INFORMED EXPLORATION ALGORITHMS FOR ROBOT MOTION PLANNING AND LEARNING

A Dissertation
Presented to
The Academic Faculty

By

Sagar Suhas Joshi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Institute for Robotics and
Intelligent Machines

Georgia Institute of Technology

May 2022

INFORMED EXPLORATION ALGORITHMS FOR ROBOT MOTION PLANNING AND LEARNING

Thesis committee:

Dr. Panagiotis Tsiotras, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Harish Ravichandar
School of Interactive Computing
Georgia Institute of Technology

Dr. Seth Hutchinson
School of Interactive Computing
Georgia Institute of Technology

Dr. Byron Boots
School of Computer Science and Engineering
University of Washington

Dr. Mathew Gombolay
School of Interactive Computing
Georgia Institute of Technology

Date approved: 04/11/2022

There is only one extraordinary thing about me, my parents.

For Vidya and Suhas Joshi.

I wish that all those calamities would happen again and again so that we could see You again and again,

for seeing You means that we will no longer see repeated births and deaths.

- Queen Kunti to Sri Krishna, Bhagavadgita

ACKNOWLEDGMENTS

I express my most sincere gratitude to my advisor Prof. Panagiotis Tsiotras for his constant support during my doctoral program. He gave me the freedom to choose and explore my thesis topic, while encouraging me to maintain a balance between the theoretical and the applications aspect of research. I learnt from Panos how to express my ideas with mathematical rigor so that an expert in the field could appreciate them. At the same time I also understood how to motivate my research so that a novice might grasp the core ideas and get inspired.

I am deeply thankful to Prof. Seth Hutchinson for giving me an opportunity to collaborate with him . Seth's expertise in the field motion planning, his critical thinking and intuitions were invaluable to me when I was tackling some of the toughest problems in my PhD research. I have thoroughly enjoyed Prof. Byron Boot's robotics courses during his tenure at Georgia Tech. His research and coursework bolstered my fundamental knowledge of robotics. I am deeply thankful to Prof. Matthew Gombolay and Prof. Harish Ravichandar for giving me insightful suggestions on my research that led to significant improvements in my thesis. Finally, I am also grateful to my undergraduate professors at IIT-Madras, especially my M.Tech advisor Prof. Shankar Ram C.S, for sparking and nurturing my interest in A.I during those formative years.

They say it takes a village to raise a PhD, and it is absolutely true. This journey would have been a lot more difficult without my friends here in Atlanta. I believe roommates occupy a special space in any PhD student's life. As I look back, some of the best moments of this journey was me laughing with Mohit and Pranjal as we had those endless discussions about the trials and tribulations of PhD life. I am also deeply grateful to Aprameya and Savanthi, Shreya, Karthik, Chaitra, Eshwar, Amrithraj and Pradyumna for being my extended family here in Atlanta and making this journey so memorable. Special thanks to my undergrad friends Viju, Kishan and Sujata as well. A big thank you also to all my

DCSL lab-mates for helping me with my qualifying exam and indulging in the brainstorming sessions.

I am eternally grateful to my family back in India. Devika, Ajinkya and sweet Niharika were a big source of joy for me during these years. Finally, no words can capture how indebted I feel to my parents. It was my Mother's dream to see me get educated from a prestigious university and my Father's unbreakable belief that no challenge was ever too tough for me. I stand tall on the shoulders of their giant sacrifices.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Sampling-based methods have emerged as a promising technique for solving robot motion-planning problems. These algorithms avoid a priori discretization of the search-space by generating random samples and building a graph online. While the recent advances in this area endow these randomized planners with asymptotic optimality, their slow convergence rate still remains a challenge. One of the reasons for this poor performance can be traced to the widely used uniform sampling strategy that naïvely explores the entire search-space. Having access to an intelligent exploration strategy that can focus search, would alleviate one of the critical bottlenecks in speeding up these algorithms. This thesis endeavors to tackle this problem by presenting exploration algorithms that leverage different sources of information available during planning time. First, a non-parametric sampling technique is proposed for the basic case of geometric path-planning in uniform cost-spaces. This method employs sparsification and utilizes heuristics and prior obstacle data to conduct a prioritized search. Second, the uniform cost-space assumption is relaxed and a "Relevant Region" sampler is proposed for efficient exploration on general cost-maps. The proposed framework reduces dependence on heuristics by utilizing the planner's tree structure information. Next, the Relevant Region framework is extended by proposing a locally exploitative procedure that formulates sampling as an optimization problem. This technique generates new samples to improve the cost-to-come value of vertices in a neighborhood. Fourth, the geometric setting is relaxed and the problem of efficient exploration for the case of kino-dynamic motion planning is discussed. A "Time-Informed" exploration procedure that focuses search by leveraging ideas from reachability analysis, while still maintaining asymptotic optimality guarantees, is presented. Fifth, this thesis also explores the problem of intelligent query sampling for data generation to improve the performance of learning-based planners. Finally, this thesis develops a software stack for implementation of the proposed planning algorithms on a 1/10 th scale RACECAR robot.

CHAPTER 1
INTRODUCTION

1.1 Motivation

Recent years have seen a meteoric rise in the development and deployment of autonomous robotic systems. It is now commonplace to see robots performing complex tasks, such as a self-driving car navigating through a difficult traffic scenario, medical robot assisting in a surgery, aerial drones performing package delivery and industrial and warehouse robots operating in cluttered environments (illustration in Figure 1.1). Given the location information and a map of the environment, these robots need an efficient algorithm to plan their motion from point A to B. The motion planning problem is of one finding the optimal control inputs to steer the robot from its initial state to a goal region, avoiding infeasible regions and optimizing some notion of cost. While a core problem in robotics and artificial intelligence (AI), motion planning has also found applications in the fields of game development, computational biology and virtual prototyping [1, 2, 3].

The motion planning problem has many variants, and is widely-studied in the literature. A popular variant goes by the name of *mover's problem* [4], or the *piano mover's problem* [5]. The objective in this problem is to find a collision-free path for the robot, which is assumed to be a rigid body that can move freely with no kino-dynamic constraints. This problem can be extended by modeling a robot, such as a manipulator arm, as a union of several rigid bodies. An important abstraction used to solve such problems is the notion of *configuration-space* [6]. The main advantage of this concept is that a robot with arbitrary complex geometric shape can be represented as a point in this abstract space. The dimension of the configuration-space is the number of degrees of freedom of the robot, or the minimum number of parameters necessary to specify its configuration. In this thesis, the

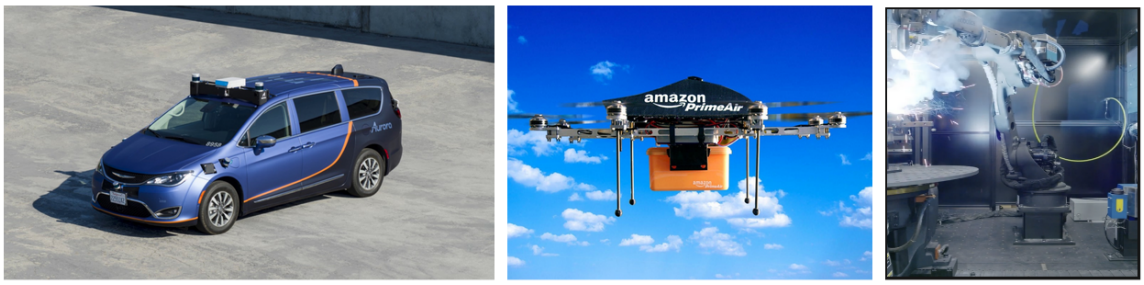


Figure 1.1: The Aurora self-driving car, ©Aurora Innovation (left), Amazon Prime Air drone package delivery system, ©Amazon Inc. (center), Path robotics autonomous welding manipulator robot, ©Path Robotics (right)

terms configuration-space and search-space are used interchangeably.

A popular technique for solving the motion planning problem is to first generate a graph representation of the feasible configuration-space in which the robot will operate. This graph encodes collision-free configurations or states of the robot as its vertices or nodes. The edges represent feasible motion or transition between the corresponding vertices. Algorithms such as Djikstra's [7] can then be used to *search* this graph and compute the optimal plan. However, Djikstra's algorithm with its uniformed search strategy becomes extremely expensive computationally as the problem size increases. In order to address this limitation, several algorithms that use a *heuristic* function to guide the exploration have been suggested. Perhaps the most celebrated of these *informed* search algorithms is A* [8]. A* conducts a prioritized exploration of the graph by expanding nodes with the lowest f value (sum of cost-to-come plus the heuristic estimate of cost-to-go). The use of an admissible heuristic leads to a focused search, and A* expands a provably minimum number of nodes for a given planning query. Variants of A*, such as ARA* [9] endow it with *anytime* property by leveraging inadmissible and admissible heuristics. Other popular extensions such as the LPA* [10] and the D* [11] algorithm address the problem of planning in dynamic environments.

Conventionally, the above deterministic search algorithms create a grid-based graph representation of the search-space. However, these algorithms are impractical for solving

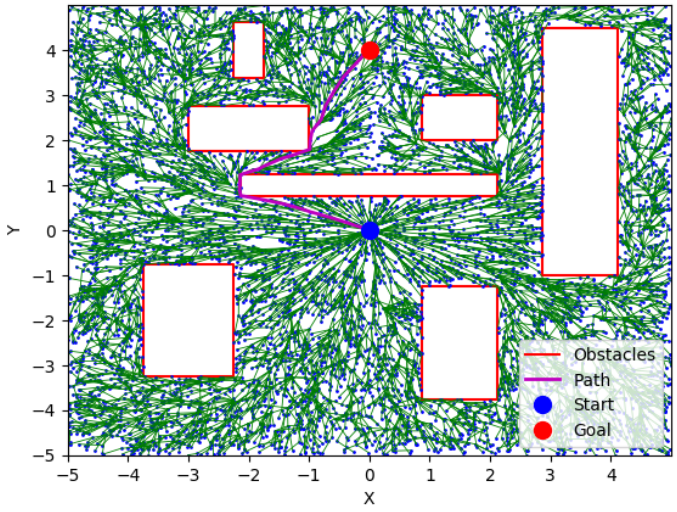


Figure 1.2: Path planning using sampling-based algorithms. The solution path (magenta) is found by incrementally growing a graph (green) in the free-space. Uniform sampling strategy results in a rapid exploration and graph-growth in all the areas of search-space.

problems in higher dimensions due to the computational cost associated with apriori discretization. Sampling-based motion planning (SBMP) algorithms, also called randomized motion planning algorithms, avoid this overhead by not resorting to discretization or explicit construction of the search-space. Instead, these algorithms use a black-box collision checking function to probe a set of random samples and local connections. These random samples are used to incrementally build a connectivity graph. Please see Figure 1.2. Due to their implicit representation of the search-space, scalability to higher dimensional problems and the ability to handle kino-dynamic constraints, SBMP algorithms have become the default choice for solving many complex robotic planning tasks.

While SBMP algorithms can effectively find a feasible solution in a broad range of problems, they may fail to return a good quality solution in many cases. One of the reasons for this poor performance can be traced to the sampling strategy used by these planners to build the connectivity graph. Ideally, a oracular sampling strategy would just generate samples along the optimal path. More realistically however, the sampler needs to balance exploration and exploitation during the planning process. This is because the planner first

needs to discover all the relevant areas in the space and then focus search on it to ensure a good quality solution is returned. Conventionally, SBMP algorithms have employed a uniform random sampling strategy. This results in a rapid, but also a naïve exploration of the entire search-space. The "exploration-bias" of uniform random strategy can have a detrimental effect on the quality of the solution returned by the planner, especially in higher dimensions. The objective of this thesis is to address these issues by devising a family of intelligent exploration algorithms that leverage different sources of information available during planning. To that end, this thesis makes the following contributions.

1.2 Thesis Contributions

- This thesis introduces a Non-parametric Informed Sampling (NP-Informed Sampling) approach for SBMP, that uses collision and graph-structure information gathered during planning. The method generates new samples from a set of well dispersed "kernel vertices" located in the "Informed Set" [12]. The proposed algorithm uses heuristics and graph state information to conduct a prioritized search. Data from the past collision checks is leveraged to learn the location of the obstacles and to avoid sampling in the obstacle space.

- Many of the intelligent exploration algorithms in the literature are catered towards uniform cost-map or length-optimal cases. However, several applications require planning algorithms to find a high quality solution with respect to a given cost function. This thesis proposes a generative "Relevant Region" sampling algorithm for intelligent exploration on cost-maps. Relevant Region, a subset of the Informed Set, leverages graph information to reduce dependence on heuristics and further focus search compared to traditional approaches such as T-RRT [13].

- Conventionally, the sampling strategy used by the randomized planners is biased towards exploration to acquire information about the search-space. In contrast, this

thesis proposes an optimization-based procedure that adds an exploitative-bias to sampling. A locally exploitative sampling (LES) procedure, an extension of Relevant Region sampling, generates new samples to improve the cost-to-come value of vertices in a neighborhood. This results in a faster convergence to the optimal solution compared to other state-of-the-art sampling techniques for geometric planners.

- While various intelligent exploration techniques have been suggested for geometric SBMP algorithms, devising analogous methods for their kino-dynamic counterparts still remains an active challenge. Using ideas from reachability analysis, this thesis defines a "Time-Informed Set (TIS)", that focuses the search for time-optimal kino-dynamic planning after an initial solution is found. Such a Time-Informed Set includes all trajectories that can potentially improve the current best solution. Hence, exploration outside this set is redundant. Benchmarking experiments show that an exploration strategy based on the TIS can accelerate the convergence of sampling-based kino-dynamic motion planners.

- In a slightly tangential direction to the above four contributions, this thesis also explores the problem of intelligent query sampling in the context of learning to plan. In the recent years, learning-based approaches have revolutionized motion planning. The data generation process for these methods involves caching high quality paths for different queries (start, goal pairs) in various environments. Conventionally, a uniform random strategy is used for sampling these queries. However, this leads to inclusion of "trivial paths" in the dataset (example, straight line paths in case of length-optimal planning), which can be solved efficiently if the planner has access to a steering function. This work proposes a "non-trivial" query sampling procedure to add more complex paths in the dataset. Numerical experiments show that a higher success rate can be attained for neural planners trained on such a non-trivial dataset.

- Finally, this thesis implements the proposed path-planning algorithms on 1/10 th

scale RACECAR robot. A Robotic Operating System (ROS) based software stack has been developed and consists of a LiDAR based obstacle avoidance module and full SLAM, planning and control-based navigation module. This stack has been tested in simulation and on physical hardware.

1.3 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 formally defines the optimal motion planning problem and the related notations. This is followed by a discussion on sampling-based motion planning and the state-of-the-art planners. Next, the exploration problem in SBMP is motivated along with a review of different intelligent exploration techniques proposed in the literature. Chapter 3 takes a step towards addressing some of the limitations of the baseline Informed Sampling [12] method by introducing the NP-Informed sampling technique. This technique efficiently utilizes graph-structure and collision information to conduct a prioritized search. Next, ideas from Chapter 3 are leveraged in Chapter 4 to devise a generative "Relevant Region" sampling algorithm. This sampling technique effectively focuses search in uniform and general cost-map settings. Chapter 5 extends the Relevant Region framework by incorporating gradient information and formulating sampling as an optimization problem. The locally exploitative sampling (LES) algorithm proposed in this chapter outperforms all other baseline techniques in higher dimensions. While the previous three chapters focus on "geometric" planning, Chapter 6 tackles the challenging problem of intelligent exploration for "kino-dynamic" planning. This chapter uses ideas from reachability analysis to focus search while still maintaining some theoretical guarantees. Chapter 7 explores the problem of intelligent (query-)sampling while creating a dataset in the context of learning to plan. Chapter 8 discusses the deployment of some of the proposed planning algorithms on the RACECAR robot. Finally, Chapter 9 summarizes the contributions of this thesis and lays out the scope for future work.

CHAPTER 2

PROBLEM DEFINITION AND LITERATURE REVIEW

2.1 Motion Planning Problem

This section formally defines the motion planning problem and the related notations. Let $\mathcal{X} \subset \mathbb{R}^d$, $d \geq 2$ and $\mathcal{U} \subset \mathbb{R}^m$, $m \geq 1$ be compact sets representing the state and admissible control spaces respectively. Let $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$ denote the obstacle space and $\mathcal{X}_{\text{free}} = \text{cl}(\mathcal{X} \setminus \mathcal{X}_{\text{obs}})$ denote the free space. Here, $\text{cl}(S)$ represents the closure of the set $S \subset \mathbb{R}^n$. Let $\lambda(S)$ denote the Lebesgue measure of the set $S \subset \mathbb{R}^n$. Let $\mathbf{x}_{\text{s}} \in \mathcal{X}_{\text{free}}$ denote the initial state and let $\mathcal{X}_{\text{g}} \subset \mathcal{X}_{\text{free}}$ represent the goal set. The kino-dynamic motion planning problem can be defined as follows:

$$\min_{\mathbf{u}} \int_0^T \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t)) \, \mathrm{d}t \tag{2.1a}$$

$$\text{subject to: } \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.1b}$$

$$\mathbf{x}(0) = \mathbf{x}_{\text{s}}, \ \mathbf{x}(T) \in \mathcal{X}_{\text{g}}, \tag{2.1c}$$

$$\mathbf{x}(t) \in \mathcal{X}_{\text{free}}, \ \mathbf{u}(t) \in \mathcal{U} \quad \text{for all } t \in [0, T]. \tag{2.1d}$$

This problem is one of finding the optimal control $\mathbf{u}$, that takes the robot from the start state $\mathbf{x}_{\text{s}}$ to the goal-region $\mathcal{X}_{\text{g}}$ (Equation 2.1c) while minimizing the cost functional (Equation 2.1a), subject to dynamics (Equation 2.1b) and state and control feasibility constraints (Equation 2.1d).

