Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Master's Thesis

Robot motion planning in dynamic environment: A comparative study

Dharmin Bakaraniya

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Erwin Prassler
Dr. Cesar Lopez Martinez

January 2019

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date

Dharmin Bakaraniya

Abstract

We compare few of the state of the art approaches of motion planning in dynamic environment for planar mobile robots. Motion planning in static environment is a solved problem, whereas motion planning in dynamic environment is a very complex problem. Finding out the best approach and best motion planner for an application would save a lot of time and money for that industry. The applications for mobile robots would not be limited to static environment. Using a better motion planner will provide more safety for humans and for robots and will provide cost effective transportation of goods, and even people!

Existing comparisons for motion planning in dynamic environments are either out of date or they do not compare the approaches with same parameters. The research papers proposing different approaches do not have any benchmark comparison on their implementations. They test their approach with different assumptions in different environment on robots with different constraints to optimise different parameters. This eliminates the possibility of finding a better motion planner without implementation. We compare few state of the art approaches and motion planners on the basis of kinematic constraints on robot, shape and motion contraints of moving obstacles and experimental results of their implementation.

From an ideal motion planner, it is expected that plans a trajectory considering kinematic and dynamic constraints, robot geometry and change in surrounding environment. This trajectory is expected to result in the robot reaching at the goal location in minimum amount of time without any collisions. Therefore, we use travel time, number of replan attempts and number of collisions as performance measure to compare motion planner.

We create 3 test cases with increasing complexity to test 5 motion planners and compare their performance. The motion planners are tested on a rectangular holonomic robot in a simulated environment. We find out which motion planner is best out of the 5 planners for the designed test scenarios.

Contents

1 Introduction **1**
  1.1 Motivation . . . 1
  1.2 Challenges and Difficulties . . . 2
  1.3 Problem Statement . . . 2

2 State of the Art **5**
  2.1 Related work . . . 5
  2.2 Motion planning approaches . . . 6
    2.2.1 Velocity based . . . 6
    2.2.2 ICS based . . . 9
    2.2.3 Roadmap based . . . 11
    2.2.4 Elastic Band . . . 12
    2.2.5 Spline based . . . 13
    2.2.6 Other approach . . . 14
  2.3 Comparison . . . 15

3 Methodology **19**
  3.1 Experimental Design . . . 19
  3.2 Setup . . . 20
  3.3 Test case 1: Single obstacle single room . . . 21
  3.4 Test case 2: Two obstacle single room . . . 21
  3.5 Test case 3: Double room . . . 22

4 Implementation **25**
  4.1 Moving obstacles . . . 25
  4.2 Obstacle trajectory to costmap . . . 25
  4.3 Planners to be evaluated . . . 26

4.3.1 TEB local planner 26
4.3.2 Spline based planner 26
4.3.3 DWA local planner 27
4.3.4 EBand local planner 27
4.3.5 EBand with obs_to_costmap 27

5 Evaluation 29
5.1 Test case 1 30
5.1.1 TEB local planner 31
5.1.2 Spline based planner 32
5.1.3 DWA planner 33
5.1.4 EBand planner 34
5.1.5 EBand2 planner 34
5.2 Test case 2 35
5.2.1 TEB local planner 36
5.2.2 Spline based planner 37
5.2.3 DWA planner 38
5.2.4 EBand planner 38
5.2.5 EBand2 planner 39
5.3 Test case 3 40
5.3.1 TEB local planner 40
5.3.2 Spline based planner 42
5.3.3 DWA planner 42
5.3.4 EBand planner 43
5.3.5 EBand2 planner 43

6 Results 45
6.1 Time comparison 45
6.2 Re-plan comparison 46
6.3 Collisions comparison 47
6.4 Overall comparison 48

7 Conclusions 49
7.1 Contributions 49

7.2 Lessons learned 49
7.3 Future work 50

Appendix A Parameters **51**
A.1 Kinodynamic parameters 51
A.2 Goal tolerance parameters 51
A.3 Machine specifications 52
A.4 Costmap parameters 52

References **55**

List of Figures

3.1 Test case 1 obstacle trajectory (green arrow is goal position) 21
3.2 Test case 2 obstacle trajectories (green arrow is goal position) . . . 22
3.3 Test case 3 velocity trajectories (green arrow is goal position) 23
3.4 Velocity vs. time plot for moving obstacles 24

5.1 Moving obstacle position and direction (black arrow) for experiments in test case 1. (Green arrow is goal position and orientation) 30
5.2 teb local planner navigating in test case 1 (experiment 1) (Line in green is global path, red arrows are local planner's path, white square is the obstacle and green arrow is the position of goal) . . . 31
5.3 Spline based planner navigating in test case 1 (experiment 1) (red line is motion planner's path, red square is moving obstacle and green arrow is goal position) 32
5.4 DWA planner navigating in test case 1 (experiment 1) (Red squares are obstacle and the fake obstacles created by `obs_to_costmap`, green line is global path, red line is local path generated by DWA and green arrow is goal position) 33
5.5 Moving obstacle positions and directions (black arrow) for experiments in test case 2. (Green arrow is goal position and orientation) 35
5.6 teb local planner navigating in test case 2 (experiment 4) (Line in green is global path, red arrows are local planner's path, green arrow is goal position and white square is the obstacle) 36
5.7 Spline based planner navigating in test case 2 (experiment 4) (red line is planner path, blue arrow is robot, red squares are obstacle positions and green arrow is goal position) 37

5.8 Moving obstacle positions and directions (black arrow) for experiments in test case 2. (Green arrow is goal position and orientation) 40
5.9 teb local planner navigating in test case 3 (experiment 2) (Line in green is global path, red arrows are local planner's path and red square is the obstacle) 41

List of Tables

2.1 Comparison of some of the state of the art approaches for motion planning in dynamic environment holonomic+: can be applied theoretically to any vehicle type cv: constant velocity, vv: varying velocity cd: constant direction, vd: varying direction sim: simulated experiments, real: experiments performed on an actual robot (Note: The experiments might have relaxed some assumptions taken for theoretical approach. The *Obstacle restriction* and *Obstacle shape* information is based on the information provided from experimental setup. If a literature provides multiple experiments with different relaxed assumption, then we consider the assumption which is most resembles the theory.) 16

5.1 Performance of teb local planner navigating in test case 1 31
5.2 Performance of spline based planner navigating in test case 1 32
5.3 Performance of DWA planner navigating in test case 1 33
5.4 Performance of EBand planner navigating in test case 1 34
5.5 Performance of EBand2 planner navigating in test case 1 34
5.6 Performance of teb local planner navigating in test case 2 36
5.7 Performance of spline based planner navigating in test case 2 37
5.8 Performance of DWA planner navigating in test case 2 38
5.9 Performance of EBand planner navigating in test case 2 38
5.10 Performance of EBand2 planner navigating in test case 2 39
5.11 Performance of teb local planner navigating in test case 3 40
5.12 Performance of spline based planner navigating in test case 3 42
5.13 Performance of DWA planner navigating in test case 1 42
5.14 Performance of EBand planner navigating in test case 3 43

5.15 Performance of EBand2 planner navigating in test case 3 43

6.1 Average time of travel for planners for different test cases 45
6.2 Maximum re-plan attempts of planners for different test cases . . . 47
6.3 Average collisions of planners for different test cases 47

A.1 kinodynamic parameters 51
A.2 Goal tolerance parameters 51
A.3 Machine specifications 52
A.4 costmap common parameters 52
A.5 Costmap global parameters 52
A.6 Costmap local parameters 53

1

Introduction

Motion planning in dynamic environment for planar mobile robots has a lot more application compared to one in static environment as the robot can be deployed in real environment among humans and other robots. The robot does not need to be in its own special and controlled environment. The robot can be used in day to day life of an average person (for example, automated taxi or public transport). A mobile robot which can navigate in dynamic environment is extremely useful for industries to automate their production line without changing their whole infrastructure. This need is increasing with the availability of cheap and easily accessible robots.

1.1 Motivation

By comparing different approaches of motion planning in dynamic environment on same constraints and parameters, a well informed decision regarding the best approach for a given application is achieved. By finding out the current best approach and its limitations, we can also start focusing on solving those limitations rather than finding solutions for solved problems. By solving problem of motion planning in dynamic environment, we can ensure

- Safe environment for humans and for robots
- Cost effective transportation of goods (and people[1])

1.2 Challenges and Difficulties

Robot motion planning can be considered solved for static environment[2]. However, the robot does not have perfect information about the surrounding environment in a dynamic environment. The task to calculate a motion control for such situation is considered a NP-hard problem[3].

Simple put, motion planner needs to generate a plan that enables the robot to reach the desired target position while avoiding static and moving obstacle in minimum time. Additionally, the planner should satisfy kinematic and dynamic constraints of the robot. The planner needs to be fast enough that the robot can react to its environment in real time. It is desirable that these tasks are performed in *efficient* manner. By efficient, we mean the planner does not require lot of sensor data or lots of computational resources.

The state of the art approaches test their methods and algorithm with different parameters such as

- kinematics of robot (holonomic, differential drive or car-type)
- availability of map
- shape of moving obstacles (circular, convex polygon, etc)
- kind of motion of moving obstacles (same direction with constant velocity, varying direction with constant velocity, varying direction with varying velocity)
- accuracy of localisation of robot
- amount of information available to robot about the obstacles (shape, position and velocity)

This makes the task of determining the better approaches from the rest difficult and the result may be unreliable.

1.3 Problem Statement

To compare motion planners and their approaches as a whole, a literature comparison is necessary. With this initial comparison, many of the approaches could be eliminated because of one or more of the following reasons

- The approach or planner does not work with certain parameter that is desired. For example, if an approach is only applicable to differential drive but we want the motion planner to navigate a car. Another example would be that an approach needs extreme amount of computation to navigate but we want use the planner on a cost effective and small mobile robot.
- The approach is highly criticised by peer reviewers

After eliminating all the approaches which are not applicable, we still have a number of approaches which are tested on different robot under different assumptions (as mentioned in 1.2). A comparative evaluation of the implementation of these approaches under specific assumption is needed to find out the best approach. To compare the performance of the tested algorithm, we choose travel time, number of re-plan attempts and number of collisions as measurements for a better motion planner for dynamic environment.

2

State of the Art

2.1 Related work

Mohanan et al.[4] covers 101 research papers that were published between 1985 and 2016 in the field of motion planning in dynamic environment. This survey is very close to the current work in terms of publication date. They have considered most of the work in this field and presented it in different sections on the basis of school of thought. Even though this work covers many approaches, the comparison is lacking. The only comparison is within a school of thought and that too is quite shallow. Additionally, the authors do not provide a valid reasoning behind any of their claims and neither does their work contain any experimental study.

Hoy et al.[5] provides a survey for algorithms which provide collision free navigation for robots. This work is fairly detailed. However, this work cannot be considered very close to our work because it mainly covers obstacle avoidance comparison, but it does contain a section for dynamic obstacle avoidance. They have given quite a lot of importance of collision avoidance using boundary following. Even though it provides a comparison between main approaches based on numerous criterion, it still does not evaluate these approaches on same parameter. They have provided their comparison solely based on the information provided by the original research papers.

Keshmiri et al.[3] provides a survey specifically for motion planning in dynamic environment. It covers all the approaches presented in research papers published between 1986 and 2008 totalling up to 150 papers. They have provided a comparison on how much contribution has been made in motion planning field based on different approach but regarding the actual approaches itself, only a summary of at most 23 sentences for each approach is provided. The authors have not performed any test on the approaches. This work was meant for just providing a survey for contribution in different school of thought.

Surveys [6] and [7] are quite dated and does not cover many state of the art approaches in motion planning for dynamic environment.

