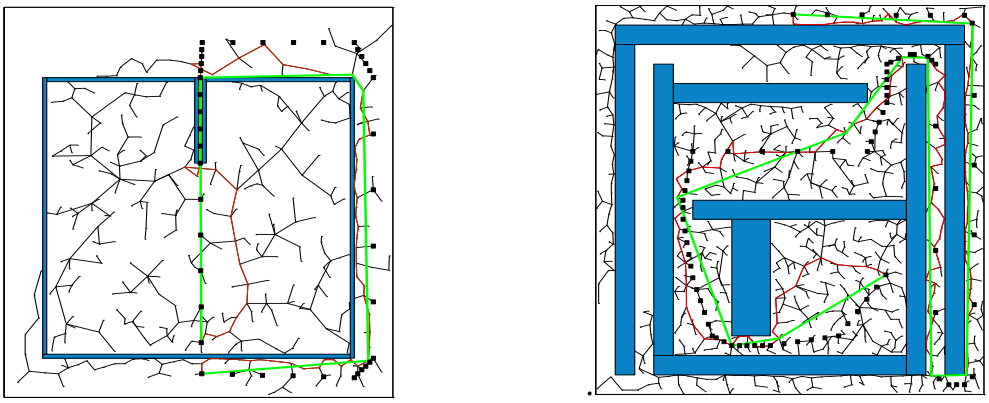


Figure 3-9: Path planning using RRTs with a bias toward CD path points, the dots represent points on CD path

Table 3-3: Test results in the building-like workspace, the bias to CD-path's points set 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%

The RRT biases toward path's points in probability of (0.2). The results are listed in Table 3-3, for the building-like workspace. The result shows that the bias enhances the RRT algorithm's completeness significantly in all cases, in comparison with the previous results. The complete tests and results are presented in the full version of the thesis.

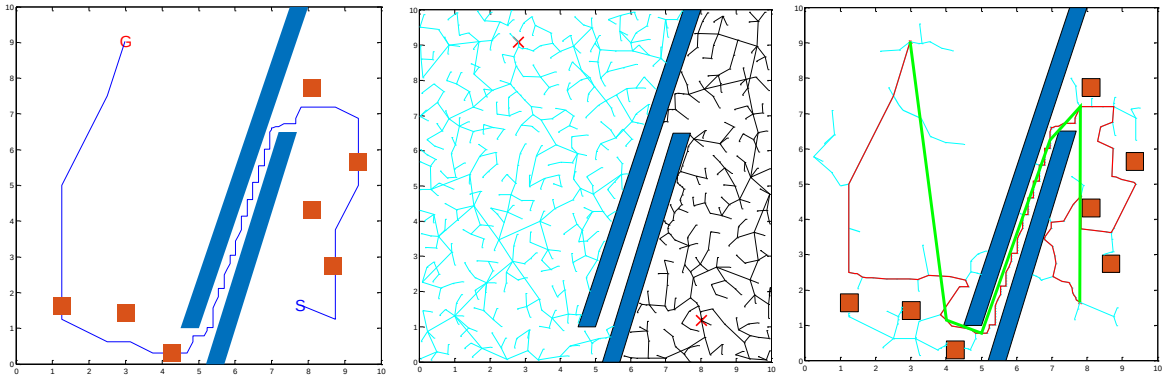
This work was a first step to build a hybrid planner, which works efficiently in continuous, high dimension workspace using the available knowledge and spatial information, and overcome the drawback of randomized sampling-base algorithms. The future work will focus on using available information to speed up the complex motion planning for robots in uncertainty and dynamic environments.

3.4 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 [34]. 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.

3.4.1 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.



(a) (b) (c)

Figure 3-10: (a) The drawback of ACD in dynamic environments, (b) the drawback of RRT in narrow passage and small regions, (c) the generated path using the combination of ACD and RRT

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 3-10-a. However, the RRT is not efficient in small areas and narrow passage as shown in Figure 3-10-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, as shown in Figure 3-10-c.

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.

