PERCEIVING GUARANTEED COLLISION-FREE ROBOT TRAJECTORIES IN UNKNOWN AND UNPREDICTABLE ENVIRONMENTS

by

Rayomand Vatcha

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2012

Approved by:

Dr. Jing Xiao

Dr. Srinivas Akella

Dr. Min Shin

Dr. Barry Wilkinson

Dr. Jiang Xie

ABSTRACT

RAYOMAND VATCHA. Perceiving guaranteed collision-free robot trajectories in unknown and unpredictable environments. (Under the direction of DR. JING XIAO)

The dissertation introduces novel approaches for solving a fundamental problem: detecting a collision-free robot trajectory based on sensing in real-world environments that are mostly unknown and unpredictable, i.e., obstacle geometries and their motions are unknown. Such a collision-free trajectory must provide a guarantee of safe robot motion by accounting for robot motion uncertainty and obstacle motion uncertainty. Further, as simultaneous planning and execution of robot motion is required to navigate in such environments, the collision-free trajectory must be detected in real-time.

Two novel concepts: (a) dynamic envelopes and (b) atomic obstacles, are introduced to perceive if a robot at a configuration $\mathbf{q}$, at a future time t, i.e., at a point $\chi = (\mathbf{q}, t)$ in the robot's configuration-time space (CT space), will be collision-free or not, based on sensor data generated at each sensing moment τ, in real-time. A dynamic envelope detects a collision-free region in the CT space in spite of unknown motions of obstacles. Atomic obstacles are used to represent perceived unknown obstacles in the environment at each sensing moment. The robot motion uncertainty is modeled by considering that a robot actually moves in a certain tunnel of a desired trajectory in its CT space. An approach based on dynamic envelopes is presented for detecting if a continuous tunnel of trajectories are guaranteed collision-free in an unpredictable environment, where obstacle motions are unknown. An efficient

collision-checker is also developed that can perform fast real-time collision detection between a dynamic envelope and a large number of atomic obstacles in an unknown environment. The effectiveness of these methods is tested for different robots using both simulations and real-world experiments.

ACKNOWLEDGEMENTS

I am greatly indebted to my advisor, Prof. Jing Xiao, for her patient guidance, advice, and encouragement. Some results in this dissertation would not have been possible without her guidance and feedback. I gratefully acknowledge all the members of my committee who have given their time to read this manuscript and provided valuable advice: Prof. Srinivas Akella, Prof. Min Shin, Prof. Barry Wilkinson, and Prof. Jiang Xie.

I would also like to thank my parents, Soli Minocher Vatcha and Bakhtawar Soli Vatcha, and my sister, Rashna Vatcha, for their love and support.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

One of the ultimate goals in robotics is to enable robots to work autonomously in real-world environments. Meeting this goal requires the robot to move intelligently in environments without colliding with any obstacles, such as, chairs, tables, people, etc. Environments can be classified as follows:

- Static environment: No obstacles can move in the environment.
- Dynamic environment: Some or all obstacles can move in the environment.

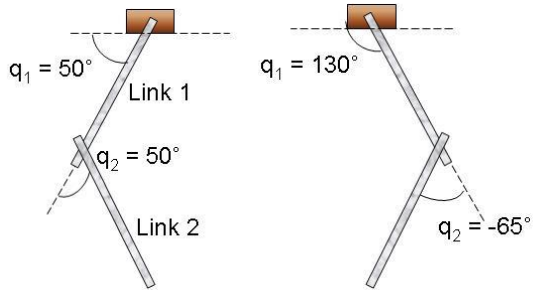
This dissertation introduces approaches for detecting *guaranteed* collision-free robot motions in dynamic environments with obstacle geometries and their future motions unknown. This enables the robot to move autonomously and safely in such environments. Finding such intelligent motions for a robot in an environment with obstacles is the task of a *planner*, which has been a central theme of robotics research. In the following, basic concepts about robots and environments are first introduced, and then the classical planners are surveyed.

1.1 Basic tool: C-Space and CT-Space

Planning can be defined by introducing Configuration Space (C-space) for a static environment, and Configuration-Time space (CT-Space) for a dynamic environment.

The C-space of a robot can be defined by using the following terms [16, 65]:

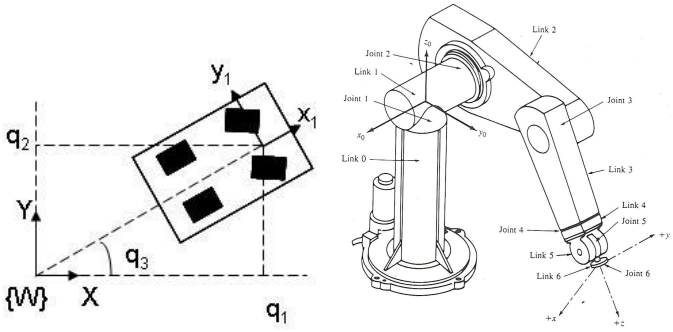
1. Configuration: A vector of independent variables **q** that defines uniquely the position of every point of a robot in a real world environment or *physical space*. Figure 1 shows a robot arm with two joints expressed by vector $[q_1, q_2]^T$ in two different configurations.



(a) (b)

Figure 1: A planar rod robot with two links and two joint variables $[q_1, q_2]^T$.

2. Degrees of freedom (DOF): The number of independent variables that uniquely define the configuration of the robot. For example (Figure 2), a mobile robot has 2 to 4 DOF, an industrial manipulator has 5 to 9 DOF, a humanoid has 29 DOF, etc.



(a) Mobile robot with 3 DOF. (b) Puma manipulator with 5 DOF.

Figure 2: Robots with different degrees of freedom (DOF).

The *configuration space* (C-space) [62] of a robot has dimensions equal to degrees of freedom of the robot. The complex volume occupied by a robot at a configuration in physical space is represented as a point in the C-space [88]. Each point in the C-space is classified either as a *C-obstacle* point if an obstacle intersects with the robot at that configuration or as a *C-free* point, otherwise.

If obstacle *poses* i.e., their positions or orientations, change with time then the C-space also changes with time, i.e., a C-point which was a C-obstacle point may now become a C-free point and vice-versa. Planning in such a space, which changes when an obstacle moves, is difficult. Thus, the notion of *configuration-time space* (CT-space) is introduced by adding one more dimension of time to the configuration space; each CT-slice at a time instant of a CT-space corresponds to a C-space at that time instant. A CT-point $\chi = (\mathbf{q}, t)$ corresponds to a robot at a particular configuration $\mathbf{q}$ at a particular time t in the physical space. The C-obstacles on all the CT-slices define *CT-obstacles* in the CT-space, and the remaining space is the *CT-free* space. Figure 3 shows an example of CT-space of a 2 DOF robot and a collision-free motion segment (trajectory) that connects the starting configuration S to the ending configuration G.

The notions of paths and trajectories for robot motion are defined below:

- Path: A 1D curve segment in the C-space of the robot representing the sequence of configurations of the robot.

- Trajectory: A 1D curve segment in the CT-space of the robot representing the sequence of poses that the robot will be at different time.

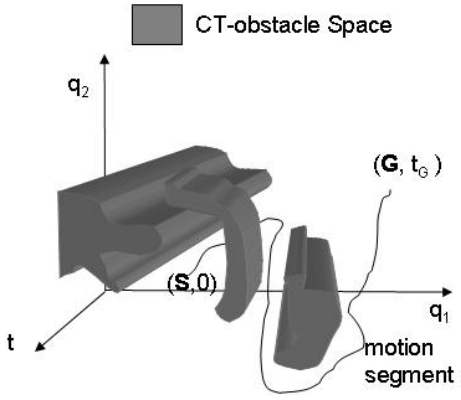


Figure 3: CT-space of a 2 DOF robot

A *planner* tries to find a sequence of motions for the robot to reach the *goal* configuration from the *start* or the current configuration, without colliding with any obstacles. This sequence of motions is a (a) *collision-free path* in the C-free space of the robot, if the environment is static, or (b) *collision-free trajectory* in the CT-free space of the robot, if the environment is dynamic. Figure 3 shows a collision-free trajectory (motion segment) in the CT-space of the robot that connects the starting configuration S at time $t = 0$ to the ending configuration G at time $t = t_G$.

1.2 Different kinds of real-world environments

Robots, working in a real-world environment, must have complete or partial information about obstacles or available free-space in that environment. Depending on whether an environment is static or dynamic and the level of information available to the robot, environments can be classified into different categories.

1.2.1 Known environment

A known environment is usually man-made, where the information about it is *completely* known. Such an environment can be static or dynamic as discussed below:

- Static environment: In a man-made static environment, precise geometries of obstacles can be known. For example, industrial environments, such as, a manufacturing site, consists of obstacles, such as, machines, parts, etc. with their geometric models usually known. Thus, most required information about the physical environment for robot motion planning is available.

- Dynamic environment: In a known dynamic environment, future motions of obstacles are precisely known along-with their geometries. For example, multiple robots working in the same environment can consider other robots as obstacles with known motions. Such a kind of environment is commonly found in industrial environments, where robots try to manipulate some objects in the environment. Thus, most information about environment for planning is available.

1.2.2 Unknown static environment

An unknown static environment is where objects are unknown but nothing moves, such as some ancient ruins or the surface of Mars. In such an environment, models of obstacles are not known, which further complicates detection of obstacles as particular objects in real-world. Thus, the required information about environment for planning robot motion has to be gained from sensing.

1.2.3 Partially unknown dynamic environment

A partially unknown dynamic environment contains obstacles, whose geometries are known but their motions are unknown. Moreover, obstacles are often constrained to move only within a specific region due to the nature of environment or behaviors of

obstacles themselves. For example, on highways, where the roads are clearly divided by lanes, vehicles can move along only one side of the road and possibly within their lanes for some time period; also, the geometries of vehicles are known as they are manufactured by various vehicle manufacturers.

Since the possible obstacle motions could be known based on the constraints on their motions, their future motions can be predicted by tracking their past motions. Sensing is required to detect obstacles for planning robot motion in such an environment.

1.2.4 Unknown and unpredictable environment

An unknown and unpredictable environment is where obstacles are completely unknown and their motions are unknown or hard to predict. For example, people moving in shopping malls or offices or at home. Moreover, in such environments, not all possible obstacles that a robot is likely to encounter can be known. Figure 4 shows an example. Since possible future obstacle motions are difficult to know, their future

Figure 4: A place with many people walking.

motions can be wrongly predicted. Thus, the information about an environment, obtained through obstacle recognition and motion prediction, may not work here. New ways are called upon to acquire information.

1.3 Planning robot motion

Planners in the robotics literature can be categorized based on the kinds of environments in which robot motions are planned and are described here.

1.3.1 Path planning

If the environment is static, a planner needs to find a path, connecting the start pose of the robot to the end pose, in the C-free space of the robot. Such kind of planning is called *path planning*. The primary focus of the planner is not only to find the existence of a collision-free path, but also to find the best path depending on optimization parameters. For finding the existence of a collision-free path there are approaches for 2D or 3D C-space that use C-obstacle features, such as, vertices of polygonal C-obstacles [63], edges of polygonal C-obstacles [47], etc. If C-obstacle geometries are not known, then approximate cell-decomposition [53] is used to compute the approximate geometry of C-free space in the C-space. The commonly used optimization parameters are shortest path, minimal robot energy, minimal time, etc.

Computing C-obstacles is not only difficult [34,93], but also infeasible if the C-space is high dimensional for a robot with high-DOF, such as, manipulators, humanoids, etc. Sampling based approaches [46,55,56] have been used extensively to avoid computing C-obstacles. These approaches randomly find a set of collision-free C-points (nodes) and then locally try to find collision-free paths that connect the nearby nodes. The

randomness in picking nodes is primarily used to ensure good coverage of C-space. Testing if a C-point is a part of any C-obstacle just requires collision-checking, which is described later in the next section. However, such planners may not be able to find a solution if one exists; increasing sampling of C-points increases the probability of finding a solution and thus, such planners are probabilistically complete. One representative planner is the probabilistic roadmap method [46] and there exists a well-studied literature on sampling schemes [2, 59, 71] for representing free-space in the C-space of a robot.

1.3.2 Reactive planning

In certain environments, only some obstacles may be dynamic among other static obstacles. Then, planning can be done in the robot's C-space as described in the above section; for some dynamic obstacles, a found collision-free path among static obstacles can be modified locally as the robot starts executing that path [45, 103]. Such local planning, where the planner just locally reacts to the obstacles is known as reactive planning. Reactive planning is computationally less expensive than path planning or motion planning (described in next section) that uses global information about an environment, i.e., account for the presence of all the obstacles. Thus, a reactive planner can easily generate a local plan in real-time; however, it cannot direct the robot to reach the goal without the help of a path planner and can often get the robot stuck in local minima.

1.3.3 Motion planning

If an environment is dynamic, a planner needs to find a trajectory connecting the starting configuration-time to the ending configuration-time in the CT-free space of the robot, and that is called motion planning.

If the future motion of an obstacle is known for a long time period, then a CT-obstacle corresponds the swept volume over time of a C-obstacle. The approaches [47, 63] mentioned in section 1.3.1 can be used when C-obstacles can be computed. However, finding the swept volume of a complex obstacle is difficult. Moreover, computing C-obstacles becomes difficult or even infeasible as the DOF of robot increases. Similar to path planning, sampling-based approaches [46, 55] could be used here except that the planning is done in the CT-space of the robot.

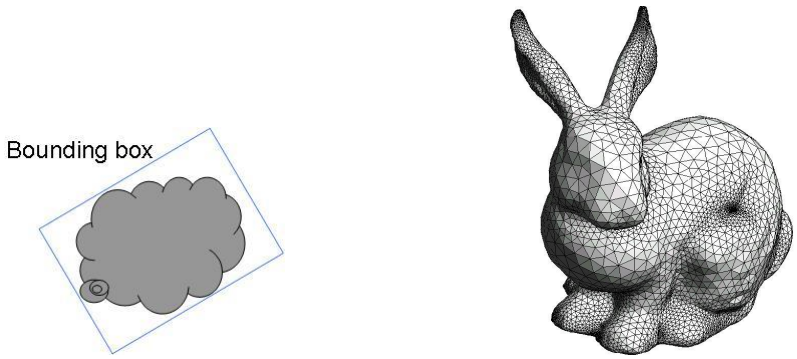
If the future motions of obstacles are not known beforehand, then it can be estimated or predicted so that the planner is able to find a solution from starting configuration to ending configuration. In general, the estimation or prediction of future motions of obstacles are usually true only for a short period immediate after the time when the prediction is made. This causes the estimated CT-space of the robot to keep on changing over time as obstacles behave differently from their predicted behaviors. This makes planning more difficult as a detected collision-free trajectory may not really be collision-free. A common approach is to re-plan those parts of trajectories that now lie on CT-obstacles (e.g., [92, 108]).

1.4 Collision-checking

Computing C-obstacles or CT-obstacles is difficult and can be infeasible if the complexity of physical obstacle geometry or the DOF of the robot is high. As discussed in section 1.3.1, sampling based planners could be used here as they only require to know if a C-point/CT-points in C-space/CT-space is collision-free or not. The following two common approaches, depending on whether the obstacle geometry is known or not, can easily compute such information in 3D physical space of the robot:

- Model based checking: When the 3D obstacle geometry is known, the obstacle can be modeled using (a) simple bounding volumes such as object oriented bounding boxes (OBBs), spheres, capsules, etc, or (b) a set of polygonal faces (called mesh) such as triangles, rectangles, etc. Figure 5 shows an example. There exist well-studied literature [15,61] in the field of computer graphics that can efficiently detect the existence of intersection between simple bounding volumes or meshes. For example, the separation axis method uses convex property of polyhedral objects to determine quickly if an intersection exists or not between them. For intersection checking between meshes, hierarchal checking is often used, where a complex mesh is represented by simple bounding volumes.

- Occupancy in physical space: If the obstacle geometry is not known then the physical space can be divided into voxels, which are small cubes; and whether the voxel is obstacle free or not can be determined by sensing (e.g. [104]). A robot at a particular configuration occupies a set of voxels in the physical space, which can be determined as the geometry of robot is known. One approach [58]



(a) Bounding box roughly approximates the obstacle.

(b) Mesh closely approximates the obstacle (Stanford bunny).

Figure 5: Approximating real obstacle geometry.

computes off-line the mapping between a set of C-points and its corresponding set of voxels in the physical space. Although such collision-checking may be faster than the model based checking, it restrains the planner to use only those C-points for which mapping has been computed.

CHAPTER 2: LITERATURE SURVEY

As this dissertation is focused on tackling the largely open problem of how to detect collision-free robot trajectories in unknown and unpredictable environments, it is important to first examine existing approaches related to detecting collision-free robot trajectories. These approaches often assume that much information about an environment is known. Related literature includes methods for continuous collision checking, approaches for acquiring information of an environment through sensing, and approaches in control focusing on handling robot motion uncertainty.

2.1 Assumptions made about dynamic environments for robot motion

Geometries of obstacles and their future motions are known: As the geometry, pose, and motion of any obstacle is completely known, planning can be done off-line using a model of the environment without the need of sensing. Sensing is used only during actual execution of a robot's motion to deal with uncertainties (e.g., [51,72,84]). Some approaches are focused on finding collision-free regions or free space in the C-space [53, 98]. If the robot has high degrees of freedom (DOF), such as, an articulated robot, the corresponding C-space and CT-space are high-dimensional. To avoid construction of high-dimensional C-obstacles (or CT-obstacles), sampling-based planners are widely used [46, 55].

Geometries of obstacles are known and their precise velocities within some time inter-

val can be computed: Here one common focus is on-line revising pre-planned paths in previously known environments to avoid robot collisions with newly added recognizable obstacles, which could be static (e.g., [58]) or performing particular kinds of motions (e.g., [103, 108]). These schemes usually assume partial changes to known C-space or CT-space to limit the scale of re-computing or re-planning for facilitating real-time computation. There is also work on motion planning to avoid static or moving obstacles with certain known velocities within some time interval [25, 52].

Only the geometries of obstacles are known: Here the obstacles are assumed known (or recognizable), but with unknown future trajectories. There are a few planners (e.g., [20, 35, 50, 89]) that address mobile-robot motion planning in such an environment. A real-time adaptive motion planning (RAMP) approach [91, 92] is very effective for planning high-DOF robot motion, characterized by simultaneous planning and execution based on sensing.

Most approaches commonly predict future obstacle trajectories by tracking the past motions of the known obstacles (e.g. [12, 21–23, 27, 32, 36]). However, the prediction is usually true only for a short period, i.e., immediately after the time when the prediction is made. As the observed behavior of an obstacle changes, the prediction of the obstacle's future trajectory changes. Thus, planning future robot's motion is based on frequently updating the predictions, and unknown changes in an environment are taken into account by repeated computations or re-computations of (some parts of) paths/trajectories, which involve repeated collision checking. Moreover, the planned motions may also fail to be collision-free due to inaccurately predicted obstacle motions. There exists work aimed at guaranteeing collision-free motions for mobile

robots. The notion of "Inevitable Collision Regions" (ICS) was introduced [28] for a mobile robot to characterize guaranteed CT-obstacles in its CT-space. Further, there has also been some approaches introduced in [29] that guarantee motion safety of a robot by assuming specific conditions, such as, the robot observes unicycle dynamics, obstacles are also robots, etc.

Geometries of obstacles are known and all possible motions of obstacles are taken into account: In [90], prediction of unknown motions of known obstacles is avoided by considering all the possible future motions of the obstacles, i.e., considering worst-case CT-obstacles so that the planned motions are guaranteed collision-free. However, the estimated free CT space is too conservative (i.e., too small due to exaggerated CT-obstacle regions) and increasingly so with time, as possible poses of obstacles increase with time. Also, in [101], the precise reachable motions of circular disc obstacles, observing unicycle dynamics, are computed to guarantee safety for infinite time horizon.

2.2 Extracting information from sensing in unknown environments

Planning requires to know information about environment, such as, geometries of obstacles and their future motions. If obstacle geometries and their motions are unknown, then sensor(s) can be used by the robot to gain information about the environment. However, the kind of sensor data generated by real sensors are very rudimentary and do not directly provide high-level information of object models. There exists a vast literature that addresses this problem of inferring information about an environment from sensor data and are discussed in this section.

2.2.1 Acquiring information of obstacles

Any moment, the planner needs to know the current pose occupied by the obstacles or the available free space. For dynamic environments, the planner additionally needs to know how the obstacles poses change with time or the way the available free-space changes with time. There are mainly two approaches that try to infer information about the occupancy of obstacles in an environment:

Finding obstacles: From sensor data, if an obstacle in the environment can be identified using its features, then based on poses of features along with the known geometry of obstacles, the occupancy of that obstacle in the physical space can be obtained. Two most common approaches for 3D modeling are (a) 3D modeling tool used while designing that object (obstacle), and (b) scanning real-world object using different kind of sensors (e.g. [24,38]), such as, laser scanner, stereo-vision, camera, etc. However, identifying obstacles from sensor data itself is an active research area as each obstacle has different kind of features and complexities. For example, in computer vision, a major focus of research is to identify specific kinds of objects, such as, human [4, 33], vegetation [7], objects that can cause a mobile robot to drop off [69], internal organs or parts in human bodies for medical analysis [44], etc. There exist common methods [73, 79, 100], especially from machine learning [48, 107], that can be used to detect various kinds of objects. Such methods often require time consuming training process, but the training could be done off-line and a mapping function, which identifies a specific object or object class from other objects, determined as a result of training, could identify that object instantaneously. The major issue with

such approaches is the lack of guarantee to have high accuracy for identifying an object due to many factors, such as different lighting conditions, occlusions, etc.

Finding free physical space: There is much research on how to represent and sense an unknown (mostly static) environment using robots with sensors mounted. One approach represents an unknown static environment with unknown obstacle geometry in terms of voxels [104]. However, detecting all voxels occupied by the actual obstacles from sensing is not a trivial matter and may not be feasible in real-time. A large body of work is focused on simultaneous robot localization and mapping (SLAM) in an unknown static environment [86], which represents the environment using probability distributions.

There is also research addressing how to move a robot to maximize sensing views (i.e., minimize occlusions) [106]. For sensor-based robot navigation, different kinds of sensors are used, either mounted in the environment to provide a world view, or mounted on a robot to provide a robot-centric local view. The planners, referred to as sensor based motion planners [31, 105], are often adapted from classical model based planners to plan paths incrementally, as the unknown static environment becomes known gradually from sensing. However, for unknown dynamic environments with obstacle geometries not known, planning becomes difficult.

2.2.2 Future changes in an environment

For planning robot motion in a changing environment, the future information about that environment must be known. The future change in the environment can be estimated if the change in obstacle poses can be estimated, or dually the change in

available free-space in physical space can be estimated.

Predicting future obstacle motion based on its past motion or its current motion has been widely used in literature. The common motion model performed by any obstacle is well defined in the literature of Physics, such as Newton's law of motion, Newtonian dynamics, Lagrangian mechanics, etc. The kind of predictions commonly made in literature for planning robot motion are described below:

Short-term prediction: If the prediction is made based on only the current state of the obstacle, then the prediction about obstacle motion is accurate as long as the obstacle is not obstructed or the obstacle, itself, decides to change its motion. As no past information is known, the event of such occurrence, i.e., obstacle changing its motion, can not be known based on the current state of the obstacle. Thus, predictions are true usually for a short period of time. Such a kind of prediction [12, 70] is widely used for reactive planning [25, 52] (see Section 1.3.2).

Long-term prediction: Using the past information of an obstacle may enable predicting for longer period. One approach [27] considers using the notion that obstacles move in a way so as to achieve their point of interest. Other approaches [5, 13, 21, 91] try to find some repetitive pattern occurring by tracking obstacle motion all the time, for long-term prediction. Since, the precise future obstacle motion may not be known, multiple possible future trajectories [67] for each obstacles need to be considered for planning, which can increase collision-checking cost. Moreover, identifying a single obstacle in a complex environment itself is difficult as discussed in Section 2.2.1. Thus, tracking multiple obstacles [12, 21–23, 27, 32, 36] is not only difficult but the computational complexity grows as the number of obstacles increases in an environment.

2.3 Collision detection

Collision detection is usually the most time consuming component of any sampling-based robot motion planner. Many fast collision-checking algorithms [15, 42, 61] exist for detecting collisions between two arbitrary stationary objects represented by polygonal meshes or sphere trees. Recently, an efficient collision detection algorithm [60] is introduced for checking collisions between the exact model of a continuum manipulator and obstacles in polygonal meshes.

Checking for collisions of a robot path or trajectory is usually done by discretizing the path/trajectory and check for each discretized configuration, which may omit collisions with small obstacles, depending on the resolution of discretization. Therefore, there also exists some work that addresses continuous collision checking of a robot path/trajectory. Most of such work requires known obstacle motions. One approach formulates trajectories of the robot and approximated obstacles in the environment as functions of time and finds the time instants when collision occurs analytically [81]. However, if the trajectory functions are nonlinear, solving for collision time instants can be difficult. The adaptive bisection algorithm [80] is based on the intuition that if the sum of the distances traveled by two objects is less than the minimum distance between them before traveling, then they cannot collide during their motions.

There are also approaches in the literature that focus on generating or approximating the continuous swept volume by a robot along a path or trajectory [11, 26], which can then be used to perform collision tests against obstacles. One approach [76] models the motion between two discrete configurations of an articulated robot in order

to avoid generating the swept volume of individual links. Graphics hardware is then used to perform fast collision queries for approximated swept volumes in a virtual prototyping environment. Another approach [3] is focused on growing the physical robot's volume at discrete configurations along a path to form a continuous region for collision tests. These approaches are focused on the moving robot rather than the obstacles, which are mostly assumed static.

2.4 Handling motion uncertainty of robot in planning

Uncertainty in robot configuration while planning robot motion is well studied in the literature. Different kind of uncertainties resulting from different factors, such as stabilizing a non-holonomic system [9], slipping of wheels [19], etc. has been studied for a mobile robot. Further in [99], it has been shown by deriving equations specific to that robot, uncertainty in robot control can be easily handled. Such approaches are focused more on reducing uncertainty for specific robots than considering collision avoidance with obstacles.

The approach [77] relies on known environment features (map) to handle uncertainty in robot configuration; also, there exists approaches (e.g., [66]) to deal with uncertainity in a map. If the environment is unknown and static, there exists a body of literature [41,86] that simultaneously handles uncertainty in robot localization and mapping of an environment.

Some researchers addressed uncertainties in motion planning algorithm, e.g., [49, 54, 72]. The approach [30] introduces the notion of a robust path that guarantees a non-holonomic mobile robot to reach its goal based on landmarks.

2.5 Limitations

The following limitations hold for the existing methods in detecting collision-free robot trajectories in an uncertain, dynamic environment:

- The approaches based on predicting known obstacle motions may be too expensive due to the non-trivial process of obstacle identification and the repeated computations of their predicted motions, especially if there are many obstacles, which leads to repeated collision checking for detecting collision-free robot trajectories, which can be infeasible in real-time for high degrees-of-freedom (DOF) robots.

- The planned robot motions may fail to be collision-free, i.e., unsafe, if future obstacle motions are predicted inaccurately.

- Planning in an environment that is unknown and unpredictable, i.e., obstacle geometries are unknown and their future motions can not be predicted, is more challenging, and to the best of the author's knowledge, no current approach can tackle that.

These limitations motivate the proposed research and shed light on the value of contributions of this dissertation.

CHAPTER 3: RESEARCH OUTLINE

One of the fundamental abilities required by motion planners is to test if a motion performed by a robot will collide with any obstacles or not. The focus of the dissertation is to present novel approaches/algorithms that enable a motion planner to know if a future motion of the robot is collision-free or not based on sensing in an unknown and unpredictable environment.

3.1 About unknown and unpredictable environments

In an unknown and unpredictable environment all or some obstacles can move in any direction. Moreover, objects can appear or disappear from a real-world environment, become separate into smaller objects or combine into bigger ones, for example, in human centered environments, such as domestic environments, office environments, etc. Even for environments well known to humans, the number and variety of obstacles can be practically uncountable and not constant.