Existing approaches generally provide critique and deficiencies on their previous works. These are generally helpful for the reader at the time of publishing but they can only provide a critique on the approaches that were published before them. Additionally, their critique are generally about the problem that they have solved in their work. Therefore, though these comparisons are helpful at that time for the reader to better understand the presented work, they might not be completely useful for an unbiased comparison.

2.2 Motion planning approaches

2.2.1 Velocity based

Dynamic window

Fox et al.[8] proposed the original idea which was simply optimizing a function which balances robot's distance from goal (*heading*), distance from nearest obstacle (*dist*) and current velocity (*vel*).

$$G(v,w) = \sigma(\alpha \cdot heading(v,w) + \beta \cdot dist(v,w) + \gamma \cdot vel(v,w))$$

Here α, β, γ and σ are constant that can be tuned according to need. This is a reactive approach, which means that the algorithms only thinks about the next iteration. It takes care of the kinematic and dynamic constraints by converting the current sensor readings in velocity space. Velocity space is a 2D space where X axis is angular velocity of the robot and Y axis is linear velocity. The robot is represented as a dot in this velocity space. The controls are bounded by the rectangle around this dot which represents the dynamic constraints of the robot. This approach, despite being robust, simple and fast did not work for dynamic environment. This method also had local minima problem.

Later, Brock et al.[9] extended this approach for global path planning and for dynamic environments by combining it with NF1 algorithm. This eradicated the problem of local minima. The experiments did not contain dynamic objects. This approach is later extended by [10] where they provide a version of DWA which is convergent but they only consider static environment.

The current best extension of dynamic window approach (time varying dynamic window (TVDW)) was proposed by [11]. This approach provides very robust obstacle avoidance method for dynamic environment. The authors have combined FD* path planning algorithm with dynamic window approach. This method uses moving cells for representing obstacles. This relieves the method from considering the shape of the obstacle. The obstacles are represented in the grid map at the location where the planner thinks they will collide. This helps the algorithm to calculate alternate trajectories efficiently.

Another extension was proposed by [12], where they emphasise on the safety of the robot. They create a map with collision distance ($d_{collision}$) which takes into account any obstacles that are outside the field of view or occluded by other static objects. This approach provides a trajectories with longer travel time but the overall safety of the robot is far more compared to classical DW approach.

Velocity obstacle

Originally developed by Fiorini et al.[13], this approach proposes to avoid obstacles by choosing velocity outside *collision cone*. The robot and the obstacles are considered circular. The robot contains the information about the position, orientation and the velocity of the obstacle. The obstacles are believed to travel at constant velocity and same orientation. The set of relative velocity of the robot and obstacle which result in collision is termed as a collision cone. By selecting an appropriate velocity outside the collision cone (avoidance velocity) of all the moving and static obstacles, the robot can ensure collision avoidance. Additionally, the choice of admissible velocity is restricted to the velocity allowed by the kinematic and dynamic constraints. These are called *admissible avoidance velocities*. This approach unifies the representation for avoiding static and dynamic obstacles. This idea has since been transformed to incorporate many scenarios [1].

One of the extensions, NLVO, was proposed by Shiller et. al. [14]. They extend the classical velocity obstacle approach that deals with obstacles moving linearly with constant velocity to propose *non linear velocity obstacles* that can deal with obstacles moving on arbitrary path at constant velocity. The concept of NLVO was used very successfully in [2], where they successfully implemented and test their extension of NLVO on a car-like robot. They have adapted the method such that it can deal with non circular robot and obstacles. As this can be computationally expensive, they used graphics library to speed up the collision calculation.

A reactive method for VO was developed in [15]. They propose the idea of virtual plane which is obtained by transforming moving obstacles into static obstacles and transforming robot to a virtual robot. This plane is used to calculate the time to collision and collision position. By using this information, even a simple planner can be used to avoid collision. The authors state that any of the previous approaches that were used to solve static motion planning can be used here for desired outcome. They implemented and test their approach with multiple moving obstacles with acceleration.

For multi robot systems, Van den berg et al.[16][17] have extended VO approach. The agents calculates the collision cone for another agents that are on the course of collision with it and chooses an admissible avoidance velocity that makes the changes only half of what is required. As all the agents are running the same algorithm and both of them will make half the effort, they will avoid collision without communicating with each other.

Velocity space

An approach using velocity time space was proposed by [18][19]. If the obstacles move with constant velocity in same direction then the collision time and distance can be calculated. This information is mapped to velocity time space. This creates a surface in this 3D space which is used to choose a velocity for the robot so that it does not collide with this obstacle. The target is also mapped to velocity space.This space was named Dynamic velocity space (DVS). Simulated experiments were performed with 3 dynamic obstacles.

2.2.2 ICS based

"Inevitable collision states (ICS) are states that will end up in a collision, no matter which trajectory is followed by the robot" [20]. This formal definition and a few important ground work was provided in [20]. This works on a state time space concept. Later, [21] declared that ICS is compliant with the safety criteria necessary for a robot to move safely in a dynamic environment. These safety criteria are as follows

1. "The robot should consider its own dynamics
2. The robot should consider the future behaviour of environment objects
3. The robot system should reason over an infinite time horizon" [21]

A partial motion planner combined with ICS-checker was proposed in [22]. They provide an additional property which proves that if the trajectory is collision free and the end state is not an ICS, then the whole trajectory is ICS-free. This allows them to calculate partial path and only check the final state for ICS. Their collision checker is rapidly exploring random tree based and only checks for braking trajectories of the robot. The algorithm was implemented on a car-like robot in simulation.

Later [23] proposes ICS-AVOID, an ICS based collision avoidance approach. They use the ICS-checker proposed by [24] and combine it with *safe control kernel*. This guarantees that the sampling output will always contain at least one safe trajectory. This enables their algorithm to go from one ICS-free state to another without failure. This is later compared with state of the art approaches based on dynamic window (TVDW[11]) and velocity obstacle (NLVO[2]). These experiments were performed in simulation and the information about the future trajectories of the obstacles (up to a limited time horizon) were provided to each of the algorithm. The results show that when same amount of information is provided, ICS-AVOID performs far better than TVDW and quite better than NLVO.

An on-line planner based on the concept of NLVO and ICS was proposed by [25][26]. It is based on selecting an appropriate time horizon such that the robot will either pass ahead of the obstacle or stop before the obstacle. The NLVO till that carefully calculated time horizon guarantees that the robot never ends up in an ICS state. The approach was implemented on a robot in simulation and it was tested with 70 moving obstacles.

The ICS method was extended to probabilistic domain by [27]. This method was later extended by the same authors in [28]. They introduce probabilistic collision state (PCS). So, instead of checking for collision, they are calculating the probability for collision. This probability is simply the minimum probability of a collision occurring by following a trajectory out of all the possible trajectory. This number determines the probability of that state being an ICS. The probabilities are calculated by considering the motion model of the obstacles and sampling from the probability density function of their resultant position at the end of some time interval. They have used only braking trajectories for collision checking since the infinite time horizon is infeasible. The authors have also considered collisions that occur among the obstacles excluding the robot. The authors have performed a string of experiments to check the effects of different parameters on PCS collision avoidance system on a car-like system.

Since guaranteed ICS free navigation is impossible, Braking ICS was proposed[29]. The authors state that Braking ICS free navigation is the next best solution. Their algorithm PASSAVOID is *provably passively safe* meaning that the robot will be at rest if a collision occurs. A braking ICS is a state such that no matter what braking trajectory is executed by the robot, it will still collide. They provide this algorithm under the assumption of limited field of view in an unknown dynamic environment. Simulated experiments were performed for a car like robot in the middle of 22 arbitrarily moving obstacles.

An extension of PCS based navigation was developed by [30]. The robot constantly updates a probabilistic collision matrix based on the model of the environment. For each obstacle, sampling is done based on the previous model which considers its velocity, acceleration, direction of movement, etc. For trajectory calculation, authors have used time dependent Dijkstra algorithm. In exchange for a little higher collision probability, the authors get a huge computational speedup along with a shorter trajectory length by setting a minimum probability restriction. This means that if the collision probability of a cell is lower than a predefined threshold, then that cell is considered safe to traverse. The authors have performed extensive simulation experiments with trajectory generation with different parameters in order to explain the algorithm. For online planning, the authors have modified their algorithm so that the robot only follows the planned trajectory for certain period of time and alters the plan based on the newly available sensor data. The algorithm was tested on a real robot and pedestrians were used as obstacles. The robot was able to get from one position to another with very less collision probability.

2.2.3 Roadmap based

Probabilistic roadmap approach was proposed by [31]. "A PRM planner samples the robots configuration space at random and retains the collision-free samples as milestones. It then tries to connect pairs of milestones with paths of predefined shape (typically straight-line segments in configuration space) and retains the collision-free connections as local paths. The result is an undirected graph, called a probabilistic roadmap, whose nodes are the milestones and the edges are the local paths"[31]. This is a state space based approach. The sampling is done while considering kinematic and dynamic constraints imposed by the robot. The authors proved that as the number of sample increases, the probability of finding a path increases exponentially. Their approach is mainly for path planning, but they have provided with some additional techniques like sensing error counter measure, trajectory optimization, safe mode planning and on-the-fly replanning which makes their approach quite suitable for dynamic environment. They performed experiments to prove that a robot with any shape, kinematic and dynamic constraint was able to traverse. Additionally they also experimented with moving obstacle with random trajectory.

An approach[32] was proposed for motion planning in dynamic environment where the global roadmap for static obstacles is already calculated. The authors provide a local and a global planning method which uses the previously given roadmap and only considers the dynamic obstacles. The problem is thus reduced to 1D state space. The local planner only considers going from one vertex to another in the roadmap, while the global planner employs multiple *probes* that individually run the local planner algorithm to find the global path. Experiments were performed for a 6 DOF robot in a 3D space in a simulation.

In [33], the authors propose a method to calculate and recalculate path to the goal in dynamic environment. This approach is similar to the extension from [31]. The method creates an anytime plan from the current position to the goal position which is constantly updated based on the sensor information. Anytime D* algorithm is used to calculate the trajectory in state time space. Experiments were performed for a point sized robot in an simulated dynamic environment. The moving obstacles can change their direction randomly but this information is not available to the robot. The robot still only collides approx. 20% of the time.

2.2.4 Elastic Band

Elastic bands approach was proposed in [34] in order to “close gap between global path planning and real time sensor based robot control”[34]. The authors proposed an intermediate step between global planner and controller which deforms the path produced by global planner based on local observations and feeds it to controller. This deformation is done by applying 2 types of forces (internal contraction and external repulsion) which makes the resultant path shortest. The elastic band is represented using a set bubbles with certain position and radius. This set is then manipulated at run time by changing the position and radius of the bubbles and adding or removing of bubbles. The authors have provided simulated experiments with multiple static and single moving obstacle with a holonomic robot.

Timed elastic band trajectory planning approach [35] proposes an approach to make adjustments to the plan generated by global planner. Here, the authors mention that they only deal with local trajectory and not the complete path when optimising. They mention that this makes the structure of optimisation problem sparse allowing them real time feedback. They explain timed elastic band as an extension of elastic band. Each configuration of elastic band is accompanied with a time duration. This duration represents the time that the robot will take to go from previous configuration to the next. The parameters such as velocity, acceleration, clearance from obstacles and compliance with robot's kinematic and dynamic constraints depend mostly on 2–3 adjacent configurations. The authors use hyper-graph to represent the constraints of this sparse system. This optimization occurs in control loop which enables the robot to get the motion commands directly. The authors preformed experiments with a simulated and a real differential drive robot in presence of an moving obstacle. They also tested 4 static obstacles. They report that each iteration takes on average 48 ms. This approach has a possibility for local minima problem because the planner only takes into account few future configuration.

