RRT BASED KINODYNAMIC MOTION PLANNING FOR MULTIPLE CAMERA INDUSTRIAL INSPECTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BURAK BILGE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

MAY 2009

Approval of the thesis:

RRT BASED KINODYNAMIC MOTION PLANNING FOR MULTIPLE CAMERA INDUSTRIAL INSPECTION

submitted by **BURAK BILGE** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by**,**

Prof. Dr. Canan Özgen _____
Dean, Graduate School **of Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Assist. Prof. Dr. Afşar Saranlı _____
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members:

Prof. Dr. Aydan Erkmen _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Afşar Saranlı _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Kemal Leblebicioğlu _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. İlhan Konukseven _____
Mechanical Engineering Dept., METU

Volkan Arıcı (MS) _____
Avionics System Engineer, Roketsan

Date: 06.05.2009

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

Name, Surname: Burak Bilge

Signature:

ABSTRACT

RRT BASED KINODYNAMIC MOTION PLANNING FOR
MULTIPLE CAMERA INDUSTRIAL INSPECTION

Bilge, Burak

M.S., Department of Electrical and Electronics Engineering

Supervisor: Assist. Prof. Dr. Afşar Saranlı

May 2009, 82 pages

Kinodynamic motion planning is an important problem in robotics. It consists of planning the dynamic motion of a robotic system taking into account its kinematic and dynamic constraints. For this class of problems, high dimensionality is a major difficulty and finding an exact time optimal robot motion trajectory is proven to be NP-hard. Probabilistic approximate techniques have therefore been proposed in the literature to solve particular problem instances. These methods include Randomized Potential Field Planners (RPP), Probabilistic Roadmaps (PRM) and Rapidly Exploring Random Trees (RRT). When physical obstacles and differential constraints are added to the problem, applying RPPs or PRMs encounter difficulties. In order to handle these difficulties, RRTs have been proposed. In this study, we consider a multiple camera industrial inspection problem where the concurrent motion of these cameras needs to be planned. The cameras are required to capture maximum number of defect locations while globally avoiding collisions with each other and with obstacles. Our approach is

to consider a solution to the kinodynamic planning problem of multiple camera inspection by making use of the RRT algorithm. We explore and resolve issues arising when RRTs are applied to this specific problem class. Along these lines, we consider the cases of a single camera without obstacles and then with obstacles. Then, we attempt to extend the study to the case of multiple camera where we also need to avoid collisions between cameras. We present simulation results to show the performance of our RRT based approach to different instrument configurations and compare with existing deterministic approaches.

Keywords: Motion planning, kinodynamic planning, RRTs, configuration space, collision avoidance

ÖZ

ÇOKLU KAMERA ENDÜSTRİYEL DENETİMİ İÇİN RRT TABANLI KİNODİNAMİK HAREKET PLANLAMASI

Bilge, Burak

Yükseklisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Yard. Doç. Dr. Afşar Saranlı

Mayıs 2009, 82 sayfa

Robotik alanında Kinodinamik Hareket Planlaması önemli bir problemdir. Robotik sistemlerin kinematik ve dinamik kısıtlamalarını dahil ederek hareket planlaması yapmaktan oluşmaktadır. Bu sınıftaki problemler için yüksek boyutluluk büyük bir zorluk ve tam zamanlı optimum bir robot hareket yolunun bulunmasının zor olduğu kanıtlanmıştır (NP-hard). Bu yüzden belirli örnekleri çözmek için literatürde olasılıklı yaklaşım teknikleri önerilmiştir. Bu metodlar Rastgele Potensiyel Alan Planlayıcılarını (RPP), Olasılıklı Yol Haritalarını (PRM) ve Hızlı Keşfeden Rastlansal Ağaçlarını (RRT) içermektedir. Engeller ve differansiyel kısıtlamalar probleme eklendiğinde, RPP'leri veya PRM'leri uygulamak zorluklarla karşılaştırır. Bu zorluklardan kurtulmak için literatürde RRT önerilmiştir. Bu çalışmada, biz kameraların eş zamanlı hareketlerinin planlanması gereken bir çoklu kamera endüstriyel denetleme problemi üzerinde durduk. Kameraların birbirleri ve engelllerle çarpışmaktan kaçınırken maksimum sayıda kusur yerini yakalamaları gerektirmektedir. Bizim yaklaşımımız

RRT algoritmasını kullanarak çoklu kamera denetiminin kinodinamik problemi için bir çözüm bulmaktır. RRT bu belirli problem sınıfına uygulandığında ortaya çıkan sorunları araştırıyor ve çözüyoruz. Bu doğrultuda, öncelikle engelsiz ve engelli tek kamera durumunu inceliyoruz. Sonra, kameralar arası çarpışmalardan kaçınma da gerektiren çoklu kamera durumunu deniyoruz. Bizim farklı araç konfigürasyondaki RRT tabanlı yaklaşımımızın performansını göstermek ve var olan deterministik yaklaşımlarla karşılaştırmak için simulayon çıktıları sunuyoruz.

Anahtar Kelimeler: Hareket planlaması, kinodinamik planlama, RRT, konfigürasyon uzayı, çarpışma sakınma

To My Family,

Teman, Hayrettin, Betül and Buğra Bilge.

ACKNOWLEDGEMENTS

I express sincere thanks to my supervisor, Assist Prof. Dr. Afşar SARANLI for providing vision, knowledge, guidance and encouragement in the preparation of this thesis.

I would further like to thank my colleagues in Avionics System Engineering of ROKETSAN Missiles Industries Inc. for their help and patience.

Finally, my deepest thanks are to my sister Betül Bilge who motivated me and gave great assistance to finalize this study, and to my parents and my brother for their support and patience. Also thanks to everyone who helped me directly or indirectly in completing this thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

DDS	Defect Detection Subsystem
DOF	Degrees of Freedom
DRS	Defect Review Subsystem
ITO	Indium Tin Oxide
POI	Point of Interest
PRM	Probabilistic Roadmaps
RPP	Randomized Potential Fields Planner
RRT	Rapidly Exploring Random Trees

CHAPTER 1

INTRODUCTION

1.1 General

Simply put, motion planning is finding a continuous path for a robot from an initial configuration to a goal configuration. Solutions developed for the general path finding problem are also very valuable for many important and difficult problems in real-life industrial and military applications. Steven M. LaValle describes this widespread usage of motion planning in his book [14]. His examples include a well known toy example, namely the “Piano Mover’s Problem” that deals with the problem of how to maneuver a piano through narrow passages; “Navigating a Mobile Robot” in which a robot is tasked to build a map of the environment; “Humanoid Robots” that attempt to imitate natural human movements; “Automotive Assembly” as well as ”Car and Trailer Parking”. Formally, motion planning can be defined as the problem of finding a trajectory for a robot from an initial state to a final state while satisfying the robot’s constraints, avoiding collisions with obstacles and self collisions [13].

According to the constraints of robots moving in the environment, motion planning can be divided into some main categories. These are called *holonomic*, *nonholonomic* and *kinodynamic* motion planning [11]. Holonomic motion planning deals with only position and orientation of the robot; and it is for holonomic robots in which controllable degrees of freedom (DOF) are greater than or equal to total number of

DOF. Most of the robots in real-life applications have fewer controllable DOF than the total DOF. Nonholonomic motion planning is interested in these robots by including their nonintegrable constraints. By considering both the kinematics and dynamics of the robot in a generic motion planning problem, one can define the kinodynamic motion planning. One important issue of this type of planning problem is its high dimensionality. Because of this issue, an exact time optimal trajectory for the robot cannot be determined by using conventional planning techniques, this has been proven to be NP-hard [13, 15]. A number of probabilistic approximate techniques have been proposed to deal with this difficulty and estimate a solution. These are Randomized Potential Field Planners (RPP), Probabilistic Roadmaps (PRM) and Rapidly Exploring Random Trees (RRT).

In RPP, a heuristic function is defined to direct a robot towards a goal configuration through the gradient descent. When robot is stuck in local minima, which typically happens in the surrounding of the obstacles, random walks are used to escape from there and sometimes this method fails and robot returns to the same local minima. So a good heuristic function must be defined to avoid this case, but when obstacles and differential constraints are added to the problem, the definition of this function becomes difficult. In PRM, random configurations are generated on configuration space and then the pairs of nearby configurations are tried to connect by using a local planner. This provides a graph that is used to find the path of the robot to the goal configuration. However, for complicated nonholonomic dynamical systems, it is hard to connect the configurations. In order to resolve these issues, LaValle has developed a new tool for motion planning that is called RRT as proposed in [10, 11]. RRT easily drives forward like RPP and explore the space quickly and uniformly like PRM. This approach is specifically designed to handle nonholonomic constraints including dynamics and high degrees of freedom [10]. This shows the appropriateness of this algorithm for kinodynamic motion planning problem.

1.2 Problem Statement

In this study, we consider a kinodynamic planning problem of a multiple camera inspection platform, where a set of multiple moving cameras are to be controlled over an inspection area. Firstly, we define mechanical and motion constraints of the system by relating it with the architecture of an existing "scanner" type inspection device. This shows important challenges of the problem, which are the presence of multiple goals and the possibility of collisions between cameras. Therefore, the main problem of this thesis is to plan the motions of each of the cameras while imaging as much locations as possible in a given time while globally avoiding collisions. In this problem, we do not aim to obtain optimal solutions, since kinodynamic motion planning is NP-hard optimal, and we try to get "good results" as long as the constraints are satisfied as LaValle suggested in [12].

After a literature research, we decide to use the RRT algorithm for our problem and resolve the difficulties of the approach. Along these lines, we first consider the case of a single camera and no obstacle. Then we extend our results to the case where obstacles are present and collisions with obstacles are to be avoided. Finally, we attempt to extend to the case of multiple camera and avoiding collisions between cameras. The available RRT based techniques in the literature are experimentally evaluated and issues are identified. We incrementally propose techniques to alleviate these problems to build a working solution whose performance is verified by experimental studies on a number of scenarios.

1.3 Scope & Contribution of the Thesis

The scope of this thesis is resolving the kinodynamic motion planning problem of a multiple camera industrial inspection in which there exist multiple robots and goals. We use the RRT algorithm for this problem in four cases that are single camera without obstacle, single camera with obstacle, multiple camera without obstacle and multiple camera with obstacle. Also we consider the extended case of the multiple camera

industrial inspection as a future work in which cameras can move vertically independent of each other. This extension represents that our approach to this problem can be applied to the motion planning problems in more generic platforms.

As the main contribution of this study, we can offer the adaptation of the RRT algorithm to our problem, in which there are multiple initials and goals. In the literature, the RRT algorithm is mainly used for single query problems that include a goal in the environment, so we need to adapt this algorithm to our problem and then we develop a RRT based algorithm in which multiple RRTs are used to find the final path that reaches as much goals as possible while avoiding collision. While implementing our algorithm, we experience some issues such as RRT performance dependence on sampling, low rate of the RRT convergence, inefficient collision avoidance, and undesired and unintuitive final path, and find solutions in the literature that is explained in chapter 2. Also, we develop the mixture of sampling techniques to improve the performance of our algorithm in the environment with obstacles and using this algorithm, we implement an RRT based motion planner model that solves our problem in this study.

1.4 Outline of the Thesis

As an outline of this thesis, firstly we give a literature survey about kinodynamic motion planning and probabilistic approximate techniques RPP, PRM and RRT in chapter 2 and explain the problem description of the multiple camera inspection in chapter 3. Then, we tell our approach to the problem with the issues and our solutions in chapter 4. Chapter 5 represents the results of our experiments. Finally, we present discussions, conclusion and future work in chapter 6.

CHAPTER 2

LITERATURE SURVEY

2.1 Kinodynamic Motion Planning

Kinodynamic motion planning problem is a problem that takes into consideration both kinematic constraints, such as the configuration (position) of robot, the joint limits and obstacles, and dynamic constraints, such as dynamic laws and bounds on velocity, acceleration and applied force. In other words, this is a motion planning problem of a robot (or robots) by satisfying the limits on its configuration and time derivative of configuration. Therefore, in kinodynamic motion planning, the dimension of constraints doubles and increases the computational complexity. This causes an open problem in robotics that is to find time-optimal kinodynamic solution. After a great deal of study on this problem, it has been shown that finding an exact solution is NP-hard [13, 15]. And probabilistic approximate techniques have therefore been developed to find solutions close to optimal.

As stated in [15], these techniques for the kinodynamic motion planning trade off computational complexity against optimality in terms of

- "execution time of the motion"
- "strictness in observing safety margin"
- "closeness to the desired start and goal configuration"

By considering this trade off, a tolerance parameter is defined to express the closeness to an optimal safe solution and using this definition, “near optimal” solution can be obtained. And in this study, we consider three successful probabilistic approximate techniques in motion planning problems that are Randomized Potential Fields Planner (RPP), Probabilistic Roadmaps (PRM) and Rapidly Exploring Random Trees (RRT). In the following sections, these techniques will be explained.

2.2 Probabilistic Approximate Techniques

2.2.1 Randomized Potential Fields Planner

In Randomized Potential Fields Planner (RPP) [13], a navigation function is defined in configuration space by using medial axes and level-set methods as shown in figure 1 to steer robot towards goal through gradient descent.

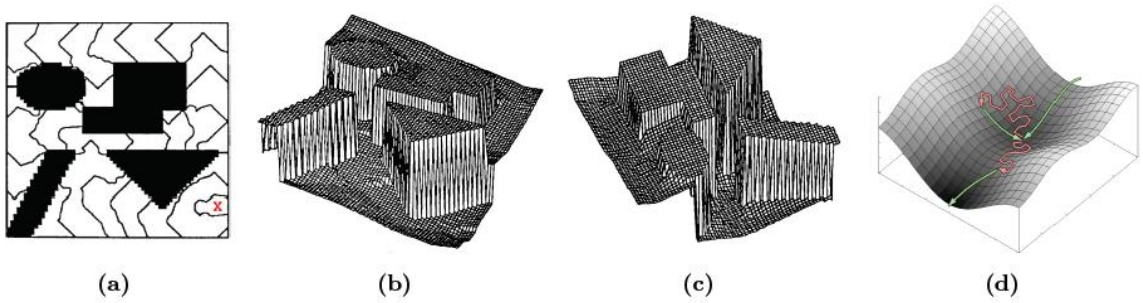

Figure 1: RPP Operation (from [13]): (a) represents the construction of navigation function using medial axes and level-set methods; (b) and (c) shows height-fields at different angles; (d) is an example of local minima.

By using navigation function, potential fields are constructed that are the sum of simpler fields and one of these fields attracts the robot towards goal and others behave like repulsive forces that push the robot from the obstacles as shown in figure 1 (b, c).

While computing navigation function, it is avoided to have local minima in which robot can get stuck. However, local minima can be developed during gradient descent, and typically the robot is trapped in the surrounding obstacles. Random walks method in

which robot is backed applying some random inputs on potential fields are used to escape from local minima as shown in figure 1 (d), but there is no guarantee in this method and if the random-walks fail to escape, the robot returns to the same local minima and cannot reach the goal configuration. Therefore, the success of RPP depends heavily on the computation of a good navigation function, but with high dimensional constraints and obstacles, this computation becomes a difficult task.

2.2.2 Probabilistic Roadmaps

Probabilistic Roadmaps (PRM) [13] is designed for multiple queries in the same environment and it has a substantial pre-computation step as shown in figure 2 to enable numerous path-planning queries to be solved.

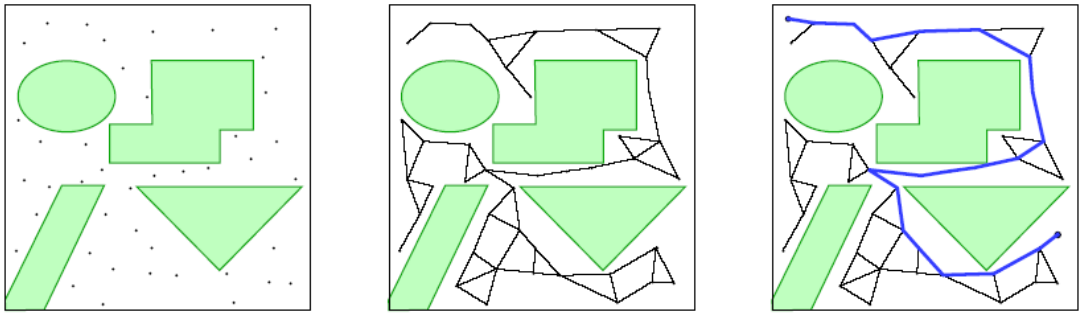

Figure 2: PRM Operation (from [13]): (a) the environment is uniformly sampled; (b) the nearest neighbors are connected; (c) the path is defined from initial to goal state

In pre-computation step, firstly, a set of collision free random points is sampled in configuration space, and then these points are connected with their neighbors by using a local planner. With these connections, a graph is constructed in configuration space and then the shortest path between initial and goal configuration through this graph is determined as shown in figure 2. However, for most dynamical systems, PRMs do not have a robust local motion planner and while constructing the graph, connecting the random points through the narrow passage is difficult that is the side effect of stochastic sampling of random point. Also for complicated and nonholonomic dynamical systems,

connection problem can be as difficult as designing nonlinear controller and the connection of thousand of configurations can be required to find solution path. This shows that PRMs are not suitable to solve problems in this kind of systems.

2.2.3 Rapidly Exploring Random Trees

Rapidly Exploring Random Trees [10, 11] is developed by LaValle to solve motion planning problems that have nonholonomic constraints and high degrees of freedom. This method is based on the incremental construction of a search tree that attempts to rapidly and uniformly explore the state space. The basic algorithm of RRT is given as follows:

```
BUILD_RRT(x_init)
 1  T.init(x_init);
 2  for k = 1 to K do
 3      x_rand ← RANDOM_STATE();
 4      EXTEND(T, x_rand);
 5  Return T
```

```
EXTEND(T, x)
 1  x_near ← NEAREST_NEIGHBOR(x, T);
 2  if NEW_STATE(x, x_near, x_new, u_new) then
 3      T.add_vertex(x_new);
 4      T.add_edge(x_near, x_new, u_new);
 5      if x_new = x then
 6          Return Reached;
 7      else
 8          Return Advanced;
 9  Return Trapped;
```

Figure 3: The Basic RRT Algorithm (from [11])

This algorithm is a simple iteration and in each step of this iteration, RRT is attempted to extend to a new point that is randomly selected by using a random sequence that is typically uniformly distributed random sequence. The EXTEND function finds the

nearest point in RRT and makes a motion towards randomly selected point ‘x’ by applying an input ‘u’ for some time increment as shown in figure 4.

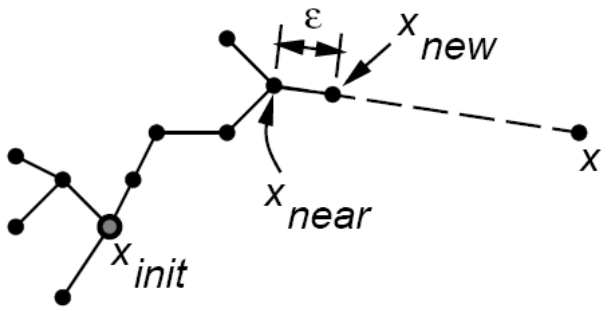

Figure 4: The Extend Operation (from [11])

The input ‘u’ for extend operation can be selected randomly or selected one of the possible inputs that is the closest input to the ‘x’ state. Also, EXTEND function includes the collision detection algorithm that checks the collision condition for new state and if no collision occurs, RRT extends towards the new state and new edge is added to RRT. This algorithm provides a tendency to grow towards unexplored portions of the state space as shown in figure 5.

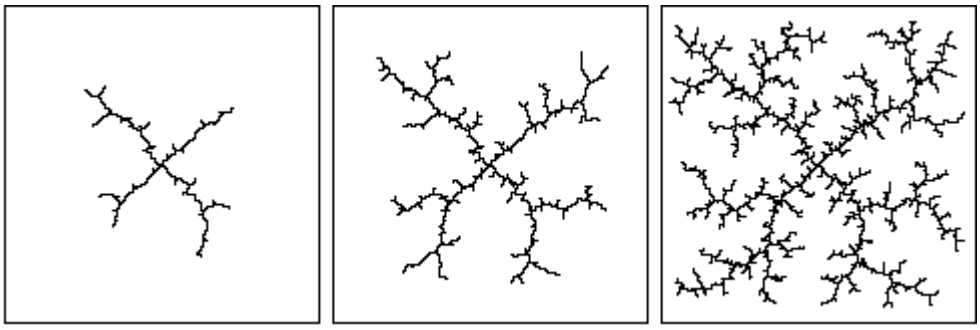

Figure 5: The Growth of RRT (from [10]): RRT grows biasing towards the unexplored regions in the state space

And this tendency is the key idea of the RRT algorithm, therefore the search tree rapidly and uniformly grows in state space to reach goals while avoiding obstacles and satisfying the constraints of the environment.

When we compare RRT with RPP and PRM, it is seen that RRT can easily drive forward like RPP and explore the free space rapidly and uniformly like PRM; furthermore it does not require a well-defined navigation function or any local planner for the connections between pairs of configurations. The RRT algorithm can be considered as a path planning module, which can be adapted into a wide variety of planning systems. And the successful applications of RRTs up to twelve degrees of freedom for holonomic, nonholonomic and kinodynamic motion planning problems are presented in studies [10, 11, 12], so we consider that the RRT algorithm is suitable for motion planning problems in high dimensional state spaces. However, this algorithm has some issues that are RRT performance dependence on sampling, slow rate of RRT convergence, inefficient collision avoidance, and undesired and unintuitive final path, and we find some methods to resolve them as told in the following section.

2.2.3.1 RRT Issues and Solutions

2.2.3.1.1 RRT Performance Dependence on Sampling

While implementing the RRT algorithm, we discover that the performance of this algorithm is heavily dependent on sampling. This is unsurprisingly true, because RRT is a sampling based motion planning technique. So it is necessary to use a good sampling technique in the RRT algorithm that produces samples in low discrepancy and dispersion [14]. Discrepancy is caused by the inconsistent behaviors of samples and it is the main issue of random sampling. In some cases, we obtain solution with few samples, and in other cases, we need many samples to reach solution. This issue changes the computation time of the algorithm. So the discrepancy level is important for the sampling techniques and this level can be measured by examining the samples in configuration space as shown in figure 6.