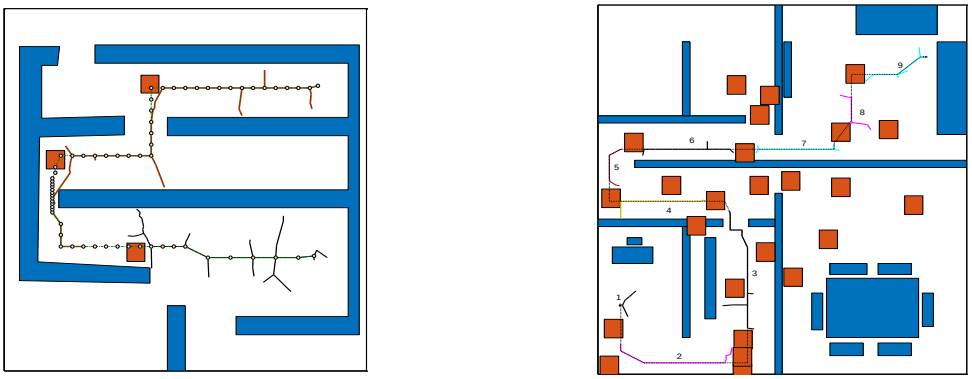
Two approaches have been proposed to benefit from this combination. These planners focus on the enhancement of navigation problem for omnidirectional robots in partially dynamic workspace.

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's 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.



(a) (b)

Figure 3-11: 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

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 3-11-a, where they try to follow the ACD' path (the dotted line).

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 3-11-b shows the local RRTs planners method in simulation. In this example, it creates nine local RRT's trees based on the original path, which is generated in the stationary workspace.

3.4.2 Tests and Results

The proposed approaches are tested in two different workspaces. 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.

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, GoalBias RRT, and the bias toward the other trees.

Testing Parameters

The bias values, which are given to the compared methods are set as follows, 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. In our proposed methods, the bias value of the validator RRTs is set to 100% when no collision occurred. Else, in the collision case, the bias value 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%.

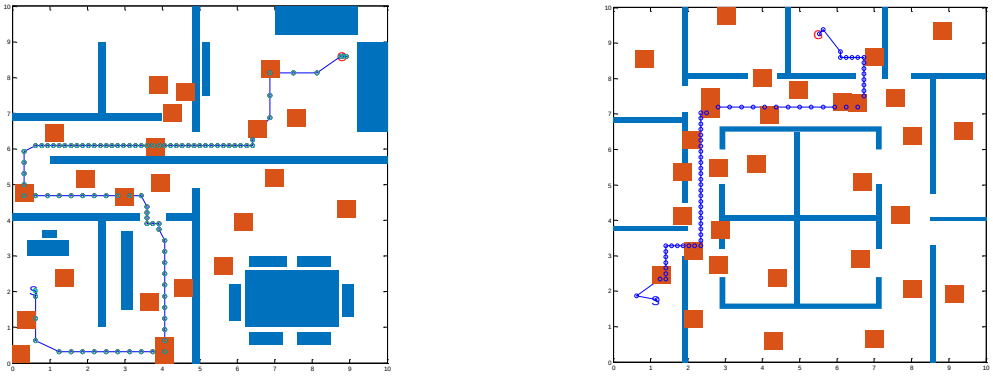
The simulation repeated 100 times and the average of the successful attempts are taken for results comparison. The results include the execution time, the number of RRT iterations, which is corresponding to the number of RRT branching attempts, and the number of successful attempts to find a path.

The probabilistic 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's path is constructed using the initial and the goal points, the free cells' centers, and the barriers' midpoint between the consequence cells. The RRT planner considered as failed if it cannot find a path after 2000 tries of branching.

Results and Discussions

In the first workspace, new obstacles are scattered in the original workspace. They are positioned to collided with the ACD path and add more difficulty to navigation process through the changed workspace. The workspace changes are shown in Figure 3-12-a, 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. The numerical results are shown in Table 3-4, where the proposed methods show more probabilistic completeness than the other methods do.