A robot operating in such an environment neither knows the possible kinds of obstacles (i.e. their geometries) nor the kinds of motions they could perform. How to enable a robot to move safely in this kind of environments is an open challenge.

3.2 Online determination of collision-free CT-points via sensing

A robot needs to actively sense to acquire information about its unknown environment. A sensor can sense only a part of a physical space, where the robot goal

configuration may not be visible. Many sensors could be used to sense the physical space, however, the amount of sensor data generated could be huge, which could make it difficult for planners to find collision-free motions in real-time.

The rate at which the information about the environment is gained, determines whether a planner can plan robot motions in real-time. Many sensors generate sensor data at the rate of 20Hz-80Hz. Within a sensing cycle, the planner must be able to detect multiple collision-free CT-points for decision making in real-time. Thus, how to quickly determine collision-free CT-points from sensor data is a major challenge, especially for high degree-of-freedom robots.

3.3 Detecting safe trajectories for a robot

For a robot to operate in a real-world environment, especially one involving humans, the robot must not hit any obstacles. While a simple solution is to make the robot stop its motion, it can get damaged by a moving obstacle. If a robot is set to execute a trajectory, the trajectory should not be merely collision-free, rather, it is better guaranteed collision-free in spite of unknown motions of obstacles.

Further, all the possible configurations that can be reached by the robot, which cannot execute a trajectory precisely, need to be guaranteed collision-free. This can be modeled as a certain "tunnel" of collision-free trajectories, which needs to be detected as *continuously* collision-free, i.e., every CT-point in that tunnel is collision-free. How to enable fast detection of a tunnel of guaranteed collision-free trajectories is a novel challenge that has not been addressed.

3.4 Outline

Our research contribution is to introduce approaches that, (a) do not require identifying individual obstacles, (b) do not require tracking to predict obstacle future motions, and (c) do not consider all possible motions of all obstacles for infinite time horizon but rather use progressive sensing to perceive collision-free high-DOF robot trajectories in real-time. Moreover, the detected trajectories will be guaranteed continuously collision-free and safe in spite of robot motion uncertainty.

The rest of the dissertation is outlined below. Chapter 4 introduces the notion of dynamic envelope for detecting a collision-free CT-region associated with a CT-point $\chi = (\mathbf{q}, t)$, under an assumption that the obstacle speeds range in $[0, v_{max}]$. As the robot cannot execute a trajectory Γ precisely, we assume that it will move within some continuous tunnel Γ^+, which describes motion uncertainty when the robot executes trajectory Γ. An algorithm is introduced in Chapter 5 to detect Γ^+ as collision-free using only a finite set of CT-points $Q(\Gamma^+)$, which when discovered guaranteed collision-free, implies that Γ^+ is guaranteed collision-free.

Chapter 6 introduces the notion of atomic obstacles to characterize the obstacles directly from sensor data for any sensing moment. The concept of atomic obstacles avoids identifying obstacles, considering that not all obstacles need to be identified and obstacle identification and tracking are often difficult or expensive to do. Chapter 7 introduces a fast real-time collision detection algorithm between a high DOF robot and a large number of atomic obstacles.

Chapter 8 merges the notion of dynamic envelopes, introduced in Chapter 4, with

atomic obstacles, introduced in Chapter 5, to detect collision-free CT-points via a general online sensor-based algorithm called Collision Free Perceiver using Atomic obstacles (CFPA). Next, we focus our attention in embedding CFPA into a real-time motion planner, taking into account the finite processing time that CFPA requires, in Chapter 9. Chapter 10 demonstrates experiments and results for the approaches. Chapter 11 concludes the dissertation and discusses directions for future work.

CHAPTER 4: DYNAMIC ENVELOPE

One of the important factors that facilitates real-time motion planning of high-DOF robots is the ease of knowing if the robot can stay safely at a configuration $\mathbf{q}$ for given future time t. Dynamic envelope is a novel concept that allows a robot to know if a configuration-time point (CT-point) $\chi = (\mathbf{q}, t)$ is guaranteed collision-free or not, without knowing future motions of obstacles. Further, it can also be used to detect a collision-free continuous CT-region that is in a neighborhood of CT-point χ. For some future time t, dynamic envelope is defined based on the robot model, v_{max} assumption about the environment, and a sensing moment τ.

4.1 Robot model

We assume that a high-DOF robot consists of multiple polyhedral links. This is a reasonable assumption since a real robot's link is commonly modeled by a polygonal mesh, or can be approximated by a polyhedral bounding box.

The following notations describe such a robot model in the Cartesian space $\mathbb{R}^3$ (physical space).

- l: the set of points constituting a polyhedral link (rigid body) of the robot.
- l_x: the set of vertices of l.
- $R(\mathbf{q})$: the set of all points of all links of the robot at configuration $\mathbf{q}$, i.e.

$R(\mathbf{q}) = \bigcup l(\mathbf{q})$.

- $R_x(\mathbf{q})$: the set of all bounding vertices of all links of the robot at configuration $\mathbf{q}$, i.e. $R_x(\mathbf{q}) = \bigcup l_x(\mathbf{q})$.

- $\mathbf{p}(\mathbf{q})$: the position of a point $p \in R$ in the Cartesian space when the robot is at configuration $\mathbf{q}$.

- $\mathbf{p}_x(\mathbf{q})$: the position of a point $p_x \in R_x$ in the Cartesian space, when the robot is at configuration $\mathbf{q}$.

4.2 v_{max} assumption about the environment

For an unpredictable environment, we assume an upper bound on the maximum possible linear speed v_{max} of any obstacle. The value of v_{max} can be obtained for different real world environments. For example, in a city environment where people and cars move, v_{max} can be determined based on the speed limit on a vehicle, and in a pedestrian only area, v_{max} can be set as the maximum walking/running speed of a person. Any obstacle may have varied actual speeds in $[0, v_{max}]$

4.3 Sensing instant τ

At every sensing moment $\tau_k, k > 0$ the sensor generates new rudimentary sensor data. For motion planning, within a sensing cycle $[\tau_{k-1}, \tau_k]$ the information about the poses of obstacles in the environment need to computed from the sensor data. We use the following notation to indicate obstacles sensed at time τ in the robot's physical space:

- $O(\tau)$: the union of regions occupied by obstacles in $\mathbb{R}^3$ at time τ.

4.4 Definition and its properties

Definition 1: For a robot's configuration-time (CT) point $\chi = (\mathbf{q}, t)$, a *dynamic envelope* $E(\chi, \tau)$, as a function of current sensing time $\tau \leq t$, is a closed surface enclosing the region $R(\mathbf{q})$ occupied by the robot at configuration $\mathbf{q}$ in the physical space $\mathbb{R}^3$ (or $\mathbb{R}^2$ for 2-D planar space) so that the minimum distance between any point on $E(\chi, \tau)$ and the region $R(\mathbf{q})$ is

$$d(t, \tau) = v_{max}(t - \tau) \tag{1}$$

Theorem 1: For a CT-point $\chi = (\mathbf{q}, t)$, if $E(\chi, \tau) \cap O(\tau) = \emptyset$ at sensing time $\tau < t$, then χ is detected as guaranteed collision-free, i.e., no obstacle will hit the robot at χ no matter how obstacles move.

Proof:

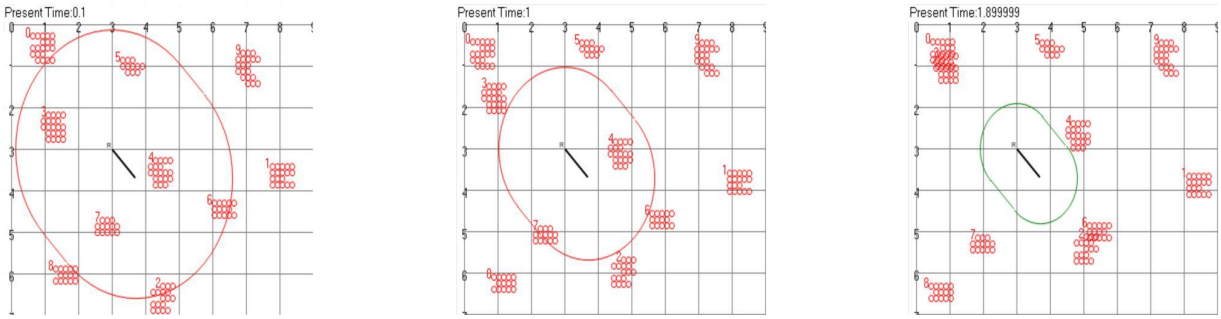
Since v_{max} is the upperbound of all actual obstacle speeds at any time, $E(\chi, \tau)$ has the following general properties:

1. It shrinks monotonically over sensing time with speed v_{max}, i.e., $E(\chi, \tau_{k+1}) \subset E(\chi, \tau_k)$, where $i > 0$, $\tau_k < \tau_{k+1} \leq t$. $E(\chi, \tau)$ shrinks to $R(\mathbf{q})$ at t.

2. An obstacle not on or inside $E(\chi, \tau_k)$ will never be on or inside $E(\chi, \tau_{k+1})$.

3. An obstacle either on or inside $E(\chi, \tau_k)$ can be outside $E(\chi, \tau_l)$ for some $\tau_l \in (\tau_k, t]$, if not moving *towards* $R(\mathbf{q})$ in maximum speed v_{max}.

Hence, at any time τ_k, if no obstacle is on or inside the dynamic envelope $E(\chi, \tau_k)$, i.e., $E(\chi, \tau_k) \cap O(\tau_k) = \emptyset$, then, based on property 2 above, $E(\chi, \tau_l) \cap O(\tau_l) = \emptyset, \tau_l \in [\tau_i, t]$,



(a) $E(\chi, 0.1)$ contains obstacles. (b) $E(\chi, 1)$ shrinks and squeezes out obstacles. (c) $E(\chi, 1.89)$ is free of obstacles.

Figure 6: A dynamic envelope of a planar rod robot.

and χ is guaranteed collision-free. ∎

Figure 6(a) shows an example of a dynamic envelope for a planar rod robot in 2-D environment, where $\chi = ((3, 3), 3)$, τ_i =0.1s and $v_{max} = 1$ unit/s. At $\tau = 1.89$s, χ is detected guaranteed collision free.

As soon as a dynamic envelope $E(\chi, \tau)$ is free of obstacles at a sensing instant τ_l, it has achieved the purpose of detecting that χ is collision-free. Thus it is no longer needed and can *expire* at τ_l. If a dynamic envelope is first used at time τ_0 and expires later at τ_l, we call $[\tau_0, \tau_l]$ its *life span*.

As $R(\mathbf{q}) \subset E(\chi, \tau_l)$, more neighboring CT-points are discovered collision-free:

- Time-wise: Since no obstacle can be inside dynamic envelope within time interval $[\tau_l, t]$, all the continuous configuration-time points within that time interval for configuration $\mathbf{q}$ are guaranteed collision-free.

- Space-wise: For other neighboring configuration $\mathbf{q}'$, we could find a dynamic envelope $E(\chi', \tau_l)$, such that $E(\chi', \tau_l) \subset E(\chi, \tau_l)$ (see Figure 8), where $\chi' = (\mathbf{q}', t')$, then χ' is also guaranteed collision-free. Finding such CT-points are

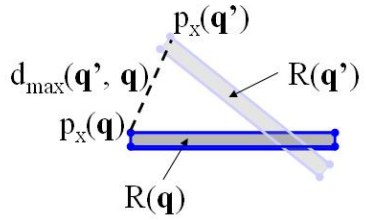


Figure 7: Illustration of $d_{max}(\mathbf{q}', \mathbf{q})$ of a rod robot.

shown in the next section.

4.5 Collision-free CT-region discovered along with a CT-point

As a dynamic envelope of a CT-point always contains $R(\mathbf{q})$, it is interesting to find an another configuration $\mathbf{q}'$, so that the dynamic envelope of that CT-point $\chi = (\mathbf{q}, t)$ contains $R(\mathbf{q}')$ within certain lifespan of the dynamic envelope. The distance $d_{max}(\mathbf{q}', \mathbf{q})$, defined as,

$$d_{max}(\mathbf{q}', \mathbf{q}) = \max_{\forall p_x \in R_x} \|\mathbf{p}_x(\mathbf{q}') - \mathbf{p}_x(\mathbf{q})\|. \tag{2}$$

is useful for this purpose. Figure 7 illustrates $d_{max}(\mathbf{q}', \mathbf{q})$ of a rod robot (which can also be considered a rectangular link of a high-DOF robot).

We have the following theorem.

Theorem 2: When a CT-point $\chi = (\mathbf{q}, t)$ is discovered collision-free at a sensing time τ_l, i.e., the dynamic envelope $E(\chi, \tau_l)$ is free of obstacles, a continuous neighborhood $F(\chi, \tau_l)$ of χ is also discovered collision-free, such that, for any CT-point $\chi' = (\mathbf{q}', t') \in F(\chi, \tau_l)$, its configuration $\mathbf{q}'$ satisfies:

$$d_{max}(\mathbf{q}', \mathbf{q}) \leq v_{max}(t - \tau_l), \tag{3}$$

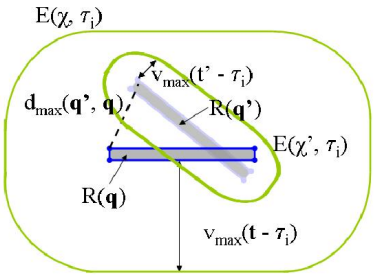


Figure 8: Illustration of inequality (5).

and its time t' satisfies

$$\tau_l \leq t' \leq t - \frac{d_{max}(\mathbf{q}', \mathbf{q})}{v_{max}} < t. \tag{4}$$

Proof: When inequality (3) is satisfied, $R(\mathbf{q}')$ is contained by the dynamic envelope $E(\chi, \tau_l)$, which is free of obstacles. If $\chi' = (\mathbf{q}', t')$ is collision-free, it means that the dynamic envelope of χ' is contained in the dynamic envelope of χ, i.e., $E(\chi', \tau_l) \subset E(\chi, \tau_l)$, and $\tau_l \leq t'$. Since $d_{max}(\mathbf{q}', \mathbf{q})$ is the maximum distance between two corresponding points of $R(\mathbf{q}')$ and $R(\mathbf{q})$, we have

$$d_{max}(\mathbf{q}', \mathbf{q}) + v_{max}(t' - \tau_l) \leq v_{max}(t - \tau_l), \tag{5}$$

which can be simplified to

$$t' \leq t - \frac{d_{max}(\mathbf{q}', \mathbf{q})}{v_{max}}.$$

Thus, the theorem is proven. ∎

Figure 8 illustrates the proof for a rod robot.

Corollary 1: If $\chi' \in F(\chi, \tau_l)$ for some $\tau_l < t$, then $\chi' \in F(\chi, \tau_{l+m}), \tau_{l+m} \in [\tau_l, t'], m \geq 0$

Proof: From inequality (4), subtracting t', we have

$$0 \leq (t - t') - \frac{d_{max}(\mathbf{q}', \mathbf{q})}{v_{max}}$$

which, multiplying v_{max}, leads to:

$$d_{max}(\mathbf{q}', \mathbf{q}) \leq v_{max}(t - t')$$

From the above, because $\tau_{l+m} \leq t'$, we have

$$d_{max}(\mathbf{q}', \mathbf{q}) \leq v_{max}(t - \tau_{l+m}) \tag{6}$$

and also

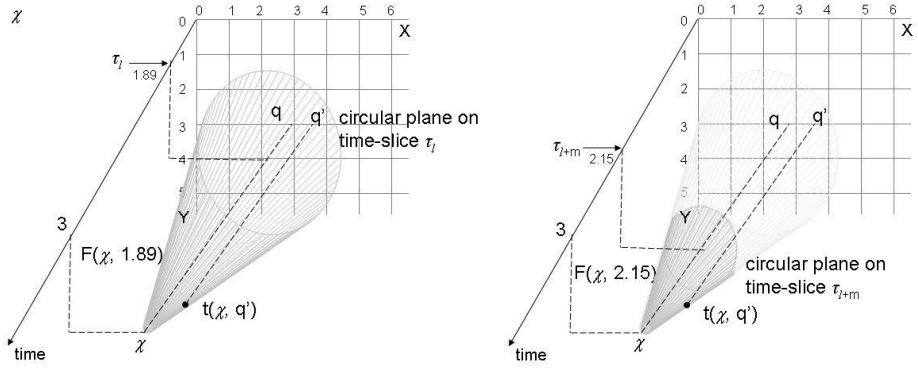
$$\tau_{l+m} \leq t' \leq t - \frac{d_{max}(\mathbf{q}', \mathbf{q})}{v_{max}} < t. \tag{7}$$

With (6) and (7), based on Theorem 2, the corollary holds. ∎

We next characterize the geometry of the CT-region $F(\chi, \tau_l)$ below, based on Theorem 2 and Corollary 1.

- $F(\chi, \tau_l)$ is on the time interval $[\tau_l, t]$.
- The region of $F(\chi, \tau_l)$ on the time-slice τ_l contains all the configurations that satisfy inequality (3), whose size and shape depend on (a) the robot kinematics, and (b) the size of the dynamic envelope $E(\chi, \tau_l)$.
- At the time-slice t, $F(\chi, \tau_l)$ contains the single CT-point $\chi = (\mathbf{q}, t)$.
- Based on Theorem 2, for $\tau_l \leq t' \leq t(\chi, \mathbf{q}')$, where

$$t(\chi, \mathbf{q}') = t - \frac{d_{max}(\mathbf{q}', \mathbf{q})}{v_{max}} \tag{8}$$



(a) $F(\chi, 1.89)$ based on the dynamic envelope $E(\chi, 1.89)$ shown in Figure 6(c).

(b) $F(\chi, 2.15)$ based on the dynamic envelope $E(\chi, 2.15)$. $F(\chi, 2.15) \subset F(\chi, 1.89)$.

Figure 9: The geometry of CT-region $F(\chi, \tau_k)$ for the 2D rod robot.

all CT-points $\chi' = (\mathbf{q}', t')$ are in $F(\chi, \tau_l)$. Equation (8) shows that the smaller $d_{max}(\mathbf{q}', \mathbf{q})$ is, the longer is the hyper-line from $(\mathbf{q}', \tau_l)$ to $(\mathbf{q}', t(\chi, \mathbf{q}'))$ in $F(\chi, \tau_l)$.

- $d_{max}(\mathbf{q}', \mathbf{q})$ decreases with slope v_{max} as $t(\chi, \mathbf{q}')$ increases from τ_l to t as indicated by the following equation derived from (8).

$$d_{max}(\mathbf{q}', \mathbf{q}) = -v_{max} t(\chi, \mathbf{q}') + v_{max} t \tag{9}$$

- $F(\chi, \tau_{l+m}) = F(\chi, \tau_l) - F_p$, where $F_p = \{(\mathbf{q}', t') | (\mathbf{q}', t') \in F(\chi, \tau_l), t' < \tau_{l+m}\}$.

As an example, Figure 9 illustrates the geometry of the CT-region $F(\chi, \tau_l)$ of the same rod robot with no width as in Figure 6 with only two translational degrees of freedom. The region of $F(\chi, \tau_l)$ on the time slice τ_i is a circular disc.

Thus, a dynamic envelope not only discovers a collision-free CT-point but also its neighboring CT-region. Moreover, without assuming any future motions of obstacles, based on v_{max} assumption, the tool successfully discovers collision-free CT-points. Next, we show the effects of over exaggerated v_{max} assumption for our approach.

4.6 Robustness of approach over exaggerated v_{max}

As v_{max}, the maximum speed of an obstacle, is the only known or estimated parameter we assume in our approach dealing with an unpredictable environment, it is necessary to investigate how robust our approach of detecting collision-free CT-space points is if v_{max} is over-estimated as $v'_{max} > v_{max}$. The effect of such over-estimation can be stated in the following theorem.

Theorem 3: Let $v'_{max} = cv_{max}, c > 1$, and let $\chi = (\mathbf{q}, t)$ be a collision-free CT-point. Let $E(\chi, \tau)$ and $E'(\chi, \tau)$ be the dynamic envelopes defined by v_{max} and v'_{max} respectively. If $(\mathbf{q}, t)$ is detected collision-free at τ_l by $E(\chi, \tau_l)$, then, $(\mathbf{q}, t)$ will also be detected collision free by $E'(\chi, \tau'_l)$, such that $\tau_l < \tau'_l$ and $\tau'_l = \tau_l + (\frac{c-1}{c+p})(t - \tau_l) \leq t$, where $-1 \leq p \leq 1$.

Proof: Suppose at time τ_0, we start observing the dynamic envelopes $E(\chi, \tau_0)$ and $E'(\chi, \tau_0)$ with respect to v_{max} and v'_{max}, where, based on equation (1),

$$d(t, \tau_0) = v_{max}(t - \tau_0), \text{ and}$$

$$d'(t, \tau_0) = v'_{max}(t - \tau_0) = cv_{max}(t - \tau_0)$$

Clearly for any time $\tau_0 \leq \tau < t$, $E'(\chi, \tau)$ is larger than $E(\chi, \tau)$. Suppose further that at least one obstacle was on or inside $E(\chi, \tau_0)$, then it was also on or inside $E'(\chi, \tau_0)$.

Suppose at time τ_l, where $\tau_0 \leq \tau_l \leq t$, the dynamic envelope $E(\chi, \tau_l)$ has shrunk enough to just "squeeze out" obstacles and detected that the CT-point $(\mathbf{q}, t)$ is collision-free. Let $d_{min}(\mathbf{q}, \tau_l)$ denote the minimum distance between $R(\mathbf{q})$ and the obstacles. Thus,

$$d(t, \tau_l) = v_{max}(t - \tau_l) = d_{min}(\mathbf{q}, \tau_l) - \epsilon \tag{10}$$

where $\epsilon > 0$ is infinitesimally small. Clearly at τ_l, $E'(\chi, \tau_l)$ still has an obstacle because it is larger than $E(\chi, \tau_l)$.

However, according to Definition 1 and equation (1), $d(t,t) = cv_{max}(t-t) = 0$. Since $(\mathbf{q}, t)$ is a collision-free CT-point, it means that $E'(\chi, t)$ at sensing time t is free of obstacle. Since $E'(\chi, \tau)$ shrinks continuously as τ progresses towards t, there exists a moment τ_l', $\tau_l < \tau_l' \leq t$, when $E'(\chi, \tau_l')$ is free of obstacle and $(\mathbf{q}, t)$ is detected collision-free.

We now see how τ_l' is related to τ. Based on equation (1),

$$d'(t, \tau_l') = cv_{max}(t - \tau_l') = d_{min}(\mathbf{q}, \tau_l') - \epsilon. \tag{11}$$

Since obstacles never move with speed greater than v_{max}, from τ_l to τ_l', the change in minimum distance between $R(\mathbf{q})$ and obstacles can be expressed as:

$$d_{min}(\mathbf{q}, \tau_l') - d_{min}(\mathbf{q}, \tau_l) = pv_{max}(\tau_l' - \tau_l),\ -1 \leq p \leq 1. \tag{12}$$

From the equations (10), (11), and (17), we have

$$d'(t, \tau_l') - d(t, \tau) = pv_{max}(\tau_l' - \tau_l),\ -1 \leq p \leq 1 \tag{13}$$

From (10), (11), and (13), we can further obtain

$$\tau_l' - \tau_l = (\frac{c-1}{c+p})(t - \tau_l) \leq t - \tau_l \tag{14}$$

■

Based on Theorem 3, $\tau_l' = t$ corresponds to the worst case scenario where $p = -1$, meaning that the closest obstacle to $R(\mathbf{q})$ at τ originally inside $E(\chi, \tau_0)$ moves towards

$R(\mathbf{q})$ with v_{max} from τ to t, and in all other cases, $\tau_l' < t$.

The significance of the theorem is that, if a CT-point $(\mathbf{q}, t)$ is collision-free, then it will be detected as collision-free *no later than* time t no matter how badly the actual v_{max} is overestimated as v'_{max}. This shows the robustness of our approach.

4.7 Perceived CT-space

An important tool required for motion-planning of robot in a dynamic environment is the CT-space (see section 1.1) of the robot. In an unknown and unpredictable environment, as the motion of obstacles are unknown, a common way is to predict obstacles motion and the corresponding CT-space of the robot we call as *Predicted CT-space*; whereas, using our approach of dynamic envelope, which requires to know only poses of obstacles at a current sensing moment τ, the corresponding CT-space we call as *Perceived CT-space*.

The motion planner must plan the motion on true collision-free regions, or free space, in the CT-space of a robot. However, with the prediction mechanism, being true only for a short period of time, does not guarantee that the motions planned lie on true collision-free regions. Whereas, with dynamic envelopes true collision-free regions can be perceived. We illustrate this by an example.

Figure 10 compares predicted vs. perceived vs. actual CT-space in a 2-D example. Both predicted CT-space and perceived CT-space will change as sensing/time progresses. However, unlike predicted CT-space, where a point predicted collision-free may not be actually collision-free, the perceived CT-space consists of *actual* collision-free regions that can only grow over time and *uncertain* regions, which can be either

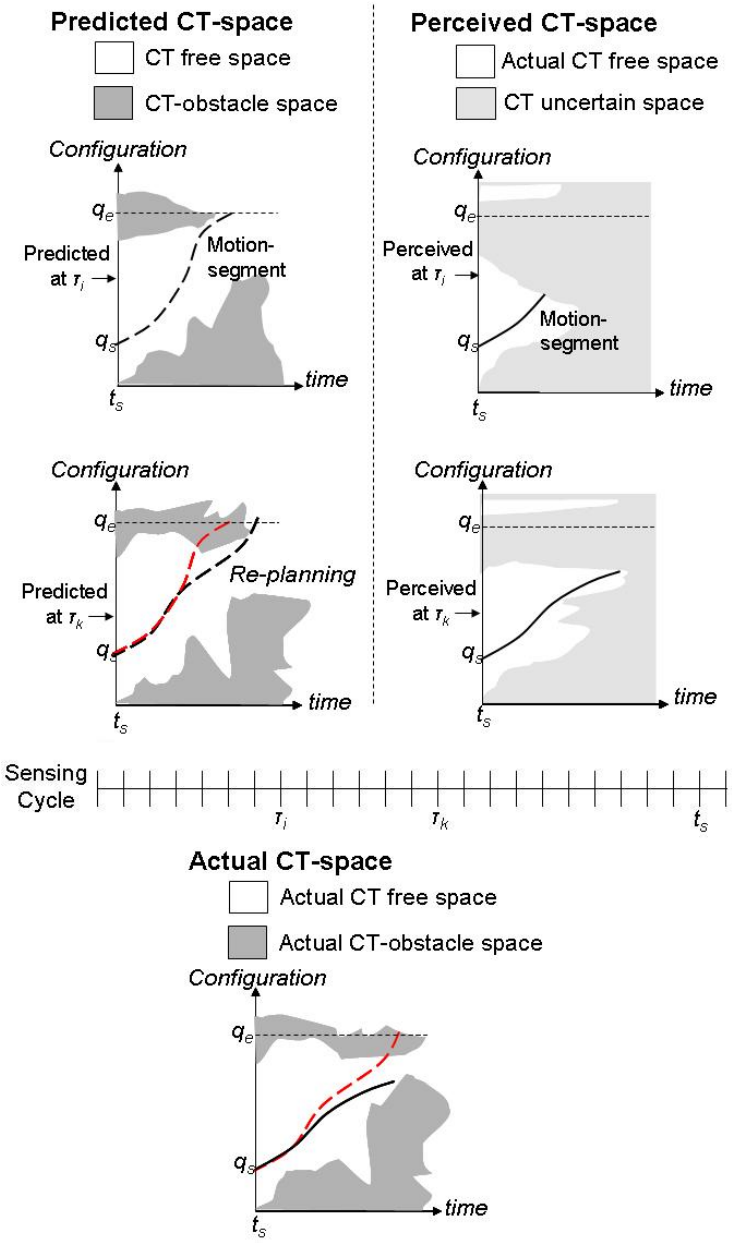


Figure 10: Predicted CT-space vs. Perceived CT-Space

free or CT-obstacle regions.