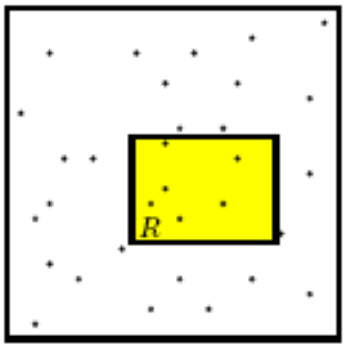

Figure 6: The Representation of Discrepancy (from [14]): The right number of points fall into box defines discrepancy

In this figure, discrepancy is defined by checking whether the right numbers of points fall into a box and using this method, sampling techniques in low discrepancy can be developed. Another issue of random sampling is dispersion that is related to the resolution of samples in configuration space and can be determined by using the method shown in figure 7.

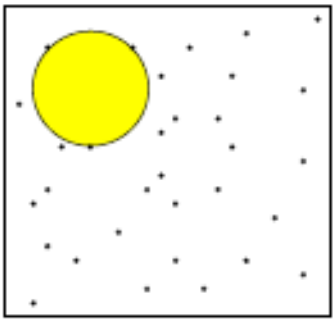

Figure 7: The Representation of Dispersion (from [14]): Dispersion is the radius of the largest empty ball.

Figure 7 represents dispersion as the largest empty ball that is created by the samples in configuration space. And for low dispersion, the largest distances between samples need to be decreased. LaValle [14] proposes that there is strong relationship between discrepancy that is measured based and dispersion that is metric based; and proves that discrepancy is a subset of dispersion. So low discrepancy implies low dispersion, and

this motivates people to develop low discrepancy sampling methods. One of these methods is a quasi random sequence, Halton that is an n-dimensional generalization of the van der Corput sequence. This sequence is used as a way of derandomizing RRTs as told in [7] and it is constructed by using a deterministic method in which prime numbers are taken as its base, and a uniformly distributed and stochastic-looking sampling pattern is generated. Figure 8 illustrates the first 196 Halton points in R^2.

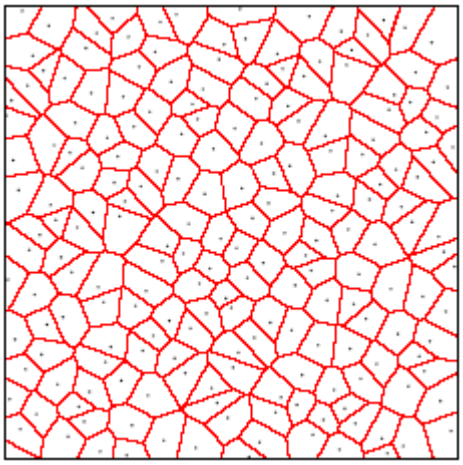

Figure 8: The Representation of Halton Sampling (from [14]): The first 196 Halton points in R^2.

As shown in figure 8, Halton sampling produces asymptotically optimal discrepancy and achieves more regularity than random sampling since it uses a deterministic method to generate samples. So Halton sampling is a nice alternative to random sampling and improves the performance of the RRT algorithm which will also be shown by a comparison between Halton and Uniform Random sampling, in chapter 4.

2.2.3.1.2 Slow Rate of RRT Convergence

One of the main difficulties in the RRT algorithm is the rate of RRT convergence. RRT grows rapidly by biasing towards unexplored part of the state space, but its convergence rate is slow especially for environments with complex constraints. An efficient approach, RRT-connect which is proposed in [8, 11, 14] increases the rate of

RRT convergence. In this method, instead of attempting to extend an RRT by a single step, the connect operation tries to reach until random sample while avoiding collision. And if the nearest point lies in an edge, then this point is connected to the sample as shown in figure 9.

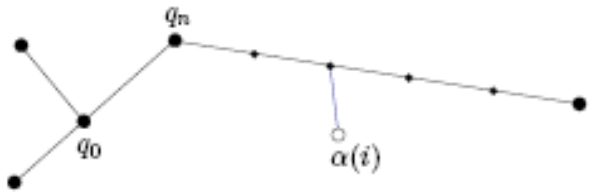

Figure 9: The Connect Operation (form [14])

To handle this, the edges are divided into vertices and added to the set of RRT vertices. And so the points in an edge can be considered in nearest neighbor search. By using this operation, RRT converges the state space faster than the extend operation and also the computation time decreases.

Another way of increasing RRT convergence is to use Goalbiased sampling as LaValle suggested in [11]. In this sampling technique, the samples are biased towards goals in the environment, so Goalbiased RRT reaches goals much faster than the basic RRT algorithm. But in the environment with obstacles, RRT can get stuck in an obstacle region because of this sampling technique that will be represented and discussed in chapter 5.

RRT convergence rate can also be improved by resolving other issues in the RRT algorithm. The primary one of these issues is the sensitivity of the performance on the choice of metrics that defines the distance between points in RRT [9]. Using inefficient metrics increase the computation time considerably. Therefore, metrics design is an important task in RRT. For ideal choice of a metric, firstly, the cost function is defined and the optimal cost function is taken as an ideal metric. This provides an efficient metrics design and so high performance of RRT can be obtained.

“Efficient nearest neighbor searching” also increases the rate of RRT convergence. While determining the nearest neighbor in RRT, simply the minimum distance is searched by considering all distances between generated sample and the points in RRT, but the computation time rises as the number of points increase. Therefore, several methods have been proposed in literature. One of these methods is the mixture of the nearest neighbor search of all points and local search in which nearest neighbor is found by searching only surroundings of sampled point [3]. Another method is to organize the points of RRT in a data structure, Kd-tree, which is multi-dimensional generalization of binary search tree [5, 6]. These methods provide efficient nearest neighbor searching algorithms and so dramatically increase the growing rate of RRT.

2.2.3.1.3 Undesired and Unintuitive Path

In RRT, the generated trajectories that connect the initial configuration to goal configuration are not optimal because of randomization. This provides inefficient motion planning. So it is necessary to make simple path smoothing to partially optimize the solution paths as suggested by LaValle in [11]. A post-processing step is added to the RRT algorithm in which small perturbations are made to the trajectory by slightly varying the inputs while satisfying the global constraints and so a nearly optimal solution path can be obtained.

2.2.3.1.4 Inefficient Collision Detection

In the RRT algorithm, the search tree explores the state space by avoiding the obstacles and as simple collision avoidance, the random samples that are in the obstacle region is directly removed and another sample is generated. But the growth rate of RRT decreases in the environment with complex obstacles, because of wasting many samples. Therefore an efficient collision detection algorithm should be used to have good performance in the environment with obstacles. LaValle suggests a Stopping-configuration procedure [14] as shown in figure 10.

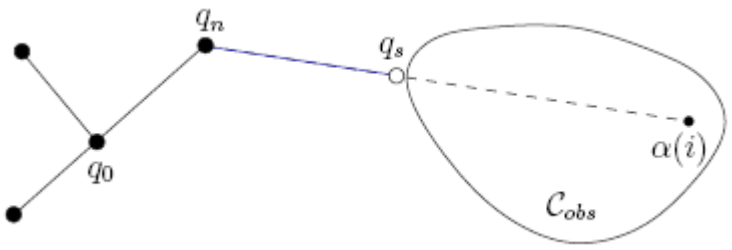

Figure 10: The Procedure Stopping-Configuration (from [14]): If an obstacle exists, the edge of RRT can grow until the obstacle boundary.

In this procedure, a branch of RRT grows towards the sampled point "$\alpha(i)$" from the nearest neighbor point "q_n", and when an obstacle is detected, it is stopped before hitting the obstacle. So new edge is made from "q_n" to "q_s" and this provides a denser RRT growth towards the boundary of obstacles, but this algorithm has an issue that computation time increases with complex obstacles. To resolve this issue, some methods are suggested in literature. One of them is using data structures to improve the distance computation in collision detection [11]. Another method proposed in [1, 2, 4] is to use local information of old branches of tree and explored regions to more effectively explore the state space avoiding obstacles. With these methods, an efficient collision detection algorithm can be defined.

CHAPTER 3

PROBLEM DESCRIPTION

In this study, our task is to solve kinodynamic motion planning problem of multiple camera inspection by using a feasible motion planning algorithm. Therefore, firstly we need to describe the environment of our problem by giving both kinematic and dynamic constraints. And then according to these constraints, we define the problem formulation of multiple camera inspection in four cases that are single camera without obstacle and with obstacles, and multiple camera without obstacle and with obstacles. Also we consider an extended case to our problem in which the cameras in multiple camera inspection can move independently with variable vertical velocities. This case is added to show that our problem can be extended to more generic problems in robotics.

In the following sections, firstly the kinematic and dynamic limits of multiple camera inspection are given to describe the environment of our problem. Then necessary formulations for a kinodynamic motion planning problem is determined by using literature and the formulations of our problem for the cases that we have considered are given and we will use these formulations in the implementation of our approach that is told in chapter 4.

3.1 The Environment of Multiple Camera Inspection

Multiple camera inspection is a large area automated optical inspection instrument that scans a large flat plate [16]. Although this instrument is applicable to the inspection of

any flat flexible media, it is particularly designed for high trough-put and in line inspection of glass plates of TFT/LCD panels.

The production of LCD panels consists of some stages that are deposition, masking, etching and stripping. During these stages, many production defects may occur that have electronic and visual implications on the final performance of LCD. These defects can be short circuits, opens foreign particles, miss-deposition feature size problems, over and under etching. And in figure 11, the most common defects are shown that are metal protrusion into Indium Tin Oxide (ITO), ITO protrusion into metal, a so-called mouse bite, an open circuit, a short in a transistor, and a foreign particle.

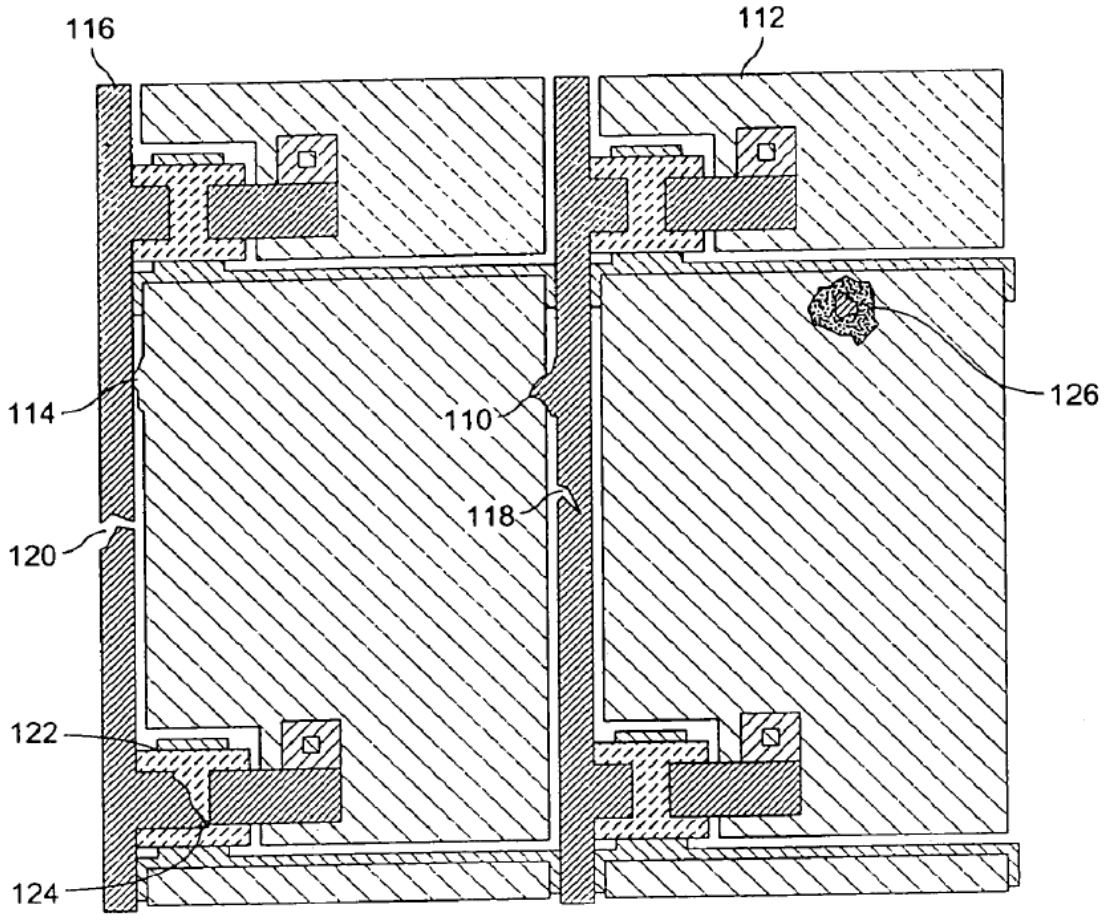

Figure 11: The Most Common Defects of a LCD (from [16]): Metal protrusion (110) into Indium Tin Oxide (ITO, 112), ITO protrusion (114) into metal (116), a so-called mouse bite (118), an open circuit (120), a short (122) in a transistor (124), and a foreign particle (126).

These defects can be as small as several microns in size and this defines the detection limits of the inspection system that is designed to detect the defects of LCD. Also, only detection of defects is not enough for the repair stage of LCD, the defects should be classified as process defects (minor imperfection) that do not undermine the finished product and can be an early indication for big damages, reparable defects that can be repaired to improve the performance of product and killer defects that disqualify product from further use.

To provide this level of detection and classification, a two stage imagining process is often required that are comparatively low resolution and high resolution imaging process. In low resolution imaging process a number of defect points of interests (POIs) over the inspected surface are detected by using a fast detection mode and in high resolution imaging process these POIs are reviewed and imaged as a part of high resolution image analysis and classification.

And these processes are achieved by the subsystems of multiple camera inspection that are Defect Detection Subsystem (DDS) and Defect Review Subsystem (DRS) as shown in figure 12. DDS is an array of 10 comparatively low resolution detection cameras and the task of this subsystem is to distinguish the defects and pre-classify them according to their location on the inspected surface. In this pre-classification, the defects can be categorized as data line, gate line, transistor, capacitor and ITO electrode. Also the results of this pre-classification can be used for prioritizing the defects by defining a review worthiness factor. DRS consists of multiple (2-6) comparatively high resolution review cameras and aims to image the defects that are identified and pre-classified to provide final defect classification.

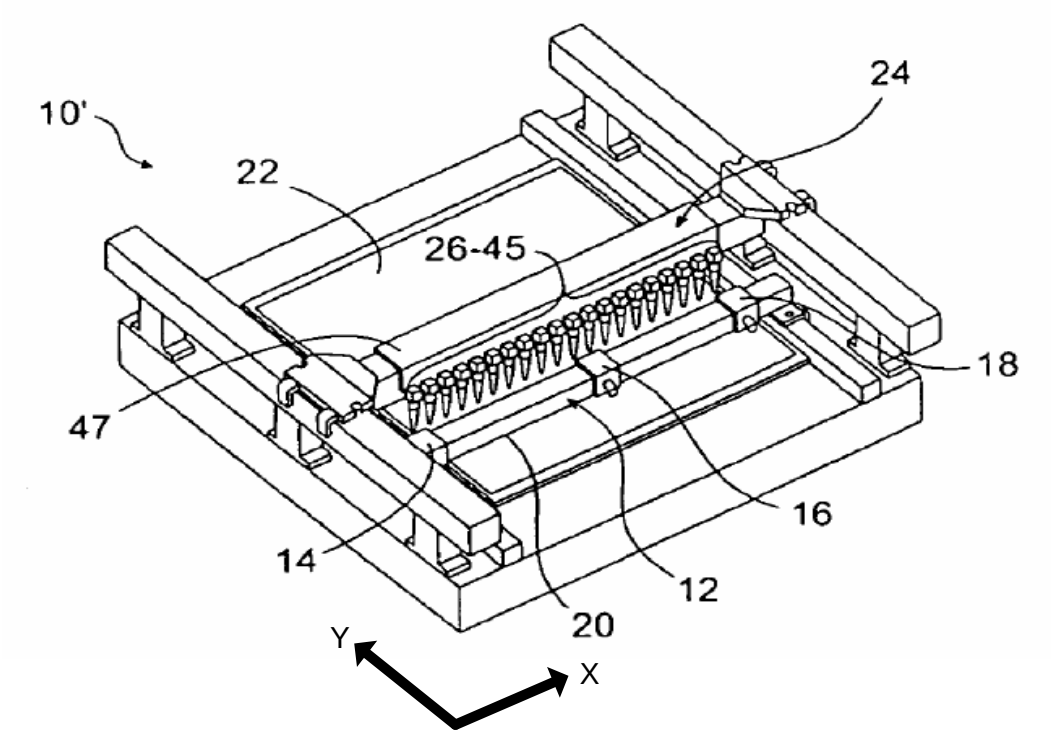

Figure 12: Multiple Camera Industrial Inspection (from [16]): It consists of detection sub-system (26-45) and review subsystem (12) that includes moving (x-axis) review cameras. The system makes multiple passes (y-axis) over inspected area.

DDS and DRS are located on a moving apparatus that performs multiple passes (y-axis) over the inspected area with a constant velocity. DDS scans through multiple passes of material motion and DRS immediately follows the detection subsystem. The review cameras in DRS can move laterally (x-axis) with respect to scan direction independently of each other and so there comes out the possibility of collision between cameras. This can be considered as dynamical obstacles that are determined according to the position and velocity of cameras. There are also static obstacles where the camera cannot be positioned because of the physical limits such as the corners, the edges and the thick regions of inspected surface.

In this study, we consider the kinodynamic motion planning problem of review cameras in DRS to make maximal use of these cameras to image as much defects POIs as possible in a given time. To solve this problem, firstly we describe the environment of

multiple camera inspection by defining the bounded map, the kinematic and dynamic constraints of moving robot (or robots), obstacle region, and initial and goal states. The summary of this description is given below:

- **Bounded map:** Inspected surface.
- **Robots:** Review cameras
    - **Kinematic constraints:**
        - 2 DOFs (x and y axis).
        - Static obstacles (physical limits over inspected surface)
    - **Dynamic constraints:**
        - Constant y-axis velocity
        - Maximum allowable x-axis velocity and Maximum allowable x-axis acceleration.
        - Dynamical obstacles (the collision region between cameras)
- **The initial and goal states:** Configurations (positions and velocities) of the review cameras and defect POIs respectively.

3.2 Problem Formulation for Multiple Camera Inspection

The formulation of a motion planning problem can be considered in two representations that are the configuration space (C-space) and the state space (X-space) according to the constraints of the environment. The C-space representation is used for the motion planning problems that have only kinematic constraints. And the problems that also include the dynamic constraints are represented in X-space. Therefore, for kinodynamic motion planning problem of multiple camera inspection, we need to use the X-space representation.

Firstly we define C-space to represent all possible position and orientation of the cameras of multiple camera inspection as given below.

$$C = (q), \text{ where q is a configuration of the robot} \qquad (2.1)$$

Then, X-space that includes the dynamic constraints of the cameras is defined by adding the time derivative of all configuration space is expressed as below:

$$X = \left(q, \dot{q} \right) \tag{2.2}$$

This shows that X-space doubles the dimensionality, and mathematics become exponentially harder. This is why the probabilistic approximate motion planning techniques; RPP, PRM and RRT have been developed that have been mentioned before.

While planning in X-space, there are also differential (nonholonomic) constraints that arise from conservation laws such as momentum conservation [12]. Using the X-space representation, the dynamics can be written as a set of 'm' implicit equations of the form below:

$$g_i(x, \dot{x}) = 0, \ for \ \ \mathrm{i} = 1,..,m \ \text{and} \ \ \mathrm{m} < n, \ where \ \mathrm{n} \ \text{is dimension of} \ \ \mathrm{X-space} \tag{2.3}$$

And the differential constraints in equation 2.3 can be expressed in a state transition equation that is proposed in [12, 14] as follows:

$$\dot{x} = f(x, u) \tag{2.4}$$

In which 'u' represents the possible inputs that are applied to the cameras for a specified time interval 'Δt' satisfying their constraints to define the new states. And the equation 2.4 is required for the incremental motion planning techniques such as RRT that is based on a state transition algorithm. Other techniques, RPP and PRM that find the solutions after a pre-computation step do not need to use this equation.

Now, we are ready to define the obstacles in the environment of multiple camera inspection in X-space. Firstly, we find a set of the configurations, C_{obst}, where the

cameras collide with the static obstacles in C-space. And if a configuration of the camera is in this set, its X-space representation is also in the set X_{obst} that defines obstacles in X-space as shown in equation 2.5.

$$x \in X_{obst} \text{ if and only if } (q) \in C_{obst} \text{ for } x = \left(q, \dot{q} \right) \tag{2.5}$$

However, there is another obstacle region in X-space. This is the region of inevitable collision, X_{ric}, in which the camera either collides with obstacle (or other camera) or can't avoid collision because of its momentum. In other words, when the camera moves in this region, there is no input that can be applied to escape from collision. An example of a point mass robot that is taken from [12] illustrates X_{ric} in figure 13.

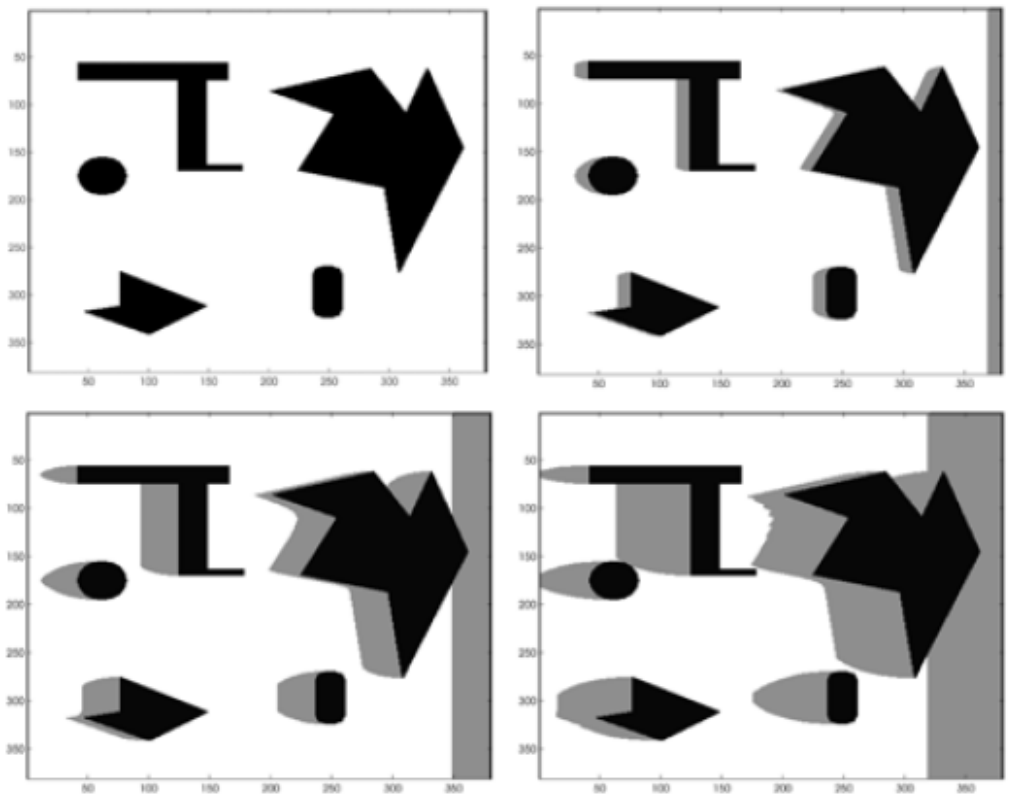

Figure 13: The Region of Inevitable Collision, X_{ric} (from [12]): This is the representation of X_{ric} for a point mass robot in two dimensions with increasing velocity (x-axis). White areas are X_{free}, black areas are X_{obst} and gray areas are X_{ric}.