The local minima problem mentioned above is addressed in [36] employing the concept of homotopic trajectories (trajectories with same initial and final position which can be continuously morphed into one another). The robot continuously tries to find homotopic trajectories around the obstacle and chooses the one which satisfies the constraints the best. The authors compare their approach with DWA and their previous approach[35] on a differential drive robot with 8 moving obstacles. They show that their approach is able to reach the goal at maximum velocity.

2.2.5 Spline based

An approach proposed in [37] extends [38] for real time motion planning for robots in dynamic environment. "Splines are piecewise polynomial functions that can easily represent complex trajectories, using few variables"[37]. The approach in [38] provides motion planning in dynamic environment. The authors of [37] extend that approach by adding moving obstacles to the set of constraints (adding hyperplanes which separate obstacles position and robot position). They add soft constraints for distance from nearest obstacles for extra safety. Additionally they add receding horizon control and suggest to use high rate of observation and control loop for better execution. They perform experiments on simulated and real holonomic robot with static and moving obstacles. They also consider the possibility of an obstacle moving faster than the highest velocity of robot.

The above approach is further extended for robot with bicycle kinematics in [39] and a more detailed mathematical explanation is provided. They conduct experiments with holonomic, unicycle and bicycle model of robots in dynamic environment in simulation. Additionally they conduct experiments on real holonomic and differential drive robot with single moving obstacle. The average time taken for each iteration is twice in case of differential drive experiment with a maximum of 0.98 seconds.

2.2.6 Other approach

Nearness diagram

A reactive navigation approach was proposed in [40]. They employ *situated activity* paradigm. The robot chooses from a set of 5 behaviour based on the sensor information. The sensor information is like vector field histogram. The robot behaves differently when there are object on its sides compared to where it is directly aligned to the goal. This algorithm was proved to not have local minima. The authors state that the proposed method only has one parameter to be selected compared to other approaches where numerous parameters determine the behavior of the robot. Experiments were performed on a real holonomic robot in a cluttered environment.

Potential field

Concept of potential field is quite dated. However a potential field approach for dynamic environment was proposed in [41]. Here the target and obstacles are moving and they generate attractive and repulsive forces respectively. Problem of local minima was addressed. Calculations for holonomic robot and differential drive robot were provided. Experiments were performed in simulation and on real robots.

Biologically inspired

A biologically inspired approach for differential drive robots was presented in [42]. Here the robot calculate the angle made by the obstacle in the field of sensing and calculates the linear and angular velocity that it must obey in order to avoid the obstacle. The robot follows two type of motion: avoidance motion and straight to goal motion. The robot switches between these modes to reach the target location. Experiments with multiple moving obstacles were performed in real and simulated environment. The approach was compared with velocity obstacle approach and the proposed method was faster to reach the target location.

2.3 Comparison

A brief comparison between the approaches discussed in 2.2 are shown in Table 2.1. There are different criteria for comparing approaches in the field of motion planning in dynamic environment. Apart from avoiding collision, the algorithms generally have at least one additional goal (generally reaching target in minimal time).

This target position is generally provided to the robot as problem statement, though the target may not be in line of sight. In this case, the robot must plan a path from its current position to the target location. For this task, it will need a map of the environment. The approaches discussed above have different assumptions regarding this parameter of the experiment. While [9, 10, 11, 12, 22, 31, 33, 41] provide global connection to the goal, most of the approaches do not. They can, however, be extended by combining with a global path planner that provides checkpoints which in line of sight of the robot.

Some approaches are only applicable to certain vehicle model types while most of them are applicable to every vehicle type. There are mainly 3 types of robot models, namely holonomic (can instantaneously move in any direction), unicycle (can move instantaneously in the direction it is facing and can turn instantaneously with certain angular acceleration) and bicycle (can instantaneously move in the direction it is facing and can turn within certain angle depending on the linear velocity)[5]. Naturally, an approach which can be applied to all types of robot model is desirable. Though, in indoor environments, bicycle type robots might not be very useful because of their high turning radius. This makes them less maneuverable in narrow and cluttered spaces.

Even though an approach can be theoretically applied to any model of robot, there are some technical reasons why that is not always desirable. For example, the approach mentioned in [39] is applicable on all models of robots and the authors show experiments proving that it is applicable. The single iteration rate was shown

Approach	Method	Vehicle type	Obstacle restriction	Obstacle shape	Experiment
DWA	[8]	Holonomic	-	All	real
	[9]	Holonomic	-	All	real
	[10]	Holonomic	-	-	sim
	[11]	Holonomic	cv, vd	All	sim
	[12]	Holonomic	cv, cd	All	real
VO	[13]	Holonomic+	cv, cd	circular	sim
	[1]	Unicycle+	cv, cd	All	real
	[14]	Holonomic+	cv, vd	circular	sim
	[2]	Bicycle	cv, vd	All	sim
	[15]	Unicycle	vv, vd	circular	sim
	[16]	Holonomic	vv, vd	circular	sim
	[17]	Holonomic	vv, vd	circular	sim
VS	[18, 19]	Unicycle+	cv, cd	All	sim
ICS	[22]	Bicycle+	vv, vd	All	sim
	[23]	Holonomic+	cv, vd	circular	sim
	[24]	Bicycle+	cv, vd	circular	sim
	[25]	Holonomic+	cv, vd	circular	sim
	[26]	Holonomic+	cv, vd	circular	sim
	[28]	Bicycle+	vv, vd	circular	sim
	[29]	Bicycle+	vv, vd	All	sim
	[30]	Unicycle+	vv, vd	All	sim, real
Roadmap	[31]	All	vv, vd	All	sim, real
	[32]	All	-	circular	sim
	[33]	All	cv, vd	circular	sim
Elastic Band	[34]	Holonomic+	vv, vd	circular	sim
	[35, 36]	All	vv, vd	All	sim, real
Spline	[37]	Holonomic	vv, vd	All	sim, real
	[39]	All	vv, vd	All	sim, real
	[40]	Holonomic	-	All	real
	[41]	Holonomic, Unicycle	cv, cd	All	sim, real
	[42]	Unicycle	cv, cd	All	sim, real

Table 2.1: Comparison of some of the state of the art approaches for motion planning in dynamic environment

holonomic+: can be applied theoretically to any vehicle type

cv: constant velocity, vv: varying velocity

cd: constant direction, vd: varying direction

sim: simulated experiments, real: experiments performed on an actual robot

(Note: The experiments might have relaxed some assumptions taken for theoretical approach. The *Obstacle restriction* and *Obstacle shape* information is based on the information provided from experimental setup. If a literature provides multiple experiments with different relaxed assumption, then we consider the assumption which is most resembles the theory.)

to be as high as 0.98 seconds for single moving obstacle for a differential drive robot. Whereas for a holonomic robot with same environmental condition it was 0.4 seconds. This may increase with additional moving obstacles which makes this approach only suitable for holonomic robots for a safe real time application.

Roadmap and ICS based approaches are not dependent on the model of the robot as they plan in state time space of the robot. This automatically takes care of kinodynamic constraints during planning phase. As stated before, for a robot motion planner must consider it own dynamic[21]. Most of the methods discussed above consider kinodynamic constraints.

When the experiments are performed in a simulated environment, the information about the position and velocity vector of the obstacles as well as the position and heading of the robot are easily available. But, in real environment, these information is erroneous/uncertain in nature. The approaches which take this uncertainty into account are clearly more desirable. This is one of the only reason why ICS based approaches are not declared as the obvious choice. Even though they have the capability to reason over infinite time, they need perfect information about the environment to guarantee safety. On the opposite end of the spectrum are reactive approaches, which only reason about next iteration. What they lack in planning long term decision, they try to make it up on the frontier of planning with lack of information.

The approaches which are applicable in the environment where the obstacles are moving at variable velocity and in variable direction are more desirable because it resembles more closely to the real world. In real environment, a human might slow down to talk to someone or accelerate to overtake other people. All the state time space based approaches can theoretically deal with this constraint.

Some approaches assume only circular obstacles for the ease of calculation or for the experiments, but most of the approaches can be extended to consider non circular moving obstacles by considering more than one circle for an obstacle[2]. Theoretically, all state time space based approaches can deal with obstacles of any shape.

Clearly, state time space based approaches are better choice if the information about the environment is fairly accurate as they can deal with any robot model travelling in environment with obstacles of any shape travelling in variable direction at variable velocity. The possible problem might be the computation time required for them to plan a path in state time space after gathering the information.

Apart from scientific reasons, an approach is more favorable when an implementation is available as open source without any proprietary software. This makes its integration with robot possible for more people and makes the approach and its claims verifiable.

3

Methodology

3.1 Experimental Design

A planar mobile robot with a kinematic model is spawned at initial position. The robot needs to travel to the goal position and it should avoid collision with static and moving obstacles. A collision is considered to happen when the physical body (model in case of simulation) is in contact with physical body of the obstacle. The position of the robot in the environment is available to the motion planner. A global planner is also available to the motion planner if it needs one. The position, velocity and shape of the moving obstacle is also available to the motion planner. The experiment is performed multiple times and each time following things are measured

- time to reach goal
- number of collisions
- number of times motion planner had to re-plan

All these values are expected to be minimum for a better planner.

Re-plan attempts signify how well does a planner considers the information it receives. According to the safety criteria in [21], a planner should take into account the future behaviour of its environment and make decision accordingly. This implies that an ideal planner will only have to plan once. The equivalent of this in real scenario is that a planner will update its plan at regular interval but it is never forced to completely re-plan because the previous plan was no longer viable. Thus, lower re-plan count is more desirable.

3.2 Setup

KUKA youbot[43] is used which uses ROS[44]. It is a omni-directional robot which is steered as holonomic robot for the experiments (Refer A.1 for parameters). A gazebo simulator[45] is used for simulating the desired environment. (Refer A.3 for machine specifications)

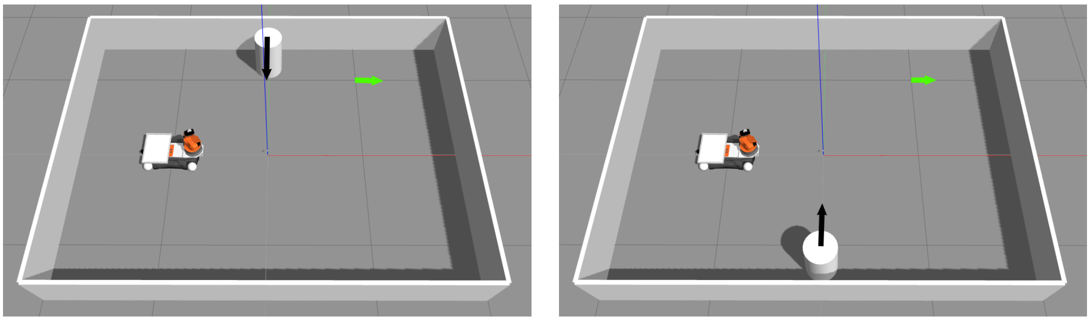
Global planner[46] is used from ROS navigation[47] if required. A planner is free to use AMCL[48] or truth value directly from simulator to get the robot's current position. If the planner uses AMCL, we localise the robot before starting motion planning.

Travel time is measured from the moment goal position is given to motion planner to the time when robot reached goal position and stops moving (For goal tolerance parameters refer A.2). The number of re-plan attempts are measured by manually counting each instance when the motion planner has to re-plan but the global planner did not re-plan. Number of collisions were measured by manually counting the number of times robot collided with any obstacle (static or moving) in simulator.

We run the planner multiple times to be unbiased against the position of moving obstacle at start time. Additionally, we also run the planner in static environment to obtain a benchmark. All obstacles are cylinders of diameter 30 cm and height 50 cm. None of the obstacles move faster than 0.4 m/s. All obstacles follow to and fro motion. The obstacles have linear acceleration and deceleration. This implies that their velocity is constantly changing as shown in Figure 3.4.