4.7.1 Collision-free region vs. uncertain region

A dynamic envelope answers if a CT-point $(\mathbf{q}, t)$ can be perceived at $\tau \leq t$ as guaranteed collision-free, and also by its property 2, if $(\mathbf{q}, t)$ is perceived at τ_k as guaranteed collision-free, then hyperline segment $[(\mathbf{q}, \tau_k), (\mathbf{q}, t)]$ in the CT-space is

also guaranteed collision-free.

Now a natural next question is: *given a configuration* $\mathbf{q}$*, what is the longest hyperline segment* $[(\mathbf{q}, \tau_k), (\mathbf{q}, t)]$*, or the furthest time* t*, that can be perceived at* τ_k *as guaranteed collision-free?* The answer to that question depends on the minimum distance $d_{min}(\mathbf{q}, \tau_k)$ between the robot (if it were) at configuration $\mathbf{q}$ and the closest obstacle sensed at τ_k. Let

$$\Delta t(\mathbf{q}, \tau_k) = \frac{d_{min}(\mathbf{q}, \tau_k)}{v_{max}}, \tag{15}$$

which is the minimum period before a collision can *possibly* occur at $\mathbf{q}$. Let

$$t_f(\mathbf{q}, \tau_k) = \tau_k + \Delta t(\mathbf{q}, \tau_k) \tag{16}$$

Clearly, as long as t is within the time interval $[\tau_k, t_f(\mathbf{q}, \tau_k))$, the hyperline segment $[(\mathbf{q}, \tau_k), (\mathbf{q}, t)]$ can be perceived at τ_k as guaranteed collision-free. Thus, the longest hyperline segment that can be perceived at τ_k as guaranteed collision-free is $[(\mathbf{q}, \tau_k), (\mathbf{q}, t_f(\mathbf{q}, \tau_k))$.

The union of all the guaranteed collision-free hyperline segments of the CT-space perceived at τ_k is the maximum collision-free region (that may include multiple connected continuous regions) perceived at τ_k, denoted as $F(\tau_k)$. $F(\tau_k)$ consists of only CT-points for $t \geq \tau_k$. The union of the rest of the regions in the CT-space for time $t \geq \tau_k$ forms the uncertain region $U(\tau_k)$.

Theorem 4: For any τ_k and $\tau_{k+m}, m > 0$, such that $\tau_k \leq \tau_{k+m}$, if a CT-point $(\mathbf{q}, t)$, where $t \geq \tau_{k+m}$, belongs to $F(\tau_k)$, then it also belongs to $F(\tau_{k+m})$. On the other hand, if the point $(\mathbf{q}, t)$ belongs to $U(\tau_k)$, it may still belong to $F(\tau_{k+m})$.

Proof: From τ_k to τ_{k+m}, the change in minimum distance at configuration $\mathbf{q}$ can be expressed as:

$$d_{min}(\mathbf{q}, \tau_{k+m}) - d_{min}(\mathbf{q}, \tau_k) = pv_{max}(\tau_{k+m} - \tau_k),\ -1 \leq p \leq 1. \tag{17}$$

From equations (15) and (16), and using equation (17), we get

$$t_f(\mathbf{q}, \tau_{k+m}) - t_f(\mathbf{q}, \tau_k) = (\tau_{k+m} - \tau_k) + \frac{d_{min}(\mathbf{q}, \tau_{k+m}) - d_{min}(\mathbf{q}, \tau_k)}{v_{max}} = (1+p)(\tau_{k+m} - \tau_k)$$

$$\Rightarrow t_f(\mathbf{q}, \tau_{k+m}) - t_f(\mathbf{q}, \tau_k) \geq 0$$

That is, if $(\mathbf{q}, t)$ is on the hyperline $[\tau_k, t_f(\mathbf{q}, \tau_k))$, then, since $t \geq \tau_{k+m}$, it is also on the hyperline $[\tau_{k+m}, t_f(\mathbf{q}, \tau_{k+m}))$. On the other hand, if $(\mathbf{q}, t)$ belongs to $U(\tau_k)$, then $t \geq t_f(\mathbf{q}, \tau_k)$, but as long as $t < t_f(\mathbf{q}, \tau_{k+m})$, $(\mathbf{q}, t)$ belongs to $F(\tau_{k+m})$. ∎

The significance of the above theorem is that more collision-free CT-space points can be discovered as sensing time progresses, i.e., the collision-free regions can only grow.

4.8 Summary

This chapter introduced the notion of a dynamic envelope to detect if a CT-point is collision-free or not without assuming about any future motions of obstacles. Through "progressive sensing", i.e., observing the poses of obstacles at different sensing instants τ, a better decision is made whether a CT-point is guaranteed collision-free or not. Further, it is shown that for a collision-free CT-point a dynamic envelope detects a collision-free CT-region in the neighborhood of that CT-point. The notion of Perceived CT-space is introduced to characterize the CT-space discovered by sensing using our approach; it is shown that the guaranteed CT-free space only grows as

sensing time progresses.

CHAPTER 5: DETECTING A COLLISION-FREE TRAJECTORY

In the real world, whether and how objects in an environment move can be unpredictable, and a robot's own motion is also subject to uncertainty. In the previous chapter, the notion of dynamic envelope was introduced to discover guaranteed collision-free CT-point $\chi = (\mathbf{q}, t)$, for unpredictable environments. Now to enable a robot to move safely in such an environment, it is necessary to be able to detect a robot trajectory that is (a) *guaranteed* collision-free in spite of unknown motions of obstacles, (b) *continuously* collision-free, i.e., not only at discrete[1] configuration-time (CT) points but also at all in-between configuration-time points, and (c) *robustly* collision-free in spite of robot motion uncertainty, all in real-time.

5.1 Approach

For an n-DOF robot, a continuous trajectory segment Γ from CT-point $\chi_s = (\mathbf{q}_s, t_s)$ to CT-point $\chi_e = (\mathbf{q}_e, t_e)$ can be formulated as:

$$\begin{aligned} \Gamma = \mathbf{q}(t) = [q_1(t), ..., q_n(t)]^T, \\ t_s \leq t \leq t_e, \end{aligned} \tag{18}$$

where $q_1(t), ..., q_n(t)$ are continuous functions of time t for respective joint variables.

We consider a trajectory Γ *robustly* collision-free if (i) it is detected continuously col-

[1]Discretizing a continuous trajectory into a sequence of discrete configuration-time points may cause a collision to be missed between two consecutive discrete points, especially if the size of an obstacle is not known beforehand.

lision free, rather than being detected collision-free only at discretized configuration-time points, and (ii) when the robot executes Γ with motion uncertainty so that it is not exactly on Γ, the robot motion is still guaranteed collision-free.

With motion uncertainty, the robot will not exactly follow Γ but its motion can occur in a *tunnel* enclosing Γ, which we call Γ^+. To guarantee that the robot motion is collision-free, we need to guarantee that not only Γ but also Γ^+ is collision-free.

Recall that a dynamic envelope of a CT-point χ when detected collision-free at sensing moment τ_l also detects a collision free CT-region $F(\chi, \tau_l)$ as described in Section 4.5. In order to detect if the tunnel Γ^+ is collision-free or not, our idea is to find a set of sparse CT-points $Q(\Gamma^+) = \chi_j, j = 1, ..., k$, such that if each χ_j is detected collision-free at sensing time τ_j, then the tunnel Γ^+ is contained in $\bigcup F(\chi_j, \tau_j)$ and is collision-free, and we say that the trajectory segment Γ is detected *robustly* collision-free.

Figure 11 illustrates this notion for a 1-DOF robot. We now explain the details of how this approach works below.

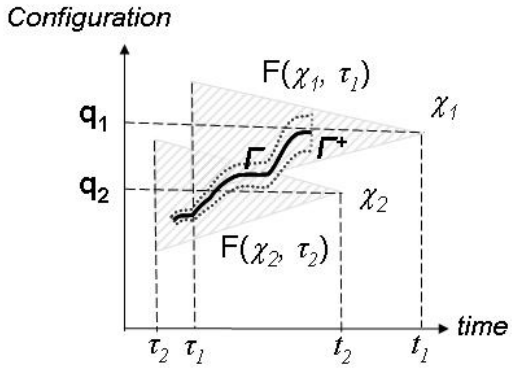


Figure 11: The CT-regions contain the tunnel Γ^+, which encloses Γ.

5.2 Associating Γ^+ to a CT-region of a single CT-point

Now we consider the condition for the CT-region $F(\chi_1, \tau_1)$ of a single CT-point $\chi_1 = (\mathbf{q}_1, t_1)$ to include all (continuous) CT-points satisfying Γ^+.

For a trajectory Γ to be contained in $F(\chi_1, \tau_1)$, by Theorem 2, we have,

$$d_{max}(\mathbf{q}(t), \mathbf{q_1}) \leq v_{max}(t_1 - t) \tag{19}$$

Let $L(t)$ be the cross-section region of Γ^+ at time t. Let $\mathbf{q}'(t)$ (or $(\mathbf{q}', t)$) be any CT-point on $L(t)$. Then, for $\mathbf{q}'(t)$ to be in $F(\chi_1, \tau_1)$, by Theorem 2, we have,

$$d_{max}(\mathbf{q}'(t), \mathbf{q_1}) \leq v_{max}(t_1 - t) \tag{20}$$

However, by triangle inequality, we have,

$$d_{max}(\mathbf{q}'(t), \mathbf{q_1}) \leq d_{max}(\mathbf{q}'(t), \mathbf{q}(t)) + d_{max}(\mathbf{q}(t), \mathbf{q_1}) \tag{21}$$

Thus, from above, if the equation,

$$d_{max}(\mathbf{q}'(t), \mathbf{q}(t)) + d_{max}(\mathbf{q}(t), \mathbf{q_1}) \leq v_{max}(t_1 - t) \tag{22}$$

is satisfied, equations (19) and (20) hold.

Further, let $\mathbf{q}_{max}(t)$ be a configuration in $L(t)$ that satisfies the inequality,

$$d_{max}(\mathbf{q}'(t), \mathbf{q}(t)) \leq d_{max}(\mathbf{q}_{max}(t), \mathbf{q}(t)) \tag{23}$$

Thus, from equation (23), if the equation:

$$d_{max}(\mathbf{q}(t), \mathbf{q_1}) + d_{max}(\mathbf{q}_{max}(t), \mathbf{q}(t)) \leq v_{max}(t_1 - t) \tag{24}$$

is satisfied, equation (22) holds. Hence, if the condition (24) is satisfied, the tunnel Γ^+ is in $F(\chi_1, \tau_1)$. From equation (2), $d_{max}(\mathbf{q}(t), \mathbf{q}_1)$ can be further computed as

$$d_{max}(\mathbf{q}(t), \mathbf{q}_1) = \max_{\forall p_x \in R_x} \|\mathbf{p}_x(\mathbf{q}(t)) - \mathbf{p}_x(\mathbf{q}_1)\|$$

where $\mathbf{p}_x(\mathbf{q}(t))$ can be obtained from $\mathbf{q}(t)$ by forward kinematics.

For a CT-point $\chi_1 = (\mathbf{q_1}, t_1)$ to satisfy the condition (24), the nonlinear function

$$g(t) = d_{max}(\mathbf{q}(t), \mathbf{q_1}) + d_{max}(\mathbf{q}_{max}(t), \mathbf{q}(t)) \tag{25}$$

on the left side of the inequality, has to be bounded by the straight line $f(t) = v_{max}(t_1 - t)$ during interval $[t_s, t_e]$. Figure 12 illustrated the condition (24) visually.

We now need to determine $(\mathbf{q_1}, t_1)$ to satisfy condition (24). To minimize g(t), we select $\mathbf{q}_1 = \mathbf{q}_e$ so that $d_{max}(\mathbf{q}(t), \mathbf{q_e})$ becomes zero at $t = t_e$, and the condition (24) becomes,

$$t_1 \geq t_e + \frac{d_{max}(\mathbf{q}_{max}(t_e), \mathbf{q}(t_e))}{v_{max}} \tag{26}$$

Let,

$$t_m = t_e + \frac{d_{max}(\mathbf{q}_{max}(t_e), \mathbf{q}(t_e))}{v_{max}} \tag{27}$$

To increase the area below $f(t)$ in order to satisfy condition (24), we have to select $t_1 > t_m$ i.e., shifting $f(t)$ along the t axis in the positive direction, as shown in Figure 12.

The subsequent question is how much t_1 should be greater than t_m (or $f(t)$ should be shifted). Apparently the greater the t_1, the more likely condition (24) will be satisfied. However, a greater t_1 can mean a delayed discovery of tunnel Γ^+ being

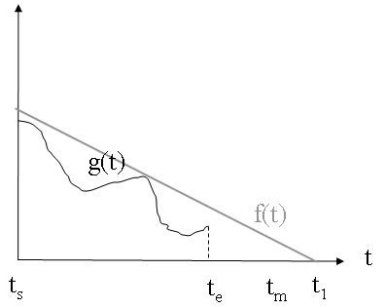


Figure 12: Illustration of the condition (24).

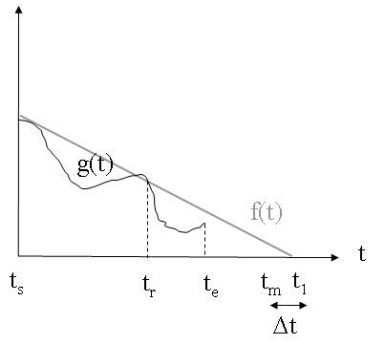


Figure 13: A situation after $f(t)$ is shifted Δt to end at t_1.

collision-free, if it is indeed collision-free. This is because, for the same configuration $\mathbf{q}_1$, if $t_1' < t_1$, it will take longer time to discover if the CT-point $\chi_1 = (\mathbf{q}_1, t_1)$ is collision-free or not than the CT-point $\chi_1' = (\mathbf{q}_1, t_1')$.

Thus, we have to limit t_1 to be only slightly later than t_m to avoid much delay, but then this small shift Δt of the line $f(t)$ may not result in condition (24) to be satisfied (see Figure 13). This is why we want to consider using not just one CT-point, but a set of CT-points $Q(\Gamma^+) = \chi_j, j = 1, ..., k$, to discover if the trajectory tunnel Γ^+ is continuously collision-free or not.

5.3 Associating Γ^+ to CT-regions of a set of CT-points

From equation (27), if $t_1 > t_m$, then for some time interval $[t_r, t_e]$ ending at t_e, $g(t)$ is below $f(t)$. Now we need to find out the value of t_r. There are the following

possible results:

- Case 1: $t_r = t_s$, implying that $g(t)$ is below $f(t)$ for the entire time interval $[t_s, t_e]$;
- Case 2: t_r is the greatest root of equation $g(t) = f(t)$.

If case 1, the single CT-point $(\mathbf{q}_1, t_1)$ is sufficient for discovering if the tunnel Γ^+ is collision-free or not (as described in previous section).

Case 2 means that only the part of Γ^+ for the time interval $[t_r, t_e]$, where $t_s < t_r < t_e$, can be contained by $F(\chi_1, \tau_1)$ of the CT-point $\chi_1 = (\mathbf{q}_1, t_1)$. That further means that the dynamic envelope of χ_1 can be used to only discover if that part of tunnel Γ^+ is collision-free or not. Therefore, we need to find additional CT-points for checking if the remaining part of Γ^+, for the time interval $[t_s, t_r]$, is collision-free. We use Algorithm 1 to find such a set $Q(\Gamma^+)$ of CT-points.

Algorithm 1 $Q(\Gamma^+)$ generator

1: Input Γ for the time interval $[t_s, t_e]$
2: $t_r = t_e$; $\mathbf{q}_r = \mathbf{q}_e$
3: $j = 0$; $Q(\Gamma^+) = \emptyset$
4: **repeat**
5: $\quad j = j + 1$
6: $\quad$ Find $\chi_j = (\mathbf{q}_j, t_j)$ for Γ in $[t_s, t_r]$ as: $\mathbf{q}_j = \mathbf{q}_r$, and $t_j = t_r + \frac{d_{max}(\mathbf{q}_{max}(t_r), \mathbf{q}(t_r))}{v_{max}} + \Delta t$ (as explained in section 4.1)
7: $\quad$ Add χ_j to $Q(\Gamma^+)$
8: $\quad$ Find new t_r and $\mathbf{q}_r$ from equation:

$$g(t) - f(t) = 0 \tag{28}$$

9: **until** $t_r = t_s$ (i.e., Case 1)
10: **return** $Q(\Gamma^+)$

Note that the value of Δt (which is the amount of shift of $f(t)$ along positive time axis) affects the number of CT-points in $Q(\Gamma^+)$. Recall that we want Δt to be small to

avoid time delay in detecting if Γ^+ is collision-free or not. However, if Δt is too small, there can be too many CT-points in $Q(\Gamma^+)$, and thus the cost of collision checking (of the CT-points via their dynamic envelopes) increases. So instead of using a fixed Δt, our strategy is to adapt the value of Δt to balance the need of fast detection of collision-free CT points and the number of CT points for detection.

Note also that solving the non-linear equation (28) to find roots may require numerical techniques. There are derivative-free[2] fast numerical methods [8,10,57] which guarantee to find the roots of a non-linear equation $g(x) = 0$ in an interval $[a, b]$, if $g(a)g(b) < 0$. If $g(a)g(b) > 0$, we can sample within this interval for potential roots, which requires evaluating a non-linear function. If the function is a high order polynomial, there are fast numerical methods for evaluation [75]. In the case of a robot, the L.H.S. of equation (28) can be converted to a polynomial (by variable substitution in transcendental equations).

5.4 Collision-free perceiver

We refer to our general sensor-based online detector of collision-free CT points as *collision-free perceiver* (CFP). The CFP algorithm is shown in Algorithm 2. The CFP keeps observing a CT point $\chi = (\mathbf{q}, t)$ from a starting sensing time τ_0 until it is detected collision-free (causing **return** from the algorithm) or the time $\tau_e < t$ is reached, i.e., a maximum computing period $\tau_e - \tau_0$ is met, as monitored by a system clock variable t_{clock}. It relies on *progressive sensing* with the latest updates. t_{clock} updates itself independently outside the CFP algorithm. CFP gives the binary output

[2]No need to differentiate L.H.S. of (28)

of either possible collision or guaranteed collision-free for the CT-point. However, if time t is reached and $E(\chi, t)$ is not free of obstacle, the CT-point $\chi = (\mathbf{q}, t)$ is definitely not collision-free.

The interval between two adjacent sensing instants is $\Delta\tau$, i.e., the sensing frequency is $1/\Delta\tau$. Note that each iteration in the **while** loop usually takes longer than $\Delta\tau$. Thus, after each iteration, there is always the updated sensing data for the next iteration.

Algorithm 2 Collision-Free Perceiver (CFP)

```
1: Input CT point χ = (q, t), τ = τ_0, τ_e < t, Δτ, t_clock = 0
2: while t_clock ≤ τ_e − τ_0 and τ ≤ τ_e do
3:    Get dynamic envelope E(χ, τ)
4:    if E(χ, τ) does not intersect with any obstacles at τ then
5:       E(χ, τ) expires
6:       return χ is guaranteed collision-free
7:    end if
8:    τ = τ + Δτ (for next sensor data)
9: end while
10: return χ may not be collision-free
```

The CFP is quite efficient for real-time operation because the dynamic envelopes are of simple shapes, and the algorithm only returns a boolean value and does not require expensive minimum distance computation. Efficient collision detection algorithms using bounding volume hierarchy can be applied here.

For each CT-point $\chi_j = (\mathbf{q}_j, t_j)$ in $Q(\Gamma^+)$, we apply CFP to detect if it is collision-free starting from some initial sensing moment $\tau_0 < t_s$. For each CT-point χ_j in $Q(\Gamma^+)$, the corresponding τ_e should be set before the time component of the earliest CT-point on Γ^+ covered by $F(\chi_j, \tau_j)$. We simply set $\tau_e < t_{j+1}$ satisfying $\mathbf{q}(\tau_e) = \mathbf{q}_{j+1}$, and for $j = k$, $\tau_e = t_s$. Figure 14 illustrates this notion.

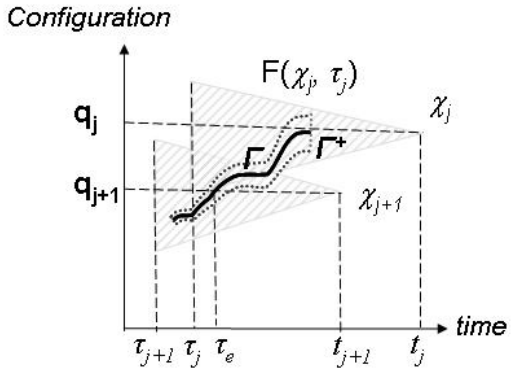
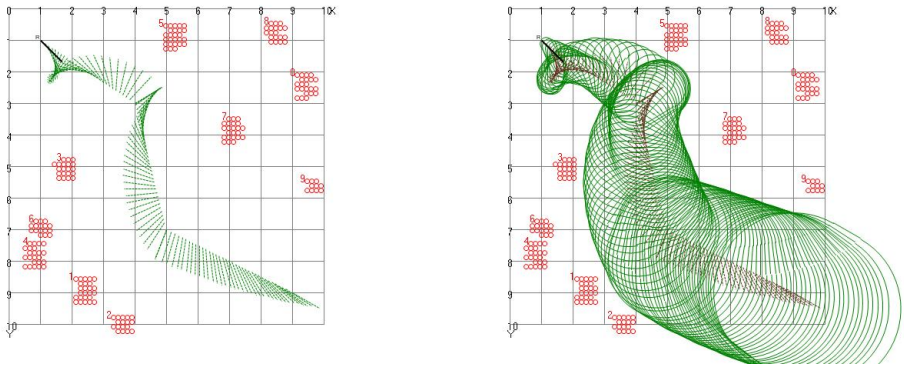


Figure 14: Illustration of τ_e for CT-point χ_j.

5.5 Implemented examples



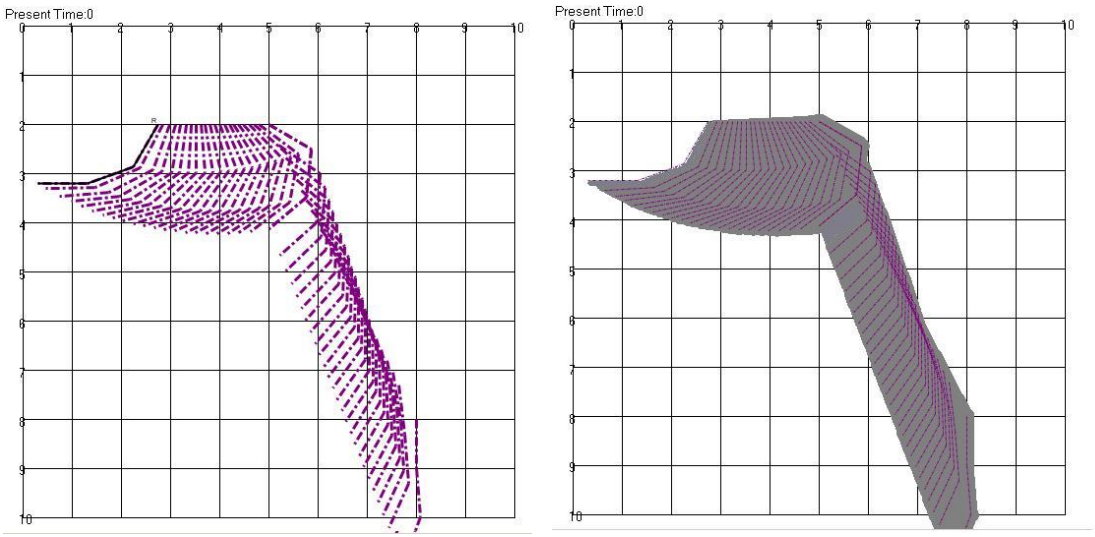
(a) Γ shown by the sequence of CT-points in $Q(\Gamma^+)$.

(b) Dynamic envelopes of the CT-points in $Q(\Gamma^+)$.

Figure 15: Piece-wise continuous trajectory Γ consisting of three segments.

Our approach was implemented in 2D simulation environments. First we demonstrate an example of continuous collision checking with a planar rod robot by simply assuming a zero width tunnel, i.e. $\Gamma^+ = \Gamma$. Later, for a mobile planar manipulator robot, we demonstrate an example assuming that the robot has motion uncertainty when it executes a trajectory, i.e. Γ^+ has a non-zero width.

The mobile rod robot has three degrees of freedom in its configuration, i.e. $\mathbf{q} = [x, y, \theta]^T$, where the origin of the robot frame was at one end of the rod. The rod



(a) Γ shown by the sequence of CT-points in $Q(\Gamma^+)$.

(b) The tunnel Γ^+ representing the motion uncertainty when the robot executes trajectory Γ.

Figure 16: Piece-wise continuous trajectory Γ with two segments.

robot moves in an unpredictable environment with obstacles shown as red circles (as in Figure 6). A continuous trajectory segment for the rod robot was generated with cubical polynomials [16].

Figure 15(a) shows the configurations of a piece-wise continuous trajectory Γ consisting of three continuous segments. The discrete configurations shown along Γ are from the set $Q(\Gamma^+)$ generated by Algorithm 1. Γ starts from time $t_s = 2s$ and ends at time $t_e = 4.4402s$. Figure 15(b) shows a snapshot of dynamic envelopes of the CT-points in $Q(\Gamma^+)$ when the continuous Γ is detected collision-free. In this environment, $v_{max} = 1$ unit/s.

The mobile planar manipulator robot consists of four links and three joints, with link 0 being a point, where the robot base frame is set. It has five degrees of freedom in its configuration: $\mathbf{q} = [\theta_1, \theta_2, \theta_3, x, y]^T$. A continuous trajectory segment for the robot was generated with cubical polynomials [16].

Figure 16(a) shows the configurations of a piece-wise continuous trajectory Γ consisting of two continuous segments. The discrete configurations shown along Γ are from the set $Q(\Gamma^+)$ generated by Algorithm 1. Γ starts from time $t_s = 0.1s$ and ends at time $t_e = 2.58s$. Figure 16(b) shows the tunnel Γ^+ representing all the CT-points that the robot could reach when it executes trajectory Γ. The motion uncertainty for each segment grows linearly from 0.01° to 0.1° for joint variables $\theta_1, \theta_2, \theta_3$ and from 0.01 unit to 0.1 unit for joint variables x and y.

Figure 17 shows the results of CFP over time, i.e., via progressive sensing, in detecting if a CT-point in $Q(\Gamma^+)$ is collision-free for the mobile manipulator in an unpredictable environment with randomly moving obstacles in red dots with $v_{max} =$ 1.5 unit/s. It also shows the robot's motion in executing the detected collision-free part of the trajectory. Each CT-point in $Q(\Gamma^+)$ is shown by the corresponding dynamic envelope at the indicated sensing time. The green dynamic envelopes are free of obstacles, representing CT-points just detected collision-free at the indicated sensing time, whereas the red dynamic envelopes contain obstacles. The moment after a dynamic envelope is free of obstacle, it expires and is no longer displayed. As sensing time progresses, most dynamic envelopes shrink enough to "squeeze out" obstacles, i.e., indicating that the corresponding CT-points are collision-free, before the robot reaches those CT-points. However, in Figure 17(f), the dynamic envelope cannot shrink enough to "squeeze out" obstacles and the robot has to make a stop because it cannot follow the portion of the trajectory that is not collision-free. The attached video clip shows the entire process depicted in Figure 17.

Table 1 and 2 shows the results for the trajectories of the planar rod robot and

Table 1: Results of $Q(\Gamma^+)$ of the continuous trajectory Γ shown in Figure 15(a) of the mobile rod robot in different environments

v_{max} (unit/s)	Smallest interval(s)	Largest interval (s)	$\lvert Q(\Gamma^+)\rvert$	Δt (s)
1	0.0154	0.0570	99	0.1
1.5	0.0250	0.0858	60	0.1
2	0.0158	0.1023	41	0.1

In all cases, the time to generate $Q(\Gamma^+)$ was within 1s.