This robot obeys the Newtonian laws without gravity and has L^2 bounded acceleration with an initial velocity in x-axis direction. White, black and gray areas represent the collision free, obstacle and inevitable collision regions respectively. As it is seen in figure 13, when the velocity of the robot increases, X_{ric} becomes larger and we notice that this region covers the obstacle region X_{obst} ($X_{obst} \subseteq X_{ric}$). Therefore, for kinodynamic motion planning, the region of inevitable region, X_{ric} represents the whole obstacle region in X-space and the collision free region is defined as follows:

$$X_{free} = X \setminus X_{ric} \tag{2.6}$$

To avoid obstacles, we make kinodynamic motion planning in this collision free region X_{free}. And in the following sections, the problem formulation will be given for the cases that we have considered.

3.2.1 Single Camera without Obstacle Case

This is the simplest case of multiple camera inspection problem that includes a single camera and no obstacle in the environment as shown in figure 14.

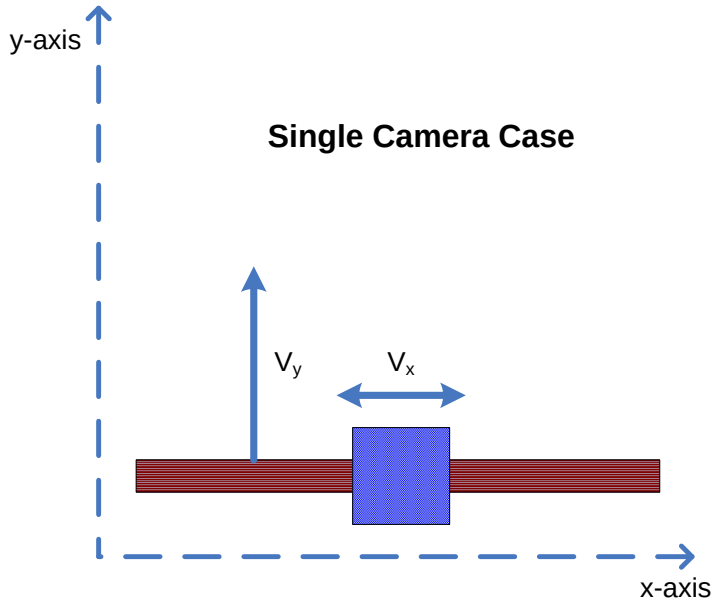

Figure 14: Single Camera Case

The review camera has 2 DOF that are in directions of x and y-axis. By using the kinematic and dynamic constraints of the camera that is defined in section 3.1, the C-space representation of the camera is given as below:

$$C = (x, y) \tag{2.7}$$

And adding the dynamics constraints of the camera, the X-space representation is defined as follows:

$$X = (x, y, V_x, V_y) \tag{2.8}$$

Since the y-axis velocity is constant, the final X-space representation is:

$$X = (x, y, V_x) \tag{2.9}$$

Using the state transition equation 2.4, the differential constraints can be determined. This function will be characterized while implementing our motion planning algorithm. And since there is no obstacle, the X-space is the collision free space, X_{free}.

3.2.2 Single Camera with Obstacles Case

In this case, we have a single camera and static obstacles that are located at specified positions of the environment. The X-space representation of the camera is same as the single camera without obstacle case. The only difference is the obstacle region because of having the static obstacles in the environment and since we use kinodynamic motion planning, it is necessary to define the collision free path in X-space. For this definition, we firstly find the region of inevitable collision and remove it from the state space. An example of single camera with obstacle case in figure 15 represents the definition of collision free path.

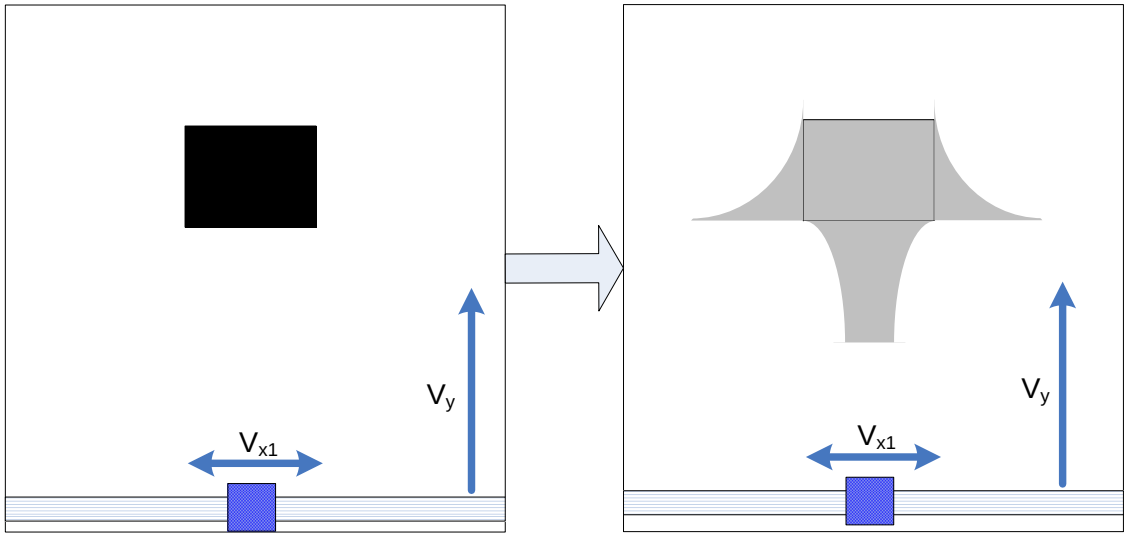

Figure 15: The Definition of Collision Free Region in Single Camera with Obstacle Case: The single camera with obstacle case is on the left and black area is the obstacle. The collision free region is represented on the right, and the gray and white areas are the region of inevitable collision and the collision free region respectively.

In this example of a single camera with obstacle case, we give just an illustration of the region of inevitable collision by considering the dynamic constraints of the camera. Since the camera can move laterally with a constant y-axis velocity, it can escape from collision by accelerating to left or right directions when it is under the obstacle and stopping the lateral motion when it moves towards the left and right sides of the obstacle, so the possible inevitable collision region can be represented as gray areas in figure 15, but the inevitable collision region depends on the current configuration of the camera that includes the position and x-axis velocity, and the computation of this region will be explained in chapter 4.

3.2.3 Multiple Camera without Obstacle Case

This case is the original configuration of multiple camera inspection problem in which there exists multiple camera and no static obstacle as shown in figure 16.

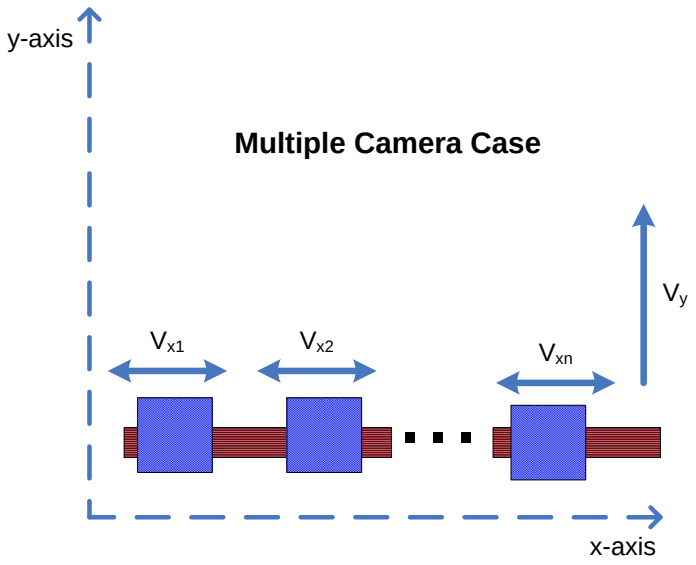

Figure 16: Multiple Camera Case

Using the same constraints in the single camera case and adding multiple camera to the environment, the C-space representation is given as below:

$$C = (x_1, x_2, ..., x_n, y) \tag{2.10}$$

By adding the time derivatives of configurations and removing the constant ones that are y-axis velocities, X-space is expressed as:

$$X = (x_1, y, V_{x1}, x_2, V_{x2}, ..., x_n, V_{xn}) \tag{2.11}$$

And the differential can be defined as equation 2.4.

Although, there is no static obstacle in this case, the dynamical obstacles arise from the possibility of the collision between cameras. We consider these obstacles as an inevitable collision region between cameras that is illustrated by an example of two cameras and no obstacle case in figure 17.

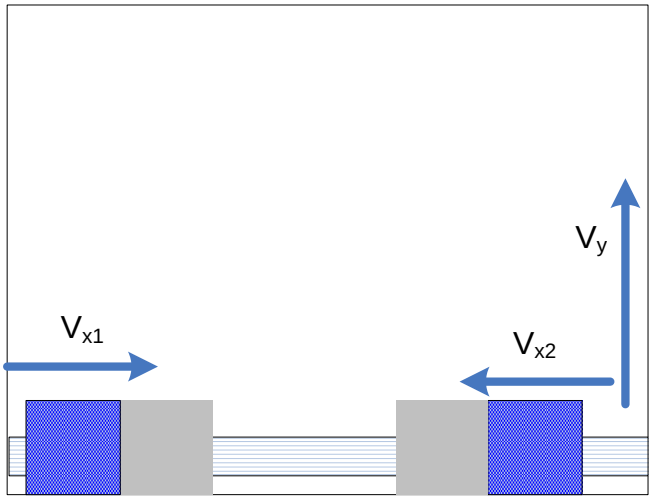

Figure 17: The Representation of Dynamical Obstacles: These obstacle regions are considered as inevitable collision region between cameras and represented by gray areas.

In this example, the stopping distance of a camera before collision is determined by assuming that it moves with the resultant velocity of the cameras and other camera is a static obstacle. This defines the inevitable collision region and so the dynamical obstacles. We find these regions for each camera with respect to its neighbor cameras, and use for collision avoidance at each state of motion planning.

3.2.4 Multiple Camera with Obstacles Case

In this case, there are multiple camera and static obstacles in the environment and the formulation is same as the multiple camera without obstacle case except the X-space representation of the obstacle. The obstacle in X-space is the region of inevitable collision in which the camera is in collision or cannot do anything to avoid collision with static obstacles (or other cameras) as shown in figure 18.

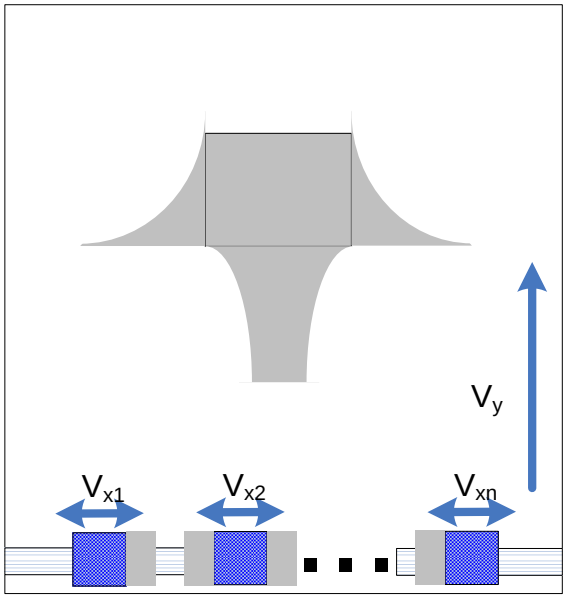

Figure 18: The X-space Representation of Obstacle in Multiple Camera with Obstacle Case: Gray areas are the region of inevitable collision region where the camera is in collision or cannot do anything to avoid collision with static obstacles (or other cameras)

The obstacle in X-space for this case can be considered as the combination of obstacles in single camera with obstacle case and multiple camera without obstacle case. For each camera, the region of inevitable collision with respect to the static obstacles and neighbor cameras are computed at each state of motion planning and the collision free region is found by extracting it from state space.

3.2.5 Extended Case (Future Work)

We consider an extended case of multiple camera inspection as a future work to show that our problem in this study can be applied to more generic robotic applications. In this case, the cameras have variable y-axis velocities as in x-axis and can move in x and y-axis independent of other cameras through the environment with obstacle as shown in figure 19.

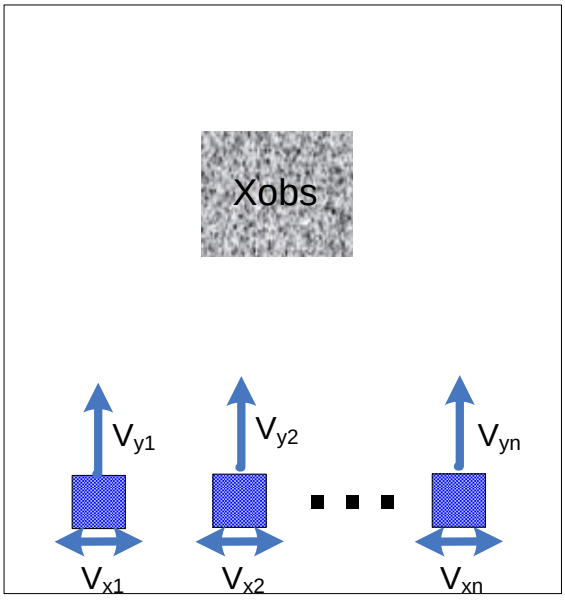

Figure 19: The Extended Case: It consists of multiple camera that move independently with variable x-axis and y-axis velocities in the environment with obstacle.

The C-space representation can be expressed as:

$$C = (x_1, y_1, x_2, y_2, ..., x_n, y_n) \tag{2.12}$$

Since the cameras can also move independently in y-axis, new 'y' constraints are included in the representation and by adding the time derivatives of configurations; X-space is given as:

$$X = (x_1, y_1, V_{x1}, V_{y1}, x_2, y_2, V_{x2}, V_{y2}, ..., x_n, y_n, V_{xn}, V_{yn}) \tag{2.13}$$

The obstacle representation in X-space consists of the same regions as in the multiple camera with obstacle case. Only the shape of this region can be changed because of the capability of having more DOF. So, it requires a complex collision avoidance algorithm that is difficult to design, but with an efficient collision avoidance algorithm, the extended case can become more successful in finding defects than the original case of multiple camera inspection.

3.3 Problem Parameters

In this section, the parameters of our problem that define the constraints of multiple camera inspection with their values are given in table 1 below:

Table 1: Problem Parameters

Problem Parameters	Values
Plate size (x, y):	1.8 m, 2 m
Number of review cameras:	1-6
Number of detection/review passes:	1-5
Maximum allowable review module x-axis acceleration:	8 m/s^2
Maximum allowable review module x-axis velocity:	0.5 m/s
Material y-axis constant velocity:	0.220 m/s
Number of defects:	50-150.

CHAPTER 4

SOLUTION APPROACH

4.1 Introduction

In this study, our aim is to make kinodynamic motion planning of the review cameras in a scanner type inspection device that is multiple camera industrial inspection to reach as much defects as possible over inspected surface while avoiding obstacles. To achieve this aim we firstly make a literature survey about kinodynamic motion planning and this survey shows an open problem that is to find a time optimal solution for kinodynamic motion planning problems. The solution to this problem is also proven to be NP-hard. This motivates us to research on the probabilistic approximate methods that are RPPs, PRMs and RRTs to find kinodynamic solutions that are close to optimal. And we select to use the RRT algorithm in our approach because of computationally hardness of other methods, and nice properties and promising results of RRT that will be told in more details.

The roadmap of this chapter is as follows: In section 4.2, we will tell our solution approach and motivation. Then we will explain the implementation of our approach by giving the implementation issues and solutions, the assumptions and the description of the main algorithm with the definition of functions and the algorithm parameters in section 4.3.

4.2 Our Approach and Motivation

Our solution approach for kinodynamic motion planning problem of multiple camera inspection is to design an RRT based motion planner to find solutions that are good enough as long as they satisfy all of the constraints. There are several reasons why we have selected the RRT technique beside other techniques, RPP and PRM. The primary reason is that the techniques RPP and PRM do not naturally extend to general problems that include differential constraints and obstacles [10, 11, 12, 13]. As mentioned in chapter 2, RPP is heavily dependent on the choice of good navigation function that will be a difficult task in high dimensional spaces and in PRM the connection problem can be as difficult as designing a nonlinear controller for nonholonomic and dynamical systems. This represents that RPP and PRM are not suitable for complex kinodynamic motion planning problems.

However, RRT is specifically designed for this kind of problems with high degrees of freedom and it rapidly and uniformly explores the state space in incremental fashion instead of having pre-processing steps like in RPP and PRM. RRT has also several properties that make it suited for wide variety of planning problems as proposed in [10]. The key property of RRT is that it searches the state space by biasing toward unexplored regions. Another property is that RRT is like a path planning module, which can be adapted into many application of motion planning. Also RRT is a simple algorithm that is easy to make its performance analysis. These nice properties of RRT provide promising results in holonomic, nonholonomic and kinodynamic planning problems of up to twelve degrees freedom as told in [10, 11]. Therefore we are motivated to use the RRT technique for the kinodynamic motion planning problem in this study. We have designed an RRT based motion planner by adapting the basic RRT algorithm to our problem and resolving the issues of RRT to improve its performance that will be told in the following section.

4.3 Implementation

In implementation of RRT based planner, we need to adapt the RRT algorithm in literature to kinodynamic motion planning problem of multiple camera inspection because of some challenges that are multiple initials and goals, and the possibility of collision between cameras. For multiple goal challenge, it is necessary to find a final path that passes through multiple goals, and we cannot achieve to connect multiple goals by using the original RRT algorithm, then we make some changes on this algorithm and discover the multiple RRT idea, in which multiple trees are created to make connections between goals. And the multiple initials are added to this algorithm by increasing the dimensions of the points in state space as defined in chapter 3. In collision avoidance part of this algorithm, the collision regions between cameras are included in the region of inevitable collision and by removing this region; we find the collision free region. And motion planning involves finding a feasible path that lies entirely in collision free region and passes through as much defect POIs as possible.

However we come upon some issues in the performance of this planner that are related to the RRT algorithm. And we make some research to find methods to resolve these issues as mentioned in chapter 2. In the following section, we will tell how to use these methods to improve the performance of our model and give some results to evaluate these methods.

4.3.1 Implementation Issues and Solutions

While analyzing our RRT based planner, we see that there are some issues caused by the drawbacks of the RRT algorithm. The main issue is the performance dependence of RRT on sampling. The generated samples of Uniform Random Sampling do not cover the state space completely and in some cases we cannot reach solutions. Also the solutions are very inconsistent that is a big problem of randomization. Therefore, we search for a sampling technique that provides low dispersion and low discrepancy, and find a quasi random sequence, Halton as told in chapter 2. It is useful for incremental

sampling techniques like RRT that can be seen from the results of our simulations in figure 20.

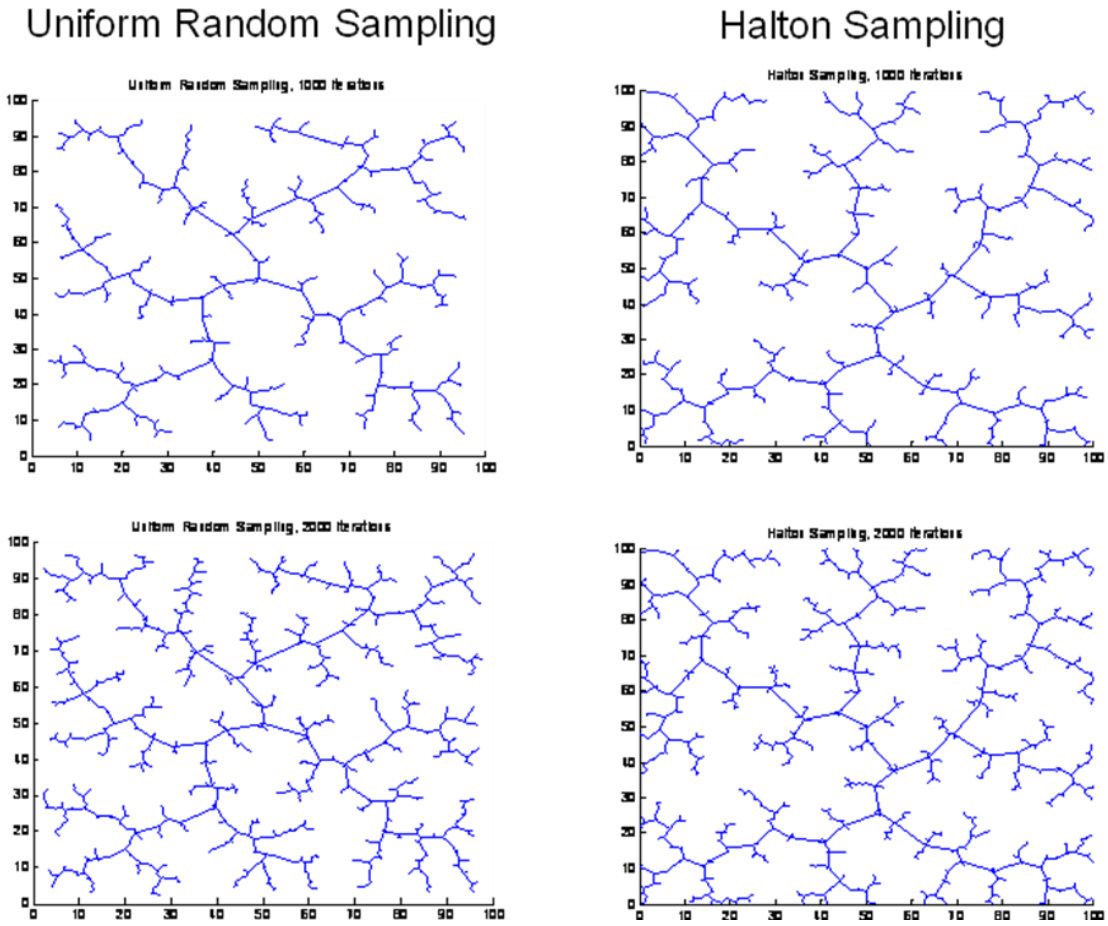

Figure 20: Uniform Random Sampling vs. Halton Sampling (from our simulations): 1000 (up) and 2000 (down) iterations of RRT in 100x100 map.

