Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Quellenverzeichnis angegebenen Quellen benutzt habe. Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.
Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Aachen, November 2019

Kandarp Gandhi

Acknowledgement

First, I would like to thank Mr. Sebastian Müller and Dr. Stefan Kesselheim from StreetScooter GmbH, for their invaluable help and guidance during the course of this thesis. Furthermore, I am thankful to Mr. Heiko Engemann and Prof. Dr.-Ing. Stephan Kallweit for their support and constant motivation.

I am grateful to Mr. Tobias Augspurger for giving me the opportunity and excellent resources to perform this work. I would like to thank Dr. Bugra Turan for his critical inputs concerning machine learning. Lastly, I would like to thank all my colleagues from StreetScooter GmbH for their enormous support.

Abstract

Autonomously manoeuvring a car-like robot in an unstructured environment is a complex problem and requires technologies from diverse areas e.g. perception and navigation, to solve. Finding an optimal path which the robot can follow to reach its destination while avoiding the obstacles is a crucial subproblem and an active area of research. Sample-based motion planning (SBMP) algorithms have shown significant results in many real-world applications such as navigation for differentially constrained robots, robotic manipulation, and even for studying protein folding pathways. SBMP generally draw random samples from a uniform distribution to cover the whole search space. However, for a particular problem, only a small proportion of the samples would contribute to the optimal path. To accelerate the planning process, it is thus desirable to develop non-uniform sampling strategies that favour sampling in those regions where an optimal solution might lie. In this work, problem-specific sampling methods are investigated and developed using first by a heuristic-based approach, which finds an optimal subspace of search space for problem-specific sampling. Later, a learned sampling method is used to directly generate critical samples which use a conditional variational autoencoder model to learn from the previous path planning solution.

Keywords: *Motion planning, Nonholonomic Robot, Sample-based Motion Planning, Sampling Methods*

Contents

1	**Introduction**	**6**
1.1	Problem Formulation	7
1.2	Objective and Scope	7
1.3	Thesis Outline	8
2	**Technical Background**	**10**
2.1	Problem Definition	10
2.2	Vehicle Model	11
2.3	Path Planning Algorithms	12
2.4	Sampling-Based Motion Planning (SBMP)	14
2.5	Sampling-Based Motion Planning for Nonholonomic Systems	18
2.6	Problem-Specific Sampling Methods	19
2.6.1	Heuristics for Biased Sampling	21
2.6.2	Informed Sampling	23
2.6.3	Adaptive Sampling	24
2.6.4	Learned Sampling	24
3	**Space Exploration Biased Sampling**	**27**
3.1	Free Space Bubble	27
3.2	Space Exploration	28
3.3	Orientation Aware Space Exploration	29
3.4	Orientation Aware Space Exploration in Complex Environments	32
3.5	Modified Orientation-Aware Space Exploration	33
3.6	Probability Distribution	35
3.7	Number of Samples	38
4	**Numerical Evaluation of Space Exploration Biased Sampling**	**39**
4.1	Test Settings	39
4.2	Collision Detection	41

4.3 Implementation of Sampling-based Motion Planning for a Car-like System 42
4.4 Numerical evaluation of Orientation-Aware Space Exploration 43
4.5 Experimental Results of Space Exploration Biased Sampling 44
4.5.1 Failure Rate Evaluation 46
4.5.2 Path Cost Evaluation 50
4.5.3 Computation Time Evaluation 53
4.6 Conclusion 55
4.7 Future directions 55

5 Learned Sampling 57
5.1 Density Estimation 58
5.2 Conditional Variational Autoencoders 59
5.3 CVAE on a toy problem 61

6 Experimental Results of Learned Sampling 64
6.1 Training data 65
6.2 CVAE Architecture 65
6.3 Evaluation results 67
6.4 Conclusion 75
6.5 Future directions 76

Bibliography 77

List of Figures 88

List of Tables 89

Appendix

A. Orientation-Aware Space Exploration pseudocode 90

B. Space Exploration results for complex problems 91

C. Problem specific samples using learned sampling 96

1. Introduction

Society and industry have shown growing interest in autonomous driving in recent years. The aim of autonomous driving is to increase safety while also making the vehicle more efficient. To manoeuvre a truck in a tight environment such as DHL Paketzentrum (Figure 1-1) needs abundant attention and skill. To boost driving safety, there is a possibility to automate these repetitive tasks, that is an active area of research.

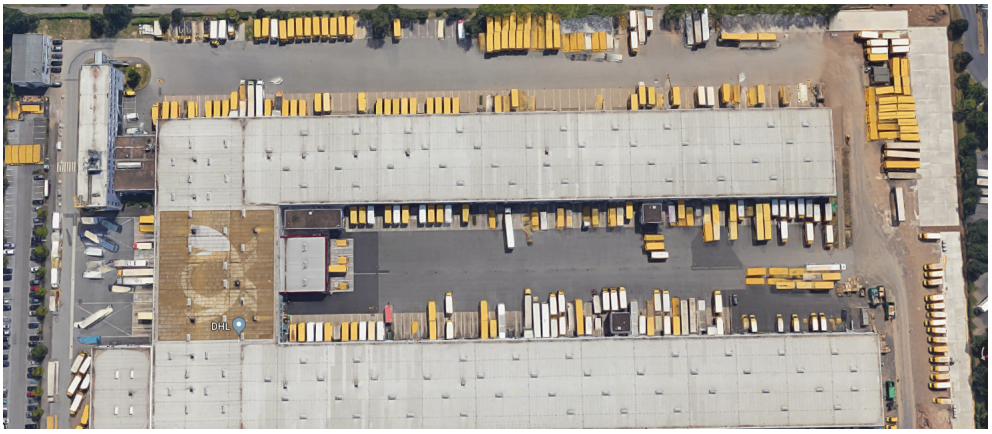


Figure 1-1: Bird view of DHL Paketzentrum at Köln-West. Image is taken from Google Maps. [1]

To autonomously navigate through a complex environment, a vehicle has to perceive its environment, determine its own position in the environment (Localization) and plan a trajectory which the robot can execute to reach the destination while avoiding obstacles in the surrounding environment. The problem of finding a safe, feasible and efficient trajectory which the robot can utilize to reach the target position is widely known as a motion planning problem. The area of motion planning is crucial for the autonomous vehicle and is the research topic of this thesis.

1.1 Problem Formulation

A non-holonomic system has constraints on its velocity and requires differential equations to represent the kinematics (13.1.2 [2]). A car-like vehicle is a nonholonomic system because it is unable to travel sideways directly and its maximum path curvature is limited by the steering angle. These kinematics constraints make manoeuvring a car-like vehicle in a complex environment challenging.

Solving a motion planning problem in a large and unstructured arena while considering all vehicle constraints (kinematic and dynamic) is non-trivial. Moreover, a small change in the environment requires a recalculation of the solution. Therefore, to make the problem tractable a two-level motion planning model has been proposed, which utilizes a global path planner and local trajectory planner [3]. A global path planner would compute a collision-free geometric path, based on a known environmental map and a local trajectory planner would utilize the global path to compute the trajectory for a short term horizon while considering the vehicle constraints and avoiding new obstacles on a local level. Fast computation of the global path is essential in an unstructured parking environment. (e.g. A path planner (Hybrid A* [4]) used in the DARPA Urban Challenge computes a path in $50 - 300\,ms$ for environment size of $160\,m \times 160\,m$.) This work focuses on developing a fast and efficient global path planner.

To date, the sample-based motion planning (SBMP) algorithms are the state-of-the-art for solving non-holonomic motion planning problems. Rather than discretizing the whole search space in a grid-based manner, SBMP algorithms represent it as discrete samples and build graph/tree structure using these samples to find a path. However, for a particular problem, only a small proportion of the samples would contribute to the optimal path. Therefore, problem-specific sampling strategies that favor sampling in regions where an optimal solution might lie can improve the performance of SBMP algorithms, which is the main focus of this thesis.

1.2 Objective and Scope

Heuristic-based algorithms are widely used in practice because they can solve the problem faster and more efficiently. In [5], the author suggested a heuristic-based method for computing a subspace to reduce the search effort of motion planning algorithm, which was utilized for problem-specific sampling method in [6]. However, it required further

development to utilize in a complex environment for problem-specific sampling, which is a part of this thesis.

In recent years, machine learning algorithms have shown excellent results on problems from various fields, which motives to utilize the advancement of machine learning approach for problem-specific sampling method. Rather than developing an end-to-end model for path planning, a hybrid model is utilized in this work. It uses classical SBMP algorithm for computing the solution over samples generated by a machine learning algorithm.

As mentioned, the main objective of this thesis is defined as:

- Implement a framework which can solve the path planning problem for a car-like robot using sample-based motion planning (SBMP) approach by utilizing an open-source C++ library for SBMP, Open Motion Planning Library (OMPL) [7].
- Develop a problem-specific sampling method by utilizing a heuristic-based approach for a car-like vehicle. Moreover, evaluate its performance in a large and unstructured environment similar to a Paketzentrum and compare it with the uniform sampling method.
- Develop a problem-specific sampling method by utilizing a learning-based approach. Furthermore, evaluate its performance for a car-like vehicle.

1.3 Thesis Outline

This thesis is structured in six chapters. Chapter 1 gives an introduction to the research problem and outlines the structure of this thesis.

Chapter 2 starts with additional details about the path planning problem and discusses available approaches for solving the path planning problem. After selecting an algorithm from the family of SBMP algorithms, it introduces different problem-specific sampling methods. From which, space exploration biased sampling method and learned sampling method for further evaluation have been considered.

Chapter 3 begins with introducing the concept of free space bubble as proposed in [8] and explaining a space exploration method as proposed in [9], which is the base algorithm for this work. Furthermore, it discusses the weakness of the presented space exploration method and suggest improvements to the existing algorithm. Lastly, it introduces sampling methods which utilize information gathered from the space exploration

method.

Chapter 4 starts with providing information about test settings and implementations. Subsequently, it illustrates the evaluation methods used for benchmarking space exploration biased sampling methods. Furthermore, it examines the results of different variances of space exploration biased sampling method and compares it with uniform sampling method. Lastly, it concludes the results and provides recommendations for future work in the direction of space exploration biased sampling.

Chapter 5 starts with explaining the need for a more generic algorithm for problem-specific sampling and introduces the concept of the data-driven learned sampling method. It gives a brief introduction of latent variable models and Conditional Variational Autoencoder (CVAE) model for learned sampling. Furthermore, it presents the result of CVAE on a toy problem.

Chapter 6 evaluates the learned sampling method for a car-like vehicle. It starts with describing a simplification of the path planning problem for the primary development of learned sampling method. Furthermore, it presents the limitation of the learned sampling model presented in [10] and suggests enhancements for the CVAE model. Moreover, it compares the results of the learned sampling method with the uniform sampling method. Lastly, it concludes the results of learned sampling method and provides direction for future work.

2. Technical Background

2.1 Problem Definition

In this work, the goal is to develop a global path planner which would guide the local trajectory planner to reach the target destination. It is a common approach to consider the robot as a holonomic system on a global scale and compute a geometric collision-free path, and then use a local planner to correct for the model error [11].

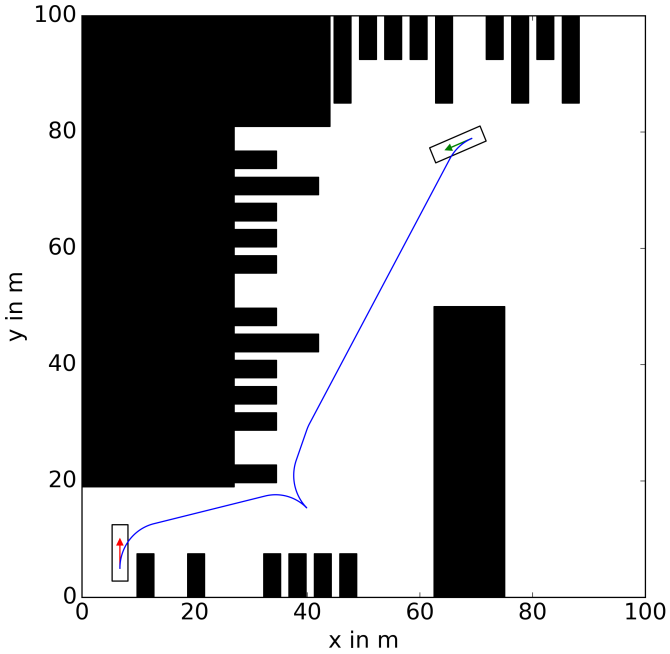


Figure 2-1: The global path (blue) for a path planning problem of reverse parking at the end of a narrow region defined by a start (green arrow), goal pose (red arrow) and obstacles represented by the black region.

Though reverse parking at the end of a narrow passage (Figure 2-1) requires switching the direction beforehand and to accomplish this only with a local planner requires larger time horizon, which results in high computational cost. Therefore consideration of more

reliable vehicle model on the global scale became necessary.

The objective of this work can be reformulated as to develop a global path planner which satisfies the kinematic constraints of the vehicle. Therefore, the search space for a solution would be the configuration space. A configuration q is a specification of the position of every point on a robot body and configuration space C is the set of all possible robot configurations [2].

The problem formulation in this work follows the problem formulation of [12]. Let $C_{obs} \in C$ be the obstacle region, such that C/C_{obs} is an open set. The obstacle-free space is defined as $C_{free} := C/C_{obs}$. An initial configuration q_{init} and a goal configuration q_{goal} are elements of C_{free}.

Given an initial configuration q_{init}, a goal configuration q_{goal} and an obstacle region C_{obs}, the path planning problem is to find a collision-free path $\sigma : [0, 1] \rightarrow C_{free}$ that starts from the initial configuration $\sigma(0) = q_{init}$ and reaches the goal configuration $\sigma(1) = q_{goal}$. Let $c : \sum_{free} \rightarrow \mathcal{R}_{\geq 0}$ be a cost functional that maps each collision-free path to a non-negative cost. The optimal path planning problem is to find a collision-free path $\sigma^* : [0, 1] \rightarrow C_{free}$ that solves the path planning problem, and minimizes the cost functional $c(\cdot)$. The optimal path planning problem for a non-holonomic system is to find a collision-free path that solves the optimal path planning problem while satisfying kinematic constraints. Next, we will discuss the vehicle model used for this work.

2.2 Vehicle Model

The car-like vehicle can be imagined as a rigid body moving in a plane, therefore the environment can be modeled as a plane, where obstacles represent the forbidden region. With this simplification, the vehicle would have three degrees of freedom and its configuration can be written as $q = (x, y, \theta)$, where (x, y) represents the position of the rear axle of the car and θ represents the heading of the car, as shown in the Figure 2-2.

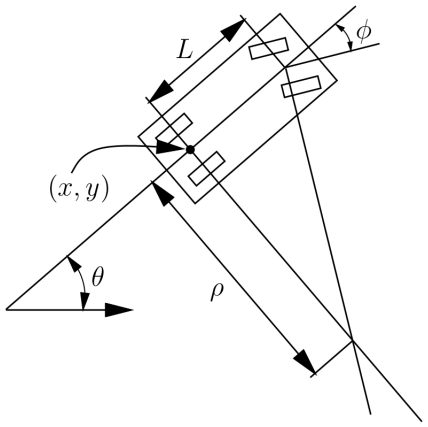


Figure 2-2: The car-like vehicle A: $R = (x, y)$ is the reference point and θ is the main orientation. ϕ is the steering angle and L is the wheelbase. [2]

To consider the non-holonomic constraints, a simple yet useful model of a car that does not consider dynamics is the bicycle model. The bicycle model has two control inputs, forward velocity and steering angle (ϕ) which is limited by maximum steering angle ($|\phi| \leq \phi_{max}$). Minimum turning radius (ρ_{min}) of a bicycle model can be defined by wheelbase (L) and the maximum steering angle (ϕ_{max}) as Equation 2.1 and inverse of the minimum turning radius (ρ_{min}) is represented by maximum curvature (κ_{max}), ($\kappa_{max} = 1/\rho_{min}$).

$$\rho_{min} = \frac{L}{\tan(\phi_{max})} \tag{2.1}$$

To further simplify the model, allowable forward velocities (u_v) and steering angles (u_θ) are restricted to a finite set, $u_v = \{-1, 0, 1\}$ & $u_\theta = \{-\phi_{max}, 0, \phi_{max}\}$, which is called as the Reeds-Shepp car model [13]. Although the Reeds-Shepp car model is very simplified, it would be sufficient to represent the nonholonomic constraints on the global scale.

2.3 Path Planning Algorithms

A fairly straightforward approach for solving the path planning problem for a car-like robot would be to first compute a geometric collision-free path while ignoring the non-holonomic constraints and then consecutively modify it to satisfy the constraints. (see e.g. [14] and [15]). However, it has several drawbacks, e.g. computation of a sub-optimal path due to the decoupling approach and fail to modify a geometric path to satisfy the

nonholonomic constraints. This motivates to develop an efficient algorithm that directly solves nonholonomic planning problems. Next, we will get an overview of algorithms that are capable of solving the nonholonomic planning problem directly.

The motion planning methods can be broadly categorized into two groups, planning by construction and planning by modification, as shown in Figure 2-3.

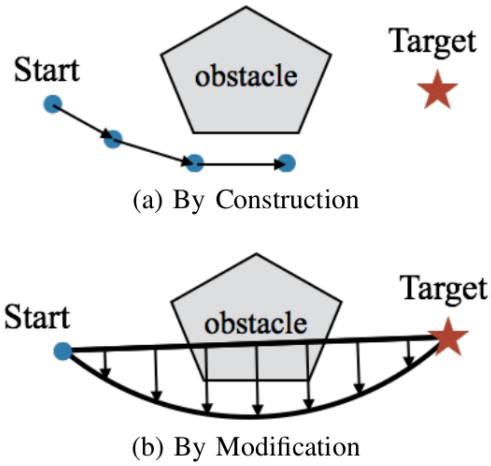


Figure 2-3: Typical Motion Planning Methods as proposed in [16]

Planning by construction method extends a trajectory by adding a new state to it until the target state is reached, e.g. search-based methods and sampling-based methods. Planning by modification method perturbs an existing trajectory such that the desired property is obtained, e.g. Optimization-based motion planning methods, where the perturbation can be interpreted as gradient descent.

Planning by modification method can require many iterations to find a feasible solution and can get stuck in local optimum if it started with a bad initial trajectory while Planning by construction method can quickly find a feasible and near-optimal solution but the trajectory is not smooth. Therefore, it is common practice is to solve the planning problem first by a construction based method and then smooth it using a modification based method. Next, we will review construction based planning algorithms.

The search-based planning algorithms construct a graph by discretizing the environment and search it with any graph-search algorithm, e.g. Dijkstra and A*. To extend the search-based planners to non-holonomic systems, a graph is generated using kinematically feasible motion primitives, as proposed in Hybrid A* [4] and State lattice [17]. The motion primitives and robot footprint for these paths are computed offline and stored, which is

utilized for online exploration-based graph search, illustrated in Figure 2-4.

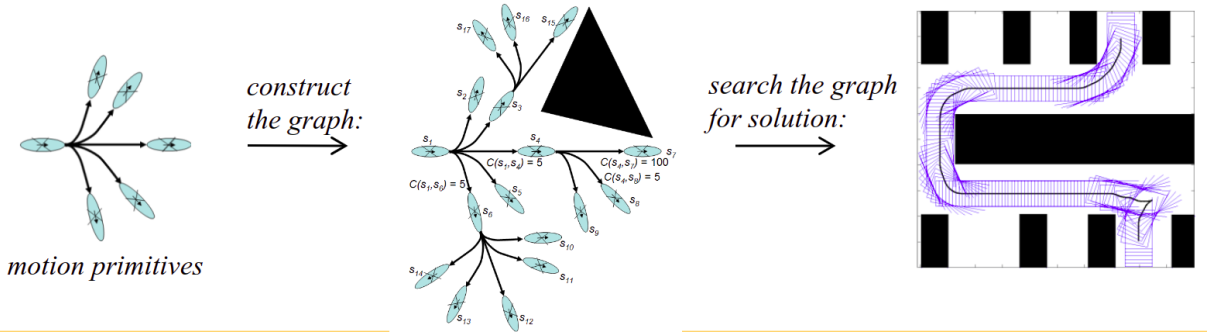


Figure 2-4: Path planning for a car-like robot using state lattice algorithm. [18]

The main drawback is that the solution path can only reach a discretized approximation of the expected target state and the algorithm may return failure even though there is a solution, due to the coarse discretization of the motion primitives. Fine discretization of the primitive motion is not a valid option as it will polynomially increase the computational complexity.

2.4 Sampling-Based Motion Planning (SBMP)

SBMP algorithms avoid the discretization of the configuration space (C space) in a grid-based manner by representing free C space as discrete samples, which are drawn generally from a uniform distribution (many times with a slight goal bias) to cover the whole C space. SBMP algorithms utilize a collision detection module to check the validity of a sample, which acts as a ”black box” to separate the motion planning from a particular environment model.

Other than sampling method and a collision detection module, the SBMP framework has two main primitive procedures, Steering function and Near neighbours function. Steering function computes the (optimal) path segment between two configurations without considering the obstacles. A holonomic robot could travel on straight lines in any degree of freedom, therefore a straight line would be the steering function. Depending on the optimization criteria, the cost to move between two configurations can be computed with steering function as well. For a holonomic system with length as optimization criteria, the euclidean distance would be the cost.

The near neighbours function for a configuration would return all nearby samples, which

can be reached within the cost threshold limit. The set of all these samples is called a reachable set. For a holonomic robot with euclidean distance as cost function, the reachable set would be an n-dimensional sphere (euclidean ball), where n is the degree of freedom of a robot. The SBMP algorithm solves the problem by connecting randomly drawn samples within the reachable set using a steering function and verifying the edge via a collision checking module.

Search based planning algorithms will return a solution in finite time, while SBMP algorithms provide a weak notion of completeness, "Probabilistic completeness", the probability that the planner fails to return a solution if one exists, decays to zero as the number of samples approaches infinity [2]. Depending on the sample connection procedure, some SBMP algorithms provide asymptotic optimality, which means it will almost assure convergence to optimal paths as algorithm progress.

SBMP fall broadly into two categories: multiple query and single query algorithms. Multiple query algorithms also referred to as graph-based methods construct a graph of possible robot motions covering the entire C space called a roadmap and then use a graph search algorithm e.g. Djikstra, to find a solution. E.g. probabilistic roadmap algorithm (PRM) [19] and variants (e.g., PRM* [20]). Two-step approach of graph-based methods allows efficient solutions to multiple queries. For a global planner, building a roadmap as the street network is a common method for industrial automated guided vehicles. However, in a rapidly changing environment, it is computationally challenging or even infeasible, therefore has not been considered in this work. Single query algorithms also called as tree-based methods such as rapidly exploring random trees algorithm (RRT) [21] and variants (e.g., RRT* [20]), have emerged as an online alternative to graph-based methods. Tree-based methods build a tree of possible robot motions rooted at the start state and/or goal state and the algorithm terminates as soon as a solution is found. If a Tree-based SBMP algorithm's solution would almost inevitably converge to the optimum, then it is considered an Optimal Tree-based SBMP algorithm.

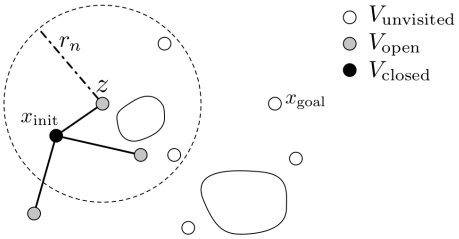
Depending on the tree extension procedure, Optimal Tree-based SBMP algorithms can be further classified into two, incremental sampling methods, e.g. RRT* [20] and batch sampling methods, e.g. FMT* [22]. The incremental sampling method adds a single sample at a time to the tree rooted at the initial state. After generating a new valid sample (x_{new}), nodes of the tree falling within the reachable set of x_{new} are considered for connections and the edge between them is checked for collision. The x_{new} is connected to the node which provides optimal cost-to-arrive from x_{init} in the tree. Followed by a rewiring step, which would rearrange the edges, if a better solution is possible by passing

though x_{new}. In incremental sampling methods, the sample set is grown during run-time and solution refinement continues until, e.g., a computation time limit is reached.

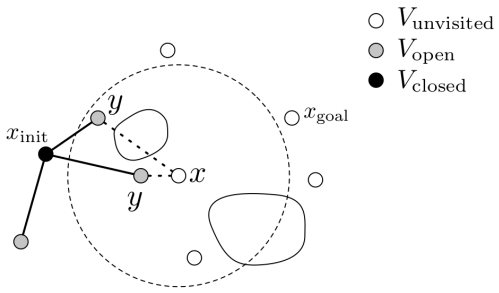
In the batch sampling methods, the states are sampled at initialization and held fixed during run-time, e.g. Fast marching tree (FMT*) [22]. FMT* incrementally builds the tree in an "outward" direction rooted at the x_{init}, therefore it does not have to update over previously evaluated sample points, as in the rewiring step in RRT*. In FMT*, the collision is checked only for the optimal edge, therefore its called Lazy collision detection method [22].

Compared to RRT* based algorithms, FMT* based algorithms provide better performance as stated in [22], therefore FMT* based algorithm has been used in this work and a detailed description based on [22] has been provided. FMT* uses three sets to store samples, $V_{unvisited}$ set consists of the samples that have not yet been added to the tree, V_{open} and V_{closed} set contains samples that have been added to the tree. Samples in V_{open} can be considered as the wave-front of the tree, active for further connections, while samples in V_{closed} set are not sufficiently close to the edge of the expanding tree to have any new connections with $V_{unvisited}$ and no longer considered for any new connections.

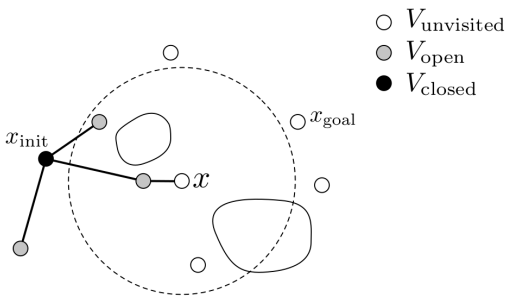
The algorithm starts by placing x_{init} into V_{open} and all other samples in $V_{unvisited}$, while V_{closed} is initially empty. The cost of a node $c(x)$ in the tree is the cost to arrive at x from x_{init}. An iteration of the FMT* algorithm can be visualized in Figure 2-5.



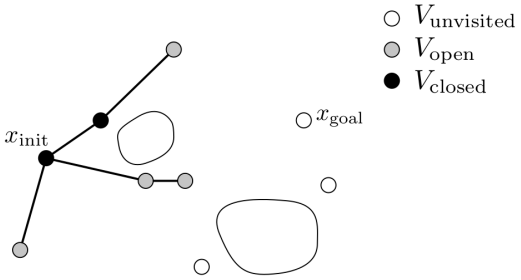
(a) FMT* selects the lowest-cost node z from set V_{open} and finds samples from $V_{unvisited}$ within the reachable set of z.



(b) Given a sample x from $V_{unvisited}$, FMT* finds nodes from V_{open} within the reachable set of x and searches for a connection which minimize the cost to reach the x from x_{init}. (Equation 2.2)



(c) After selecting the locally-optimal one-step connection to x ignoring obstacles, the edge is added to the tree if it is collision-free.