The "geometric" version of the above problem ignores the kino-dynamic constraints of the robot. Then, the cost of moving from a point $\mathbf{x}_1 \in \mathcal{X}$ to $\mathbf{x}_2 \in \mathcal{X}$ along a path

$\pi : [0, 1] \to \mathcal{X}$, $\pi(0) = \mathbf{x}_1$, $\pi(1) = \mathbf{x}_2$ can be denoted by $\mathrm{c}_\pi(\mathbf{x}_1, \mathbf{x}_2)$

$$\mathrm{c}_\pi(\mathbf{x}_1, \mathbf{x}_2) = \int_0^1 C(\pi(s)) \, \| \frac{\mathrm{d}\pi(s)}{\mathrm{d}s} \|_2 \, \mathrm{d}s. \tag{2.2}$$

Here, $C : \mathcal{X} \to \mathbb{R}_{\geq 0}$ denotes a continuous state cost function. Note that Equation 2.2 represents the integral of state-cost (IC) metric as a measure of path quality [13]. The path π in Equation 2.2 is assumed to be collision free. The path-cost is infinite otherwise. The geometric, optimal path planning problem can be formally defined as the search for minimum cost path π^* from the set of feasible paths Π connecting the start state $\mathbf{x}_\mathrm{s} \in \mathcal{X}_\mathrm{free}$ to the goal region $\mathcal{X}_\mathrm{goal} \subset \mathcal{X}_\mathrm{free}$,

$$\begin{aligned} &\min_{\pi \in \Pi} \mathrm{c}_\pi(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{g}), \\ &\text{subject to: } \pi(0) = \mathbf{x}_\mathrm{s}, \ \pi(1) = \mathbf{x}_\mathrm{g} \in \mathcal{X}_\mathrm{goal}, \\ &\pi(s) \in \mathcal{X}_\mathrm{free}, \quad s \in [0, 1]. \end{aligned} \tag{2.3}$$

SBMP algorithms solve the above problem in Equation 2.3 by constructing a connectivity graph $\mathcal{G} = (V, E)$ with a finite set of vertices $V \subset \mathcal{X}_\mathrm{free}$ and a set of edges $E \subseteq V \times V$. Conventionally, geometric sampling-based planners construct an edge $(\mathbf{u}, \mathbf{v}) \in E$ using a straight line path $\pi(s) = \mathbf{u} + (\mathbf{v} - \mathbf{u})s$, $s \in [0, 1]$ connecting $\mathbf{u}$ and $\mathbf{v}$. Using Equation 2.2, the edge-cost can be denoted as

$$\mathrm{c}_\ell(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2 \int_0^1 C(\mathbf{u} + (\mathbf{v} - \mathbf{u})s) \, \mathrm{d}s. \tag{2.4}$$

SBMP algorithms can perform numerical integration to calculate the edge-cost $\mathrm{c}_\ell(\mathbf{u}, \mathbf{v})$ for any edge $(\mathbf{u}, \mathbf{v}) \in E$. The graph $\mathcal{G}$ embeds a spanning tree $\mathcal{T} = (V_t, E_t)$ with $V_t = V$ and $E_t = \{(\mathbf{u}, \mathbf{v}) \in E \mid \mathbf{v} = \mathsf{parent}(\mathbf{u})\}$. Here, $\mathsf{parent} : V \to V$ denotes the function mapping a vertex to its unique parent in the tree. By definition, we have $\mathsf{parent}(\mathbf{x}_\mathrm{s}) = \mathbf{x}_\mathrm{s}$. The cost-to-come value $\mathrm{g}_\mathcal{T}(\mathbf{v})$ for a vertex $\mathbf{v}$ denotes the sum of edge-costs along the path

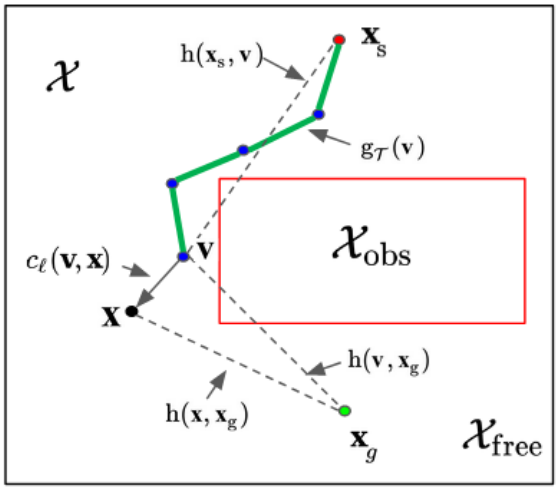


Figure 2.1: A schematic for the geometric planning problem.

from $\mathbf{v}$ to the root $\mathbf{x}_\mathrm{s}$ in $\mathcal{T}$. The function $\mathrm{g}_\mathcal{T} : V \to \mathbb{R}_{\geq 0}$ can be written recursively as

$$\mathrm{g}_\mathcal{T}(\mathbf{v}) = \mathrm{g}_\mathcal{T}(\mathbf{v}_\mathrm{p}) + \mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v}), \tag{2.5}$$

where $\mathbf{v}_\mathrm{p} = \mathsf{parent}(\mathbf{v})$. By definition, the recursion ends at $\mathbf{x}_\mathrm{s}$ with $\mathrm{g}_\mathcal{T}(\mathbf{x}_\mathrm{s}) = 0$. Let the set of children for vertex $\mathbf{v}$ be denoted by $V_\mathbf{v} = \{\mathbf{u} \in V \mid \mathbf{v} = \mathsf{parent}(\mathbf{u})\}$ and the number of children by $n_\mathbf{v} = |V_\mathbf{v}|$. Descendants of a vertex $\mathbf{v}$ are all the vertices $\mathbf{u} \in V$ whose path from $\mathbf{u}$ to the root $\mathbf{x}_\mathrm{s}$ in $\mathcal{T}$ contains $\mathbf{v}$. Let $D_\mathbf{v}$ denote the set of vertices that are descendants of $\mathbf{v}$ and let $d_\mathbf{v} = |D_\mathbf{v}|$. Then,

$$d_\mathbf{v} = n_\mathbf{v} + \sum_{\mathbf{u} \in V_\mathbf{v}} d_\mathbf{u}. \tag{2.6}$$

Note that for a leaf vertex $\mathbf{v} \in V$, we have $n_\mathbf{v} = d_\mathbf{v} = 0$.

Let $\mathrm{h} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ denote a consistent heuristic function. This function obeys the triangle inequality and gives an under-estimate of the path-cost $\mathrm{c}_\pi(\mathbf{x}_1, \mathbf{x}_2)$ between any two points $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$. An example of function h is the L_2-norm (Euclidean distance). Let $\mathcal{B}^\epsilon(\mathbf{x}_\mathrm{o})$ denote an ϵ-ball around $\mathbf{x}_\mathrm{o} \in \mathcal{X}$, given by $\mathcal{B}^\epsilon(\mathbf{x}_\mathrm{o}) = \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}_\mathrm{o}\|_2 < \epsilon\}$, for $\epsilon > 0$. Please see the illustration in Figure 2.1.

Algorithm 1: Sampling-Based Planning Flow

$V \leftarrow \{\mathbf{x}_{\mathrm{s}}\}$; $E \leftarrow \phi$; $\mathcal{G} \leftarrow (V, E)$;
for $i = 1 : N$ **do**
  $\mathbf{x}_{\mathrm{rand}} \leftarrow \mathsf{getRandomSample}(\mathcal{X})$;
  $\mathbf{x}_{\mathrm{near}} \leftarrow \mathsf{nearestNeighbhor}(\mathbf{x}_{\mathrm{rand}}, \mathcal{G})$;
  $\mathbf{x}_{\mathrm{new}} \leftarrow \mathsf{Extend}(\mathbf{x}_{\mathrm{near}}, \mathbf{x}_{\mathrm{rand}})$;
  $\mathsf{GraphProcessing}(\mathcal{G})$;
return $\mathcal{G}$

2.2 Sampling-Based Motion Planning

A defining feature of sampling-based motion planning algorithms is that they solve the motion planning problem in Equation 2.1 and Equation 2.3 by generating random samples in $\mathcal{X}$ and incorporating the feasible ones in $\mathcal{G}$. These algorithms can be broadly divided into two classes, namely, Probabilistic Road Maps (PRM) [14] and Rapidly-exploring Random Trees (RRT) [15]. PRM type of algorithms are catered towards multi-query problems and consist of two phases: a learning phase and a query phase. In the learning phase, a probabilistic roadmap is computed and stored as a graph. In the query phase, the given start and goal point are first connected to the nearest nodes in this graph. A search algorithm is then used to find an optimal path in the roadmap connecting the start and goal. In contrast, single-query algorithms such as RRT do not invest computational resources in the offline computation of a roadmap. Instead, these algorithms incrementally build a connectivity graph online to quickly explore the search-space. However, SBMP algorithms such as RRT and PRM come with a relaxed notion of *probabilistic completeness*, i.e., the probability of finding a feasible solution, if it exists, approaches unity as the number of samples tends to infinity [16]. Also,these algorithms have been proven to yield sub-optimal paths almost surely [17]. In order to address this, *asymptotically optimal* variants of these algorithms such as RRT*, RRG and PRM* have been proposed [17]. These algorithms converge to the optimal solution almost-surely, as the number of samples tend to infinity.

Single-query, asymptotically optimal SBMP algorithms, such as RRT*, generally fol-

low the sequence of steps given in algorithm 1. These algorithms perform two fundamental tasks, namely, graph growth and graph processing. The graph-growth module performs the following set of steps to grow the planner graph $\mathcal{G}$. First, it generates a random sample $\mathbf{x}_{\mathrm{rand}}$. Second, nearest-neighbor calculations are performed to get the vertex $\mathbf{x}_{\mathrm{near}} \in V$, which is nearest to $\mathbf{x}_{\mathrm{rand}}$. The Extend function obtains a candidate vertex $\mathbf{x}_{\mathrm{new}}$ by either forward propagating the system from $\mathbf{x}_{\mathrm{near}}$ with random control inputs or locally steering in the direction of $\mathbf{x}_{\mathrm{rand}} - \mathbf{x}_{\mathrm{near}}$. The motion from $\mathbf{x}_{\mathrm{near}}$ to $\mathbf{x}_{\mathrm{new}}$ is then collision-checked. If found feasible, $\mathbf{x}_{\mathrm{new}}$ along with the appropriate set of edges are included in the graph.

Given $\mathcal{G}$, the graph-processing module tries to improve the cost-to-come value of the vertices by performing operations such as edge rewiring. The graph is said to be rewired if the parent of a vertex changes, improving its cost-to-come value. In particular, the "local rewiring" procedure of RRT* first selects the best parent for a newly initialized vertex. It then sees if this new vertex can be a better parent for any of the vertices in its neighborhood. The RRT$^{\#}$ [18] algorithm provides an extension to the RRT* procedure by "globally rewiring" the graph using dynamic programming. It uses value-iteration [18] or policy-iteration [19] to optimally connect each vertex in the graph in order to minimize their cost-to-come values. Recently proposed methods such as BIT* [20] and FMT* [21] also use ideas from dynamic programming and heuristics to obtain faster convergence than RRT*. Extensions of BIT* such as ABIT* [22] uses advanced graph-search techniques to find the first solution quickly. AIT* [23] employs a asymmetric bi-directional search to adaptively estimate a problem specific heuristic. The DRRT algorithm [24] does the graph-processing by performing a gradient-descent procedure and moving the graph vertices to a better location. The RRdT* planner [25] efficiently explores the search-space using multiple local trees.

The geometric versions of the sampling-based algorithms ignore kino-dynamic constraints of the robot and connect any two points in a Euclidean search space with a straight line. However, a general kino-dynamic problem Equation 2.1 requires the solution of a

two-point boundary value problem (TPBVP), also called the "local steering" problem, for optimally connecting any two states. Karaman and Frazzoli extended the RRT* algorithm for kino-dynamic planning by incorporating such steering functions in [26]. Perez et al [27] linearized the system dynamics and solved the infinite-horizon linear quadratic regulator (LQR) problem to obtain a locally optimal steering procedure. The kino-dynamic RRT* algorithm [28] penalizes the control effort and the trajectory duration while connecting any two states. The authors of [28] solve a fixed final state, free final time, optimal control problem for linear time invariant (LTI) systems to derive a steering function. A kino-dynamic version of FMT* is presented in [29]. Note that these algorithms rely on the availability of a local steering module to ensure asymptotic optimality. However, developing such computationally efficient TPBVP solvers may not be possible for many cases. The GR-FMT algorithm [30] proposes a local steering method based on polynomial basis functions and segmentation for controllable linear systems. The recently introduced Stable Sparse RRT (SST) and SST* [31] algorithms guarantee asymptotic optimality, while having access only to a forward propagation model of the system's dynamics. This eliminates the need for TP-BVP solvers. The SST procedure promotes the propagation of states with good path costs and performs a selective pruning operation to keep the number of stored nodes small.

2.3 Exploration Problem

The sampling strategy used by a randomized planner to generate $\mathbf{x}_{\mathrm{rand}}$ (algorithm 1, line 3) plays a critical role in ensuring a fast convergence to the optimal solution. The easiest and most popular way to generate these random samples is to uniformly sample $\mathcal{X}$. This leads to an implicit Voronoi bias in RRT-style algorithms and a rapid exploration of the search-space. Please see Figure 2.2. However, this naive strategy suffers from the following limitations. First, uniform sampling prioritizes acquisition of new information over the improvement of current paths in the planner's graph. This bias towards exploration can have a detrimental effect on convergence, especially in higher dimensions [12], [24]. Sec-

ondly, this strategy does not utilize two sources of information that are available during the planning process: the previous collision checks and the planner's current graph structure. The collision checks provide data regarding the location of the obstacles, while the graph structure can provide insight about areas that are already well represented by the planner, and areas that need to be explored further. Thirdly, after an initial solution is discovered, the search should ideally focus on a subset of the space that can possibly further improve the current solution. Sampling outside this subset is redundant and wastes computational resources.

2.3.1 Geometric Planning

In order to remedy the above problems, several intelligent exploration strategies have been suggested for geometric sampling-based planners. Akgun and Stilman [32] introduce local biasing to sample in the vicinity of the path vertices after an initial solution is found. Urmson and Simmons [33] aim to focus search by calculating a heuristic-based quality measure for every vertex in the tree Diankov and Kuffner [34] propose a continuous randomized version of A* in their RA* algorithm. Burns and Brock propose an information-theoretic framework in a multi-query [35] and a single-query setting [36] to guide exploration towards regions of maximum information gain. The Exploring/Exploiting tree planner of [37], modulates the variance of Gaussian distributions based on success/failure of new samples. The RRM algorithm [38] adds edges to the current roadmap to balance exploration and refinement. Techniques such as [39], [40] use obstacle information to guide search during planning. MARRT [41] retracts samples onto the medial axis of the free space to obtain high clearance paths. The Expansive space trees (EST) algorithm [42] proceeds by selecting a vertex (with probability inversely proportional to number of vertices in its neighborhood) and generates a new sample in its vicinity. Guided ESTs [43] add the A* cost and an exploration term to the vertex weights. Persson and Sharf [44] present a formal analysis for the generalization of the A* cost. Their SBA* algorithm, another variant of

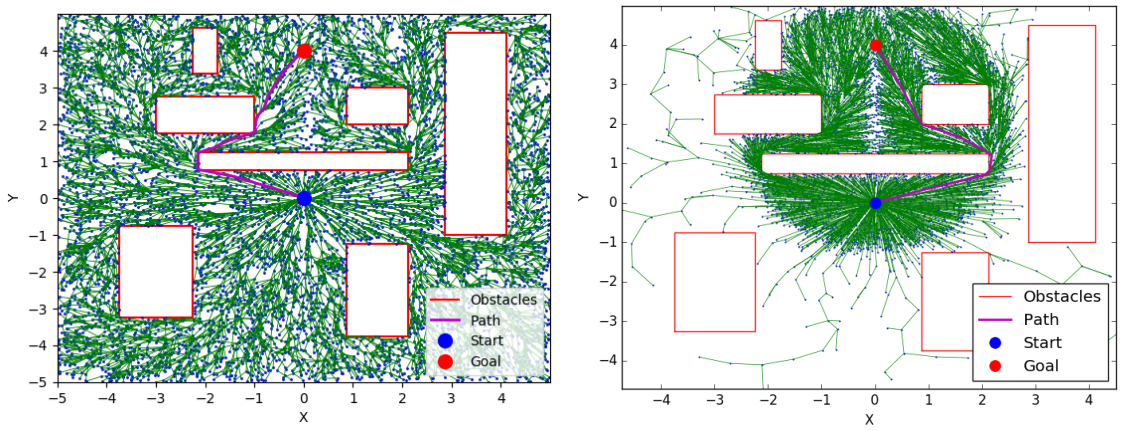


Figure 2.2: Uniform random sampling (Left) naively explores the entire search-space, while the L_2-Informed Sampling (Right) focuses on the ellipsoidal subset after an initial solution is discovered.