This figure presents a comparison between Uniform Random and Halton samplings. It is seen that RRT does not grow to the edges of the map, while using Uniform Random sampling and there come out spaces in these places, but in Halton sampling, RRT grows towards every places of map and a resolution complete sampling can be obtained. Also there is no inconsistent solution in Halton Sampling since it uses a deterministic method to generate samples. So we reach a solution with a same number of samples

and the computation time does not change. This shows that using Halton Sampling improves the performance of our algorithm.

Another important issue is the slow rate of RRT convergence that is caused by slow growth of RRT. To resolve this issue, we use the connect operation, in which generated samples are directly connected to the nearest points in RRT and the edges are divided into vertices to increase density. This is shown in figure 21 that is taken from our simulation result of the comparison between extend operation and RRT-connect operation.

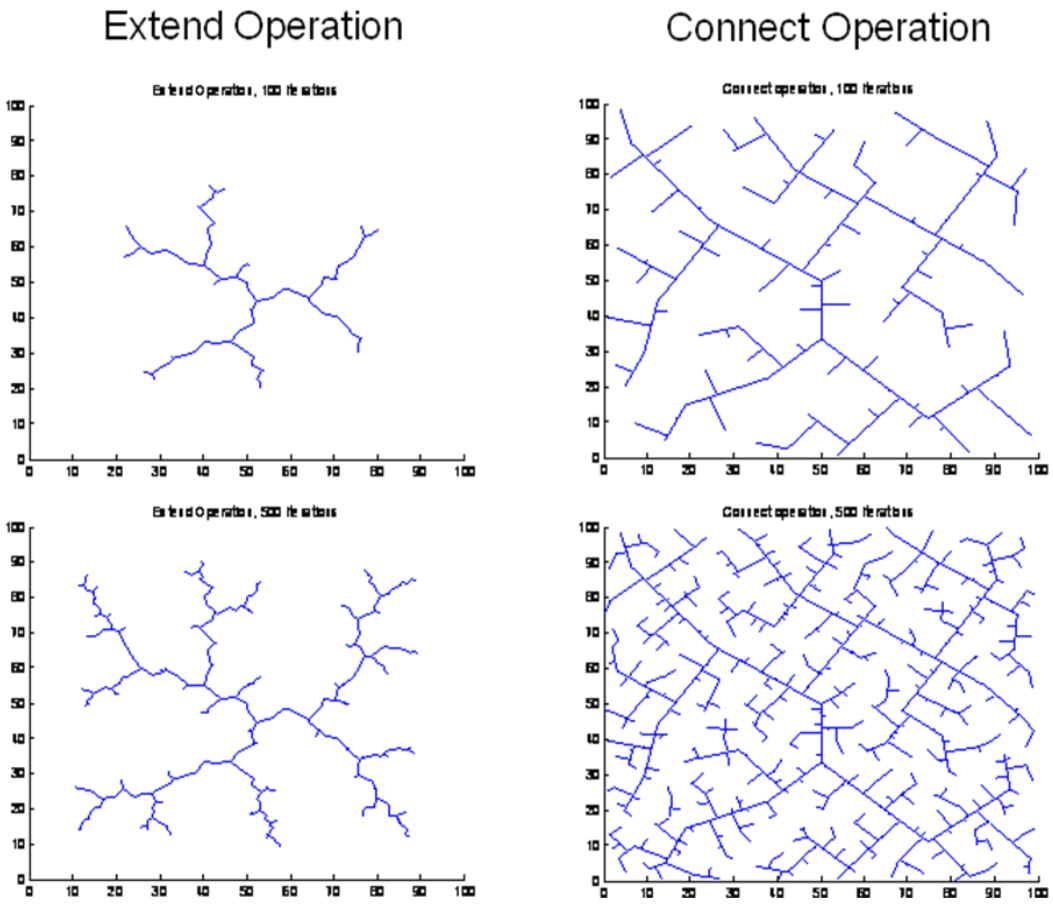

Figure 21: Extend Operation vs. Connect Operation (from our simulations): 100 (up) and 500 (down) iterations of RRT in 100x100 map.

In this figure, we see that connect operation improves the growth of RRT and the density of RRT is increased, so it converges the state space rapidly and the computation time to reach goals is decreased.

Also, we find other ways that are Goalbiased sampling, efficient choice of metrics and efficient nearest neighbor searching to increase the rate of RRT convergence. Goalbiased sampling, in which samples are biased towards goals, accelerates the RRT growth to find solution. We implement this sampling technique by using the Gaussian distribution that is biased towards goals with a specified standard deviation. A 2D Goalbiased behavior is illustrated in figure 22.

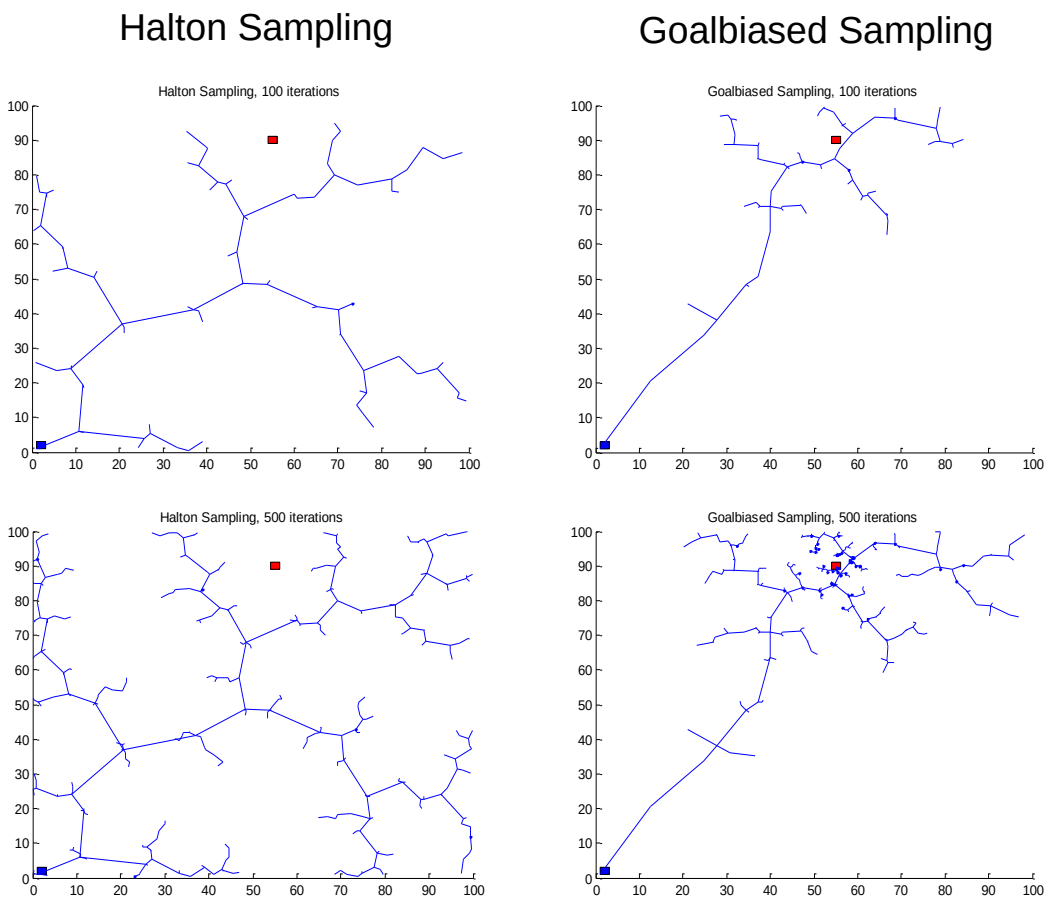

Figure 22: Halton Sampling vs. Goalbiased Sampling (from our simulations): 100 (up) and 500 (down) iterations of RRT in 100x100 map.

As it is seen in figure 22, RRT grows faster towards goal in Goalbiased sampling and this increases the rate of RRT convergence, but in the environment with obstacles RRT can get stuck. To solve this problem, we increase the variance of Goalbiased sampling and also we define new sampling techniques that are the percent and the interleaving mixture of Halton and Goalbiased samplings that will be told later in more details. In our algorithm, we implement Euclidean metrics and original nearest neighbor searching, in which all points in RRT are searched to find nearest point, without using any methods of efficient metrics design and efficient nearest neighbor searching, because the RRT convergence rate is enough to reach solution for our problem, and we consider the efficiency on these parts as a future work.

Other issues of implementation are about collision detection and solution path. For collision detection, directly removing the generated sample in the obstacle region is an inefficient method. This method increases the time to find solution and making impossible to reach the goals near obstacle. And then we implement the Stopping-configuration procedure in which the edges are stopped near the obstacle region to improve the collision detection algorithm. Also, there is a problem that the generated solution path is not optimal. We add a post processing step, in which solution path is smoothed by applying the inputs that are close to optimal and satisfying the global constraints. And this is illustrated in figure 23.

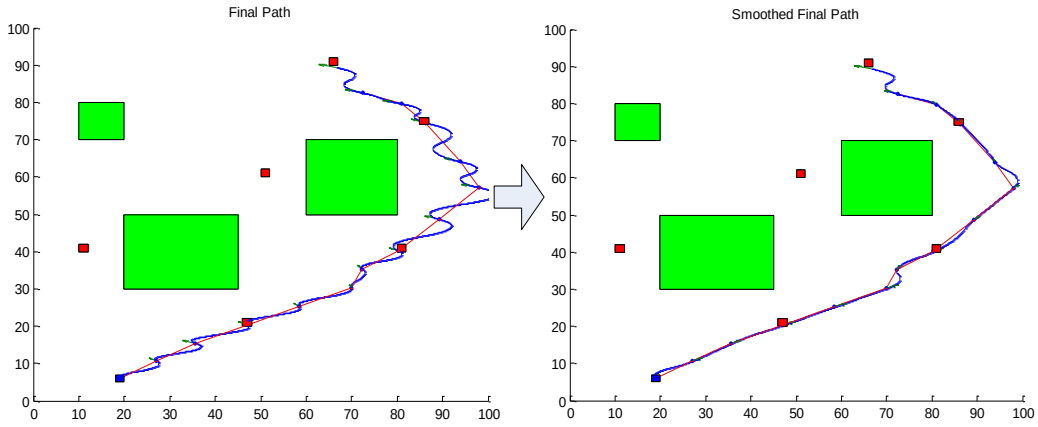

Figure 23: **An Example of Smoothing Final Path:** The plot on the left shows the final path of RRT and the smoothed path is given in the right-hand plot.

As we see in this example, the RRT algorithm provide an undesired path and arrival velocities is in one direction through the path, because while RRT is growing in state space, it has a tendency to select velocity samples in one direction that is a side effect of randomization. And by using the post processing step, we smooth the final path that is shown in the right-hand plot of figure 23, so “near optimal” trajectory is obtained.

4.3.2 Assumptions

Before constructing the algorithm for our RRT based motion planner, we consider some assumptions that are given below:

- Inspection instrument performs one pass over scan area starting from one side (bottom in experiments).
- Y-axis velocity is always constant.
- All defect POIs are assumed to be known at the beginning of the motion planning.
- Review cameras image defects within a tolerance area (assumed to “catch” the defect once in that area).
- X-axis settling time and auto-focusing allowance times of review cameras are not considered.

4.3.3 The Description of the Main Algorithm

As we told before, we consider implementing the multiple camera inspection instrument in four cases that are single camera without obstacle and with obstacle; and multiple camera without obstacle and with obstacle. We define constrains for these cases that are given in the problem formulation, chapter 3 and then we construct a main algorithm that satisfies these constraints. In this algorithm, the key idea is to use multiple RRTs that grow independently. Since our problem involves capturing multiple goals, we need to find a final path that passes through multiple goals. And using one search tree does not satisfy this, because a tree has tendency to reach a sample point from its nearest branch and two goals at different location cannot be connected through

a continuous path, so we use the multiple RRTs technique in the main algorithm as shown in figure 24.

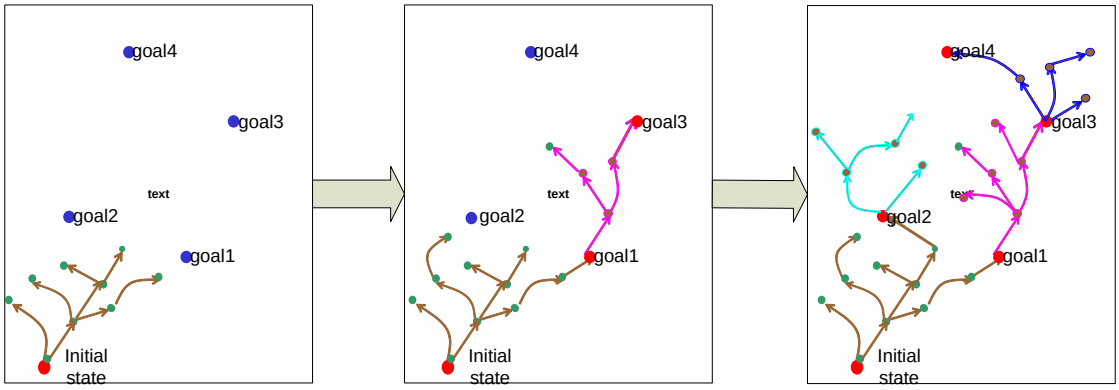

Figure 24: The Multiple RRTs Technique: In this technique, when tree is reached to a goal configuration, a new tree is started to grow from this configuration and so the connection between goals is defined.

In this technique, a tree grows starting from initial configurations as defined in the RRT algorithm of literature. When a goal configuration is reached, this configuration is taken as a root for a new tree and also the number of reached goals until this root is recorded. At each state of algorithm, a sample is generated in the state space and the nearest tree grows towards this sample. Trees are independent from each other and are aimed to reach goals that are above their roots, while satisfying the constraints of the environment and avoiding obstacles. Finally the connection between goals and initial configuration is defined. We implement this technique in the main algorithm and the flowchart of the main algorithm is given in figure 25.

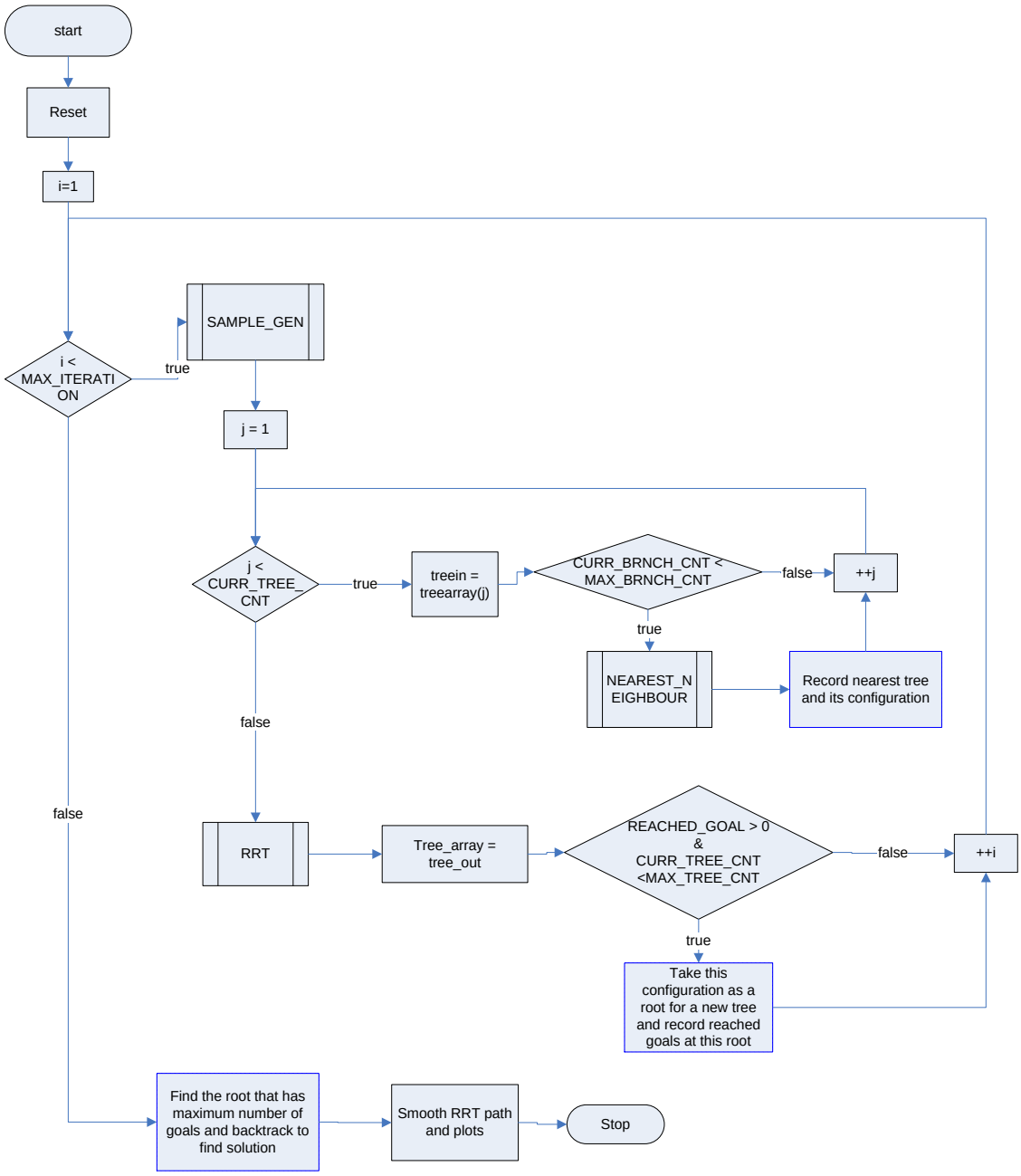

Figure 25: The Flowchart of the Main Algorithm

In this algorithm, the kinodynamic motion planning of multiple camera inspection is performed and no motion takes place before all planning is finished. As seen in figure 25, the algorithm starts with an initial reset that involves defining the dimension and variables of a configuration in RRT by using the X-space representation of the camera as told in chapter 3, and initial configuration, and other constant and variable

declarations. Then it enters into a main for loop and in this loop firstly a sample point is generated in state space by using SAMPLE_GENERATOR function. Another for loop is performed to find the nearest tree to the generated sample by calling NEAREST_NEIGHBOR function for each tree in state space. And the nearest tree is selected to grow by using RRT iteration function. This function attempts to connect its nearest point to the sample while satisfying the kinematic and dynamic constraints of cameras and avoiding collision with obstacles that are provided by the functions VELOCITY_CHECK and MAP_CHECK. If a goal configuration is reached that is also detected by the MAP_CHECK function, this configuration is considered as a root for a new tree and the number of reached goals until this root is recorded. Also, the number of trees and branches are limited to ensure low computation time that will be shown and discussed in the experimental results, chapter 5. In the end of main loop, we obtain a final tree that consists of multiple trees with roots at specified goal configurations and has information about the number of reached goals at these roots. This is illustrated in the following result from our experiments for the single camera with obstacle case.

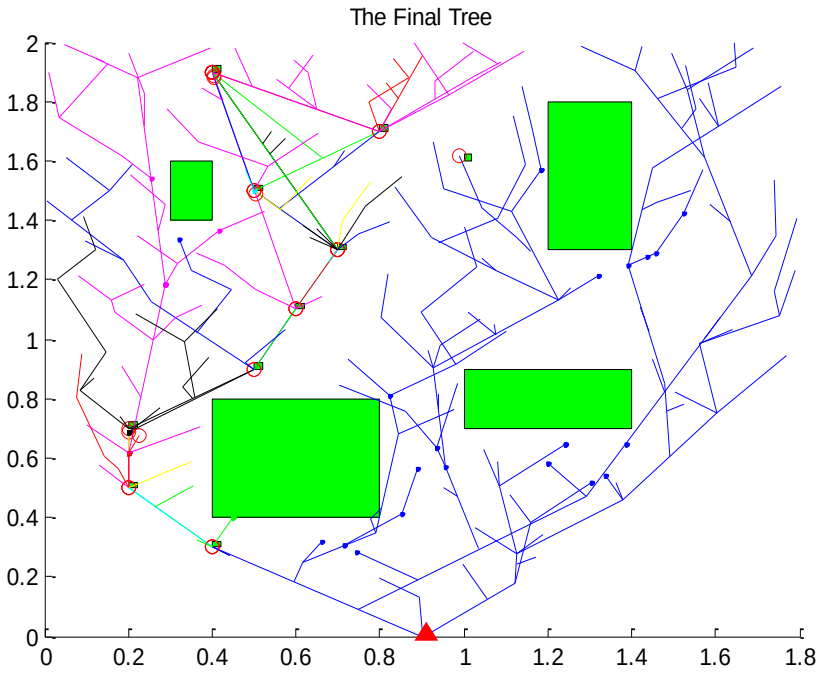

Figure 26: An Example of Final Tree Projected to 2D: It is for single camera with obstacle case. The map includes 10 defects and reached ones are shown by circles

This figure shows that the final tree defines the connection between the goals in the environment and using this final tree, we find the solution path by backtracking from the root that has the maximum number of reached goals to initial configuration. In this path, the prioritization of goals can be easily handled by defining defect values, but we consider that all defects have same weights in the experiments of this study. Then the solution path is smoothed to define nearly optimal path and this provide the final solution path for our problem as shown in figure 27.