(d) After all samples of the reachable set of z in $V_{unvisited}$ have been explored, FMT* adds successfully connected nodes to V_{open} , places z in V_{closed} , and moves to the next iteration.

Figure 2-5: An iteration of the FMT* algorithm. [22]

Selection process of a parent node y from Y_{near}, ($Y_{near} \in V_{open}$) for a new sample x ($x \in V_{unvisited}$) is expressed in equation 2.2.

$$y_{min} \leftarrow \underset{y \in Y_{near}}{\arg\min} \left(c(y) + Cost(y, x) \right) \tag{2.2}$$

Where $Cost(y, x)$ represents the minimum cost of moving from a node y to new sample x. This process will select the parent node y_{min} which provides the lowest cost of moving through it.

The algorithm terminates when the lowest-cost node in V_{open} is in the goal region or when V_{open} becomes empty. If sampling-based solution refinement is desired, the tree output by FMT* can be imported directly into an RRT* routine for further processing. A bidirectional extension, the Bidirectional-FMT* (BFMT*) [23], relies on a two tree implementation of FMT* algorithm [22] rooted at start and goal states. Next, the extension of SBMP algorithms for a nonholonomic system is presented.

2.5 Sampling-Based Motion Planning for Nonholonomic Systems

To extend the SBMP to nonholonomic systems, mainly three approaches are common in the literature [24]. Algorithms such as Expansive Space Trees (EST) [25] and Stable Sparse-RRT* (SST) [26], extend the tree using forward simulation of the random control input through a vehicle model. Although these algorithms can be applied to a variety of robotic systems, they can take a long time to find a solution and random control inputs would be inefficient when the dynamics are complex and/or unstable. To address this issue, Closed-loop RRT (CL-RRT) [27] utilizes a two tree approach. CL-RRT connects the new sample to the tree by a reference path, e.g. straight lines for a car-like vehicle, and forward simulates the reference path through the controller and the vehicle model, which would generate a vehicle drivable state trajectory. The feasibility of the trajectory is checked against environmental constraints such as obstacle avoidance and if successful, it is added to the state trajectory tree.

The third and most popular approach samples configurations directly and tries to connect them with a vehicle drivable curve (steering function). However, this approach is only feasible if easy to compute vehicle drivable curve between two configurations (two-point boundary value problem (TPBVP)) is available. The reachable set for nonholonomic systems would be much complex than a euclidean ball, therefore numerical approximation has been suggested.

Kinodynamic-RRT* [28] and Differential-FMT* (DFMT*) [29] would be the natural extension of RRT* [21] and FMT* [22] respectively by utilizing the steering function and the reachable set modifications. However, DFMT* provides a better theoretical guarantee of optimality and requires less computation time compared to Kinodynamic RRT*, stated in [29].

For the Reed Shepp car model, the analytical solution of the optimal path length problem between two configurations can be computed using the Reed Shepp curve [13]. A Reed Shepp car with an arc-length budget of r car can move a distance r forward or change its angle by $r \times \kappa_{max}$ through moving with a maximum steering angle (orientational reachability), but it has to move forward and backward while changing steering angle between its extremes to move sideways, which is on the order of r^2. The reachable set for a Reed Shepp car model can be approximated using a rectangular box with an aspect ratio of $r : r^2 : r$ in x, y and θ dimension respectively (weighted euclidean box), proposed in [28].

Until now, we have discussed the SBMP algorithm and its extension for a nonholonomic system. Next, we will review the problem-specific sampling methods which can improve the performance of SBMP algorithms.

2.6 Problem-Specific Sampling Methods

SBMP algorithms generally draw random samples from a uniform distribution to cover the entire C space and utilise them to find a solution. However, not all samples would contribute to the solution for a particular path planning problem, which makes them inefficient. Therefore sampling strategy which reduces the exploration in the non-promising region would be suitable for better computational performance. These sampling methods take into account the problem-specific constraints, e.g. start and goal states, obstacles and bias the samples in the regions, where an optimal solution may lie. Before describing different problem-specific methods, a concept of the optimal problem-specific sample set has been presented.

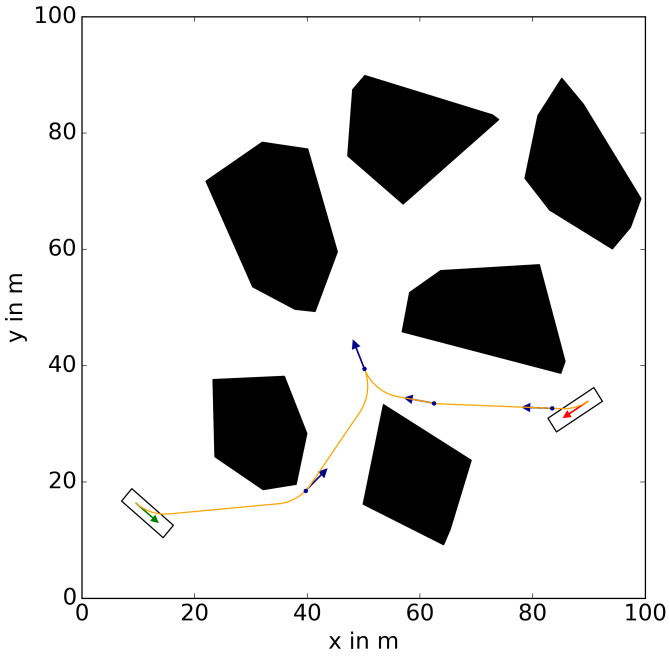


Figure 2-6: The optimal problem-specific sample set (blue poses) to find the path between start pose (green) and goal pose(red)

The optimal problem-specific sample set would be the minimum states required to generate the collision-free optimal path, with the defined steering function. In Figure 2-6, the optimal path can be computed by connecting the samples represented by blue arrows using Reed Shepp curve, which would be the optimal problem-specific sample set for this problem. However, directly generating the optimal problem-specific sample set from the problem definition is non-trivial and many easy to compute approximation methods have been proposed.

Two types of approaches are common in problem-specific sampling methods. Either modify the sampling distribution over the whole C space or compute the optimal region/subspace for sampling. These non-uniform sampling methods can be classified into four groups [10]: heuristics for biased sampling, informed sampling, adaptive sampling, and learned sampling. Next, we will get a brief description of the available algorithms for each problem-specific sampling method.

2.6.1 Heuristics for Biased Sampling

Heuristics biased sampling utilizes heuristics designed for a particular type of problem for biasing the samples. The earliest approach uses information about obstacles in the environment to change the sampling distribution over the C space, e.g. increase the sampling probability near obstacles and in narrow regions presented in [30] and [31]. In the DARPA challenge, the team from MIT used a similar method [32] with CL-RRT[27], which resulted in significant performance gains.

For a robot with complex dynamics, computation of a sub-space, which includes the optimal solution, for sampling has been significantly successful in the past. To compute the sub-space, a variety of methods by ignoring or simplifying the vehicle model has been suggested. Theta*-RRT[33] first computes a geometric collision-free path with discrete any-angle search and sample around the path for SBMP algorithm, which can be visualized in Figure 2-7. The two-stage RRT[34] uses a similar approach by solving the problem for a holonomic system using SBMP and uses normal distribution around the path to represent the sub-space.

Figure 2-7: Theta* RRT approach for a car-like vehicle. S and G represent the start pose and goal configuration respectively. The any-angle path of Theta* (in red) is used to guide the samples. The motion tree is represented in blue. [33]

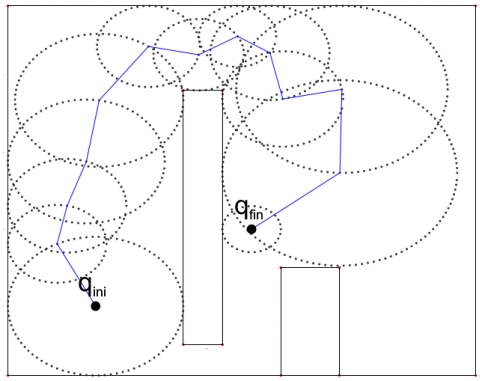


Figure 2-8: Free subspace computed using Probabilistic Foam Method (PFM) [35].

From Figure 2-7, it is clear that sampling subspace would contain obstacle regions, which would generate many in-collision samples. Therefore, the computation of collision-free

subspace is more beneficial.

Probabilistic Foam Method (PFM) [35] builds a tree similar to the RRT. PFM extends the tree by first computing free space around a configuration and then generating random samples on the surface of free space. By repeating this process, a tree of free space volume can be obtained, which is visualized in Figure 2-8. This would be a better representation for sampling subspace as it does not contain in-collision sample.

Until now, all the methods for subspace computation consider the robot as a holonomic system. However, it will lead to sub-optimal sampling subspace because of the decoupling approach. Therefore, consideration of the vehicle model for computation of sampling subspace is quite important. Orientation-Aware Space Exploration (OASE) [5] utilize a simplified car model with free space around a configuration as a node similar to the PFM approach for the computation of sampling subspace (Figure 2-9). In [6], OASE has been used as a benchmark for problem-specific sampling for a car-like robot.

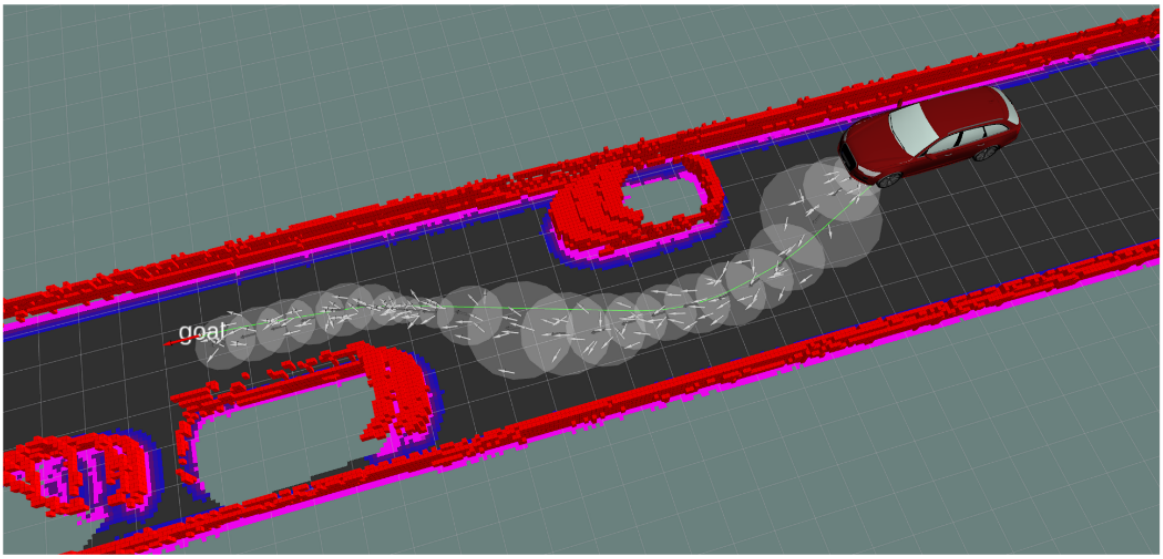


Figure 2-9: Space exploration biased sampling as proposed in [6]. The subspace is visualized by white circles on the ground and the samples are represented by grey arrows. The final path is represented by a green curve.

From available heuristic biased sampling methods, OASE method is the most promising approach. Other than free space volume, OASE utilizes a heuristic to guide the tree expansion, which can reduce the cost of computing the sampling subspace. Therefore, it has been considered for further evaluation.

2.6.2 Informed Sampling

Rather than generating a static solution for sampling distribution, Informed sampling methods try to improve sampling quality by leveraging the cost of current best path. Assume the cost of current best solution is $Cost_{best}$, then if the optimal cost to move from the start state x_{init} through a new sample state x_{new} to the goal state x_{goal}, ignoring the obstacles in-between, is less than $Cost_{best}$, then only x_{new} can improve the path, as written in Equation 2.3 and set of states that can provide a better solution can be defined as the omniscient set. [36].

$$Cost(x_{init}, x_{new}) + Cost(x_{new}, x_{goal}) < Cost_{best} \tag{2.3}$$

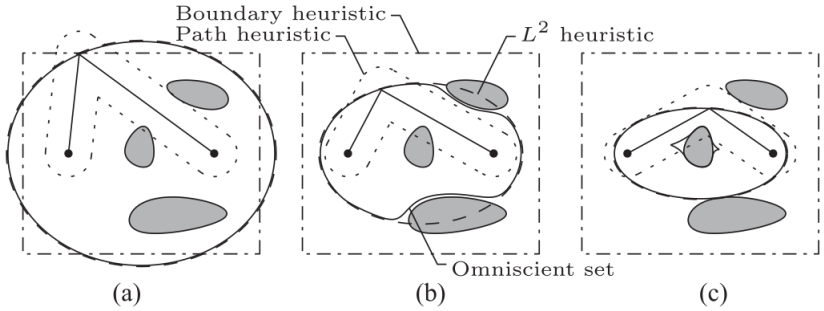


Figure 2-10: Illustration of the informed set with changing best cost from (a) to (c) as proposed in [36]. Path heuristic represents a tunnel around the current best path as subspace for sampling while the omniscient set describes a subset which can provide samples for a better solution, which can be approximated by L^2 informed set.

The omniscient set does not contain in-collision configurations, therefore it has been approximated using informed set, which can contain in-collision configurations, as can be understood by Figure 2-10 (b), where a solid line represents the omniscient set while a dotted line represents the informed set (L^2 heuristic). For a 2D holonomic system with the euclidean distance as cost function, ellipses can understand as the informed set, as shown in Figure 2-10 and generalization of it for n-dimensional holonomic systems would be symmetric n-dimensional ellipses called as L^2 informed set. [36]

For systems with differential constraints, informed subset would be more complex and no method has been proposed to directly sample from it. Instead, samples within the informed subset can be possible using rejection sampling which is inefficient because the ratio of the informed set volume to the overall configuration volume could be quite low,

which results in high rejection rate. To improve the performance, Hierarchical rejection sampling[37] and Markov Chain Monte Carlo method [38] have been proposed.

This procedure requires an initial solution to the planning problem before the sampling may be refined. For our problem finding an initial feasible path is challenging, therefore it is not considered for further evaluation.

2.6.3 Adaptive Sampling

Adaptive sampling methods try to model the C_{free} space by utilizing collision checking information of previous samples, to reduce the generation of in-collision samples. Adaptive sampling methods are particularly useful for high-dimensional planning problem, e.g. Robot manipulator, where collision detection is expensive.

In an earlier approach, Utility guided sampling [39] tries to model the C_{free} space using the k-nearest neighbour. For a robot manipulator, use of online Gaussian mixture model (GMM) learning has been proposed in [40].

For path problem in a large environment, modelling the C_{free} space would not result in much performance gain. Therefore, adaptive sampling is not considered for further evaluation.

2.6.4 Learned Sampling

Learned sampling methods use machine learning models to learn from the solution of previous planning problems. Learning-based approaches can improve the sampling quality dramatically, as they do not require information to be acquired in a given motion planning problem before refining sampling distribution as in Informed sampling and Adaptive sampling and eliminate the computation of space exploration as in heuristics for biased sampling. The main objective of learned sampling methods is to avoid manual algorithm development for problem-specific sampling.

In machine learning, data used for training the model, e.g. previous path planning solutions, is termed as training data. Machine learning models can be further classified into Discriminative models and Generative models.

Discriminative models

The discriminative model tries to learn the mapping between observation or condition (y) and target (x) from the training dataset [41] , e.g. Image classification. For learned sampling, discriminative models have been used to map probability distribution over the C space from the planning problem and use them for direct sampling.

Recently, convolutional neural networks (CNN) have shown great success in image processing, which has been utilized in learned sampling by generating the sampling distribution as a 2D histogram represented by an image. e.g. for social navigation, a fully convolutional neural network has been suggested in [42], it predicts the path and outputs it as an image for sampling. In another approach, a convolutional autoencoders model has been used to identify critical regions for sampling. [43]

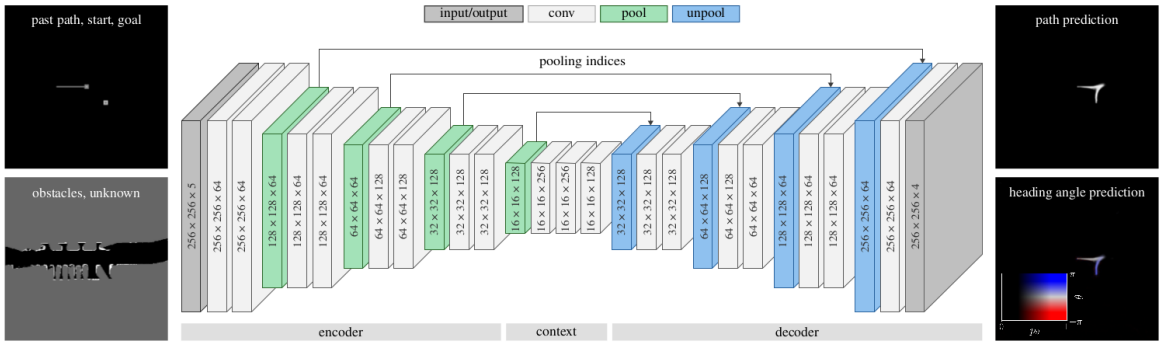


Figure 2-11: The architecture of the CNN proposed in [6] for learned sampling method. It predicts a two-dimensional sampling distribution over future vehicle positions (top right) and the heading angle (bottom right). The five input girds, which describe the current driving situation, are blended into two images on the left.

For a car-like robot, the convolutional autoencoders model has been proposed, which can be understood by Figure: 2-11. The model takes the problem definition and raw sensor data, such as lidar data, obstacle map, start pose and goal pose and generates the future path represented as images for sampling [6].

Generative models

The generative model tries to learn the probabilistic model of training data and can generate new samples following the same probabilistic distribution of the training data [44]. In literature, mainly two models are popular for generative modelling, Generative adversarial networks (GAN) [45] and variational autoencoder (VAE) [46]. Both models have been

utilized in the area of motion planning and shown good results. For optimization-based planners, a bad initial path can lead to a sub-optimal solution. With Conditional GAN, better initial path generation from problem definition has been proposed in [47].

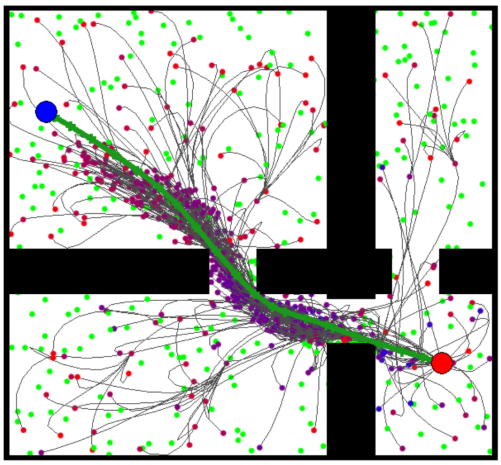


Figure 2-12: A fast marching tree (FMT*) generated using learned sampling method for a double integrator system proposed in [10]. The problem-specific samples are generated using the initial state (red circle), goal region (blue circle), and workspace obstacles (black). Moreover, purple, red and green dots represent samples from V_{closed}, V_{open} and $V_{unvisited}$ respectively. The green line represents the final path computed by FMT*.

In [48], the author has used Conditional Variational Autoencoder (CVAE) for problem-specific sampling, which can be understood by Figure 2-12. Compared to image-based learned sampling methods, CVAE provides a more flexible model, capable to scale to a higher dimensional problem. Moreover, CVAE is capable of learning complex manifolds, which would be helpful for learned sampling for nonholonomic systems. Therefore, CVAE model has been considered for learned sampling.

In the next chapter, we will get comprehensive information about space exploration biased sampling method for a car-like robot. Then, we will proceed to a more generic algorithm of the learned sampling method.

3. Space Exploration Biased Sampling

In Motion planning, to extract C_{free} space information from the environment description, space decomposition methods, e.g. cell decomposition [12], are quite popular. They use simple geometric shapes to decompose the environment and build a graph like structure of C_{free} space. Instead of decomposing the entire environment systematically, exploration-based space decomposition algorithm tries to understand the connectivity of C_{free} space for a particular motion planning problem, which reduces the problem complexity. Before discussing the space exploration algorithm, it would be helpful to understand free space bubble.

3.1 Free Space Bubble

The concept of free space bubble was first introduced with Elastic Band in [8]. A free space bubble β_q represents a local subset of C_{free} space around a configuration q of the robot. A bubble can be computed efficiently by measuring the minimum distance to the obstacles. For example, β_q for a point mass robot at configuration q would be a n-dimensional sphere (n is the degree of freedom) with radius equals to the distance to the nearest obstacle $d(q)$, which can be written as Equation 3.1, where $|| \cdot ||_2$ is the euclidean distance.

$$\beta_q = \{p : \|p - q\|_2 < d(q)\} \tag{3.1}$$

In this work, the vehicle model is considered as a planar robot. The distance from the origin of the robot to the farthest point on the robot is defined by r_{max}. The movement of a planar robot can be decomposed into two, translation and rotation. In only translation movement, each point on the robot moves the same distance. While, in only rotational movement, the distance travelled by each point on the robot depends on its distance from the origin of the robot. However, point with distance r_{max} would travel the largest

distance, which is referred to as rotational distance. The free space bubble for a planar robot can be approximated by the sum of the translational ($d_{trans}(q,p)$) and rotational ($d_{rot}(q,p)$) distances, upper bounded by $d(q)$ expressed by Equation 3.2 as proposed in [8]. A bubble for a planar robot can be imagined by a double-sided cone shown in Figure 3-1.

$$\begin{aligned} \beta_q &= \{p : d_{trans}(q,p) + d_{rot}(q,p) < d(q)\} \\ d_{trans}(q,p) &= \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2} \\ d_{rot}(q,p) &= r_{max}|q_\theta - p_\theta| \end{aligned} \tag{3.2}$$

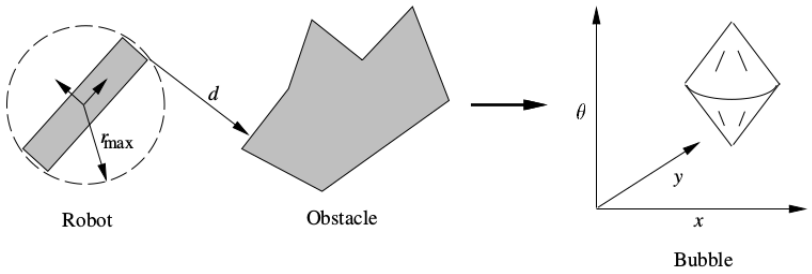


Figure 3-1: A free space bubble for a planar robot represented by a double-sided cone as proposed in [8]. The bubble is computed by utilizing the minimum distance d from the obstacles.

3.2 Space Exploration

The space exploration algorithm utilizes a bubble as a building block for exploring C_{free} space. The algorithm starts by computing $\beta_{q_{init}}$ around the start configuration q_{init} and discretizing the bubble surface in a grid-based or random manner, which becomes the wave-front and by repeating the process iteratively a tree structure rooted at q_{init} can be obtained [49]. The algorithm would terminate when the goal region overlaps the tree. Subsequently, a sequence of adjacent bubbles is assembled by backtracking over the tree, which is referred to as a C_{free} space tunnel T, which can be written as

$$T = \{\beta_{q_{init}}, \beta_{q_1}, \beta_{q_2}, ..., \beta_{q_{goal}}\} \tag{3.3}$$

The space exploration algorithm generally requires only a few types of queries such as binary collision detection and minimum distance to the obstacles. Hence, a generic design

of the algorithm is possible for flexible modification according to the different environment models, e.g., an object list, an occupancy grid.

Other than for sampling step in SBMP algorithm, the bubbles computed by space exploration algorithm can be used for avoiding actual collision detection in the explored region. Furthermore, by combining the nearest-neighbour searches with the bubbles computed during exploration, collision checking can be eliminated in the explored region as proposed in [50]. For path smoothing algorithm, the bubbles can act as a safe region for optimization, proposed in [8] and [51].

As mentioned earlier, a double-sided cone would be the bubble for a car-like robot. However, not all configurations on the surface of the double-sided cone would be easily reachable for a car-like robot from the centre configuration, e.g, the top point of the double-sided cone represents the same translational position with a shift in heading angle, which requires back and forth motion for the car, similar to the parallel parking. Therefore, relaxation of the bubble shape, which can be easily discretized while considering the vehicle constraints, becomes important.

3.3 Orientation Aware Space Exploration

In [5], the author suggested a path planning algorithm for a car-like robot, Orientation Aware Space Exploration Guided Heuristic Search (OSEHS). It first computes a tunnel and then guides a search-based planning algorithm similar to State lattice [17] using the tunnel information. In this work, the first part of OSEHS, which is Orientation Aware Space Exploration (OASE) is used for tunnel computation.

Orientation Aware Space Exploration (OASE) [5] utilizes a cylinder in place of a double-sided cone to represent the bubble, although the cylinder would be partially free. However, it allows a straightforward discretization respecting vehicle constraints. The base of the cylinder is identical to the double-sided cone but height is estimated by vehicle orientational reachability from the centre configuration (q), defined by

$$h_\theta = 2r_q \times \kappa_{max} \tag{3.4}$$

where r_q is the radius of the cylinder around the configuration q which is same as the minimum distance from the obstacles. From here on, a partially free space cylinder would be referred to as a bubble. A bubble β_q in the C space can be defined by the configuration

q and radius r_q. For car-like robot, the configuration is (x_q, y_q, θ_q), therefore a bubble can be defined by

$$\beta_q = [x_q, y_q, \theta_q, r_q] \tag{3.5}$$

OASE discretizes the bubble in a grid-based manner, the bird view of it can be visualized in Figure 3-2. The configurations on the surface of β_q would be child vertices of a parent vertex q. The translational position of the child vertices is uniformly interpolated over the circle and the angle is defined by a vector aligning with the parent vertex and child vertex. The child vertices in a front half-circle can be reached easily by driving forward, therefore, it has given orientation pointing outward and the opposite for others. Furthermore, two discrete child vertices are created on both sides at $\pi/2$ with opposite directions. This discretization method is also referred to as All angle discretization. After computing the configurations of child vertices, the minimum distance to the obstacles is also computed for each child vertex.

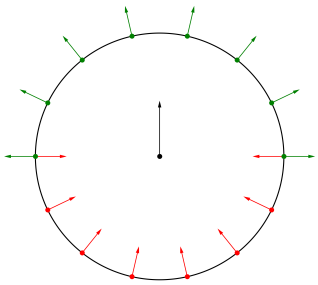
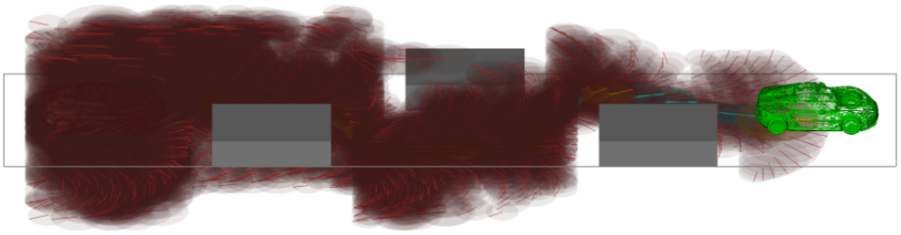


Figure 3-2: The bubble discretization proposed in OASE [5] (bird view). The black circle represents the bubble β_q around configuration q (black arrow). The green and red arrows represent the discretization of free space surface with the outward and inward motion direction respectively.