3.3 Test case 1: Single obstacle single room

This test case is similar to the classic road crossing benchmark[16]. The goal is in line of sight of the robot. The motion planner does not need to rely on global planner. A single obstacle is moving between the robot and the goal as shown in Figure 3.1. The velocity of the moving obstacle is constantly changing as shown in Figure 3.4a. The size of the room is 4 meters in length and 3 meters in width. This represents a small room or a small subsection of a large room in a typical indoor environment. Apart from walls of the room, no additional static obstacles are simulated. The robot starts at `x=-1.0, y=0.0, theta=0.0` and the goal is at `x=1.0, y=1.0, theta=0.0`. The maximum opening for the robot is 2.7 meters when the obstacles is at the end points of its linear path.



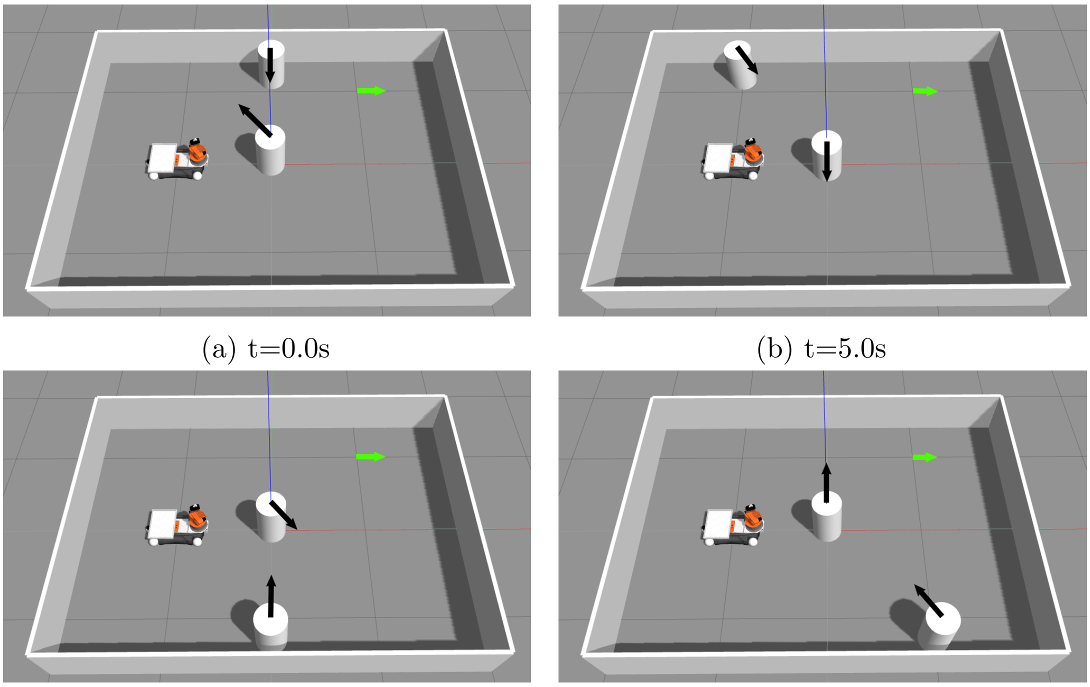
(a) t=0.0s

(b) t=10.0s

Figure 3.1: Test case 1 obstacle trajectory (green arrow is goal position)

3.4 Test case 2: Two obstacle single room

The goal is still in line of sight of the robot. This test case is a logical extension of the previous test case. There are 2 obstacles between the robot and the goal. The difficulty is almost double because the opening for the robot is always less than 1.7 meters. This is because the obstacles are offset by 50% as shown in Figure 3.2. The initial and goal positions are same as test case 1. The second obstacle (the obstacle moving diagonally) slows down near origin to coordinate with the offset. This can be seen in Figure 3.4c.



(a) t=0.0s

(b) t=5.0s

(c) t=10.0s

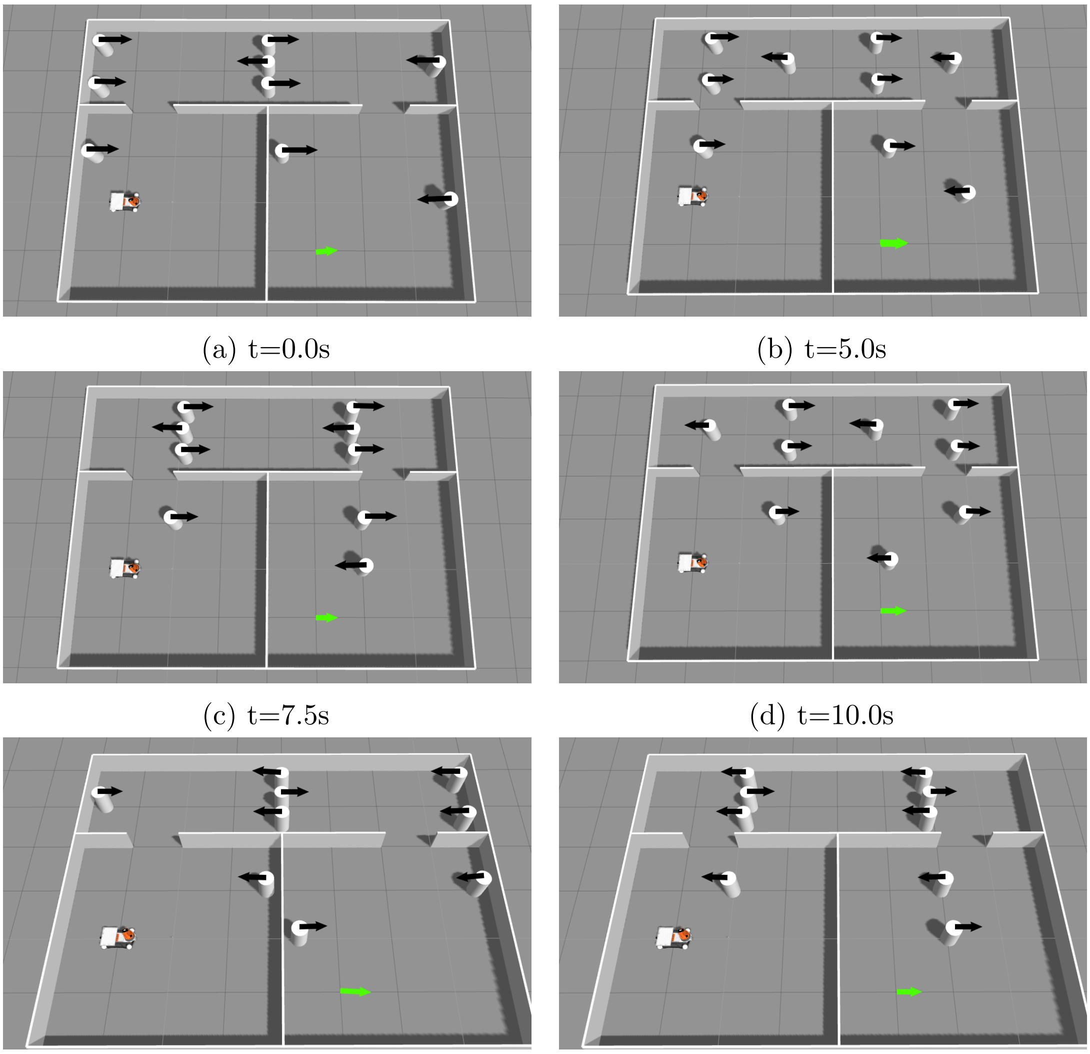
(d) t=15.0s

Figure 3.2: Test case 2 obstacle trajectories (green arrow is goal position)

3.5 Test case 3: Double room

Unlike the previous test cases, the goal is not in line of sight in this test case. The robot has to exit its current room, travel through a corridor filled with moving obstacles and enter another room to get to the goal position. The corridor is 2 meters wide and 8 meters in length.

The corridor contains 6 obstacles which move in 3 parallel tracks as shown in Figure 3.3. We expect a zigzag trajectory from an ideal motion planner. However, the ideal planner still would have to wait for the obstacles to move near time `t=0.0s` and `t=15.0s` and their multiples. Additionally, there is one obstacle in the room the robot is spawned and 2 moving obstacles in the next room. All obstacles have the same velocity for their trajectory, only their directions are different. The velocity of the moving obstacles in this test case (Figure 3.4c) is almost same as test case 1, the only difference is the time of their travel is extended before they repeat their behaviour. This makes their maximum velocity a little less than the maximum velocity of moving obstacles in test case 1. The robot starts at `x=-1.0,` `y=0.0, theta=0.0` and the goal position is `x=3.0, y=-1.0, theta=0.0`.

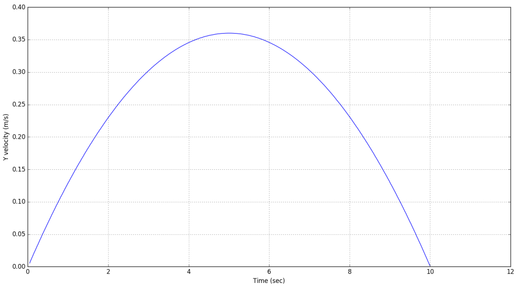


(a) t=0.0s (b) t=5.0s

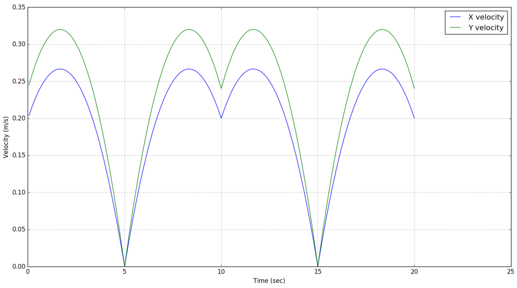
(c) t=7.5s (d) t=10.0s

(e) t=15.0s (f) t=22.5s

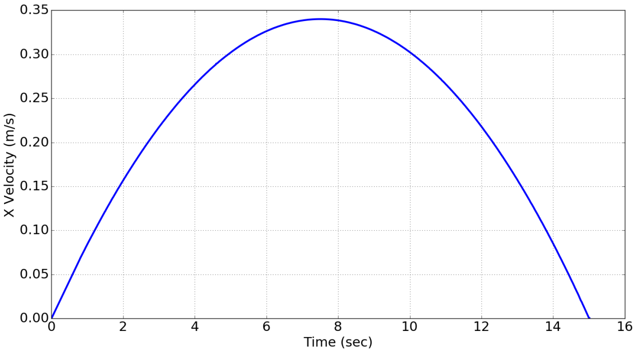
Figure 3.3: Test case 3 velocity trajectories (green arrow is goal position)



(a) First obstacle in test case 1 and 2 (the obstacle moving vertically)



(b) Second obstacle in test case 2 (the obstacle moving diagonally)



(c) Obstacles in test case 3

Figure 3.4: Velocity vs. time plot for moving obstacles

4

Implementation

4.1 Moving obstacles

The moving obstacles are animated with the help of a gazebo plugin modified from[49]. The plugin takes information like key frames and position and orientation of object at each key frame and animates the objects in gazebo. It also advertises the position and orientation of the obstacles at each iteration to a gazebo channel. This is then read by a ROS node and published as a ROS message every 15 ms. Additionally we have created small modular ROS nodes which

- calculate relative position of obstacles w.r.t. robot
- calculate instantaneous velocity of obstacles
- publish robot's position in the world
- converting moving obstacle information to costmap (Refer **??**)

to be used if a planner needs it. The source code for these plugins and helper nodes are available at [50]

4.2 Obstacle trajectory to costmap

This ROS node script (`obs_to_costmap`) calculates the instantaneous velocity of the moving obstacle and creates fake occupancy information for the future track of the moving obstacle. This tricks a planner into thinking certain areas are occupied whereas, in reality, they are not. This node can be used with any static local planner which considers costmap[51] information.