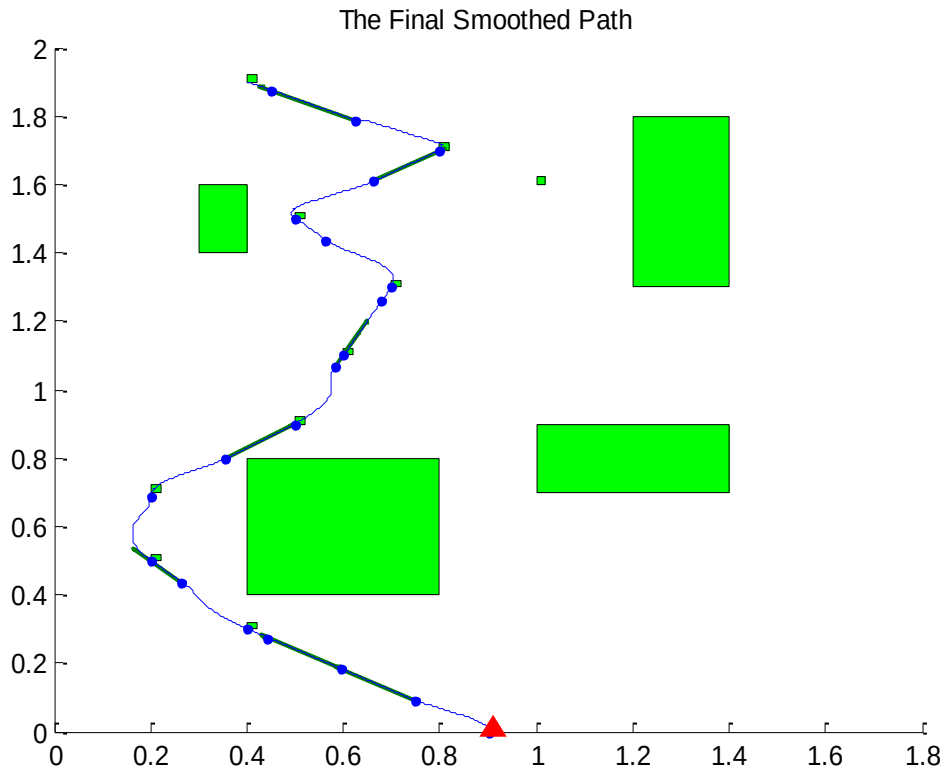

Figure 27: An Example of Final Smoothed Path: It is the smoothed solution path for the example in figure 26.

So the kinodynamic motion planning of our problem is obtained and the review cameras of multiple camera inspection are moved through this solution path to reach as much defects as possible. In the following sections, the functions of this algorithm will be told in details by giving their flowcharts.

4.3.3.1 The Definition of Functions

The main algorithm consists of some functions that are MULTIPLE_RRT, NEAREST_NEIGHBOR, SAMPLE_GENERATOR, RRT, MAP_CHECK and VELOCITY_CHECK. These functions have a hierarchic relation as shown in figure 28 with their inputs and outputs.

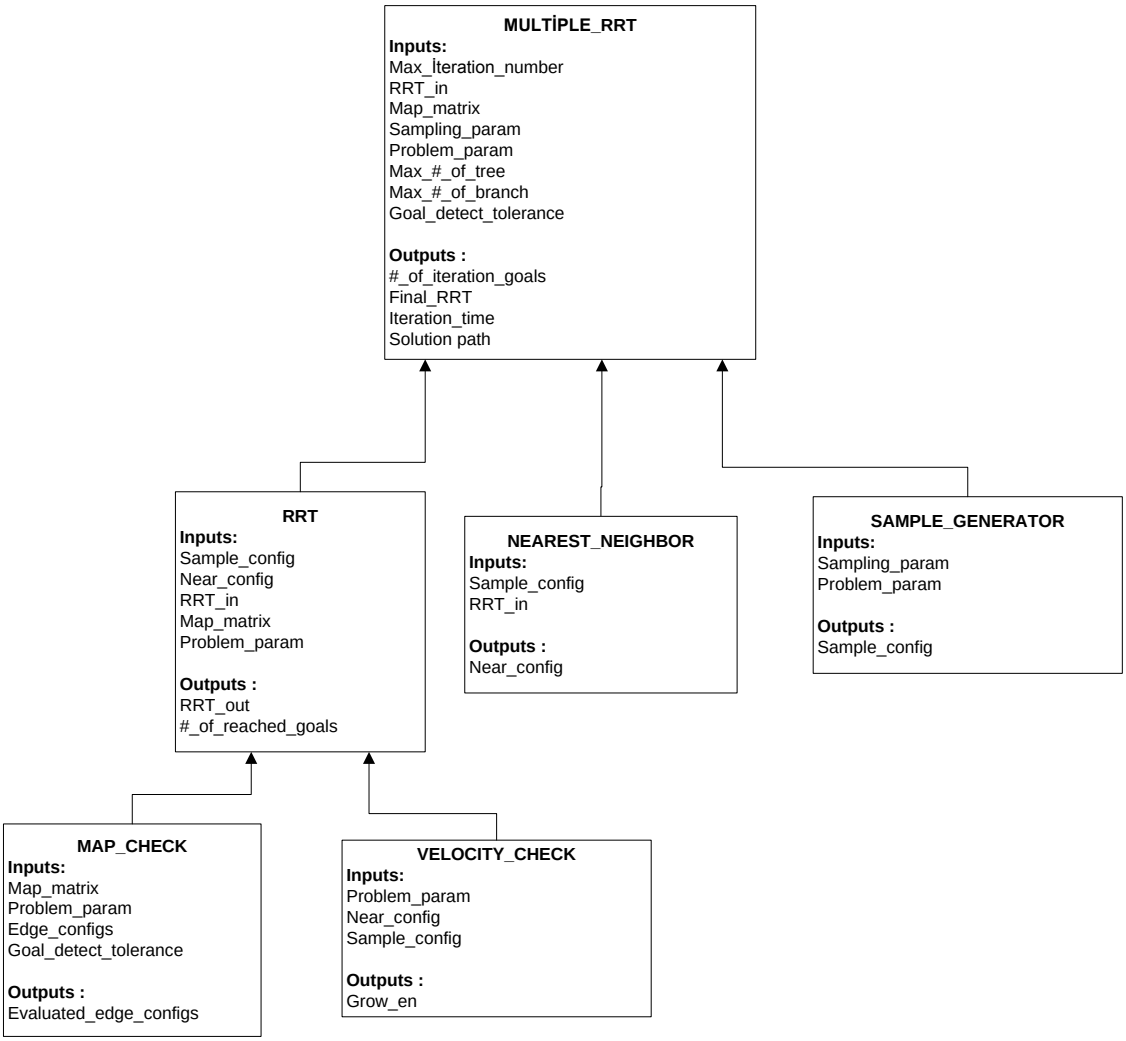

Figure 28: The Hierarchic Representation of Functions in the Main Algorithm

The top of this hierarchic structure is the MULTI_RRT function that carries out the main algorithm by using the child functions and these functions are described in the following sections.

4.3.3.1.1 The MULTI_RRT Function

The main algorithm is executed in the MULTI_RRT function by using child functions as shown in figure 28. So it is the main function of the algorithm; and the inputs and outputs can be described as follows:

- **Inputs:**
  - **Max_iteration_number:** defines the maximum number of iterations of the main for loop.
  - **RRT_in:** is the RRT input that is initially the configuration of initial state.
  - **Map_matrix:** defines the configuration of goals and obstacles in a specified map.
  - **Sampling_param:** represents the sampling parameters that include the maximum standard deviation, the percentage of Halton Sampling and the number of repetition of Halton Sampling. These parameters will be described later.
  - **Problem_param:** represents problem parameters as defined in the end of chapter 3.
  - **Max_#_of_tree:** is a limit to the number of generated trees.
  - **Max_#_of_branch:** is a limit to the number of branches of each tree.
  - **Goal_detect_tolerance**: is the tolerance distance to detect a goal.
- **Outputs:**
  - **#_of_iteration_goals:** defines the maximum number of reached goals in the end of main for loop.
  - **Final_RRT:** is the generated final RRT that consists of multiple RRTs and includes the number of reached goals in their roots.
  - **Iteration_time:** defines the spent time in the end of main for loop.
  - **Solution_path:** is the smoothed final path.

4.3.3.1.2 The NEAREST_NEIGHBOR Function

In this function the nearest neighbor configuration in RRT to sample configuration is determined. Its inputs and outputs are given below:

- **Inputs:**
  - **Sample_config:** is the configuration of generated sample.
  - **RRT_in:** is the RRT input.
- **Outputs:**
  - **Near_config:** is the nearest configuration in RRT to sample.

This function firstly finds the points in RRT that are below the sample configuration since the inspection instrument scans the surface with a constant y-axis velocity and RRT cannot grow downwards. Then by searching these points, the nearest configuration to the sample is determined.

4.3.3.1.3 The SAMPLE_GENERATOR Function

This function generates a sample point in the state space according to the sampling techniques Uniform Random, Goalbiased, Halton, the percent mixture of Halton and Goalbiased, and the repeat mixture of Halton and Goalbiased samplings. The inputs and outputs of this function are given as follows:

- **Inputs:**
  - **Sampling_param:** represents the sampling parameters that include the maximum standard deviation, the percentage of Halton Sampling and the number of repetition of Halton Sampling.
  - **Problem_param:** represents problem parameters.
- **Outputs:**
  - **Sample_config:** is the configuration of generated sample.

The sampling technique in this function is selected by using a case function as shown in its flowchart.

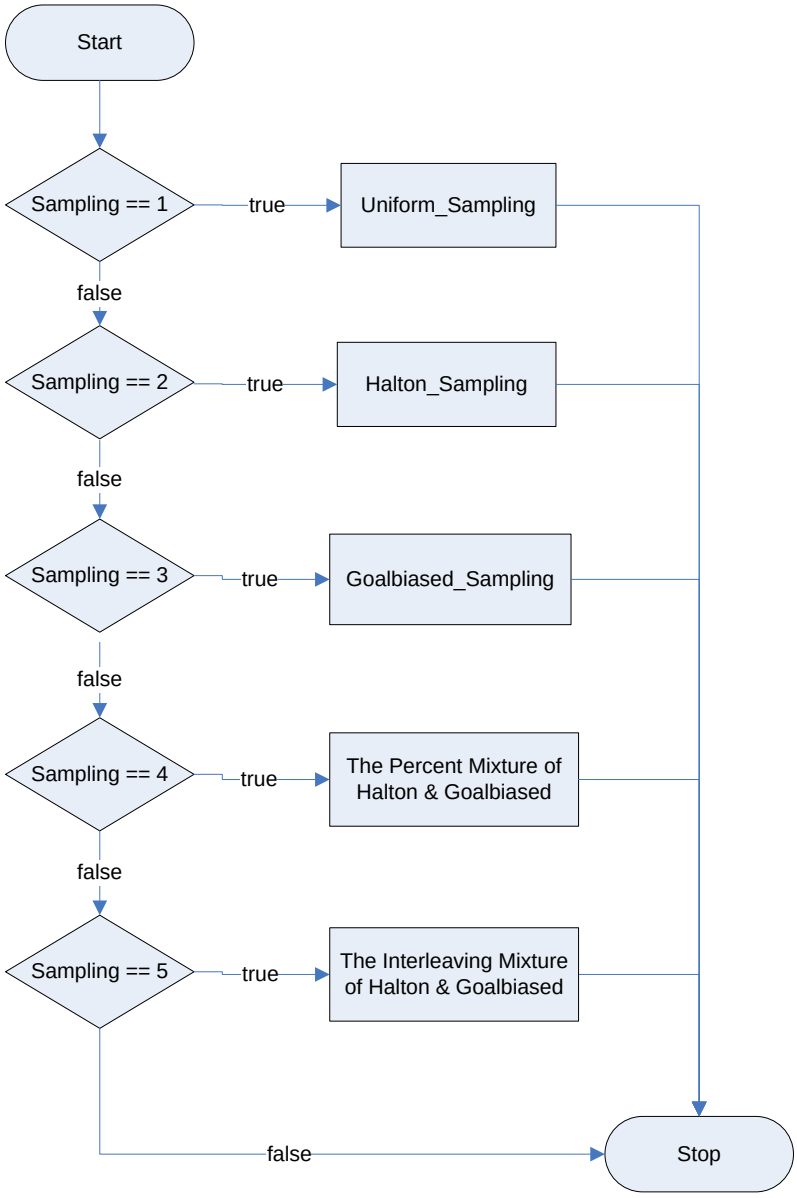

Figure 29: The Flowchart of the SAMPLE_GENERATOR Function

One of the sampling techniques and its corresponding parameters are chosen according to the case of the experiment. This function is developed to make comparison between the sampling techniques that we have considered in this study to improve the main algorithm. And in the following sections these sampling techniques are described in details.

4.3.3.1.3.1 The Uniform Random Sampling

In this sampling technique, the uniformly distributed samples are used and in our algorithm we use the MATLAB function “rand” to generate this kind of samples and it has no parameter. This technique is suggested by LaValle [10, 11] to use in the RRT algorithm, but we have some problems about the resolution of samples in the state space and inconsistent solutions as told in chapter 2. And we try to solve this by using a low discrepancy sampling method, Halton Sampling.

4.3.3.1.3.2 The Halton Sampling

This is a quasi-random sequence that uses a deterministic method to generate low discrepancy samples as told in chapter 2. We consider this sampling technique to get rid of low resolution and high inconsistency of Uniform Random sampling and as it is seen in section 4.3.1, it is useful for generating resolution complete samples, but it reaches a solution in slow rate and so we look for another sampling technique that provides faster solutions and find Goalbiased Sampling that is described below.

4.3.3.1.3.3 The Goalbiased Sampling

This sampling technique is proposed by LaValle to increase the convergence rate of RRT. In this technique, the samples are biased towards the goals in the environment. For our problem, we use Gaussian random distribution to bias the sample towards the position of goals in the state space with a specified standard deviation and other variables in the configuration of sample such as the velocities are generated using Halton sampling since these ones do not affect the goal detection that will be told in the definition of MAP_CHECK function. Starting from bottom to top, the samples are biased towards each goal for a number of times with gradually decreasing its standard deviation. And so we define the maximum standard deviation as a parameter for this sampling technique.

4.3.3.1.3.4 The Percent Mixture of Halton and Goalbiased

This is a mixture technique of Halton and Goalbiased samplings. In this technique, the first part is Halton sequence and the other part is Gaussian sequence that is biased towards goals. This technique is aimed to be used in the environment with obstacles, because while analyzing Goalbiased sampling in this kind of environments, we see a problem about being stuck in an obstacle region and so an initial Sampling technique is necessary before Goalbiased sampling to scatter the samples in the state space and decrease the possibility of encountering this problem. We use Halton sampling for this technique and define the percent mixture of Halton and Goalbiased samplings. It has some parameters that are the maximum standard deviation for Goalbiased sampling, and the percentage of Halton sampling.

4.3.3.1.3.5 The Interleaving Mixture of Halton and Goalbiased

This is another mixture technique of Halton and Goalbiased samplings that also aims to avoid getting stuck in obstacle region as the percent mixture. In this technique, firstly samples are generated by using Halton sampling for a number of times and then Goalbiased sampling for once, and this is repeated. And its parameters are the maximum standard deviation and the number of repetition of Halton sampling that defines how many times Halton sampling is applied in the first part of this mixture.

4.3.3.1.4 The RRT Function

This function attempts to construct an RRT while satisfying the constraints and avoiding the obstacles. The input and outputs of this function is given below:

- **Inputs:**
    - **Sample_config:** is the configuration of generated sample.
    - **Near_config:** is the nearest configuration in RRT to sample.
    - **RRT_in:** is the RRT input.

- **Map_matrix:** defines the configuration of goals and obstacles in a specified map.
- **Problem_param:** represents problem parameters.

- **Outputs:**
  - **RRT_out:** is the RRT output.
  - **#_of_reached_goals:** defines the number of reached goals when the RRT function is executed.

And the flowchart of this function is given as follows:

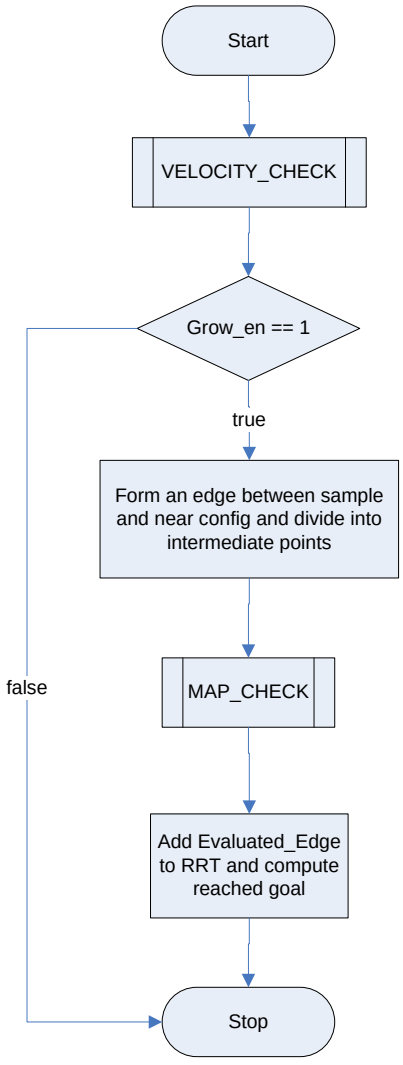

Figure 30: The Flowchart of the RRT Function

As shown in the flowchart, firstly the VELOCITY_CHECK function is called to check if the motion from the nearest configuration to sample configuration satisfies the velocity constraints that are described in chapter 3. If this is satisfied, an edge is formed between these configurations and intermediate points are inserted to provide denser RRT. These points are checked for collision and goal detection in MAP_CHECK to evaluate the edge. Then this evaluated edge is added to input RRT to form the output RRT and the number of reached goals at the end of constructing this edge is computed.

4.3.3.1.5 The VELOCITY_CHECK Function

This function checks the velocity constraints of the motion from the nearest configuration in RRT to the sample configuration before the construction of an edge. The inputs and outputs of this function are given below:

- **Inputs:**
  - **Problem_param:** represents problem parameters.
  - **Sample_config:** is the configuration of generated sample.
  - **Near_config:** is the nearest configuration in RRT to sample.
- **Outputs:**
  - **Grow_en:** is a flag that enables the construction of an edge.

In this function, we examine if a camera can move from the position and velocity of the nearest configuration to the position and velocity of the sample configuration. To do this, firstly the spent time of this motion is computed by dividing the y-axis distance between these configurations with the constant y-axis velocity. And the velocities of nearest and sample configurations are taken as initial and final velocities respectively. The spent time and the velocities are used to determine the maximum distances that can be travelled in the directions of positive and negative x-axes by considering that the camera moves in these directions with maximum acceleration, so the lateral range between the nearest and sample configurations is found that is also illustrated in figure 31.

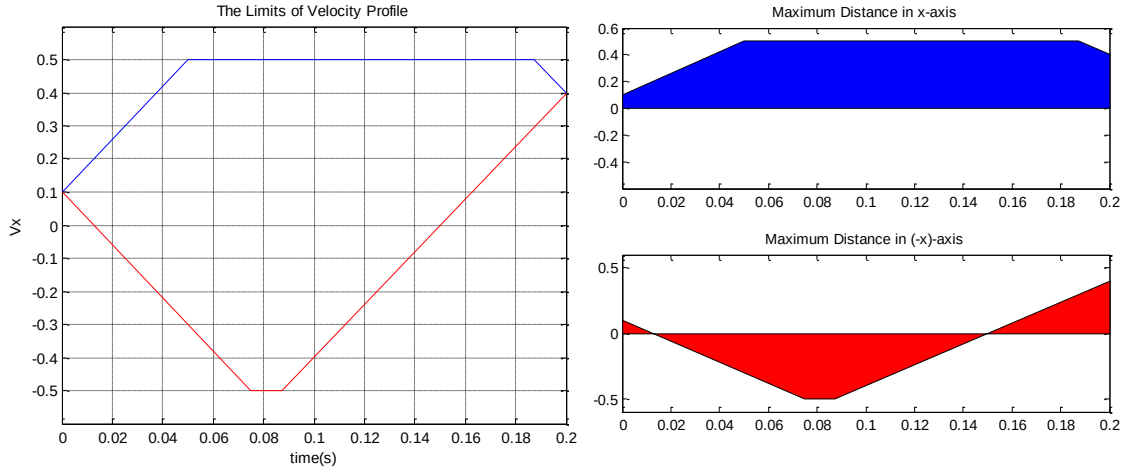

Figure 31: An Example of Lateral Range Computation: The limits are determined by moving in the directions of x and -x axes with maximum acceleration and the maximum distances in these directions are computed to find the lateral range. The x-axes velocities are limited.

In figure 31, firstly the lateral limits are found by moving in the x and -x axes directions with maximum acceleration and by computing the lateral distances in these directions, lateral range between a sample and nearest point to this sample is obtained. If the lateral motion between these configurations is in this range, connection is permitted to form an edge and this edge is divided into intermediate points to be checked in the MAP_CHECK function that is described in the following section.

4.3.3.1.6 The MAP_CHECK Function

This function provides obstacle avoidance and goal detection and it has some inputs and outputs that are given below:

- **Inputs:**
  - **Map_matrix:** defines the configuration of goals and obstacles in a specified map.
  - **Problem_param:** represents problem parameters.
  - **Edge_configs:** is an array of intermediate points that form an edge.
  - **Goal_detect_tolerance:** is a tolerance distance to detect goals

- **Outputs:**
  - **Evaluated_edge_configs:** is the evaluated edge in which the points in the obstacle region are removed and the goal configuration is defined if it is detected.