EST, uses graph density and constriction value to estimate the likelihood of sampling a new and free point in the neighborhood of a vertex.

While the above exploration algorithms leverage heuristics and collision-checking information, they end up naively exploring the entire search-space. In contrast, the Informed Sampling approach [12] avoids redundant exploration after an initial solution is discovered. It focuses search onto a subset of the search-space, called the Informed Set, that contains all the points that can potentially improve the current solution. Please see Figure 2.2 for a comparison between uniform and Informed Sampling. Let c_i denote the cost of the (sub-optimal) solution discovered by the planner after i iterations. Then, the Informed Set is defined as,

$$\mathcal{X}_{\text{inf}} = \{\mathbf{x} \in \mathcal{X} \mid \text{h}(\mathbf{x}_\text{s}, \mathbf{x}) + \text{h}(\mathbf{x}, \mathbf{x}_\text{g}) < c_i\}. \tag{2.7}$$

Note that $\mathcal{X}_{\text{inf}}$ considers a heuristic estimate of cost-to-come $\text{h}(\mathbf{x}_\text{s}, \mathbf{x})$, and a heuristic estimate of cost-to-go $\text{h}(\mathbf{x}, \mathbf{x}_\text{g})$ to get an under-estimate of the solution cost $\hat{f}(\mathbf{x}) = \text{h}(\mathbf{x}_\text{s}, \mathbf{x}) + \text{h}(\mathbf{x}, \mathbf{x}_\text{g})$ constrained to pass through $\mathbf{x} \in \mathcal{X}$. The Informed Set considers all the points for which this under-estimate is less than the current best solution cost c_i. If heuristic h is admissible, sampling in the Informed Set is a necessary (but not sufficient) condition to

improve the current solution. Exploration outside is set is thus redundant. For the case of length optimal with Euclidean distance heuristic $\mathrm{h}(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|_2$, the Informed Set take the following form,

$$\mathcal{X}_{\text{inf}} = \{\mathbf{x} \in \mathcal{X} \mid (\mathbf{x} - \mathbf{x}_{\text{c}})^{\text{T}} \mathbf{S}^{-1} (\mathbf{x} - \mathbf{x}_{\text{c}}) < 1 \}, \tag{2.8}$$

where, $\mathbf{S} \in \mathbb{R}^{d \times d}$, is a symmetric, positive-definite hyper-ellipsoid matrix and $\mathbf{x}_{\text{c}} = \frac{\mathbf{x}_{\text{s}} + \mathbf{x}_{\text{g}}}{2}$ is the center point. The above set in Equation 2.8, known as the L_2-Informed Set, is a prolate hyper-spheroid with focii at the start and the goal states and its transverse diameter is equal to the current best solution cost. Uniform random samples $\mathbf{x}_{\text{ellipse}}$ can be generated efficiently in the L_2-Informed Set using the following equation,

$$\mathbf{x}_{\text{ellipse}} = \mathbf{L}\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{c}}. \tag{2.9}$$

Here, $\mathbf{x}_{\text{ball}}$ represents a uniformly distributed sample in the d dimensional unit-ball $\mathcal{B}^1(\mathbf{0})$ and $\mathbf{L} \in \mathbb{R}^{d \times d}$ is the lower-triangular Cholesky decomposition of $\mathbf{S}$, $\mathbf{L}\mathbf{L}^{\text{T}} = \mathbf{S}$.

Thus, direct Informed Sampling using Equation 2.9 provides a scalable way to focus search while maintaining the asymptotic optimality guarantee. However, this approach suffers from certain limitations, which are delineated in the subsequent chapters. This thesis proposes and validates different frameworks to address these shortcomings.

2.3.2 Kino-dynamic Planning

Prior work on intelligent exploration described above, such as [32, 33, 12], utilized heuristics to improve the performance of geometric sampling-based planners. The Informed SST (iSST) [45] also leverages heuristics to guide search for kino-dynamic planning. DIRT [46] uses dominance informed regions along with heuristics to balance exploration and exploitation. However, iSST and DIRT may be ineffective in focusing the search for the cases where a good heuristic function is unavailable.

Concepts from reachability analysis have also been used for guiding exploration in sampling-based kino-dynamic planning. Shkolnik et al [47] used reachable sets in their RG-RRT algorithm to shape the Voronoi bias so as to find a feasible solution quickly. A discretized representation of the reachable space is proposed in [48] to be used for sampling and nearest neighbor search. Chiang et al [49] trained an obstacle-aware time-to-reach (TTR) reachability estimator network to guide the RRT search process. However, the above techniques do not focus search on a subset of the search space based on current solution cost, which can lead to redundant exploration.

The algorithms proposed in [50] and [51] are most relevant to this thesis, as they address the problem of Informed Sampling for kino-dynamic motion planning. Kunz et al [50] proposed a hierarchical rejection sampling (HRS) method to generate informed samples for higher-dimensional systems. HRS essentially is a "bottom up" procedure that generates samples along the individual dimensions and combines them. An accept/reject decision is taken for each partial sample until a complete sample in the informed set is generated. Yi et al [51] proposed a Hit-and-Run Markov Chain Monte-Carlo (HNR-MCMC) algorithm to improve the sampling efficiency compared to HRS. Given a previous sample in the Informed Set, the HNR-MCMC first samples a random direction and then uses rejection sampling to find the largest step-size so that the new sample lies inside the Informed Set. However, both HRS and HNR-MCMC assume availability of a "local steering function", that gives the optimal cost (or a good under-estimate) connecting any two states. For minimum time problems, the above two methods can only be applied to specific systems, such as the double integrator. In this thesis, we address this issue by using ideas from reachability analysis to define a "Time-Informed Set". The proposed algorithm can be applied to a wide variety of systems.

CHAPTER 3
NON-PARAMETRIC INFORMED EXPLORATION

3.1 Motivation

This chapter introduces a non-parametric sampling technique that leverages collision checking information and conducts a prioritized exploration of the Informed Set (Equation 2.7). The previous chapter briefly explained the L_2-Informed Sampling approach for the case of length-optimal planning. This method focuses search by generating samples in the hyper-ellipsoidal set, given in Equation 2.8. However, it does not leverage obstacle data from past collision checks and has to rely on rejection sampling to ensure that the new sample lies in $\mathcal{X}_{\text{free}}$. If the ratio $\lambda(\mathcal{X}_{\text{obs}} \cap \mathcal{X}_{\text{inf}})/\lambda(\mathcal{X}_{\text{inf}})$ is high, the probability of generating samples in the obstacle-space using Informed Sampling is also high. Secondly, it does not conduct a prioritized search by utilizing sources of information such as the structure of graph $\mathcal{G}$. In this work, the above issues are addressed by proposing a non-parametric sampling method. The method generates new samples from a set of "kernel vertices" located in the Informed subset of the search space. The proposed algorithm uses heuristics and graph state information to focus the search during the initial phase of planning. After a solution is found, new points are sampled only in the vicinity of kernel vertices that can potentially provide better solution. Data from the past collision checks is leveraged to learn the location of the obstacles and to avoid sampling in the obstacle space.

Expansive search trees (EST) [42] algorithm is especially relevant to this work. The EST algorithm picks a vertex with probability inversely proportional to the number of vertices in the neighborhood, for generating the next sample. A common theme in EST variants [44], [43], [52] is to include the graph density along with the A* cost to conduct a prioritized exploration of the search space. Although similar concepts are used in this work,

a key difference is that the proposed algorithm generates samples only in the Informed Set after the first connection is made. This saves the computational effort of redundant exploration. Secondly, while the EST algorithm and its variants expand vertices, the proposed algorithm generates samples in the vicinity of the set of well dispersed "kernel vertices" inside the Informed Set. A new kernel vertex is initialized only if it lies in the Informed Set and it is at a threshold distance away from all the existing kernel vertices. Thus, although the number of vertices may increase indefinitely as the iterations progress, the number of kernel vertices remain fixed after the entire informed set is explored. This greatly reduces the computational load of otherwise storing information and processing every vertex for expansion.

3.2 Algorithm Overview

The proposed method generates new samples in the neighborhood of a set of kernel vertices $K = \{k_1, k_2, ...k_n\}, n \in \mathbb{Z}^+$ (see Figure 3.1). The set of kernel vertices is a subset of graph vertices, i.e, $K \subset V$. Thus, not all graph vertices are kernel vertices. Let the state (i.e., location vector) of the kernel vertex k be $\boldsymbol{\mu}_k$. Let d_k be the number of edge connections from kernel vertex k. This represents the "graph density" around k. Let p_k be the number of sampling attempts from k and let $\hat{f}_k$ be the estimate of the cost path constrained to pass through $\boldsymbol{\mu}_k$, i.e. $\hat{f}_k = \mathrm{h}(\mathbf{x}_\mathrm{s}, \boldsymbol{\mu}_k) + \mathrm{h}(\boldsymbol{\mu}_k, \mathbf{x}_\mathrm{g})$. The set of *informed kernel vertices* is defined as

$$K_\mathrm{inf} = \{k \in K \mid \hat{f}_k < c_i\}. \tag{3.1}$$

The outline of the proposed method is given in algorithm 6. The algorithm is initialized by a kernel vertex at $\mathbf{x}_\mathrm{s}$. A new sample $\mathbf{x}_\mathrm{rand}$ is generated using conventional direct Informed Sampling with probability ϵ_k. This adds robustness and aids in exploration. Else, a kernel vertex is chosen to generate a new sample $\mathbf{x}_\mathrm{rand} \in \mathcal{X}_\mathrm{inf}$ in its vicinity (algorithm 6, lines 9-10). The Extend procedure (algorithm 6, line 11) involves conventional rapidly exploring

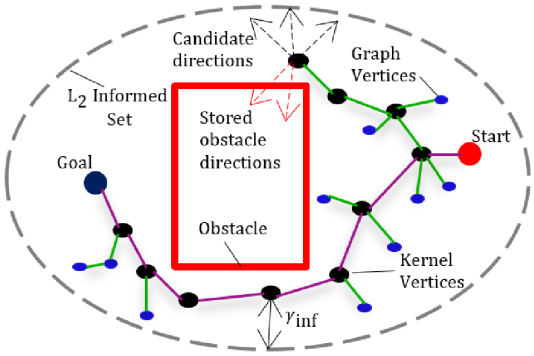


Figure 3.1: Schematic of NP-Informed sampling, showing kernel vertices(black) generating new samples in the L_2-informed set after an initial solution is found.

random graphs (RRG) modules: 1) finding the vertex nearest to $\mathbf{x}_{\text{rand}}$ in graph $\mathcal{G}$; 2) local steering from the nearest vertex to initialize a new vertex with state $\mathbf{x}_{\text{new}}$; 3) making new edge connections by finding vertices in the neighborhood of $\mathbf{x}_{\text{new}}$. Approximate nearest neighbors can be found efficiently using data structures such as KD-trees. We use FLANN [53] for nearest neighbor calculations. The Extend procedure also outputs δ_{min}: the distance of $\mathbf{x}_{\text{new}}$ to the closest kernel vertex. Step 3 can be used to search the neighborhood of $\mathbf{x}_{\text{new}}$ for kernel vertices to compute δ_{min}. If no kernel vertex is found, a separate KD-tree built with kernel vertex states $\boldsymbol{\mu}_k, \forall k \in K$, as input is used for calculating δ_{min}. If this distance is greater than a threshold value η', then a new kernel vertex is initialized at $\mathbf{x}_{\text{new}}$ (algorithm 6, lines 12-14). Thus, a new kernel vertex is initialized only if it is η' distance away from the existing kernel vertices. This avoids clumping all kernels in the same region and hence aids exploration. Though the graph vertices can be arbitrarily close, the kernel vertex inclusion procedure leads to evenly spaced kernels representing salient regions in $\mathcal{X}_{\text{inf}}$. Also, as the set $\mathcal{X}_{\text{inf}}$ is bounded, there can only be a finite number of kernel vertices.

3.2.1 Kernel Selection

The procedure for kernel selection is described in algorithm 3. A value q_k is calculated for every $k \in K_{\text{inf}}$ and the kernel with minimum q value is selected to generate the next

Algorithm 2: Sampling Algorithm Flow

1 $V \leftarrow \{\mathbf{v}_\mathrm{s}\}$; $E \leftarrow \phi$; $\mathcal{G} \leftarrow (V, E)$;
2 $\mu_{k_1} \leftarrow \mathbf{x}_\mathrm{s}$; $p_{k_1} \leftarrow 0$; $K \leftarrow \{k_1\}$;
3 **for** $i = 1 : N$ **do**
4 $c_i \leftarrow \min_{v \in V_\mathrm{goal}} \mathrm{g}_\mathcal{T}(v)$;
5 $u_\mathrm{rand} \sim \mathcal{U}[0, 1)$;
6 **if** $u_\mathrm{rand} < \epsilon_k$ **then**
7 $\mathbf{x}_\mathrm{rand} \leftarrow \mathrm{InformedSampling}()$
8 **else**
9 $k_i \leftarrow \mathrm{chooseKernel}(K_\mathrm{inf}, \mathcal{G})$;
10 $\mathbf{x}_\mathrm{rand} \leftarrow \mathrm{generateSample}(k_i)$;
11 $\delta_\mathrm{min}, \mathbf{x}_\mathrm{new} \leftarrow \mathrm{Extend}(\mathbf{x}_\mathrm{rand})$;
12 **if** $\delta_\mathrm{min} \geq \eta'$ **then**
13 $\mu_{k_\mathrm{new}} \leftarrow \mathbf{x}_\mathrm{new}$; $p_{k_\mathrm{new}} \leftarrow 0$;
14 $K \leftarrow K \cup \{k_\mathrm{new}\}$;
15 $\mathrm{Exploitation}(\mathcal{G})$;
16 **return** $\mathcal{G}$

sample. We consider the position information given by $\hat{f}_k$, the graph density information given by d_k, and the number of previous sampling attempts p_k to calculate the value q_k. The normalizing terms $(\hat{f}_\mathrm{max}, \hat{f}_\mathrm{min})$, $(p_\mathrm{max}, p_\mathrm{min})$ and $(d_\mathrm{max}, d_\mathrm{min})$, are as defined in algorithm 3, lines 2-4. The value q_k is a weighted linear combination with weights $\lambda_1, \lambda_2, \lambda_3$, such that $\sum_i \lambda_i = 1$, as defined in Equation 3.2. Here, the first term containing $\hat{f}_k$ represents the "greedy" term that penalizes selection of kernel vertices away from $\mathbf{x}_\mathrm{s}$ and $\mathbf{x}_\mathrm{g}$ and subsequently generating samples in those regions.

$$q_k = \lambda_1 \frac{\hat{f}_k - \hat{f}_\mathrm{min}}{\hat{f}_\mathrm{max} - \hat{f}_\mathrm{min}} + \lambda_2 \frac{p_k - p_\mathrm{min}}{p_\mathrm{max} - p_\mathrm{min}} + \lambda_3 \frac{d_k - d_\mathrm{min}}{d_\mathrm{max} - d_\mathrm{min}}. \tag{3.2}$$

The graph density (third) term promotes sampling in the regions that are relatively unexplored. The second term adds random exploratory behavior and brings uniformity in the selection of kernels. Different behaviors can be extracted by modulating the values of $\lambda_1, \lambda_2, \lambda_3$. A high value of λ_1 would focus search, minimizing exploratory behavior. Increasing λ_3 biases the search outwards (towards the leaf vertices), in the regions of low

graph density and prioritizes exploration. A very high value of λ_2 would lead to random selection of kernel vertices. As described in the previous section, a new kernel is initialized only if it is more than a threshold distance away from the existing kernels. Hence, the new kernel would have low d and p values. This results in bias towards choosing new kernel vertices and exploring new regions.