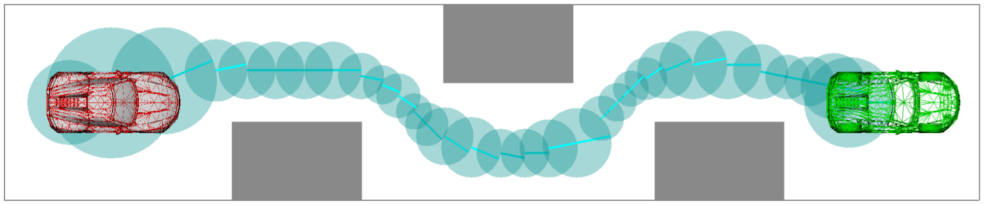
Rather than expanding the tree in all directions, OASE guides the exploration in goal direction using heuristics and perform graph search as A* algorithm. In OASE, the actual cost (g) to reach from parent to child vertex and heuristic cost (h) to reach from a child to the goal vertex is expressed by d_{ij}, cost to move from i vertex to j vertex. The d_{ij} is calculated by Equation 3.6, where, $\overrightarrow{\mathrm{p_i}} - \overrightarrow{\mathrm{p_j}}$ represents the euclidean distance and $|\theta_i - \theta_j|/\kappa_{max}$ represents the minimum arc length required to turn between θ_i and θ_j.

$$d_{i,j} = max(\overrightarrow{\mathrm{p_i}} - \overrightarrow{\mathrm{p_j}}, |\theta_i - \theta_j|/\kappa_{max}) \tag{3.6}$$

As mentioned earlier, the algorithm starts by computing the bubble $\beta_{q_{init}}$ around initial configuration q_{init} and generating the child vertices. During the tree expansion, the vertices are maintained in two sets: a closed set V_{closed} for the already evaluated vertices and an open set V_{open} for the fresh ones from expansion. So the child vertices are added to the open set, while the parent vertex is moved from the open set to the closed set. Vertex with the minimum sum of actual cost and heuristic cost (f = g + h) is selected from the open set V_{open} for further expansion and the tree is constructed by expanding vertices iteratively. To reduce redundancy in the tree, before expansion, selected vertex from V_{open} is checked for overlapping from any vertices from V_{closed}.



(a) Side view of the progress of OASE. The brown circles represent the explored region.



(b) Bird view of a tunnel (blue circles) computed by OASE. The blue line represents the configuration q of a bubble β_q.

Figure 3-3: Space exploration of OASE method for a path planning problem defined by start pose (red), goal pose (green) and obstacles (gray). [9]

If the expanded vertex is overlapping with goal vertex, the space exploration returns success and a tunnel is constructed with backtracking over the tree as shown in Figure 3-3. The space exploration fails when the goal is unreachable and no more vertices can be expanded, i.e., the open set V_{open} is empty. For completeness, the pseudo-code for OASE is given in Appendix A. In the next section, we will discuss the issues of OASE on a Paketzentrum use-case and address potential flaws with the algorithm.

3.4 Orientation Aware Space Exploration in Complex Environments

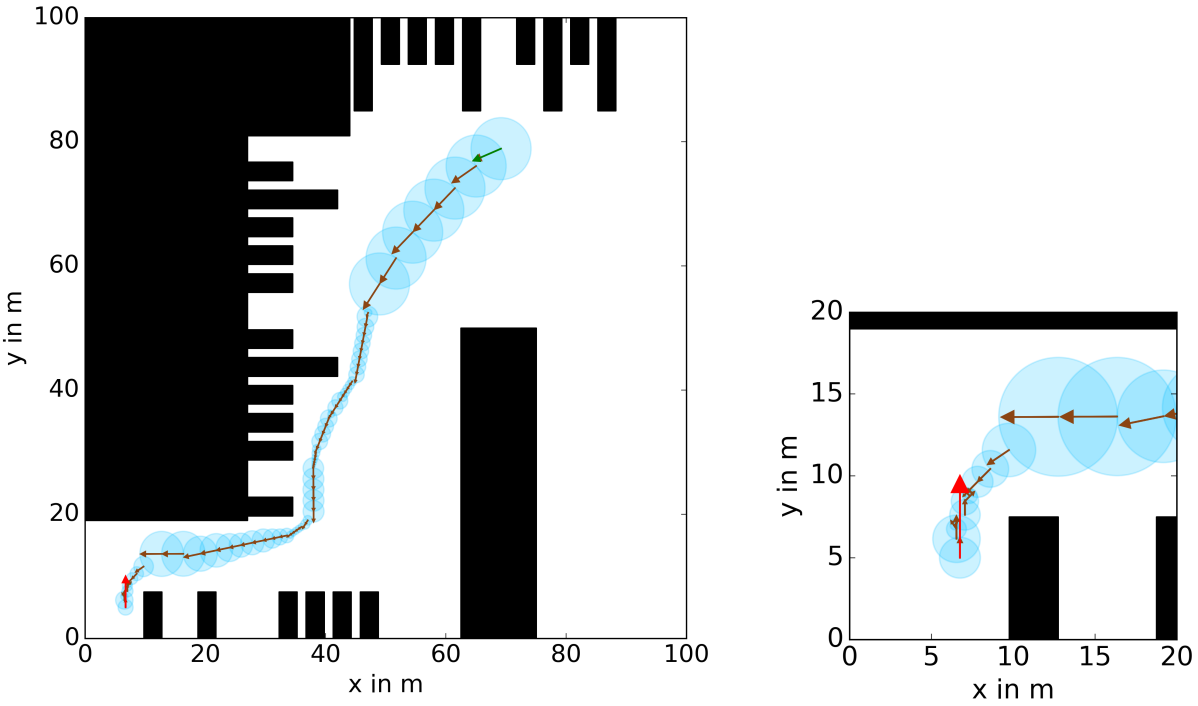


Figure 3-4: Path planning problem of reverse parking at the end of a narrow region defined by a start (green arrow) and goal (red arrow) pose. The tunnel is computed by OASE method and the bird view of the tunnel is represented by circles (blue). The brown arrow represents the configuration q of the bubble β_q. The OASE algorithm tries to switch the moving direction after entering the narrow region which is not feasible. A zoomed-in section is provided for better visualization in the right.

Reverse parking at the end of a narrow region is a common use-case for a Paketzentrum, described in Figure 2-1. It requires switching the direction of movement (from forward to backward direction) before entering the narrow region. However, OASE tries to switch the direction at the very end (Figure 3-4) which is not possible for a large vehicle e.g. truck. The computed tunnel using OASE does not contain a feasible path and would not be helpful in the sampling step for SBMP. Similar flaws with OASE algorithm can be observed in Appendix B.

The main reason for this behaviour could be the discretization or cost function. The OASE discretizes the whole circle, as presented in Figure 3-2, although depending on the bubble's radius, only a few vertices are easily reachable from the centre configuration.

Moreover, the heuristic function considers the cost of switching angle only when the minimum arc length is greater than the euclidean distance. As mentioned earlier, before expanding a vertex from V_{open}, it is checked for overlapping with any vertices from V_{closed}. An inadequate heuristic function could expand non-promising vertices and cover the C_{free} space. However, eliminating overlapping check would increase the computation by a large margin, hence better heuristic function should be considered.

3.5 Modified Orientation-Aware Space Exploration

Two improvements to mitigate, the above-mentioned problems in OASE algorithm, are described in this work.

- a modified discretization
- a more reliable cost function.

The modified discretization scheme considers configurations only inside the reachable angle computed by

$$\triangle\theta = r_q \times \kappa_{max} \tag{3.7}$$

where r_q is the radius of the bubble β_q. The translational positions for child vertices are uniformly interpolated between the angle limits as shown in Figure 3-5. Moreover, three discrete child vertices are created at each translational interpolation. The orientation of the centre child vertex is the same as the original discretization and side child vertices are created with an angle difference of $\triangle\theta$. Modified discretization method is also referred to as Limited angle discretization. Limited angle discretization shares some configurations with All angle discretization. However, it is not a subset of All angle discretization.

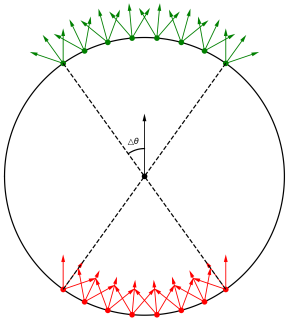
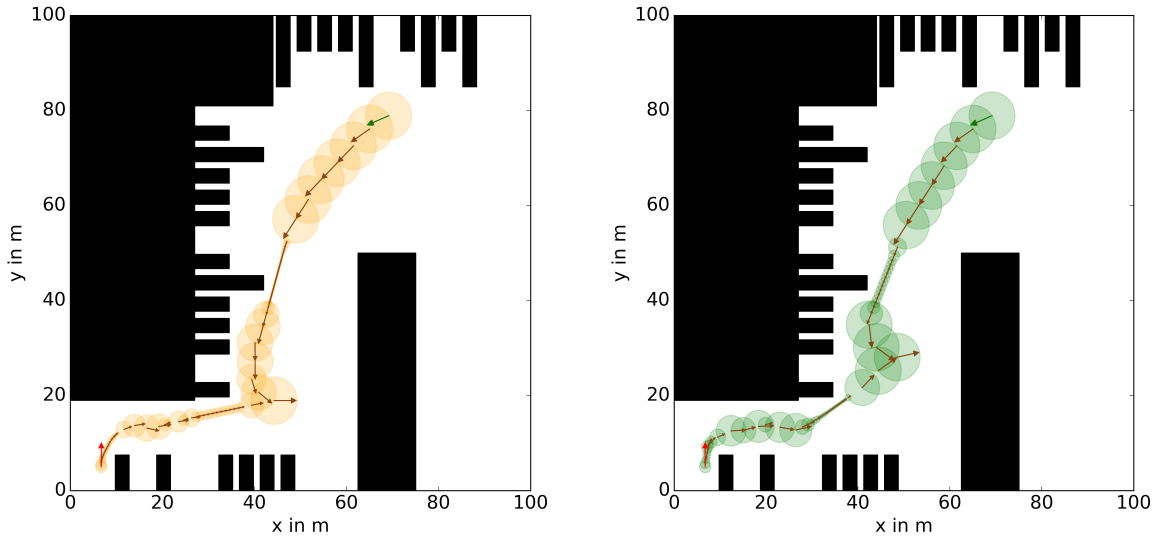


Figure 3-5: Modified bubble discretization (bird view). The black circle represents the bubble β_q around configuration q (black arrow). The green and red arrows represent the discretization of free space surface with the outward and inward motion direction respectively.

With Reed Shepp curve as the steering function for SBMP algorithm, length of Reed Shepp curve (RS length) would be a better cost function than Max(Euclidean distance, Minimum arc length for turning), in shorthand Max(Euclid, Curve), presented in Equation 3.6. Therefore, for each variance of space exploration method, actual cost (g) is considered as RS length. To measure the improvement with modified bubble discretization, the same problem of reverse parking at the end of a narrow passage is solved with both heuristic functions. (Max(Euclid, Curve) and RS length)



(a) Heuristic Function : Max(Euclid, Curve) **(b)** Heuristic Function : RS length

Figure 3-6: Planning problem of reverse parking at the end of a narrow region solved using modified discretization method and both heuristic functions. The circles represent the bird view of the bubbles and the arrow represents the centre configuration.

Figure 3-6 illustrates the solution computed by Limited angle discretization. The Limited angle discretization forces the tunnel to switch the direction before entering the tunnel. However, heuristic functions do not show much difference on a visual level and require numerical evaluation.

A very narrow region in the tunnel can be avoided by post-processing of space exploration tunnel as suggested in [9], however, it has not been implemented in this work. Next, different approaches to utilize the tunnel information for sampling in SBMP are discussed.

3.6 Probability Distribution

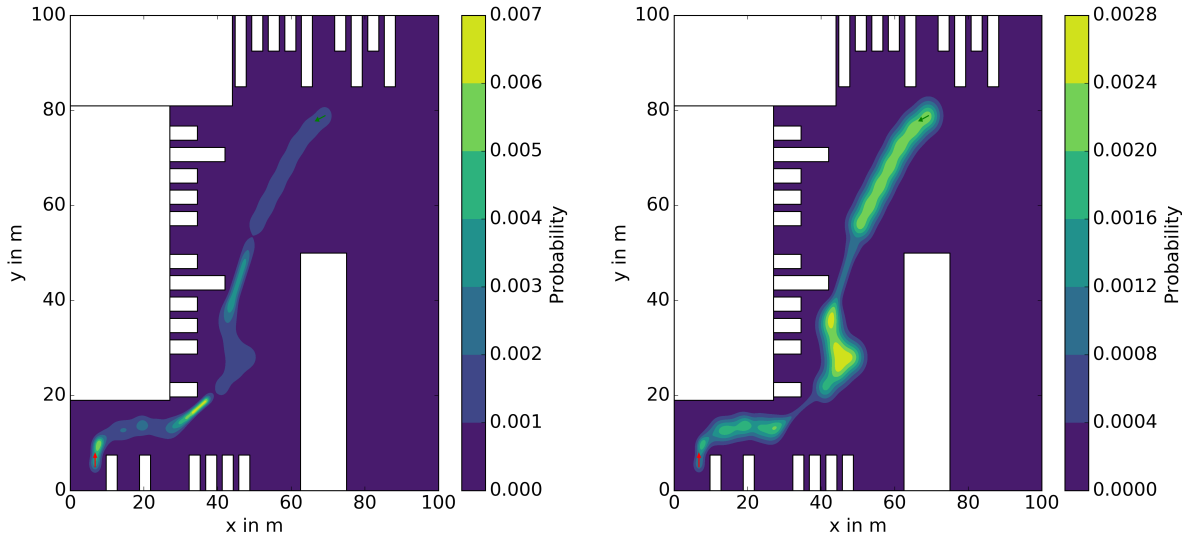
To sample from the tunnel, different approaches are possible. To sample uniformly from the tunnel, rejection sampling would be the most straightforward approach, but not efficient. The tunnel is a sequence of adjacent bubbles, which in this work is overlapping cylinders. Therefore, critical regions for the tunnel, e.g. narrow area and regions with high curvature change, can be identified easily and should be exploited.

In [6], the author proposed to sample directly from a bubble with a three-dimensional normal distribution, however, did not mention about sampling a bubble from the tunnel. To extend the idea, a two-step approach for generating a sample from the tunnel is described in this work. In the first step, a bubble β_{q_i} is sampled from the tunnel T, where i is the index of bubble. After picking the bubble β_{q_i}, a configuration is sampled from it.

For sampling, a bubble from the tunnel, the probability is distributed over all the bubbles according to its features, e.g. volume of a bubble, angle difference of consecutive bubbles and whether the tunnel switches direction of movement at a bubble. Probability of each bubble can be written by,

$$Pr(\beta_{q_i}) = \frac{r_{q_i}^{\alpha} f(\beta_{q_i})}{\sum_{i=0}^{n} r_{q_i}^{\alpha} f(\beta_{q_i})} \qquad \alpha \in \mathbb{R} \tag{3.8}$$

Where r_q represents the radius of the bubble and all the other features of β_q are combinedly represented as $f(\beta_q)$. To sample uniformly across all the cylinders, $\alpha = 3.0$ (equivalents to volume) would be a good option, however, to assign higher sample density in narrow regions $\alpha < 3.0$ has to be considered. For illustration purposes, translational probability is represented in Figure 3-7 for $\alpha = 1.0$ (a) and $\alpha = 2.0$ (b) with normal distribution over each bubble with the centre as the mean and radius as the standard deviation. It is clear from Figure 3-7 that $\alpha = 1$ is a more suitable option for high sample density in the narrow region, therefore it is selected for further evaluations.



(a) $Pr(\beta_{q_i}) \propto r_{q_i}^1$ **(b)** $Pr(\beta_{q_i}) \propto r_{q_i}^2$

Figure 3-7: The translational probability distribution over free space path corridor for different values of α in equation 3.8

To sample discrete vehicle pose from the selected bubble, a three-dimensional probability distribution $P(x, y, \theta)$ over the bubble is used. To compensate for near optimality because of discretization, sampling around the tunnel would be also a suitable option and to generate collision-free samples, giving more priority to the region inside the tunnel is beneficial. To find the effect of sampling distribution on the solution, four probability distributions over a bubble are utilized, which are listed in Table 3-1.

Number	Translational Distribution	Orientational Distribution
1	Normal distribution $X \sim \mathcal{N}(x_q, r_q)$ $Y \sim \mathcal{N}(y_q, r_q)$	Normal distribution $\Theta \sim \mathcal{N}(\theta_q,$ $min(r_q \times \kappa_{max}/3, \pi/6)$
2	Normal distribution $X \sim \mathcal{N}(x_q, r_q/3)$ $Y \sim \mathcal{N}(y_q, r_q/3)$	Normal distribution $\Theta \sim \mathcal{N}(\theta_q, \pi/6)$
3	Uniform distribution $R \sim U(0, r_d)$ $\Phi \sim U(-\pi, \pi)$ $X = x_q + R \cdot cos(\Phi)$ $Y = y_q + R \cdot sin(\Phi)$	Normal distribution $\Theta \sim \mathcal{N}(\theta_q, \pi/6)$
4	Normal distribution $X \sim \mathcal{N}(x_q, r_q)$ $Y \sim \mathcal{N}(y_q, r_q)$	Normal distribution $\Theta \sim \mathcal{N}(\theta_q, \pi/6)$

Table 3-1: Three-dimensional probability distributions over a bubble for sampling.

3.7 Number of Samples

The number of samples required to solve the problem increase with the length of the tunnel and to distribute samples evenly, the number of samples should be proportional to the volume of the tunnel, which can be approximated by the sum of the volume of bubbles, represented by,

$$N_{samples} = 2\pi\kappa_{max}\gamma \sum_{i=0}^{n} r_{q_i}^3 \tag{3.9}$$

where γ represents the ratio between the number of samples and volume which is a tuning parameter.

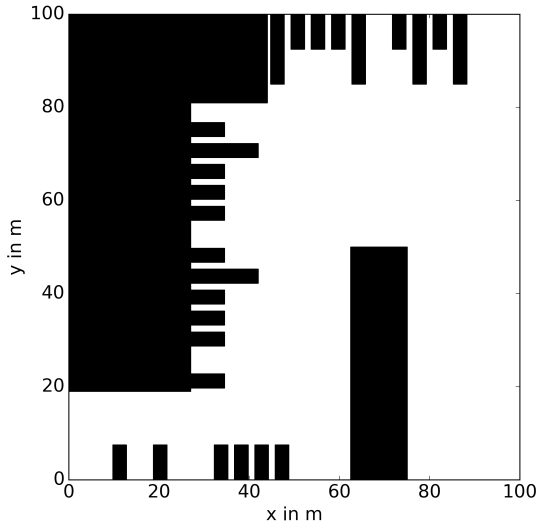
4. Numerical Evaluation of Space Exploration Biased Sampling

Before analysing the performance of space exploration biased sampling, we will get an overview of test settings used for evaluation. For all the experiments in this work, a single core of Intel i7@2.10 GHz computer has been used.

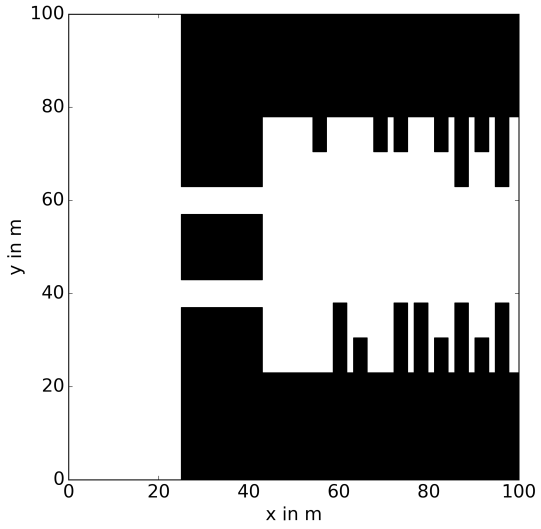
4.1 Test Settings

The typical size of the vehicle considered in this work is $9\,m \times 2.3\,m$ with the distance between the rear axle and the rear bumper around $1.5\,m$. Moreover, κ_{max} is considered to be $0.127\,(1/m)$. To represent a sufficiently large area of a Paketzentrum, the environment size of $100\,m \times 100\,m$ has been considered for all problems. To evaluate the performance of different space exploration biased sampling (SE biased sampling) methods, four environments are developed to replicate complex parking scenarios (Figure 4-1). The obstacles in the environment are defined by polygons, represented in black in Figure 4-1.

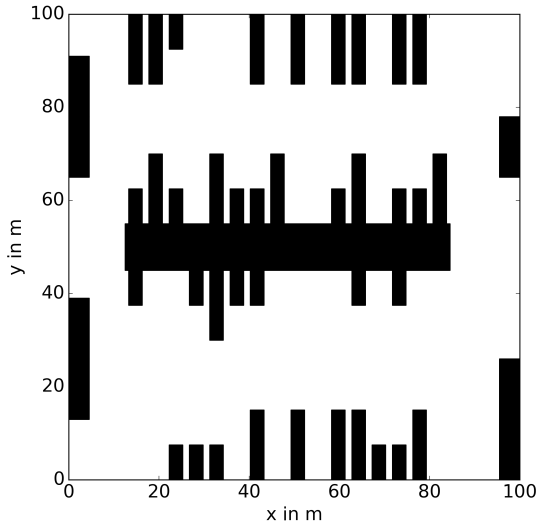
The test environment in Figure 4-1a replicates the path planning problem of reverse parking at the narrow end passage. This environment is designed to examine whether the planning algorithm is capable of switching the moving direction before entering the narrow region. The environment in Figure 4-1b displays the problem of crossing a narrow region. Planning through a narrow region is particularly hard for SBMP with uniform sampling method, as the probability of sampling from the narrow region is quite low [31]. A standard parking arena is represented in Figure 4-1c and an environment with random obstacles represented by convex polygons is represented in Figure 4-1d.



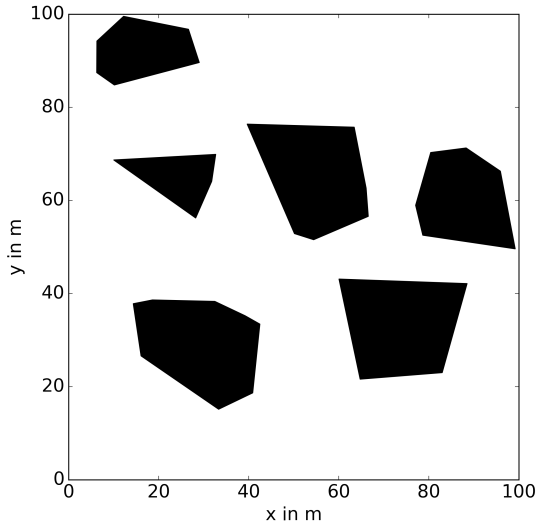
(a) Parking at the end of a narrow region.



(b) Manoeuvring though a narrow tunnel.



(c) Standard parking arena



(d) Environment with random obstacles

Figure 4-1: Complex Parking Environments.

4.2 Collision Detection

Fast collision detection is essential for SBMP algorithms because every new extension has to check for collisions with the environment, which can consume most of the computation power. Figure 4-2 presents an environment for path planning, where the obstacles are represented in convex polygons and the robot at a random pose (green arrow).

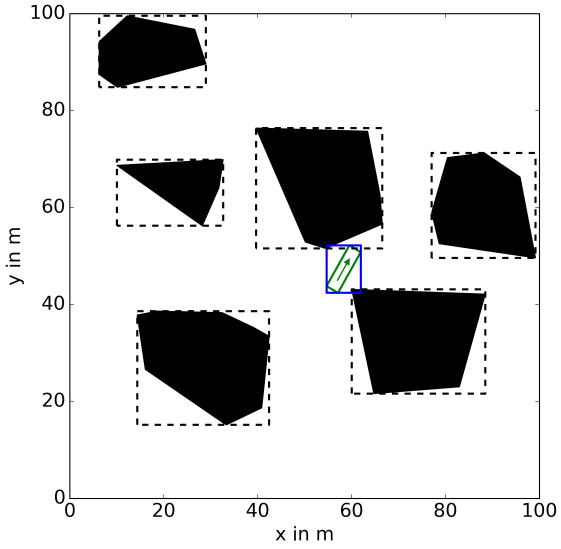


Figure 4-2: Illustration of the robot's body (green rectangle), Axis Aligned Bounding Box around the robot (blue rectangle), the obstacles (polygons in black) in the environment and Axis Aligned Bounding Box around obstacles (dotted black rectangle)

To avoid the collision check with each obstacle, in the first step Axis Aligned Bounding Box (AABB) with R-Tree data structure has been used to identify possible obstacles in a collision (Broad Phase) and then perform polygon to polygon collision check only with possible obstacles (Narrow Phase). The algorithm can be understood by Figure 4-2, with AABB collision check, two obstacles are in a collision with the vehicle (green), which is refined by polygon to polygon collision check. The collision detection module has been implemented using the Boost Geometry Library [52] and takes 1.47 ± 0.54 μs for binary collision check.

4.3 Implementation of Sampling-based Motion Planning for a Car-like System

In this work, open-source C++ library for SBMP, Open Motion Planning Library (OMPL) [7], is utilized. Moreover, the SBMP algorithm is modified for nonholonomic constraints and problem-specific sampling modules are developed for OMPL. Optimality criteria in SBMP would be varying based on the applications. For slow manoeuvring in a parking lot, the robot should drive mostly in the forward direction, maintain some clearance distance to the obstacles and minimize the path length. However, for initial development, only path length as the optimization criteria is considered.

For the Reed Shepp car model, the analytical solution of the optimal path length problem between two states can be computed using the Reed Shepp curve [13] (Figure 4-3), which has been used in this work.