We find that choosing appropriate time duration (`lookahead_time`) for extrapolating future trajectory is a trade off. If the time is too short, the planner might collide with the obstacle. On the other hand, if the time is too long, the planner's path may be blocked completely and it will be forced to wait until the costmap clears (this will increase the time to reach goal).

4.3 Planners to be evaluated

4.3.1 TEB local planner

This planner[52] is implementation of Timed elastic band approach[36]. The planner is used with ROS navigation [47]. It is used as local planner in `move_base` which publishes `cmd_vel` message to control robot. The planner makes small changes to the path generated by the global planner. To get the information about the dynamic obstacles, teb local planner uses `obstacles` ROS message. It can contain the information about the shape of the robot (as polygon), its position and its velocity. This information is provided to teb local planner by one of the helper nodes in [50]. `move_base` listens to ROS topic `move_base_simple/goal` to get the position and orientation of the goal.

4.3.2 Spline based planner

This planner[53] is implementation of Spline based planner approach[39]. By default the planner works in its own environment and simulator. The planner can be incorporated with ROS framework[54]. We have, however, not made it compatible with ROS navigation. Even though the planner is incorporated with ROS, it does not use gazebo simulator. The plans in its own simulator and the wrapper built around this code are used to translate the information from one simulation to another. This could be a major handicap for this planner because it uses 2 simulation simultaneously which will slow down the simulation and increase the response time. The planner simulator requires the size of rooms and position and shape of obstacles before starting. This may become a problem if the robot does not have information about each moving obstacle. For the purpose of this experiment, we provide the planner with information about all moving obstacle no matter if the robot can sense the obstacle with its sensors or not.

4.3.3 DWA local planner

DWA local planner[55] is implementation of Dynamic window approach [8]. This planner works as a local planner in ROS navigation[47]. The DWA approach alone cannot work without the global planner because it only consider a limited area around the robot to plan a local path. The planner needs to be configured to make changes to the global plan up to a certain amount. By default it is used for static motion planning. This puts DWA planner at a major disadvantage because it cannot predict the position of moving obstacle in future. For this reason, we couple the planner with a `obs_to_costmap` which changes the costmap for the environment according to the velocity of the moving obstacles.

4.3.4 EBand local planner

This planner[56] is implementation of Elastic band approach[34]. Similar to DWA planner and teb planner, EBand local planner also works as a local planner in ROS navigation[47]. This is because it makes changes to the path generated by the global planner. The planner gets the information of the obstacles from costmap and does not use any other source of information.

4.3.5 EBand with obs_to_costmap

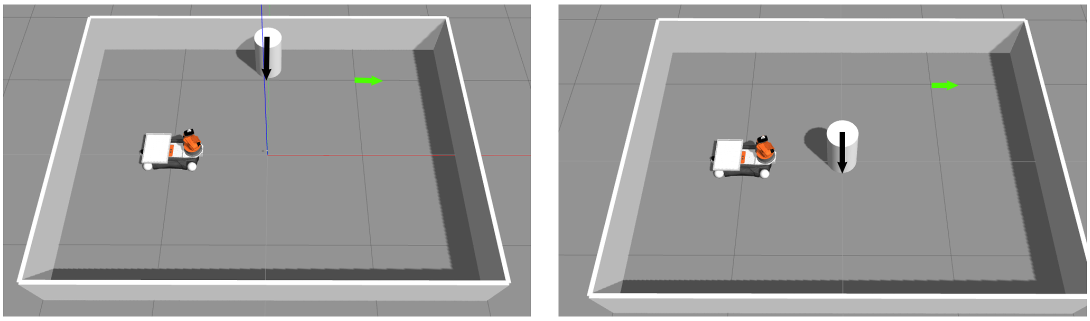
Similar to DWA local planner, we couple EBand planner[56] with `obs_to_costmap`. As the planner only considers costmap information, the new costmap will provide information about the future poses of the moving obstacles to the planner. We expect this approach to perform better than Eband (4.3.4) in terms of number of collisions and worse in terms of time taken to reach goal. We expect that because this approach will be forced to choose safer trajectory which might force the robot to wait for the path to clear or force the robot to go around the moving obstacle rather than passing it. From now on, we will refer to EBand with obs_to_costmap as Eband2.

5

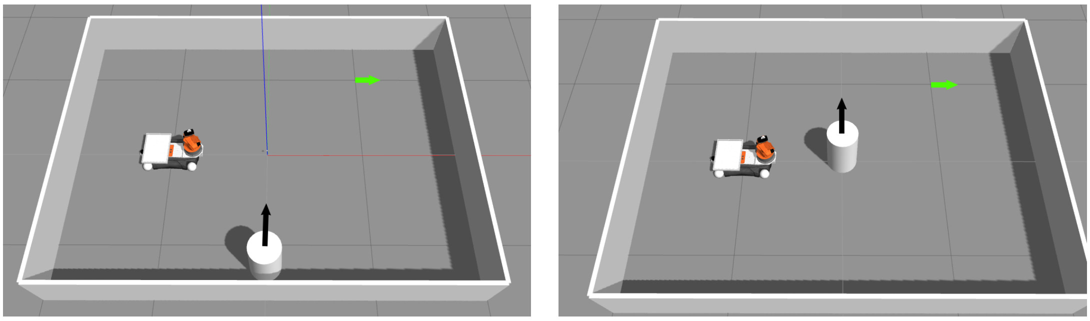
Evaluation

5.1 Test case 1

4 experiments were performed starting at different time. This makes the evaluation unbiased of the starting position of the obstacles. There is a single obstacle moving along y axis. Figure 5.1 shows the start position and direction of the moving obstacle for each experiment. The trajectory and the velocity of the moving obstacle is same for each experiment as described in **??**.



(a) Experiment 1 (b) Experiment 2



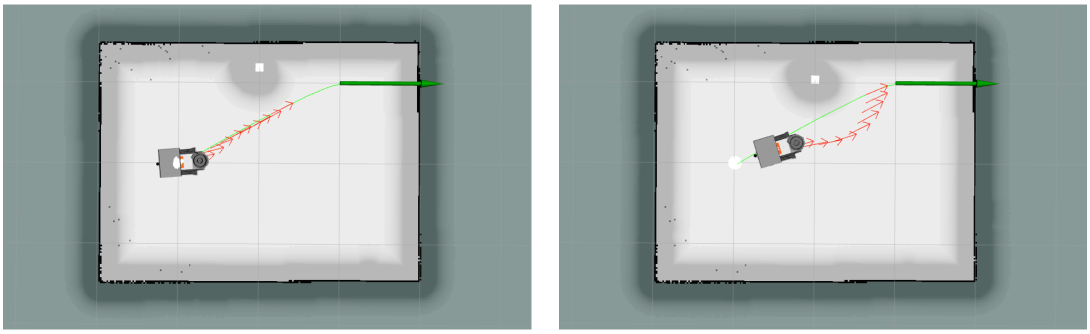
(c) Experiment 3 (d) Experiment 4

Figure 5.1: Moving obstacle position and direction (black arrow) for experiments in test case 1. (Green arrow is goal position and orientation)

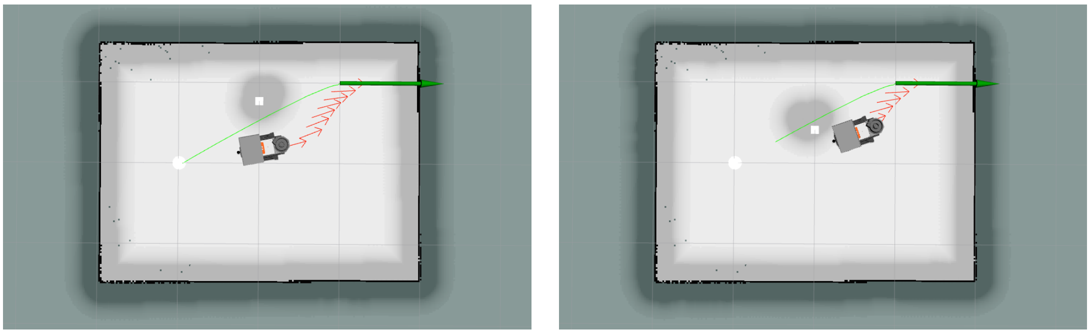
5.1.1 TEB local planner

Experiment	Travel time	Re-plans	Collisions
static	5.205	0	0
1	6.085	0	0
2	5.493	0	0
3	5.295	0	0
4	7.016	0	0

Table 5.1: Performance of teb local planner navigating in test case 1



(a) t=0.39s (b) t=1.58s



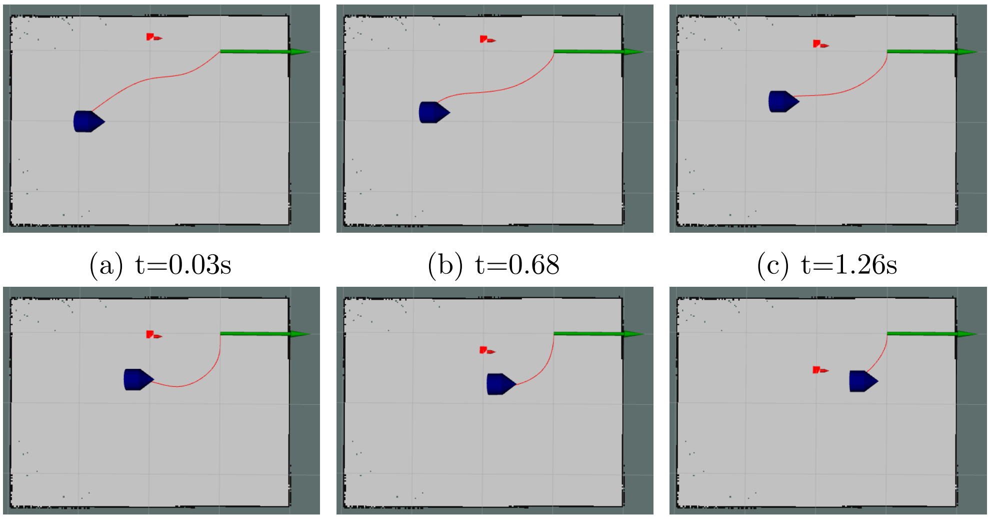
(c) t=2.67s (d) t=3.81s

Figure 5.2: teb local planner navigating in test case 1 (experiment 1) (Line in green is global path, red arrows are local planner's path, white square is the obstacle and green arrow is the position of goal)

5.1.2 Spline based planner

Experiment	Travel time	Re-plans	Collisions
static	5.421	0	0
1	5.958	0	0
2	5.347	0	0
3	5.326	0	0
4	5.664	0	0