Algorithm 3: Kernel Selection Algorithm

chooseKernel (K_{inf},$\mathcal{G}$)**:**
- $\hat{f}_{\max} \leftarrow \max_{k \in K_{\text{inf}}} \hat{f}_k$, $\hat{f}_{\min} \leftarrow \min_{k \in K_{\text{inf}}} \hat{f}_k$;
- $p_{\max} \leftarrow \max_{k \in K_{\text{inf}}} p_k$, $p_{\min} \leftarrow \min_{k \in K_{\text{inf}}} p_k$;
- $d_{\max} \leftarrow \max_{k \in K_{\text{inf}}} d_k$, $d_{\min} \leftarrow \min_{k \in K_{\text{inf}}} d_k$;
- $q^* \leftarrow \infty$;
- **foreach** $k \in K_{\text{inf}}$ **do**
  - $q_k = \lambda_1 \frac{\hat{f}_k - \hat{f}_{\min}}{\hat{f}_{\max} - \hat{f}_{\min}} + \lambda_2 \frac{p_k - p_{\min}}{p_{\max} - p_{\min}} + \lambda_3 \frac{d_k - d_{\min}}{d_{\max} - d_{\min}}$;
  - **if** $q_k < q^*$ **then**
    - $q^* \leftarrow q_k$; $k^* \leftarrow k$;
- $p_{k^*} \leftarrow p_{k^*} + 1$;
- **return** k^*;

Algorithm 4: Step Limit for informed Sampling

InformedStepLimit ($\mathbf{p}, \hat{\mathbf{e}}, \mathbf{S}$)**:**
- $\mathbf{p'} \leftarrow \mathbf{p} - \mathbf{x}_{\text{centre}}$;
- $\gamma_{\text{inf}} \leftarrow \frac{-(\hat{\mathbf{e}}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{p'}) + \sqrt{(\hat{\mathbf{e}}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{p'})^2 + (1 - \mathbf{p'}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{p'})(\hat{\mathbf{e}}^{\mathrm{T}}\mathbf{S}^{-1}\hat{\mathbf{e}})}}{(\hat{\mathbf{e}}^{\mathrm{T}}\mathbf{S}^{-1}\hat{\mathbf{e}})}$;
- **return** γ_{inf};

3.2.2 Sample Generation

After a kernel k is chosen, a new sample is generated by deciding the direction and magnitude of travel from $\boldsymbol{\mu}_k$. The magnitude of the travel in a given direction needs to be restricted so as to confine the samples to $\mathcal{X}_{\text{inf}}$. Consider a point $\mathbf{p} \in \mathcal{X}_{\text{inf}}$ and a direction vector $\hat{\mathbf{e}}$. Let the maximum distance that can be travelled from point $\mathbf{p}$ in the direction $\hat{\mathbf{e}}$

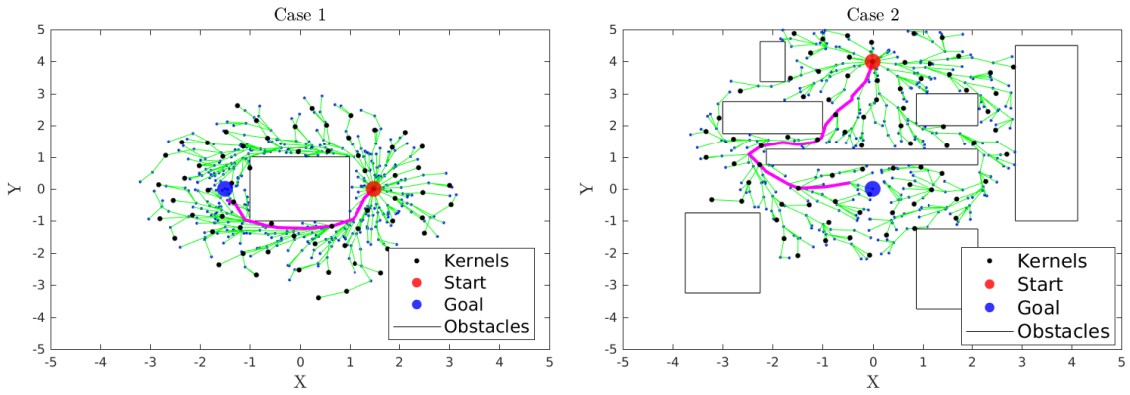


Figure 3.2: Single obstacle environment (Left); Multiple obstacle environment (Right). The figure shows 500 iterations of NP-Informed sampling in both cases.

until the surface of hyper-ellipsoid (defined in Equation 2.8) is reached be γ_{inf}. Thus,

$$(\mathbf{p} + \gamma_{\text{inf}}\hat{\mathbf{e}} - \mathbf{x}_{\text{c}})^{\text{T}}\mathbf{S}^{-1}(\mathbf{p} + \gamma_{\text{inf}}\hat{\mathbf{e}} - \mathbf{x}_{\text{c}}) = 1, \tag{3.3}$$

Let, $\mathbf{p}' = \mathbf{p} - \mathbf{x}_{\text{c}}$. Then, simplifying the above equation, we get

$$\gamma_{\text{inf}}^2(\hat{\mathbf{e}}^{\text{T}}\mathbf{S}^{-1}\hat{\mathbf{e}}) + \gamma_{\text{inf}}(2\hat{\mathbf{e}}^{\text{T}}\mathbf{S}^{-1}\mathbf{p}') + (\mathbf{p}'^{\text{T}}\mathbf{S}^{-1}\mathbf{p}' - 1) = 0. \tag{3.4}$$

Solving this quadratic equation, we obtain two solutions. Only the positive solution is considered, as we wish to move in the direction given by $\hat{\mathbf{e}}$. Thus,

$$\gamma_{\text{inf}} = \frac{-(\hat{\mathbf{e}}^{\text{T}}\mathbf{S}^{-1}\mathbf{p}') + \sqrt{(\hat{\mathbf{e}}^{\text{T}}\mathbf{S}^{-1}\mathbf{p}')^2 + (1 - \mathbf{p}'^{\text{T}}\mathbf{S}^{-1}\mathbf{p}')(\hat{\mathbf{e}}^{\text{T}}\mathbf{S}^{-1}\hat{\mathbf{e}})}}{(\hat{\mathbf{e}}^{\text{T}}\mathbf{S}^{-1}\hat{\mathbf{e}})}. \tag{3.5}$$

In the proposed algorithm, we consider the point $\mathbf{p}$ to be the kernel vertex state $\boldsymbol{\mu}_k$ for $k \in K_{\text{inf}}$. Hence, from equations Equation 2.8, Equation 3.1 we have $\mathbf{p}'^{\text{T}}\mathbf{S}^{-1}\mathbf{p}' < 1$ in Equation 3.5. Thus, γ_{inf} will always have a real, positive value.

The sample generation procedure is outlined in algorithm 10. A random exploration direction is generated with probability ϵ_s. Otherwise, the set of past directions O_k that resulted in collision for kernel k are considered. Before an initial solution is found, the

Algorithm 5: Sample Generation Algorithm

generateSample (k) **:**
  while *true* **do**
    $u_{\text{rand}} \sim \mathcal{U}[0, 1)$;
    if $u_{\text{rand}} < \epsilon_s$ **then**
      $\mathbf{e}^* \leftarrow \text{generateDirection}()$
    else
      for $i = 1 : T$ **do**
        $\mathbf{e}_i \leftarrow \text{generateDirection}()$;
        foreach $\mathbf{e}_{\text{obs}} \in O_k$ **do**
          if $(\mathbf{e}_i^{\text{T}} \mathbf{e}_{\text{obs}}) > u_{\text{obs}}$ **then**
            $u_{\text{obs}} \leftarrow (\mathbf{e}_i^{\text{T}} \mathbf{e}_{\text{obs}})$;
        if $u_{\text{obs}} < u^*$ **then**
          $u^* \leftarrow u_{\text{obs}}$; $\mathbf{e}^* \leftarrow \mathbf{e}_i$;
    $\gamma_{\text{inf}} \leftarrow \text{InformedStepLimit}(\boldsymbol{\mu}_k, \mathbf{e}^*, \mathbf{S})$;
    $u_{\text{rand}} \sim \mathcal{U}[0, 1)$;
    $\gamma \leftarrow (u_{\text{rand}})^{\frac{1}{d}} \min(\gamma_k, \gamma_{\text{inf}})$;
    $\mathbf{x}_{\text{rand}} \leftarrow \boldsymbol{\mu}_k + \gamma\, \mathbf{e}^*$;
    if $\text{IsFree}(\boldsymbol{x}_{\text{rand}})$ **then**
      return $\mathbf{x}_{\text{rand}}$;
    else
      $O_k \leftarrow O_k \cup \{\mathbf{e}^*\}$;
      $k \leftarrow \text{random}(K_{\text{inf}})$; $p_k \leftarrow p_k + 1$;

non-greedy direction $(\boldsymbol{\mu}_k - \mathbf{x}_{\text{goal}})/\|\boldsymbol{\mu}_k - \mathbf{x}_{\text{goal}}\|_2$ for kernel k, is also considered as an obstacle direction in order to bias the exploration efforts to reach $\mathcal{X}_{\text{goal}}$. A number of candidate directions T are produced and the dot product is calculated with each direction vector in O_k. The cost associated with a candidate direction $\mathbf{e}_i$ is the maximum of these dot product values (algorithm 10, lines 9-11). The candidate direction with the least cost is then chosen for exploration (algorithm 10, lines 12-13). This direction is used to calculate γ_{inf}, the maximum step-size, that will ensure that samples remain in $\mathcal{X}_{\text{inf}}$. The minimum of γ_{inf} and γ_k is considered to calculate the travel magnitude (algorithm 10, line 16). Here, γ_k represents the radius of the ball around $\boldsymbol{\mu}_k$ in which new samples would be generated if $\gamma_k < \gamma_{\text{inf}}$. The sample generation algorithm terminates if the new sample $\mathbf{x}_{\text{rand}} \in \mathcal{X}_{\text{free}}$, else this direction of travel is stored in O_k and a new kernel vertex in K_{inf} is chosen randomly to generate a new sample (algorithm 10, line 22).

3.3 Experiments and Discussion

The performance of the NP-Informed sampling was benchmarked against conventional Informed Sampling and Uniform rejection sampling. The above exploration strategies were paired with local rewiring for exploitation. A C++ implementation of the above samplers was used (Informed sampling implementation based on the open-source OMPL version [54]). All experiments were run on a 64-bit PC with 64 GB RAM and Intel Xeon(R) Processor. The operating system used was Ubuntu 16.04. Data was recorded over 100 trials for all the cases. The algorithms were tested in a single obstacle and a multiple obstacle setting in $\mathbb{R}^2, \mathbb{R}^3, \mathbb{R}^4$ and $\mathbb{R}^6$ (Figure 3.2). A (hyper)cube problem environment $\mathcal{X}$ of width 10 units was considered in both cases. In the first case, a single (hyper)cube obstacle with width of 2 units was placed with its center at the origin. The obstacle lies between the start and the goal point, $\mathbf{x}_{\text{init}} = [1.5, 0, ..., 0]^{\text{T}}$, $\mathbf{x}_{\text{goal}} = [-1.5, 0, ..., 0]^{\text{T}}$. The second case consists of multiple obstacles with $\mathbf{x}_{\text{init}} = [0, 4, ..., 0]^{\text{T}}$, $\mathbf{x}_{\text{goal}} = [0, 0, ..., 0]^{\text{T}}$.

The 2D environment in case 2 was extended to higher dimensions by imparting a length

of 2 units (symmetrically) to each obstacle in dimensions $d \geq 3$. A goal bias of 10% was used in Informed and Uniform rejection sampling. Experiments were performed with following the step-sizes (maximum edge length) η: $0.3, 0.6, 1., 2.$ for $\mathbb{R}^2$, $\mathbb{R}^3$, $\mathbb{R}^4$, $\mathbb{R}^6$ respectively. Simulations were run for $2, 3, 4, 8$ seconds in $\mathbb{R}^2$, $\mathbb{R}^3$, $\mathbb{R}^4$, $\mathbb{R}^6$ respectively. The parameters of the proposed algorithm were set as follows. All simulations were performed with $\epsilon_k = 0.5$, i.e., 50 % split between conventional Informed sampling and NP-Informed sampling. Weighting constants $(\lambda_1, \lambda_2, \lambda_3)$ in Equation 3.2 were set to $(3/9, 2/9, 4/9)$ before an initial solution is found and then changed to $(0, 1/2, 1/2)$. This causes the sampler to first focus on finding a good initial solution and then prioritize exploring the informed set. The probability of generating a random direction ϵ_s (Alg. algorithm 10, line 4) was initialized to 0 and changed to 0.5 after an initial solution was found. The maximum magnitude of travel for generating a random sample γ_k was initialized to 1.5η for every kernel and then decreased by a small quantity $(\Delta\gamma = 0.01)$ every time a sampling attempt was made from that kernel. The threshold distance for initializing a new kernel vertex: (η') was set to η. The number of candidate directions generated T, were 5 in $\mathbb{R}^2$, $\mathbb{R}^3$ and 10 in $\mathbb{R}^4$, $\mathbb{R}^6$.

The simulation results are summarized in Table 3.1, Table 3.2 and Figure 3.3. Table 3.1 and Table 3.2 document the average time (in milliseconds) until the first solution was found and its corresponding length, while the numbers in the brackets signify the standard deviation. It can be seen that the NP-Informed sampling consistently finds a first solution of better quality (lesser length) and with smaller standard deviation than either Informed sampling or Uniform rejection sampling. This can be attributed to the use of heuristics during the kernel vertex selection, which focuses and prioritizes the search.

The convergence behavior of all three algorithms is illustrated in Figure 3.3. The plots show that NP-Informed sampling gets a "head start" over the other two samplers, as it finds first solution of better quality. The figure also shows the plot of collision fraction versus

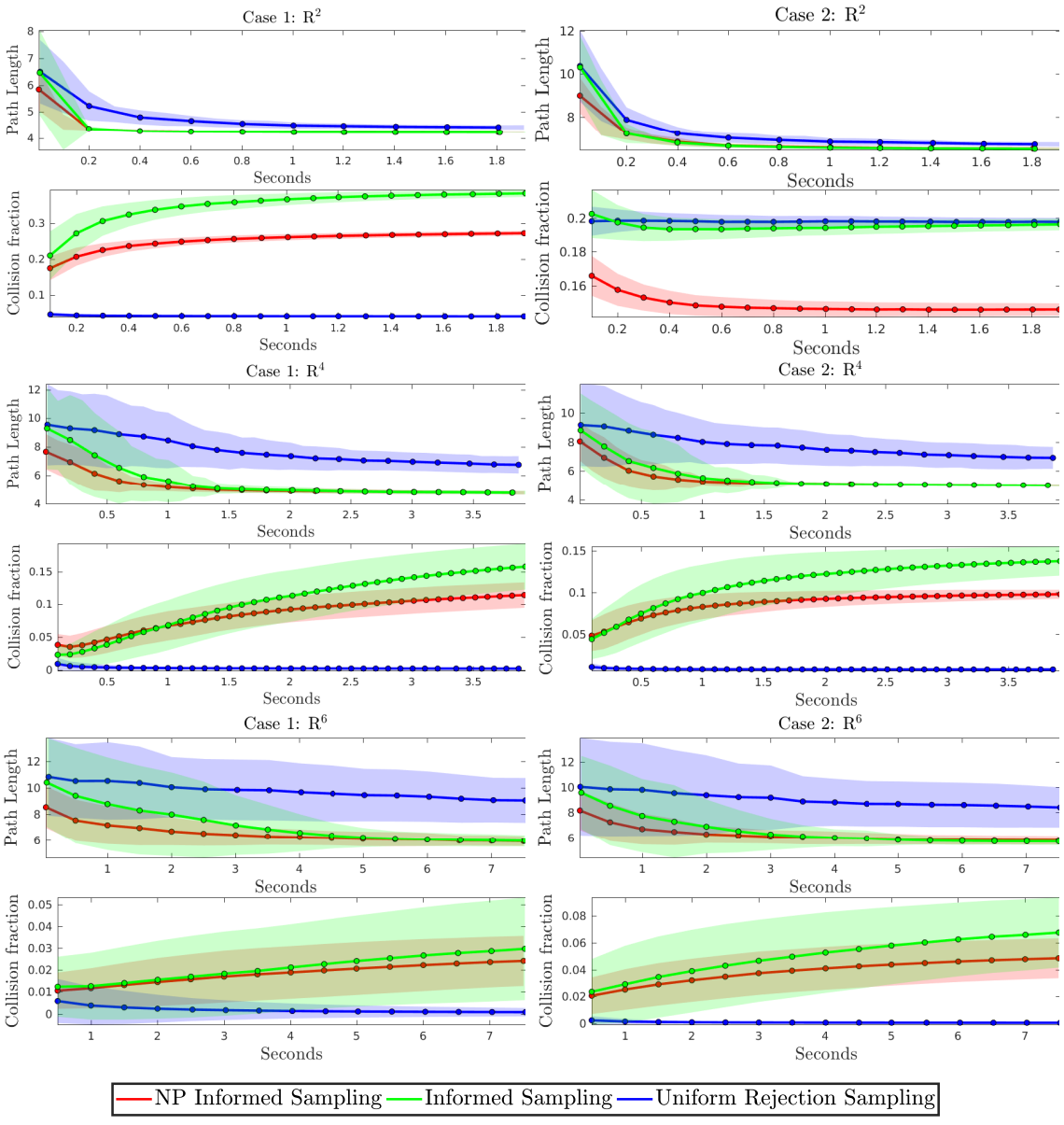


Figure 3.3: Convergence behavior and fraction of samples generated in obstructed space for case 1 and 2 in $\mathbb{R}^2$, $\mathbb{R}^4$ and $\mathbb{R}^6$. Results are averaged over 100 trials and the standard deviation is shaded