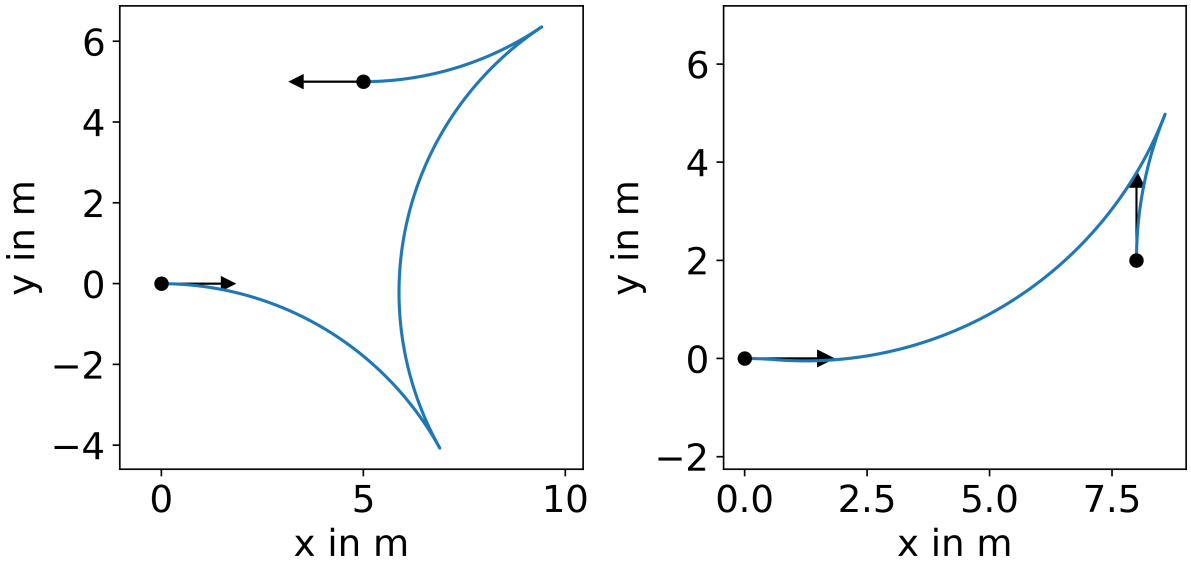


Figure 4-3: Reeds-Shepp Curves with $\kappa_{max} = 0.127$[53]

For the experiments, BFMT* from OMPL library is modified for nonholonomic constraints and Reed Shepp curve is used as steering function from OMPL library. For BFMT*, the cost threshold r for the reachable set can be computed using Equation 4.1,

$$r = 8(1+\eta)^{1/D} \left(\frac{\mu(C_{free})}{D}\right)^{1/D} \left(\frac{log(n)}{n}\right)^{1/D} \tag{4.1}$$

where $\mu(C_{free})$ represents the free space volume, which is approximated by the percentage of rejection samples during initial sampling. n represents the number of samples while D signifies the dimension of configuration space and η is a tuning parameter. As mentioned earlier, the reachable set for a Reed Shepp car model can be approximated using a weighted euclidean box. However, it is well defined for the cost threshold $r \to 0$ [54] and to achieve it, very high sample density is required. Therefore, a euclidean ball with radius computed by cost threshold is used. To reduce the computational cost, a better approximation of reachable set could be an area of investigation.

The convergence rate would increase with η, as it will consider more nodes for connection. However, it will also increase the amount of computation per sample. To optimize the convergence rate with respect to time, the balance of these two competing effects is required. Initial experiments suggest that $\eta = 2.0$ is suitable option. For purely geometric planning for a planner robot, Dimension (D) would be 3, but to ensure that connection radius r falls more slowly (as $N \to \infty$) than for purely geometric planning, it has been suggested to use $D = 4$ [55].

For path smoothing, a standard method from the OMPL package is used as a post-processing step. In the path smoothing technique, interactively configurations on the path are randomly perturbed and a short-cutting routine is applied.

4.4 Numerical evaluation of Orientation-Aware Space Exploration

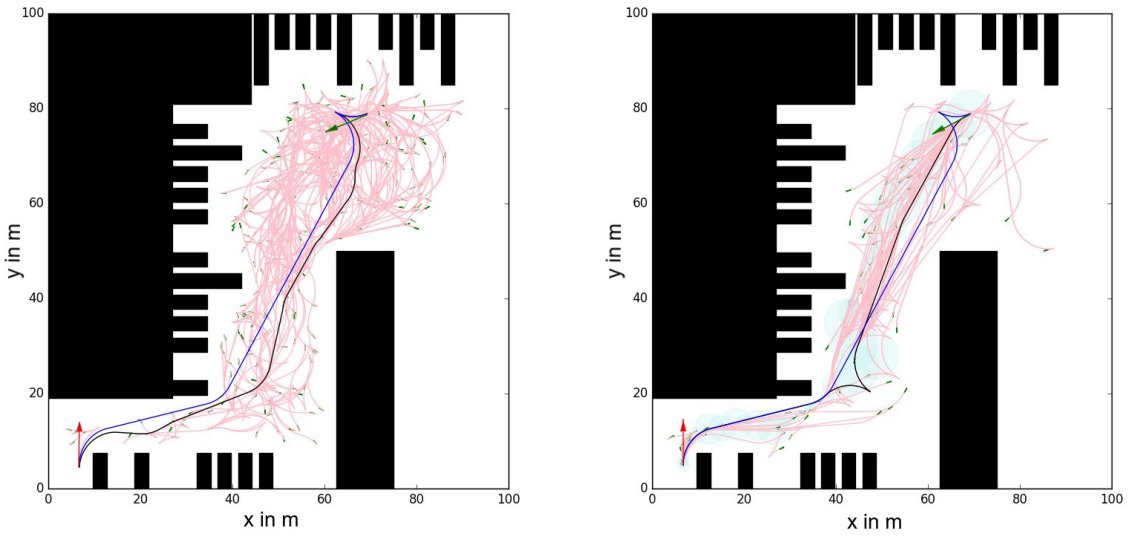
As mentioned earlier, the OASE algorithm computes a tunnel for sampling in SBMP . To avoid the exploration of very small bubbles, the radius is lower bounded by 0.2 m and to limit the width of the tunnel, the radius is upper bounded by 5 m. From the initial investigation, it was noticed that 32 discrete configurations from the bubble surface for child vertices would be sufficient. The radius for the bubble is bounded, therefore to reduce the online computation templet of the bubble discretization and the cost between parent and child vertex is calculated offline. Although the heuristic is calculated online here, for practical implementation, the look-up table could reduce the computational run-time to linear as suggested in [4].

The computational complexity of graph search algorithm depends on the number of explorations. The implementation of OASE in [9] computes the solution by exploring 2419-

states on an Intel Core i7 2.90 GHz CPU in $26.11 \pm 6.33\,ms$, where the number before and after $\pm$ represents the mean and the standard deviation of 100 trials respectively. While for a similarly complex problem, my implementation takes $3.32 \pm 0.7\,s$ on an Intel Core i7 2.10 GHz CPU. From initial profiling, it became clear that minimum distance to the obstacles computation is the bottleneck which takes $20.4 \pm 4.1\,\mu s$, which should be optimized for better performance.

4.5 Experimental Results of Space Exploration Biased Sampling

As illustrated in Figure 4-4, the space exploration biased sampling can reduce the search space of the SBMP algorithm. To evaluate the performance of different space exploration biased sampling (SE biased sampling) methods, a benchmark problem set has created. The benchmark problem set has 47 problems which replicate complex parking manoeuvre.



(a) Uniform sampling **(b)** Space exploration biased sampling

Figure 4-4: Illustration of motion tree (pink) generated by connecting samples (green arrow). The solution path (black) computed by the SBMP algorithm and a smooth path (blue).

In the earlier chapter, different heuristic functions and discretization methods have been introduced for space exploration, which are listed in Table 4-1. Together with 4 sampling

methods described in Table 3-1, total 16 variance of space exploration bias sampling can be obtained. Furthermore, the main objective of SE biased sampling is to reduce the minimum samples required to generate near-optimal paths. Which would also reduce the computational cost of the algorithm as it depends on the number of samples. However, the failure rate would increase with the reduced sample size. Therefore to find a balance among these two effects, the sample to volume ratio γ is also added as a parameter to the configurations. In this work, 9 values of γ are considered, varying in log range from 10^{-2} to 10. So a configuration would define the method to generate the tunnel (space exploration method), the probability distribution over the tunnel (sampling method), and the sample to volume ratio γ which will determine the number of samples.

Number	Space Exploration Parameters
1	Heuristic Function : Max(Euclid, Curve) Discretization Method : All Angle
2	Heuristic Function : RS length Discretization Method : All Angle
3	Heuristic Function : Max(Euclid, Curve) Discretization Method : Limited Angle
4	Heuristic Function : RS length Discretization Method : Limited Angle

Table 4-1: Space Exploration Parameters

As the SBMP algorithms use probabilistic approach, to get a reliable estimate of the performance, every problem is computed 100 times. Distribution of samples would only influence the failure rate and cost of the solution, not the computation time, as it only depends on the number of samples. Therefore, to find the most beneficial configuration of SE bias sampling, failure rate and path cost has been considered as an evaluation metric. First, we will examine the failure rate of all the configurations of SE bias sampling with one another. Then, we will compare the path cost computed by different configurations of SE bias sampling with uniform sampling.

4.5.1 Failure Rate Evaluation

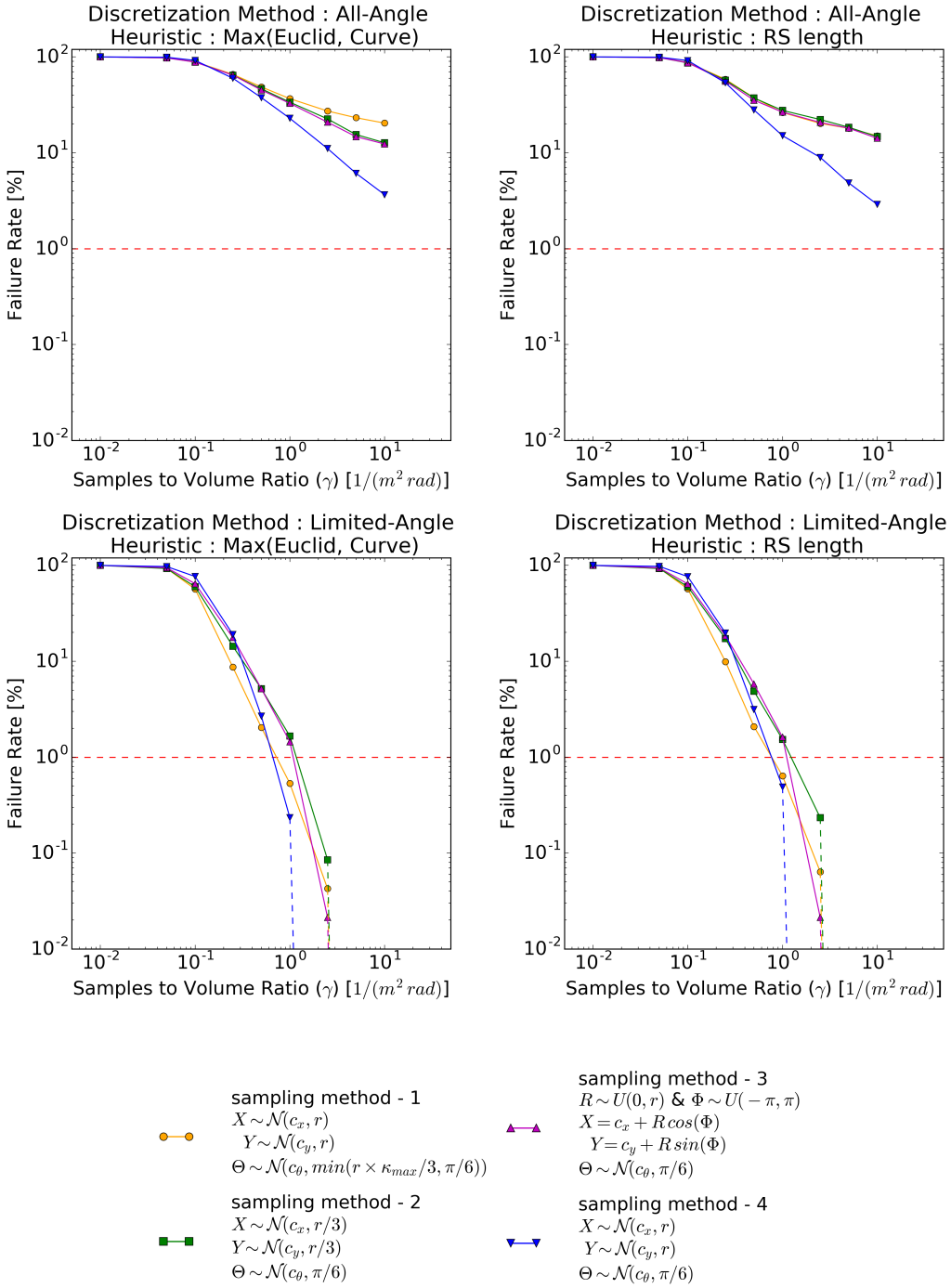


Figure 4-5: The average failure rate of the benchmark problem set plotted against samples to volume ratio γ. The different variance of SE biased sampling is grouped by tunnel computation method. The dotted red line represents the failure threshold.

Figure 4-5 illustrates the variation in the average failure rate of the benchmark problem set with the sample to volume ratio (γ), clustered by space exploration method.

As expected, the failure rate decreases with a higher number of samples. However, the modified discretization (Limited angle) scheme shows significantly good results compared to the original discretization (All angle) scheme. This result confirms our hypothesis that the tunnel computed by All angle discretization may not contain a feasible path for all problems. (Figure 3-4)

Hence only Limited angle discretization will been considered for further evaluation. Furthermore, different sampling distributions and heuristic functions do not show a notable difference in the average failure rate. Nevertheless, with Limited angle discretization, sample to volume ratio γ of 1 $(1/m^2\,rad)$ would result in near 1% failure rate (represented by a dotted red line in Figure 4-5). In this work, the failure rate below 1% is considered as an adequate threshold. However in practice, if a solution is not found with batch sampled, RRT* extension is possible, which can guarantee probabilistic completeness. To further evaluate different sampling methods and space exploration heuristics, it would be helpful to compare the failure rate of individual problems for the sample to volume ratio γ of $1(1/m^2\,rad)$.

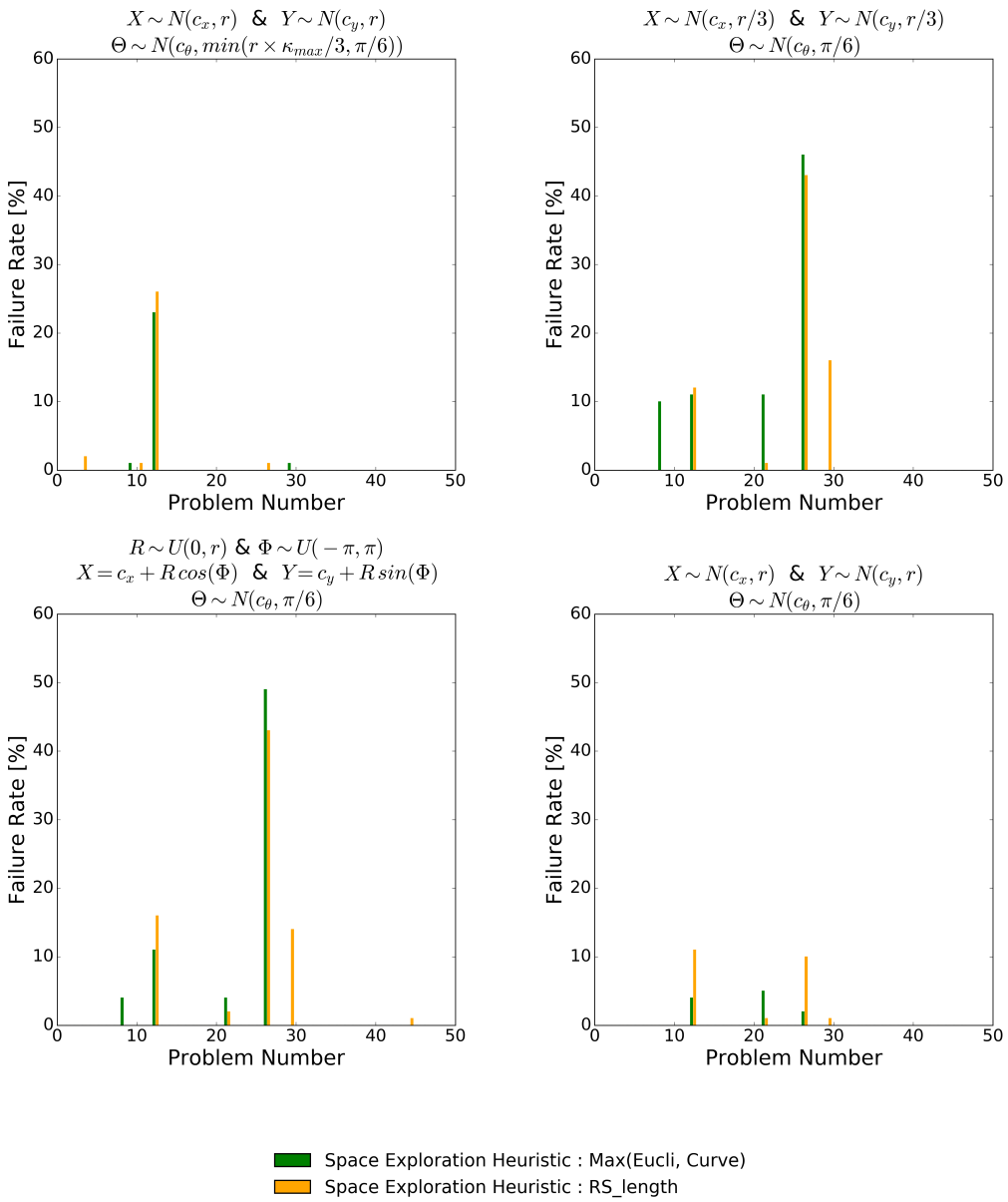
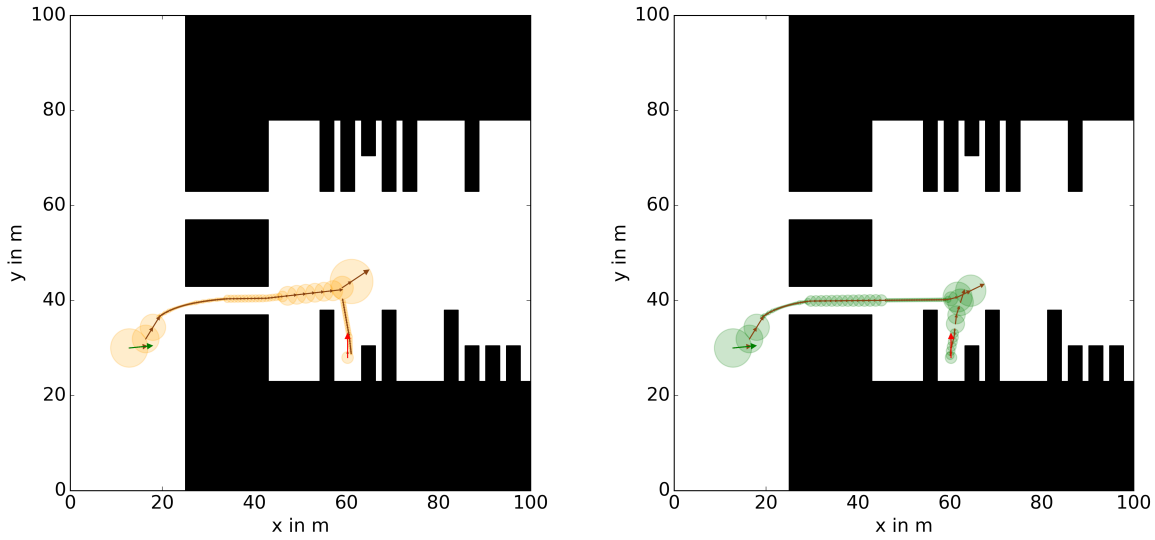


Figure 4-6: The individual failure rate of the benchmark problem set for Limited angle discretization and sample to volume ratio γ of $1(1/m^2\,rad)$.

Figure 4-6 represents the individual failure rate of all 47 problems, clustered by sampling method. There is not so much difference in failure rate between two heuristic functions. Furthermore, it can be deduced that the sampling method 1 and 4 perform better as they have less failure rate across the benchmark problems. However, a peak is visible in sampling method 2 and 3. To closely examine that particular problem, the tunnel computed by both the heuristic functions has plotted in Figure 4-7.



(a) Heuristic Function : Max(Euclid, Curve) **(b)** Heuristic Function : RS length

Figure 4-7: Path planning problem of crossing a narrow region solved using Limited angle discretization of OASE.

Figure 4-7 represents the path planning problem of manoeuvring through a narrow region. The tunnel computed by both the heuristic functions have a very narrow passage and by merely sampling within the tunnel would not result in a feasible solution. Sampling method 1 and 4 utilize normal distribution for position sampling with the standard deviation equal to the radius of the bubble. Therefore, it generates samples not only within the tunnel but also in the surrounding. On the other hand, the other two sampling methods sample positions mostly within the tunnel.

It would be ideal to have the optimal path inside the tunnel, however because of the discretization, only resolution optimal tunnel can be achieved. However, the absence of a feasible solution within the tunnel can be compensated by sampling around the tunnel. Furthermore, SBMP algorithm does the collision check during the tree expansion. Therefore, samples outside the tunnel, possibly in a collision, would be detected during the search.

By evaluating the failure rate of different configurations of SE biased sampling, it has been observed that Limited angle discretization and sampling method 1 and 4 perform better. Hence, these parameters are considered for further evaluation. Next, the path cost of SE biased sampling method is compared with uniform sampling method.

4.5.2 Path Cost Evaluation

In uniform sampling, sample size indicates the sample density as the probability over the whole C space is uniform. Furthermore, in SE biased sampling, sample to volume ratio γ also indicates similar quantity though, it can't be called as sample density, because the probability distribution over the tunnel T is not uniform. Hence for simplicity, while comparing with uniform sampling, path cost and computational time are plotted against the sample size. To compare the path cost of SE biased sampling against uniform sampling, first the problem of reverse parking at the end of the narrow passage defined in Figure 2-1 is considered.

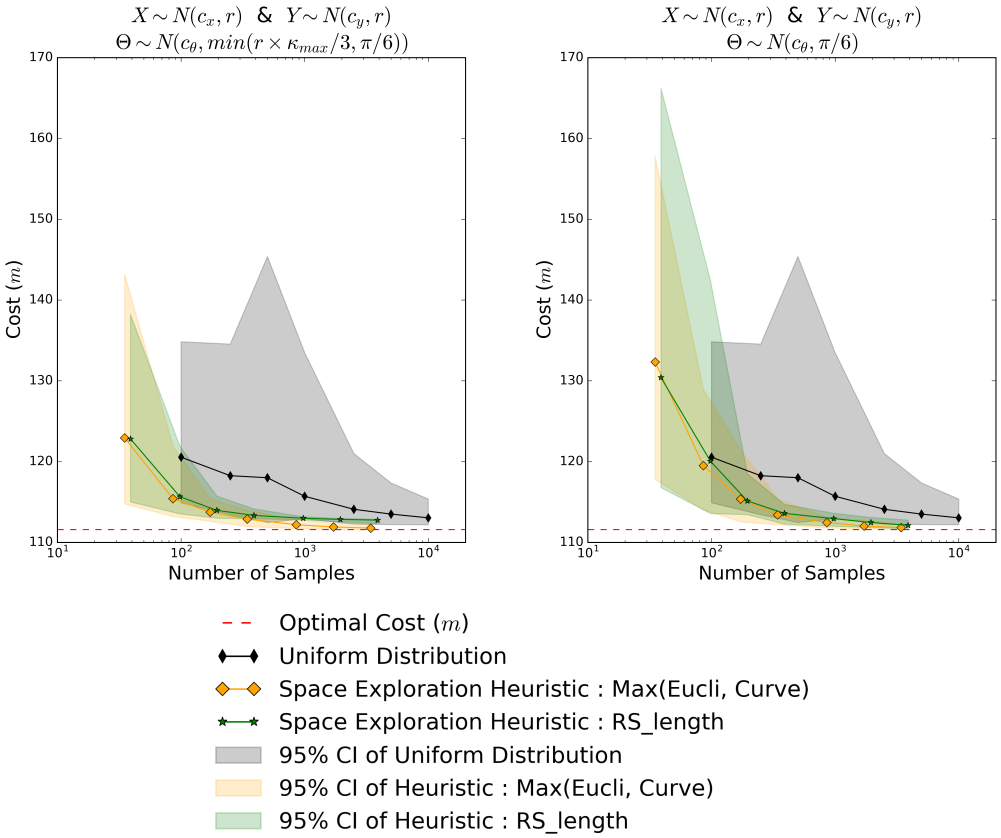


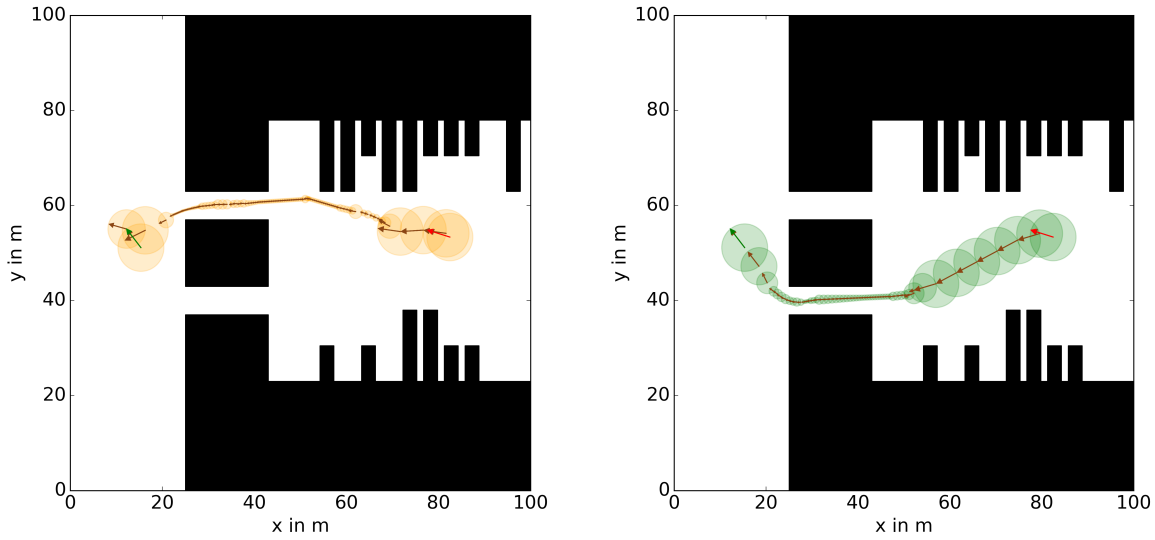
Figure 4-8: Path cost for reverse parking at the end of the narrow passage defined in Figure 2-1.

In Figure 4-8, path cost of different sampling methods is plotted against the number of samples, with 95% confidence interval around the mean value. The optimal cost is represented in a dotted red line. Here, the optimal cost is computed by solving the problem with uniform sampling method and 10^4 number of samples, and subsequently smoothing the path using a path smoothing algorithm. As expected, the path cost would

converge to the optimal value as the number of samples increases. For uniform sampling, failure rate before 10^2 sample size is quite high, therefore the path cost of uniform sampling method starts from 10^2 sample size.

From Figure 4-8, it can be concluded that SE biased sampling method converges to the optimal solution with less number of samples compared to the uniform sampling method. Furthermore, for both heuristic functions, sampling method-1 converges faster to the optimal cost. The only difference between both sampling methods is the standard deviation in the orientational dimension. The sampling method-1 is tightly bounded in the orientational dimension compare to the other method, which can be a reason for this behaviour.

To further evaluate the different heuristic functions, a path planning problem of crossing a narrow region is introduced in Figure 4-9, which has alternative solutions with almost similar path cost.



(a) Heuristic Function : Max(Euclid, Curve) **(b)** Heuristic Function : RS length

Figure 4-9: Path planning problem of crossing a narrow region solved using Limited angle discretization of OASE.

Figure 4-9 illustrates the tunnel computed by both heuristic functions for crossing a narrow region. The algorithms compute alternative paths because of the different heuristic functions and the termination condition for space exploration algorithm. The algorithm would terminate once the goal configuration is overlapping with child vertex, which does

not guarantee optimality. However, from merely visual inspection it is not clear which algorithm performs better. Hence, path cost evaluation is needed.