Table 5.2: Performance of spline based planner navigating in test case 1



(a) t=0.03s (b) t=0.68 (c) t=1.26s

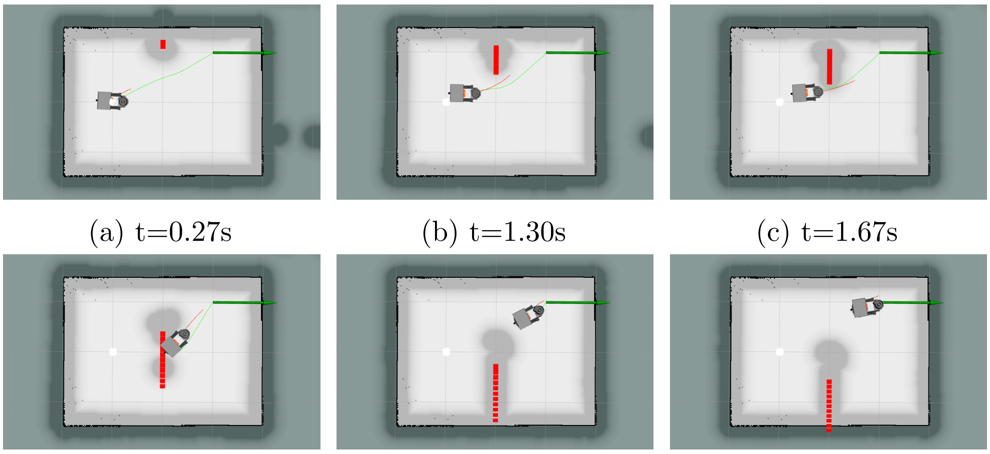
(d) t=1.90s (e) t=2.77s (f) t=3.65s

Figure 5.3: Spline based planner navigating in test case 1 (experiment 1) (red line is motion planner's path, red square is moving obstacle and green arrow is goal position)

5.1.3 DWA planner

Experiment	Travel time	Re-plans	Collisions
static	19.526	0	0
1	20.064	0	0
2	20.887	0	0
3	15.935	0	0
4	13.614	0	0

Table 5.3: Performance of DWA planner navigating in test case 1



(a) t=0.27s (b) t=1.30s (c) t=1.67s

(d) t=3.96s (e) t=5.80s (f) t=6.67s

Figure 5.4: DWA planner navigating in test case 1 (experiment 1) (Red squares are obstacle and the fake obstacles created by `obs_to_costmap`, green line is global path, red line is local path generated by DWA and green arrow is goal position)

5.1.4 EBand planner

Experiment	Travel time	Re-plans	Collisions
static	6.270	0	0
1	8.230	0	1
2	6.034	0	0
3	6.309	0	0
4	5.705	1	0

Table 5.4: Performance of EBand planner navigating in test case 1

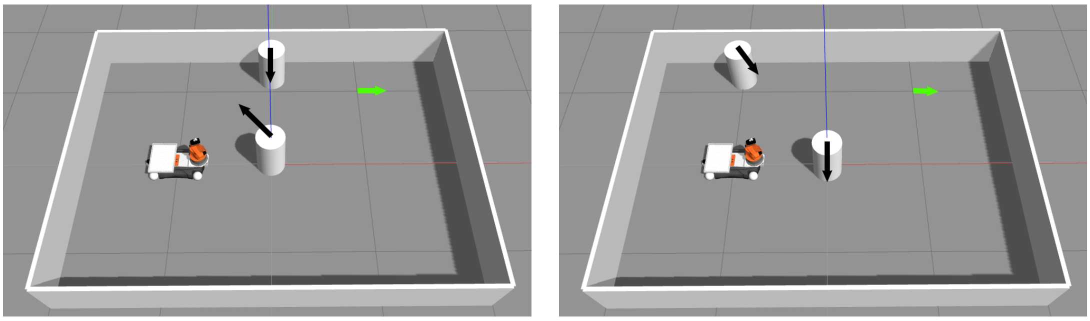
5.1.5 EBand2 planner

Experiment	Travel time	Re-plans	Collisions
static	6.270	0	0
1	5.814	1	0
2	5.884	0	0
3	5.718	0	0
4	6.684	1	0

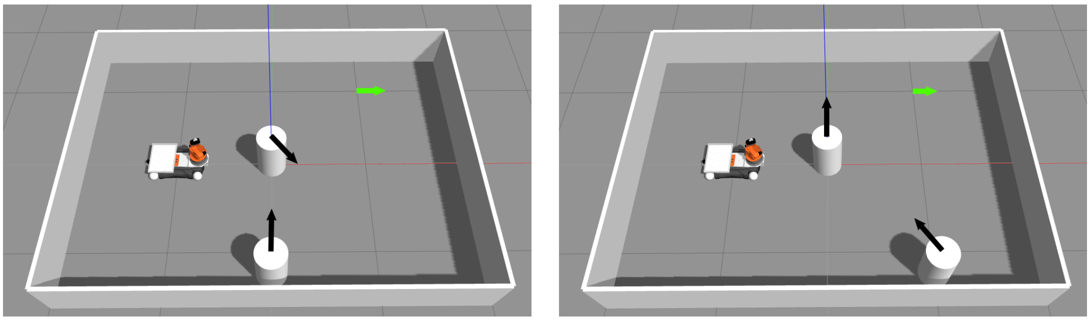
Table 5.5: Performance of EBand2 planner navigating in test case 1

5.2 Test case 2

This test case contains an additional moving obstacle compared to test case 1. We perform 4 experiments starting at different time. This makes the evaluation unbiased with respect to the starting position of the obstacles. Figure 5.5 shows the start positions and directions of the moving obstacles for each experiment. The trajectory and the velocity of the moving obstacle is same for each experiment as described in **??**.



(a) Experiment 1 (b) Experiment 2



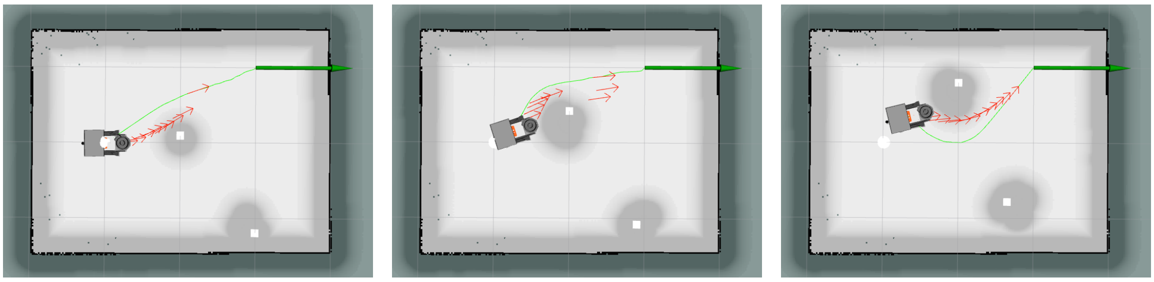
(c) Experiment 3 (d) Experiment 4

Figure 5.5: Moving obstacle positions and directions (black arrow) for experiments in test case 2. (Green arrow is goal position and orientation)

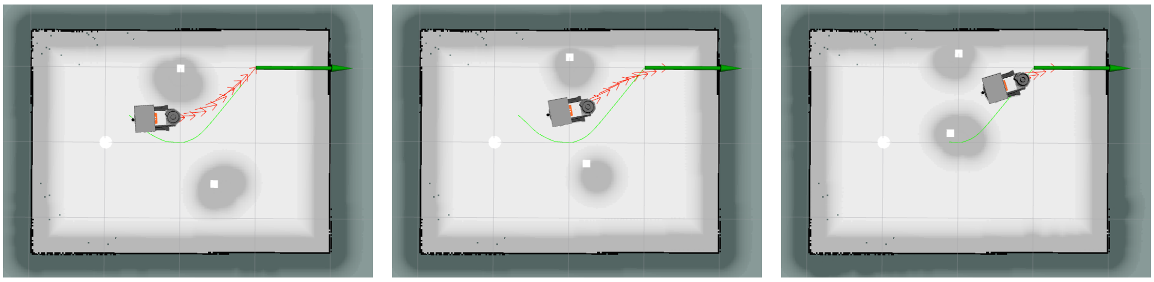
5.2.1 TEB local planner

Experiment	**Travel time**	**Re-plans**	**Collisions**
static	5.205	0	0
1	7.381	1	0
2	5.097	0	0
3	5.400	0	0
4	6.454	1	0

Table 5.6: Performance of teb local planner navigating in test case 2



(a) t=0.27s (b) t=1.18s (c) t=2.38s



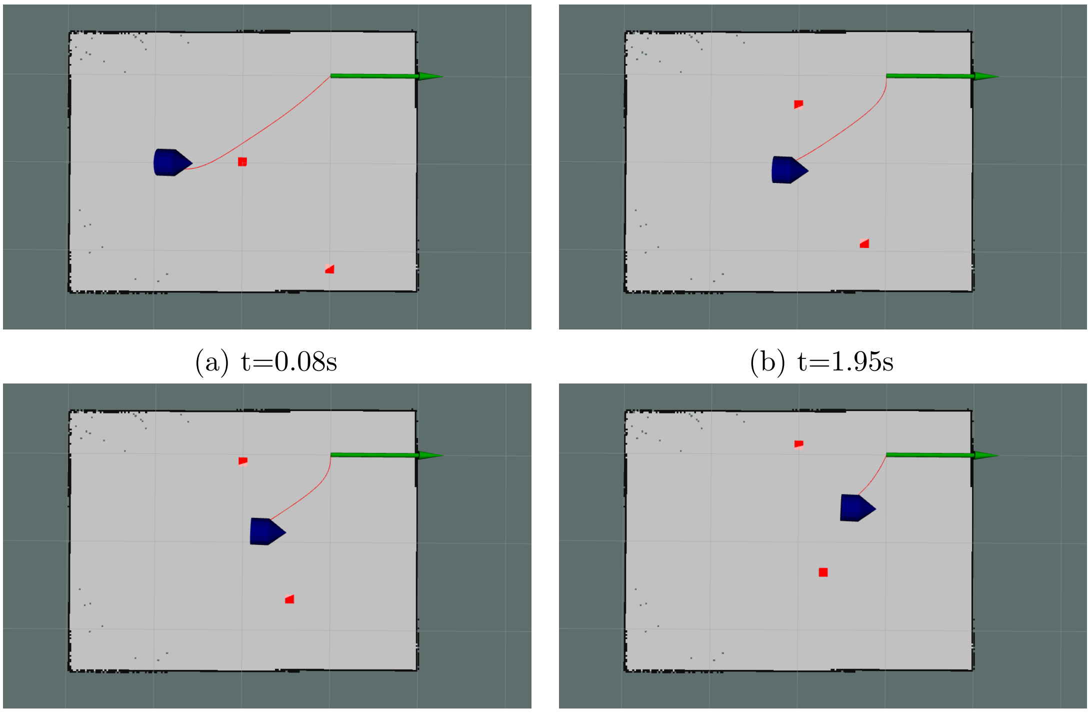
(d) t=3.15s (e) t=4.00s (f) t=5.45s

Figure 5.6: teb local planner navigating in test case 2 (experiment 4) (Line in green is global path, red arrows are local planner's path, green arrow is goal position and white square is the obstacle)

5.2.2 Spline based planner

Experiment	Travel time	Re-plans	Collisions
static	5.421	0	0
1	6.449	0	0
2	5.405	0	0
3	5.423	0	0
4	6.000	0	0

Table 5.7: Performance of spline based planner navigating in test case 2



(a) t=0.08s

(b) t=1.95s

(c) t=2.83s

(d) t=3.82s

Figure 5.7: Spline based planner navigating in test case 2 (experiment 4) (red line is planner path, blue arrow is robot, red squares are obstacle positions and green arrow is goal position)

5.2.3 DWA planner

Experiment	**Travel time**	**Re-plans**	**Collisions**
static	19.526	0	0
1	17.773	0	0
2	10.877	0	0
3	14.046	0	0
4	25.760	2	2

Table 5.8: Performance of DWA planner navigating in test case 2

5.2.4 EBand planner

Experiment	**Travel time**	**Re-plans**	**Collisions**
static	6.270	0	0
1	-	Failed	2
2	5.590	0	0
3	6.242	0	0
4	6.887	1	0