Table 3.1: Case 1:Average time/length of first solution

Dimension		2D	3D	4D	6D
Step Size		0.3	0.6	1.0	2.0
Time (ms)	Inf	7.7(4.8)	10.4(6.1)	11.5(17.5)	50.1(94.2)
	NP-Inf	4.3(1.7)	3.2(1.6)	2.6(1.5)	37.9(43.9)
Length	Inf	6.5(1.6)	8.0(1.6)	9.3(2.8)	10.4(3.4)
	NP-Inf	5.8(0.8)	7.0(1.0)	7.7(1.3)	8.5(1.6)

Table 3.2: Case 2:Average time/length of first solution

Dimension		2D	3D	4D	6D
Step Size		0.3	0.6	1.0	2.0
Time (ms)	Inf	17.2(6.6)	13.6(10.8)	9.2(11.2)	51.4(142.6)
	NP-Inf	16.7(6.4)	4.9(2.8)	3.3(2.5)	30.2(33.3)
Length	Inf	10.3(1.5)	8.6(2.1)	8.8(2.5)	9.6(2.9)
	NP-Inf	9.0(0.8)	7.6(1.2)	8.0(1.3)	8.2(1.6)

time, where

$$\text{Collision Fraction} = \frac{\#\text{Positive collisions}}{\#\text{Calls to collision checker}}. \tag{3.6}$$

Thus, the collision fraction represents the fraction of samples generated in $\mathcal{X}_{\text{obs}}$ out of all the samples generated. Theoretically, the probability of generating a sample in $\mathcal{X}_{\text{obs}}$ would be $\lambda(\mathcal{X}_{\text{obs}})/\lambda(\mathcal{X})$ for uniform rejection sampling and $\lambda(\mathcal{X}_{\text{obs}} \cap \mathcal{X}_{\text{inf}})/\lambda(\mathcal{X}_{\text{inf}})$ for Informed Sampling. Here, $\mathcal{X}_{\text{inf}}$ represents the set of points contained inside the hyper-ellipsoid defined in Equation 2.8 and $\lambda(S)$ denotes the Lebesgue measure of the set $S \subset \mathbb{R}^n$. However, the proposed method generates new samples in the vicinity of kernel vertices that are initialized in $\mathcal{X}_{\text{free}}$. Moreover, past collision data is used to avoid sampling in the obstacle space. This results in the reduction of the collision fraction for NP-Informed sampling. After the NP-Informed sampling and the Informed Sampling converge to similar path lengths (which means that $\lambda(\mathcal{X}_{\text{inf}})$ is comparable for both), the NP-Informed sampling shows lesser collision fraction than Informed Sampling. This can be seen in Figure 3.3, especially Case2: $\mathbb{R}^2$, where $\lambda(\mathcal{X}_{\text{obs}})/\lambda(\mathcal{X})$ is highest. Rejection sampling shows lowest collision fraction in higher dimensions as the ratio $\lambda(\mathcal{X}_{\text{obs}})/\lambda(\mathcal{X})$ diminishes rapidly.

The ideas proposed in this chapter are extended in the next to efficiently sample the "Relevant Region set" defined in [55]. The relevant region set considers the estimate of

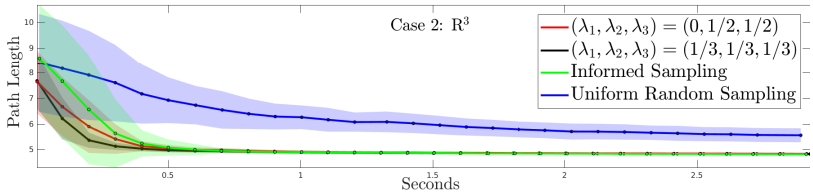


Figure 3.4: Convergence for NP-Informed sampling with different weights in Equation 3.2 set after an initial solution is found

g value (sum of edge-costs from start state) for cost-to-come instead of the heuristic approximation used in Informed Set (Equation 2.7). Relevant region does not have a 100% recall like the Informed set. However, it can possibly have a higher precision (probability of sampling a point in the set that improves the current solution) as the iterations progress. It also reduces the dependence on heuristics, which can be misleading in many cases.

CHAPTER 4
RELEVANT REGION EXPLORATION

4.1 Motivation

The SBMP algorithms and the exploration strategies mentioned previously are traditionally geared towards finding the (length) optimal path in uniform cost spaces. This includes the NP-Informed Sampling approach proposed in the previous chapter. However, many applications require planning algorithms to find the optimal path with respect to a provided cost function. These include the problem of navigation on a rough terrain for a mobile robot (see Figure 4.6), safety critical path planning with clearance cost-map (example in Figure 4.7), human aware motion planning [56], and planning on energy landscapes [57]. The Transition-based RRT (T-RRT) algorithm [13] takes a user-defined cost function as an additional input and adds a transition test based on the Metropolis criterion to accept or reject potential new states. The transition test favors exploration of low-cost regions of space and leads to better quality paths. An enhanced, bi-directional version of T-RRT is presented in [58]. Berenson et al [59] combine gradient information within the T-RRT framework to address the issue of navigating cost-space chasms. Finally, Devaurs et al [60] combine the filtering properties of the transition test with the local rewiring procedure of RRT* to obtain the asymptotically optimal T-RRT* algorithm. While the transition test promotes exploration of low-cost regions, unlike the Informed set, it does not focus exploration on to a subset of the search-space based on the current solution. Secondly, the probabilistic rejection strategy of the transition test might not scale well to higher dimensional spaces, as the probability of generating a "good" sample that can pass the transition test may decrease rapidly. The "Relevant Region" sampler proposed in this chapter addresses these issues by employing a generative sampling approach. It utilizes heuristics, the current solution

cost and the cost function information to effectively focus the search in general cost-space settings.

The L_2-Informed Set includes all points that can potentially improve the current solution. Exploration outside this set is thus redundant. The Lebesgue measure of the L_2-Informed set decreases as the solution improves, leading to a focused search. However, Informed Sampling effectively resorts to uniform random sampling if the Lebesgue measure of the Informed Set is comparable to that of the entire search space. This can happen if the heuristic estimate of the solution cost fails to provide a good enough approximation of the true solution cost. This work addresses these issues by utilizing cost-to-come information from the planner's tree structure and reducing dependence on heuristics.

The Relevant Region, introduced in [55], leverages the current solution cost and the planner's tree structure information to focus search. A selective vertex inclusion procedure [55] and a machine learning approach [61] has been proposed to generate new samples in the Relevant Region. However, these approaches fall into the category of rejection sampling methods, which do not scale well for high dimensional problems. The current work rigorously defines the Relevant Region set, analyzes its theoretical properties and presents a *generative* method to sample it. This work also extends the Relevant Region framework for planning on general cost-maps.

4.2 Relevant Region Set

Consider the set of *relevant vertices* defined as

$$V_{\mathrm{rel}} = \{\mathbf{v} \in V \mid \mathrm{g}_{\mathcal{T}}(\mathbf{v}) + \mathrm{h}(\mathbf{v}, \mathbf{x}_{\mathrm{g}}) < c_i\}. \tag{4.1}$$

Let $\epsilon > 0$ ball around a relevant vertex $\mathbf{v} \in V_{\mathrm{rel}}$ be defined as

$$\mathcal{B}^{\epsilon}(\mathbf{v}) = \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{v}\|_2 < \epsilon,\ \mathbf{v} \in V_{\mathrm{rel}}\}. \tag{4.2}$$

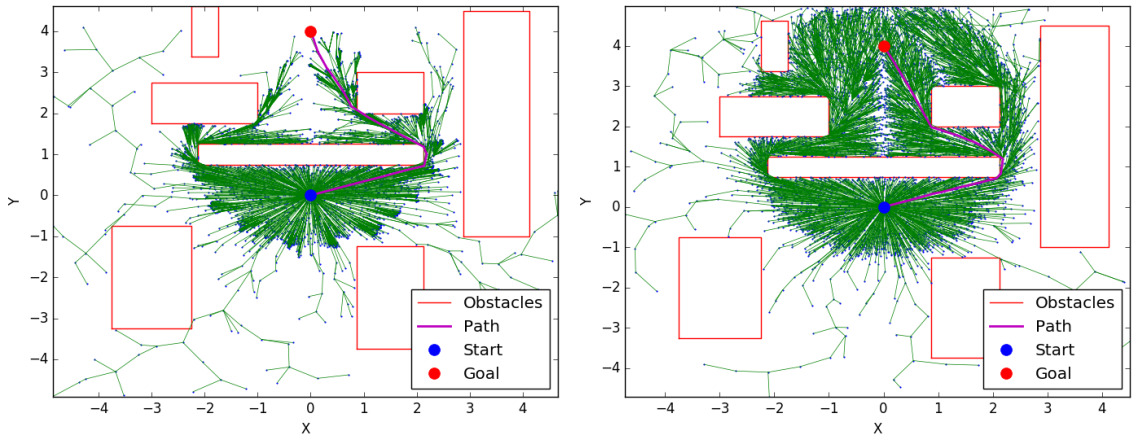


Figure 4.1: Planning in a multiple obstacle environment with Relevant Region sampling (left) and Informed Sampling (right). Note that the Relevant Region focuses on two pertinent homotopy classes whereas the Informed Sampling generates uniform samples inside the ellipsoidal region.

Consider the estimate of the solution cost constrained to pass through $\mathbf{x} \in \mathcal{B}^{\epsilon}(\mathbf{v})$

$$\hat{f}_{\mathbf{v}}(\mathbf{x}) = \mathrm{c}_{\ell}(\mathbf{v},\mathbf{x}) + \mathrm{g}_{\mathcal{T}}(\mathbf{v}) + \mathrm{h}(\mathbf{x}, \mathbf{x}_{\mathrm{g}}). \tag{4.3}$$

The *Relevant Set* around $\mathbf{v} \in V_{\mathrm{rel}}$ is defined as

$$\mathcal{B}^{\epsilon}_{\mathrm{rel}}(\mathbf{v}) = \{\mathbf{x} \in \mathcal{B}^{\epsilon}(\mathbf{v}) \mid \hat{f}_{\mathbf{v}}(\mathbf{x}) < c_i\}. \tag{4.4}$$

Using Equation 4.1, Equation 4.4, the *Relevant Region* is defined as the union of the relevant sets around all relevant vertices

$$\mathcal{X}^{\epsilon}_{\mathrm{rel}} = \bigcup_{\mathbf{v} \in V_{\mathrm{rel}}} \mathcal{B}^{\epsilon}_{\mathrm{rel}}(\mathbf{v}). \tag{4.5}$$

Note that, in contrast to $\mathcal{X}_{\mathrm{inf}}$ which uses the heuristic estimate $\mathrm{h}(\mathbf{x}_{\mathrm{s}}, \mathbf{x})$ of the cost-to-come, $\mathcal{X}^{\epsilon}_{\mathrm{rel}}$ uses $\mathrm{c}_{\ell}(\mathbf{v},\mathbf{x}) + \mathrm{g}_{\mathcal{T}}(\mathbf{v})$ from Equation 4.3. This approximation considers the cost-function information see Equation 2.4, the structure of $\mathcal{T}$, and hence the topology of $\mathcal{X}_{\mathrm{free}}$. While the L_2-norm is still a consistent heuristic for cost-maps with $C(\mathbf{x}) \geq 1$ for all $\mathbf{x} \in \mathcal{X}$,

it does not take into account C or $\mathcal{X}_{\text{obs}}$. It may provide a poor estimate of the solution cost, leading to $\lambda(\mathcal{X}_{\text{inf}}) \approx \lambda(\mathcal{X})$. Informed Sampling effectively resorts to uniform random sampling in this case. The set $\mathcal{X}_{\text{rel}}^{\epsilon}$ alleviates this dependence on a heuristic. The value of ϵ, which controls the size of the Relevant Set, is taken to be slightly greater than the step-size parameter η (in our benchmarking simulations, we used $\epsilon = 1.5\eta$). The step-size parameter η in SBMP controls the maximum edge length in $\mathcal{G}$ [12]. Note that a very small value of ϵ would hinder exploration, while a large value of ϵ may provide a poor estimate of the cost-to-come in Equation 4.3, as the edge $(\mathbf{x}, \mathbf{v})$ may not be feasible. The following theorem proves that for any $\epsilon > 0$, $\mathcal{B}_{\text{rel}}^{\epsilon}(\mathbf{v})$ is not a singleton.

Theorem 1. *For every* $\mathbf{v} \in V_{\text{rel}}$*, there exists* $\delta > 0$*, such that, for all* $\mathbf{x} \in \mathcal{B}^{\delta}(\mathbf{v})$*, it follows that* $\hat{f}_{\mathbf{v}}(\mathbf{x}) < c_i$.

Proof. Note from Equation 4.3 that for each given $\mathbf{v} \in V_{\text{rel}}$ the function $\hat{f}_{\mathbf{v}}$ is continuous in $\mathbf{x}$ since both $\mathrm{c}_{\ell}(\cdot, \mathbf{v}), \mathrm{h}(\cdot, \mathbf{x}_{\mathrm{g}})$ are continuous. Also, $\hat{f}_{\mathbf{v}}(\mathbf{v}) = \mathrm{g}_{\mathcal{T}}(\mathbf{v}) + \mathrm{h}(\mathbf{v}, \mathbf{x}_{\mathrm{g}}) < c_i$ since $\mathbf{v} \in V_{\text{rel}}$. Since $\hat{f}_{\mathbf{v}}$ is continuous at $\mathbf{v}$, it follows that for any $\zeta > 0$ there exists $\delta > 0$ such that $\mathbf{x} \in \mathcal{B}^{\delta}(\mathbf{v})$ implies that $|\hat{f}_{\mathbf{v}}(\mathbf{x}) - \hat{f}_{\mathbf{v}}(\mathbf{v})| < \zeta$. Choosing $\zeta = c_i - \hat{f}_{\mathbf{v}}(\mathbf{v}) > 0$ one then obtains that for all $\mathbf{x} \in \mathcal{B}^{\delta}(\mathbf{v})$ we have that $|\hat{f}_{\mathbf{v}}(\mathbf{x}) - \hat{f}_{\mathbf{v}}(\mathbf{v})| < c_i - \hat{f}_{\mathbf{v}}(\mathbf{v})$ and hence $\hat{f}_{\mathbf{v}}(\mathbf{x}) < c_i$. □

Corollary 2. *Let* $\mathbf{v} \in V_{\text{rel}}$*. For every* $\epsilon > 0$ *there exists* $\delta > 0$ *such* $\mathcal{B}^{\delta}(\mathbf{v}) \subset \mathcal{B}_{\text{rel}}^{\epsilon}(\mathbf{v})$.

Theorem 3. *For any* $\epsilon > 0$*, the Relevant Region* $\mathcal{X}_{\text{rel}}^{\epsilon}$ *is a subset of the Informed Set* $\mathcal{X}_{\text{inf}}$.

Proof. Let $\mathbf{x} \in \mathcal{X}_{\text{rel}}^{\epsilon}$. Then there exists $\mathbf{v} \in V_{\text{rel}}$, so that $\mathbf{x} \in \mathcal{B}_{\text{rel}}^{\epsilon}(\mathbf{v})$, and hence $\mathrm{c}_{\ell}(\mathbf{x},\mathbf{v}) + \mathrm{g}_{\mathcal{T}}(\mathbf{v}) + \mathrm{h}(\mathbf{x}, \mathbf{x}_{\mathrm{g}}) < c_i$. Since the heuristic function is consistent, $\mathrm{h}(\mathbf{x},\mathbf{v}) < \mathrm{c}_{\ell}(\mathbf{x},\mathbf{v})$ and $\mathrm{h}(\mathbf{v}, \mathbf{x}_{\mathrm{s}}) < \mathrm{g}_{\mathcal{T}}(\mathbf{v})$. Using the triangle inequality, it follows that, $\mathrm{h}(\mathbf{x}_{\mathrm{s}}, \mathbf{x}) < \mathrm{h}(\mathbf{x},\mathbf{v}) + \mathrm{h}(\mathbf{v}, \mathbf{x}_{\mathrm{s}})$. Combining the above inequalities yields $\mathrm{h}(\mathbf{x}_{\mathrm{s}}, \mathbf{x}) + \mathrm{h}(\mathbf{x}, \mathbf{x}_{\mathrm{g}}) < \mathrm{c}_{\ell}(\mathbf{x},\mathbf{v}) + \mathrm{g}_{\mathcal{T}}(\mathbf{v}) + \mathrm{h}(\mathbf{x}, \mathbf{x}_{\mathrm{g}}) < c_i$. Hence, $\mathbf{x} \in \mathcal{X}_{\text{inf}}$. It follows that $\mathcal{X}_{\text{rel}}^{\epsilon} \subset \mathcal{X}_{\text{inf}}$. □

Theorem 3 implies that generating samples in $\mathcal{X}_{\text{rel}}^{\epsilon}$ does not lead to redundant exploration outside $\mathcal{X}_{\text{inf}}$. However, note that sampling in $\mathcal{X}_{\text{rel}}^{\epsilon}$ is not a necessary condition for

improving the current solution, i.e., there may be points $\mathbf{x} \in \mathcal{X}_{\text{inf}}$ such that $\mathbf{x} \notin \mathcal{X}^{\epsilon}_{\text{rel}}$ which may improve the current solution. Relevant Region sampling is thus utilized in conjunction with Informed/Uniform Sampling. As shown in the numerical examples later on, this interplay of exploration by Informed Sampling, combined with focusing properties of Relevant Region, leads to accelerated convergence.