Table 2: Results of $Q(\Gamma^+)$ of the continuous trajectory Γ shown in Figure 16(a) of the mobile manipulator in different environments

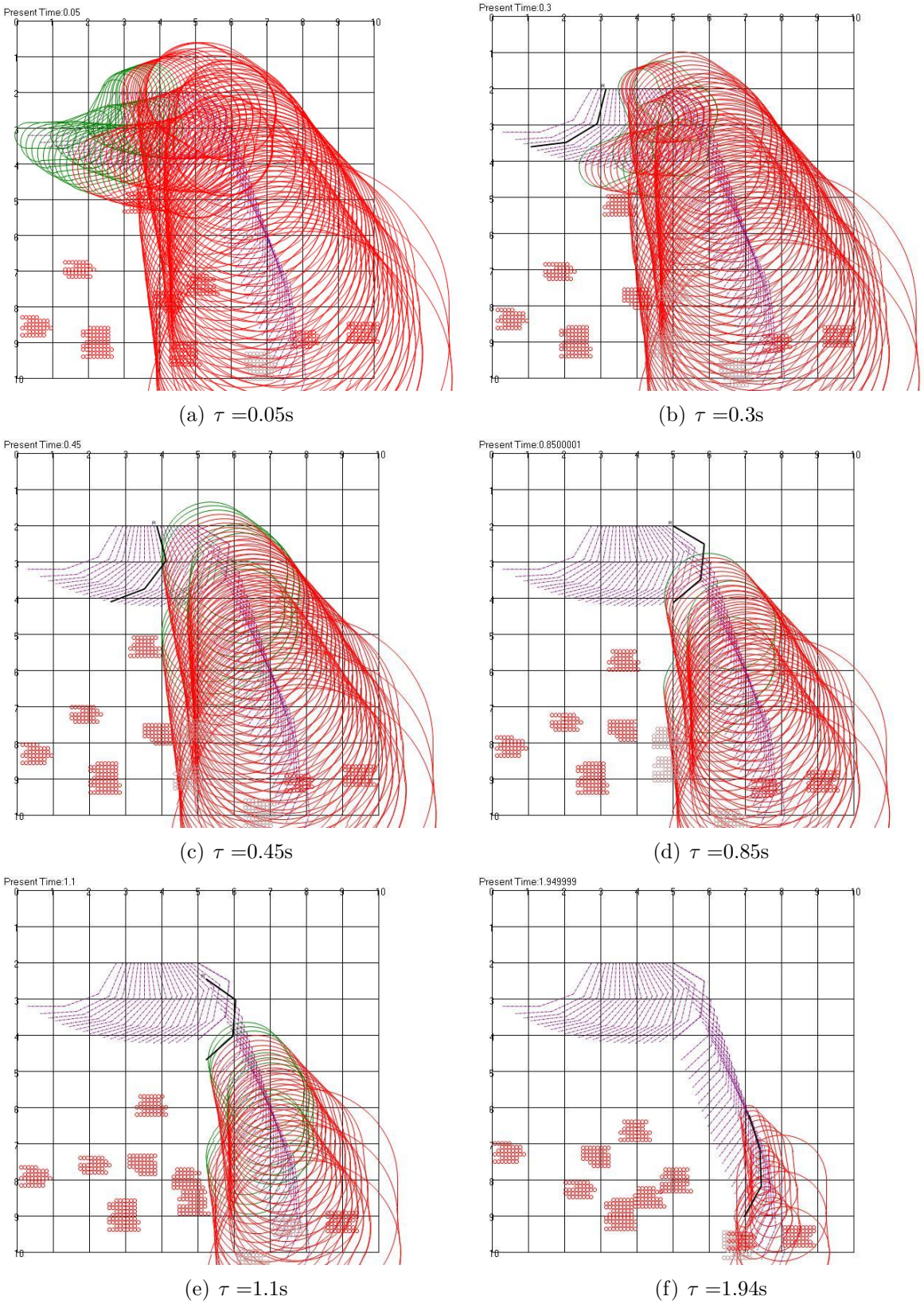
v_{max} (unit/s)	Smallest interval(s)	Largest interval(s)	$\lvert Q(\Gamma^+)\rvert$	Δt (s)
1	0.0079	0.2460	121	0.2
1.5	0.0192	0.3798	48	0.2
2	0.0319	0.5119	27	0.2

mobile manipulator respectively in three environments of different levels of obstacle activities, as indicated by different values of v_{max} (which is the upper-bound of obstacle speeds). Each table shows the smallest time interval and the largest time interval between two consecutive CT-points $(\mathbf{q}_j, t_j)$ and $(\mathbf{q}_{j+1}, t_{j+1})$ in $Q(\Gamma^+)$, the number of CT-points in $Q(\Gamma^+)$, and Δt in each case. Note that as v_{max} increases, the number of CT-points in $Q(\Gamma^+)$ decreases in both examples. This is due to that the increase of v_{max} enlarges the dynamic envelope $E(\chi_j, \tau)$ of $\chi_j = (\mathbf{q}_j, t_j)$ at any sensing time τ, and, if $\chi_j = (\mathbf{q}_j, t_j)$ is detected collision free at τ, the associated collision-free CT-region $F(\chi_j, \tau)$ is also enlarged by the greater v_{max}.

5.6 Summary

This chapter addressed the novel problem of how to detect a collision-free trajectory robustly in unpredictable environments. It shows how to find a set of sparse CT-points $Q(\Gamma^+) = \chi_j, j = 1, ..., k$, such that if each χ_j is detected collision-free at sensing time

τ_j, then the tunnel Γ^+ is contained in $\bigcup F(\chi_j, \tau_j)$ and is collision-free.



(a) τ =0.05s (b) τ =0.3s

(c) τ =0.45s (d) τ =0.85s

(e) τ =1.1s (f) τ =1.94s

Figure 17: Snapshots of robot moving along a trajectory Γ.

CHAPTER 6: ATOMIC OBSTACLES (AO)

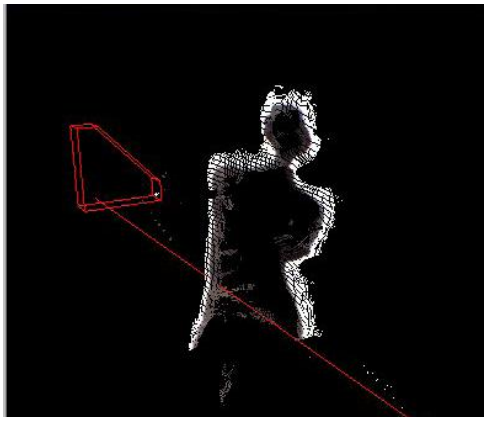
A robot can navigate safely in an environment if it knows the regions (obstacles) that could be responsible for collision, on right time. Human often group or divide these regions to refer them as objects such as, table, chair, etc. While human eyes can quickly detect individual objects, sensors can detect only 2D regions that are occupied by these objects. Reconstructing these objects from sensor data is a tedious process [24, 38] and limits real-time performance of robot to navigate safely in the environment. In this chapter, we introduce the notion of atomic obstacles that instantly represents an environment, directly from sensor data.

6.1 Sensor data generated at a sensing moment

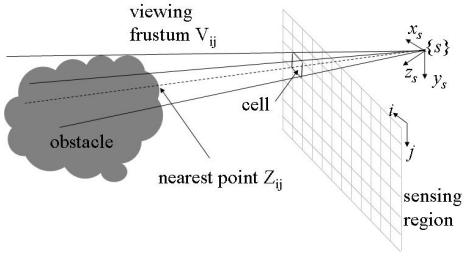
Most sensors, such as, stereo-vision, laser range-finder, sonar, etc., generate 3D point clouds based on disparity calculations [83] or time-of-flight calculations, within sensing interval $[\tau_{k-1}, \tau_k], k > 0$. An example of point cloud is shown in Figure 18(a), which is obtained from stereo-vision sensor. Using a sensor, only the front part of obstacle is detected and the *occluded* part, i.e., where sensors cannot see, are represented as empty spaces. Thus, the kind of sensor data generated is often referred to as having 2.5 dimension(D), e.g., [85].

Every sensor s has a sensing region[1] that is discretized into $M \times N$ cells (sensor

[1]For a stereo-vision sensor, CCD of one of the camera is a sensing plane.



(a) Point cloud image of a human from Point Grey's Digiclops sensor. Reprint from Digiclops manual.



(b) Sensor data Z_{ij} is the nearest point to the sensor frame lying within viewing frustum of cell at index (i, j).

Figure 18: Sensor data generated at a sensing moment.

resolution). Each cell (or *pixel*) at index (i, j) is associated with a *viewing frustum* V_{ij}, bounded by the four rays originating from the sensor frame $\{s\}$ and passing through four corners of the cell (see Figure 18(b)). For stereo-vision sensor, which observes perspective projection geometry, the viewing frustum is a *trapezoidal ray*, whereas, for laser-range finder or sonar, which observes parallel projection geometry, the viewing frustum is a *rectangular ray*. The sensor data Z_{ij} is the nearest obstacle point present in a viewing frustum V_{ij}. Figure 18(b) shows an example for a stereo-vision sensor that has sensing region as a plane.

Thus, each point of a point-cloud, representing the front part of an object, is associated with (i) cell (sensing pixel) at (i, j) on the sensing region, and (ii) sensor data $Z_{ij} = \{x, y, z\}$.

The sensor data generated is quite rudimentary in nature, although it can capture most real-world environments that consist of many objects with complex geometry structures. For a particular environment, let *obstacle space* $O(\tau_k)$ be the set of all

objects present at time τ_k. The numerous kinds of objects can appear or disappear from a real-world environment, become separate into smaller objects, or combine into bigger ones, i.e., the set $O(\tau_k) \neq O(\tau_{k+m}), m > 0$ and, thus, changes with time.

A sensor s, also, has a limited sensing frustum and thus may only see part of objects $O^s(\tau_k)$ in the environment at a particular sensing time, i.e., $O^s(\tau_k) \subseteq O(\tau_k)$.

6.2 Definition, properties and examples

An *atomic obstacle* $O_{ij}(\tau_k)$, at a sensing moment τ_k, has simple default geometry as a part of viewing frustum V_{ij}, starting from sensor data Z_{ij} and including all the regions that could possibly be a part of obstacles in viewing frustum V_{ij}, with (i, j) defining one-to-one mapping to a part of an object in the environment, such that, if $O^s(\tau_k)$ is the set of volumes of all objects viewed in the sensing frustum of sensor s, and the set of all atomic obstacles is $O^s_A(\tau_k) = \bigcup_{\forall ij} O_{ij}(\tau_k)$, then,

$$O^s(\tau_k) \subseteq O^s_A(\tau_k), \forall k \tag{29}$$

Figure 19 shows an example.

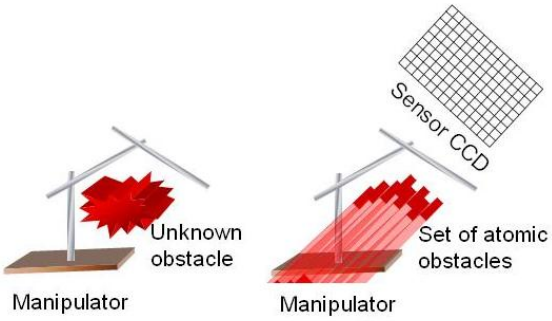


Figure 19: An environment is viewed as a set of atomic obstacles.

Following are the general properties of atomic obstacles:

1. The generation of atomic obstacles does not need elaborate sensor information processing as required for obstacle detection [18] or map building [37]. Thus, the sensing cycle is same as the cycle where information about obstacles in the environment are updated.

2. The occluded part of the environment is directly represented as an obstacle space by the group of atomic obstacles. Thus, a collision between obstacles in environment and robot model can never be missed if one exists.

3. Collision checking between an atomic obstacle and a robot model is fast as it has simple default geometry.

4. The number of atomic obstacles generated in sensing interval $[\tau_{k-1}, \tau_k]$ is at most equal to the sensor resolution $M \times N$, i.e., # of sensor data generated by the sensor at a sensing time.

5. There may not exist any relationship between $O_A^s(\tau_k)$ and $O_A^s(\tau_{k+m}), m > 0$. Thus, the memory requirement for storing atomic obstacles is a constant size of $M \times N$.

The examples of atomic obstacles are given next.

6.2.1 Polygons/Circles as 2D atomic obstacles

Most literature in robotics (e.g., [14, 64, 68, 74]) plans mobile robot motion in 2D space and execute it in real environment, which is 3D. The sensor provides top-view of the environment and collision-checking is done in 2D space.

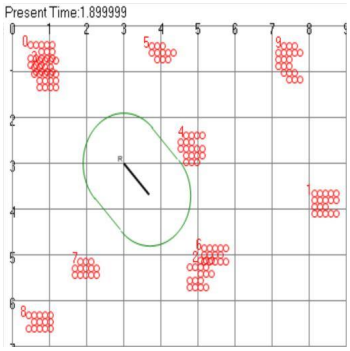


Figure 20: Red circles as atomic obstacle.

For any sensor, the sensor data $Z_{ij} = \{x, y, z\}$ generated often represents its z value with depth (height) information. As objects with height can be obstacles to the robot, we can distinguish floor from object using the z value of sensor data. Further, all objects in the environment can be projected on x-y plane where the plane includes the top of the robot. The x and y of the sensor data can be used to determine the centre of circle/polygon of fixed size depending on the resolution of sensor. Figure 20 shows an example with red circles as atomic obstacles, taken from a sensor that observes parallel projection geometry. The circle/polygon forms the front *face of an atomic obstacle* $O_{ij}(\tau_k)$.

Note that the sensor data is sufficient to represent the environment for 2D mobile robot motion planning, i.e., $O^s(\tau_k) = O^s_A(\tau_k), \forall k$.

6.2.2 Polygonal projection rays as 3D atomic obstacles

Motion planning for a high-DOF of robot, such as manipulator, humanoid, etc., needs to know the environment in 3D in order to perform complex task, such as, serving food in restaurant. For motion planning the exact model of objects need not be known as the robot has to only avoid collision from them.

The sensor generates the 3-D point $\{x, y, z\}$ in W_{ij} that is closest to the image plane, with distance d_{ij}. W_{ij} can be viewed as the projection of the square pixel at (i, j) on the image plane to the sphere centered at the origin of the sensor frame $\{s\}$ with radius d_{ij}. Thus, W_{ij} is the front *face of atomic obstacle* O_{ij} (see Figure 21).

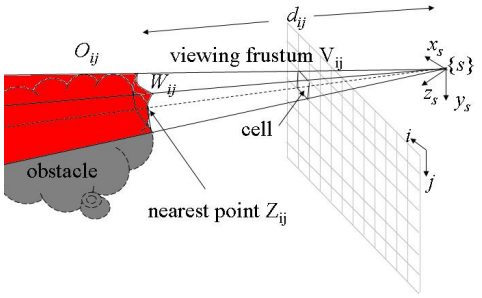


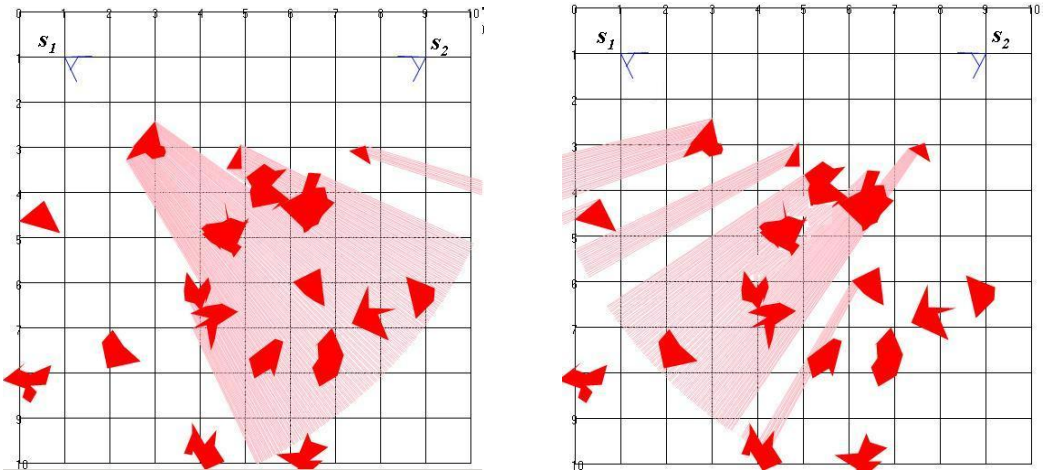
Figure 21: The geometry of an atomic obstacle O_{ij} (shown in red color).

Note that the sensor data is insufficient to represent the environment, and thus atomic obstacle have infinite size that causes some free-space to be represented as atomic obstacles i.e. $O^s(\tau_k) \subset O^s_A(\tau_k), \forall k$.

6.3 Some free space represented as atomic obstacles

As $O^s(\tau_k) \subseteq O^s_A(\tau_k), \forall k$, some free space in the environment is represented as if they were a part of obstacle space by atomic obstacles. Figure 22 shows an example for a planar environment containing obstacles (shown in red) and two stereo-vision sensors s_1 and s_2.

At any particular sensing moment, free space is required by the robot to move to its goal configuration, and if a collision-free goal configuration is occluded at sensing moment τ_k, i.e., placing the robot at the goal configuration will make it considered inside some atomic obstacles at τ_k, then no planning will be able to find a collision-free trajectory for the robot to reach the goal with the current sensing information.



(a) Atomic obstacles from sensor s_1. (b) Atomic obstacles from sensor s_2.

Figure 22: Some free-space is represented as a part of atomic obstacles.

However, by continuously observing the goal configuration, it is possible that the occluding object moves away, and the goal configuration is visible again at some future moment $\tau_{k+m}, m > 0$.

As atomic obstacles are directly derived from sensor data, different sensor views for the same set of obstacles will result in different set of atomic obstacles. For example, for the same set of obstacles, Figure 22(a) shows the set of atomic obstacles from s_1 and Figure 22(b) shows the set of atomic obstacles from sensor s_2. Some free space viewed as atomic obstacles from one sensor, will be viewed as free space by the other sensor. Hence, more free space can be found using multiple sensors at the same sensing moment if obstacles are viewed from different direction; Figure 23 shows the union of free space from two stereo-vision sensor s_1 and s_2, has the blue boundary, and includes more free space than that from a single sensor.

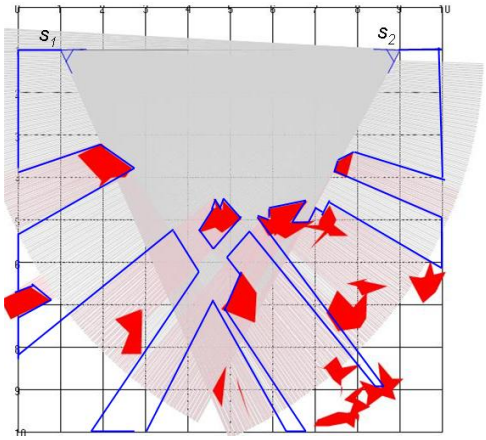


Figure 23: The union of free space visible from two sensors s_1 and s_2.

6.4 Collision checking between the robot model and obstacles

As obstacle geometries are unknown, collision checking between the robot model and obstacles is possible if obstacles in the environment are represented by a set of atomic obstacles obtained from each sensor at the same time. Algorithm 3 shows the collision checking between the robot model and obstacles with n different sensors that can see the robot model, i.e., whose sensing frustums contain the robot model.

Algorithm 3 Collision checking between the robot model and obstacles

1: Input robot model and atomic obstacles $O_A^{s_1}(\tau_k)$,...,$O_A^{s_n}(\tau_k)$ from n sensors s_1,...,s_n.
2: **for** $s = s_1$ to s_n **do**
3: **if** the robot model is closer to sensor s than any atomic obstacle in $O_A^s(\tau_k)$ **then**
4: *return* no collision
5: **end if**
6: **if** the face of any atomic obstacle in $O_A^s(\tau_k)$ intersects with the robot model **then**
7: *return* collision
8: **end if**
9: **end for**
10: *return* collision may exist

If the robot model does not intersects with obstacles, there exists a sensing view

from a sensor s where the robot model is closer to that sensor s than any face of atomic obstacles. Otherwise, if the face of an atomic obstacle, which is associated with sensor data, intersects with the robot model, then a collision between the robot model and obstacles in the environment surely exists. If the robot model is not in front of the atomic obstacles of any sensor and it does not intersect with the face of any atomic obstacle, it means the robot model is behind the faces of all atomic obstacles of the n sensors that are considered. Thus, it is uncertain whether the robot model is just occluded or in collision with some obstacle, i.e., a collision may exist.

6.5 Summary

This chapter introduced the novel notion of atomic obstacles that instantly represent moving obstacles in an environment. Such an atomic obstacle has geometry similar to that of a viewing frustum, where it starts from the front face W_{ij}, including all the occluded space in that viewing frustum. To minimize occlusions caused by infinite volumes of atomic obstacles, multiple sensors can be used.

CHAPTER 7: COLLISION FREE PERCEIVER WITH AO

For a robot to move safely in an environment, a motion planner must be able to detect collisions with obstacles in order to avoid them. Such collisions must be detected in real-time for the robot to avoid obstacles. Even if obstacle geometries are unknown, using atomic obstacles, collision checking between the robot model R and obstacles is shown in Algorithm 3. Algorithm 3 uses two computational steps to check:

1. if the robot model is closer to a sensor than the atomic obstacles, and
2. if there is intersection between the robot model and faces of atomic obstacles of a sensor.

In this chapter, we try to solve these two computational steps efficiently to operate in real time.

Algorithm 3 used atomic obstacles to instantly represent an unknown environment that changes unpredictably with time. As atomic obstacles are directly derived from sensor data, the number of atomic obstacles $O_A^s(\tau_k)$ present at a given sensing moment τ_k, from a sensor s equals the resolution of sensor $M \times N$, which can be a large number. For example, even a sensor with a coarse resolution of 188×120 generates up to 22,560 atomic obstacles. Thus, key to the real-time efficiency of the collision-detection algorithm with atomic obstacles is how to manage a large number of atomic

obstacles to minimize the number of intersection computations with the robot model.

Our algorithm uses the following strategies, which will be detailed in the following sections:

- Extraction: Consider only those atomic obstacles that are likely to intersect with a robot model, i.e., the atomic obstacles whose indices (i, j) are on the projection $P(R)$ of the robot model, on the image plane.

- Grouping: Partition pixels on $P(R)$ into multi-size *super pixels*, such that each super pixel corresponds to a $m \times n$ image region of $P(R)$, with varied $m(\geq 1)$ and $n(\geq 1)$ values[1]. The atomic obstacles corresponding to a super pixel on $P(R)$ form a *combined atomic obstacle*. With such grouping, intersection checking is reduced to that between the robot model and combined atomic obstacles (which are far fewer than the atomic obstacles).

- Hierarchical Checking: Perform intersection checks efficiently through multi-level simplified computations by subdividing the combined atomic obstacle into smaller ones when an intersection is detected between a robot model and that combined atomic obstacle. Thus, if no intersection is detected at a high-level, than there is no intersection for sure; else, re-check intersection at a lower level.

7.1 Extraction and grouping

The idea here is to identify super pixels of varied m and n values to partition the projection $P(R)$ of the robot model on the image plane without explicitly computing the boundary of $P(R)$. A super pixel has four corner points as shown in Figure 24,

[1]When $m = n = 1$, the super pixel is reduced to a normal pixel.

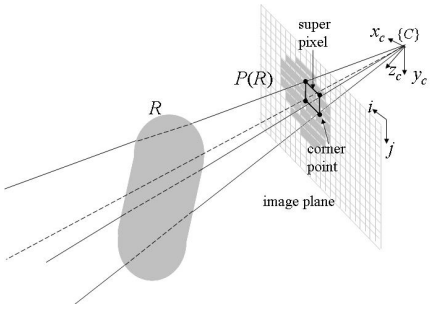


Figure 24: Ray intersection tests to detect an internal super pixel.

and a ray that originates from the origin of the camera frame $\{C\}$ and passes a corner point is called a *corner ray*. If all corner rays of the super pixel intersect the robot model, then the super pixel is called an *internal super pixel*; else, if at least one corner ray intersects with the robot model, the super pixel is called a *boundary super pixel*.

Our strategy simultaneously discovers $P(R)$ and covers it by a partition of super pixels of multiple sizes. We say the region already discovered and partitioned by internal super pixels the *discovered region* of $P(R)$, and the remaining region of $P(R)$ the *undiscovered region* of $P(R)$.

Starting from an entirely undiscovered $P(R)$, our strategy first finds a seed $m \times n$ super pixel on the image plane by determining the upper leftmost pixel of the region and the m and n values. The ideal seed should be the maximum axis-aligned rectangular region that fits inside $P(R)$. However, since $P(R)$ is not known precisely, we make a reasonable estimate based on the OBB of the robot model and its projection on the image plane. Values of m and n should be chosen large enough because they will only be reduced later.

Next, our strategy consists of the following steps:

- Extract: from the seed super pixel, find all neighboring internal or boundary

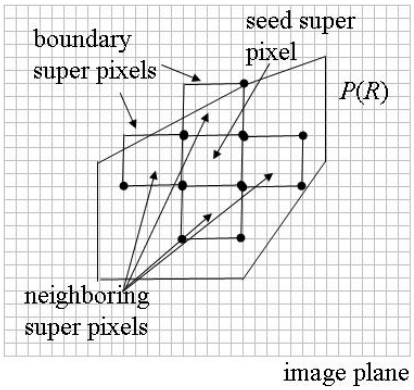


Figure 25: Neighborhood expansion of super pixels.

super pixels on the originally undiscovered regions of $P(R)$ and their neighbors, etc., in a fashion similar to connected neighborhood expansion of the flood-fill algorithm [39,87]. Put every boundary super pixel in a first-in-first-out (FIFO) queue B. Figure 25 illustrates this process.

- Re-seed: Remove the first boundary super pixel from B and if it is larger than a minimum size $m_0 \times n_0$[2], then
  - for every corner point p inside $P(R)$, reduce m and n to halfs with p fixed to get a smaller super pixel, and go to Extract.
- **Return** the internal and boundary super pixels of various sizes of $P(R)$.

Note that the FIFO queue B is in the order of decreasing size of the boundary super pixels. Therefore, when the first $m_o \times n_o$ boundary super pixel is removed from the queue, the remaining queue has only $m_o \times n_o$ boundary super pixels. Thus, the partition is complete.

[2] $m_0(\geq 1)$ and $n_0(\geq 1)$ are proportional to the initial m and n and are set to make sure that the area of partition covering $P(R)$ is not much larger than $P(R)$ and yet also without using too many super pixels, i.e., to provide a good tradeoff.

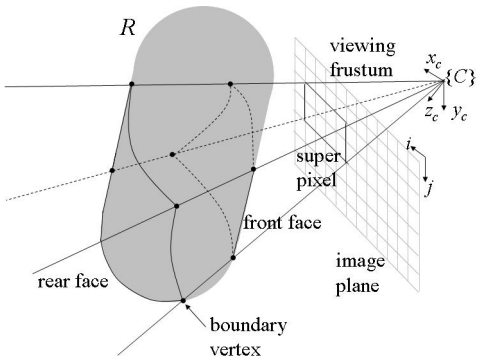


Figure 26: Illustration of some notations.

The above strategy generates a partition of super pixels for $P(R)$ efficiently without requiring excessive intersection checking that a standard cell decomposition method, such as a quadtree [53] requires. Our method discovers pixels or super pixels of $P(R)$ through connected expansion (from a seed by flood-fill) and thus avoids the problem of having a cell with all four corner points outside of $P(R)$ but possibly some internal points inside $P(R)$. Whereas, such a cell is possible with quadtree decomposition, and it is expensive to check if the cell is entirely outside $P(R)$, which may involve intersection checking beyond just between corner rays and the robot model.

7.2 Hierarchical checking

Once a super pixel of $P(R)$ is determined, we can next check if its corresponding combined atomic obstacle intersects with the robot model or not.

We first introduce a few related terms and notations (see Figure 26) as following: Viewing frustum of a super pixel: the rectangular pyramid region defined by the four rays originating from the origin of $\{C\}$ and passing through the four corner points of a super pixel. It extends to infinity and is the union of viewing frustums of all pixels that form the super pixel.