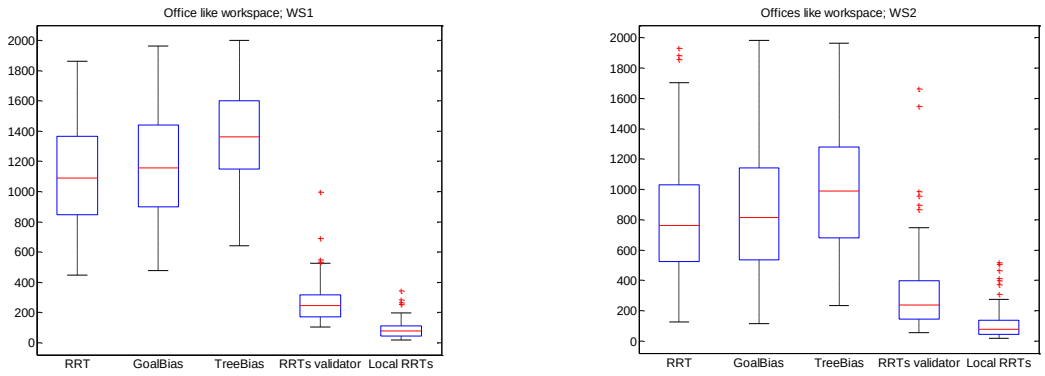


(a) (b)

Figure 3-12: (a) New obstacles in office-like workspace WS1, (b) new obstacles in offices-like workspace WS2. ACD path is represented by a solid line, and the bias points represented as cycle markers

In the second workspace, the partially changes are introduced by scattering new obstacles in the stationary workspace. The obstacles collided with ACD path and produce more narrow passages. Figure 3-12-b, shows the changes in the workspace, where the obstacles are represented by the boxes. The numerical results are presented in Table 3-5. 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's trees method gives the best results in terms of efficiency; it has the lowest execution time, and the lowest iteration average. Figure 3-13-b condenses the iteration results for WS2 using the boxplots.

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 3-13-a sums up the iteration results for the first workspace WS1 using the boxplot representation.



(a) (b)

Figure 3-13: RRT iteration boxplot for WS1 (a) and WS2 (b)

Table 3-4: 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**

Table 3-5: 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**

3.5 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.

3.5.1 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. 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*.

Based on these values, the regions' *weight* are updated. Figure 3-15 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 3-14 shows the RRTs' growth and the regions classifications. In explored areas, the algorithm blocks RRTs' 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.

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.

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 3-16 shows the rules-based for this fuzzy planner.

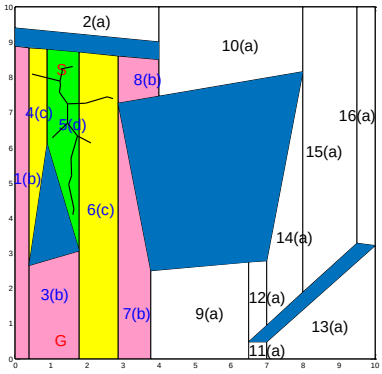


Figure 3-14: RRT growth and rules-based classification of the free regions; a: *far* regions. b: *neighbor* regions, c: *boundary* regions, d: *expanded* regions, *S* represents the initial position, *G* represents the goal position, and the blue regions represent the obstacles

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 3-15: The adviser's rules of "bias toward boundaries" planner

Adviser's rules in planner 2					
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 3-16: The adviser's rules of fuzzy bias planner

3.5.2 Simulations and Results

The tests are made in four workspaces to simulate the holonomic robots movements in offices and cluttered or crowded areas. 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 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's 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 bias value of every method is shown in Table 3-6. 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 3-6: Bias values in the testing methods

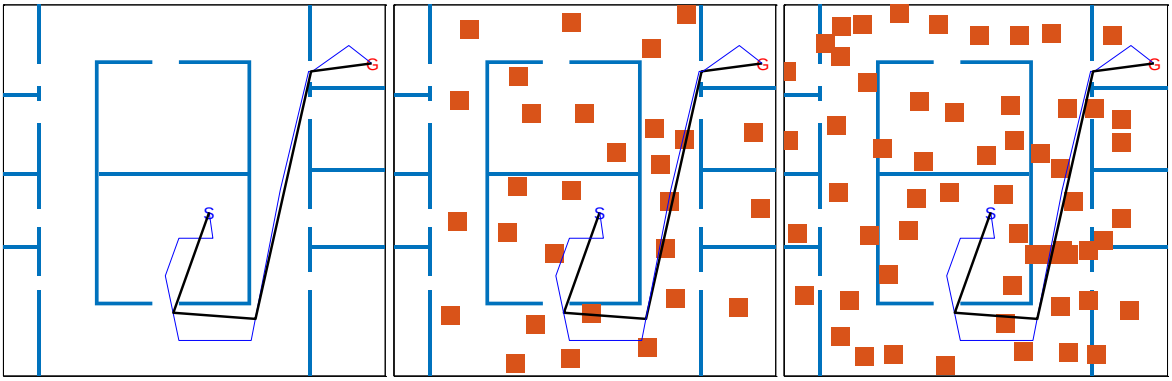
Goal	Other Trees	CD path	Fuzzy	Boundaries
0.1	0.3	0.5	1	1

Results

In the first scenario, the path-planning problem in the "WS1" workspace is simulated. Figure 3-17 shows the original workspace, 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 probabilistic completeness results are presented in Table 3-7, while, the iterations values are shown in Table 3-8. 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.