4.3 Relevant Region Sampling Algorithm

Since $\mathcal{X}^{\epsilon}_{\text{rel}}$ depends on $\mathcal{T}$, a direct sampling strategy is not possible. Hence, the proposed sampling strategy proceeds by first selecting a relevant vertex $\mathbf{v}_{\text{p}} \in V_{\text{rel}}$, sampling a random direction $\hat{\mathbf{e}}$, $\|\hat{\mathbf{e}}\|_2 = 1$ and finding the maximum magnitude of travel $\gamma_{\text{rel}} > 0$ along $\hat{\mathbf{e}}$, so that for all $\gamma \in (0, \gamma_{\text{rel}})$ the new sample $\mathbf{x} = \mathbf{v}_{\text{p}} + \gamma\hat{\mathbf{e}} \in \mathcal{B}^{\epsilon}_{\text{rel}}(\mathbf{v}_{\text{p}})$. Please see Figure 4.2. Note that Theorem 1 guarantees the existence of γ_{rel}. Concretely, the following optimization problem needs to be solved:

$$\begin{aligned} &\sup_{\gamma \in (0,\epsilon)} \ \gamma, \\ &\text{subject to: } \hat{f}_{\mathbf{v}_{\text{p}}}(\mathbf{v}_{\text{p}} + \gamma\hat{\mathbf{e}}) < c_i. \end{aligned} \tag{4.6}$$

4.3.1 Case 1: Uniform Cost-Map

Consider the problem Equation 4.6 with $C(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathcal{X}$. Using the L_2-norm heuristic in Equation 4.3, the inequality in Equation 4.6 yields,

$$\hat{f}_{\mathbf{v}_{\text{p}}}(\mathbf{v}_{\text{p}} + \gamma\hat{\mathbf{e}}) = \gamma + \text{g}_{\mathcal{T}}(\mathbf{v}_{\text{p}}) + \|\mathbf{v}_{\text{p}} + \gamma\hat{\mathbf{e}} - \mathbf{x}_{\text{g}}\|_2 < c_i. \tag{4.7}$$

Rearrange the terms in Equation 4.7 to obtain

$$\|\mathbf{v}_{\text{p}} + \gamma\hat{\mathbf{e}} - \mathbf{x}_{\text{g}}\|_2 < c_i - \text{g}_{\mathcal{T}}(\mathbf{v}_{\text{p}}) - \gamma. \tag{4.8}$$

To ensure that the RHS in Equation 4.8 is positive, choose

$$\gamma < c_i - \mathrm{g}_{\mathcal{T}}(\mathbf{v}_\mathrm{p}). \tag{4.9}$$

Let $\mathbf{x}_\mathrm{pg} = \mathbf{v}_\mathrm{p} - \mathbf{x}_\mathrm{g}$ and $g_\mathrm{gp} = c_i - \mathrm{g}_{\mathcal{T}}(\mathbf{v}_\mathrm{p})$. Also note that $\mathbf{x}_\mathrm{pg}^\mathsf{T}\mathbf{x}_\mathrm{pg} = \mathrm{h}^2(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})$ and $\mathbf{x}_\mathrm{pg}^\mathsf{T}\hat{\mathbf{e}} = \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})\cos\theta$, where θ is the angle between the vectors $\mathbf{x}_\mathrm{pg}$ and $\hat{\mathbf{e}}$. Squaring both sides in Equation 4.8 yields,

$$\mathrm{h}^2(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g}) + 2\gamma\mathbf{x}_\mathrm{pg}^\mathsf{T}\hat{\mathrm{e}} + \gamma^2 < g_\mathrm{gp}^2 - 2\gamma g_\mathrm{gp} + \gamma^2$$
$$\text{and hence } \gamma < \frac{g_\mathrm{gp}^2 - \mathrm{h}^2(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})}{2(\mathbf{x}_\mathrm{pg}^\mathsf{T}\hat{\mathbf{e}} + g_\mathrm{gp})}.$$

Define the RHS in the above inequality as

$$\gamma_\mathrm{uni} = \frac{(c_i - \mathrm{g}_{\mathcal{T}}(\mathbf{v}_\mathrm{p}))^2 - \mathrm{h}^2(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})}{2\big[\mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})\cos\theta + (c_i - \mathrm{g}_{\mathcal{T}}(\mathbf{v}_\mathrm{p}))\big]}. \tag{4.10}$$

Note that $\gamma_\mathrm{uni} > 0$ for $\mathbf{v}_\mathrm{p} \in V_\mathrm{rel}$, and attains its maximum value $\overline{\gamma}_\mathrm{uni}$ at $\theta = \pi$, in which case,

$$\overline{\gamma}_\mathrm{uni} = \big(c_i - \mathrm{g}_{\mathcal{T}}(\mathbf{v}_\mathrm{p}) + \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})\big)/2,$$

and also, $\overline{\gamma}_\mathrm{uni} < c_i - \mathrm{g}_{\mathcal{T}}(\mathbf{v}_\mathrm{p})$ for $\mathbf{v}_\mathrm{p} \in V_\mathrm{rel}$, satisfying Equation 4.9. Thus, the solution to problem Equation 4.6 for uniform cost-map is

$$\gamma_\mathrm{rel} = \min(\gamma_\mathrm{uni}, \epsilon). \tag{4.11}$$

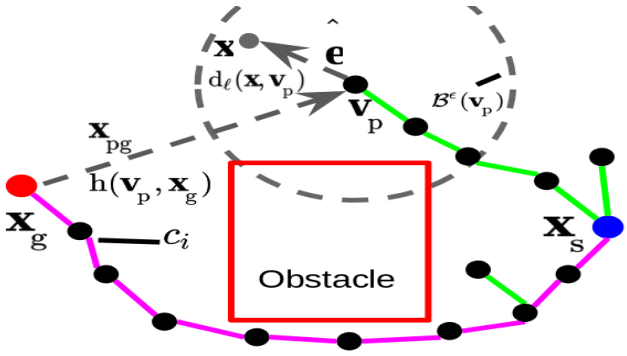


Figure 4.2: A schematic for Relevant Region sampling.

4.3.2 Case 2: General Cost-Maps

Now consider the problem Equation 4.6 with $C(\mathbf{x}) > 1$ for all $\mathbf{x} \in \mathcal{X}$. The following inequality needs to be solved for γ,

$$\gamma \int_0^1 C(\mathbf{v}_\mathrm{p} + \gamma\hat{\mathbf{e}}\mathrm{s})\mathrm{ds} + \mathrm{g}_\mathcal{T}(\mathbf{v}_\mathrm{p}) + \mathrm{h}(\mathbf{v}_\mathrm{p} + \gamma\hat{\mathbf{e}}, \mathbf{x}_\mathrm{g}) < c_i. \tag{4.12}$$

Often, C may not have a tractable closed-form expression and hence the planner has access only to the value of C at any point in the search space. In order to avoid a computationally expensive procedure to solve Equation 4.12, we let

$$\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v}_\mathrm{p} + \gamma\hat{\mathbf{e}}) = \gamma \int_0^1 C(\mathbf{v}_\mathrm{p} + \gamma\hat{\mathbf{e}}\mathrm{s})\mathrm{ds} \approx \gamma C(\mathbf{v}_\mathrm{p}) \tag{4.13}$$

Note that Equation 4.13 uses a zeroth-order approximation of the integrand to estimate the integral. Higher order approximations are possible, but these will result in a computationally more involved process to find γ (see below). It follows from Equation 4.13 that

$$\gamma C(\mathbf{v}_\mathrm{p}) + \mathrm{g}_\mathcal{T}(\mathbf{v}_\mathrm{p}) + \|\mathbf{v}_\mathrm{p} + \gamma\hat{\mathbf{e}} - \mathbf{x}_\mathrm{g}\|_2 < c_i, \tag{4.14}$$

$$\text{or, } \|\mathbf{v}_\mathrm{p} + \gamma\hat{\mathbf{e}} - \mathbf{x}_\mathrm{g}\|_2 < c_i - \mathrm{g}_\mathcal{T}(\mathbf{v}_\mathrm{p}) - \gamma C(\mathbf{v}_\mathrm{p}). \tag{4.15}$$

To ensure that the RHS of Equation 4.15 is positive, choose

$$\gamma < \big(c_i - \mathrm{g}_{\mathcal{T}}(\mathbf{v}_\mathrm{p})\big)/C(\mathbf{v}_\mathrm{p}). \tag{4.16}$$

Let again $\mathbf{x}_\mathrm{pg} = \mathbf{v}_\mathrm{p} - \mathbf{x}_\mathrm{g}$, $g_\mathrm{gp} = c_i - \mathrm{g}_{\mathcal{T}}(\mathbf{v}_\mathrm{p})$, $\mathbf{x}_\mathrm{pg}^\mathsf{T}\mathbf{x}_\mathrm{pg} = \mathrm{h}^2(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})$ and $\mathbf{x}_\mathrm{pg}^\mathsf{T}\hat{\mathbf{e}} = \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})\cos\theta$, where θ is the angle between the vectors $\mathbf{x}_\mathrm{pg}$ and $\hat{\mathbf{e}}$. Squaring both sides in Equation 4.15 and simplifying yields,

$$\gamma^2(C^2(\mathbf{v}_\mathrm{p}) - 1) - 2\gamma(g_\mathrm{gp}C(\mathbf{v}_\mathrm{p}) + \mathbf{x}_\mathrm{pg}^\mathsf{T}\hat{\mathbf{e}}) + g_\mathrm{gp}^2 - \mathrm{h}^2(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g}) > 0. \tag{4.17}$$

Let γ_1, γ_2 be the roots of the quadratic equation corresponding to inequality Equation 4.17, and assume $\gamma_2 > \gamma_1$.

$$\begin{aligned}
\gamma_2 &= \frac{g_\mathrm{gp}C(\mathbf{v}_\mathrm{p}) + \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})\cos\theta + \sqrt{\Delta}}{(C^2(\mathbf{v}_\mathrm{p}) - 1)} \\
\gamma_1 &= \frac{g_\mathrm{gp}C(\mathbf{v}_\mathrm{p}) + \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})\cos\theta - \sqrt{\Delta}}{(C^2(\mathbf{v}_\mathrm{p}) - 1)} \\
\Delta &= (g_\mathrm{gp}C(\mathbf{v}_\mathrm{p}) + \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})\cos\theta)^2 - (C^2(\mathbf{v}_\mathrm{p}) - 1)(g_\mathrm{gp}^2 - \mathrm{h}^2(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})).
\end{aligned} \tag{4.18}$$

The maximum and minimum values of the radicand Δ are obtained at $\theta = 0$ and $\theta = \pi$, respectively, where

$$(g_\mathrm{gp} - \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})C(\mathbf{v}_\mathrm{p}))^2 \leq \Delta \leq (g_\mathrm{gp} + \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})C(\mathbf{v}_\mathrm{p}))^2. \tag{4.19}$$

Hence, $\gamma_1, \gamma_2 \in \mathbb{R}_{\geq 0}$ for $\mathbf{v}_\mathrm{p} \in V_\mathrm{rel}$. Then Equation 4.17 yields,

$$(\gamma - \gamma_1)(\gamma - \gamma_2) > 0. \text{ equivalently, } \gamma > \gamma_2 \text{ or } \gamma < \gamma_1. \tag{4.20}$$

Algorithm 6: Sampling Algorithm

$V \leftarrow \{\mathbf{x}_\mathrm{s}\}$; $E \leftarrow \phi$; $\mathcal{G} \leftarrow (V, E)$;
for $i = 1 : N$ **do**
  $c_i \leftarrow \min_{\mathbf{v} \in V_\mathrm{goal}} \mathrm{g}_\mathcal{T}(\mathbf{v})$;
  $u_\mathrm{rand} \sim \mathcal{U}(0, 1)$;
  if $u_\mathrm{rand} < p_\mathrm{rel}$ and $c_i < \infty$ **then**
    $\mathbf{v}_\mathrm{p} \leftarrow \mathsf{chooseVertex}(V_\mathrm{rel})$;
    $\hat{\mathbf{e}} \leftarrow \mathsf{generateDirection}()$;
    $\gamma_\mathrm{rel} \leftarrow \mathsf{RelevantStepLimit}(\mathbf{v}_\mathrm{p}, \hat{\mathbf{e}})$;
    $u_\mathrm{rand} \sim \mathcal{U}(0, 1)$;
    $\mathbf{x}_\mathrm{rand} \leftarrow \mathbf{v}_\mathrm{p} + (u_\mathrm{rand})^{\frac{1}{d}} \gamma_\mathrm{rel} \hat{\mathbf{e}}$;
  else
    $\mathbf{x}_\mathrm{rand} \leftarrow \mathsf{InformedSampling}()$
  $\mathbf{x}_\mathrm{new} \leftarrow \mathsf{Extend}(\mathbf{x}_\mathrm{rand})$;
  $\mathsf{Exploitation}(\mathcal{G})$;
return $\mathcal{G}$

Consider the larger root γ_2 from Equation 4.18. The minimum value of γ_2 is attained when $\theta = \pi$, so that

$$\gamma_2 \geq \frac{g_\mathrm{gp} C(\mathbf{v}_\mathrm{p}) - \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g}) + |g_\mathrm{gp} - \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g}) C(\mathbf{v}_\mathrm{p})|}{(C^2(\mathbf{v}_\mathrm{p}) - 1)}. \tag{4.21}$$

Define the RHS in Equation 4.21 as $\overline{\gamma}_2$. Simplifying yields,

$$\overline{\gamma}_2 = \begin{cases} \frac{g_\mathrm{gp} + \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})}{C(\mathbf{v}_\mathrm{p}) + 1}, & g_\mathrm{gp} < \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g}) C(\mathbf{v}_\mathrm{p}), \\ \frac{g_\mathrm{gp} - \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g})}{C(\mathbf{v}_\mathrm{p}) - 1}, & g_\mathrm{gp} > \mathrm{h}(\mathbf{v}_\mathrm{p}, \mathbf{x}_\mathrm{g}) C(\mathbf{v}_\mathrm{p}). \end{cases} \tag{4.22}$$

Note that $\overline{\gamma}_2 > g_\mathrm{gp}/C(\mathbf{v}_\mathrm{p})$. This implies $\gamma_2 > g_\mathrm{gp}/C(\mathbf{v}_\mathrm{p})$, violating Equation 4.16. Thus, $\gamma > \gamma_2$ is an infeasible solution of Equation 4.14. Next, consider γ_1. Differentiating with respect to θ, the extrema are obtained at $\theta = 0, \pi$. Calculating the second derivative yields, $\gamma_1''(\theta = 0) > 0$ and $\gamma_1''(\theta = \pi) < 0$. The maximum value of γ_1 obtained at $\theta = \pi$ is given

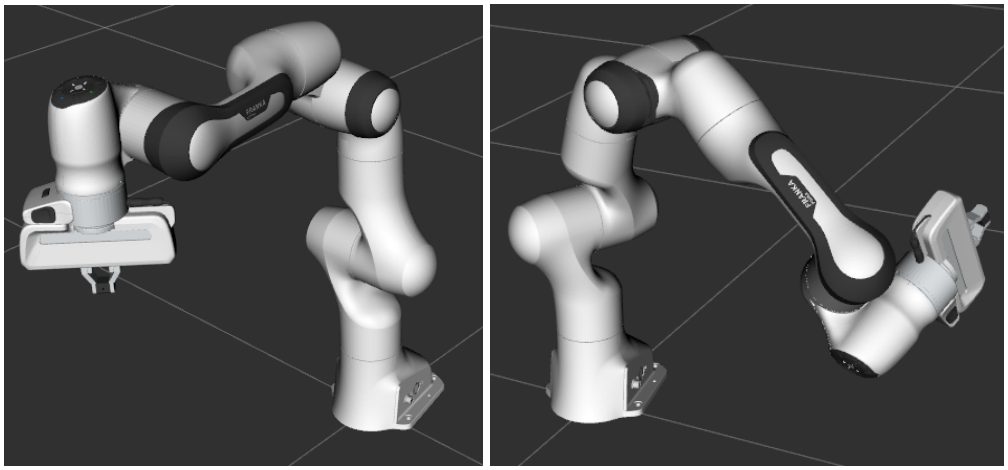


Figure 4.3: Planning for 7 DOF Panda Arm in the joint space from the start state (left) to a given joint goal state (right).

by

$$\overline{\gamma}_1 = \begin{cases} \frac{g_{\mathrm{gp}}+\mathrm{h}(\mathbf{v}_\mathrm{p},\mathbf{x}_\mathrm{g})}{C(\mathbf{v}_\mathrm{p})+1}, & g_{\mathrm{gp}} > \mathrm{h}(\mathbf{v}_\mathrm{p},\mathbf{x}_\mathrm{g})C(\mathbf{v}_\mathrm{p}), \\ \frac{g_{\mathrm{gp}}-\mathrm{h}(\mathbf{v}_\mathrm{p},\mathbf{x}_\mathrm{g})}{C(\mathbf{v}_\mathrm{p})-1}, & g_{\mathrm{gp}} < \mathrm{h}(\mathbf{v}_\mathrm{p},\mathbf{x}_\mathrm{g})C(\mathbf{v}_\mathrm{p}). \end{cases} \tag{4.23}$$

Now, $\overline{\gamma}_1 < g_{\mathrm{gp}}/C(\mathbf{v}_\mathrm{p})$. This implies $\gamma_1 < g_{\mathrm{gp}}/C(\mathbf{v}_\mathrm{p})$. It follows that $\gamma < \gamma_1$ satisfies Equation 4.16. Thus, the solution to problem Equation 4.6 with the approximation in Equation 4.14 is

$$\gamma_{\mathrm{rel}} = \min(\gamma_1, \epsilon). \tag{4.24}$$

For the special case when $\Delta = 0$ and $\gamma_1 = \gamma_2 = \gamma_\mathrm{c}$, inequality Equation 4.17 simplifies to $(\gamma - \gamma_\mathrm{c})^2 > 0$. Considering Equation 4.16 yields $\gamma_{\mathrm{rel}} = \min(g_{\mathrm{gp}}/C(\mathbf{v}_\mathrm{p}), \epsilon)$. Note that if $C(\mathbf{v}_\mathrm{p}) = 1$, then inequality Equation 4.12 reduces to Equation 4.7 and the analysis for uniform cost-maps is applicable.