And its flowchart is given as follows:

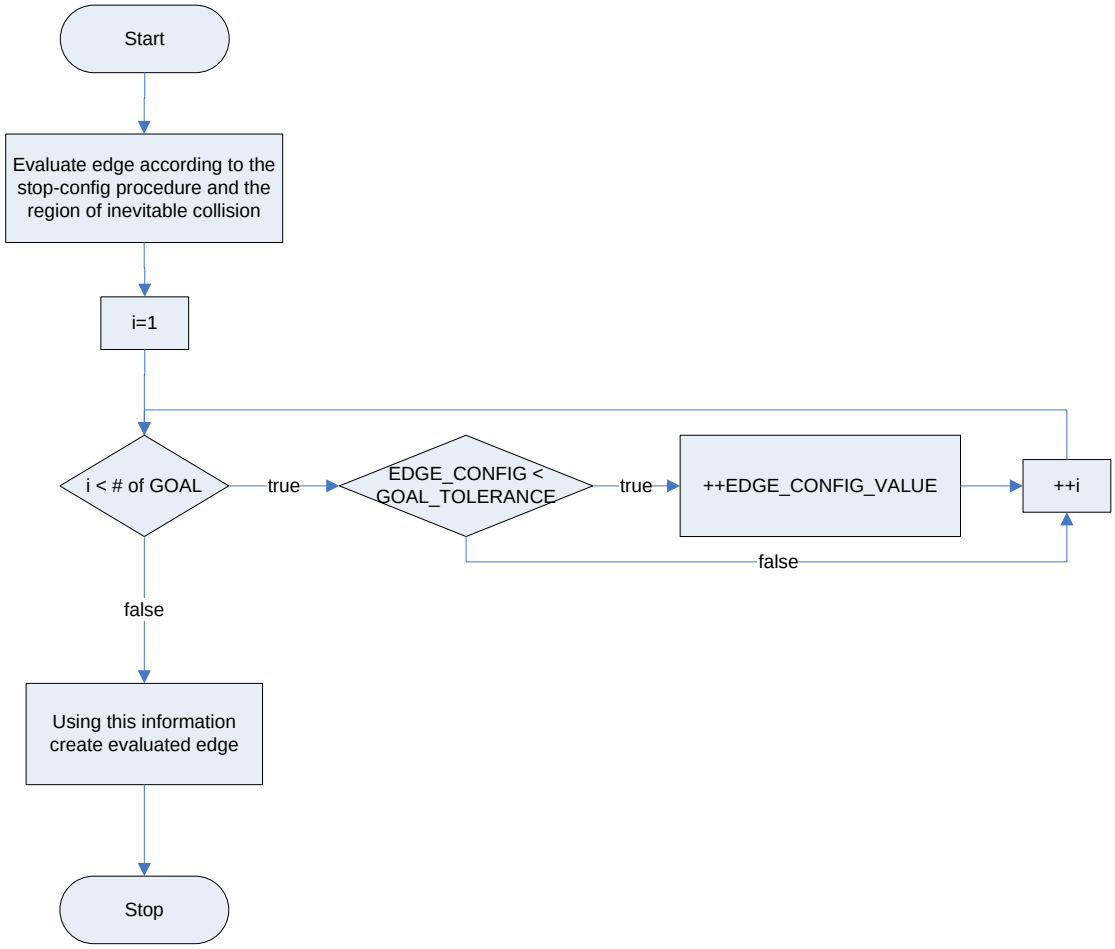

Figure 32: The Flowchart of the MAP_CHECK Function

Firstly, it takes an array of points, edge_configs that represents the edge between the nearest point in RRT and sample point. And the region of inevitable collision is defined for the configuration of each point in edge and this region includes the locations where a camera collides with obstacle (or other cameras) or can't avoid collision. We consider

three cases for the definition of inevitable collision region about an obstacle that is shown in figure 33.

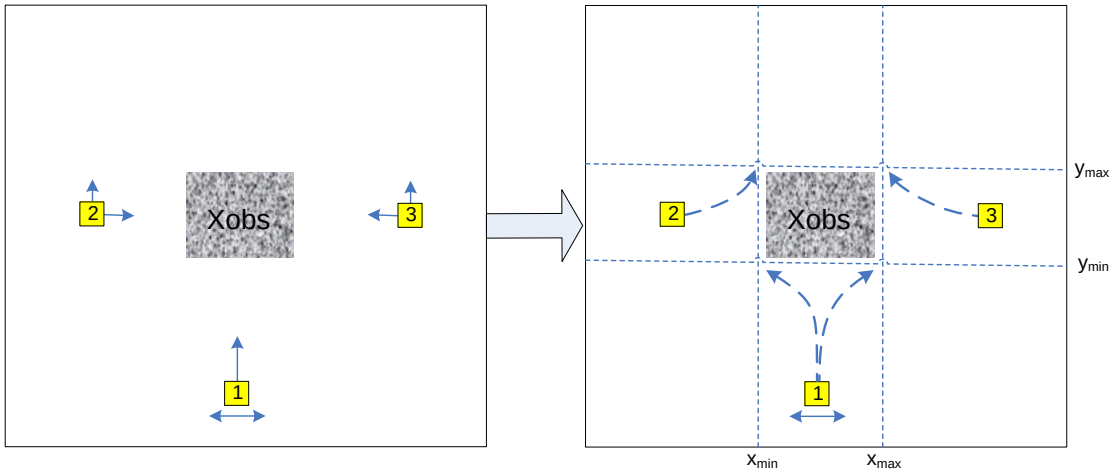

Figure 33: The Cases for the Region of Inevitable Collision: Case 1 represents a camera that is under the obstacle, in case 2 and 3 camera is on the left and on the right respectively with the velocities towards obstacle and these cases are the inevitable collision region if the escape motion profiles in the right-hand drawing cannot be followed.

These cases occur when the camera is under the obstacle, and on the left and on the right with the velocities towards the obstacle. And to determine the inevitable collision region for each configuration of the points in edge_configs about an obstacle, we check if the camera in these cases can escape from collision by following the motion profiles in the right-hand drawing of figure 33. If these profiles cannot be followed, we consider that camera cannot do anything to avoid colliding with obstacle and so its configuration is in the inevitable collision region.

In case 1 the camera can escape from collision by accelerating in the directions of x and -x axes since its y-axis velocity is constant. For this escape motion, we use the motion profile that is formed by increasing the x-axis velocity with maximum acceleration in these directions as illustrated in figure 34.

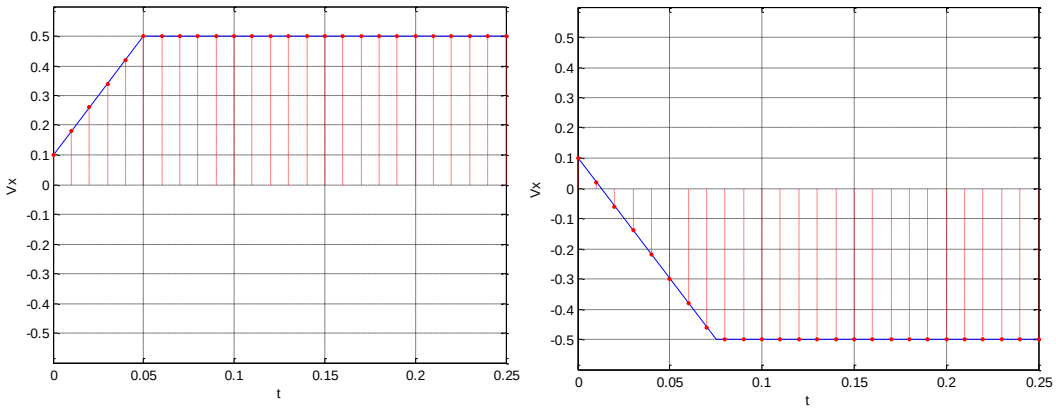

Figure 34: An Example of Velocity/time Profiles for Case 1: The profiles on the left and on the right represent motions with maximum acceleration in direction of x and -x axis respectively. The maximum time is the spent time to reach y_{min} of obstacle and also the velocity is limited to the maximum x-axis velocity.

These plots in figure 34 show the escape motions from collision by using the possible lateral motion capability of the camera. For case 2 and 3, collision can be avoided by stopping the lateral motion as shown in figure 35.

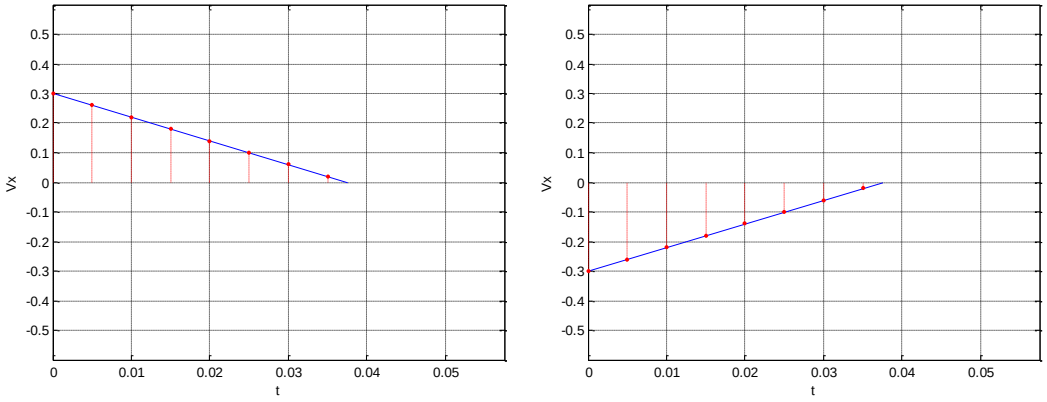

Figure 35: An Example of Velocity/time Profiles for Case 2 and 3: Stopping lateral motions with maximum acceleration for case 2 and 3 respectively.

In this profile, it is considered to decrease the lateral velocity with maximum acceleration to stop before colliding with obstacle. And these escape motions from collision are used to see whether the camera is in the inevitable collision region or in the free region. To check the collision between cameras, we consider that one of the cameras is a static obstacle and the other camera moves towards this obstacle with

resultant velocity, and the motion profiles of case 2 and 3 are used to define the inevitable collision region. So the free region is determined by removing the sum of all inevitable collision regions. Then the points of edge that are in the region of inevitable collision are removed by using the Stopping-configuration procedure that is defined in chapter 2. So new edge is obtained which is in collision free region. Then goal detection is performed on this new edge. In goal detection operation we compute the distances between the points of new edge and the goals by considering their positions in the state space, and if one of these distances is smaller than or equal to the goal_detect_tolerance, this is taken as a goal configuration and the number of reached goals are recorded. Finally evaluated_edge_configs is formed in which goal configuration and the number of reached goals in this configuration is defined.

4.3.4 Algorithm Parameters

In this section, the parameters of the main algorithm with their values that are used in our experiments are given below:

Table 2: Algorithm Parameters

Sampling Parameters	**Values**
The maximum standard deviation:	0-0.2
The percentage of Halton sampling:	%10-%100
The number of repetition of Halton sampling:	1-10
Other Parameters	**Values**
The tolerance distance in goal detection:	0.01
The maximum number of trees:	30, 50, 100, 200, 500
The maximum number of branches:	30, 50,100, 500, 1000
The maximum number of iterations:	500, 1000, 2000.

Algorithm parameters consist of sampling parameters and other parameters as shown in table 2. The sampling parameters are defined in the sampling techniques, Goalbiased sampling, the percent mixture of Halton and Goalbiased samplings, and the interleaving mixture of Halton and Goalbiased samplings. The maximum standard deviation is the

limit for the standard deviation of Gaussian distribution in Goalbiased sampling. The percentage of Halton sampling and the number of repetition of Halton sampling define the amount of Halton sampling in the percent and interleaving mixtures respectively.

Other parameters involve the tolerance distance in goal detection, the maximum number of trees and the maximum number of branches. The tolerance distance is an assumption to define a tolerance area for goal detection. The maximum number of trees and branches are considered to put limits to the number of trees in final tree and the number of branches in each tree respectively. By using these parameters, we aim to decrease the computation time without changing the solution that will be shown and discussed in the experimental results.

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Introduction

In this study, our aim is to plan motion of the review cameras in multiple camera industrial inspection to image as much defect locations as possible in a given time while satisfying the kinematic and dynamic constraints of cameras and avoiding the obstacles in the environment; and we consider this problem in four cases that are single camera without obstacle and with obstacle, and multiple camera without obstacle and with obstacle. And we implement an RRT based motion planner model for these four cases based on the main algorithm that is described in chapter 4. In the following sections, this model will be explained with its parameters, and also a deterministic method that is used for comparison with our model will be described. Then the performance measure of our experiments will be told and finally the results of these experiments will be shown with some evaluations.

5.2 RRT Based Motion Planner Model

RRT based motion planner is developed by using the main algorithm, and four cases of multiple camera inspection that are single camera without and with obstacle, and multiple camera without and with obstacle are implemented in this model. The task of this model is to make kinodynamic motion planning of the cameras in multiple camera

inspection to reach as much defect POIs as possible in a given time. In this model, we use different sampling techniques that are Uniform Random, Halton, Goalbiased, and the percent and interleaving mixture of Halton and Goalbiased samplings to obtain the best performance and analyze the parameters of this model to reach nearly optimal solutions. And in the following section, these parameters are described.

5.2.1 Model Parameters

The parameters of our model that are considered in this study are given in table 3 with the values that are used in the experiments:

Table 3: Model Parameters

Experimental Parameters	**Values**
The number of Monte Carlo, N:	10
Problem Parameters	**Values**
Plate size (x, y):	1.8 m, 2 m
Number of review cameras:	1-3
Number of detection/review passes:	1
Maximum allowable review module x-axis acceleration:	1-15 m/s^2
Maximum allowable review module x-axis velocity:	0.1-1 m/s
Material y-axis constant velocity:	0.01-0.5 m/s
Number of defects:	50
Sampling Parameters	**Values**
The maximum standard deviation:	0-0.2
The percentage of Halton sampling:	%10-%100
The number of repetition of Halton sampling:	1-10
Other Parameters	**Values**
The tolerance distance in goal detection:	0.01
The maximum number of trees:	30, 50, 100, 200, 500
The maximum number of branches:	30, 50, 100, 500, 1000
The maximum number of iterations:	500, 1000, 2000

5.2.2 Deterministic Method

In this section, we describe a deterministic method for motion planning of the cameras in multiple camera inspection that has been designed and implemented by Afşar Saranlı. This method uses a graph theoretic deterministic method to make motion planning of the cameras. The map is divided into “bands” for each camera and the collision between cameras is handled by using a post-processing step. And it may require multiple re-planning to resolve collisions. The obstacles in inspection area cannot be handled. Therefore, we consider this deterministic method as a “baseline” to compare its result with the results of our model in the environments without obstacle, and the results for these cases will be discussed to evaluate our model.

5.3 Performance Measure

We consider the following performance measures in the experiments of our model.

- The ability to generate samples in low dispersion and low discrepancy.
- The ability to avoid obstacles in the environment as well as dynamic obstacles.
- The maximum number of goals reached in a fixed time.
- The minimum time to reach fixed number of goals.

5.4 Experiments

In this section, we give the results of some experiments that are made to evaluate the performance of our model according to performance measures in section 5.3 by using different baselines and parameters.

5.4.1 The Optimization of the Sampling Parameters

In this experiment, we are aimed to optimize the sampling parameters, which are the maximum standard deviation, the percentage of Halton sampling and the number of

repetition of Halton sampling to provide the best performance in sampling techniques that are Goalbiased, the percent mixture of Halton and Goalbiased, and the repeat mixture of Halton and Goalbiased. The values of the parameters that are used in this experiment are given as follows:

- Number of review cameras: 1
- Maximum allowable review module x-axis acceleration: 8 m/s^2
- Maximum allowable review module x-axis velocity: 0.5 m/s
- Material y-axis constant velocity: 0.220 m/s
- The maximum standard deviation: 0-0.2
- The percentage of Halton sampling: %10-%100
- The number of repetition of Halton sampling: 1-10
- The maximum number of trees: infinite
- The maximum number of branches: infinite
- The maximum number of iterations: 1000
- The values of other parameters are defined in table 3

And the maps that are used in this experiment are shown in figure 36.

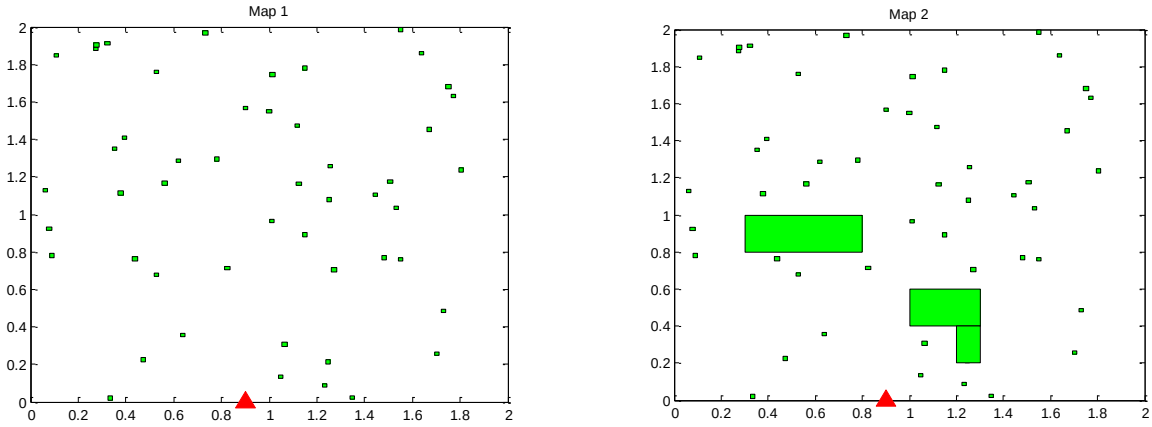

Figure 36: The Maps of the Experiment 1: Map 1 (left) and map 2 (right) includes 50 defects without and with obstacle respectively.

Firstly the maximum standard deviation is attempted to be defined for Goalbiased sampling in map1 and the result for this case is given in figure 37.

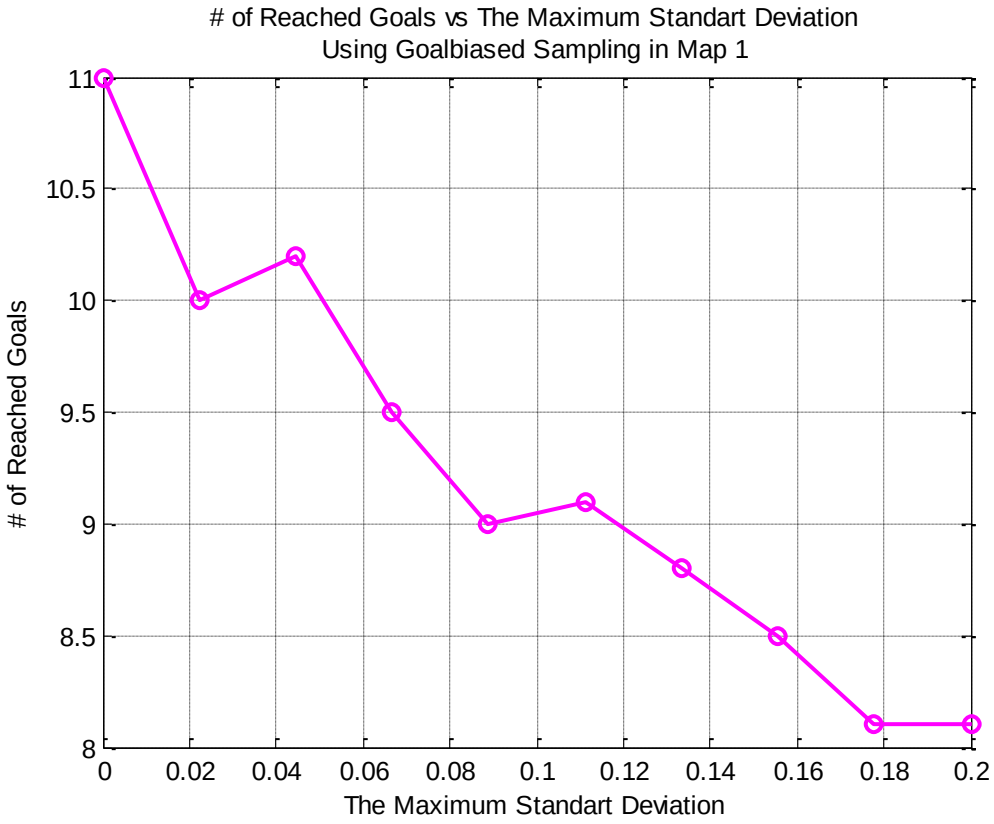

Figure 37: The Number of Reached Goals vs. the Maximum Standard Deviation Using Goalbiased Sampling in Map 1

This figure represents the number of reached goals in map 1 that includes 50 defects without obstacle for different values of the maximum standard deviation and as it is seen in the result, the performance of our model increases when the maximum standard deviation decreases and even becomes zero. And zero standard deviation means that the sample is directly biased to the position of the goals in the state space and the other variables such as velocities are generated by Halton sampling as defined in chapter 4. This is an expected result for this case, because the goals are easily detected by using this direct bias. Therefore we consider that Goalbiased sampling with no deviation is the best choice for the environment without obstacle cases.

However, when we include the obstacle to the environment, we come upon a problem that RRT can get stuck in obstacle region while it is growing in the state space. And

this is illustrated by an example in which Goalbiased sampling without deviation is used in map 2 as shown in figure 38.

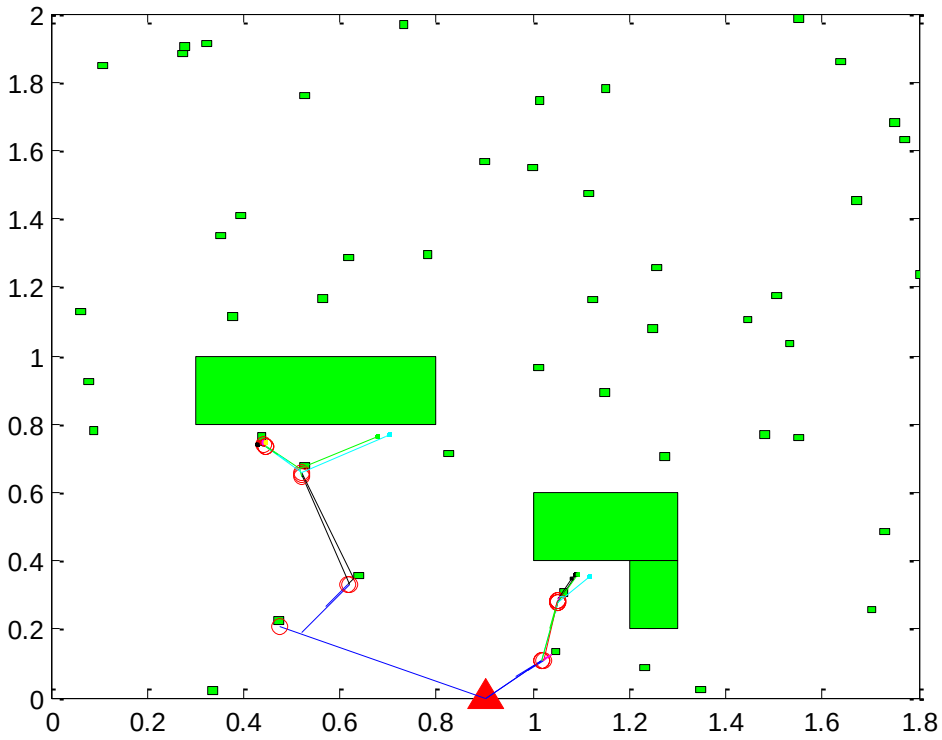

Figure 38: An Example of Stuck RRT: While using Goalbiased sampling without deviation in an environment with obstacle, RRT gets stuck in obstacle region.