Table 5.9: Performance of EBand planner navigating in test case 2

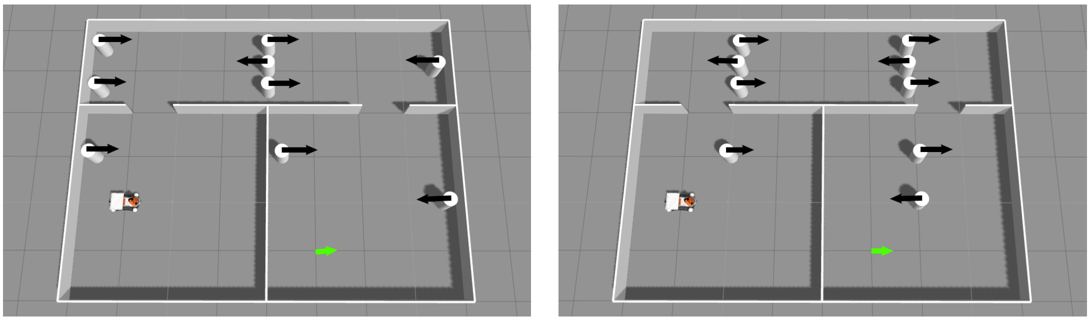
5.2.5 EBand2 planner

Experiment	**Travel time**	**Re-plans**	**Collisions**
static	6.270	0	0
1	6.933	1	0
2	5.952	0	0
3	5.694	0	0
4	6.355	1	0

Table 5.10: Performance of EBand2 planner navigating in test case 2

5.3 Test case 3

There are 2 rooms connected by a corridor in this test case environment. We perform 2 experiments starting at different time. This makes the evaluation unbiased with respect to the starting position of the obstacles. The moving obstacles are all in sync in terms of velocity. They all stop at the same time and have a highest velocity at the same time. Figure 5.8 shows the start positions and directions of the moving obstacles for each experiment. The trajectory and the velocity of the moving obstacle is same for each experiment as described in **??**.



(a) Experiment 1

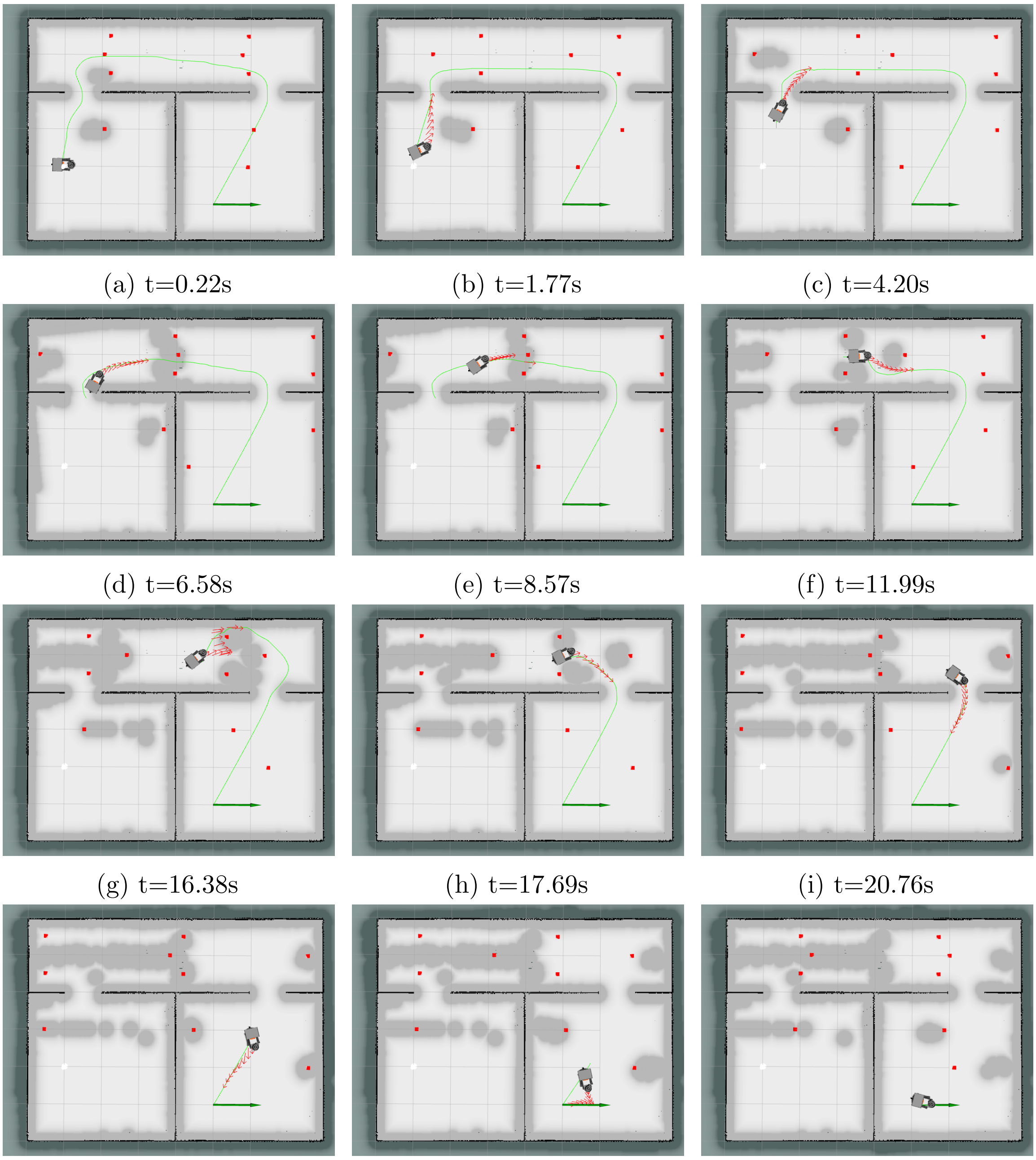
(b) Experiment 2

Figure 5.8: Moving obstacle positions and directions (black arrow) for experiments in test case 2. (Green arrow is goal position and orientation)

5.3.1 TEB local planner

Experiment	**Travel time**	**Re-plans**	**Collisions**
static	27.139	0	0
1	37.399	4	0
2	27.204	2	0

Table 5.11: Performance of teb local planner navigating in test case 3



(a) t=0.22s (b) t=1.77s (c) t=4.20s

(d) t=6.58s (e) t=8.57s (f) t=11.99s

(g) t=16.38s (h) t=17.69s (i) t=20.76s

(j) t=24.51s (k) t=27.16s (l) t=04.91s

Figure 5.9: teb local planner navigating in test case 3 (experiment 2) (Line in green is global path, red arrows are local planner's path and red square is the obstacle)

5.3.2 Spline based planner

Spline based planner has its own method for combining global planner with the local motion planner. It uses iterative method to form a rectangular area around the robot as large as possible. It then gets a global plan (using A* algorithm) inside this region. The local planner uses this information to create a smooth and collision free path. This method works in the planner's simulator, however the parameters are extremely sensitive for the planner to work correctly (as of yet). When combined with ROS framework, this method does not work because the current method is only configured for a circular and very small robot navigating in a relatively large environment. The method fails completely in narrow passages. In test case 3, we have 2 narrow spaces (doorways) with static obstacles alone. We tried to combine the motion planner with the global planner so that the motion planner can only change trajectory between 2 global waypoints. The motion planner still fails at the narrow regions (even in a static environment).

Experiment	**Travel time**	**Re-plans**	**Collisions**
static	-	Failed	0
1	-	Failed	0
2	-	Failed	0

Table 5.12: Performance of spline based planner navigating in test case 3

5.3.3 DWA planner

Experiment	**Travel time**	**Re-plans**	**Collisions**
static	39.004	0	0
1	44.329	6	3
2	-	Failed	2

Table 5.13: Performance of DWA planner navigating in test case 1

5.3.4 EBand planner

Experiment	Travel time	Re-plans	Collisions
static	30.862	0	0
1	76.982	8	6
2	71.808	7	2

Table 5.14: Performance of EBand planner navigating in test case 3

5.3.5 EBand2 planner

Experiment	Travel time	Re-plans	Collisions
static	30.862	0	0
1	54.244	3	1
2	57.104	5	3

Table 5.15: Performance of EBand2 planner navigating in test case 3

6

Results

6.1 Time comparison

All planners are configured to minimise travel time. They start and end their journey at a predefined position and have same environmental conditions (Refer 3.2). The average travel time for all robots are shown in Table 6.1.

Planner	Static single room	Test case 1	Test case 2	Static double room	Test case 3
TEB	**5.205**	5.972	6.083	**27.139**	**32.578**
Spline based	5.421	**5.574**	**5.819**	-	-
DWA	19.526	17.625	17.114	39.004	44.329
EBand	6.270	6.570	6.240	30.862	74.395
EBand2	6.270	6.025	6.234	30.862	55.674

Table 6.1: Average time of travel for planners for different test cases

All planners are able to traverse in test case 1 in almost same time as their static environment counterpart (static single room). However, the planners take a significant amount of additional time for test case 2 and 3 than its static counterpart (static single room and static double room respectively). For test case 2, this can be attributed to the increase in number of obstacles compared to test case 1. For test case 3, there are 2 main reasons for the large additional time. First reason is larger environment (even the smallest room in test case 3 is 33% larger than the room in test case 1). Second reason is that for a few seconds (4–5 seconds every 30 seconds), complete corridor is blocked. The planner is forced to wait during this time until it can calculate a valid path again.

DWA planner takes the most amount of time among all planners in all cases. This is because the DWA approach favors circular motions which places the robot near the goal quite fast (as fast as other planners) but it then takes a lot of time to get to the exact goal in exact orientation. This can be seen in Figure 5.4. The robot is in the vicinity of the goal at `t=6.67s` but the total travel time is still 20 seconds.

EBand2 planner is slightly faster than EBand planner in test case 1 and 2. But the difference in their time is significantly large in test case 3. We expected the travel time to increase as mentioned in 4.3.5. The results are opposite in terms of travel time. This could be because EBand2 planner has more information about the moving obstacle than Eband planner. This information is velocity of the moving obstacle (implicitly calculated by `obs_to_costmap`). This allows EBand2 planner to take better decisions compared to EBand planner. Eband2 chooses safer but longer trajectory whereas EBand chooses faster trajectory but after a small duration it finds out that the trajectory is no longer possible and it is force to wait for a short amount of time.

Spline based planner performs best in terms of travel time for test case 1 and 2. TEB planner performs best in test case 3. If the spline based planner is correctly merged with a global planner, we believe that it would have worked better than TEB in test case 3 as well. The difference in travel time between spline based planner and TEB planner is very small. TEB planner is considered better because it is able to traverse all test cases and it does that in least amount of time.

6.2 Re-plan comparison

Number of re-plan attempts signify how well the planner processes environment information (Refer 3.1) We take the maximum of re-plan attempt of all experiments (Table 6.2). This provides the worst case scenario of a test case for that planner. The re-plan attempts are dependent on the initial position of the moving obstacle. Taking the worst case scenario eliminates the dependency for unbiased comparison.

Planner	**Static single room**	**Test case 1**	**Test case 2**	**Static double room**	**Test case 3**
TEB	0	0	1	0	3
Spline based	0	0	0	Fail	Fail
DWA	0	0	2	0	Fail
EBand	0	1	Fail	0	8
EBand2	0	1	1	0	5

Table 6.2: Maximum re-plan attempts of planners for different test cases

We see a general trend of increase in number of re-plans as the environment gets more complex. TEB planner is better than other planners in all test cases except test case 2. Spline based planner and TEB planner have less re-plan attempts compared to other planners in most of the test cases. This is because these planners are able to use information about the moving obstacles directly whereas the other planners get this information indirectly through costmap.

6.3 Collisions comparison

We compare average number of collisions for each planner across all experiments in Table 6.3.