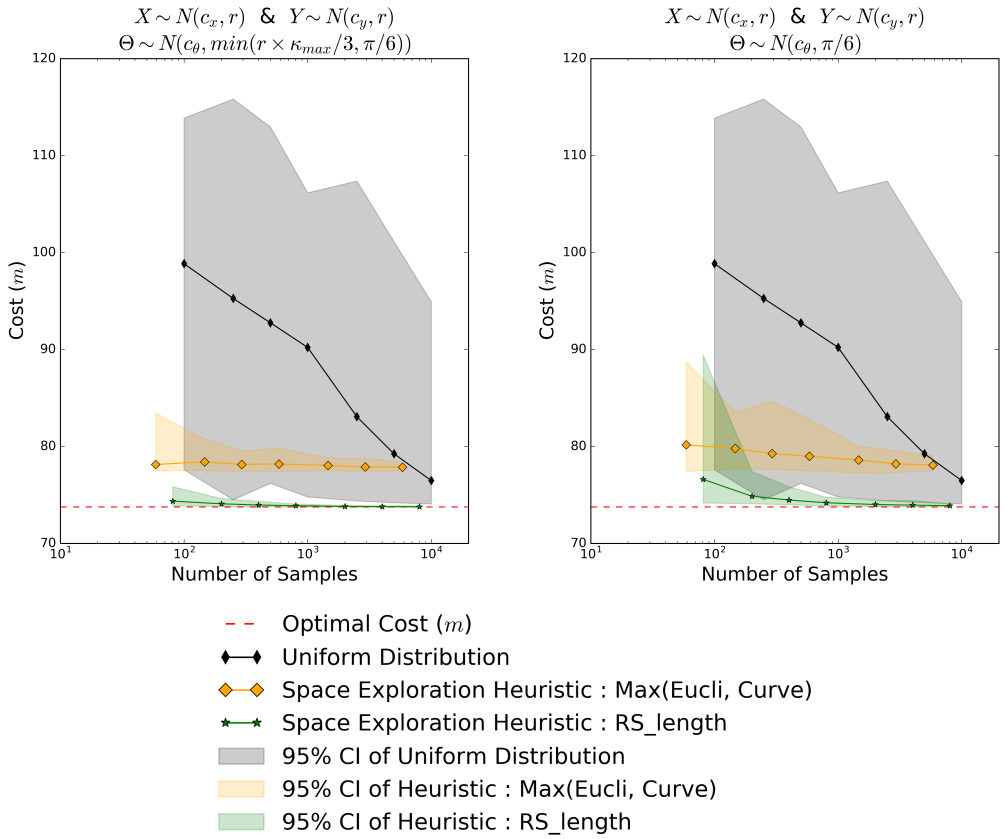


Figure 4-10: Path cost for crossing a narrow region defined in Figure 4-9.

Figure 4-10 concludes that the path cost computed by Max(Euclid, Curve) heuristic converges to a slightly higher cost than the optimal cost, while RS length converges to the optimal cost. Furthermore, similar behaviour with the sampling methods as discussed earlier can be observed.

As mentioned earlier, it is a common practice to compute a solution with construction-based algorithm and then optimize it using modification based algorithm. With the modification based algorithm, it is possible to reach the optimal path if the initial solution is in the global minimum region. If the tunnel computed by space exploration method is in the global minimum region, then a feasible path computed by sampling from this tunnel would also be in the global minimum region. Therefore, the confidence interval of path cost can be overlooked as long as the failure rate is in a given threshold.

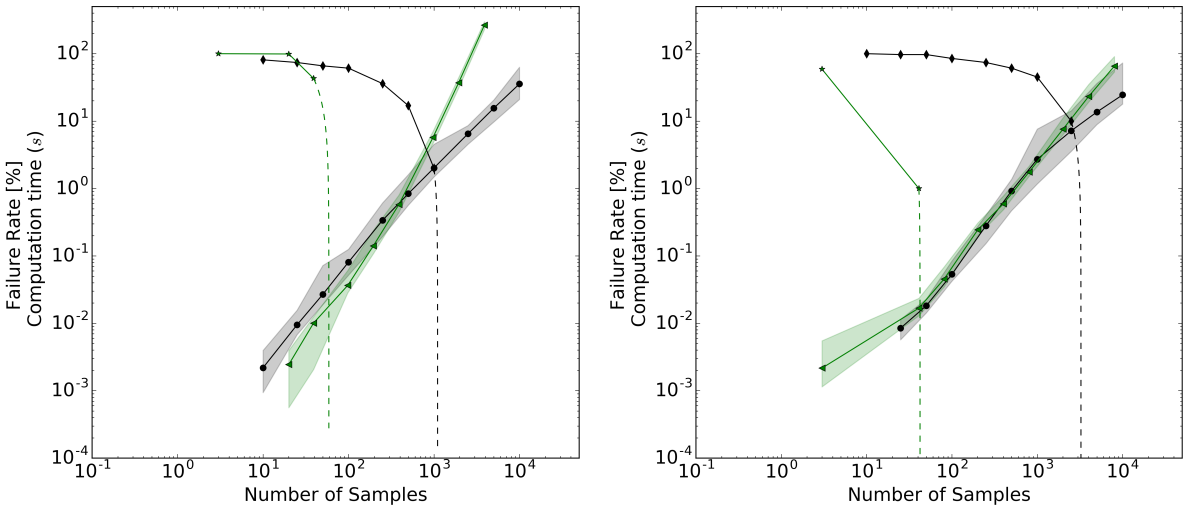
However, paths computed by uniform sampling method can take an alternative path if the

sample density is low. Hence, a large sample size is required for uniform sampling method to obtain a solution in the global minimum region. Therefore, for SE biased sampling the failure rate is a sufficient measure while it is not true for uniform sampling. It has been observed that to obtain a solution in the global minimum region, sample size $N \geq 10^4$ is required for planning problem discussed in this work.

By examining the path cost of SE biased sampling and uniform sampling, it can be concluded that sampling method 1 and RS length heuristic function perform better. Therefore, these parameters are used for comparing the computational cost with uniform sampling. Moreover, it is concluded that for SE biased sampling, the failure rate is an adequate criterion for selecting the sample to volume ratio.

4.5.3 Computation Time Evaluation

The computational complexity of FMT* based algorithm is $O(n \log(n))$, where n is the number of samples. So depending on the number of samples, computational time will increase. To examine the computational reduction, the computational time of both above-mentioned problems have been plotted against sample size.



(a) Reverse parking at the end of the narrow region defined in Figure 2-1

(b) Crossing a narrow region problem defined in Figure 4-9

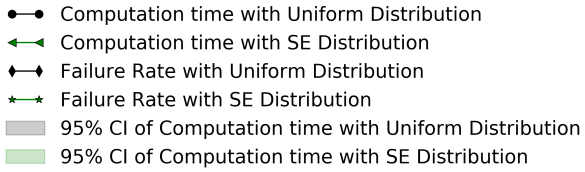


Figure 4-11: Computation time and failure rate versus the number of samples for different sampling methods

Figure 4-11 plots on a log-log scale, mean of computational time with 95% confidence interval for both the problems mentioned earlier (reverse parking at the end of the narrow passage, Figure 2-1 and crossing a narrow region, Figure 4-10). For better visualization, the failure rate is also plotted with it. The dotted line is a quadratic interpolation of the failure rate for the respective methods.

For the reverse parking problem defined in Figure 2-1, to achieve 1% failure rate, SE biased sampling takes around $20\,ms$ by utilizing around 50 samples. While uniform sampling requires more than 10^3 samples to achieve 1% failure rate. As mentioned earlier to attend a path in global optimal region, a large sample size is required. For $N = 10^4$ sample size, uniform sampling method takes more than $50\,s$. Similar results can be observed for the narrow region crossing problem.

From Figure 4-11, it can be observed that the computational time of SE biased sampling increases rapidly compared to the uniform sampling method. For SE biased sampling, the free space volume $\mu(C_{free})$ is approximated by

$$\mu(C_{free}) = (1 - \%_{Rejection\ Rate}) \times vol(T) \tag{4.2}$$

where $vol(T)$ represents the volume of the tunnel and $\%_{Rejection\ Rate}$ represents the rejection rate by collision detection. For SE biased sampling, samples are drawn from a partially free tunnel. Therefore, $\mu(C_{free})$ would be high and so is the cost threshold r (Equation 4.1). Hence, it will consider more samples for connection than the uniform sampling method. So the computational time increases rapidly for SE biased sampling than the uniform sampling method.

4.6 Conclusion

The main objective of SE biased sampling is to reduce the search space of SBMP by computing subspace (tunnel) for sampling. In this work, the performance of SE biased sampling is demonstrated for a car-like vehicle. Moreover, it was observed that parameters, such as discretization scheme and heuristic function, used for computing the tunnel can influence the final result computed by SBMP algorithm. After proposing a few modifications for the parameters (Limited angle discretization scheme and RS length heuristic function), the performance of all available parameters was evaluated. From which, we concluded that proposed modifications performs much better than the original parameters. Furthermore, it was recognised that sampling around the tunnel can compensate for an absence of path inside the tunnel.

Lastly, the computational time required to obtain a feasible path in the global optimal region was examined. It concluded that SE biased sampling requires around $20\,ms$ to compute a path in the global optimal region. However, this computational time does not include the computational time required for computing a tunnel. As mentioned in Section 4.4, computation of tunnel is possible in the range of $20\,ms - 30\,ms$. Therefore, the computation of a feasible path in the global optimal region is possible in $50\,ms$ by optimizing the minimum distance to the obstacles module.

4.7 Future directions

As mentioned earlier, for the practical implementation of SE biased sampling, optimization of minimum distance to the obstacle function is needed. Furthermore, the tunnel T

computed by space exploration method is represented by discrete overlapping bubbles, which increases the sampling probability in the overlapping region. However, as observed earlier, sampling around the tunnel would reduce the failure rate. Sampling from many normal distributions distributed over bubbles can be seen as sampling from a mixture of gaussian. But rather than representing the tunnel as discrete bubbles, continuous representation such as a tube, could improve the sampling method. By connecting the centres of the bubbles with a straight line, a simple path can be constructed, which would be the centerline for the tube. The size of the tube can be interpolated using the bubble size. To sample from the tube, two-step approach similar to the earlier method can be applied. The tube representation would allow efficient development of sampling methods and can improve performance further, which can be a future direction. Moreover, while sampling, the algorithm does not utilize the information gathered over time by samples for further improvement of the sampling distribution, which would be a direction for further investigation.

5. Learned Sampling

Problem specific sampling methods can reduce the search space of SBMP as shown in previous chapters. For a nonholonomic robot, SE biased sampling method is one of the most reliable methods. However, it requires manual algorithm development and fine-tuning of parameters. Moreover, it adds the extra computational cost of estimating the tunnel. The space exploration method tries to approximate the optimal sample set with a tunnel, though, it still requires more than 10 X samples compared to the optimal sample set. For a robot with a different vehicle model, SE biased sampling requires many modifications and lots of effort. This motivates the development of a generic algorithm which provides more flexibility and requires less manual intervention.

Learned sampling method can adapt to a new vehicle model very easily, e.g. In [10], the author showed the CVAE model can generate problem specific samples for a mobile robot and robot manipulator. Moreover, learned sampling method also eliminates the computation of the tunnel as in SE biased sampling.

Problem specific sampling can be interpreted as conditional probability distribution $P(x|y)$, where x would be the samples and y is the problem definition. As mentioned earlier, Conditional variational autoencoder (CVAE) appears the most promising approach, which has been selected for further evaluation. However, before introducing CVAE, it would be helpful to understand the concept of estimating the probability distribution from data, which is called as density estimation. To be consistent with the literature, the same terminology as in [56] is used.

5.1 Density Estimation

If a dataset $\mathcal{D}$ consisting of $N \geq 1$ datapoints, $\mathcal{D} = \{x_1, x_2, ..., x_N\}$, where the datapoint x is a random sample from an unknown underlying probability distribution $p^*(x)$, written as

$$x \sim p^*(x) \tag{5.1}$$

Density estimation is the process of approximating the true underlying probability distribution $p^*(x)$ with a parametric model $p_\theta(x)$, described as

$$p_\theta(x) \approx p^*(x) \tag{5.2}$$

The process of searching parameters θ, which can best approximate the true distribution $p^*(x)$, is commonly referred to as learning. E.g. approximating the distribution of sensor data using a normal distribution. Then, $\mathcal{D}$ would be the sensor dataset and $p_\theta(x)$ is a normal distribution. Using maximum likelihood estimation (MLE), parameters θ (μ and σ) can be approximated.

Oftentimes, we are interested in learning a conditional model $p_\theta(x|y)$ which approximates the underlying conditional distribution $p^*(x|y)$, distribution over the variable x, conditioned on the variable y. E.g. image generation from the description. Then x would be an image and y would be the image description (e.g. text).

$$p_\theta(x|y) \approx p^*(x|y) \tag{5.3}$$

With learned sampling method, the goal is to generate samples that lie near the optimal path, based on a particular planning problem. Hence, the samples can be referred to as x and a finite-dimensional encoding of the planning problem as y. e.g. start-goal configurations and environment description. However to obtain a sufficiently accurate model, more flexible model is required, which is discussed in the next section.

5.2 Conditional Variational Autoencoders

To achieve more flexibility, the parametric model $p_\theta(x|y)$ can be extended using latent variables z. The latent variables or hidden variables z are variables that are not directly observed and hence not part of the dataset $\mathcal{D}$. The conditional distribution using latent variables can be written as $p_\theta(x|z,y)p_\theta(z|y)$ and the marginal distribution over the observed variables $p_\theta(x|y)$, can be written as,

$$p_\theta(x|y) = \int p_\theta(x|z,y)p_\theta(z|y)dz \tag{5.4}$$

By using latent variables, the parametric model $p_\theta(x|y)$ can be described as a product of simple distributions. e.g. To estimate the probability distribution of unconditional multimodel data (House prices around the city), Gaussian mixture model (GMM) is a popular choice. In GMM, $p_\theta(x|z)$ is a Gaussian distribution and z is a discrete variable indicating the index of Gaussian distribution. (e.g. taken from [56])

Recently Conditional Variational Autoencoders (CVAE) which follows a similar concept of the latent variable model has been hugely popular for conditional density estimation problem. CVAE use continuous latent variable z and a neural network (NN) for the parametric model $p_\theta(x|z,y)$. A simple neural network can be understood by Figure 5-1.

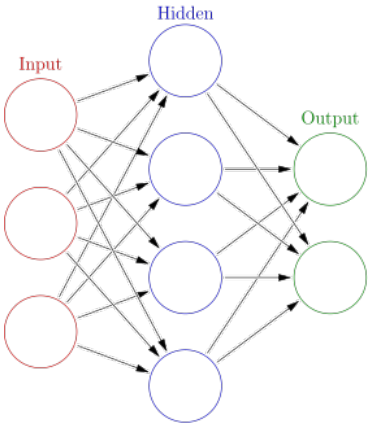


Figure 5-1: A neural network is an interconnected group of nodes, inspired by neurons in a brain. Here, each circular node represents a perceptron and an arrow represents a connection from the output of one perceptron to the input of another. [57]

A perceptron is a nonlinear function and the output of a perceptron is computed by some

non-linear function (activation function) of the weighted sum of its inputs. Here, the weights would be the learning parameters θ. A multilayer perceptron is referred to as a neural network. [57]

To optimize the model, we have to compute the marginal probability of data (Equation 5.4) by integrating over all the values of the latent variable, which is intractable. Therefore, the encoder $q_\phi(z|x,y)$ is used which maps data x to the latent space z. Given a sample $z \sim q_\phi(z|x,y)$, decoder reconstructs the data $\hat{x}$. (for more information, refer [56])

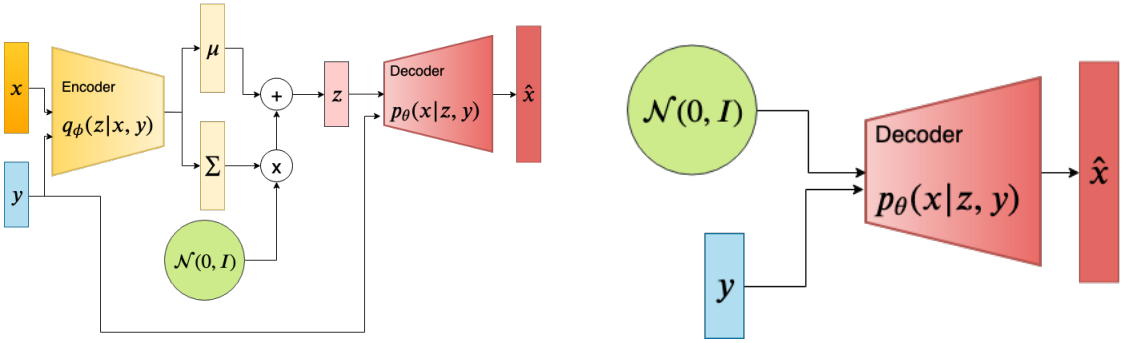
During training, the encoder and decoder are optimized using mini-batch gradient descent to minimize the reconstruction error between original data x and reconstructed data $\hat{x}$, $\mathcal{L}_{rec}(x,\hat{x})$. Moreover, with minimizing Kullback-Leibler (KL) divergence D_{KL} between encoder distribution $q_\phi(z|x,y)$ and prior distribution $p(z|y)$, it ensures data generation by sampling from prior distribution $z \sim p(z|y)$ after training.

$$\mathcal{L}_{cvae} = \mathcal{L}_{rec}(x,\hat{x}) - D_{KL}(q_\phi(z|x,y)||p(z|y)) \tag{5.5}$$

KL-divergence measures the similarity between two probability distributions and measured in "nats" if natural logarithm (base e) is used [58].In this work, $p_\theta(z|y) = \mathcal{N}(0,I)$ and $q_\phi(z|x,y)$ is diagonal-covariance Gaussians $\mathcal{N}(z|\mu(x,y),\Sigma(x,y))$ considered as proposed in [10]. To control the relative weight of the reconstruction error and KL-divergence, a weighting term β is added in practice [59].

$$\mathcal{L}_{cvae} = \mathcal{L}_{rec}(x,\hat{x}) - \beta\, D_{KL}(\mathcal{N}(\mu(x,y),\Sigma(x,y),\mathcal{N}(0,I)) \tag{5.6}$$

The CVAE model can be visualized in Figure 5-2.



(a) Training **(b)** Sampling

Figure 5-2: CVAE architecture [48]. x represents the target variable, y the condition and z the latent variable. The encoder-decoder model (a) is used for training and while sampling only decoder model (b) is used.

Before applying the CVAE model to the actual path planning problem, examining it on a toy problem that though being simplified still describes some characteristics of the path planning problem is advantageous.

5.3 CVAE on a toy problem

A Bézier curve is a parametric curve use for modeling smooth curves in computer graphics [60]. The bézier curve is defined by n points, where n defines the order of the curve (n = 1 for linear, 2 for quadratic, etc.). The first and last points are the endpoints of the curve and intermediate points are the control points, which influence the curvature of the curve. A second-order bezier curve is presented in Figure 5-3

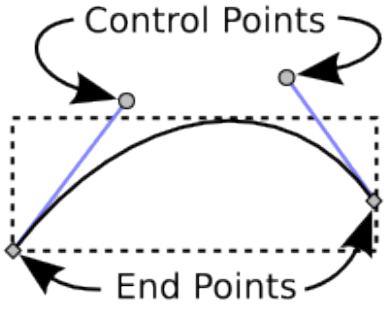


Figure 5-3: Second-order bézier curve [61]

The bézier curve exhibits many similarities with the path planning problem, such as the

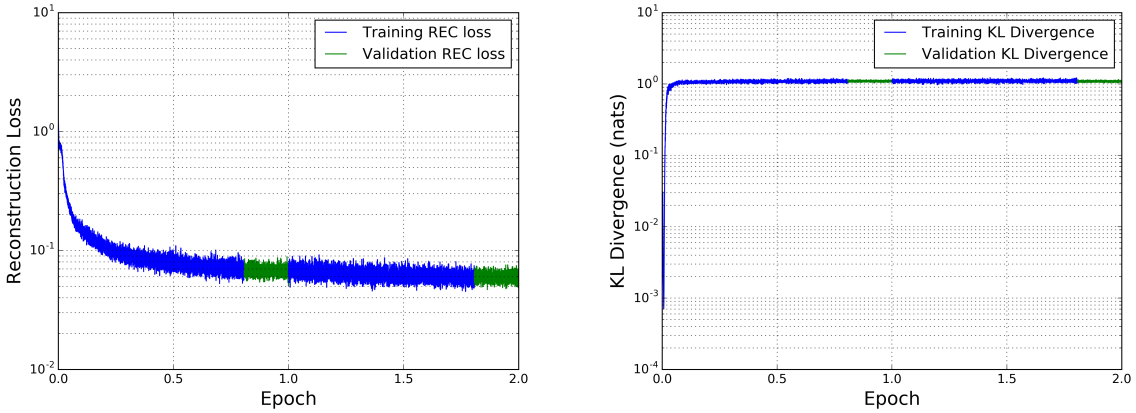
deformation of the curve by control points is similar to the obstacles in the environment. Therefore, bézier curve has been used as a toy problem for CVAE model.

Let, x be a point b on the bézier curve and y would be the end points and control points of the bézier curve. For simplicity, one endpoint is fixed to the origin. Hence, the curve can be defined by two control points p^1 and p^2, and an endpoint p^3. To make a problem more similar to a car-like vehicle, the angle of the tangent at each point is also added to x. The angle is encoded as imaginary and real part as suggested in Complex-YOLO [62]. Then, x and y for the toy problem can be represented as,

$$\begin{aligned} x &= [b_x, b_y, \sin(b_\theta), \cos(b_\theta)] \\ y &= [p_x^1, p_y^1, p_x^2, p_y^2, p_x^3, p_y^3] \end{aligned} \tag{5.7}$$

The encoder and decoder network consist of 5 hidden layers with 128 perceptrons and rectified linear unit (ReLU) as the activation function in between. To fully describe a point on a bézier curve, a parameter t, $0 \leq t \leq 1$, is required other than endpoints and control points. Hence, a single latent variable is enough to pass critical information. However, to understand the behaviour of the CVAE model, size of latent variables is 3 considered.

As the model is trained using mini-batch gradient descent, mean squared error (MSE) of a mini-batch is used for the reconstruction loss $\mathcal{L}_{rec}$ and β is 0.1 considered. An epoch is one cycle through the full training dataset. To train the CVAE model, 48000 bézier curves are used. Figure 5-4 illustrates $\mathcal{L}_{cvae}$ loss of two epoch of the training of toy problem.



(a) Reconstruction loss **(b)** KL divergence

Figure 5-4: Training and validation loss $\mathcal{L}_{cvae}$ during 2 epochs for a toy problem.

Figure 5-4 illustrates reconstruction loss $\mathcal{L}_{rec}$ and Kl-divergence D_{KL} for the toy problem example. At the end of the training, $\mathcal{L}_{rec}$ is around 0.06 and KL-divergence is near 1.0 nats. During the training, fluctuation of the $\mathcal{L}_{rec}$ is in the range of 0.02 and for the D_{KL} fluctuation below 0.5 nats. Therefore, it can be concluded that training is quite stable.

Moreover, it is observed that only one latent variable is active, which means latent variable distribution generated by the encoder differs from $\mathcal{N}(0, I)$ and encodes information from data x. While the other two latent variable collapses to $\mathcal{N}(0, I)$. The active latent variable showed similar characteristics with parameter t. To evaluate the model further, data is generated by sampling $z \sim \mathcal{N}(0, I)$ and passing through the decoder with the condition y. This procedure is referred to as sampling from the model.

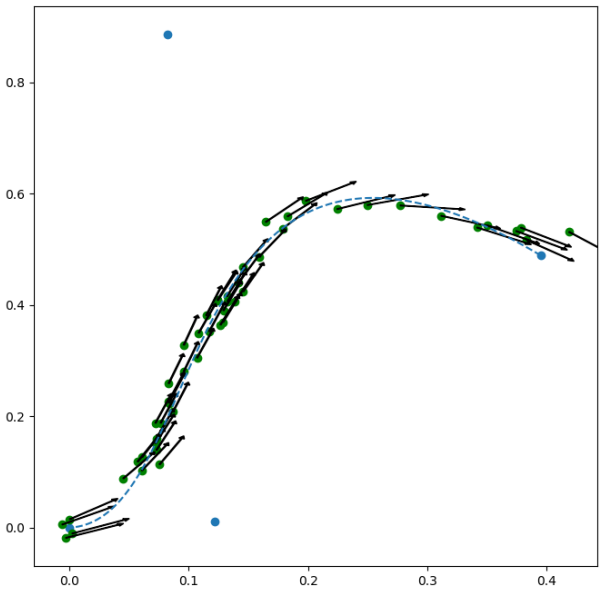


Figure 5-5: Samples (green dot with an arrow representing the tangent) generated by the CVAE model for the toy problem of a bézier curve. The bézier curve (blue dotted line) is defined by endpoints and control points (blue dots).

Figure 5-5 illustrates the bézier curve defined by endpoints and control points, and samples generated by CVAE model. The generated data $\hat{x}$ (green dots) appear very close to the a bézier curve and the tangent (represented by an arrow) also appears to follow the true tangent. Successful results on the toy problem motivate to proceed further with the CVAE model, to develop problem-specific sampling method.

6. Experimental Results of Learned Sampling

Ideally, we want our learned sampling model to work out of the box on any problem definition (extrinsic property) and any robot model (intrinsic property). However, to initialize in the field, the development is restricted to consider variation in only extrinsic properties and fix intrinsic property to a car-like vehicle. Before applying the learned sampling method on the complex parking problem, it would be helpful to understand the behaviour of the CVAE model on a simplified problem.

In this work, only 4 axis-aligned (with workspace) rectangle obstacles of varying size and position in the environment are considered. This simplification would allow encoding of obstacles in a neural network easier. However, for further development, the environment can be represented as a grid as proposed in [10]. Here, obstacles are denoted as o^i where i represents the index. A obstacle is defined by its centre (c_x and c_y) and size (s_x and s_y). Other than environment simplification, everything else is identical to Chapter 4.

As stated previous, for learned sampling method x would be the problem-specific sample and y would be the problem definition. The Reed Shepp car can move in forward as well as backwards direction. Initial development suggested that adding moving direction information ($d \in \{-1, 1\}$, where -1 and 1 are backwards and forward direction respectively) in the sample x can improve the performance.

Let p, s and g be a problem-specific sample pose, start pose and goal pose respectively, then a sample x and a condition y can be described as following,

$$x = [p_x, p_y, \sin(p_\theta), \cos(p_\theta), p_d] \tag{6.1}$$

$$\begin{aligned} y = [&s_x, s_y, \sin(s_\theta), \cos(s_\theta), \\ &g_x, g_y, \sin(g_\theta), \cos(g_\theta), \\ &o^1_{c_x}, o^1_{c_y}, o^1_{s_x}, o^1_{s_y}, \ldots \quad , o^4_{s_y}] \end{aligned} \tag{6.2}$$

6.1 Training data

For training the CVAE model, the optimal path is computed using the uniform sampling method followed by a smoothing step. To have a diverse training set, 8000 different environments are generated and 160000 paths are used for training. The main objective of the learned sampling method is to generate critical samples for a particular problem. Therefore, it is important to train the model with such samples. For a car-like vehicle, the pose where vehicle switches direction is particularly important from the entire path. Other than that, near obstacle poses and optimal sample set as defined in Section 2.6 are considered for training. Figure 6-1 illustrates two such paths used for training.