The outline of the proposed algorithm in given in algorithm 6. The procedure initializes a vertex at the start state $\mathbf{x}_\mathrm{s}$. At every iteration, the current best solution cost c_i is updated (line 3). If a sub-optimal solution exists (c_i is finite), with probability p_{rel} (line 5), Relevant Region sampling is employed to generate a new random sample $\mathbf{x}_{\mathrm{rand}}$. Otherwise, conventional Informed Sampling is used. Relevant Region sampling consists of first choosing a relevant vertex $\mathbf{v}_\mathrm{p}$, generating a random direction $\hat{\mathbf{e}}$ and calculating the maximum magni-

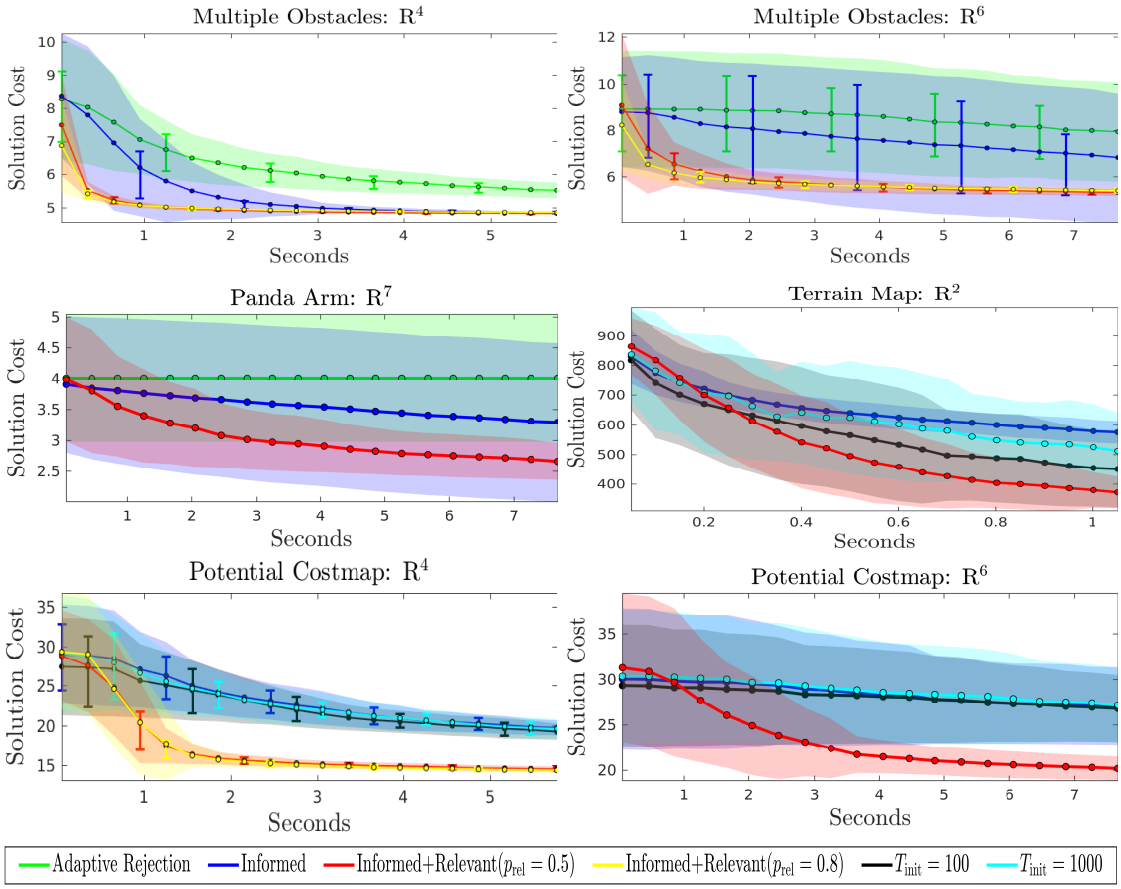


Figure 4.4: Convergence plots for different sampling methods in various test environments. Solid lines indicate the average value and the standard deviation is shaded. Error bar indicate the upper and lower quartiles.

tude of travel along $\hat{\mathbf{e}}$ (line 6-8). If $C(\mathbf{v}_\mathrm{p}) = 1$, then Equation 4.11 is used for obtaining γ_rel along $\hat{\mathbf{e}}$, else Equation 4.24 is used. The exponent $1/d$ (line 10) biases the travel magnitude towards γ_rel and promotes exploration. After $\mathbf{x}_\mathrm{rand}$ is generated, conventional SBMP modules incorporate a new vertex $\mathbf{x}_\mathrm{new}$ in $\mathcal{G}$ (line 13). These include: a) finding the nearest neighbor $\mathbf{x}_\mathrm{nearest}$ to $\mathbf{x}_\mathrm{rand}$ in $\mathcal{G}$; b) local steering from $\mathbf{x}_\mathrm{nearest}$ in the direction of $\mathbf{x}_\mathrm{rand}$ to obtain $\mathbf{x}_\mathrm{new}$; c) ensuring feasibility of edge-connections in the neighborhood of $\mathbf{x}_\mathrm{new}$. This is followed by the exploitation module (local/global rewiring, etc).

The chooseVertex module selects a relevant vertex to be expanded from the set V_rel.

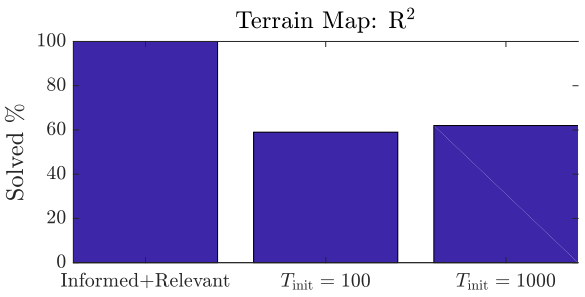


Figure 4.5: Percentage of successful trials (where planner found a feasible solution) with different sampling strategies.

Similar to the procedure in Guided-ESTs [43] a weight $q_{\mathbf{v}}$ is allocated for each $\mathbf{v} \in V_{\text{rel}}$.

$$q_{\mathbf{v}} = \lambda_1 p_{\mathbf{v}} + \lambda_2 d_{\mathbf{v}} + \lambda_3 \big(\mathrm{g}_{\mathcal{T}}(\mathbf{v}) + \mathrm{h}(\mathbf{v}, \mathbf{x}_{\mathrm{g}})\big)/c_i. \tag{4.25}$$

Here, $p_{\mathbf{v}}$ represents the number of times $\mathbf{v}$ has been selected in the past. This penalizes multiple selections and the exploration of the region around a particular vertex. The second term, $d_{\mathbf{v}}$ is the number of edges connected to $\mathbf{v}$. It promotes sampling in relatively unexplored regions. The last term $0 < \big(\mathrm{g}_{\mathcal{T}}(\mathbf{v}) + \mathrm{h}(\mathbf{v}, \mathbf{x}_{\mathrm{g}})\big)/c_i < 1$ is the estimate of the solution cost through $\mathbf{v}$, normalized by the current best cost. This prioritizes exploration of regions with low solution cost estimates. The parameters $(\lambda_1, \lambda_2, \lambda_3) > 0$ modulate the behavior of the selection algorithm. A large value of λ_3 leads to a greedy focus on low solution cost areas, whereas increasing λ_1, λ_2 promotes exploration. A binary heap is used to update and sort V_{rel} according to the weight in Equation 4.25. A relevant vertex $\mathbf{v}_{\mathrm{p}}$ is selected by choosing randomly from the top n_q elements in the sorted list. This injects randomness in the selection process and promotes desirable exploration.

4.4 Experiments and Discussion

The performance of the proposed sampling method was benchmarked against direct Informed Sampling [12] and the third variant of adaptive rejection sampling (described in [55]) in uniform cost-space environments (length-optimal planning). For all experiments, the exploration strategies were paired with RRT$^{\#}$'s dynamic programming based global rewiring

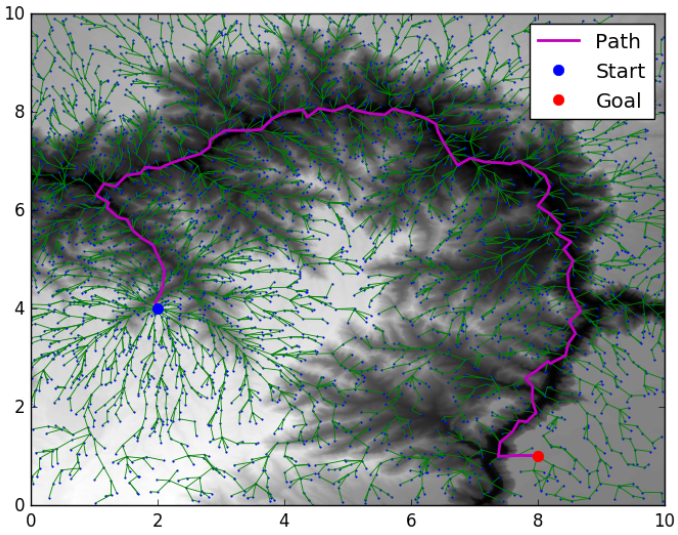


Figure 4.6: Planning on a terrain cost-map with the proposed sampling strategy. Here, white regions represent rough (high cost) areas and the blacks signify smooth sections.

for exploitation. In general cost-map environments, benchmarking was done against Informed Sampling and T-RRT$^{\#}$ (combining conventional RRT$^{\#}$ with the transition-test described in [13]) with different initial temperatures T_{init}. All the algorithms were implemented in C++ using the popular OMPL framework [54], and the tests were run using OMPL's standardized benchmarking tools [62]. A 64-bit desktop PC with 64 GB RAM and an Intel Xeon(R) Processor running Ubuntu 16.04 OS was used. The data was recorded over 100 trials for all the cases. The proposed algorithm used the following parameter values:$\epsilon = 1.5\eta$, $(\lambda_1, \lambda_2, \lambda_3) = (10, 5, 100), n_q = 10$. A goal bias of 5% was used in all sampling methods. A description of the different environments is provided below.

Uniform Cost-Map Cases:

Multiple Obstacle World: This environment is illustrated in Figure 4.1. The 2D environment was extended to $\mathbb{R}^4$ and $\mathbb{R}^6$ by imparting a length of 2 units symmetrically to all of the obstacles. A step-size of $\eta = 0.6$ and $\eta = 1.2$ was used in $\mathbb{R}^4$ and $\mathbb{R}^6$ respectively.

Panda Arm: A planning problem for Panda Arm (by Franka Enmika) is illustrated in Figure 4.3. The objective was to find a minimum length path in a 7-dimensional configuration (joint) space with joint limits ($\mathbb{R}^7$). These limits and collision checking module were

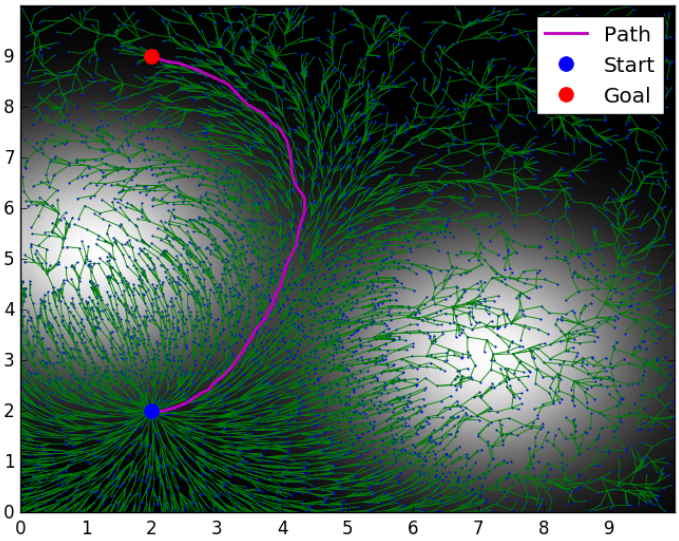


Figure 4.7: Planning on a "potential-field" like cost-map. The objective is to reach the goal state while avoiding the two danger (white) regions.

implemented using MoveIt! [63]. The step-size was set to $\eta = 0.7$ for this example.

General Cost-Map Cases:

Terrain Map: A 2D terrain map shown in Figure 4.6 consists of rough, high-cost white areas and the easily navigable black regions. The step-size was set to $\eta = 0.3$ for this example.

Potential Cost-Map: The environment in Figure 4.7 emulates the problem of finding the shortest path while staying away from danger areas (white regions). The cost function is defined as

$$C(\mathbf{x}) = 1 + 9\big(e^{-\frac{\|\mathbf{x}_1^d - \mathbf{x}\|_2^2}{5}} + e^{-\frac{\|\mathbf{x}_2^d - \mathbf{x}\|_2^2}{5}}\big). \tag{4.26}$$

Here, $\mathbf{x}_1^d$, $\mathbf{x}_2^d$ are the center points of the danger regions. A step-size of $\eta = 0.6$ and $\eta = 1.5$ was used in $\mathbb{R}^4$ and $\mathbb{R}^6$ version of the environment respectively.

Numerical experiments validate the utility of Relevant Region sampling in conjunction with Informed/Uniform Sampling. The proposed method leads to faster convergence in all cases (see Figure 4.4). This is observed especially in higher dimensional problem instances. Transition-test based exploration is more effective than purely Uniform/Informed Sampling for planning on general cost-maps. However, the tendency to (probabilistically)

reject samples may hinder exploration in some cases. This can be seen in the terrain cost-map (Figure 4.6) which is similar to the cost-space chasms scenario described in [59]. As conveyed in Figure 4.5, the transition-test based exploration fails to find a feasible solution in roughly 40% of total trials, whereas the proposed method finds a solution in all trials and also accelerates the convergence.

This chapter rigorously defines, analyzes properties of, and presents a generative technique to sample the Relevant Region set. Note that, while sampling the Informed Set is a necessary condition to improve the current solution, it is not sufficient. Concretely, an "Informed Sample" is not guaranteed to improve the current solution or the cost-to-come value of the vertices In the next chapter, the Relevant Region framework is extended to address this limitation by incorporating gradient information from the planner's tree structure.

CHAPTER 5
LOCALLY EXPLOITATIVE RELEVANT REGION SAMPLING

5.1 Motivation

As discussed in the introductory chapters, uniform random sampling biases the graph growth towards vertices with larger Voronoi regions in RRT-style methods [15]. This results in a rapid exploration of the search-space and is effective for finding an initial solution in single-query scenarios. However, this strategy, like many others in the literature (e.g., [64], [39], [40]), prioritizes acquisition of new information over the improvement of current paths in the planner's graph. This bias towards exploration can have a detrimental effect on convergence, especially in higher dimensions [12]. In contrast to such exploration-biased techniques, the algorithm proposed in this chapter aims to generate new samples that can improve the cost-to-come value of vertices and initiate rewirings. The proposed algorithm first selects a vertex and then generates a new sample in its vicinity. This sample is generated by considering an optimization problem, wherein the objective is to minimize the sum of cost-to-come value of a chosen vertex and its randomly selected descendants. The proposed sampling algorithm thus leverages local information to provide an *exploitative* bias. This combination of global exploratory and locally exploitative sampling results in faster convergence for SBMP algorithms, as demonstrated by several benchmarking experiments in this chapter.