In the second scenario, the highly cluttered obstacles situation is simulated, where the robot should pass through very small regions. While in the third scenario, we simulate the situation where some paths are closed and the robot should find an alternative route and avoid the cluttered obstacles. In the fourth scenario, the narrow passage and narrow area problems are simulated. The robot should pass through narrow and long corridors, which contains cluttered obstacles, and narrow connection between free regions. The graphical and the numerical results are present in the full version of this thesis.



(a) (b) (c)

Figure 3-17: (a) The basic workspace WS1, (b) the minor changes in WS1, (c) the major changes in WS1. The thin line is a CD path, and the bold line is the shortened path. *G* and *S* represent the goal and the initial locations

Table 3-7: Number of successful attempts to reach the goal in WS1 workspace

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

Table 3-8: The average of RRTs branching attempts to reach the goal in WS1 workspace

	No bias	Goal bias	Other Trees bias	CD path bias	Fuzzy bias	Boundaries bias
Without change	439	470	461	208	79	77
Minor change	693	780	821	647	397	428
Major change	1253	1302	1116	1404	590	669

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 probabilistic 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.

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.

3.6 SAFE PATH PLANNING USING CELL DECOMPOSITION APPROXIMATION

In this work [35], 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 edges 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.

3.6.1 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.

3.6.2 Result and Discussion

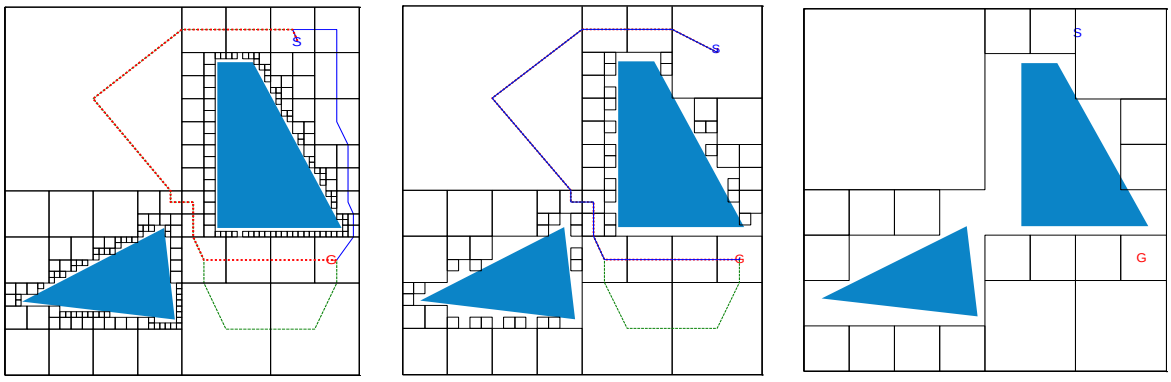
In the first proposed method, the weights of the graph's edges are uniformed to the cost of (1) unit, which corresponding to the cost of translating from one cell to another one, regardless of the cells' size.

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).

Dijkstra's algorithm is used as a graph search algorithm to find the path over the graph. 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 for the first work space are shown in Figure 3-18.



a b c

Figure 3-18: Safe paths planning in WS1, the safety distance is set to be a: (0.1), b: (0.3), c: (0.75). 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, the dashed-green line represents the disproportional penalty to 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 3-18-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. The results shows also that the algorithm does not find a path in the third test, the (0.75) due to the safety distance, where the collided cells is removed from the graph, and break the path between the initial position and the goal one. The other tests and results is presented in the full version of the thesis.

3.7 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 [36], 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.

3.7.1 Proposed methods

Narrow passage problem faces most of sampling based approaches. The problem occurs when a uniform distribution is used to take samples from 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.