Minimum distance d_{cao} of a combined atomic obstacle: the distance from the origin of $\{C\}$ to the atomic obstacle closest to the origin of $\{C\}$ in the combined atomic obstacle.

Front and rear faces FF and RF of a robot model intersected by a viewing frustum: the intersection surface regions between the viewing frustum of a super pixel (or a pixel) and the robot model, where the region closer to the camera is the *front face* and the other region is the *rear face*. Note that the two faces become one if the viewing frustum partially intersects the robot model, i.e., not all of its corner rays intersect the robot model (such a viewing frustum corresponds to a boundary super pixel of $P(R)$).

Given a super pixel, if the corresponding combined atomic obstacle is behind RF of the robot model as viewed from the camera, then there is no intersection between the combined atomic obstacle and the robot model; otherwise, there is an intersection.

We adopt a hierarchical refinement approach to combine simple checking and narrowing scope of consideration. Our approach takes advantage of the intersection results between corner rays of the combined atomic obstacle and the robot model (obtained in the step of "Extraction and Grouping" described in section 7.1) to obtain eight *boundary vertices* of the front face and rear face of the robot model. The recursive algorithm *HierCheck* implements our approach that returns either intersection or no intersection, given a super pixel.

Note that in *HierCheck* step 1, d_{cao} is obtained from comparing the distances of all atomic obstacles, which are the readings of stereo vision sensor, in the combined atomic obstacle. Moreover, the considered atomic obstacles are divided into small

Algorithm 4 HierCheck (super pixel)

1: Find d_{cao} and the corresponding atomic obstacle O_{ij}.
2: Compute d_R as a small upper bound on the greatest distance from RF to the origin of $\{C\}$.
3: **if** $d_{cao} \leq d_R$ (i.e., possible intersection), **then**
4: **if** super pixel is a pixel or d_R is less than the minimum distance from a boundary vertex to the origin of $\{C\}$ **then**
5: **return** intersection
6: **end if**
7: Divide the super pixel into one pixel at (i, j) and four smaller super pixels as shown in Figure 27.
8: **if** *HierCheck*(pixel) = no intersection **then**
9: **for each** smaller super pixel **do**
10: **if** *HierCheck*(smaller super pixel) = intersection **then**
11: *return* intersection
12: **end if**
13: **end for**
14: **end if**
15: **end if**
16: **return** no intersection

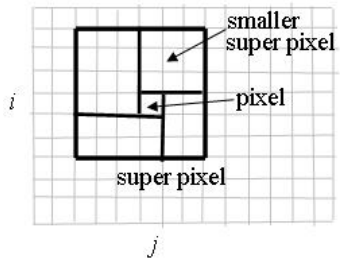
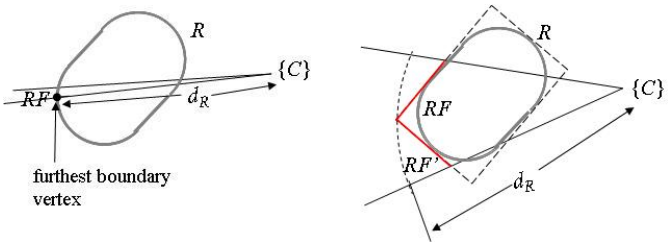


Figure 27: Division of a super pixel into smaller super pixels.

groups and the local minimum distance in each group is computed first. Next the minimum distance is computed from the group minimum. In that way, many group minimum distances can be re-used if the corresponding atomic obstacles are shared by other robot model at the same sensing time τ_k.

In *HierCheck* step 2, d_R is obtained based on the OBB of the robot model if RF is so large that it cannot be viewed as a flat surface. Let RF' be the surface corresponding to RF on the OBB of the robot model. d_R is the distance from the furthest feature

(i.e., vertex, edge, or face) of RF' to the camera origin. If, however, RF is small enough to be considered as a flat surface, then d_R is the maximum distance from a boundary vertex of R_{de} to the origin of $\{C\}$. See Figure 28 for illustrations of the two cases.



(a) RF is a planar surface (b) RF is not a planar surface

Figure 28: Illustrations of two cases of face RF

Steps 3-16 of *HierCheck* shows that if a possible intersection is detected based on the simple inequality, it is necessary to decompose the combined atomic obstacle into smaller ones and do the checking again on them, i.e., recursively moving down to lower levels of the computation hierarchy, until either no intersection is detected or an intersection is detected between a single atomic obstacle and the robot model (or its OBB). Hence, our approach hierarchically refines the accuracy of computation based on need.

7.3 Real-time collision-detection algorithm

With the above strategies (i.e., extraction, grouping, and hierarchical checking) the Algorithm 3 can run in real-time.

7.3.1 Implementation of computational step 1

We conduct the checking of whether the robot model is closer to a sensor than the atomic obstacles from the sensor in the following way. After generating one super

pixel on $P(R)$ using "Extraction and Grouping", proceed to hierarchical checking of whether the corresponding combined atomic obstacle intersects with the robot model or not. If an intersection is detected, collision-detection simply returns "there is intersection" and halt. Otherwise, it generates another super pixel on $P(R)$ (in the flood-fill fashion) and do the hierarchical checking again, and so on. Therefore, this process will only generate enough combined atomic obstacles to find an intersection, and if there is no intersection, the algorithm will halt only after processing all the combined atomic obstacles whose super pixels partition $P(R)$, through hierarchical checking.

7.3.2 Implementation of computational step 2

This step is reached when an intersection is found in step 3 of the algorithm 3. Since "Extraction and Grouping" described in section 7.1 is required initially, the intersection results could be used directly in determining if the face of an atomic obstacle intersects with the robot model. The intersection results include eight *boundary vertices* of the front face and rear face of the robot model (see Figure 26). If the distance d_{cao} of the atomic obstacle is greater than the distance to the front face of the robot model from the sensor frame, then the face of that atomic obstacle intersects the robot model. If there is no intersection, then again repeat the above process of calling "Extraction and Grouping" and then hierarchical checking, until all the atomic obstacles partitioning $P(R)$ are found.

7.4 Algorithm

The notion of dynamic envelope coupled with atomic obstacles enables the detection of collision-free CT-points without requiring to know, recognize, or track obstacles in an unknown changing environment. We refer to our general sensor-based online collision-checker as *collision-free perceiver using atomic obstacles* (CFPA) – it keeps on *perceiving* (sensing) for possible collision until a *collision-free* situation is found, where no obstacle can be responsible for collision. Thus, the binary output given by our algorithm is possible collision or guaranteed collision-free CT-point at a sensing moment.

Our general algorithm, to check whether a CT-point $\chi = (\mathbf{q}, t)$ is collision-free or not, is shown in Algorithm 5. The CFPA discovers if χ is collision-free or not for each sensing instant starting from τ_0 until either the point is discovered collision-free (causing **return** from the algorithm), or maximum computing time $t - \tau_0$ is met, as monitored by a system clock variable t_{clock}. t_{clock} updates itself independently outside the CFPA algorithm. The interval between two adjacent sensing instants is $\delta\tau$, i.e., the sensing frequency is $1/\delta\tau$. Note that each iteration in the **while** loop usually takes longer than $\delta\tau$ for acquiring the image corresponding to the atomic obstacles. Thus, after each iteration, there is always the updated sensing data for the next iteration.

CFPA is quite efficient for real-time operation because of the following:

- CFPA, being a classifier, returns a boolean value and does not require expensive minimum distance computation.

- CFPA only needs to consider a subset of sensed atomic obstacles – those that

Algorithm 5 Collision-Free Perceiver using Atomic obstacles (CFPA)

1: Input CT point $\chi = (\mathbf{q}, t)$, $\tau = \tau_0$, $\delta\tau$, $t_{clock} = 0$
2: **while** $\tau < t$ and $t_{clock} < t - \tau_0$ **do**
3: Get dynamic envelope $E(\chi, \tau)$
4: Use $E(\chi, \tau)$ as robot model to Algorithm 3
5: **if** Algorithm 3 returns no intersection at τ **then**
6: $E(\chi, \tau)$ expires
7: **return** χ is guaranteed collision-free
8: **end if**
9: $\tau = \tau + \delta\tau$ (for next sensor data)
10: **end while**
11: **return** χ may not be collision-free

could be enclosed in $E(\chi, \tau_i)$. As locations of atomic obstacles (x, y, z) are *directly* indexed by (i, j) with a one-to-one mapping, CFPA only considers those atomic obstacles whose indices (i, j) are on the projection of $E(\chi, \tau_i)$ onto the image plane. Figure 29 shows an example with a stereo vision sensor. The number of such atomic obstacles is related to the size of the dynamic envelope that shrinks over time.

- Both the atomic obstacles and the dynamic envelope are of simple shapes.

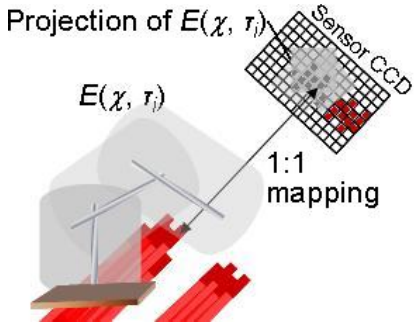


Figure 29: CFPA only considers a subset of atomic obstacles.

Clearly the key step in Algorithm 5 is **step 5** to check whether any atomic obstacle intersects with the dynamic envelope of a CT-point for any sensing instant. Various

Table 3: Costs of computing intersections between a ray and an object

	Operations				
Object	$+, -$	$\times$	$<, >, =$	$/$	$\sqrt{\ }$
OBB	120	108	48	6	0
Sphere	8	11	0	1	1
Capsule	29	48	0	3	3

techniques such as extraction and grouping, and hierarchical checking introduced in Chapter 7 are used to achieve real-time collision-detection algorithm with atomic obstacles.

7.5 Time and space coherence

CFPA repeatedly calls the Algorithm 3 for collision checking for different CT-points and at different sensing intervals. It can take advantage of time and space coherence to reduce computation significantly.

For the same CT-point $\chi = (\mathbf{q}, t)$, if the sensor does not move from sensing moment τ_k to τ_{k+1}, then many super pixels (or pixels) on $P(E(\chi, \tau_k))$ (i.e., the projection of $E(\chi, \tau_k)$ on the image plane) are also on (the smaller) $P(E(\chi, \tau_{k+1}))$. Thus, the low-level intersection checking results between the corresponding rays and $E(\chi, \tau_k)$ can be re-used.

Similarly, if two CT-points have sufficiently close configurations $\mathbf{q}$ and $\mathbf{q} + \Delta\mathbf{q}$, even with different future times t and $t + \Delta t$, their respective dynamic envelopes for any sensing moment τ_k could overlap significantly, and then, again, some low-level intersection checking results of one CT-point can be re-used for the other CT-point.

Table 3 shows the costs of computing the intersection between one ray and one object of each of the given types. If the result is re-usable, such costs are saved.

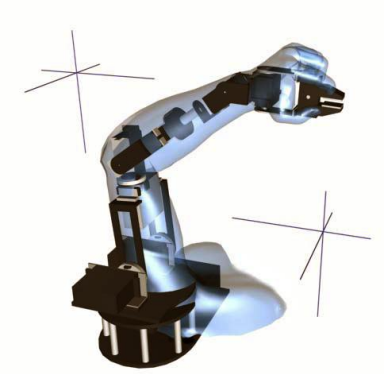


Figure 30: A 7-DOF Cyton arm.

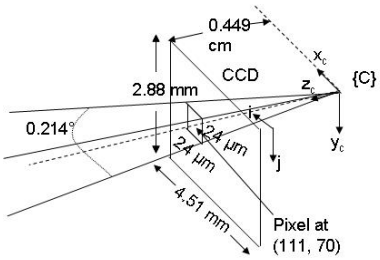


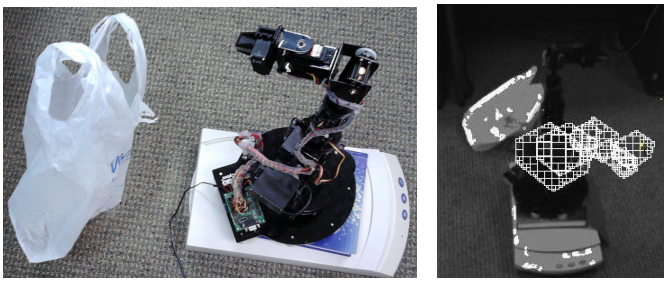
Figure 31: Dimension of atomic obstacles for resolution 752 × 480.

7.6 Implementation and experimental results

We have implemented Algorithm 3, the collision-detection algorithm, as the core algorithm for the CFPA (Algorithm 5) on a Dell Precision T5400 computer and tested the algorithm in real-world experiments using a real 7-DOF Robai's Cyton robot arm (see Figure 30) and an indoor MobileRanger's stereo vision camera. The robot has 7 revolute joints. Each robot link is approximated by an OBB and so is its dynamic envelope.

An atomic obstacle's geometry along with the dimensions is shown in Figure 31. The sensor resolution was fixed to 752×480 and thus, at most 360,960 atomic obstacles were generated at each sensing moment. The sensing frequency was set at 8 Hz. v_{max}= 1 cm/s.

Figure 32(a) shows the environment of the robot, where two obstacles unknown to the robot are nearby, one is a scanner underneath the robot, and the other is a plastic bag. Figure 32(b) shows the same environment as viewed by the stereo vision sensor at sensing time τ_0=0s, superimposed with the projection image of the dynamic envelope of every link of the robot, which is partitioned by super pixels, for the CT-point $\chi_1=(\mathbf{q}_1, t_1)$, and $\mathbf{q}_1$ is the configuration where every joint angle of the robot is at -90^o, and t_1=2s. Note that $\mathbf{q}_1$ is not the robot's current configuration in the figure.



(a) Testing environment

(b) Stereo vision image at τ_0= 0(s) and for the CT-point χ_1, with the partitioned projection image $P(E)$ of the robot's dynamic envelope $E(\chi_1, \tau_0)$.

Figure 32: Experiment and result.

Table 4 shows the results of running the collision-detection algorithm at sensing time τ_0 for two different CT-points χ_1 as introduced above and $\chi_2 = (\mathbf{q}_2, t_2)$, where $\mathbf{q}_2$ is the configuration with all 7 joint angles of the robot at 90^o respectively and t_2=5.57s. The results are for the entire robot (i.e., summing up the results for the dynamic envelopes of 7 individual links). Initial m and n were both set to be 16. $m_0 \times n_0$ was set to 2×2.

As shown in the table, the total number of combined atomic obstacles checked for each CT-point is more than 10 times smaller than the total number of atomic obstacles involved. That implies great saving of computation time in the similar order of magnitude by the collision-detection algorithm rather than using a naive algorithm to directly check intersections of atomic obstacles with the dynamic envelope. Note that the total number of combined atomic obstacles is greater than the total number of super pixels generated in each case because hierarchical checking divided some combined atomic obstacles into smaller ones for checking (see step 7 of **HierCheck**).

For χ_1, the result of no intersection means that the CT-point is guaranteed collision-free. $P(E)$ was actually covered by the super pixels generated (as shown in Figure 32(b)). Because of the efficiency of collision-detection algorithm, χ_1 is discovered collision-free 1.73s *before* its time t_1=2s, which is important for on-line motion planning to utilize that CT-point. The time cost 0.265s of the detection is only about 1% of the total time between the sensing moment τ_0=0s and t_1= 2s. Recall that no actual obstacle was ever identified or recognized.

For χ_2, because collision-detection algorithm halted as soon as an intersection was found, the super pixels generated were only a subset of super pixels for covering $P(E)$. Note that $P(E)$ and the corresponding dynamic envelope for χ_2 was much larger than those for χ_1 because t_2=5.57s was close to three times of t_1=2s. The result of intersection means that it was not certain whether χ_2 was collision-free or not based on the sensing data at τ_0=0s. Further checking with new sensing data was necessary.

Table 5 shows the results of running collision-detection algorithm by CFPA for a

Table 4: Results of running collision-detection algorithm for two CT-points at τ_0

CT-points		χ_1	χ_2
# super pixels	16×16	13	41
	8×8	77	78
	4×4	156	182
	2×2	291	402
	boundary 2×2	640	729
	total	1177	1432
# hierarchy		2	5
# combined atomic obstacles checked		1248	1542
# atomic obstacles involved		14,476	22,924
time cost (s)		0.265	0.328
intersection		no	yes

CT-point $\chi_3 = (\mathbf{q}_3, t_3)$ multiple times with sensing data from six different sensing instants τ, where $\mathbf{q_3}$ is the configuration that every joint angle of the robot is set to 23.39^o, and t_3=4.05s. The results show the reduction of the computation cost by exploiting the sensing time coherence. Collision-detection algorithm was used to check the CT-point χ by generating all low-level intersection checking results from scratch based on the sensing data at $\tau = 0.66$s. Later, however, for $\tau \in (0.66, 0.86]$ (s), useful previous results were repeatedly re-used so that even though the total number of super pixels decrease only slightly as the dynamic envelope shrunk over time, the drop in computation time was far more drastic. Similarly, for $\tau \in (1.25, 1.52]$ (s), the results obtained from $\tau = 1.25$s was re-used repeatedly to reduce the time cost of running collision-detection algorithm.

Notice that the time cost for τ= 0.66s is smaller that for $\tau = 1.25$s, even though the dynamic envelope for the latter was much shrunk from that of the former. This was because as the dynamic envelope shrunk over time, at $\tau = 1.25$s, some atomic obstacles were "squeezed out" of the dynamic envelope. Thus, with fewer intersections

Table 5: Results of running collision-detection algorithm for a CT-point χ_3 with data from different sensing instants within $[0.66, 1.52]$ (s)

τ (s)	time cost (s)	# super pixels generated
0.66	0.109	950
0.74	0.047	889
0.86	0.031	889
1.25	0.344	858
1.38	0.094	852
1.52	0.047	842

present, the algorithm had to spend more time to find one of those intersections.

Note that the CT-point $\chi_3 = (\mathbf{q}_3, t_3)$ was discovered collision-free (i.e., there was no intersection between the dynamic envelope and atomic obstacles) at τ=2s, which was 2.05s. ahead of the time t_3=4.05s.

7.7 Summary

This chapter introduced a fast real-time collision detection algorithm between a robot model and a large number of atomic obstacles. The algorithm uses a number of strategies to speed up computation, including extraction, grouping and hierarchical checking. Using the notion of dynamic envelope coupled with atomic obstacles a general online sensor-based algorithm called Collision Free Perceiver using Atomic obstacles (CFPA) is developed to detect a collision-free CT-point directly from sensor data. Real experiments are demonstrated to show the real-time capability of the approach in detecting a CT-point.

CHAPTER 8: MOTION PLANNING IN PERCEIVED CT-SPACE

A motion planner needs to come up with collision-free trajectories so that the robot, when executing those trajectories, can avoid obstacles and reach its goal configuration. In the previous chapter, for an unknown and unpredictable environment, discovering guaranteed collision-free trajectories (or set of CT-points) via sensing was shown using the algorithm collision-free perceiver using atomic obstacles (CFPA), which is based on notion of dynamic envelope and atomic obstacles. In this chapter, we focus our attention on integrating CFPA with a motion planner. CFPA needs finite time for processing (such as, collision checking) of each collision-free CT-point. Such computational cost must be accounted by motion planners to successfully plan motions in real-world environments.

8.1 Real-time adaptive motion planner (RAMP)

The RAMP paradigm [92] is motivated by the need of real-time motion planning of high-DOF robots, such as (mobile) manipulators, in dynamic environments of unknown obstacle motions. A well known fact about motion planning for high-DOF robots is that no complete algorithm is feasible even for known and static environment due to the formidable challenge of constructing high-dimensional C-obstacles. Thus, sampling-based planners, notably PRM [46] and RRT [55] planners and variants, are widely used.

RAMP is also sampling-based, but it is especially effective in planning high-DOF robot motion in dynamically unknown environments because of the following characteristics:

- real-time *simultaneous* planning and execution of high-DOF robot path/trajectory based on sensing;
- *anytime* and *parallel planning* with optimization, as inspired by evolutionary computation [6], through maintaining and repeatedly updating/improving a set of trajectory candidates for a robot from its current configuration to a goal configuration;
- great structural flexibility to allow for both on-line adaptation to different environmental scenarios and off-line extension to robots of very different nature.

All major components of the RAMP algorithm can be customized. The strength of RAMP lies in both its generality and its flexibility for adaptation and extension. Indeed, it has recently been extended to real-time continuum manipulator motion planning [102].

RAMP always maintains a set of diverse trajectories in the CT-space of the robot, called a *population*. The initial population of trajectories can be formed randomly. Each trajectory starts from the robot's current configuration and ends at the goal configuration and may be only partially *feasible* – defined as both collision-free and singularity free. A partially feasible trajectory is one that has a beginning feasible segment followed by an infeasible segment. The quality of a trajectory, in terms of feasibility and optimality, is evaluated through a *fitness evaluation function* that

combines optimization criteria, such as shortest overall time, maximum time of the feasible segment, and so on.

As soon as a trajectory has a feasible segment starting from the robot's current configuration, RAMP allows the robot to move along it while planning for subsequent feasible trajectory segments simultaneously so that the robot can switch to the best subsequent feasible trajectory segment as it finishes the current feasible one. Three repeated cycles of processes are run simultaneously in the classical RAMP:

- Sensory data are updated in each *sensing cycle*.
- Trajectory modification and evaluation (or re-evaluation) based on sensing data is conducted in each *planning cycle*.
- The robot switches to a better trajectory from the currently executed one in each *control or adaptation cycle*.

Key to RAMP is efficient on-line detection of feasible trajectory segments of candidate trajectories. The original RAMP assumes known obstacle geometry and conducts collision checking based on predicting obstacle motions. It was also only implemented in simulation.

8.2 E-RAMP as practical motion planner

The CFPA takes a finite time to detect a collision-free trajectory segment through detecting collision-free CT points. The actual time for CFPA to detect a collision-free CT point $\chi = (\mathbf{q}, t)$ depends on two factors:

1. the size of the dynamic envelope $E(\chi, \tau)$, which is decided by $v_{max}(t - \tau)$ and

shrinks as τ increases, and

2. the processing power of the computer and sensors.

Factor (1) is usually the dominating factor. Let τ_0 be the time to start observing and checking if $(\mathbf{q}, t)$ is collision-free. If $E(\chi, \tau)$ is free of atomic obstacles by time $\tau_1 < t$, then CFPA takes at least the time duration $\tau_1 - \tau_0$ to detect that χ is collision-free, even if the computation cost of detection, i.e., factor (2), is omitted.

Hence, the combined time of CFPA and RAMP to decide a subsequent feasible trajectory segment can be longer than the time period of the feasible trajectory segment that the robot executes. Therefore, a subsequent feasible trajectory segment may not be found when the robot finishes executing the current segment, resulting in a *forced stop* of the robot. During such a forced stop, the robot may be hit by obstacles and is thus unsafe.

Therefore, it is important that we extend the motion planner RAMP to minimize forced stops.

8.2.1 Algorithm

The parallelism and flexibility of RAMP enables us to do so in the following ways:

- We add to the evaluation function of RAMP, as an additional optimization criterion, maximizing the safe time δt_{safe} for the robot to pause at the end CT point $(\mathbf{q}_e, t_e)$ of a collision-free trajectory segment without being hit. With this added optimization criterion, RAMP is able to select a feasible trajectory segment (among all feasible trajectory segments found) that maximizes the total time $\Delta t_{safe} = \Delta t_{move} + \delta t_{safe}$, where Δt_{move} is the time period of the segment.

Note that δt_{safe} at CT point $\chi_e = (\mathbf{q}_e, t_e)$ cannot be known precisely *before* the robot reaches χ_e, but it can be (under)estimated as $d_{min}(\mathbf{q}_e, \tau)/v_{max} - (t_e - \tau)$ for $\tau < t_e$, where $d_{min}(\mathbf{q}_e, \tau)$ is the sensed minimum distance at τ between the robot and the atomic obstacles. Note also that since δt_{safe} is meant for a collision-free CT point, τ is greater than the time τ_e when the CT point χ_e is found collision-free. We developed a method to obtain $d_{min}(\mathbf{q}_e, \tau)$ among atomic obstacles near the robot at χ_e.

- We separate collision-checking using CFPA from evaluation of the fitness of a trajectory, rather than embedding collision-checking into the fitness evaluation. Collision-checking is ran constantly in the background and provides the E-RAMP with the information of the collision-free segment of a trajectory, while fitness evaluation simply uses the information to compute the value of the fitness function, which is much faster and almost instant.

We call the extended RAMP algorithm, the E-RAMP, as illustrated in Algorithm 6.

In Algorithm 6, there are four simultaneous threads for *sense*, *collision check*, *plan* and *adapt*, and *move* respectively. The main **while** loop describes adaptation cycles. Each adaptation cycle consists of multiple planning cycles. Each sensing cycle lasts $\delta\tau$, determined by the planner, such that sufficient new information can be obtained in each sensing cycle for the CFPA (Algorithm 5) to use, which is called by the collision-checking algorithm (Algorithm 7).

In each adaptation cycle, the robot simultaneously moves along the feasible segment of Γ_{best} and plans for its next subsequent feasible segment, which starts from the end

Algorithm 6 E-RAMP

$m \leftarrow$ # of sensing cycles in a planning cycle
$n \leftarrow$ # of planning cycles in an adaptation cycle
$\Delta t_{min} \leftarrow$ small constant time
$\mathbf{q_e} \leftarrow$ starting configuration of the robot
$t_e \leftarrow$ current time τ $\{\tau$ is the clock time of the system that automatically updates$\}$
initialize a set S of trajectories connecting the start configuration to the goal configuration of the robot
$\Delta t_{move} \leftarrow 0$
$\delta t_{safe} \leftarrow 0$

while the robot has not reached the goal **do**

simultaneously *sense*, *collision check*, *plan* and *adapt*, and *move*:

sense: repeat sensing cycles

plan at every m-th sensing cycle or when robot stops
if $t_e < \tau$ **then**
$\Delta t \leftarrow \max\left(\Delta t_{move} + \delta t_{safe}, \Delta t_{min}\right)$
$t_e = \tau + \min\left(mn\delta\tau, \Delta t\right)$
Set starting time of all trajectories in S to t_e
end if
modify S

adapt: when $\tau = t_e$ or at every mn-th sensing cycle
evaluate trajectories in S
$\Gamma_{best} \leftarrow$ best trajectory
$\mathbf{q_e} \leftarrow$ last configuration on the first collision-free trajectory segment of Γ_{best}
$\Delta t_{move} \leftarrow$ the time taken by the robot to move to configuration $\mathbf{q_e}$
$t_e \leftarrow \tau + \Delta t_{move} + \delta t_{safe}$
if $\Delta t_{move} + \delta t_{safe} = 0$ **then**
flag "unsafe" (robot is at risk of collision)
end if

collision check: call Algorithm 7

move: move the robot along Γ_{best} the time period Δt_{move}
end while

Algorithm 7 collision-checking

1: input trajectory segments of N trajectories in S, where each trajectory segment i, $0 < i \leq N$, is a sequence of CT points $\chi_1^i, \chi_2^i, ...$ with time duration of $m \times n \times \delta\tau$
2: $C \leftarrow$ a sequence of CT-points in order $\chi_1^1, \chi_1^2, ..., \chi_1^N, \chi_2^1, \chi_2^2, ..., \chi_2^N, ...$
3: run CFPA (Algorithm 5) for every CT point in C until S is updated
4: report collision-free CT points found in each trajectory; compute and report δt_{safe} for the end collision-free CT point in each trajectory.

CT point $\chi_e = (\mathbf{q}_e, t_e)$ of the feasible segment of Γ_{best}. If, when the robot reaches χ_e, no subsequent collision-free segment is found (i.e., not a single new CT free point is found), then the robot will stop its motion while continuing simultaneous collision-checking and planning until it finds a collision-free segment. If the time period that the robot stops is less than δt_{safe}, it means that the robot stopped safely at χ_e to continue planning for a while; otherwise the robot is forced to stop longer at χ_e at the risk of being hit by an obstacle.

The constants m and n are decided based on v_{max} of obstacles and the size of the environment.