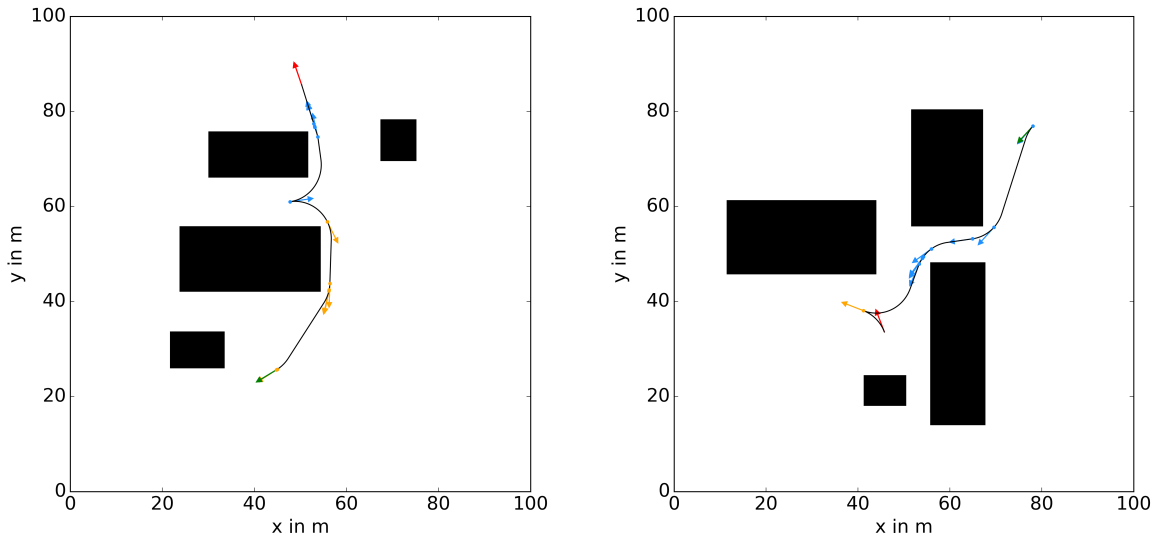


Figure 6-1: Training data for CVAE. Obstacles represented by black rectangles. Start pose and goal pose is represented by green and red arrow respectively. Blue (Forward direction) and orange (Backwards direction) arrows represent the problem-specific samples used training for CVAE.

6.2 CVAE Architecture

The CVAE architecture is similar to the one used in toy problem, with 10 hidden layers and 10 latent variables. From initial development, it has been observed that the encoder $q_\phi(z|x, y)$ is not capable of encoding information only using diagonal-covariance Gaussians distribution. Therefore, to improve the flexibility of encoder, Normalizing Flow (NF) [63] is utilized here. A flow is an invertible nonlinear transformations.

The general idea of using NF with CVAE is as follows: first, a latent variable is drawn from a simple distribution, $\mathcal{N}(z_0|\mu(x,y),\Sigma(x,y))$ in this work. Then, it is transformed with several flows. After applying K flows, the final latent variables are given by $z_K = f_K \circ ...f_2 \circ f_1(z_0)$. In [63], the authour proposed planner flow, which is decribed by

$$z' = z + uh(w^T z + b) \tag{6.3}$$

Where, $u, w \in \mathbb{R}^D, b \in \mathbb{R}$ and h is tanh. For $u^T w \geq -1$, the flow is invertible. To produce input depending flow parameters (u, w, b), hypernet [64] is used as proposed in [65]. A neural network use to generate the parameters for another network is referred to as a hypernet. In this work, the hypernet is a one layer linear network and produces flow parameters from the last hidden layer of the encoder, which can be visualized in Figure 6-2. However, adding NF would not change the sampling method as described in Figure 5-2. From here on, the combination of NF and CVAE is referred to as a model.

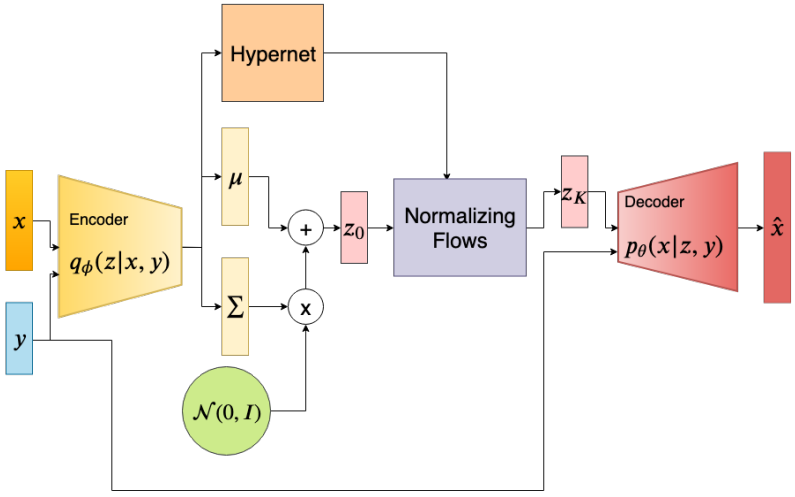


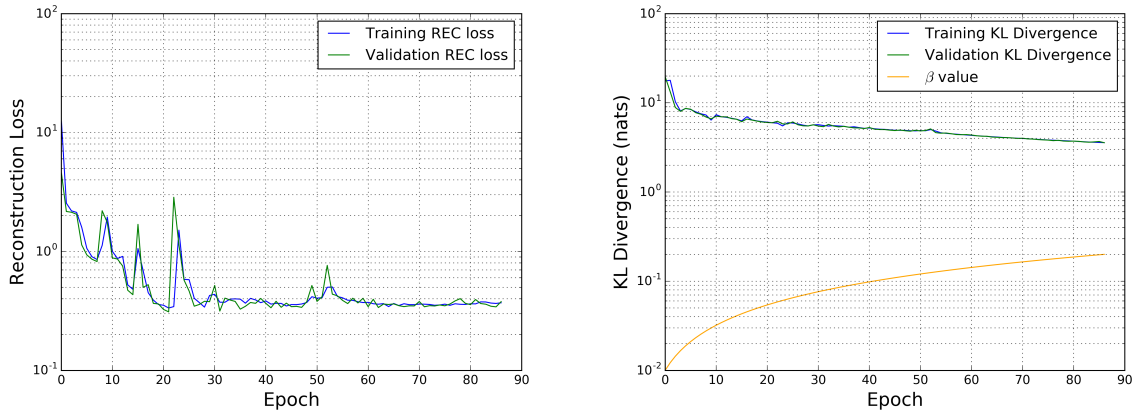
Figure 6-2: CVAE architecture with a Normalizing flow.

In [66], the author proposed to start the training with a small β value. This will allow $q_\phi(z|x,y)$ to encode the data efficiently in the initial phase. Then gradually increase β to reduce the KL-divergence loss. From initial evaluation, it has been observed that varying β linearly from 0.01 to 0.2 during training results in better performance.

As described earlier, mean squared error (MSE) is considered for reconstruction loss $\mathcal{L}_{rec}$. However, for relative weight between the orientation information $(\sin, \cos \in [-1, 1])$ and the translational information $(x, y \in [0, 100])$, the translational information is scaled (by a factor of $1/10$). The scaling procedure is similar to the weighted mean squared error.

6.3 Evaluation results

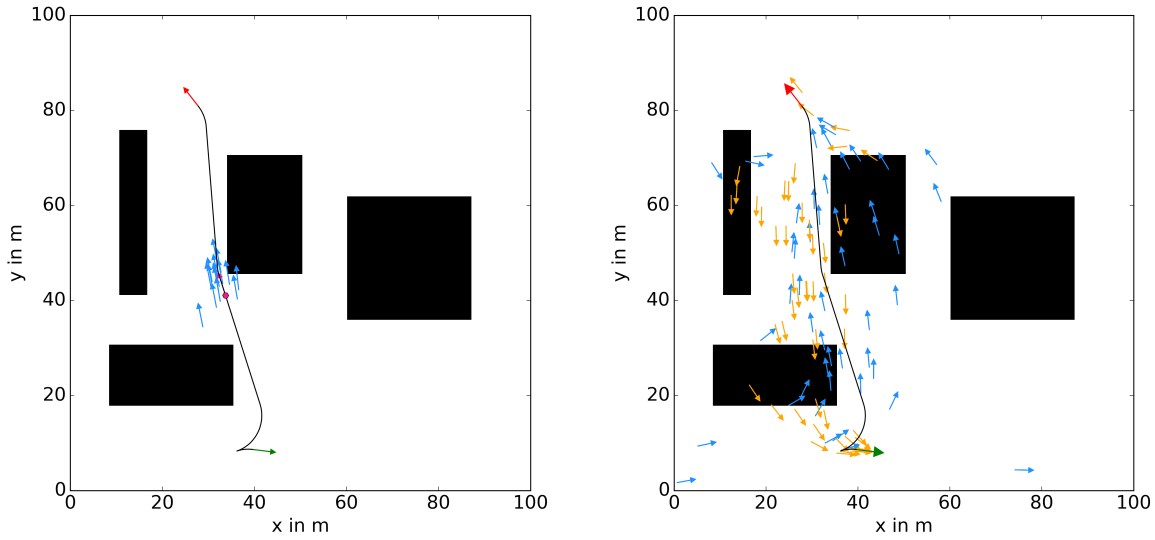
Before evaluating the model for learned sampling in SBMP, the loss values during training are examined. Figure 6-9 illustrates training and validation loss during training.



(a) Reconstruction loss **(b)** KL divergence

Figure 6-3: Training and validation loss $\mathcal{L}_{cvae}$ during 85 epochs for a simplified path planning problem.

Figure 6-9 shows no notable difference between training and validation loss, which indicates that the model is able to generalize on unseen data. Moreover, the reconstruction loss is around 0.5 and Kl divergence is below 3.0 nats at the end of the training, which is quite high compared to the toy problem. To further evaluate the model, the reconstruction of a sample and random samples generated by the model is presented.



(a) Reconstruction (blue arrows) of the original pose (pink arrow) on the path.

(b) Problem specific samples generated by the model.

Figure 6-4: Problem specific samples (blue and orange arrows, which represent forward and backwards direction respectively) generated using learned sampling method.

In Figure 6-4a, reconstruction $\hat{x}$ (blue arrows) of a sample x (pink arrow) can be visualize. As the encoder generates a distribution over latent space followed by a sampling step, a small variation can be observed in reconstructed poses. However, variation in the position is larger than in the orientational dimension (p_θ). The relative weighting of the orientation information and translational information can be considered a reason for high variation in the position of samples.

Figure 6-4b illustrates problem-specific samples generated by the model. It can be observed that the model generates many samples around the pose where path switches the direction of movement, which is a significant pose from the whole path. It is clear that the model is able to recognise start and goal pose, and generates sample in a forward-moving direction from start to goal and opposite for backwards direction, which became possible by adding moving direction information d.

However, it can be observed that while reconstructing and sampling the problem-specific samples, the model does not consider obstacles. For problem defined in Figure 6-4, around 40% of generated samples are in collision.

In this work, only path length as the optimization objective is considered. Therefore, the

optimal path passes very near to the obstacles. By analysing the loss function, it is clear that the loss function does not penalize the model for generating in-collision samples. The reconstruction loss forces the model to replicate the original data. However, the training data is quite near to the obstacles. Moreover, Gaussian distribution in latent space adds some noise, which causes in-collision reconstruction samples $\hat{x}$.

If the distance to the obstacles is included in the optimization objective, paths would have clearance distance to obstacles. By training the model with such kind of paths would generate collision free $\hat{x}$ after some noise as well. More results of generated samples by the model is presented in Appendix C.

Nevertheless, a mere visual inspection could not conclude the performance of the model. Therefore, the learned sampling method was tested for 37 problems.

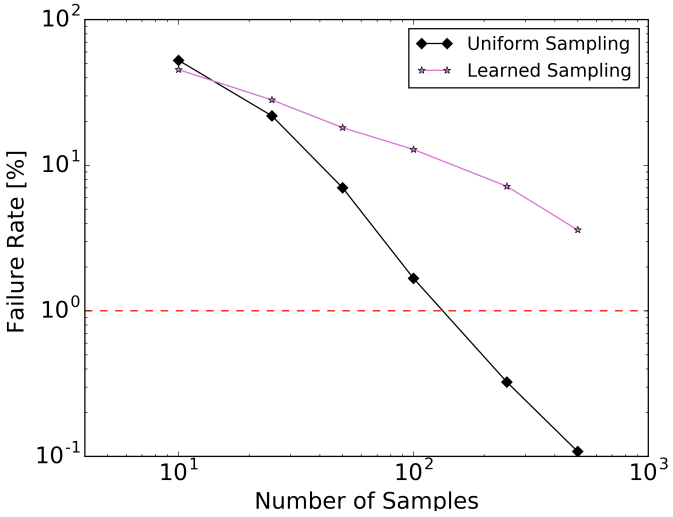
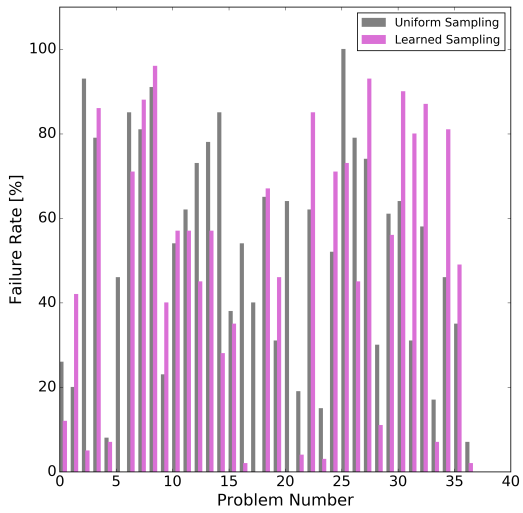
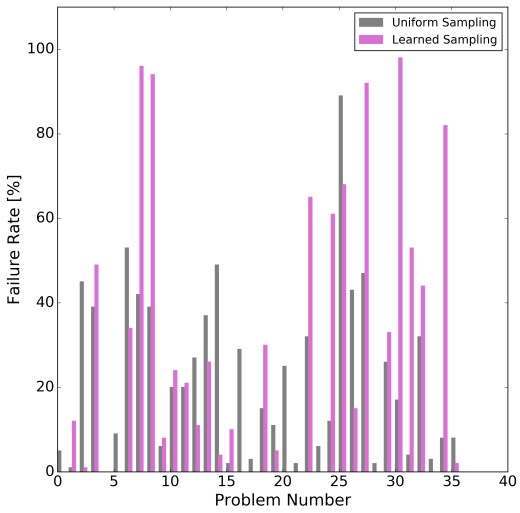


Figure 6-5: The average failure rate of 37 problems for uniform sampling method and learned sampling method plotted against the number of samples.

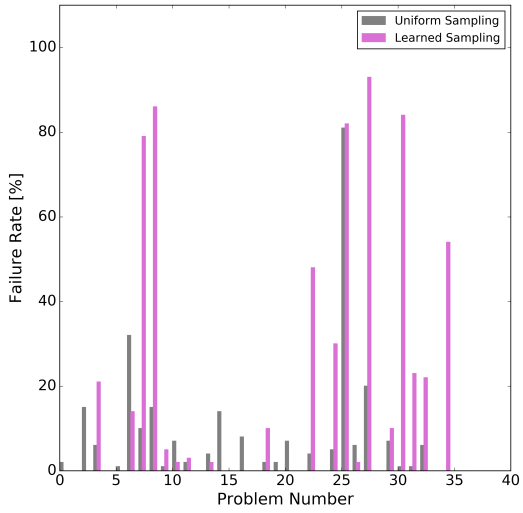
Figure 6-5 illustrates the failure rate of learned sampling method and uniform sampling method. It can be observed that the average failure rate drops quickly for uniform sampling method compare to learned sampling method. Therefore, it can be concluded that uniform sampling method surpasses the model used for learned sampling method. To understand the weakness of the model, it would be helpful to examine which kind of problems are difficult for the model.



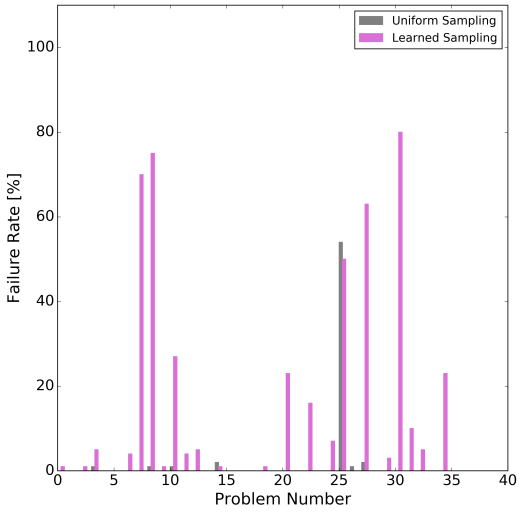
(a) Number of samples $N = 10$



(b) Number of samples $N = 25$



(c) Number of samples $N = 50$



(d) Number of samples $N = 100$

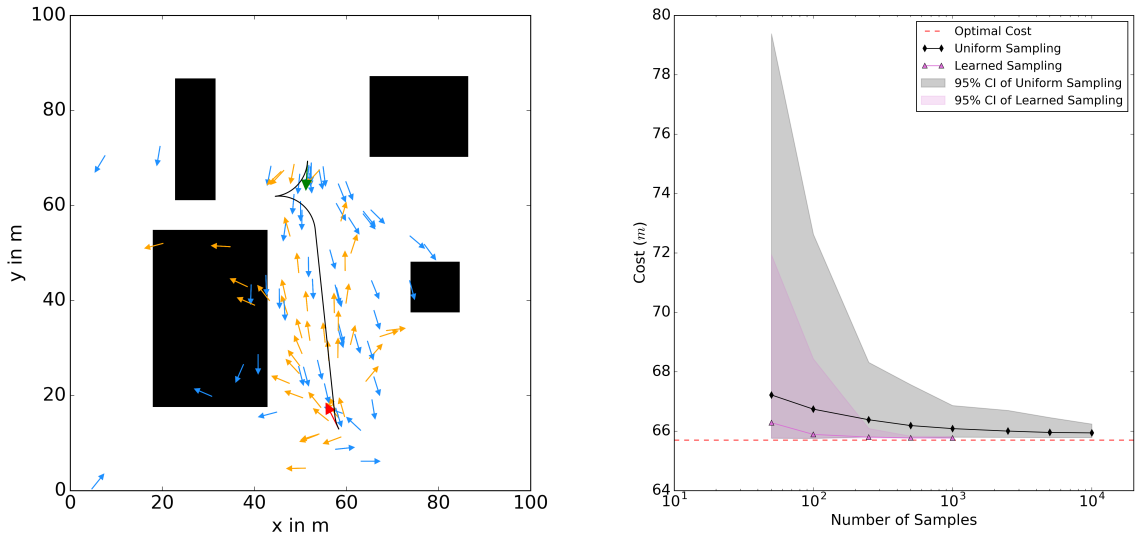
Figure 6-6: The failure rate of individual problems for learned sampling method and uniform sampling method plotted against problem number for different sample size.

Figure 6-6 illustrates the failure rate of individual problems for different sample size N. Initially ($N = 10$) both the sampling methods have almost same failure rate. However, with an increase in sample size, the failure rate for learned sampling method of some problems reduce only a bit. Lastly, for sample size $N = 100$, the failure rate of uniform

sampling method is almost zero, while for some problems it is quite high for learned sampling method.

To understand problems more closely, the entire benchmark problem set is divided into three categories by considering the failure rate of the learned sampling method. An easy problem if the failure rate of learned sampling method converges faster than uniform sampling method. A medium problem if both performs almost similar and A hard problem if learned sampling method has a large failure rate. The path cost for solvable problems (easy and medium problems) by learned sampling method is examined to measure whether learned sampling method converges to an optimal solution as sample size increases.

For better visualization, the problem definition, optimal path and problem-specific samples generated by the model are presented on the right side. Furthermore, on the left side, path cost against sample size for uniform sampling method and learned sampling method with 95% confidence interval and optimal cost represented in the dotted red line are plotted.

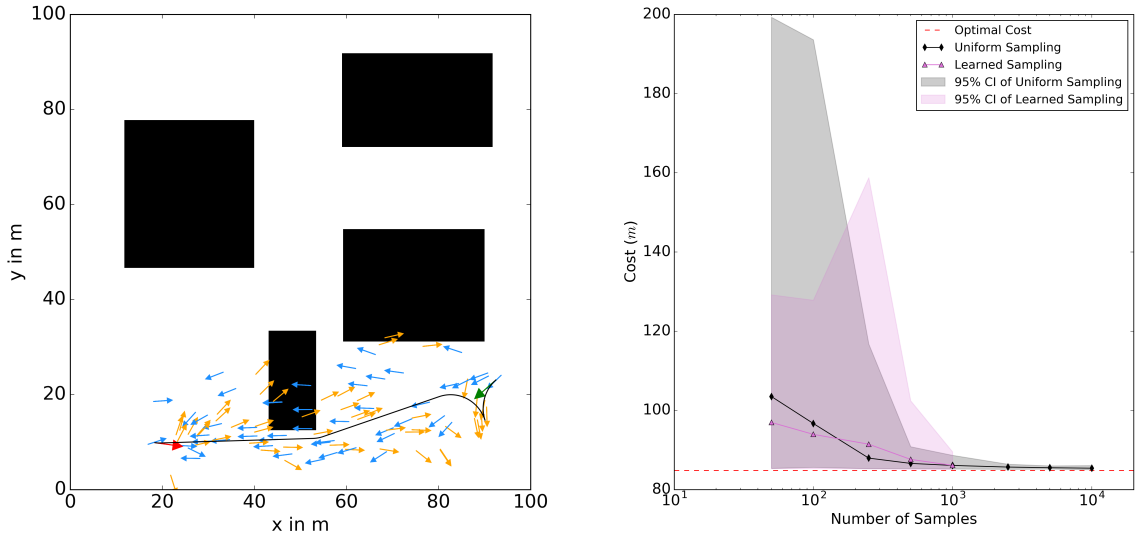


(a) Path planning problem **(b)** Path cost

Figure 6-7: Easy path planning problem.

Figure 6-7a present an easy path planning problem, where the model performs much better than the uniform sampling method. Although the problem is not directly solvable by connecting start and goal pose with reed sheep curve, the optimal path is not bypassing any obstacles. Many samples generated near the switching direction pose of the path can

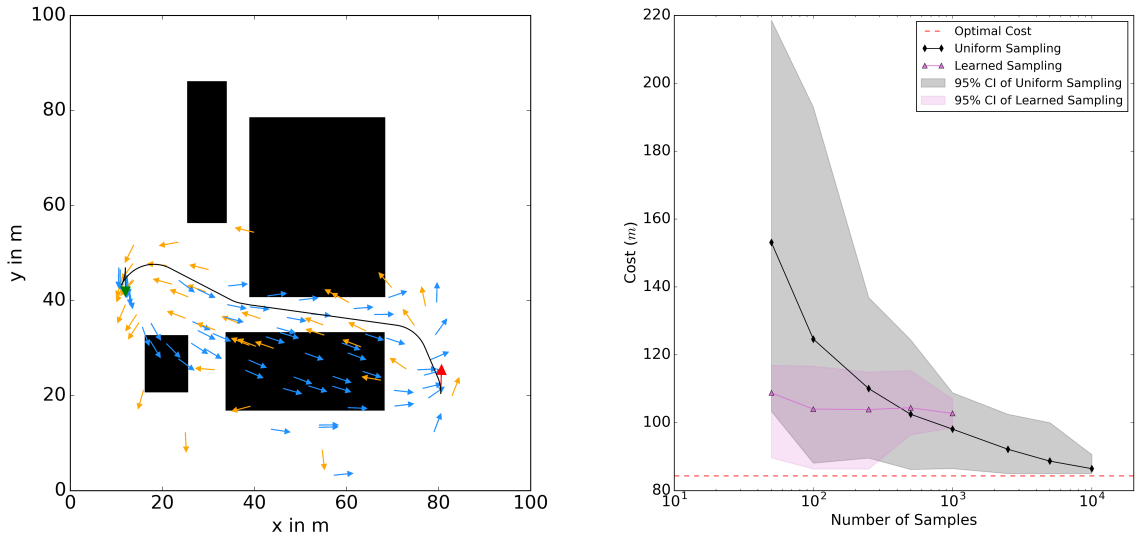
be observed. From path cost comparison, it is clear that the learned sampling method converges to optimal cost faster than the uniform sampling method. Therefore, it can be concluded that the model can learn vehicle model from the samples and performs better than the uniform sampling method in problems where it does not have to bypass obstacles. Next, we will increase the problem complexity for a path cost comparison.



(a) Path planning problem **(b)** Path cost

Figure 6-8: Medium path planning problem.

Figure 6-8a illustrates a little difficult problem (medium problem) than the earlier problem. By observing the variation in path cost, it can be concluded that although learn sampling method converges to the optimal solution, but requires more samples compare to uniform sampling. Next, the model is examined for a path planning problem of crossing a narrow region similar to Figure 4-9.



(a) Path planning problem **(b)** Path cost

Figure 6-9: Medium path planning problem.

Figure 6-9a represents the problem of planning through a narrow tunnel. It is interesting that learn sampling method has very low confidence interval at the beginning compared to the uniform sampling method, however, learn sampling method does not converge to the optimal solution with increasing sample size.

The failure rate of learned sampling method is high for hard problems. Therefore, most of the time solution path is not available and hence path cost comparison is not possible. Therefore, samples generated by the model for hard problems are presented.