Planner	**Static single room**	**Test case 1**	**Test case 2**	**Static double room**	**Test case 3**
TEB	0	0	0	0	0
Spline based	0	0	0	-	-
DWA	0	0	0	0	2.5
EBand	0	0.25	0.5	0	4
EBand2	0	0	0	0	2

Table 6.3: Average collisions of planners for different test cases

TEB planner is better in this regards without any exceptions. If the global planner can be integrated with spline based planner, we believe it performing as good as TEB planner. DWA planner performs better in test case 1 and test case 2, however it failed completely test case 3 experiment 2. Before failing, it collided into 2 obstacles, but left unchecked, it would collide into many more. We also observe that DWA planner collides in corridor when passing between 2 moving obstacles because of its preference to circular motion instead of straight. After TEB planner, EBand2 planner performs better. This is because it indirectly knows the velocity of moving obstacles whereas EBand does not. We expected this to result in 4.3.5.

6.4 Overall comparison

Considering comparison of all 3 criteria, TEB planner is better than other planners. Spline based planner could be better if it was integrable with a global planner. EBand2 planner would be next better.

7

Conclusions

7.1 Contributions

We compared some of the state of the art approaches in motion planning in dynamic environment. We setup a experiment and test criteria to compare motion planners in dynamic environment. We compare 5 motion planners in 3 test cases with increasing complexity in gazebo simulator. We analysed a better motion planner out of the 5 based on the test criteria set earlier. We provide software for easily modifying a ROS navigation code base for static environment to one for static and dynamic environment.

7.2 Lessons learned

A motion planner which uses information about moving obstacles directly performs better than a motion planner which learn that information indirectly. A mathematical simulation with circular robots, circular obstacles and robot fed with all true values along with perfect controller tests motion planning algorithm less robustly than a simulation which uses real robot models. A motion planner written in python programming language is much slower than a motion planner binary compiled from C++ source code. We observe that the gazebo simulation runs at 40 to 50% speed of real time when spline based planner[53] planner is used whereas ROS navigation planners[55, 52, 56] enables gazebo to run at 70 to 80% speed of real time.

7.3 Future work

Using perception techniques to gather information about the moving obstacles rather than gaining it from simulator would provide a comparison for more realistic world. Comparing motion planners on real robot with humans as moving obstacles would be the next step to compare motion planners. Comparing planners for different robot models with different kinematic and dynamic constraints would give a better idea about a more general and robust motion planner. Trying variable amount of `lookahead_time` for `obs_to_costmap` for static motion planners and comparing them with dynamic motion planner. Finally, combining and creating a motion planner by learning from the results of the comparison, might provide best results.

A

Parameters

A.1 Kinodynamic parameters

Parameter	**Value**
max X velocity	0.6 m/s
max Y velocity	0.4 m/s
max angular velocity	0.75 rad/s
max X acceleration	2.0 m/s^2
max Y acceleration	2.0 m/s^2
max angular acceleration	1.0 rad/s^2
min turning radius	0.0 m

Table A.1: kinodynamic parameters

A.2 Goal tolerance parameters

Parameter	**Value**
xy goal tolerance	0.1 m
yaw goal tolerance	0.1 rad

Table A.2: Goal tolerance parameters

A.3 Machine specifications

Parameter	**Value**
CPU	Intel (R) Core (TM) i5–4200U CPU @ 1.60GHz
RAM	8 GiB

Table A.3: Machine specifications

A.4 Costmap parameters

Parameter	**Value**
transform tolerance	0.4
obstacle range	2.5
raytrace range	4.0
inflation radius	0.5

Table A.4: costmap common parameters

Parameter	**Value**
global frame	/map
robot base frame	base footprint
update frequency	3.0
publish frequency	3.0
static map	true

Table A.5: Costmap global parameters

Parameter	**Value**
global frame	map
robot base frame	base footprint
update frequency	5.0
publish frequency	5.0
static map	false
rolling window	true
width	2.5
height	2.5
resolution	0.04
origin x	0.0
origin y	0.0

Table A.6: Costmap local parameters

References

[1] E. Prassler, J. Scholz, and P. Fiorini, “A robotics wheelchair for crowded public environment,” *IEEE Robotics & Automation Magazine*, vol. 8, no. 1, pp. 38–45, 2001.

[2] F. Large, C. Laugier, and Z. Shiller, “Navigation among moving obstacles using the nlvo: Principles and applications to intelligent vehicles,” *Autonomous Robots*, vol. 19, pp. 159–171, Sep 2005.

[3] S. Keshmiri and S. Payandeh, “An overview of mobile robotic agents motion planning in dynamic environments,” in *Proceedings of the 14th IASTED International Conference, Robotics and Applications (RA20), MA, Boston*, pp. 152–159, 2009.

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

[5] M. Hoy, A. S. Matveev, and A. V. Savkin, “Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey,” *Robotica*, vol. 33, no. 3, p. 463497, 2015.

[6] K. Fujimura, *Motion planning in dynamic environments*. Springer Science & Business Media, 1991.

[7] T. Tsubouchi, “Motion planning for mobile robots in a time-varying environment: A survey,” *Journal of Robotics and Mechatronics*, vol. 8, no. 1, pp. 15–24, 1996.

[8] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[9] O. Brock and O. Khatib, “High-speed navigation using the global dynamic window approach,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, pp. 341–346, IEEE, 1999.

[10] P. Ogren and N. E. Leonard, “A convergent dynamic window approach to obstacle avoidance,” *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188–195, 2005.

[11] M. Seder and I. Petrovic, “Dynamic window based approach to mobile robot motion control in the presence of moving obstacles,” in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1986–1991, IEEE, 2007.

[12] W. Chung, S. Kim, M. Choi, J. Choi, H. Kim, C. b. Moon, and J. B. Song, “Safe navigation of a mobile robot considering visibility of environment,” *IEEE Transactions on Industrial Electronics*, vol. 56, pp. 3941–3950, Oct 2009.

[13] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[14] Z. Shiller, F. Large, and S. Sekhavat, “Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories,” in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4, pp. 3716–3721, IEEE, 2001.

[15] F. Belkhouche, “Reactive path planning in a dynamic environment,” *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 902–911, 2009.

[16] J. Van Den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1928–1935, IEEE, 2008.

[17] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, “Reciprocal n-body collision avoidance,” in *Robotics research*, pp. 3–19, Springer, 2011.

[18] E. Owen and L. Montano, “Motion planning in dynamic environments using the velocity space,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2833–2838, Aug 2005.

[19] E. Owen and L. Montano, “A robocentric motion planner for dynamic environments using the velocity space,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4368–4374, Oct 2006.

[20] T. Fraichard and H. Asama, “Inevitable collision statesa step towards safer robots?,” *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[21] T. Fraichard, “A short paper about motion safety,” in *IEEE int. conf. on robotics and automation*, 2007.

[22] S. Petti and T. Fraichard, “Safe motion planning in dynamic environments,” in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 2210–2215, IEEE, 2005.

[23] L. Martinez-Gomez and T. Fraichard, “Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation,” in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 100–105, IEEE, 2009.

[24] L. Martinez-Gomez and T. Fraichard, “An efficient and generic 2d inevitable collision state-checker,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 234–241, IEEE, 2008.

[25] O. Gal, Z. Shiller, and E. Rimon, “Efficient and safe on-line motion planning in dynamic environments,” in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 88–93, IEEE, 2009.

[26] Z. Shiller, O. Gal, T. Fraichard, *et al.*, “The nonlinear velocity obstacle revisited: The optimal time horizon,” in *Workshop on guaranteeing safe navigation in dynamic environments. In IEEE international conference on robotics and automation*, 2010.

[27] D. Althoff, M. Althoff, D. Wollherr, and M. Buss, “Probabilistic collision state checker for crowded environments,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1492–1498, IEEE, 2010.

[28] D. Althoff, J. J. Kuffner, D. Wollherr, and M. Buss, “Safety assessment of robot trajectories for navigation in uncertain and dynamic environments,” *Autonomous Robots*, vol. 32, no. 3, pp. 285–302, 2012.

[29] S. Bouraine, T. Fraichard, and H. Salhi, “Provably safe navigation for mobile robots with limited field-of-views in dynamic environments,” *Autonomous Robots*, vol. 32, no. 3, pp. 267–283, 2012.

[30] J. Hernandez-Aceituno, L. Acosta, and J. D. Pineiro, “Application of time dependent probabilistic collision state checkers in highly dynamic environments,” *PLoS one*, vol. 10, no. 3, p. e0119930, 2015.

[31] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, “Randomized kinodynamic motion planning with moving obstacles,” *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[32] J. P. Van Den Berg and M. H. Overmars, “Roadmap-based motion planning in dynamic environments,” *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 885–897, 2005.

[33] J. Van Den Berg, D. Ferguson, and J. Kuffner, “Anytime path planning and replanning in dynamic environments,” in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 2366–2371, IEEE, 2006.

[34] S. Quinlan and O. Khatib, “Elastic bands: Connecting path planning and control,” in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pp. 802–807, IEEE, 1993.

[35] C. Rosmann, W. Feiten, T. Wosch, F. Hoffmann, and T. Bertram, “Efficient trajectory optimization using a sparse model,” in *Mobile Robots (ECMR), 2013 European Conference on*, pp. 138–143, IEEE, 2013.

[36] C. Rosmann, F. Hoffmann, and T. Bertram, "Planning of multiple robot trajectories in distinctive topologies," in *Mobile Robots (ECMR), 2015 European Conference on*, pp. 1–6, IEEE, 2015.

[37] T. Mercy, W. Van Loock, and G. Pipeleers, "Real-time motion planning in the presence of moving obstacles," in *Control Conference (ECC), 2016 European*, pp. 1586–1591, IEEE, 2016.

[38] W. V. Loock, G. Pipeleers, and J. Swevers, "B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction," *Mechanical Sciences*, vol. 6, no. 2, pp. 163–171, 2015.

[39] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Transactions on Control Systems Technology*, 2017.

[40] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.

[41] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous robots*, vol. 13, no. 3, pp. 207–222, 2002.

[42] A. V. Savkin and C. Wang, "A simple biologically inspired algorithm for collision-free navigation of a unicycle-like robot in dynamic environments with moving obstacles," *Robotica*, vol. 31, no. 6, pp. 993–1001, 2013.

[43] R. Bischoff, U. Huggenberger, and E. Prassler, "Kuka youbot-a mobile manipulator for research and education," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, IEEE, 2011.

[44] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[45] N. P. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator.," in *IROS*, vol. 4, pp. 2149–2154, Citeseer, 2004.

[46] D. Lu, "Global planner ros." `http://wiki.ros.org/global_planner`.

[47] E. Marder-Eppstein, "Ros navigation." `http://wiki.ros.org/navigation`.

[48] B. P. Gerkey, "Amcl ros." `http://wiki.ros.org/amcl`.

[49] "Animated box." `http://gazebosim.org/tutorials?cat=build_robot&tut=animated_box`.

[50] "Moving obstacle gazebo plugins." `https://github.com/DharminB/moving_obstacle_gazebo`.

[51] E. Marder-Eppstein, D. V. Lu, and D. Hershberger, "Costmap 2d." `http://wiki.ros.org/costmap_2d#Rate_parameters`.

[52] C. Rsmann, "Teb local planner." `http://wiki.ros.org/teb_local_planner`.

[53] "omg-tools." `https://github.com/meco-group/omg-tools`.

[54] "Ros compatible omg-tools motion planner." `https://github.com/DharminB/p3dx_motionplanner`.

[55] E. Marder-Eppstein, "Dwa local planner." `http://wiki.ros.org/dwa_local_planner`.

[56] C. Christian, M. Bhaskara, and K. Piyush, "Eband local planner." `http://wiki.ros.org/eband_local_planner`.