The following subroutines are used in Algorithm 6, in addition to Algorithm 7 for collision check:

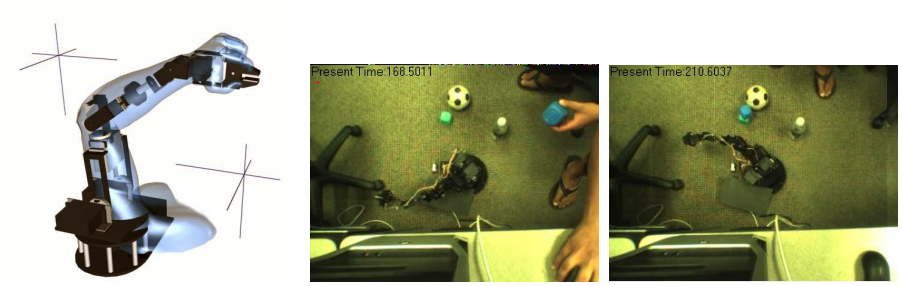
- *initialize* a set of trajectories as in [92], through randomly creating intermediate knot configurations between the start and the goal configurations.

- *evaluate* the fitness function value for each trajectory, which is a cost function to both maximize the time of the feasible trajectory segment and minimize the total time of the trajectory.

- *modify* a randomly picked trajectory in S to change its shape via adding/deleting or changing coordinates of knot configurations or CT points, evaluate the new trajectory, and use it to replace a non-best trajectory in S.

8.2.2 Implementation and experiments

We applied the E-RAMP to plan motions of a real 7-DOF Robai's Cyton arm (see Figure 33(a)), using an indoor Point Grey's Digiclops stereo vision camera for overhead sensing and a DELL Precision T5400 computer with four cores and 4 GB RAM. Each robot link is approximated by an oriented bounding box. Each revolute joint has a maximum speed of 90 (deg/s) in both directions. To eliminate noise in sensing data, the image pixels were classified as (a) pixels of the robot and its accessories (i.e., robot circuit board, battery, etc.), identified by the colors, and (b) obstacle pixels, by checking if they are in robot workspace. The software was built using the latest .NET 4.0 framework, and the program was done in C#.



(a) A 7-DOF Cyton arm. (b) The robot at q_s in an environment. (c) The robot at q_g.

Figure 33: An experimental environment with the stereo-vision sensor.

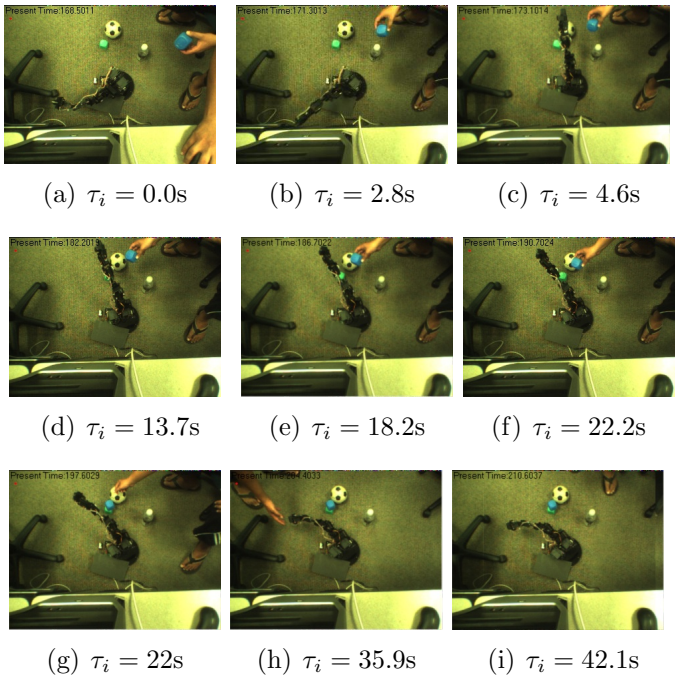
We conducted six experiments with the robot in an environment consisting of a number of random obstacles such as football, blocks, table, plastic covers, half-filled water bottles, as well as a person moving those objects. The objects (except for the

Table 6: Planner and Task Parameters

$\|S\|$	v_{max}	Δt_{min}	$\mathbf{q}_s$, $\mathbf{q}_g$	Sensor Resolution	$\delta\tau$
5	1 cm/s	0.5s	$\{-45°, -45°, -45°, 0°, -45°, -45°, -45°\}$ $\{45°, 45°, 45°, 45°, 45°, 45°, 45°\}$	320 × 240	0.05s

Table 7: Experiments

Experiment #	1	2	3,4	5,6
Obstacle moved	Blue block	Half-filled water bottle	Toy soccer ball	Plastic cover



(a) $\tau_i = 0.0$s (b) $\tau_i = 2.8$s (c) $\tau_i = 4.6$s

(d) $\tau_i = 13.7$s (e) $\tau_i = 18.2$s (f) $\tau_i = 22.2$s

(g) $\tau_i = 22$s (h) $\tau_i = 35.9$s (i) $\tau_i = 42.1$s

Figure 34: Snapshots of experiment #1 with a blue obstacle.

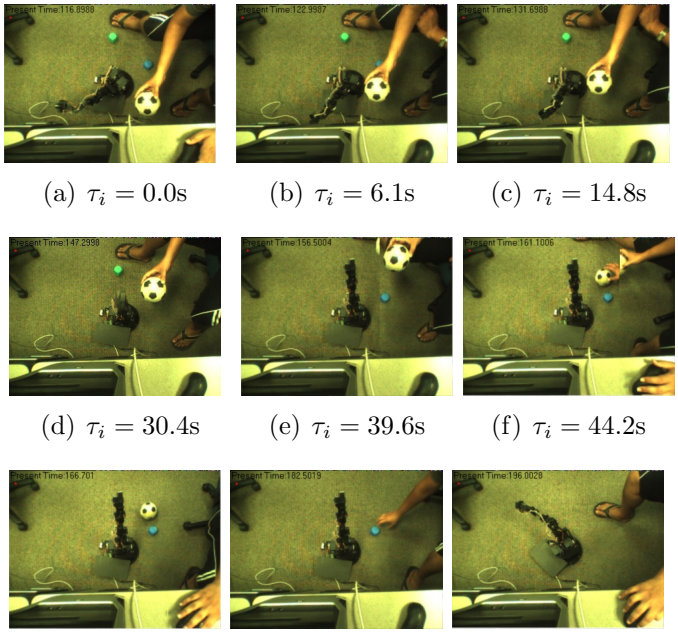
table) are moved by the person to create arbitrary motions unknown to the robot. Figures 33(b) and 33(c) show the starting configuration $\mathbf{q_s}$ and the goal configuration $\mathbf{q_g}$ of the robot, which values are shown in Table 9.1. Note that in order to reach $\mathbf{q_g}$ from $\mathbf{q_s}$, the robot end-effector cannot follow the straight-line path in the workspace because of the joint limitations. Table 9.1 shows the common parameter values used in those experiments. Except for v_{max}, which characterized the environment, the

other parameter values were determined empirically based on the processing speed of the planner.

In these six experiments, different obstacles were moved by a person in more or less the same way during the robot's motion from the common $\mathbf{q_s}$ to $\mathbf{q_g}$, as shown in Table 7. These six experiments used three different sets of m and n values. Each set of m and n values were shared by a pair of experiments, as shown in Table 8 and Table 9, which we will discuss later.

We now describe one experiment for each set of m and n values.

In experiment #1 (see Figure 34), the person moved the blue block to approach the robot (see snapshots 34(a)–34(c)). Then the robot tried to avoid it and other obstacles while moving towards the goal as the person moved the block closer to the robot (shown in snapshots 34(d)–34(i)) and finally reached the goal (see 34(i)).



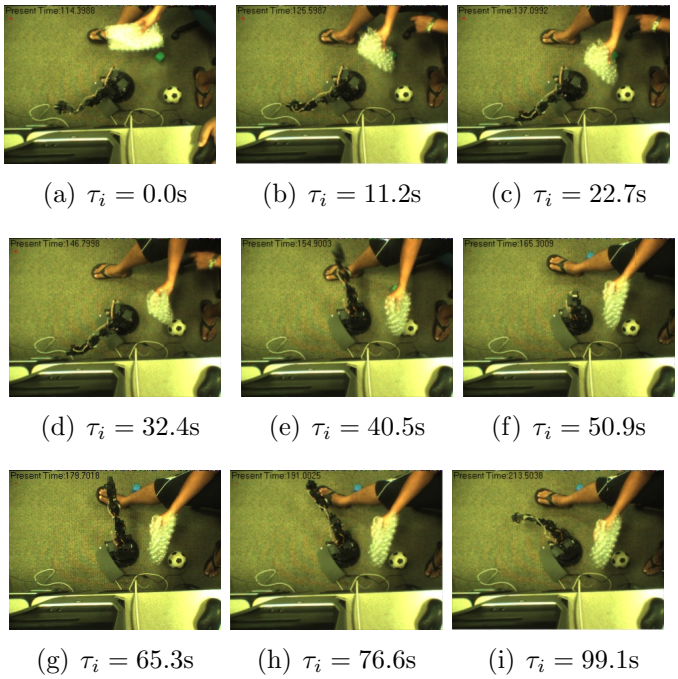
(a) $\tau_i = 0.0$s (b) $\tau_i = 6.1$s (c) $\tau_i = 14.8$s

(d) $\tau_i = 30.4$s (e) $\tau_i = 39.6$s (f) $\tau_i = 44.2$s

(g) $\tau_i = 49.8$s (h) $\tau_i = 65.6$s (i) $\tau_i = 82.3$s

Figure 35: Snapshots of experiment #3 with a soccer ball as an obstacle.

In experiment #3 (see Figure 35), initially the robot encountered the soccer ball moved by the person and performed a motion by bending half of the arm and then moved away from both obstacles (see snapshots 35(a)–35(c)). As the person moved the soccer ball towards the robot (similar to experiment #1), the robot started moving away from it (see snapshots 35(d)–35(i)) and successfully reached the goal as shown in Figure 35(i).



(a) $\tau_i = 0.0$s (b) $\tau_i = 11.2$s (c) $\tau_i = 22.7$s

(d) $\tau_i = 32.4$s (e) $\tau_i = 40.5$s (f) $\tau_i = 50.9$s

(g) $\tau_i = 65.3$s (h) $\tau_i = 76.6$s (i) $\tau_i = 99.1$s

Figure 36: Snapshots of experiment #5 with a plastic cover as an obstacle.

In experiment #5 (See Figure 36), initially the robot encountered the plastic cover moved by the person and performed a motion by bending half of the arm to stay away from the plastic cover (see snapshots 36(a)–36(c)). Later, as the plastic cover moved closer to the robot, the robot moved away from the table and near the plastic bag while staying away from the person's leg (see snapshots 36(d)–36(f)). While the plastic cover kept approaching the robot; the robot was able to find a motion to reach

the goal successfully (see snapshots 36(f)–36(i)).

8.2.3 Results and main experimental insights

The results for the first four experiments performed are shown in Table 8 and the remaining two are shown in Table 9. As it shows, there are unsafe stops in all cases. However, the data show that by increasing n, i.e., increasing the number of planning cycles per adaptation cycle, both the number of unsafe stops and the average duration of an unsafe stop decreased. Having multiple planning cycles within an adaptation cycle (i.e. $n > 1$) leads to finding on average a longer feasible path segment of Γ_{best}. Note that since we assign a constant speed for the robot, the time duration Δt_{move} of a feasible segment is proportional to its path length.

It can be seen that the # adaptation cycles during which the robot moved was smaller than the minimum total # adaptation cycles (which is the total # planning cycles divided by n), and in some cases far smaller. This shows that there were more stops than the total # unsafe stops, meaning that there were safe stops and in some cases a lot of them, as in experiments 2, 3, and 5. Note that the # of δt_{safe} computations show the number of end points of collision-free trajectory segments, that is, the number of collision-free segments. This number is smaller for cases with longer average path length of feasible segments.

As shown in Table 8 and Table 9, the time needed to do one iteration in CFPA for finding a collision-free CT point (i.e., CT-free point) on average ranged from 3 to 15 sensing cycles. Since many CT-free points in the experiments require two iterations to be found (i.e., based on sensing data of two different sensing instants), the total

time for finding a CT-free point is even longer. In spite of this high cost of CFPA, the Algorithm 6 was able to lead the robot to its goal in those experiments successfully.

However, for a greater v_{max}, one iteration in CFPA will take even more time because the dynamic envelope is larger. Thus, faster computation for the intersection check in CFPA is necessary, which remains one of our on-going research topics.

Table 8: Experimental Results

Parameters		m = **20**, n = **1**		m = **10**, n = **3**	
Experiment		1	2	3	4
Total time (s)		42.09	83.99	82.29	137.99
Total # planning cycles		21	42	12	128
# adaptation cycles that the robot moved		20	22	3	9
Total # unsafe stops		7	9	2	5
Duration of an usafe stop (s)	Avg.	0.69	0.48	0.1	0.33
	Min.	0.1	0.1	0.1	0.1
	Max.	1.3	1.3	0.1	0.4
Path length of feasible seg. of Γ_{best} (deg)	Avg.	34.01	26.45	119.05	47.62
	Min.	13.22	13.22	66.14	13.22
	Max.	145.51	79.37	171.97	92.60
Total # CT-free points found		97	618	64	521
Time for one iteration in CFPA that found a CT-free pt. (ms)	Avg.	712.22	637.08	156.51	184.66
	Min.	109.375	15.62	109.37	46.87
	Max.	11625	5109.37	343.7	859.37
Total # of δt_{safe} **computations**		83	196	51	452
Time for computing one δt_{safe} (ms)	Avg.	681.48	1850.153	173.99	178.47
	Min.	31.25	140.62	109.37	15.62
	Max.	2656.25	13828.13	281.25	718.75

8.3 Summary

This chapter extended the Real-time Adaptive Motion Planner (RAMP) to accommodate CFPA and also provided sufficient time to the motion planner for coming up with the near-optimal next collision-free trajectory. Real experiments were demonstrated to show that the robot made few unsafe stops while moving amongst unknown

Table 9: Experimental Results

Parameters		m **= 20,** n **= 2**	
Experiment		5	6
Total time (s)		99.09	86.49
Total # planning cycles		20	8
# adaptation cycles that the robot moved		5	3
Total # unsafe stops		3	2
Duration of	Avg.	0.16	0.1
an usafe stop (s)	Min.	0.1	0.1
	Max.	0.2	0.1
Path length of	Avg.	79.37	119.05
feasible seg. of	Min.	26.45	66.14
Γ_{best} (deg)	Max.	171.97	171.97
Total # CT-free points found		200	98
Time for one	Avg.	137.89	142.49
iteration in CFPA	Min.	15.62	93.75
that found a CT-free pt. (ms)	Max.	406.25	343.75
Total # of δt_{safe} **computations**		100	31
Time for	Avg.	281.40	192.02
computing one	Min.	109.37	125
δt_{safe} (ms)	Max.	1671.87	312.5

and unpredictable obstacles.

CHAPTER 9: EXPERIMENTS AND RESULTS

Using the notion of dynamic envelope and atomic obstacles, we have shown on how to perceive guaranteed collision-free trajectories in environments that are unknown and unpredictable. To test the effectiveness and practicality of our approaches we conducted experiments in motion planning of different robots in simulation environments and real-world environments. The robot is initially at a collision-free configuration $\mathbf{q_s}$ and needs to move to a goal configuration $\mathbf{q_g}$ amongst unknown obstacles that move unpredictably. The only given input about the environment is v_{max}, which can be over-estimated.

9.1 Performance data

The following performance data are gathered for testing our approach:

- **Total time** (T): the total time for the robot to plan and move simultaneously from a start configuration $\mathbf{q_s}$ to a goal configuration $\mathbf{q_g}$.

- **# Stops** ($\#S$): the number of stops the robot is forced to make during its journey from $\mathbf{q_s}$ to $\mathbf{q_g}$. The robot makes a forced stop when it reaches the end of the current collision-free motion segment but has not found the next one for execution. It resumes its motion when a new collision-free motion segment is found.

- **# Hits**: the number of forced stops when the robot got hit by obstacles. As will be shown in the simulation, the robot may only get hit by an obstacle during a forced stop.

- $\mathbf{L}_f$: the total joint-space length of a feasible trajectory segment found by CFPA.

- $\mathbf{T}_f$: the time duration of a feasible trajectory segment found by CFPA.

- **# AO**: the number of atomic obstacles checked by CFPA to decide if a dynamic envelope $E(\chi, \tau_k)$ for a CT-point χ contains atomic obstacles or not.

- $\mathbf{T}_c$: the time required to check if a dynamic envelope $E(\chi, \tau_k)$ of a CT-point χ contains any atomic obstacle.

9.2 Simulation environment

We tested using the concept of dynamic envelopes for collision detection in an unpredictable environment in simulation. We used two robots, a simple planar rod robot, and a planar continuum manipulator in our testing.

9.2.1 With 2-DOF rod robot

We have conducted simulations for a planar rod robot, which can only translate on a plane having a fixed orientation $\theta = 45°$, i.e., the robot has two translational degrees of freedom. The reference frame for the robot, which is aligned with the world frame, has its origin at an end point of the rod. The robot is initially at a collision-free configuration $\mathbf{q_s}$ (shown as S) and needs to reach a goal configuration $\mathbf{q_g}$ (shown as G) in Figure 37. The environment consists of unknown obstacles of arbitrary shapes that are perceived as a set of identical red circles at any sensing moment, which are

atomic obstacles. The actual obstacles are either static, or can move randomly with changing speeds no greater than v_{max} units/s. Further, the robot can over-estimate v_{max} as $v'_{max} > v_{max}$. The sensing frequency is 20 Hz. The simulation was conducted on a Dell Optiplex GX620.

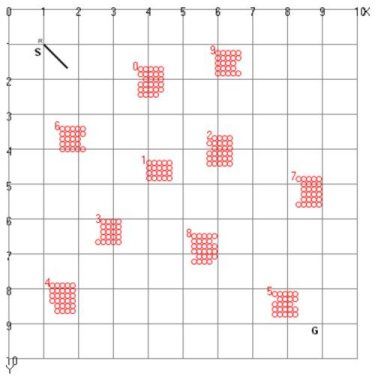
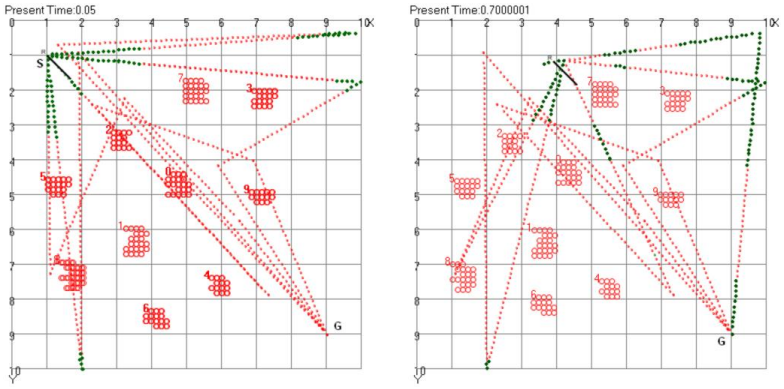


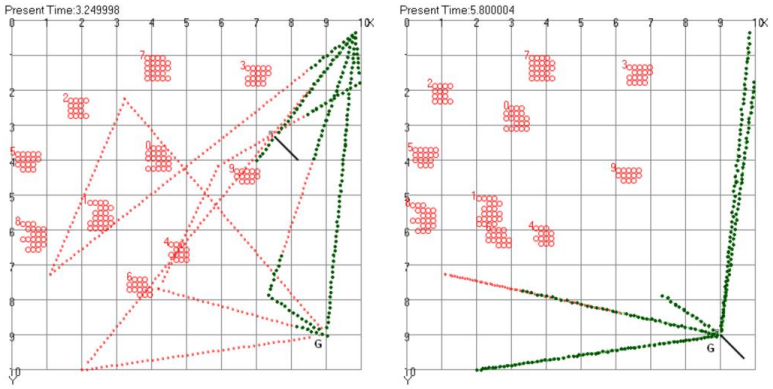
Figure 37: Simulation environment

Figure 38 shows the snapshots of an example run, when the rod robot of unit length moves from the starting configuration $\mathbf{q_s} = (1, 1)$ to the goal configuration $\mathbf{q_g} = (9, 9)$ with speed 5 units/s. Although the obstacles can change their velocities instantly, the maximum speed each obstacle can have is $v_{max} = 1$ unit/s. The sequences of green (or dark in B/W) reference positions show the perceived collision-free motion segments (without showing the time instants). The sequences of red (or light in B/W) reference positions indicate uncertain motion segments at each moment of perception, which may or may not be collision-free. The robot executes the best green option found. Note that the robot never hits an obstacle *while moving along a green trajectory* because it is guaranteed collision-free.

As v_{max} increases, obstacles move faster on average, and it is more likely that the robot gets hit during forced stops. Over 30 tested runs, the average #hits for the



(a) $\tau_i = 0.05$s (b) $\tau_i = 0.7$s

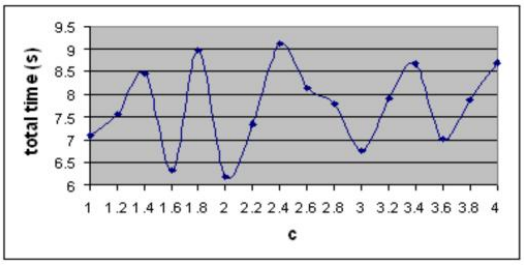


(c) $\tau_i = 3.25$s (d) $\tau_i = 5.8$s

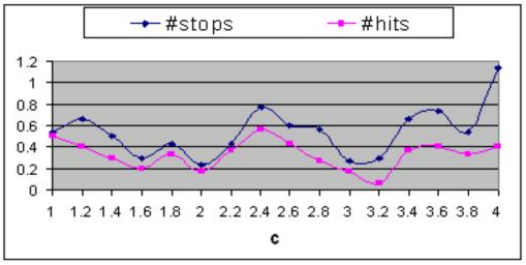
Figure 38: Snapshots of an example run in simulation for $v_{max} = 1$ unit/s.

case $v_{max} = 1$ unit/s and the case $v_{max} = 5$ unit/s are 0.4 and 2 respectively.

We also tested the effects of overestimating the speed bound v_{max} of obstacles in the same environment of the example run, i.e., $v'_{max} = cv_{max}$, $c \geq 1$. Figure 39 shows the average results over 30 runs for each c. The results show that c, indicating the level for over-estimation of v_{max}, can be increased in a wide range (until $c = 3.8$) without affecting much the performance parameters. Within the range, the ups and downs in the curves reflect the randomness in the environment. This is because although a greater v_{max} estimate results in a larger dynamic envelope, the envelope also shrinks faster (i.e., with a faster speed), and so there is a balance for a certain range of c.



(a) c vs. avg. total time (sec) over 30 runs



(b) c vs. avg. # hits and avg. # stops over 30 runs

Figure 39: Effects of over-estimating v_{max} as $v'_{max} = cv_{max}$, $c \geq 1$.

As can be seen in Figure 39, once c increases more than 4, it seems that shorter collision-free segments were found by the CFPA so that #stops and #hits increased.

We also tested our approach for a static environment, i.e., $v_{max} = 0$, containing a narrow passage as shown in Figure 40: the robot had to move through the narrow passage to reach the goal as shown by every position of the path. We performed experiments by varying the level of over-estimation $v'_{max} \in [1, 4]$. In all cases, the travel time for the robot from the start to the goal position was constant: 2.49s.

9.2.2 With continuum manipulator

We have tested our algorithm on a planar continuum robot in simulation environments. While a continuum robot can in fact have an infinite number of degrees of freedom because it is deformable, there are only a finite number of controllable degrees of freedom when the robot is not in contact. These are the degrees of freedom

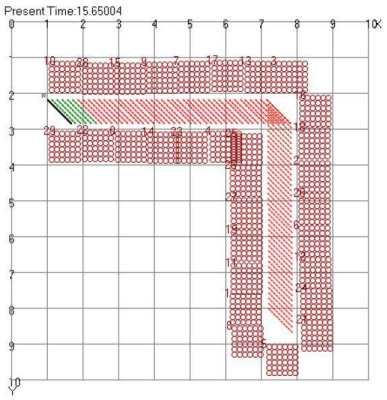


Figure 40: An example of static narrow passage.

that can be directly changed by the actuators. The continuum manipulator designed at Clemson University has three sections. According to [43], each section $i, i = 1, 2, 3$ is a part of a circle with two end points: a *base* point p_{i-1} and a *tip* point p_i. The base of the robot is set at p_0 with z_0 axis tangent to section 1's curve. The section i's frame is formed at p_{i-1} with the z axis tangent to the section curve at p_{i-1}. The base of section i is the tip of section $i-1$. Each section has three controllable variables: curvature κ_i (which can be either negative or positive), length s_i, and rotation angle ϕ from axes y_{i-1} to y_i about z_{i-1}. Note that the circle center of section i, p_{ic}, always lies on the x_i axis.

A configuration $\mathbf{C}$ of the continuum manipulator can be expressed by the controllable variables as $[\kappa_1, s_1, \phi_1, \kappa_2, s_2, \phi_2, \kappa_3, s_3, \phi_3]^T$. Thus, we can treat this (κ, s, ϕ) space the *configuration space* of the continuum robot. Given the position of the base point p_{i-1}, and the κ_i, s_i, and ϕ_i values, the position of the tip point p_i of the section can be computed [43].

If ϕ_i=0 for all the sections, the continuum is planar. Figure 41 shows a planar three continuum and an example section. Figure 41(a) shows an example section of a

continuum manipulator, and (b) shows a continuum manipulator with three sections.

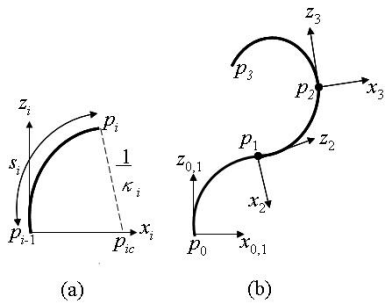
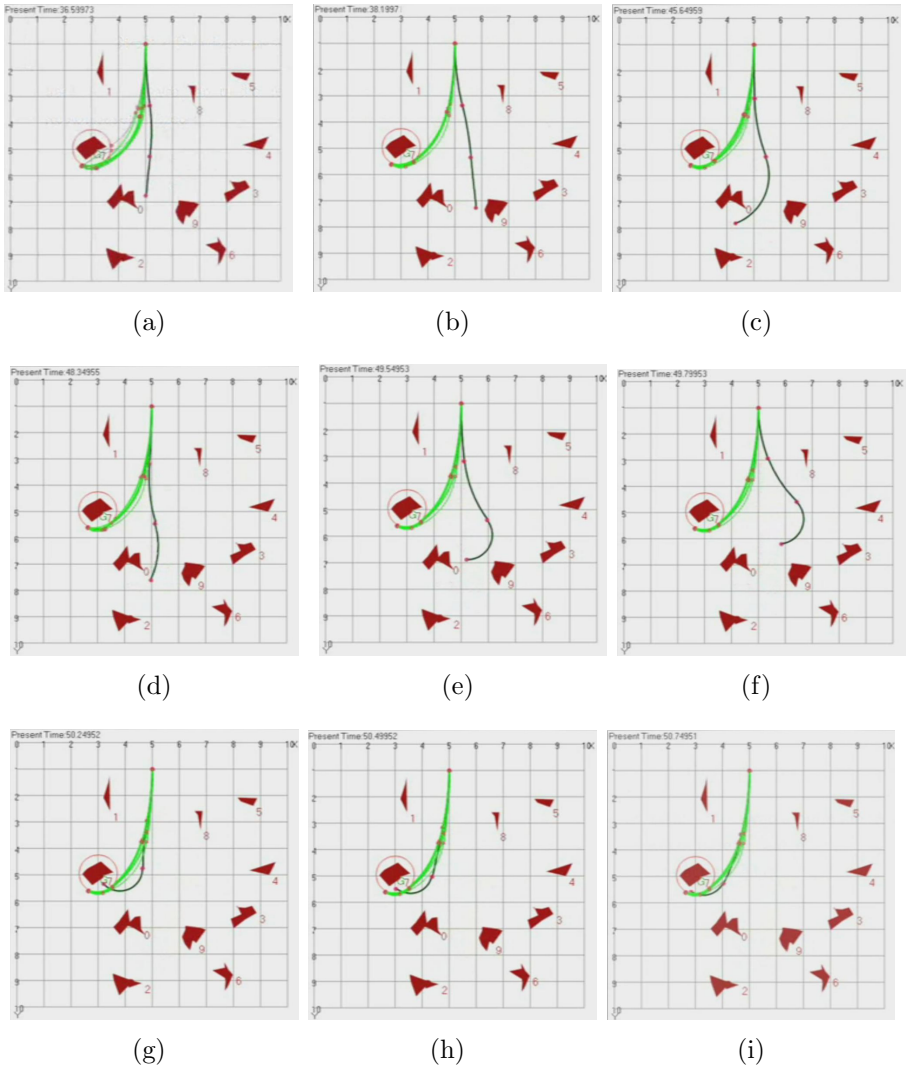


Figure 41: A planar continuum manipulator.