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. 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

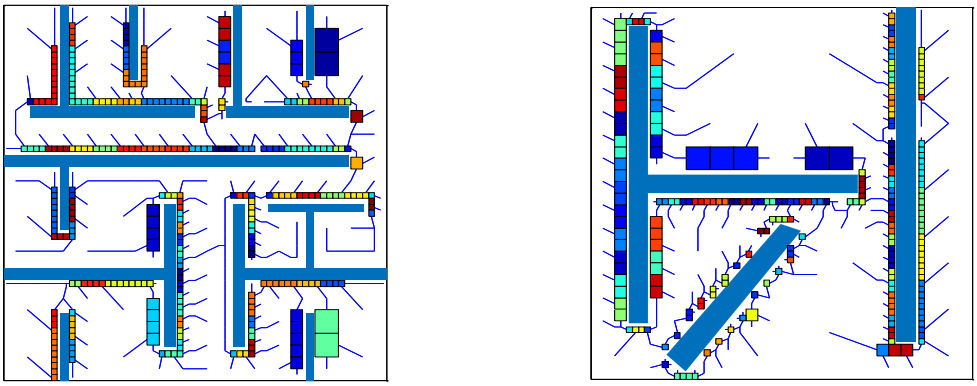
not satisfy the narrow passage condition “bigger-smaller-bigger,” instead it will has the pattern “bigger-smaller”. The MST in this case constructs narrow passage pattern just when it is necessary.

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.

3.7.2 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.



(a) (b)

Figure 3-19: Real distance cost, (a,c) show the identified narrow passages, each color represents one passage, the blue area represent the obstacles, this approach failed in detecting the correct passages.

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 3-19. Where (a,b) show the tested workspaces, and the identified narrow passages. Each passage is denoted using a color. As seen from the figures the extracted narrow passages using this methods is not accurate.

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

The fourth method, which gives lower cost to the translation between equal cells size as shown in Figure 3-20, generates better results, it has the ability to find all narrow passage. But, it generates very long sequences.

The last proposed method, which has disproportional cost to the distance, 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.

The graphical results of these methods are presented in the full version of this thesis.

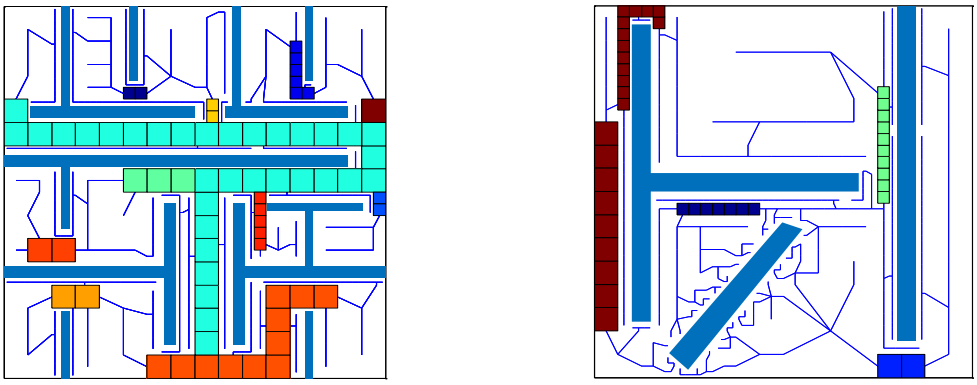


Figure 3-20: The identified narrow passages using the bias to the equal cells size method, each color represents one narrow passage, the blue area represent the obstacles

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.

4 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

[1] S. M. LaValle, *Planning algorithms*. Cambridge ; New York: Cambridge University Press, 2006.

[2] Milos Seda, “Roadmap methods vs. cell decomposition in robot motion planning,” in *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*, 2007, pp. 127–132.

[3] J.-C. Latombe, *Robot motion planning*. Boston: Kluwer Academic Publishers, 1991.

[4] R. A. Brooks and T. Lozano-Perez, “A subdivision algorithm in configuration space for findpath with rotation,” *IEEE Trans. Syst. Man Cybern.*, vol. SMC-15, no. 2, pp. 224–233, Mar. 1985.

[5] J. T. Schwartz and M. Sharir, “On the ‘piano movers’ problem. II. General techniques for computing topological properties of real algebraic manifolds,” *Adv. Appl. Math.*, vol. 4, no. 3, pp. 298–351, Sep. 1983.

[6] N. H. Sleumer and N. Tschichold-Gürman, *Exact Cell Decomposition of Arrangements used for Path Planning in Robotics*. 1999.