In this example, RRT gets stuck in obstacle region, because when RRT reaches the obstacle region, it attempts to connect its nearest point to the samples that are biased towards goals above the obstacles and collision avoidance algorithm stops the growth of the RRT, so we get this result. To handle this problem, we consider that the standard deviation should be increased to generate samples that are far from the obstacles and this can decrease the possibility of getting stuck. Therefore, we examine the maximum standard deviation of Goalbiased sampling in the environment with obstacle, map 2 as shown in figure 39.

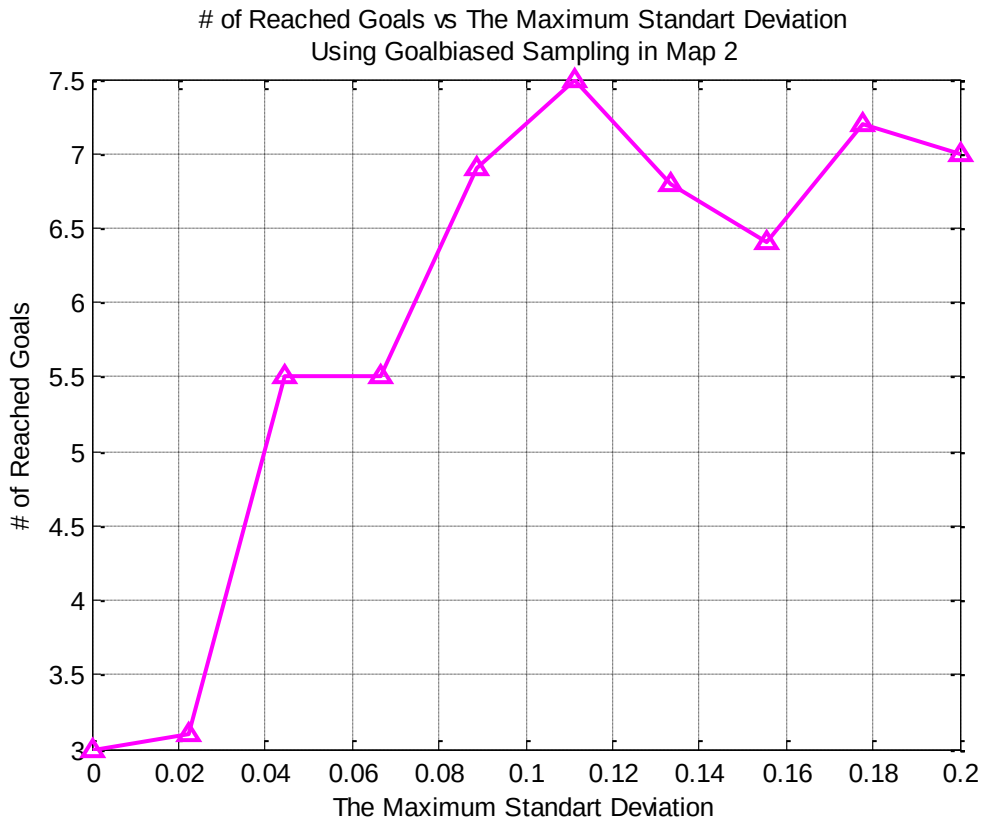

Figure 39: The Number of Reached Goals vs. the Maximum Standard Deviation Using Goalbiased Sampling in Map 2

This result shows that Goalbiased sampling in the environment with obstacle performs well when its standard deviation is increased to a middle value, because in low deviation, RRT gets stuck in obstacle region with high probability as mentioned before, and in high deviation, the bias towards goals is vanished, so by using the deviation of middle level as shown in figure 39, the best performance is obtained for Goalbiased sampling with obstacle case. However this sampling technique is not sufficient to be successful in the environments with more complex obstacles that will be shown in the performance experiments. It is necessary to have a sampling technique that generates samples with complete resolution as Halton sampling to avoid getting stuck in obstacle and also biases these samples towards goals as Goalbiased sampling to increase the RRT growth, and we design two mixture techniques, the percent and interleaving mixture of Halton and Goalbiased samplings especially for the environments with

obstacle. And in the following results, we try to determine the parameters of these techniques.

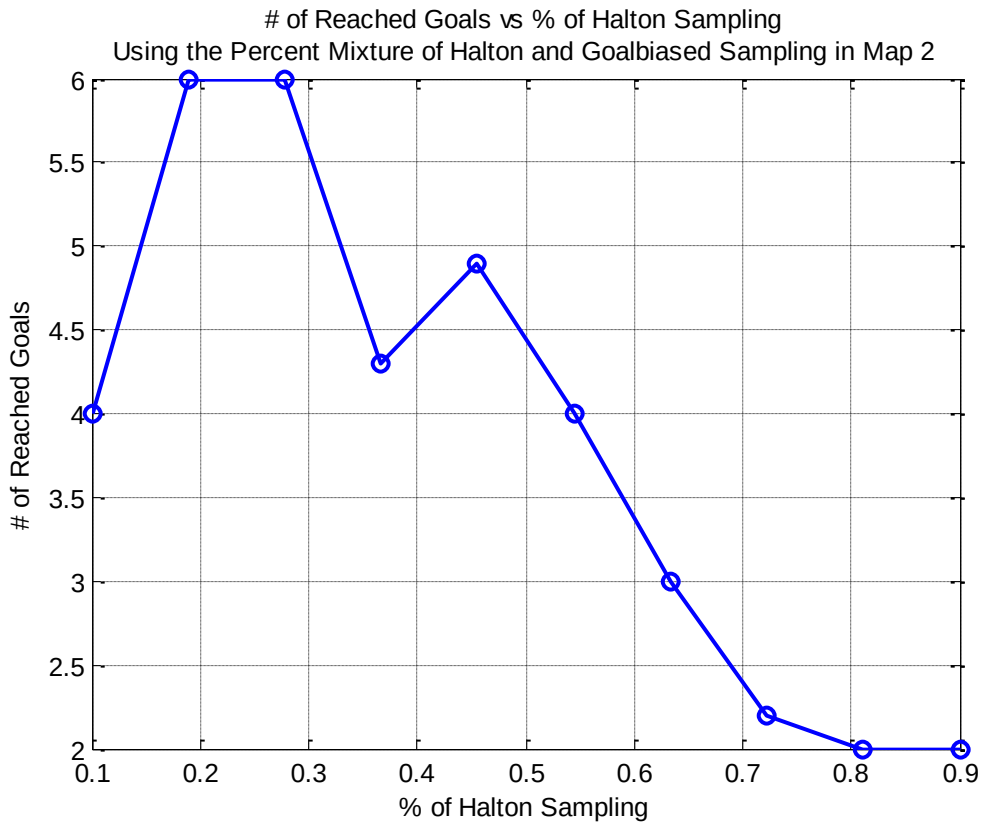

Figure 40: The Number of Reached Goals vs. the Percentage of Halton Sampling Using the Percent Mixture Sampling in Map 2

In this figure, the performance of the percent mixture sampling in the environment with obstacle is examined about different percentages of Halton sampling, and the maximum standard deviation is taken as 0.1 that is defined in figure 39. And this result shows that low percentage of Halton sampling gives the best performance, because this is enough to scatter the samples in state space to escape from obstacle region and then Goalbiased sampling is used to accelerate to reach solution. And As a result of the plot in figure 40, in the percent mixture sampling, firstly we need to use about 25% Halton sampling and then Goalbiased sampling to have the best performance. For the interleaving mixture, its parameters are defined in figure 41.

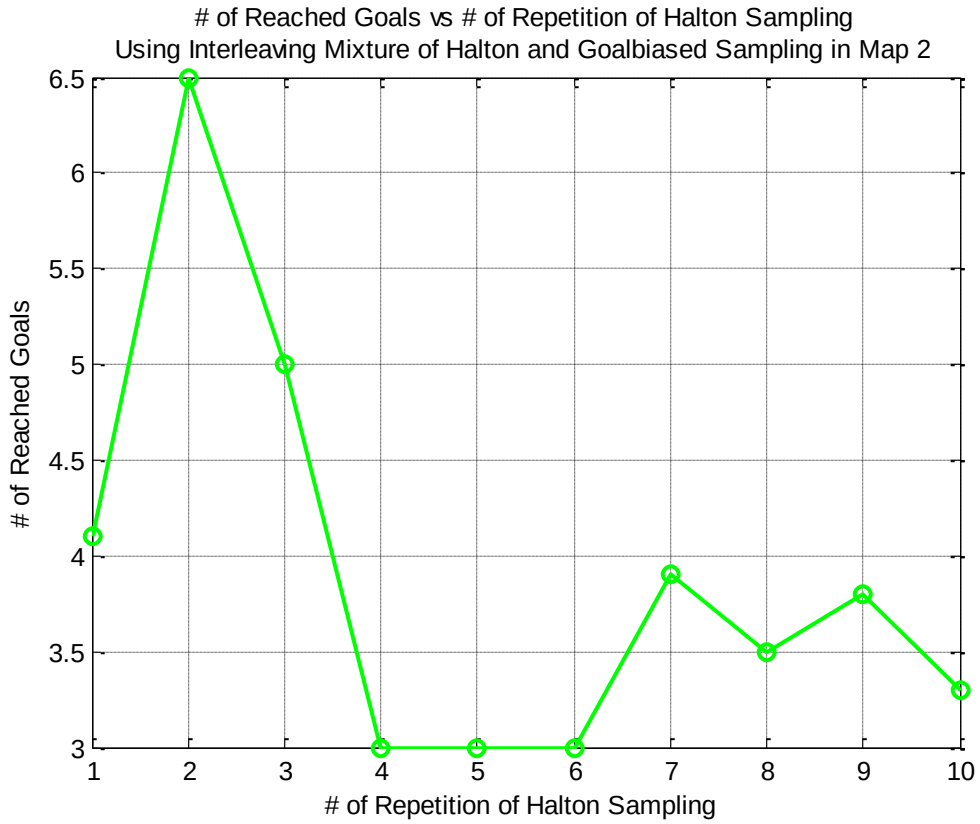

Figure 41: The Number of Reached Goals vs. the Number of Repetition of Halton Sampling Using the Interleaving Mixture Sampling in Map 2

For this case, the maximum standard deviation is also taken as 0.1 and we look for the performance of the interleaving mixture about the number of repetition of Halton sampling. It is seen that low repetition of Halton is the best choice for this sampling technique as in the percent mixture, and the result in figure 41 shows that the optimum value for the number of repetition of Halton sampling is 2 in the interleaving mixture sampling technique. As a result of this experiment, the optimum values of sampling parameters can be summarized as table 4.

Table 4: The Optimum Values of Sampling Parameters

	The Maximum Standard Deviation	The Percentage of Halton Sampling	The Number of Repetition of Halton Sampling
Goalbiased Sampling without Obstacle	0	-	-
Goalbiased Sampling with Obstacle	0.1	-	-
The Percent Mixture	0.1	0.25	-
The Interleaving Mixture	0.1	-	2

And in the following experiments, we use the values in table 4 for the parameters of these sampling techniques.

5.4.2 The Performance of Our Model

This experiment is carried out to define the performance of our RRT based motion planner model for different sampling techniques and baselines, and find the best performance for the cases of multiple camera inspection that we have considered in this study. The values of parameters that are used in this experiment are given as follows:

- Number of review cameras: 1
- Maximum allowable review module x-axis acceleration: 8 m/s^2
- Maximum allowable review module x-axis velocity: 0.5 m/s
- Material y-axis constant velocity: 0.220 m/s
- The maximum standard deviation: 0, 0.1
- The percentage of Halton sampling: 0.25
- The number of repetition of Halton sampling: 2
- The maximum number of trees: infinite
- The maximum number of branches: infinite
- The maximum number of iterations: 2000

- The values of other parameters are defined in table 3

In this experiment, we also use map 1 and 2 as shown in figure 36 and another environment with more complex obstacle, map 3 that will be shown later. Firstly we look for the number of reached goals in the environment without obstacle, map 1 for different sampling techniques.

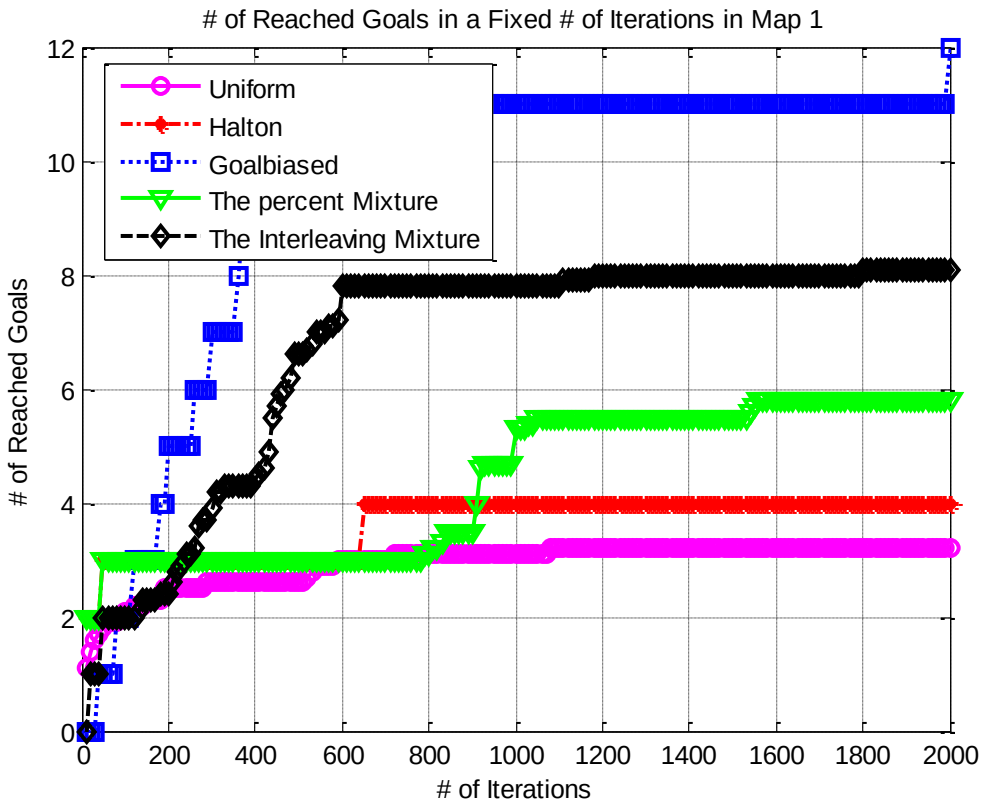

Figure 42: The Number of Reached Goals vs. the Number of Iterations in Map 1

As it is seen in this result, Goalbiased sampling is the best choice for the environment without obstacle, since it biases towards goals and increases the rate of RRT convergence, and when we compare the performance of Goalbiased sampling with the deterministic method as told in section 5.2.2, we see that the deterministic methods reaches 12 goals in map1 for same parameters and this shows that our model for this case provides 92% performance of the deterministic method. And when we use map 2 to include obstacles, we get the result in figure 43.

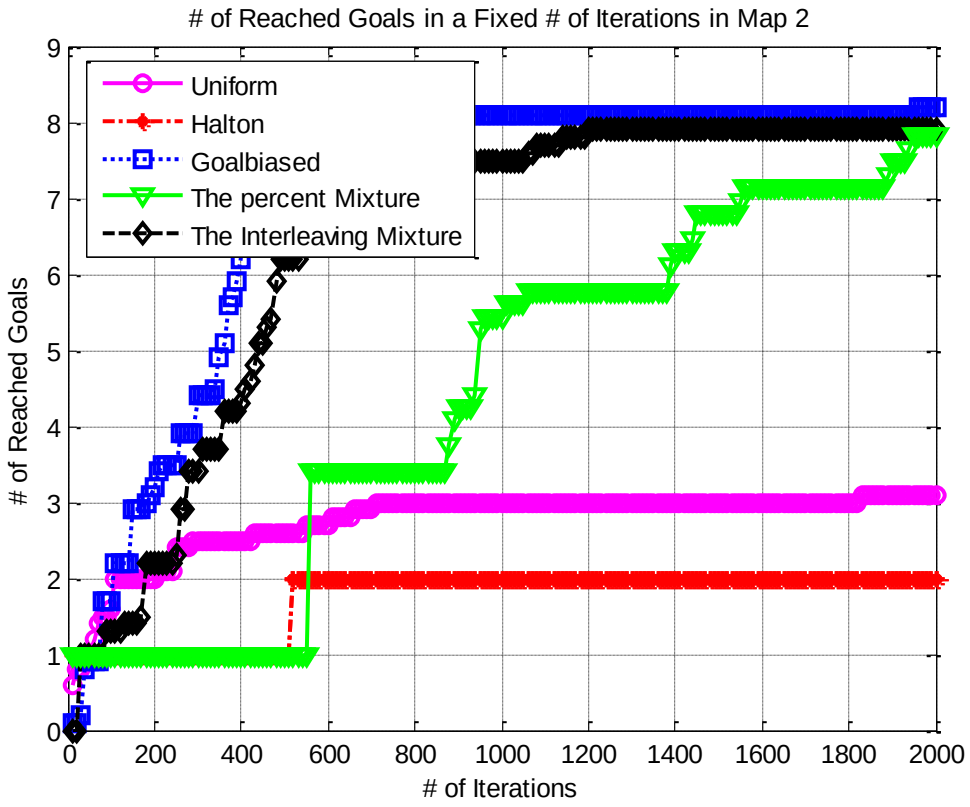

Figure 43: The Number of Reached Goals vs. the Number of Iterations in Map 2

In this case, the standard deviation of Goalbiased sampling is increased to 0.1 as defined in table 4 to avoid getting stuck in obstacle region. And once more we get better performance while using Goalbiased sampling. The percent and interleaving mixture techniques can reach this performance at high number of iterations, because Halton sampling that are used in the first part of these techniques slows down to reach goals while generating samples to cover whole state space. From this result, it can be believed that using only Goalbiased sampling is better than using other techniques, but when we include more complex obstacles as shown in figure 44, Goalbiased sampling cannot be successful to escape from getting stuck, although its standard deviation is increased.

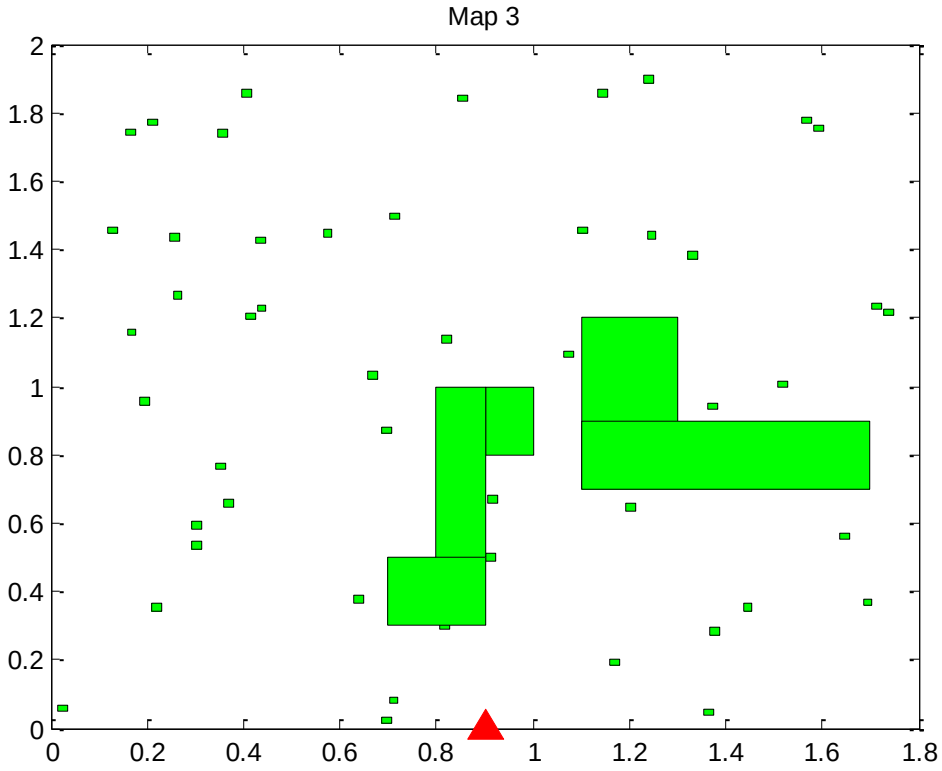

Figure 44: Another Map for the Experiment 2: It includes 50 defects with an obstacle.

In figure 44, map 3 is represented that includes 50 defects and two obstacles in the environment. These obstacles form a narrow passage and it difficult to grow RRT through this passage. Goalbiased sampling cannot handle this and so we need to use the mixture techniques. This is shown in figure 45 that is the analysis of the performance of our model for different sampling techniques in map 3.

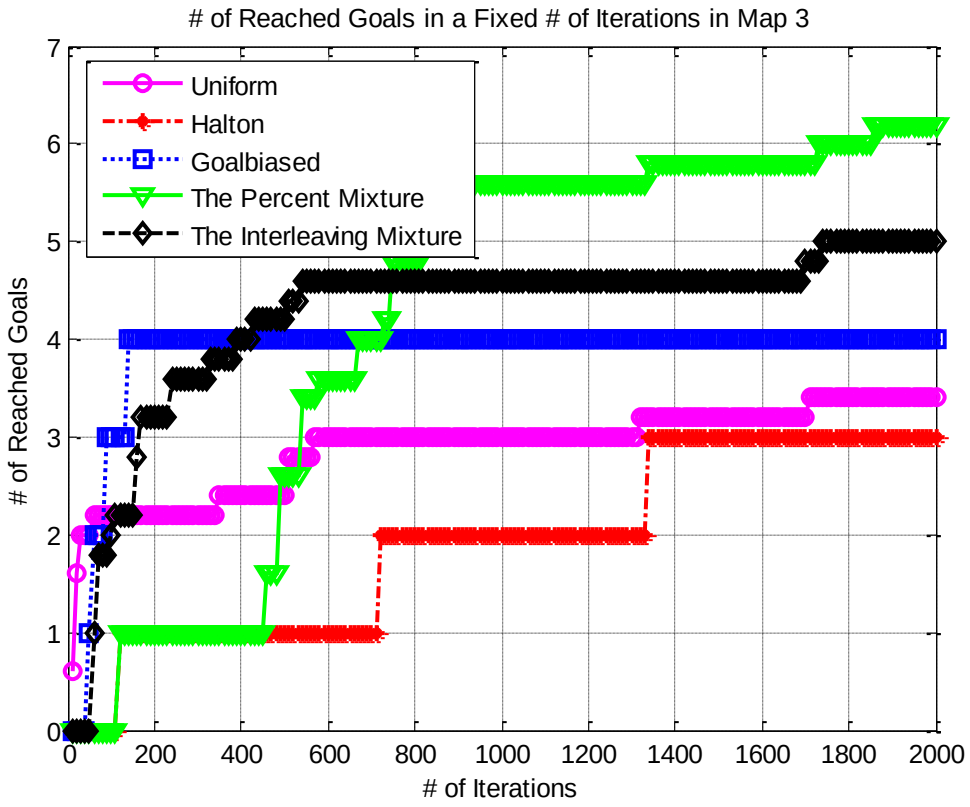

Figure 45: The Number of Reached Goals vs. the Number of Iterations in Map 3.