Each environment consists of randomly generated polygonal obstacles of arbitrary shapes and sizes. It also includes a randomly generated target object for the robot, also in arbitrary shape and size (in the wide range that the robot can "grasp" it). The obstacles can be either static or move randomly. The simulated robot has the same value ranges on (κ, s) for each section as the actual robot at Clemson University. It can have either a static base or a base that translates horizontally. The simulation was conducted on a Dell Optiplex GX620. In the simulation environment, we assume that the poses of both the target object and the obstacles are sensed in sensing cycles, with a frequency of 20 Hz. Even though the obstacles may move randomly, they will not run over a stopped robot. This essentially assumes that the obstacles, which can be either moved by people or are other robots, are not malicious. Our robot is capable of avoiding others during its motion, but when it is static, others are not assumed to harm it.

We assume the robot can move with a constant speed with instant acceleration/deceleration to simplify trajectory generation. We also ignore the width of the continuum robot for simplicity.



(a) (b) (c)

(d) (e) (f)

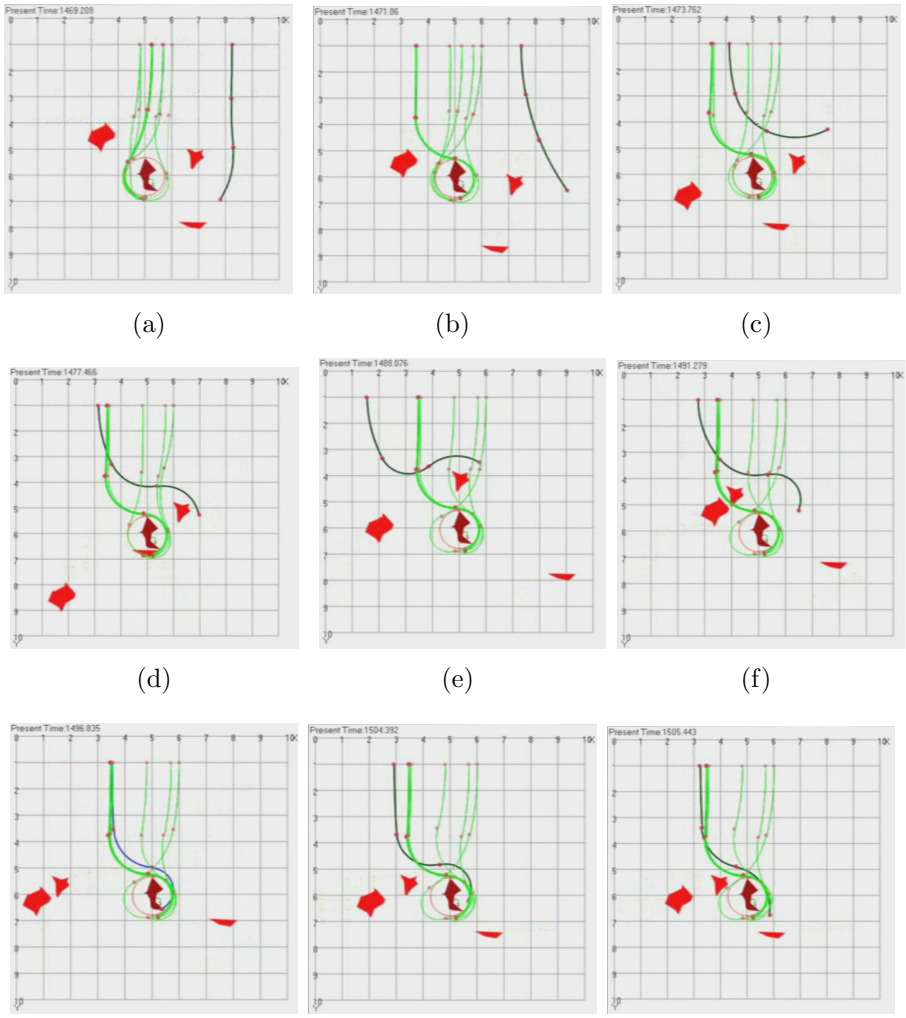
(g) (h) (i)

Figure 42: Experiment with continuum manipulator in static environment.

We applied our real-time planner to different simulation environments of randomly generated obstacles. In the graphical display, 1 unit=15cm in real world. The upper bound on the obstacle speed is $v_{max} = 1$ unit/s. The robot speed is greater than 1 unit/s.

Figure 42 shows the snapshots of continuum manipulator with a fixed-base planning its motion in a static environment. Figure 43 and Figure 44 shows the snapshots of a robot with a horizontally moving base in two dynamic environments with unknown

obstacle motions: in the first environment (task 1) with three dynamic obstacles and in the second dynamic environment (task 2) with three static and six dynamic obstacles. In all cases the target object is indicated by the bounding circle. Also, for both the cases, the robot is initially in the vertical configuration stretched (i.e., with zero curvature for all sections).



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 43: Experiment with continuum manipulator (task 1).

Table 10 shows the results averaged over 20 runs for task 1 and task 2 respectively. The results show that there are just a few forced stops in both cases (when the robot

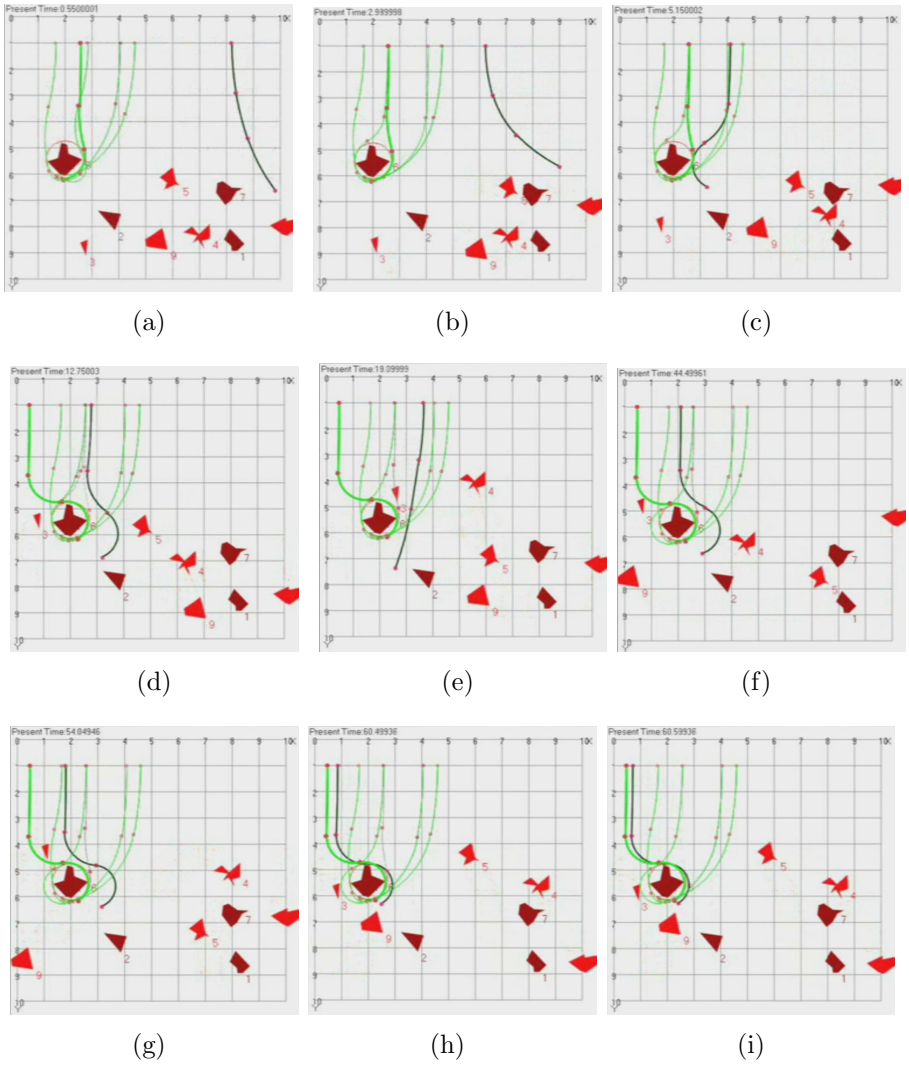


Figure 44: Experiment with continuum manipulator (task 2).

did not find a collision-free segment to follow).

9.3 Real experiments

Extensive testing was done in simulation environments with the assumption that sensor data are accurate and geometry of obstacles are simple 2D shapes. Thus the cost of collision-detection was negligible. However, in real environments for robot manipulators the obstacles have to be represented by 3D atomic obstacles and the computational resources available are limited for real-time operations. For 5-DOF

Table 10: Average Results of 20 Runs of Each Task

Tasks	Total time(s)	#Stops
1	28.74	3.05
2	55.41	2.35

Robix Rascal robot a low-end PC was used to test the feasibility of approach, whereas with 7-DOF Cyton arm comparatively high end PC was used, having four CPU cores.

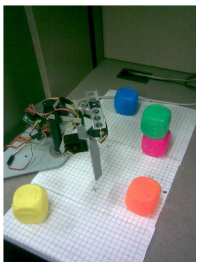
9.3.1 With 5-DOF Robix Rascal robot

We have also tested the CFPA by embedding it in a simple real-time motion planner for a real desktop 5-DOF robot manipulator with revolute joints in an unknown and unpredictable environment, sensed via an overhead stereovision sensor (Figure 45). Our real-time motion planner finds a collision-free straight-line segment in the CT-space as the next-step motion for the robot to execute, with a search method compromising randomized and greedy search[1] and using CFPA for discovering collision-free motion. As the robot moves, the planner simultaneously finds again the subsequent next step until the goal is reached.

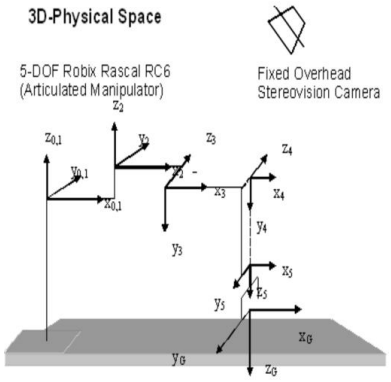
The planner was implemented in C++ on a low-end PC (Dell Optiplex GX260). The 5-DOF manipulator is made from the Robix Rascal RC6 kit. The stereo vision camera is PGR's Digiclops. The obstacles are blocks unknown to the robot, which can be moved in ways also unknown to the robot. Table 1 shows the input parameter values to the planner, where **S** and **G** are the starting and goal configurations respectively. $\dot{q}_{-ve}$ and $\dot{q}_{+ve}$ are the negative and positive bounds on the joint speeds

[1]In this way the planner is able to overcome local minima, but the details of search and handling local minima is not the focus of this research since a number of different strategies can be used.

of the robot. Note that the number of atomic obstacles of an actual obstacle increase if the obstacle is close to the origin of the camera frame.



(a) Robix Rascal RC 6 and obstacles of unknown geometry in its workspace.



(b) Kinematics of the manipulator.

Figure 45: Experimental setup for Robix Rascal RC 6.

The atomic obstacles representing the robot itself and the known desk surface (as "floor") were filtered out. The shape of an atomic obstacle was approximated as a straight-line ray. The shape of a link of the robot was simplified by a cylindrical bounding volume. The number of atomic obstacles in the test environment were in the range of 345–800. The average rate of collision checking in the CFPA computation was 1430.64 CT-points/second.

Table 11: Input Parameters and Values

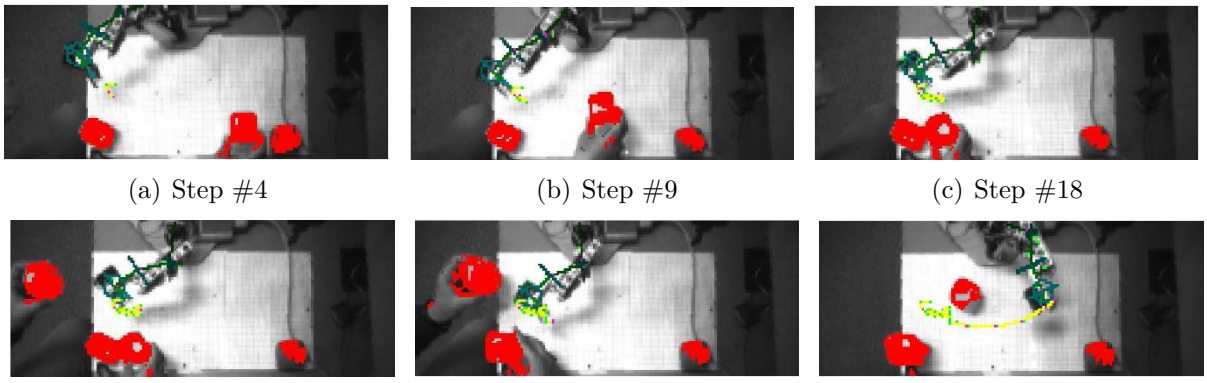
S, **G** (degrees)	v_{max} cm/sec	$(\dot{q}_{-ve}, \dot{q}_{+ve})$ (degrees/sec)	*Sensor Image resolution*	$min\#O(i,j)$ *per obstacle*
$[-70, 45, 0, 0, 0]^T$ $[70, -45, 0, 0, 0]^T$	1	$([-6, -6, -5, -6, -7]^T,$ $[\ 6,\ 6,\ 5,\ 6,\ 7]^T)$	160×120	115

Figure 46 shows a test environment and two different resulting paths that the robot traveled. The environment had 4 blocks as obstacles, where two were placed at the

(a) Path 1 (b) Path 2

Figure 46: An environment (Env1) and two traveled paths by the robot.

corners and two were stacked together to form a taller obstacle in between. The taller obstacle created a local optima for the given robot structure with limited dexterity, which our planner was able to overcome.



(a) Step #4 (b) Step #9 (c) Step #18

(d) Step #19 (e) Step #25 (f) Step #27

Figure 47: Selected steps taken by the robot in Env2.

Figure 47 shows a sequence of selected snapshots of the robot motion in another test environment, where there are four obstacles, and two of them are dynamic, moved by the two hands of a human operator. Note that a grid of 1×1 cm^2 squares on the desk was used as a guidance to move obstacles close to $v_{max} = 1$cm/s. As shown, the operator first moved one block towards the robot. In 47(a), robot is near the configuration **S** and in (f), robot is at configuration **G**.

Between step 9 to step 18, the planner tried to get most of links closer to their goal positions while avoiding the moving block and the block at the bottom left corner.

After step 18, the moving block decreased its speed, and a new block was moved into the visible robot workspace. The planner noticed the reduced speed of the first moving block in time due to the non-conservative nature of the dynamic envelopes and simply guided the robot to pass by the moving block and the static block, while moving away from the newly entered block to reach the goal in step 27. Table 2

Table 12: Average results from two environments (for the same start and goal configurations of the robot)

Env	**Path length** (deg)	**#Steps**	**Total time** (*sec*)	**#Sensing cycle** (*Hz*)
Env1	514.924	62.2	82.8	3.5
Env2	235.98	27	49	4.58

shows the resulting statistics characterizing the planner performance in the two task environments.

9.3.2 With 7-DOF Cyton Arm robot

We have conducted real-world experiments using a real 7-DOF Robai's Cyton arm (see Figure 48), an indoor MobileRanger's stereo vision camera (used overhead), and a DELL Precision T5400 computer. We have implemented the full-fledged CFPA here, using realistic models of atomic obstacles (see Figure 49) rather than treating them as rays[2].

The robot has 7 revolute joints. Each robot link is approximated with a oriented bounding box. Each revolute joint has a maximum speed of 90 (deg/s) in both directions and a constant acceleration or de-acceleration of 57 (deg/s^2).

Each unknown object in the environment was perceived as a set of atomic obstacles

[2]In the simple experiments with a 5-DOF cyton arm robot reported earlier, atomic obstacles were treated as rays.

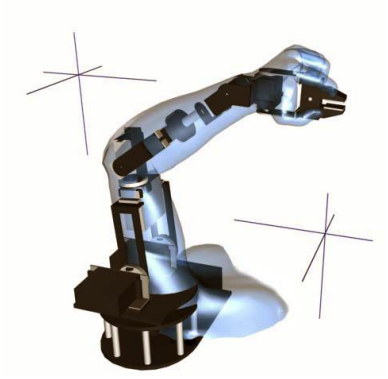


Figure 48: A 7-DOF Cyton arm.

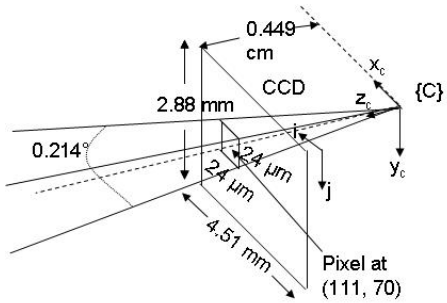


Figure 49: Dimension of atomic obstacles for resolution 188 × 120.

at each sensing instant. The dimension of an atomic obstacle corresponding to a pixel is shown in Figure 49. The sensor resolution was fixed to 188×120 and thus, at most 22,560 atomic obstacles were generated from each sensing moment. The sensing frequency was in the range of 16-20 Hz. As the stereo vision camera provides highly inaccurate data for objects with low image intensity values, only objects with high intensity values were used as obstacles to the robot.

In this implementation with a real sensor and a real high-DOF robot the robot often requires more "thinking" (i.e., planning) time to come up with a reasonably long collision-free trajectory segment. To facilitate that need, one more criterion is added by RAMP to select the best trajectory segment to execute: maximizing the safe time for the robot to pause and think at the end CT-point of a collision-free segment

of a trajectory. This so-called safe time indicates the time period that the robot can pause its motion and think without being hit by any obstacle, which is determined by the distance between the robot and its nearest atomic obstacle at that CT-point. With this modification, the robot is allowed to pause deliberately to think about its next movement. Therefore, the robot can either stop safely (i.e., deliberately pause its motion within the safe time) or unsafely (i.e., forced to stop with possibilities to be hit by obstacles).

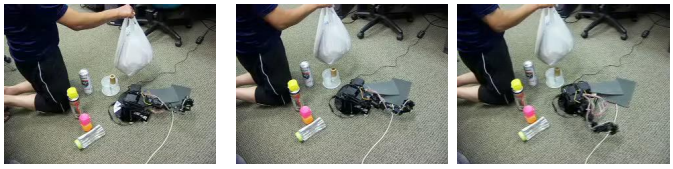
Figures 50 and 51 show the snapshots of two real-world experiments: Experiment #1 and Experiment #2 respectively, where the start and goal configurations of the robot are given in Table 13. For Experiment #1, the stereo-vision sensor provides an isometric view. For Experiment #2, the stereo-vision sensor provides a side view of the workspace.

We deliberately used rather irregular and colorless objects – white plastic bags with some objects in it and a half-full plastic bottle of water, as obstacles, which are difficult to model and recognize, to take advantage of our approach that does not require modeling and recognizing obstacles. From the size of the arm, one can also see that the robot's workspace was crowded with obstacles. The empty space was mostly beyond the robots workspace and was not reachable.

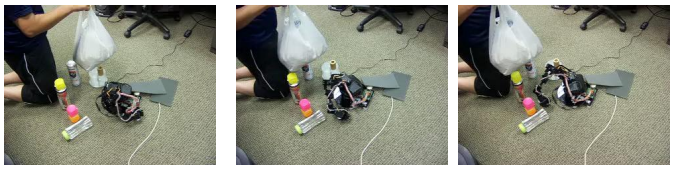
Table 13: Robot start and goal configurations for experiments #1 and #2

Exp#1	$\mathbf{q_s}$ (deg)	$[79.9, 99, 81.5, 113.6, 122, 85.1, 105]^T$
	$\mathbf{q_g}$ (deg)	$[-100, 0, 0, 55, 0, 85.1, 0]^T$
Exp#2	$\mathbf{q_s}$ (deg)	$[100, 60, 0, 0, 0, 0, 0]^T$
	$\mathbf{q_g}$ (deg)	$[-80, 0, 0, 55, 0, 85.1, 0]^T$

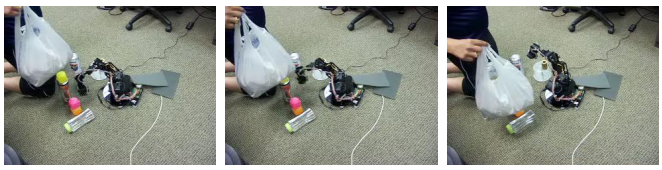
In experiment #1, in addition to the plastic bag and the half-filled bottle, the real-



(a) $\tau_k = 0.05$s (b) $\tau_k = 5.1$s (c) $\tau_k = 6.3$s



(d) $\tau_k = 9.2$s (e) $\tau_k = 17.4$s (f) $\tau_k = 19.6$s



(g) $\tau_k = 22$s (h) $\tau_k = 23.5$s (i) $\tau_k = 26.6$s

Figure 50: Snapshots of experiment #1 with $v_{max} = 1$cm/s.

(a) $\tau_k = 0.0$s (b) $\tau_k = 7.07$s (c) $\tau_k = 16.23$s

(d) $\tau_k = 23.32$s (e) $\tau_k = 24.89$s (f) $\tau_k = 38.01$s

(g) $\tau_k = 54.8$s (h) $\tau_k = 59.53$s (i) $\tau_k = 70.67$s

Figure 51: Snapshots of experiment #2 with $v_{max} = 3$cm/s.

world obstacles also include a part of a person who holds the filled plastic bag and some cans and blocks. Initially, the robotic arm tried to extend vertically to reach the

goal (see snapshots 50(a)–50(c)). However, the arm encountered the moving plastic bag near its goal configuration as seen in snapshot 50(c). It avoided the collision by bending itself to move through the passage formed by the obstacles resting on the floor (shown in snapshots 50(d)–50(i)) and finally reached the goal (see 50(i)), while the person was trying to place the plastic bag within the workspace of the robot. The total time of combined planning and execution of the robot motion was 26.6s.

In experiment #2, the goal position of the robot gripper is above the wooden cylindrical block (see 51(a)). A person tried to move two plastic bags towards the robot as it tried to reach its goal configuration. Initially, while the robotic arm began to move (see snapshots 51(a)–51(b)), it encountered the plastic bag on its right moving towards it; the arm avoided collision by moving closer to the left plastic bag (see snapshot 51(c)). However, left plastic bag also started moving towards the robot (see snapshot 51(d)); so the arm moved beneath the right plastic bag, by bending itself, while minimizing the joint distance to the goal (see snapshots 51(e)–51(g)). Later, as the left bag moved towards the arm, while further avoiding collisions, the arm started moving towards the goal (see snapshot 51(h)) and successfully reached the goal. The total combined planning and execution time was 70.67s.

These experiments clearly show the applicability of our approach in an arbitrary, unknown and unpredictable environment.

Next we further investigated the performance of our approach statistically over different and random arrangements of obstacles and their movements. In order to do that, we used randomly moving virtual obstacles and generated artificial atomic obstacles corresponding to those virtual obstacles as viewed by the real stereo-vision

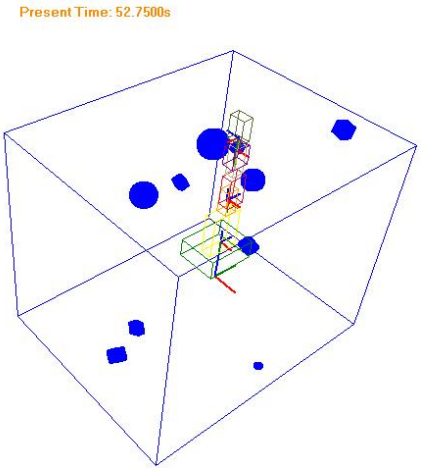


Figure 52: Snapshot of the simulated workspace.

sensor. Each obstacle can translate and rotate in a random fashion, which satisfies only that any point on the obstacle does not move faster than v_{max}. We applied our approach to planning and executing movements of the model of the real robot in this environment of unknown and unpredictable artificial obstacles. Figure 52 shows one example of such environment as viewed from the real stereo-vision sensor, containing the model of the real manipulator.

Given the start and goal configurations of the robot respectively as $\mathbf{q_s}$ where all joint angles are -90^o, and $\mathbf{q_g}$ where all joint angles are 90^o, we run our approach multiple times to move the robot to its goal and collected related performance data. 10 virtual obstacles, with volumes ranging from 8 cm^3 to 125 cm^3, were randomly generated with random initial locations and move randomly, with random translations and rotations, within a bounding box of size $80 \times 100 \times 80$ cm^3, which is larger than the robot workspace (see Figure 52), in each run. This creates an extremely unpredictable environment every run for the robot (which is more challenging than many real-world environments). Note that as the base of the manipulator was fixed, both the base

and the link 1 of the robot cannot move to avoid obstacles, and thus the robot was more likely to have forced stops, and #Hits was comparable to the number of stops $\#S$ in our results.

Table 14 shows some performance data that characterize the real-time motion planning and execution results for $v_{max} = 4$ cm/s, averaged over 30 runs. From the data we can see that there are very few forced stops on average, and the time spent on forced stops is about 22% of the total time of planning and execution (including the time spent on deliberate pauses by the robot). On average, a collision-free trajectory segment has a reasonable length. This indicates that the motion planner combined with CFPA can guide the motion of a high-DOF robot reasonably well in unknown and extremely unpredictable environments.

Table 14: Data of discovered collision-free trajectory segment averaged over 30 runs

$v_{max} = 4$ cm/s, $T = 52.73$s , $\#S = 2.03$,
total time of forced stops= 11.774s

	Avg.	Min.	Max.
L_f (deg.)	38.12	15.93	93.12
T_f (sec)	0.649	0.152	0.947

Table 15 provides CFPA performance data for different v_{max}, each averaged over 30 runs. The average values of T and $\#S$ show that in all cases, the executions were completed in a reasonable amount of time with a very few number of forced stops.

From Table 15, as v_{max} increases from 2 to 6 (cm/s), the corresponding dynamic envelope for a CT-point is larger; thus, the projected range Max. $\#AO$ of atomic obstacles that CFPA may check (see Section IV.c) for a CT-point and the corresponding Max. T_c increases so that the average #AO and T_c may increase, which explains the

results.

However, as v_{max} further increases from 6 to 8 (cm/s), both the average #AO and T_c decrease. This can be explained as follows: CFPA, being a boolean classifier, takes less (or equal) time to detect a dynamic envelope containing atomic obstacles than to detect it containing no atomic obstacle. As the size of dynamic envelopes further increased, more of them contained atomic obstacles, which resulted in the decrease of the average number of atomic obstacles checked by the CFPA along with the average checking time T_c. Thus, the average $\#AO$ and T_c may increase or decrease with increased v_{max}, but Max. $\#AO$ and Max. T_c always increase as v_{max} increases. The experiment shows that even with increasing actual v_{max} (i.e., not over-estimated v_{max}), the CFPA algorithm may not be less efficient.

Note that for obstacles in the small workspace of the fixed-base manipulator, $v_{max} = 8$ (cm/s) is relatively fast.