[7] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[8] S. M. Lavalle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” 1998.

[9] S. M. LaValle and J. J. Kuffner, “Randomized Kinodynamic Planning,” *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[10] P. Cheng and S. M. LaValle, “Resolution complete rapidly-exploring random trees,” in *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA ’02*, 2002, vol. 1, pp. 267–272 vol.1.

[11] M. Sasikumar, S. Ramani, S. M. Raman, K. S. R. Anjaneyulu, and R. Chandrasekar, *A Practical Introduction to Rule Based Expert Systems*. Narosa Publishing House, New Delhi, 2007.
[12] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems*, 2nd ed. Harlow, England ; New York: Addison-Wesley, 2005.
[13] P. S. Sajja and Rajendra Akerkar, Eds., *Advanced Knowledge-Based Systems: Models, Applications and Research*. 2010.
[14] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3671–3678.
[15] J. Lien and Y. Lu, "Planning Motion in Environments with Similar Obstacles," presented at the Robotics: Science and Systems V, Seattle, USA, 2009.
[16] S. R. Martin, S. E. Wright, and J. W. Sheppard, "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*, 2007, pp. 1131–1136.
[17] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for Rapid Replanning in Dynamic Environments," in *2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 1603–1609.
[18] M. Zucker, J. Kuffner, and J. A. Bagnell, "Adaptive workspace biasing for sampling-based planners," presented at the Robotics and Automation, 2008. ICRA 2008., Pasadena, CA, 2008.
[19] C. G. Atkeson and J. Morimoto, "Nonparametric Representation of Policies and Value Functions: A Trajectory-Based Approach," in *In NIPS 15*, 2003, pp. 1611–1618.
[20] M. Stolle and C. G. Atkeson, "Policies based on trajectory libraries," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, 2006, pp. 3344–3349.
[21] E. Aguirre and A. González, "Fuzzy behaviors for mobile robot navigation: design, coordination and fusion," *Int. J. Approx. Reason.*, vol. 25, no. 3, pp. 255–289, Nov. 2000.
[22] S. M. Sharef, W. K. Sa'id, and F. S. Khoshaba, "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, pp. 1–7.
[23] Petr Krcek and Jiří Dvorak, "MOBILE ROBOT MOTION CONTROL BY MEANS OF FUZZY RULES," presented at the Engineering Mechanics 2004, Svratka, 2004.
[24] M. B. Montaner and A. Ramirez-Serrano, "Fuzzy knowledge-based controller design for autonomous robot navigation," *Expert Syst. Appl.*, vol. 14, no. 1–2, pp. 179–186, Jan. 1998.
[25] D. Driankov and A. Saffiotti, Eds., *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, vol. 61. Heidelberg: Physica-Verlag HD, 2001.
[26] A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Comput. - Fusion Found. Methodol. Appl.*, vol. 1, no. 4, pp. 180–197, Dec. 1997.
[27] M. A. K. Jaradat, M. H. Garibeh, and E. A. Feilat, "Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field," *Soft Comput.*, vol. 16, no. 1, pp. 153–164, Jan. 2012.
[28] Y. Chen, L. Cheng, H. Wu, X. Zhao, and J. Han, "Knowledge-driven path planning for mobile robots: relative state tree," *Soft Comput.*, May 2014.
[29] A. Abbadi and R. Matousek, "RRTs Review and Statistical Analysis," *Int. J. Math. Comput. Simul.*, vol. 6, no. 1, 2012.
[30] A. Abbadi, R. Matousek, P. Krček, and P. Soustek, "RRTs Review and Options," in *computational Engineering in Systems Applications*, 2011, vol. 2, pp. 194–199.
[31] A. Abbadi, R. Matousek, S. Jancik, and J. Roupec, "Rapidly-exploring random trees: 3D planning," presented at the Mendel, 2012, pp. 594–599.
[32] E. Catmull and R. Rom, "A CLASS OF LOCAL INTERPOLATING SPLINES," in *Computer Aided Geometric Design*, Elsevier, 1974, pp. 317–326.
[33] 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.
[34] A. Abbadi and V. Prenosil, "Collided path replanning in dynamic environments using RRT and Cell decomposition algorithms," presented at the Modelling & Simulation for Autonomous Systems Workshop, Prague, 2015.
[35] 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.
[36] 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.

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.

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.