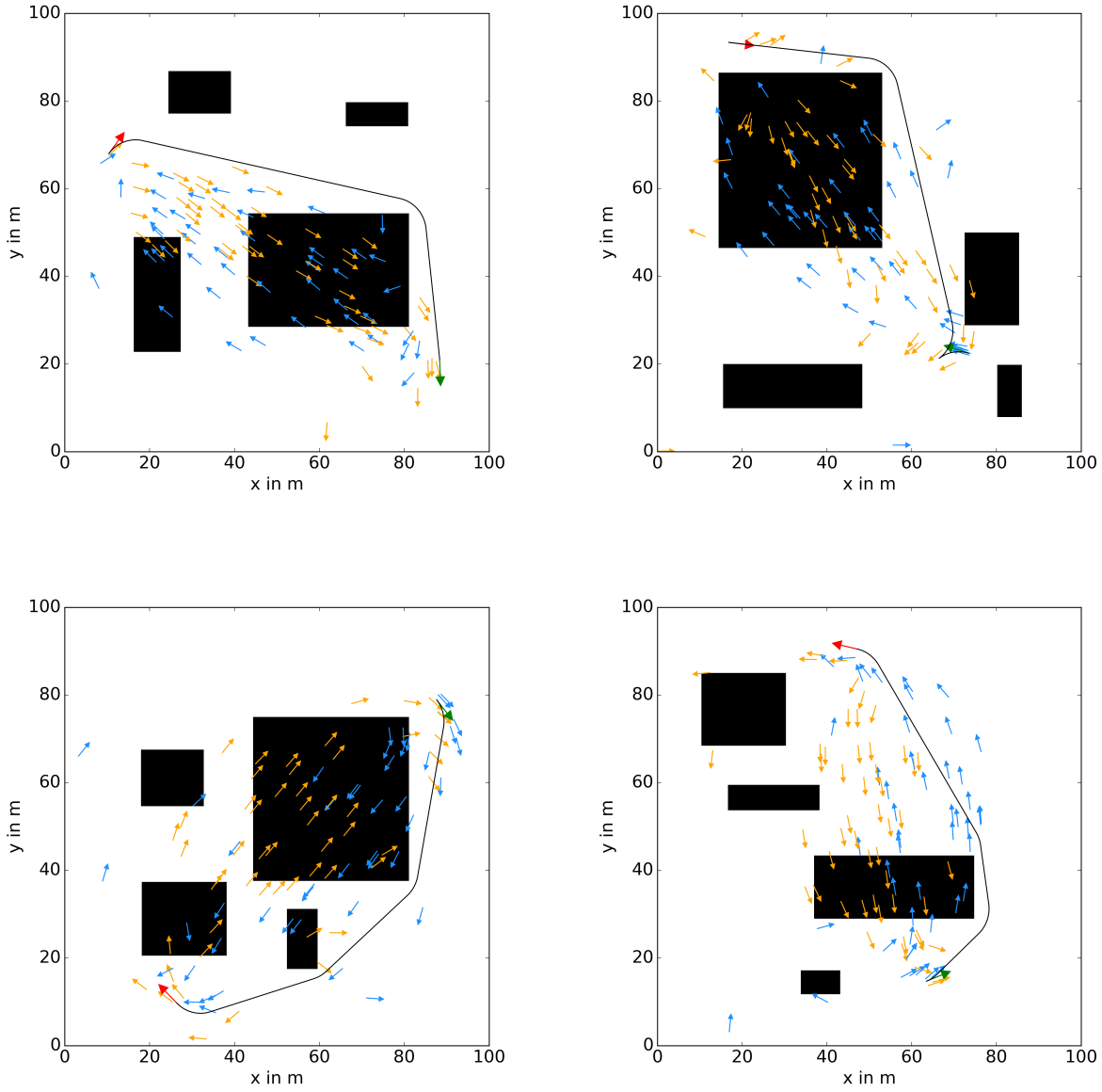


Figure 6-10: Samples generated by the model for hard problems

For all the hard problem, a large obstacle can be found between the start and the goal pose. From Figure 6-10, it is clear that the model is not able to generate the samples by avoiding the obstacles. Hence, the large failure rate of learned sampling method for hard problems can be understood.

6.4 Conclusion

The main objective of the learned sampling method in the context of this work is to develop a generic problem-specific sampling method which works for a car-like robot and can be extended to any robotic system. In this work, a conditional variational autoencoder model has been used for learned sampling method and some modifications have been proposed for better performance. However, after testing the model against the uniform sampling method, the model performs worse on average than the uniform sampling method. Though the model is able to capture the vehicle model from the data and it performs better than uniform sampling method when the optimal path is not bypassing the obstacles.

However, in the presence of obstacles between start and goal pose, the model is generating samples in the forbidden region. In my understanding, the model is generating samples around the Reed Shepp curve joining the start and the goal pose. If the obstacle between start and goal pose is larger than the randomness which model is producing around the Reed Shepp curve, then the learned sampling method is not able to find a feasible path. Next, we will discuss a few reasons for failure and propose some extension for better performance.

The main purpose for forcing posterior distribution $q_\phi(z|x, y)$ to has a minimum distance to prior distribution $p(z|y)$ with KL-divergence is that later we can generate samples $\hat{x}$ by sampling $z \sim p(z|y)$. In this work, KL-divergence D_{KL} is computed between latent variable distribution $q_\phi(z|x, y)$ generated by each sample x and prior distribution $p(z|y)$. However, for better sampling, overall latent variable distribution computed by all training data should match with a prior distribution, not latent variable distribution computed by individual data. This approach was proposed in Wasserstein Auto-Encoders [67] and I believe that it could improve the performance of the model.

In this work, planner flows are utilized for increasing the flexibility of encoder. However, by closely examine Equation 6.3 of the planner flow, we can observe that it would effectively act as a single perceptron. Recently, more flexible Normalizing flow, Sylvester Normalizing Flows, which does not have a single-neuron bottleneck, have been proposed in [65], which could improve performance.

Normalizing flows were placed after the encoder network to make the encoding distribution more flexible. However, the prior distribution is a unit Gaussian and does not depend on condition y, which is a poor assumption. Hence, the KL divergence would force latent variable distribution generated by each sample x to match with a unit Gaussian, which

reduces the flexibility of the model. A conditional prior network proposed in [68] with a conditional normalizing flow proposed in [69] can generate problem-specific multimodel prior distribution $p(z|y)$, which could result in better performance. Lastly, training paths which have some clearance distance to the obstacles can provide better performance.

6.5 Future directions

With the CVAE model, the approach was to learn the problem-specific sample distribution from training data and then utilize the model to generate problem-specific samples. However, recently the problem of generating problem-specific samples was reformulated as a Reinforcement learning problem which does not require training data [70]. Furthermore, the latent variable model can convert the problem from configuration space to latent space, where it is comparatively easy to solve. This approach was utilized in [71] for computing a path using the SBMP algorithm in latent space, which could be a future research direction.

Bibliography

[1] Google. (n.d.), "Google maps image for dhl paketzentrum köln-west am eifeltor," 2019. [Online; accessed 15-October-2019].

[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[3] L. C. Wang, L. S. Yong, and M. H. Ang, "Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment," in *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*, pp. 821–826, IEEE, 2002.

[4] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[5] C. Chen, M. Rickert, and A. Knoll, "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1148–1153, IEEE, 2015.

[6] H. Banzhaf, P. Sanzenbacher, U. Baumann, and J. M. Zöllner, "Learning to predict ego-vehicle poses for sampling-based nonholonomic motion planning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1053–1060, 2019.

[7] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, December 2012. `http://ompl.kavrakilab.org`.

[8] S. Quinlan, *Real-time modification of collision-free paths*. No. 1537, Stanford University Stanford, 1994.

[9] C. Chen, *Motion planning for nonholonomic vehicles with space exploration guided heuristic search*. PhD thesis, Technische Universität München, 2016.

[10] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7087–7094, IEEE, 2018.

[11] F. Esposto, J. Goos, A. Teerhuis, and M. Alirezaei, "Hybrid path planning for nonholonomic autonomous vehicles: An experimental evaluation," in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 25–30, IEEE, 2017.

[12] S. LaValle, "Motion planning: The essentials," *Robotics and Automation Magazine, IEEE*, vol. 18, pp. 79–89, 03 2011.

[13] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific journal of mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[14] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2427–2433, IEEE, 2009.

[15] D. Kiss and G. Tevesz, "Autonomous path planning for road vehicles in narrow environments: an efficient continuous curvature approach," *Journal of advanced transportation*, vol. 2017, 2017.

[16] C. Liu, C.-Y. Lin, Y. Wang, and M. Tomizuka, "Convex feasible set algorithm for constrained trajectory smoothing," in *2017 American Control Conference (ACC)*, pp. 4177–4182, IEEE, 2017.

[17] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3231–3237, IEEE, 2005.

[18] M. Likhachev, "Search-based planning with motion primitives,"

[19] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[21] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[22] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.

[23] J. Starek, E. Schmerling, L. Janson, and M. Pavone, "Bidirectional fast marching trees: An optimal sampling-based algorithm for bidirectional motion planning," in *Workshop on Algorithmic Foundations of Robotics*, 2014.

[24] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

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

[26] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[27] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using rrt," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1681–1686, IEEE, 2008.

[28] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for nonholonomic dynamical systems," in *2013 IEEE International Conference on Robotics and Automation*, pp. 5041–5047, IEEE, 2013.

[29] E. Schmerling, L. Janson, and M. Pavone, "Optimal sampling-based motion planning under differential constraints: the driftless case," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2368–2375, IEEE, 2015.

[30] R. Geraerts and M. H. Overmars, "Sampling techniques for probabilistic roadmap planners," 2004.

[31] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[32] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *2003 IEEE international conference on robotics and automation (cat. no. 03CH37422)*, vol. 3, pp. 4420–4426, IEEE, 2003.

[33] L. Palmieri, S. Koenig, and K. O. Arras, "Rrt-based nonholonomic motion planning using any-angle path biasing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2775–2781, IEEE, 2016.

[34] Y. Wang, D. K. Jha, and Y. Akemi, "A two-stage rrt path planner for automated parking," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pp. 496–502, IEEE, 2017.

[35] Y. Silveira and P. Alsina, "A new robot path planning method based on probabilistic foam," in *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, pp. 217–222, IEEE, 2016.

[36] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966–984, 2018.

[37] T. Kunz, A. Thomaz, and H. Christensen, "Hierarchical rejection sampling for informed kinodynamic planning in high-dimensional spaces," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 89–96, IEEE, 2016.

[38] D. Yi, R. Thakker, C. Gulino, O. Salzman, and S. Srinivasa, "Generalizing informed sampling for asymptotically-optimal sampling-based kinodynamic planning via markov chain monte carlo," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7063–7070, IEEE, 2018.

[39] B. Burns and O. Brock, "Toward optimal configuration space sampling.," in *Robotics: Science and Systems*, pp. 105–112, 2005.

[40] J. Huh and D. D. Lee, "Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 63–69, IEEE, 2016.

[41] Wikipedia contributors, "Discriminative model — Wikipedia, the free encyclopedia," 2019. [Online; accessed 3-November-2019].

[42] N. Pérez-Higueras, F. Caballero, and L. Merino, "Learning human-aware path planning with fully convolutional networks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–5, IEEE, 2018.

[43] D. Molina, *Identifying Critical Regions for Robot Planning Using Convolutional Neural Networks*. PhD thesis, Arizona State University, 2019.

[44] Wikipedia contributors, "Generative model — Wikipedia, the free encyclopedia," 2019. [Online; accessed 3-November-2019].

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[46] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[47] T. Barbi and T. Nishida, "Trajectory prediction using conditional generative adversarial network," in *2017 International Seminar on Artificial Intelligence, Networking and Information Technology (ANIT 2017)*, Atlantis Press, 2017.

[48] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in neural information processing systems*, pp. 3483–3491, 2015.

[49] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: Towards real-time planning forrobots with many degrees of freedom," tech. rep., 2000.

[50] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning," in *Algorithmic Foundations of Robotics X*, pp. 365–380, Springer, 2013.

[51] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 835–842, IEEE, 2015.

[52] Boost, "Boost C++ Libraries." `http://www.boost.org/`, 2015. Last accessed 10-09-2019.

[53] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, "Python-robotics: a python code collection of robotics algorithms," *arXiv preprint arXiv:1808.10703*, 2018.

[54] V. Varricchio, B. Paden, D. Yershov, and E. Frazzoli, "Efficient nearest-neighbor search for dynamical systems with nonholonomic constraints," *arXiv preprint arXiv:1709.07610*, 2017.

[55] E. F. Schmerling, *Multimodal Modeling and Uncertainty Quantification for Robot Planning and Decision Making*. PhD thesis, Stanford University, 2019.

[56] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *arXiv preprint arXiv:1906.02691*, 2019.

[57] Wikipedia contributors, "Artificial neural network — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=924892664`, 2019. [Online; accessed 7-November-2019].

[58] Wikipedia contributors, "Kullback–leibler divergence — Wikipedia, the free encyclopedia," 2019. [Online; accessed 23-October-2019].

[59] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-vae," *arXiv preprint arXiv:1804.03599*, 2018.

[60] Wikipedia contributors, "Bézier curve — Wikipedia, the free encyclopedia," 2019. [Online; accessed 29-October-2019].

[61] Tavmjong Bah, "Inkscape: Guide to a vector drawing program, bezier curves," 2019. [Online; accessed 29-October-2019].

[62] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds," in *European Conference on Computer Vision*, pp. 197–209, Springer, 2018.

[63] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," *arXiv preprint arXiv:1505.05770*, 2015.

[64] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[65] R. v. d. Berg, L. Hasenclever, J. M. Tomczak, and M. Welling, "Sylvester normalizing flows for variational inference," *arXiv preprint arXiv:1803.05649*, 2018.

[66] C. Li, "Less pain, more gain, anyone? this vae trainer alleviates kl vanishing," Jun 2019.

[67] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," *arXiv preprint arXiv:1711.01558*, 2017.

[68] O. Ivanov, M. Figurnov, and D. Vetrov, "Variational autoencoder with arbitrary conditioning," *arXiv preprint arXiv:1806.02382*, 2018.

[69] A. Bhattacharyya, M. Hanselmann, M. Fritz, B. Schiele, and C.-N. Straehle, "Conditional flow variational autoencoders for structured sequence prediction," *arXiv preprint arXiv:1908.09008*, 2019.

[70] R. Madaan, S. Zeng, B. Okorn, and S. Scherer, "Learning adaptive sampling distributions for motion planning by self-imitation," 2018.

[71] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.

List of Figures

List of Figures

Figure 1-1 Bird view of DHL Paketzentrum at Köln-West. Image is taken from Google Maps. [1] 6

Figure 2-1 The global path (blue) for a path planning problem of reverse parking at the end of a narrow region defined by a start (green arrow), goal pose (red arrow) and obstacles represented by the black region. 10

Figure 2-2 The car-like vehicle A: $R = (x, y)$ is the reference point and θ is the main orientation. ϕ is the steering angle and L is the wheelbase. [2] 12

Figure 2-3 Typical Motion Planning Methods as proposed in [16] 13

Figure 2-4 Path planning for a car-like robot using state lattice algorithm. [18] . . 14

Figure 2-5 An iteration of the FMT* algorithm. [22] 17

Figure 2-6 The optimal problem-specific sample set (blue poses) to find the path between start pose (green) and goal pose(red) 20

Figure 2-7 Theta* RRT approach for a car-like vehicle. S and G represent the start pose and goal configuration respectively. The any-angle path of Theta* (in red) is used to guide the samples. The motion tree is represented in blue. [33] 21

Figure 2-8 Free subspace computed using Probabilistic Foam Method (PFM) [35]. . 21

Figure 2-9 Space exploration biased sampling as proposed in [6]. The subspace is visualized by white circles on the ground and the samples are represented by grey arrows. The final path is represented by a green curve. 22

Figure 2-10 Illustration of the informed set with changing best cost from (a) to (c) as proposed in [36]. Path heuristic represents a tunnel around the current best path as subspace for sampling while the omniscient set describes a subset which can provide samples for a better solution, which can be approximated by L^2 informed set. 23

Figure 2-11 The architecture of the CNN proposed in [6] for learned sampling method. It predicts a two-dimensional sampling distribution over future vehicle positions (top right) and the heading angle (bottom right). The five input girds, which describe the current driving situation, are blended into two images on the left. 25

Figure 2-12 A fast marching tree (FMT*) generated using learned sampling method for a double integrator system proposed in [10]. The problem-specific samples are generated using the initial state (red circle), goal region (blue circle), and workspace obstacles (black). Moreover, purple, red and green dots represent samples from V_{closed}, V_{open} and $V_{unvisited}$ respectively. The green line represents the final path computed by FMT*. 26

Figure 3-1 A free space bubble for a planar robot represented by a double-sided cone as proposed in [8]. The bubble is computed by utilizing the minimum distance d from the obstacles. 28

Figure 3-2 The bubble discretization proposed in OASE [5] (bird view). The black circle represents the bubble β_q around configuration q (black arrow). The green and red arrows represent the discretization of free space surface with the outward and inward motion direction respectively. 30

Figure 3-3 Space exploration of OASE method for a path planning problem defined by start pose (red), goal pose (green) and obstacles (gray). [9] 31

Figure 3-4 Path planning problem of reverse parking at the end of a narrow region defined by a start (green arrow) and goal (red arrow) pose. The tunnel is computed by OASE method and the bird view of the tunnel is represented by circles (blue). The brown arrow represents the configuration q of the bubble β_q. The OASE algorithm tries to switch the moving direction after entering the narrow region which is not feasible. A zoomed-in section is provided for better visualization in the right. 32

Figure 3-5 Modified bubble discretization (bird view). The black circle represents the bubble β_q around configuration q (black arrow). The green and red arrows represent the discretization of free space surface with the outward and inward motion direction respectively. 34
Figure 3-6 Planning problem of reverse parking at the end of a narrow region solved using modified discretization method and both heuristic functions. The circles represent the bird view of the bubbles and the arrow represents the centre configuration. 35
Figure 3-7 The translational probability distribution over free space path corridor for different values of α in equation 3.8 37

Figure 4-1 Complex Parking Environments. 40
Figure 4-2 Illustration of the robot's body (green rectangle), Axis Aligned Bounding Box around the robot (blue rectangle), the obstacles (polygons in black) in the environment and Axis Aligned Bounding Box around obstacles (dotted black rectangle) 41
Figure 4-3 Reeds-Shepp Curves with $\kappa_{max} = 0.127$[53] 42
Figure 4-4 Illustration of motion tree (pink) generated by connecting samples (green arrow). The solution path (black) computed by the SBMP algorithm and a smooth path (blue). 44
Figure 4-5 The average failure rate of the benchmark problem set plotted against samples to volume ratio γ. The different variance of SE biased sampling is grouped by tunnel computation method. The dotted red line represents the failure threshold. 46
Figure 4-6 The individual failure rate of the benchmark problem set for Limited angle discretization and sample to volume ratio γ of $1(1/m^2\,rad)$. 48
Figure 4-7 Path planning problem of crossing a narrow region solved using Limited angle discretization of OASE. 49
Figure 4-8 Path cost for reverse parking at the end of the narrow passage defined in Figure 2-1. 50
Figure 4-9 Path planning problem of crossing a narrow region solved using Limited angle discretization of OASE. 51
Figure 4-10 Path cost for crossing a narrow region defined in Figure 4-9. 52
Figure 4-11 Computation time and failure rate versus the number of samples for different sampling methods 54

Figure 5-1 A neural network is an interconnected group of nodes, inspired by neurons in a brain. Here, each circular node represents a perceptron and an arrow represents a connection from the output of one perceptron to the input of another. [57] 59
Figure 5-2 CVAE architecture [48]. x represents the target variable, y the condition and z the latent variable. The encoder-decoder model (a) is used for training and while sampling only decoder model (b) is used. 61
Figure 5-3 Second-order bézier curve [61] 61
Figure 5-4 Training and validation loss $\mathcal{L}_{cvae}$ during 2 epochs for a toy problem. . . 62
Figure 5-5 Samples (green dot with an arrow representing the tangent) generated by the CVAE model for the toy problem of a bézier curve. The bézier curve (blue dotted line) is defined by endpoints and control points (blue dots). . . 63

Figure 6-1 Training data for CVAE. Obstacles represented by black rectangles. Start pose and goal pose is represented by green and red arrow respectively. Blue (Forward direction) and orange (Backwards direction) arrows represent the problem-specific samples used training for CVAE. 65
Figure 6-2 CVAE architecture with a Normalizing flow. 66
Figure 6-3 Training and validation loss $\mathcal{L}_{cvae}$ during 85 epochs for a simplified path planning problem. 67
Figure 6-4 Problem specific samples (blue and orange arrows, which represent forward and backwards direction respectively) generated using learned sampling method. 68
Figure 6-5 The average failure rate of 37 problems for uniform sampling method and learned sampling method plotted against the number of samples. . . . 69
Figure 6-6 The failure rate of individual problems for learned sampling method and uniform sampling method plotted against problem number for different sample size. 70
Figure 6-7 Easy path planning problem. 71
Figure 6-8 Medium path planning problem. 72
Figure 6-9 Medium path planning problem. 73
Figure 6-10 Samples generated by the model for hard problems 74
Figure 0-11 The tunnel T computed using different variations of space exploration method. 91
Figure 0-12 The tunnel T computed using different variations of space exploration method. 92

Figure 0-13 The tunnel T computed using different variations of space exploration method. 93
Figure 0-14 The tunnel T computed using different variations of space exploration method. 94
Figure 0-15 The tunnel T computed using different variations of space exploration method. 95
Figure 0-16Path planning problem defined by start pose (green) and goal pose (red). The black curve represents the optimal solution. Samples (blue and orange arrow) generated by the model (Number of samples = 100). blue is forward direction and orange is reverse direction. 96
Figure 0-17Path planning problem defined by start pose (green) and goal pose (red). The black curve represents the optimal solution. Samples (blue and orange arrow) generated by the model (Number of samples = 100). blue is forward direction and orange is reverse direction. 97
Figure 0-18Path planning problem defined by start pose (green) and goal pose (red). The black curve represents the optimal solution. Samples (blue and orange arrow) generated by the model (Number of samples = 100). blue is forward direction and orange is reverse direction. 98

List of Tables

List of Tables

Table 3-1 Three-dimensional probability distributions over a bubble for sampling. . 38

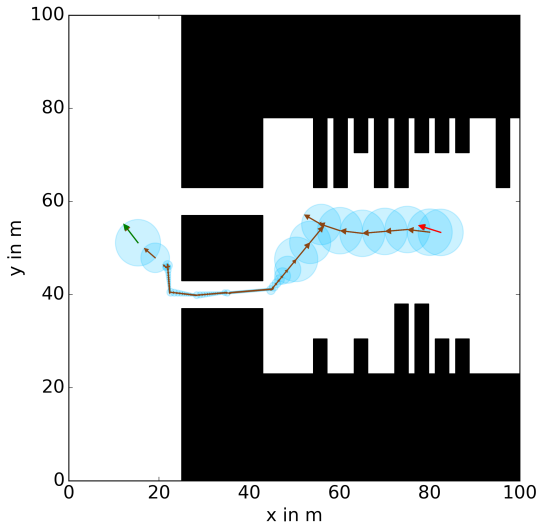
Table 4-1 Space Exploration Parameters 45

A. Orientation-Aware Space Exploration pseudocode

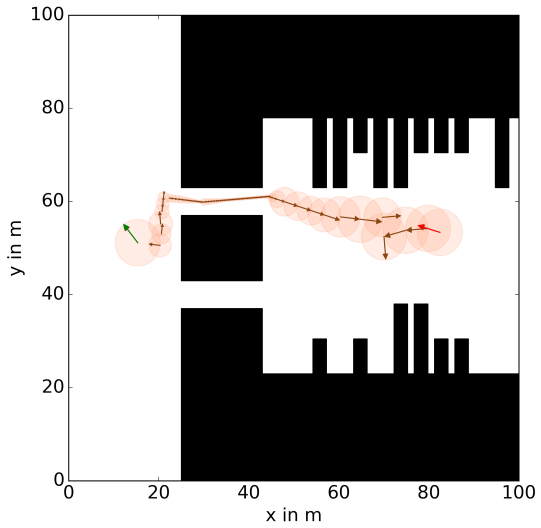
Algorithm 1 Orientation-Aware Space Exploration

$v_{start} \leftarrow (p_{start}, d_{clearance}(p_{start}), 0, d_{heuristic}(p_{start}, p_{goal}))$
$f_{goal} \leftarrow \infty$
$V_{closed} \leftarrow \emptyset$
$V_{open} \leftarrow \{v_{start}\}$
$E \leftarrow \emptyset$
while $V_{open} \neq \emptyset$ **do**
    $Sort(V_{open})$
    $v_i \leftarrow PopTop(V_{open})$
    if $f_{goal} < f[v_i]$ **then**
        $break$
    else if $Exist(v_i, V_{closed})$ **then**
        $continue$
    else
        $(V_i, E_i) \leftarrow Expand(v_i)$
        $V_{open} \leftarrow V_{open} \cup V_i$
        $E \leftarrow E \cup E_i$
        $V_{closed} \leftarrow V_{closed} \cup \{v_i\}$
        if $Overlap(v_i, p_{goal})$ **then**
            $f_{goal} = min(f[v_i], f_{goal})$
if $f_{goal} < \infty$ **then**
    return $success$;
else
    return $failure$

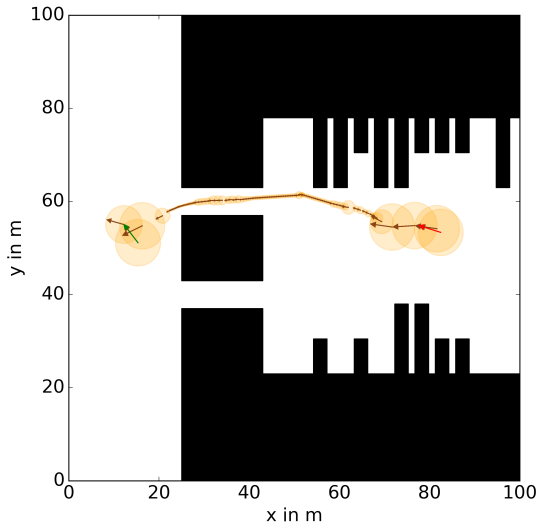
B. Space Exploration results for complex problems



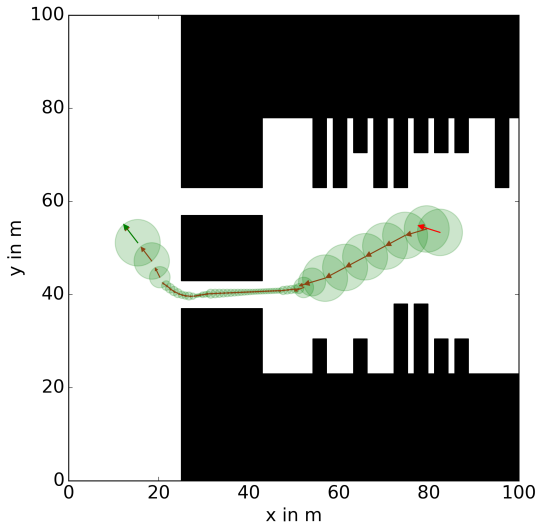
(a) Heuristic Function : Max(Euclid, Curve)
Discretization Method : All Angle



(b) Heuristic Function : RS length
Discretization Method : All Angle

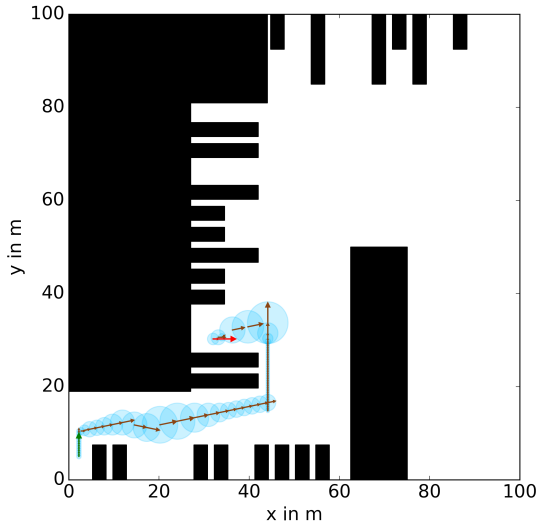


(c) Heuristic Function : Max(Euclid, Curve)
Discretization Method : Limited Angle

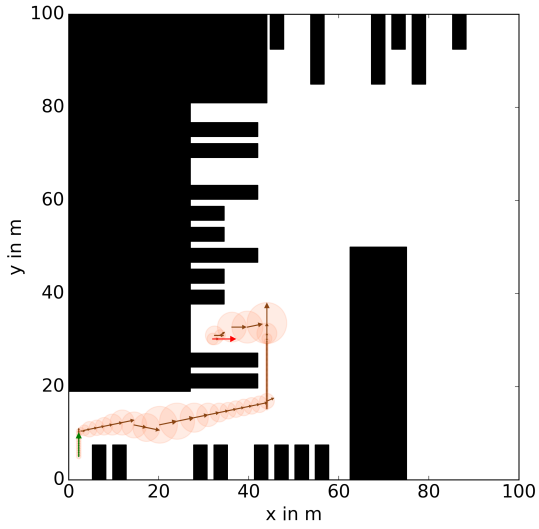


(d) Heuristic Function : RS length
Discretization Method : Limited Angle

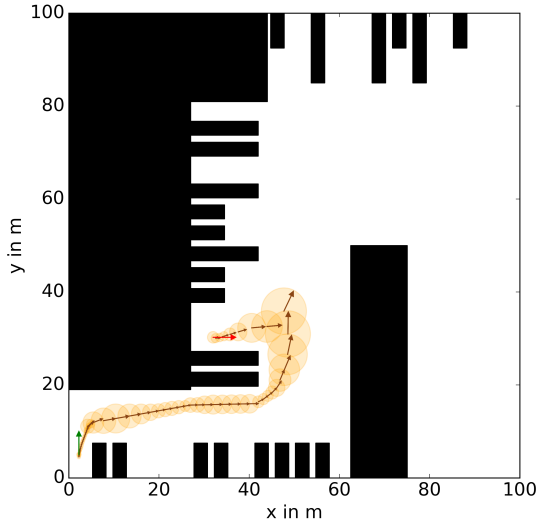
Figure 0-11: The tunnel T computed using different variations of space exploration method.