From Table 15, we can also see that the percentage of maximum #AO over the total number of atomic obstacles (equaling the total number of image pixels 22,560) is at most 26% (which corresponds to the highest v_{max} value) but on the average much lower than that. This again shows the efficiency of the CFPA algorithm.

Table 15: CFPA performance data averaged over 30 runs

v_{max} (cm/s)	T (sec)	$\#S$	Avg. #AO	Avg. T_c (sec)	Max. #AO	Max. T_c (sec)
2	61.16	2.93	845.0	0.050	1613.83	0.285
4	72.65	1.93	1312.5	0.078	2999.2	0.330
6	85.42	1.23	1596.2	0.092	4707.3	0.391
8	107.53	3.48	1315.7	0.082	5906.5	0.534

CHAPTER 10: CONCLUSION AND FUTURE WORK

One of the open challenges in robotics is how to enable a robot to move autonomously along collision-free trajectories (i.e., avoiding obstacles) in real world environments with unknown obstacles or unpredictable obstacle motions. While there exists much literature on planning collision-free robot motion in known or mostly known environments, there is hardly any study for unknown and unpredictable environments. This dissertation has presented novel approaches and algorithms that can detect if a robot future motion in an unknown and unpredictable environment will be guaranteed collision-free or not in real-time based on sensing.

10.1 Contributions

By the novel concept of dynamic envelopes, we can determine if a robot will safely stay at a configuration at a certain future time, called a CT point in the robot's configuration-time (CT) space, through observing the poses of obstacles for some period of time while the dynamic envelope of the CT point shrinks, which we call "progressive sensing". Note that progressive sensing here is different from tracking obstacle motions because it does not need to remember past obstacle poses. By using dynamic envelopes, we avoid both tracking and predicting obstacle motions, which could be expensive and inaccurate.

With dynamic envelopes, we have shown that by detecting a collision-free CT

point in the robot's CT-space, a continuous neighborhood of CT points is also detected collision-free. Based on this fact, we have developed an efficient algorithm for detecting a guaranteed continuously collision-free robot trajectory based on detecting only a discrete set of collision-free CT points. Moreover, our algorithm can detect a continuously collision-free tunnel of trajectories to enable safe robot motion in the presence of motion uncertainty.

We have introduced the notion of atomic obstacles directly from low-level sensing data to characterize perceived obstacles in an unknown environment in any given sensing moment. Using atomic obstacles, we have avoided identifying obstacles that are too numerous to be countable, can appear or disappear, and can merge or separate and also avoided the associated computation cost. Since atomic obstacles are directly obtained from sensing data, there is no construction cost. Moreover, since they are of identical and simple geometry, collision-checking can be efficient, as illustrated by the collision-checker we have developed. Although atomic obstacles cover not only actual obstacles but also occlusions, we have shown that by using multiple sensors, occlusions can be reduced.

Using either dynamic envelopes or atomic obstacles requires finite time to provide collision-checking results. To detect a collision-free CT point usually requires observing the shrinking dynamic envelope for some period of time until it is free of obstacles. To conduct collision checking with respect to atomic obstacles also cannot be done instantaneously. Thus, an important question is how to enable real-time detection of collision-free robot trajectories in an unknown and unpredictable environment when the tools we use require finite time to produce results. We have addressed this ques-

tion by taking advantage of the fact that whenever a robot is following a detected collision-free trajectory segment, it is guaranteed to be safe in spite of motion uncertainty and thus can simultaneously detect the subsequent collision-free trajectory segment while executing the current one in finite time. As the result, the robot can keep moving and detecting without stop most of the time in an unknown and unpredictable environment.

To demonstrate this capability, we have extended an existing real-time motion planner by incorporating the collision-free perceiver based on both dynamic envelopes and atomic obstacles (as described in Chapter 7). We have conducted simulations and real experiments to show that a robot, even a high-DOF one, only makes a few stops when it is moving towards a goal configuration in an unknown and unpredictable environment. The robot stops when it has finished following the current collision-free trajectory segment but requires more time to detect the next collision-free segment. Note that when the robot stops, it may get hit by a moving obstacle in theory, but it is reasonable to assume that other obstacles (e.g., people or things people move) will not deliberately hit a robot. The most important point is that, with our approaches, the robot is guaranteed not to hit anything actively.

Thus, we have made a first step in ensuring that a robot moving in an unknown and unpredictable environment does not hit any obstacles. We have published many results reported in this dissertation, see [94–97, 102].

10.2 Future work and open challenges

Much future research can sprout from this work, especially related to sensing and processing of atomic obstacles.

Handling noise in sensor data: Data generated by a sensor are important input for our approaches to successfully detect collision-free robot trajectories. However, sensor data can be noisy due to various lighting conditions, different textures of objects, etc., and errors in sensor data may cause a robot to hit an obstacle. We have experienced three kinds of noise in sensor data: (a) the sensor data values are inaccurate by some margin, (b) some sensor data are false alarms because they do not indicate real objects, and (c) there are missing sensor data values with respect to an object sometimes.

Noise type (a) can be handled relatively easily. For instance, by growing the front face of an atomic obstacle by the error margin, along the viewing frustum, towards the sensor can avoid missing obstacles, even though this is a conservative approach. Noise type (b) refers to outliers of sensor data, which can be filtered out based on some environmental information, e.g., one can assume that each object in an environment is greater than some threshold in size.

Handling noise type (c) is challenging. One approach [82] uses a maximum-likelihood framework to find the missing data. However, such computation is expensive. It is important to explore how to effectively find missing data in real-time. There also exist methods [40, 78] that produce more accurate sensor data efficiently, either by controlling the lighting [78], or by better data matching for stereo vision [40].

Detecting occluded space as free space by using past information about atomic obstacles: Atomic obstacles include both true obstacles and occluded free space. An interesting question is whether it is possible to detect occluded space in an atomic obstacle from a fixed sensor in a changing environment with moving obstacles without using additional sensors, even though the obstacle motions are unknown. It seems that by observing the past changes in atomic obstacles (which reflect environmental changes), we could reason that a certain part of an atomic obstacle at the current sensing moment is actually a part of occluded free space. For example, for a pixel (i, j), if the corresponding atomic obstacle suddenly appears very close to the sensor comparing to the atomic obstacle of (i, j) at the previous sensing moment, and the atomic obstacle of a neighboring pixel suddenly appears far from the sensor comparing to the previous moment, it may suggest that a neighboring obstacle has moved to occlude the previous obstacle at pixel (i, j). The size of occluded free space could be detected based on the changing depths of the neighboring atomic obstacles.

Increasing visible free space by changing sensing directions: Each sensor has a limited viewing range and is also subject to occlusions. Thus, not everywhere in an environment can be "visible" to a single sensor. We have discussed in the dissertation how multiple sensors can be used to reduce occlusion and increase visible free-space. How to maximize viewing ranges by placing multiple sensors is a problem that has been studied in the literature for environments that are mostly static (e.g. [1, 17]). However, how to increase a sensor's limited viewing range by changing its sensing directions is an interesting problem that should be investigated. Since atomic obstacles include both obstacles and occluded regions that could be free space, changing

sensing directions can also reduce occlusion represented as obstacles. On the other hand, frequent change of viewing directions can increase the cost of the collision-checker for atomic obstacles because the time coherence can be lost from changing viewing direction at one sensing moment to the next sensing moment. Thus, minimizing changes in orientations of sensors while still maximizing the viewing capability is another research problem.

Since a detected collision-free CT point is guaranteed collision-free by our approach, once the detection is done, the sensor can change its viewing direction to perform detections for other CT points without continuing to monitor the already detected collision-free CT point.

Increasing visible free space by changing sensor poses: Since the visible free space viewed from different sensors with fixed bases may not form a connected region that is necessary for a robot to move to its goal configuration, it may be desirable to have moving sensors, i.e., sensors mounted on the robot or other robots. This presents more open research challenges in an unknown and unpredictable environment, from how a sensor should move while avoiding obstacles "seen" by itself and other sensors, to how it coordinates its viewing direction with the sensing need of the main robot. The latter is related to viewing planning problems [105, 106] but existing work on viewing planning is focused on static environment with no obstacle motion.

Making CFPA faster by parallel implementation: The main computationally intensive algorithm is CFPA for detecting if a CT-point is collision-free or not. It is mainly based on the flood-fill algorithm. If a parallel version of the flood-fill algorithm is implemented then CFPA for detecting a CT-point would be much faster. Also,

CFPA uses sensor data from different sensors to determine if a CT-point is collision-free or not. Again the checking of individual sensor data can be made on parallel processors for faster computation.

Further detecting if a trajectory Γ is collision-free or not requires checking of multiple CT-points in a set $Q(\Gamma^+)$. These CT-points can be individually checked on parallel processors.

10.3 Applications

A robot autonomously trying to move among unknown and unpredictable obstacles to reach some goal configuration is the primary application of the work done in this dissertation. A robot could be of any form, including mobile robots, manipulators, mobile manipulators, etc. The set of assumptions required by the algorithms of this dissertation to guide motions of such a robot is as follows:

1. v_{max} assumption: The maximum linear speed any obstacle can have is the only required parameter about the environment.

2. The entire environment is visible from fixed sensors: The current approaches mentioned in this dissertation work well if the sensors in the environment are fixed and their combined viewing ranges cover the entire environment.

3. The robot can be localized w.r.t. fixed sensor poses: The motion uncertainty of robot is commonly handled by using known environmental features. However, if an environment is unknown and unpredictable, the environmental features are unknown and thus poses of fixed sensors need to be used for localizing the robot.

The above set of assumptions can be easily realized for indoor environments, such as offices, houses, restaurants, etc. The sensors can be placed on known static obstacles, such as walls, ceiling, etc. There can be dynamic obstacles, such as people, other robots, etc. that the robot has to avoid collision with.

For outdoor environments, where there are street lights, the sensors could be mounted on them. The robot could be an autonomous vehicle navigating streets, avoiding collisions with other vehicles.

REFERENCES

[1] Acar, E. U., and Choset, H. Sensor-based coverage of unknown environments. *International Journal of Robotics Research 21*, 4 (2002), 345–366.

[2] Amato, N. M., Bayazit, O. B., Dale, L. K., Jones, C., and Vallejo, D. OBPRM: an obstacle-based PRM for 3d workspaces. In *WAFR '98: Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics* (Natick, MA, USA, 1998), A. K. Peters, Ltd., pp. 155–168.

[3] Baginski, B. Efficient dynamic collision detection using expanded geometry models. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (1997), pp. 1714–1719.

[4] Bellotto, N., and Hu, H. Multisensor-based human detection and tracking for mobile service robots. *IEEE Trans. on Systems, Man, and Cybernetics – Part B 39*, 1 (2009), 167–181.

[5] Bennewitz, M., Burgard, W., Cielniak, G., and Thrun, S. Learning Motion Patterns of People for Compliant Robot Motion. *Intl. J. of Robotics Research 24*, 1 (2005), 31–48.

[6] Bonissone, P. P., Subbu, R., Eklund, N., and Kiehl, T. R. Evolutionary algorithms + domain knowledge = real-world evolutionary computation. *IEEE Trans. Evolutionary Computation 10*, 3 (2006), 256–280.

[7] Bradley, D., Unnikrishnan, R., and Bagnell, J. A. Vegetation detection for driving in complex environments. In *IEEE Intl. Conf. on Robotics and Automation* (April 2007).

[8] Brent, R. P. An algorithm with guaranteed convergence for finding a zero of a function. *Computer Journal 14* (1971), 422–425.

[9] Brockett, R. W. *Asymptotic Stability and Feedback Stabilization.* Birkhauser, Boston, 1983, pp. 181–191.

[10] Bus, J. C. P., and Dekker, T. J. Two efficient algorithms with guaranteed convergence for finding a zero of a function. *ACM Trans. Math. Softw. 1*, 4 (1975), 330–345.

[11] Cameron, S. Collision detection by four-dimensional intersection testing. *IEEE Trans. on Robotics and Automation 6*, 3 (1990), 291–302.

[12] Chang, C. C., and Song, K.-T. Environment prediction for a mobile robot in a dynamic environment. *IEEE Trans. on Robotics and Automation 13*, 6 (Dec. 1997), 862–872.

[13] Chen, Z., Ngai, D. C. K., and Yung, N. H. C. Behavior prediction based on obstacle motion patterns in dynamically changing environments. In *Proceedings of the 2008 IEEE/WIC/ACM Intl. Conf. on Intelligent Agent Technology* (2008), pp. 132–135.

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

[15] Cohen, J. D., Lin, M. C., Manocha, D., and Ponamgi, M. I-collide: An interactive and exact collision detection system for large-scale environments. In *Proc. of ACM Interactive 3D Graphics Conf.* (1995), pp. 189–196.

[16] Craig, J. J. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[17] Dhillon, S. S., and Chakrabarty, K. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *Proc. of IEEE Wireless Communications and Networking Conference* (2003), pp. 1609–1614.

[18] Discant, A., Rogozan, A., Rusu, C., and Bensrhair, A. Sensors for obstacle detection - a survey. In *Electronics Technology, 30th International Spring Seminar on* (May 2007), pp. 100 –105.

[19] Dixon, W., Walker, I., and Dawson, D. Fault detection for wheeled mobile robots with parametric uncertainty. In *Advanced Intelligent Mechatronics, 2001. Proceedings. 2001 IEEE/ASME International Conference on* (2001), vol. 2, pp. 1245 –1250 vol.2.

[20] Du Toit, N., and Burdick, J. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics 28*, 1 (Feb. 2012), 101 –115.

[21] Elnagar, A., and Gupta, K. Motion prediction of moving objects based on autoregressive model. *IEEE Trans. on Systems, Man, and Cybernetics, Part A 28*, 6 (1998), 803–810.

[22] Elnagar, A., and Hussein, A. An adaptive motion prediction model for trajectory planner systems. In *Intl. Conf. on Robotics and Automation* (Sep. 2003), pp. 2442–2447.

[23] Ess, A., Leibe, B., Schindler, K., and Gool, L. V. Moving obstacle detection in highly dynamic scenes. In *IEEE Intl. Conf. on Robotics and Automation* (May 2009), pp. 56–63.

[24] Esteban, C. H., Hernandez, C., and Schmitt, E. F. Multi-stereo 3d object reconstruction. In *3D Data Processing Visualization and Transmission, 2002* (2002), pp. 159–166.

[25] Fiorini, P., and Shiller, Z. Motion planning in dynamic environments using velocity obstacles. In *Intl. J. of Robotics Research* (1998), vol. 17, pp. 760–772.

[26] Foisy, A., and Hayward, V. A safe swept volume method for collision detection. In *The Sixth Int. Symp. of Robotics Research* (Pittsburgh (PE), Oct. 1993), pp. 61–68.

[27] Foka, A. F., and Trahanias, P. E. Predictive autonomous robot navigation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS* (2002), pp. 490–495.

[28] Fraichard, T., and Asama, H. Inevitable collision states - a step towards safer robots? *Advanced Robotics 18*, 10 (2004), 1001–1024.

[29] Fraichard, T., and Kuffner, J. Guaranteeing motion safety for robots. *Autonomous Robots* (2012), 1–3.

[30] Fraichard, T., and Mermond, R. Path planning with uncertainty for car-like robots. In *IEEE International Conference on Robotics and Automation* (May 1998), vol. 1, pp. 27 –32 vol.1.

[31] Fu, Y., Jin, B., Wang, S., and Cao, Z. Real-time sensor-based motion planning for robot manipulators. In *IEEE Intl. Conf. on Robotics and Automation* (2005), pp. 3108–3113.

[32] Gallagher, G., Srinivasa, S. S., Bagnell, J. A., and Ferguson, D. Gatmo: a generalized approach to tracking movable objects. In *IEEE Intl. Conf. on Robotics and Automation* (May 2009), pp. 2043–2048.

[33] Gavrila, D. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding 73* (1999), 82–98.

[34] Ghosh, P. K. A unified computational framework for Minkowski operations. *Computers & Graphics 17*, 4 (1993), 357–378.

[35] Govea, V., Alejandro, D., Large, F., Fraichard, T., and Laugier, C. High-speed autonomous navigation with motion prediction for unknown moving obstacles. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (Oct. 2004), pp. 82–87.

[36] Govea, V., Alejandro, D., Large, F., Fraichard, T., and Laugier, C. Moving obstacles' motion prediction for autonomous navigation. In *Int. Conf. on Control, Automation, Robotics and Vision* (Dec. 2004).

[37] Hahnel, D., Schulz, D., and Burgard, W. Map building with mobile robots in populated environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2002), vol. 1, pp. 496–501.

[38] Han, M., and Kanade, T. Creating 3d models with uncalibrated cameras. In *proceeding of IEEE Computer Society Workshop on the Application of Computer Vision (WACV2000)* (Dec. 2000).

[39] Heckbert, P. S. A seed fill algorithm. In *Graphics gems*. Academic Press Professional, Inc., San Diego, CA, USA, 1990, pp. 275–277.

[40] Hirschmller, H. Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), IEEE Computer Society, pp. 807–814.

[41] Huang, Y., and Gupta, K. RRT-SLAM for motion planning with motion and map uncertainty for robot exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sept. 2008), pp. 1077 –1082.

[42] Jiménez, P., Thomas, F., and Torras, C. 3D collision detection: A survey. *Computers and Graphics 25* (2000), 269–285.

[43] Jones, B., and Walker, I. Kinematics for multisection continuum robots. *IEEE Transactions on Robotics 22*, 1 (Feb. 2006), 43 – 55.

[44] Kamoun, W., Schmugge, S., Kraftchick, J., Clemens, M., and Shin, M. Liver microcirculation analysis by red blood cell motion modeling in intravital microscopy images. In *IEEE Trans. on Biomedical Engineering* (2008).

[45] Kant, K., and Zucker, S. W. Toward efficient trajectory planning: the path-velocity decomposition. *Int. J. Rob. Res. 5*, 3 (1986), 72–89.

[46] Kavraki, L., Svestka, P., Latombe, J., and Overmars, M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE Trans. on Robotics and Automation* (1996), vol. 12, pp. 566–580.

[47] Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research (IJRR) 5*, 1 (1986), 90–98.

[48] Kodratoff, Y., and Moscatelli, S. Machine learning for object recognition and scene analysis. *International Journal of Pattern Recognition and AI 8* (1994), 259–304.

[49] Kurniawati, H., Yanzhu, D., Hsu, D., and Wee, S. L. Motion planning under uncertainty for robotic tasks with long time horizons. *The International Journal of Robotics Research 30*, 3 (2011), 308–323.

[50] Kushleyev, A., and Likhachev, M. Time-bounded lattice for efficient planning in dynamic environments. In *IEEE Intl. Conf. on Robotics and Automation* (May 2009), pp. 1662–1668.

[51] Lambert, A., and Le Fort-Piat, N. Safe actions and observations planning for mobile robots. In *IEEE Intl. Conf. on Robotics and Automation* (1999), pp. 1341–1346.

[52] Large, F., Sckhavat, S., Shiller, Z., and Laugier, C. Using non-linear velocity obstacles to plan motions in a dynamic environment. In *IEEE Intl. Conf. on Control, Automation, Robotics and Vision (ICARCV)* (2002), pp. 734–739.

[53] Latombe, J. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[54] LaValle, S., and Sharma, R. Robot motion planning in a changing, partially predictable environment. In *IEEE International Symposium on Intelligent Control* (Aug. 1994), pp. 261 –266.

[55] LaValle, S. M. *Planning Algorithms*. Cambridge University Press, May 2006.

[56] LaValle, S. M., and Jr., J. J. K. Randomized kinodynamic planning. In *IEEE International Conference on Robotics and Automation* (1999), pp. 473–479.

[57] Le, D. An efficient derivative-free method for solving nonlinear equations. *ACM Trans. Math. Softw. 11*, 3 (1985), 250–262.

[58] Leven, P., and Hutchinson, S. A framework for real-time path planning in changing environments. *Intl. J. of Robotics Research 21* (2002), 999–1030.

[59] Leven, P., and Hutchinson, S. Using manipulability to bias sampling during the construction of probabilistic roadmaps. *IEEE Trans. on Robotics and Automation 19*, 6 (Dec. 2003), 1020–1026.

[60] Li, J., and Xiao, J. Exact and efficient collision detection for a multi-section continuum manipulator. In *IEEE International Conference on Robotics and Automation* (Saint Paul, Minnesota, May 2012).

[61] Lin, M. C., and Gottschalk, S. Collision detection between geometric models: A survey. In *Proc. of IMA Conf. on Mathematics of Surfaces* (1998), pp. 37–56.

[62] Lozano-Pérez, T. Spatial planning: A configuration space approach. In *IEEE Trans. on Computers* (Feb. 1983), vol. C-32, pp. 108–120.

[63] Lozano-Pérez, T., and Wesley, M. A. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the Association for Computing Machinery (ACM) 22*, 10 (1979), 560–570.

[64] Lumelsky, V. J., and Stepanov, A. A. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica* (1987).

[65] Mason, M. T. *Mechanics of Robotic Manipulation.* MIT Press, Cambridge, MA, Aug. 2001.

[66] Missiuro, P., and Roy, N. Adapting probabilistic roadmaps to handle uncertain maps. In *IEEE International Conference on Robotics and Automation* (May 2006), pp. 1261 –1267.

[67] Miura, J., Uozumi, H., and Shirai, Y. Mobile robot motion planning considering the motion uncertainty of moving obstacles. In *Proceedings. 1999 IEEE Intl. Conf. on Systems, Man, and Cybernetics* (1999), pp. 692–697.

[68] Miura, J., Uozumi, H., and Shirai, Y. Mobile robot motion planning considering the motion uncertainty of moving obstacles. In *Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics* (1999), pp. 692–697.

[69] Murarka, A., Sridharan, M., and Kuipers, B. Detecting obstacles and drop-offs using stereo and motion cues for safe local motion. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (2008), pp. 702–708.

[70] Nam, Y. S., Lee, B. H., and Kim, M. S. View-time based moving obstacle avoidance using stochastic prediction of obstacle motion. In *IEEE Intl. Conf. on Robotics and Automation* (1996), pp. 1081–1086.

[71] Overmars, M. H. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. on Robotics and Automation* (1999), pp. 1018–1023.

[72] Page, L., and Sanderson, A. Robot motion planning for sensor-based control with uncertainties. In *IEEE International Conference on Robotics and Automation* (May 1995), vol. 2, pp. 1333 –1340 vol.2.

[73] Pham, T., and Smeulders, A. Object recognition with uncertain geometry and uncertain part detection. *Computer Vision and Image Understanding 99* (2005), 258.

[74] Pivtoraiko, M., and Kelly, A. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics 26*, CMU-RI-TR- (March 2009), 308–333.

[75] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Aug. 2007.

[76] Redon, S., Lin, M. C., Manocha, D., and Kim, Y. J. Fast continuous collision detection for articulated models. *Journal of Computing and Information Science in Engineering 5*, 2 (2005), 126–137.

[77] Roy, N., Burgard, W., Fox, D., and Thrun, S. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *IEEE International Conference on Robotics and Automation* (1999), vol. 1, pp. 35–40.

[78] Scharstein, D. High-accuracy stereo depth maps using structured light. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2003), pp. 195–202.

[79] Schneiderman, H., and Kanade, T. Object detection using the statistics of parts. *Intl. Journal of Computer Vision 56* (2004), 151–177.

[80] Schwarzer, F., Saha, M., and Latombe, J. Adaptive dynamic collision checking for single and multiple articulated robots in complex environments. *IEEE Trans. on Robotics 21*, 3 (2005), 338–353.

[81] Schweikard, A. Polynomial time collision detection for manipulator paths specified by joint motions. In *IEEE Trans. on robotics and automation* (1991), vol. 7, pp. 865–870.

[82] Sharp, G., Lee, S., and Wehe, D. Maximum-likelihood registration of range images with missing data. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 1 (Jan. 2008), 120 –130.

[83] Szeliski, R. *Computer Vision: Algorithm and Applications*. Springer, 2010.

[84] Takeda, H., Facchinetti, C., and Latombe, J.-C. Planning the motions of a mobile robot in a sensory uncertainty field. *IEEE Trans. Pattern Anal. Mach. Intell. 16*, 10 (1994), 1002–1017.

[85] Thompson, S., and Kagami, S. Stereo vision and sonar sensor based view registration for 2.5 dimensional map generation. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (Sept. 2004), vol. 4, pp. 3444 – 3449 vol.4.

[86] Thrun, S. In *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2002, ch. Robotic mapping: A survey.

[87] Treuenfels, A. An efficient flood visit algorithm. *C/C++ Users J. 12*, 8 (1994), 39–62.

[88] Udupa, S. M. *Collision detection and avoidance in computer controlled manipulators.* PhD thesis, Pasadena, CA, USA, 1977.

[89] van den Berg, J., Ferguson, D., and Kuffner, J. Anytime path planning and replanning in dynamic environments. In *IEEE Intl. Conf. on Robotics and Automation* (May 2006), pp. 2366–2371.

[90] van den Berg, J., and Overmars, M. Planning time-minimal safe paths amidst unpredictably moving obstacles. In *Intl. J. on Robotics Research* (2008), pp. 1274–1294.

[91] Vannoy, J., and Xiao, J. Real-time motion planning of multiple mobile manipulators with a common task objective in shared work environments. In *IEEE Intl. Conf. on Robotics and Automation* (April 2007), pp. 20–26.

[92] Vannoy, J., and Xiao, J. Real-time Adaptive Motion Planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes. In *IEEE Trans. on Robotics* (2008), vol. 24(5), pp. 1199–1212.

[93] Varadhan, G., and Manocha, D. Accurate Minkowski sum approximation of polyhedral models. *Graphical Models 68*, 4 (2006), 343–355.

[94] Vatcha, R., and Xiao, J. Perceived CT-space for motion planning in unknown and unpredictable environments. In *Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)* (Dec. 2008).

[95] Vatcha, R., and Xiao, J. Discovering guaranteed continuously collision-free robot trajectories in an unknown and unpredictable environment. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems* (Oct. 2009).

[96] Vatcha, R., and Xiao, J. An efficient algorithm for on-line determination of collision-free configuration-time points directly from sensor data. In *IEEE Intl. Conf. on Robotics and Automation* (May 2010).

[97] Vatcha, R., and Xiao, J. Practical motion planning in unknown and unpredictable environment. In *12th International Symposium on Experimental Robotics* (Dec 2010).

[98] Ward, J., and Katupitiya, J. Free space mapping and motion planning in configuration space for mobile manipulators. In *IEEE Intl. Conf. on Robotics and Automation* (2007), pp. 4981–4986.

[99] Widyotriatmo, A., Pamosoaji, A., and Hong, K.-S. Robust configuration control of a mobile robot with uncertainties. In *Control Conference (ASCC), 2011 8th Asian* (May 2011), pp. 1036 –1041.

[100] Withagen, P., Schutte, K., and Groen, F. Object detection and tracking using a likelihood based approach. In *Proc. IEEE Int. Conf. on Image Processing* (2003).

[101] Wu, A., and How, J. Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles. *Autonomous Robots 32* (2012), 227–242. 10.1007/s10514-011-9266-8.

[102] Xiao, J., and Vatcha, R. Real-time adaptive motion planning for a continuum manipulator. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems* (Oct. 2010).

[103] Yang, Y., and Brock, O. Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments. In *Robotics Science and Systems II* (2006), The MIT Press.

[104] Yu, Y., and Gupta, K. An efficient on-line algorithm for direct octree construction from range images. In *IEEE Intl. Conf. on Robotics and Automation* (1998), pp. 3079–3084.

[105] Yu, Y., and Gupta, K. Sensor-based probabilistic roadmaps: experiments with an eye-in-hand system. In *Advanced Robotics* (2000), pp. 515–536.

[106] Yu, Y., and Gupta, K. C-space entropy: A measure for view planning and exploration for general robot-sensor systems in unknown environments. In *Intl. J. of Robotics Research* (2004), pp. 1197–1223.

[107] Zhu, X., and Goldberg, A. B. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning 3*, 1 (2009), 1–130.

[108] Zucker, M., Kuffner, J., and Branicky, M. Multipartite rrts for rapid replanning in dynamic environments. In *IEEE Intl. Conf. on Robotics and Automation* (2007), pp. 1603–1609.