In this result, Goalbiased sampling with a middle deviation is saturated to a specified number of reached goals because of getting stuck in obstacle, but the mixture techniques, especially the percent mixture give the best performance for this case and this shows that these mixture techniques are more suitable for the environment with obstacles than using only Goalbiased Sampling.

5.4.3 The Performance Dependence on Critical Parameters

In this experiment, we try to define the performance dependence of our model on critical parameters that are the maximum x-axis velocity, the maximum x-axis acceleration, y-axis constant velocity and the number of review cameras. To see this dependence, we divide this experiment into four parts and in each part; the performance of the system about one of these parameters is analyzed by assigning constant values to

others. And the values of the common parameters that are used in four parts of this experiment are given below:

- The maximum standard deviation: 0
- The maximum number of trees: infinite
- The maximum number of branches: infinite
- The maximum number of iterations: 500, 2000
- The values of other parameters are defined in table 3

For every part of this experiment, we use only Goalbiased sampling in map 1 that does not include obstacle, and in the first part of the experiment, we examine the maximum x-axis velocity by using the values of the critical parameters as given below:

- Number of review cameras: 1
- Maximum allowable review module x-axis acceleration: 8 m/s^2
- Maximum allowable review module x-axis velocity: 0.1-1 m/s
- Material y-axis constant velocity: 0.220 m/s

We determine the number of reached goals in 500 iterations by using these values of the critical parameters and get the performance plot about the maximum x-axis velocity as shown in figure 46.

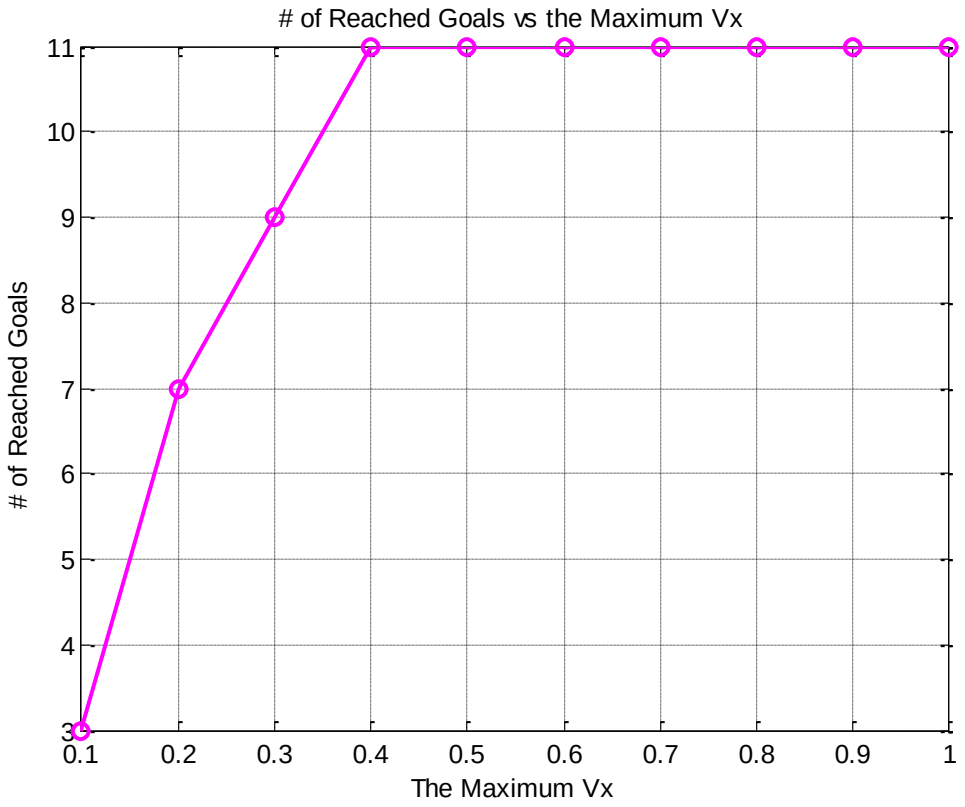

Figure 46: The Number of Reached Goals vs. the Maximum x-axis Velocity.

This result shows that in low velocities, the performance becomes worse as expected, but in high velocities, it increases and becomes saturated at some point. This saturation happens because of that the x-axis acceleration is still limited to a specified value and the VELOCITY_CHECK function in the algorithm does not permit to reach high velocities, so increasing the x-axis velocity to high values does not affect the performance of our model for this case. Then we examine the effect of the maximum acceleration on the performance of our model by using the values of the critical parameters as follows:

- Number of review cameras: 1
- Maximum allowable review module x-axis acceleration: 1-15 m/s^2
- Maximum allowable review module x-axis velocity: 2.0 m/s
- Material y-axis constant velocity: 0.220 m/s

And the result of this experiment is given in figure 47.

Figure 47: The Number of Reached Goals vs. the Maximum x-axis Acceleration.

This shows that cameras can reach more defects when the maximum lateral acceleration is increased since difficult motions can be easily handled. Another part of this experiment is related with the y-axis velocity and the values of the critical parameters are given below:

- Number of review cameras: 1
- Maximum allowable review module x-axis acceleration: 8 m/s^2
- Maximum allowable review module x-axis velocity: 0.5 m/s
- Material y-axis constant velocity: 0.01-0.5 m/s

According to these values, the performance plot about the y-axis velocity is obtained as shown in figure 48.

Figure 48: The Number of Reached Goals vs. the y-axis Velocity.

As expected, we can reach more goals in the environment by decreasing the y-axis velocity. And this shows that the performance of our model can be improved by varying y-axis velocity. This is considered in extended case in which the cameras can move independently with variable y-axis velocities as told in chapter 3. We have not implemented this case in our algorithm, but it is applicable by only changing the configuration definition of the cameras and the region of inevitable collision since it has more DOFs, and we consider this case as a future work. The final part of this experiment is about examining the performance about the number of review cameras of the system with the values of critical parameters below:

- Number of review cameras: 1-3
- Maximum allowable review module x-axis acceleration: 8 m/s^2
- Maximum allowable review module x-axis velocity: 0.5 m/s
- Material y-axis constant velocity: 0.220 m/s

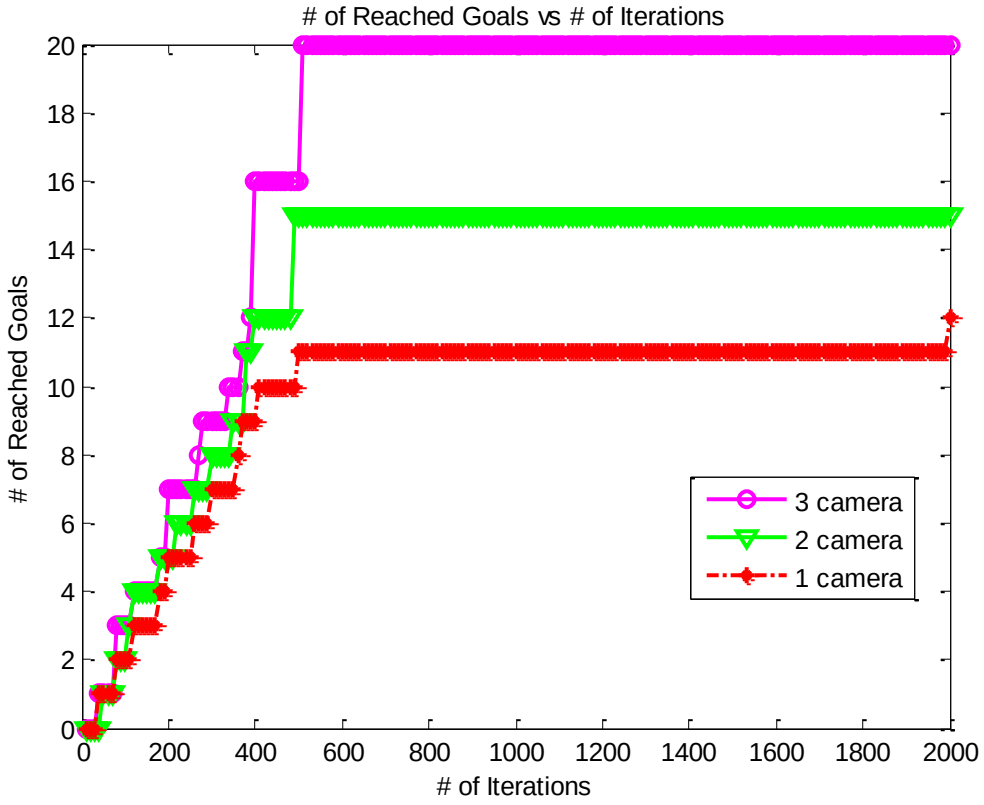

Figure 49: The Number of Reached Goals vs. the Number of Iterations for Different Number of Cameras.

In this result, we get the best performance while using the largest number of camera as expected, and since this experiment is performed in the environment without obstacle, its result can be compared with deterministic method for the same case. And this comparison is given in table 5.

Table 5: The Comparison between Our Model and Deterministic Method

# of Cameras	**# of Reached Goals**	
	Our model	**Deterministic Method**
1	11	12
2	15	22
3	20	32

As shown in these results, the performance of our model is not good as the deterministic method in multiple camera cases; this is because of increase on the collision possibility between cameras. As the number of cameras is increased, the generated samples have high probability to be in collision regions between cameras and so many samples are removed in validation part of the RRT algorithm and cannot be used, and if we increase the number of generated samples, we can reach the results of multiple camera in deterministic method, but the most important advantage of our model is the ability to handle obstacles and also dynamic obstacles that the deterministic method cannot provide.

5.4.4 The Time Performance of Our Model

In this experiment, we attempt to improve the computation time of our model by using two parameters that are the maximum number of trees and branches. In our algorithm, new trees are created at every goal detection and each tree grows when it is the nearest one to generated sample, so if we increase the number of defects in the environment and iteration time, the computation time rises because of many trees and their branches in the environment. So we consider a limit to the number of trees and branches to reach the same number of goals in lower computation time. And we define the values of parameters as follows.

- Number of review cameras: 1
- Maximum allowable review module x-axis acceleration: 8 m/s^2
- Maximum allowable review module x-axis velocity: 0.5 m/s
- Material y-axis constant velocity: 0.220 m/s
- The maximum standard deviation: 0
- The maximum number of trees: 30, 50, 100, 200, 500
- The maximum number of branches: 30, 50, 100, 500, 1000
- The maximum number of iterations: 1000
- The values of other parameters are defined in table 3

We use only Goalbiased Sampling in map 1 for this experiment, and search for the minimum time to reach fixed number of goals (10 for this experiment) by putting limits on the number of trees and branches. Firstly we determine the successful cases in which 10 goals are reached as shown in figure 50.

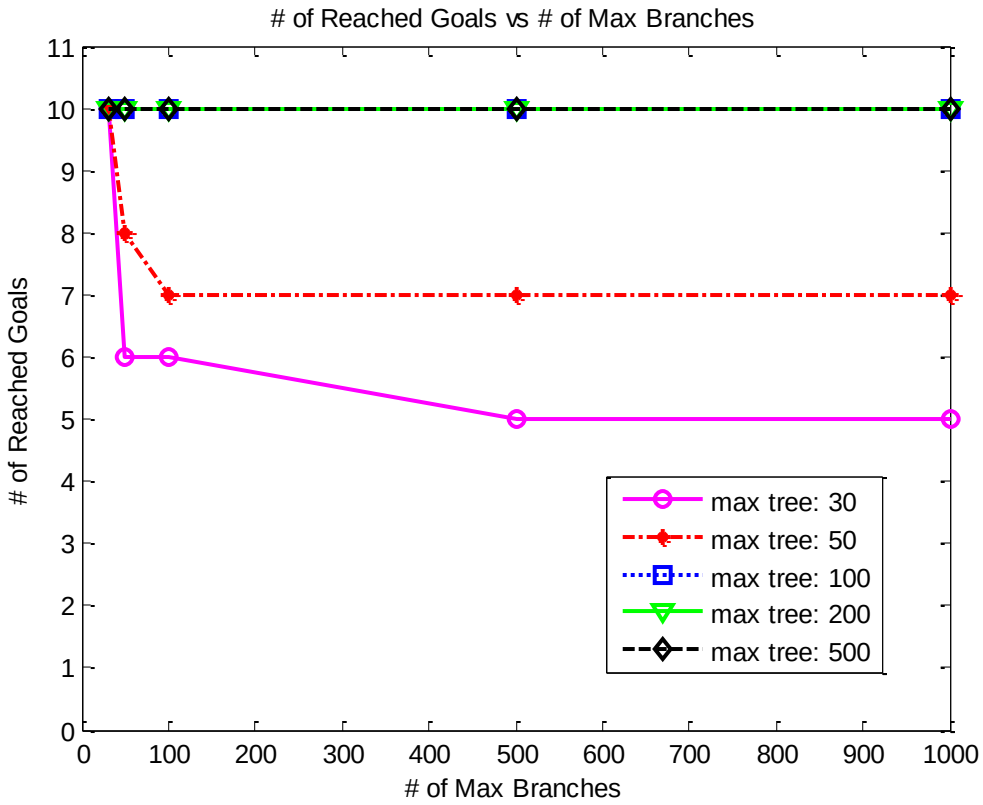

Figure 50: The Number of Reached Goals vs. the Maximum Number of Branches Corresponding to Different Maximum Number of Trees.

In this plot, the number of reached goals is determined by changing the maximum number of branches for 5 different cases in which different limits are put on the number of trees. As it is seen, at low limits on the number of trees, desired number of goals cannot be reached. But if we use sufficient limit on the number trees, we do not change the performance of our model and then we examine the computation time for these cases in the following figure.

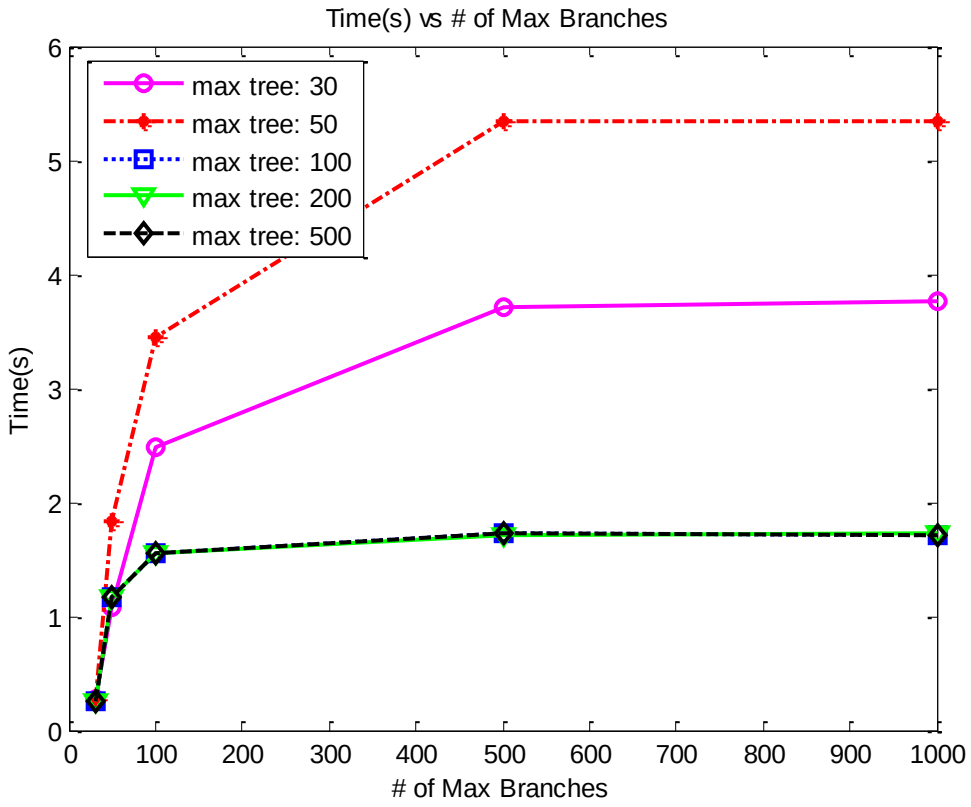

Figure 51: Time vs. the Maximum Number of Branches Corresponding to Different Maximum Number of Trees.

When we consider the successful cases of this figure in which the maximum number of trees are 100, 200 and 500, we see that as the maximum number of branches is decreased, and the computation time to reach 10 goals decreases. Also the maximum number of trees does not change the computation time for the successful cases. As a result of these plots, the computation time of our model can be improved without affecting the solution by only putting a low limit on the number of branches for each tree.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

When we search the available techniques to solve the kinodynamic motion planning problem of the multiple camera industrial inspection, we see three important techniques that are RPP, PRM and RRT. As we have mentioned before, the RPP and PRM algorithms need complex computations when we include the differential constraints and obstacles, and the RRT algorithm handles this difficulty. This is why we have used the RRT algorithm in our approach.

In this study, we firstly described our environment and formulated the problem in four cases that were single camera without obstacle, with obstacle, and multiple camera without obstacle and with obstacle as told in chapter 3. We defined the main algorithm in chapter 4 by adapting the RRT algorithm to our problem. Using this algorithm we implemented a RRT based motion planner model that aimed to reach as much defect POIs as possible over inspection area while satisfying the constraints of our problem and avoiding collision with obstacles, and made some experiments on this planner to analyze its performance. And we saw that the sampling techniques were very important in the performance of the RRT based motion planning. Goalbiased sampling had better performance than others in environments without obstacles, but when we added some obstacles, there came out a problem that was the possibility of getting stuck in obstacle region. We attempted to solve this problem by increasing the standard deviation, but in the environment with complex obstacles that include narrow passages, only using Goalbiased sampling was not sufficient to have good results. And this motivated us to

define new sampling techniques for these cases that were the percent and interleaving mixture of Halton and Goalbiased samplings, and the results of these techniques showed us that they were successful in the environment with obstacles. Then we examined the effect of the critical parameters on the performance of our model that were the maximum x-axis velocity, the maximum x-axis acceleration, and y-axis constant velocity as well as the number of review cameras, and we got the expected results. When we compared our results with the results of deterministic method as described in section 5.2.2, we saw that our model did not outperform the deterministic method for no obstacle case; this was because of increase on the possibility of collision between cameras. But our model was able to handle inspection area with obstacle as well. Also we examined the time performance of our model and improved the computation time by putting low limits on the number of branches for each tree.

As a future work, the sampling techniques in the RRT algorithm can be developed to provide better performance. The sample generation is important task for an RRT based planner since RRT is a sampling based algorithm and we should give importance to this topic. Our algorithm can be improved by using efficient nearest neighbor search and choosing efficient metrics that are important issues of the RRT algorithm. And the collision avoidance algorithm can be developed by using data structures, and local information of old branches and explored regions. Also we can attempt to implement the extended case of multiple camera inspection that was told in chapter 3 and this case is applicable to our algorithm by only changing the configuration definition and collision avoidance part. We consider that this case has potential to provide better results because of involving more DOFs and so we can show that our problem is applicable to general problems in robotics.

REFERENCES

[1] Ferguson, D., Stentz, A. "Any time RRTs". IEEE international Conference on Intelligent Robots and Systems, pp. 5369-5375. October, 2006.

[2] Ferguson, D., Kalra, N., Stentz, A. "Replanning with RRTs". IEEE International Conference on Robotics and Automation, pp. 1243-1248. May, 2006.

[3] Ege, E., Saranlı A. "A New Approach to Increase Performance of RRT in Mobile Robotics". IEEE. METU. Ankara. 2006.

[4] Rodriguez, S., Tang, X., Lien, J., Amato, N. "An Obstacle Based RRT". IEEE International Conference on Robotics and Automation, pp. 895-900. May 2006.

[5] Atramentov, A., LaValle, S. M "Efficient Nearest Neighbor Searching for Motion Planning". IEEE International Conference on Robotics and Automation, pp. 632-637. May, 2002.

[6] Yershove, A., LaValle, S. M. "Improving Motion Planning Algorithms by Efficient Nearest Neighbor Searching". IEEE Transactions on Robotics. November, 2002.

[7] Lindemann, S. R., LaValle, S. M. "Steps toward Derandomizing RRTs". 4th International Workshop on Robot Motion and Control. June, 2004

[8] Kuffner, J. J., LaValle, S. M. "RRT-Connect: An Efficient Approach to Single-Query Path Planning". IEEE International Conference on Robotics and Automation. 2000.

[9] Cheng, P., LaValle, S. M. "Reducing Metric Sensitivity in Randomized Trajectory Design". Computer Science Department, Iowa State University.

[10] LaValle, S. M. “Rapidly-Exploring Random Trees: A New Tool for Path Planning”. Report No. TR 98-11, Computer Science Department, Iowa State University. 1998.

[11] LaValle, S. M., Kuffner., J. J. “Rapidly-Exploring Random Trees: Progress and Prospects”. In Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR), pp. 45-59. 2000.

[12] LaValle, S. M., and Kuffner, J. J. “Randomized Kinodynamic Planning”. The International Journal of Robotics Research, pp. 378-400. May, 2001.

[13] Kalisiak, M. “Kinodynamic Motion Planning with Viability Models”. A Research & Thesis Proposal. Computer Science Department, University of Toronto. 2007.

[14] LaValle, S. M. ”Planning Algorithms”. University of Illinois. Cambridge University Press. New York. 2006.

[15] Donald, B., Xavier, P., Canny, J., Reif, J. “Kinodynamic Motion Planning”. Journal of the ACM, vol. 40, pp. 1048-1066. 1993.

[16] Weiss, A., Saranlı A., Ghelman, E., Baldwin, D. US patent, no: US 7,137,309 B2. November, 2006