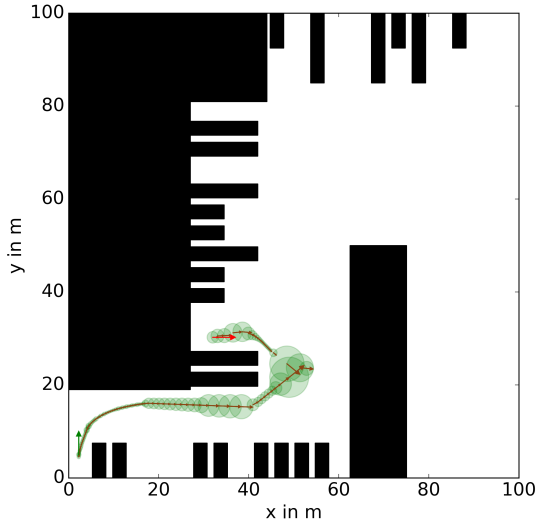
(a) Heuristic Function : Max(Euclid, Curve)
Discretization Method : All Angle



(b) Heuristic Function : RS length
Discretization Method : All Angle

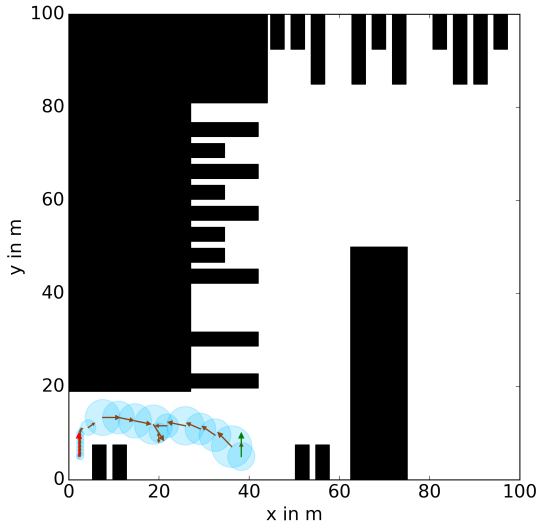


(c) Heuristic Function : Max(Euclid, Curve)
Discretization Method : Limited Angle

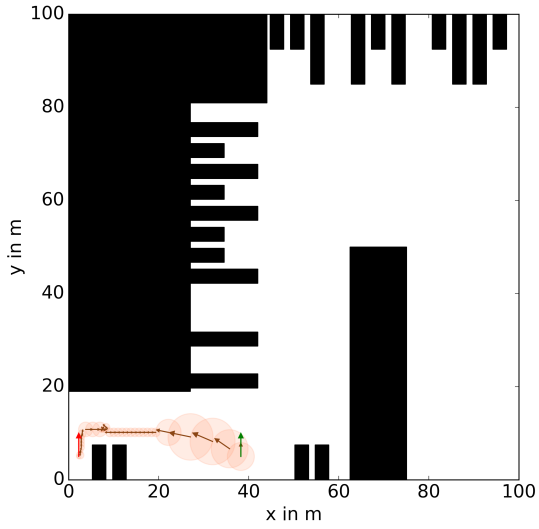


(d) Heuristic Function : RS length
Discretization Method : Limited Angle

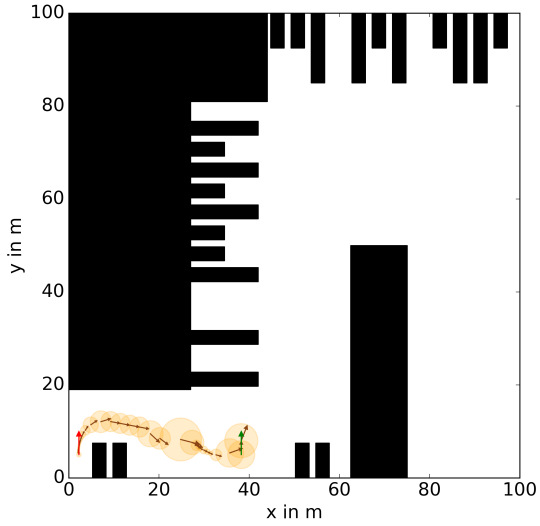
Figure 0-12: The tunnel T computed using different variations of space exploration method.



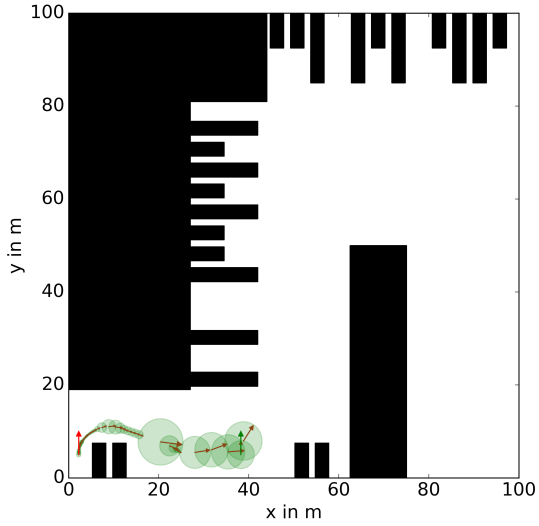
(a) Heuristic Function : Max(Euclid, Curve)
Discretization Method : All Angle



(b) Heuristic Function : RS length
Discretization Method : All Angle

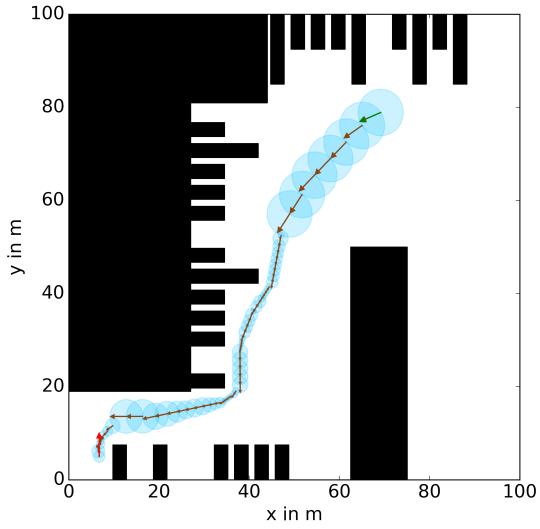


(c) Heuristic Function : Max(Euclid, Curve)
Discretization Method : Limited Angle

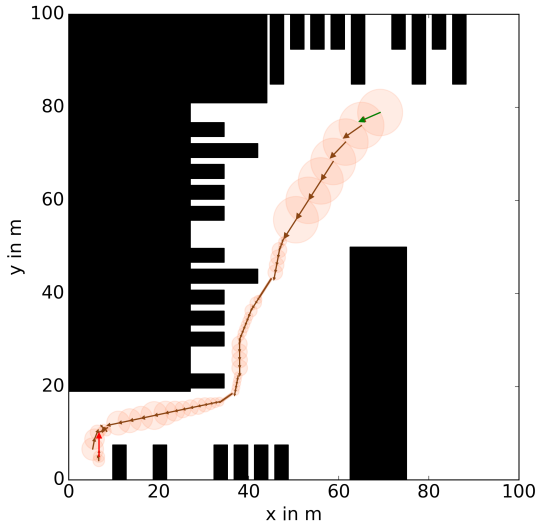


(d) Heuristic Function : RS length
Discretization Method : Limited Angle

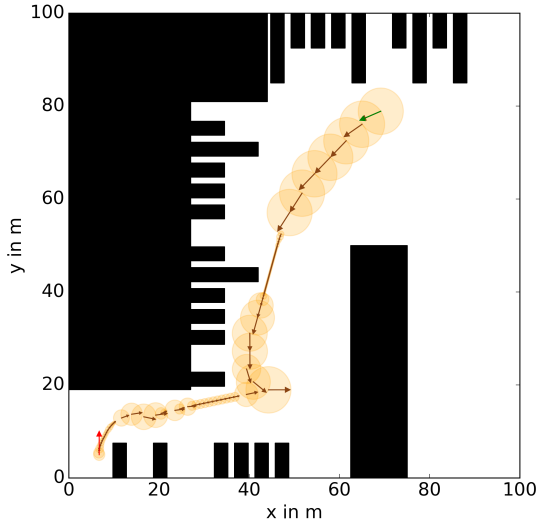
Figure 0-13: The tunnel T computed using different variations of space exploration method.



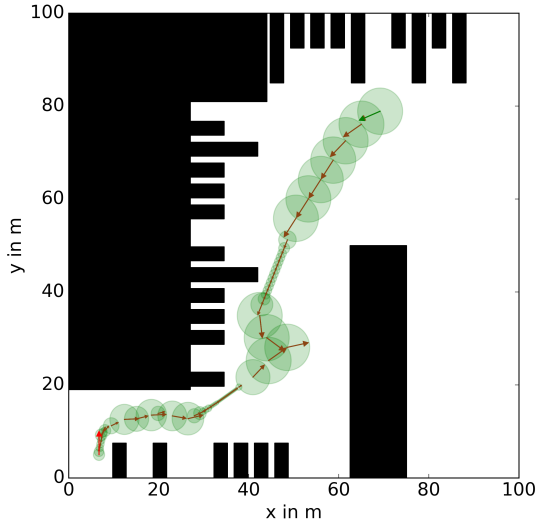
(a) Heuristic Function : Max(Euclid, Curve)
Discretization Method : All Angle



(b) Heuristic Function : RS length
Discretization Method : All Angle

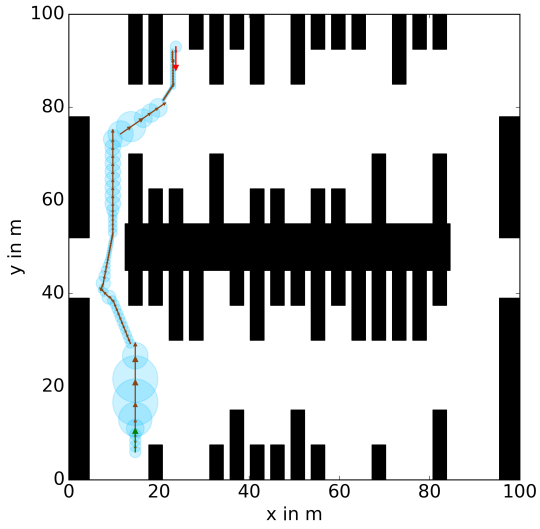


(c) Heuristic Function : Max(Euclid, Curve)
Discretization Method : Limited Angle

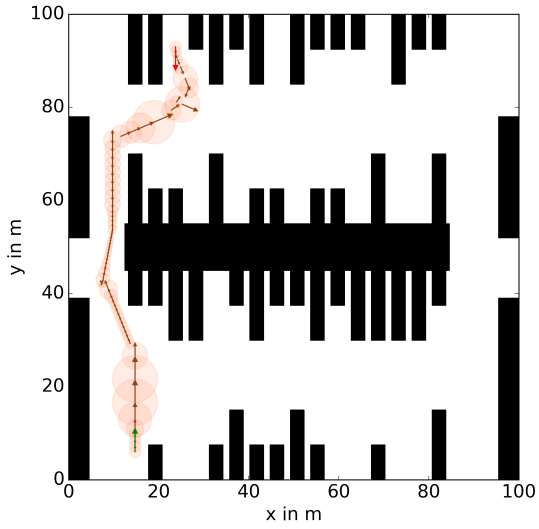


(d) Heuristic Function : RS length
Discretization Method : Limited Angle

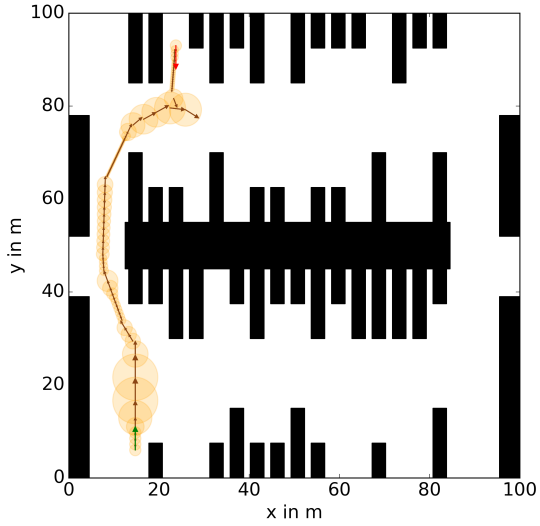
Figure 0-14: The tunnel T computed using different variations of space exploration method.



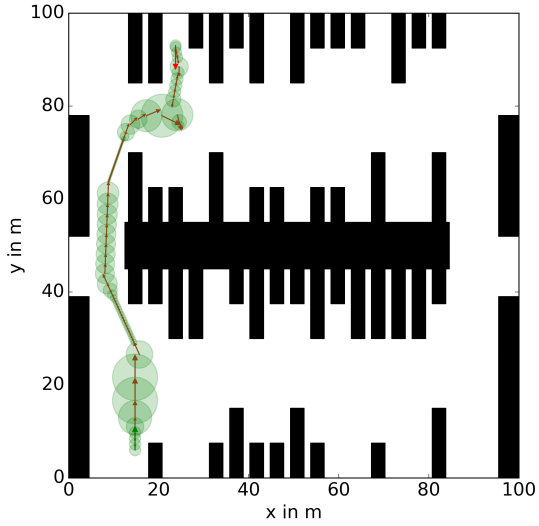
(a) Heuristic Function : Max(Euclid, Curve)
Discretization Method : All Angle



(b) Heuristic Function : RS length
Discretization Method : All Angle



(c) Heuristic Function : Max(Euclid, Curve)
Discretization Method : Limited Angle



(d) Heuristic Function : RS length
Discretization Method : Limited Angle

Figure 0-15: The tunnel T computed using different variations of space exploration method.

C. Problem specific samples using Learned sampling.

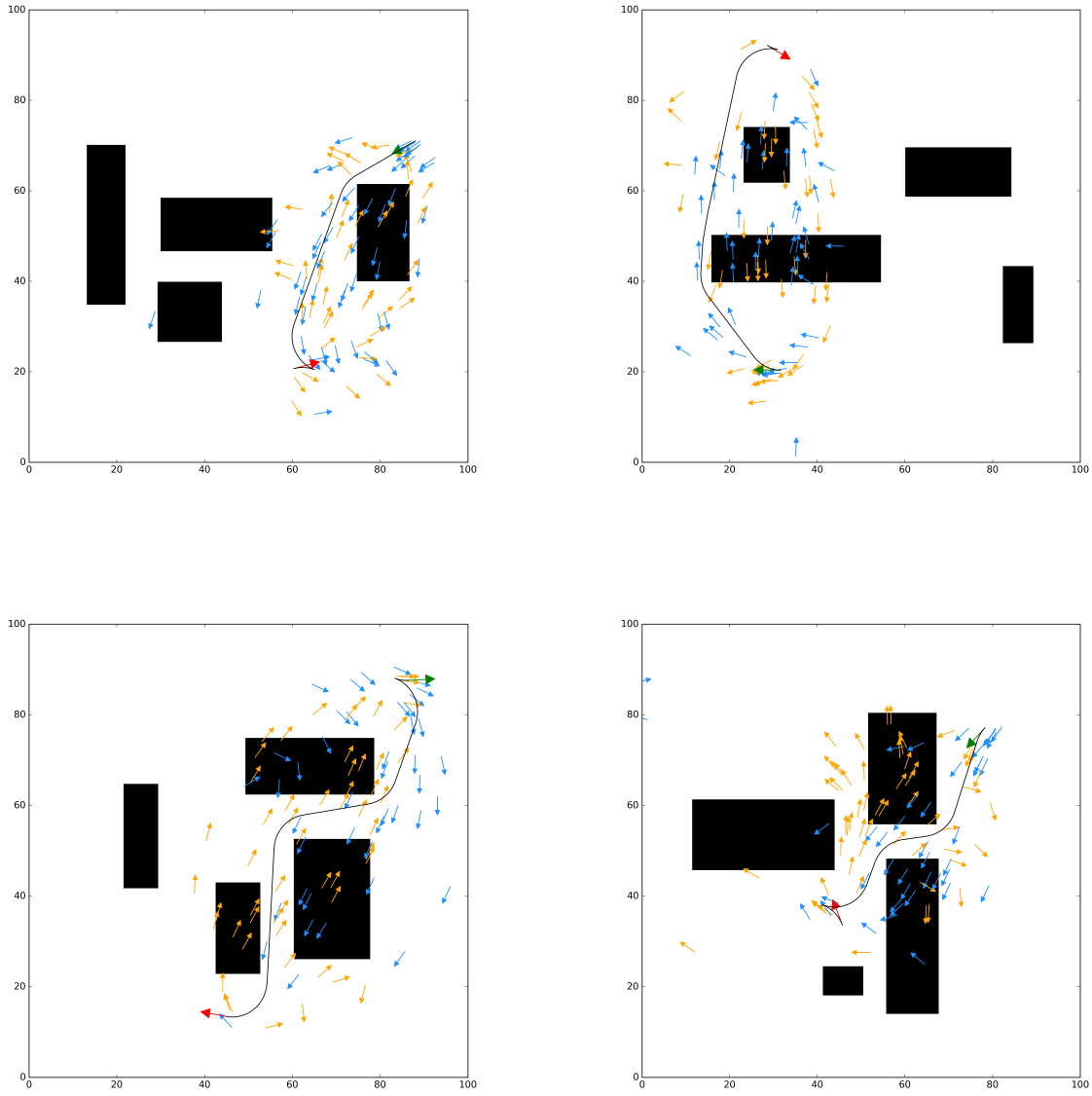


Figure 0-16: Path planning problem defined by start pose (green) and goal pose (red). The black curve represents the optimal solution. Samples (blue and orange arrow) generated by the model (Number of samples = 100). blue is forward direction and orange is reverse direction.

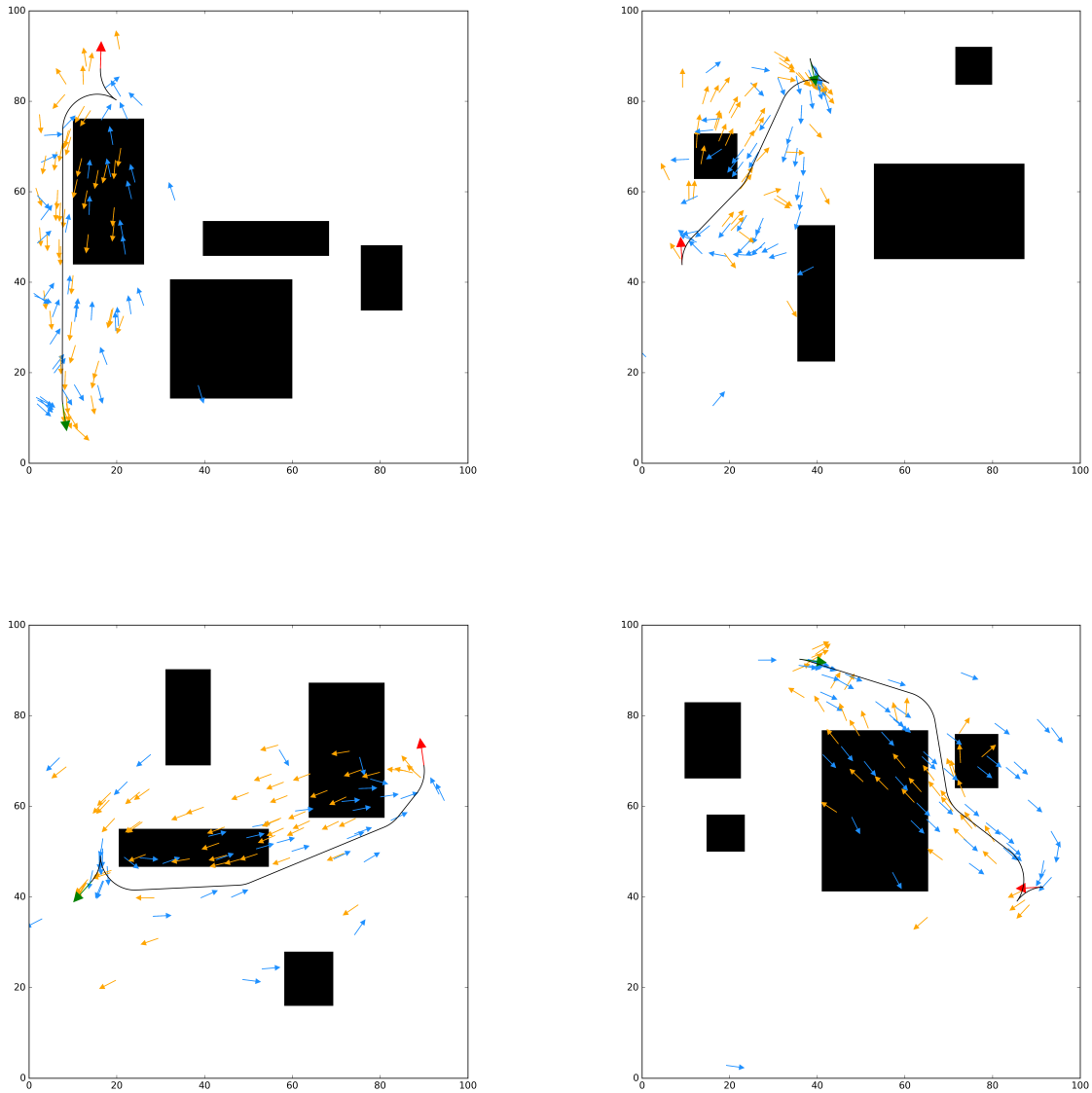


Figure 0-17: Path planning problem defined by start pose (green) and goal pose (red). The black curve represents the optimal solution. Samples (blue and orange arrow) generated by the model (Number of samples = 100). blue is forward direction and orange is reverse direction.

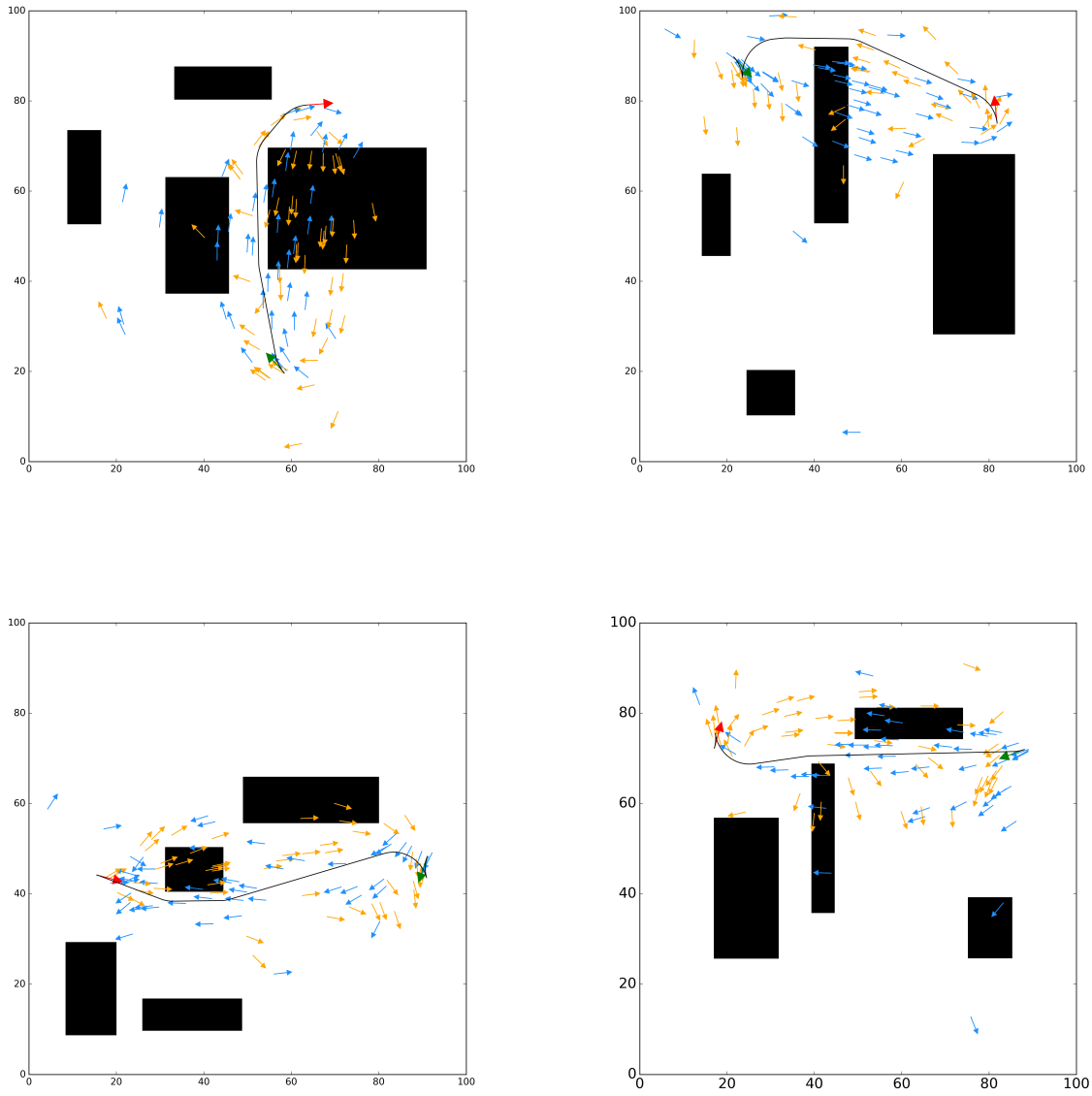


Figure 0-18: Path planning problem defined by start pose (green) and goal pose (red). The black curve represents the optimal solution. Samples (blue and orange arrow) generated by the model (Number of samples = 100). blue is forward direction and orange is reverse direction.