As seen in the previous chapter, the combination of Relevant Region and Informed Sampling results in accelerated convergence in uniform and general cost-space environments. However, these techniques do not generate samples to directly improve the cost-to-come value of vertices. Hence, some of the samples may fail to trigger any improvement in the planner's graph. The sampling algorithm proposed in this chapter also generates

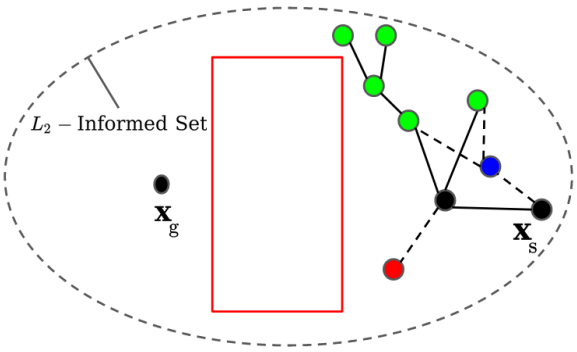


Figure 5.1: Schematic motivating the proposed LES algorithm, which leverages local information and considers an optimization problem to generate the blue sample. In contrast to the red sample, the blue sample can initiate rewirings and improve cost-to-come value of (green) vertices in the graph.

new samples in the Relevant Region to avoid redundant exploration. However, it does so by considering an optimization problem aimed towards improving the cost-to-come value of vertices in the graph. Application of the proposed sampling algorithm thus initiates a higher number of rewirings and results in a faster convergence. Please see Figure 5.1 for an illustration of this.

Approaches combining sampling-based planning and local optimizers have also been explored. RABIT* [65] uses CHOMP [66] to get feasible, high quality edges connecting any two vertices during a global search performed by BIT*. However, RABIT* requires pre-computed domain information, such as an obstacle potential function, which may not be available in many practical problems. Volumetric Tree* [67] addresses this limitation by constructing an approximation of the obstacle-free configuration space on-the-fly. However, it relies on uniform random sampling for graph construction, which may lead to redundant exploration. DRRT [24] employs a gradient-descent based procedure in the graph-processing module. However, this "vertex movement" procedure incurs a very high computational cost due to the extra calls to the nearest-neighbor and collision-checking function to ensure feasibility. The sampler proposed in this chapter does not require extra collision-checking/nearest-neighbor calculations and can be used in conjunction with any graph-processing module.

5.2 Sampling As Optimization

Given $\mathcal{G}$, the objective of the graph-processing module is to minimize the cost-to-come value of all vertices. This objective can be written as

$$J_{\mathcal{T}} = \sum_{\mathbf{u} \in V} \mathrm{g}_{\mathcal{T}}(\mathbf{u}). \tag{5.1}$$

Let $J_{\mathcal{T}}(\mathbf{v})$ denote the terms of $J_{\mathcal{T}}$ that are dependent only on a particular vertex $\mathbf{v} \in V$. The position of vertex $\mathbf{v}$ impacts the cost-to-come value of itself and its descendants. Then,

$$J_{\mathcal{T}}(\mathbf{v}) = \mathrm{g}_{\mathcal{T}}(\mathbf{v}) + \sum_{\mathbf{w} \in D_{\mathbf{v}}} \mathrm{g}_{\mathcal{T}}(\mathbf{w}). \tag{5.2}$$

Using Equation 2.5, the above equation for $J_{\mathcal{T}}(\mathbf{v})$ can be written in terms of the edge-costs $\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v})$ and $\mathrm{c}_\ell(\mathbf{v}, \mathbf{u})$. Here, $\mathbf{v}_\mathrm{p} = \mathsf{parent}(\mathbf{v})$ and $\mathbf{u}$ is any child of $\mathbf{v}$. The edge-cost $\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v})$ will appear $1 + d_{\mathbf{v}}$ times in total, to calculate the cost-to-come value of $\mathbf{v}$ and its descendants. Similarly, the edge-cost $\mathrm{c}_\ell(\mathbf{v}, \mathbf{u})$ will appear $1 + d_{\mathbf{u}}$ times in total, to calculate the cost-to-come value of $\mathbf{u}$ and its descendants. Then,

$$J_{\mathcal{T}}(\mathbf{v}) = k_1 + (1 + d_{\mathbf{v}})\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v}) + \sum_{\mathbf{u} \in V_{\mathbf{v}}} (1 + d_{\mathbf{u}})\mathrm{c}_\ell(\mathbf{v}, \mathbf{u}). \tag{5.3}$$

Note that Equation 5.3 for $J_{\mathcal{T}}(\mathbf{v})$ only contains terms dependent on $\mathbf{v}$. Other terms are incorporated in the constant k_1. Also, $d_{\mathbf{v}}$ and $d_{\mathbf{u}}$ in Equation 5.3 are linked by Equation 2.6. A new sample can be generated by first selecting a vertex $\mathbf{v}$ and then finding a "better" position for it by optimizing $J_{\mathcal{T}}$ with respect to $\mathbf{v}$. Note that $\arg\min_{\mathbf{v}} J_{\mathcal{T}} = \arg\min_{\mathbf{v}} J_{\mathcal{T}}(\mathbf{v})$.

However, calculating the values of the coefficients $d_{\mathbf{v}}, d_{\mathbf{u}}$ in Equation 5.3 requires a depth-first search with time complexity of $O(|V_t|)$. This may get computationally cumbersome, especially as the planner tree grows larger with the number of iterations. Hence, the

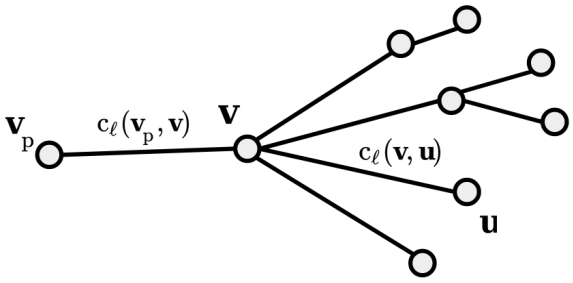


Figure 5.2: Neighborhood around a vertex $\mathbf{v}$. Here, $n_{\mathbf{v}} = 4$ and $\widehat{d}_{\mathbf{v},V_{\mathbf{v}}} = 4 + (1 + 2) = 7$.

following objective function is considered instead.

$$\begin{aligned} \widehat{J}_{\mathcal{T},V_{\mathbf{v}}}(\mathbf{v}) &= k_2 + (1 + \widehat{d}_{\mathbf{v},V_{\mathbf{v}}})\mathrm{c}_\ell(\mathbf{v}_{\mathrm{p}}, \mathbf{v}) + \sum_{\mathbf{u} \in V_{\mathbf{v}}} (1 + n_{\mathbf{u}})\mathrm{c}_\ell(\mathbf{v}, \mathbf{u}), \\ \widehat{d}_{\mathbf{v},V_{\mathbf{v}}} &= n_{\mathbf{v}} + \sum_{\mathbf{u} \in V_{\mathbf{v}}} n_{\mathbf{u}}. \end{aligned} \tag{5.4}$$

Please see Figure 5.2.

Note that minimizing $\widehat{J}_{\mathcal{T},V_{\mathbf{v}}}(\mathbf{v})$ in Equation 5.4 with respect to $\mathbf{v}$ is equivalent to minimizing the cost-to-come values of $\mathbf{v}$, the set of children $V_{\mathbf{v}}$ and their children. The objective $\widehat{J}_{\mathcal{T},V_{\mathbf{v}}}(\mathbf{v})$ can be calculated efficiently with the information contained in the data structure of vertex $\mathbf{v}$, without recursing deeper down the tree. Effectively, $\widehat{J}_{\mathcal{T},V_{\mathbf{v}}}(\mathbf{v})$ considers descendants of $\mathbf{v}$ up to a depth of 2, whereas $J_{\mathcal{T}}(\mathbf{v})$ considers full depth. This can be generalized to depth-k descendants. Finally, a random subset of the children, denoted by $\widehat{V}_{\mathbf{v}} \subseteq V_{\mathbf{v}}$, can be selected and a new sample generated by minimizing $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v})$. This serves two purposes. First, it promotes a desirable randomness in the sampling process. Second, focusing on the subset $\widehat{V}_{\mathbf{v}}$ effectively assigns a weight of zero for the terms corresponding to the vertices $V_{\mathbf{v}} \setminus \widehat{V}_{\mathbf{v}}$ in the objective Equation 5.3, Equation 5.4. This can lead to a better improvement in the cost-to-come value of vertices corresponding to $\widehat{V}_{\mathbf{v}}$.

5.3 Curse Of Dimensionality For Sampling

The proposed "Locally Exploitative Sampling (LES)" procedure, described in the next section, first selects a vertex $\mathbf{v}$ and then generates a new sample considering $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v})$. Expan-

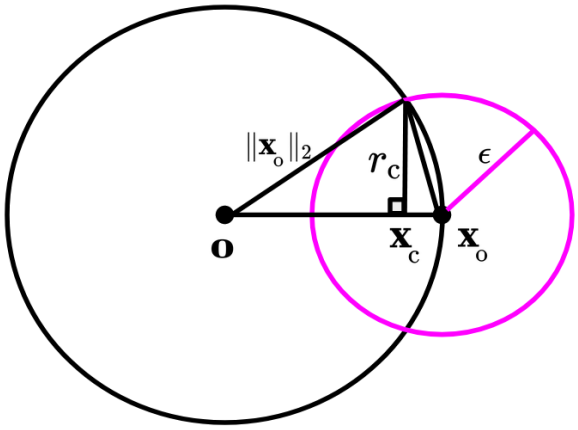


Figure 5.3: Schematic for the analysis in Appendix. Black and magenta circles illustrate the set $\mathcal{B}^{\|\mathbf{x}_o\|_2}(\mathbf{0})$ and $\mathcal{B}^{\epsilon}(\mathbf{x}_o)$ respectively. The intersection $\mathcal{B}^{\epsilon}(\mathbf{x}_o) \cap \mathcal{B}^{\|\mathbf{x}_o\|_2}(\mathbf{0})$ can be over-approximated by hyper-sphere centered at $\mathbf{x}_c$ with radius r_c.

sive Space Trees (EST) [42] and its variants, such as [43], [44], also proceed by selecting a vertex and generating a random sample in its vicinity. However, the probability of generating a "good" sample (that can improve $\widehat{J}_{\mathcal{T},\widehat{V}_v}(\mathbf{v})$) with such random search decreases rapidly in higher dimensions. This is illustrated in the analysis below by considering the problem of minimizing a quadratic function $J_q(\mathbf{x}) = \mathbf{x}^\mathsf{T}\mathbf{x}$ with random local search. The probability of generating a sample that can improve J_q diminishes exponentially with the dimension d.

Consider the problem of minimizing a quadratic objective function $J_q(\mathbf{x}) = \mathbf{x}^\mathsf{T}\mathbf{x}$ with random local search. This analysis is similar to the one provided in [68]. Let the starting state be $\mathbf{x}_o \in \mathbb{R}^d$ with the corresponding objective cost $J_q(\mathbf{x}_o)$. Random search generates samples in the set $\mathcal{B}^{\epsilon}(\mathbf{x}_o)$ to find a new state with cost less than $J_q(\mathbf{x}_o)$. Assume $\epsilon < \|\mathbf{x}_o\|_2$. The set of states that provide an improvement over $J_q(\mathbf{x}_o)$ satisfy $\mathbf{x}^\mathsf{T}\mathbf{x} < \mathbf{x}_o^\mathsf{T}\mathbf{x}_o$. This set can be denoted as $\mathcal{B}^{\|\mathbf{x}_o\|_2}(\mathbf{0})$, where $\mathbf{0}$ is the origin. The set of good samples thus lie in the set $\mathcal{B}^{\epsilon}(\mathbf{x}_o) \cap \mathcal{B}^{\|\mathbf{x}_o\|_2}(\mathbf{0})$. Please see Figure 5.3. This intersection between two hyper-spheres can be over-approximated by $\mathcal{B}^{r_c}(\mathbf{x}_c)$, where

$$r_c = \epsilon\sqrt{1 - \frac{\epsilon^2}{4\|\mathbf{x}_o\|_2^2}}. \tag{5.5}$$

Algorithm 7: LES Algorithm Flow

$V \leftarrow \{\mathbf{x}_\mathrm{s}\}$; $E \leftarrow \phi$; $\mathcal{G} \leftarrow (V, E)$;
for $i = 1 : N$ **do**
  $c_i \leftarrow \mathsf{getBestSolutionCost}()$;
  $u_\mathrm{rand} \sim \mathcal{U}(0, 1)$;
  if $u_\mathrm{rand} < p_\mathrm{LES}$ and $c_i < \infty$ **then**
    $\mathbf{v} \leftarrow \mathsf{chooseVertex}(V_\mathrm{rel})$;
    $\hat{\mathbf{e}} \leftarrow \mathsf{getGradientDirection}(\mathbf{v})$;
    $\gamma \leftarrow \mathsf{getStepSize}(\mathbf{v}, \hat{\mathbf{e}})$;
    $\mathbf{x}_\mathrm{rand} \leftarrow \mathbf{v} - \gamma\hat{\mathbf{e}}$;
  else
    $\mathbf{x}_\mathrm{rand} \leftarrow \mathsf{InformedSampling}(\mathsf{c_i})$
  $\mathsf{Extend}(\mathbf{x}_\mathrm{rand})$;
  $\mathsf{GraphProcessing}(\mathcal{G})$;
return $\mathcal{G}$

Algorithm 8: Calculate Gradient Direction

getGradientDirection ($\mathbf{v}$) **:**
  $\widehat{V}_\mathbf{v} \leftarrow \mathsf{getRandomSubset}(V_\mathbf{v})$;
  $\mathbf{e} \leftarrow (1 + \widehat{d}_{\mathbf{v},\widehat{V}_\mathbf{v}})\frac{\partial}{\partial \mathbf{v}}\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v}) + \sum_{\mathbf{u}\in\widehat{V}_\mathbf{v}}(1 + n_\mathbf{u})\frac{\partial}{\partial \mathbf{v}}\mathrm{c}_\ell(\mathbf{v}, \mathbf{u})$;
  $\hat{\mathbf{e}} \leftarrow \mathbf{e}/\|\mathbf{e}\|_2$
return $\hat{\mathbf{e}}$

The probability of generating a good sample using random search is given by

$$\begin{aligned}\mathbb{P}(\mathbf{x} \in \mathcal{B}^\epsilon(\mathbf{x}_\mathrm{o}) \cap \mathcal{B}^{\|\mathbf{x}_\mathrm{o}\|_2}(\mathbf{0})) &= \frac{\lambda\big(\mathcal{B}^\epsilon(\mathbf{x}_\mathrm{o}) \cap \mathcal{B}^{\|\mathbf{x}_\mathrm{o}\|_2}(\mathbf{0})\big)}{\lambda(\mathcal{B}^\epsilon(\mathbf{x}_\mathrm{o}))} \\ &< \frac{\lambda(\mathcal{B}^{r_c}(\mathbf{x}_\mathrm{c}))}{\lambda(\mathcal{B}^\epsilon(\mathbf{x}_\mathrm{o}))} = \big(1 - \frac{\epsilon^2}{4\|\mathbf{x}_\mathrm{o}\|_2^2}\big)^{\frac{d}{2}}.\end{aligned} \tag{5.6}$$

Note that, if $\psi = 1 - \frac{\epsilon^2}{4\|\mathbf{x}_\mathrm{o}\|_2^2}$, then $0 < \psi < 1$, as $\epsilon < \|\mathbf{x}_\mathrm{o}\|_2$. Hence, ψ^d goes to 0 exponentially as d increases. Thus, the probability of generating a good sample decreases exponentially with the dimension d.

Algorithm 9: Calculate Step-size

```
1  getStepSize ( v, ê ) :
2      γ_rel ← getMaxStepSize(v, ê);
3      γ_max ← γ_rel;
4      while γ_max > δ do
5          u_rand ~ U(0,1); γ ← (u_rand)^{1/d} γ_max;
6          x_rand ← v − γê;
7          if Ĵ_{T,V̂_v}(x_rand) < Ĵ_{T,V̂_v}(v) then
8              break;
9          else
10             γ_max ← γ;
11     if γ_max < δ then
12         u_rand ~ U(0,1); γ ← (u_rand)^{1/d} γ_rel;
13 return γ
```

5.4 Locally Exploitative Sampling Algorithm

This motivates the LES procedure, given in algorithm 7. With probability p_{LES}, LES is used to generate a new sample $\mathbf{x}_{\mathrm{rand}}$ (algorithm 7, line 6-8). Otherwise, a new sample is generated using the conventional Informed Sampling technique given in [12] (algorithm 7, line 11). This ensures a balance between exploration-exploitation (controlled by the parameter p_{LES}) and graph growth in all the relevant homotopy classes. The Extend function takes this random sample and performs relevant procedures (nearest-neighbor, local steering and collision checking) to incorporate a new vertex in the graph (algorithm 7, line 12). Finally, the graph-processing module operates on $\mathcal{G}$ considering the addition of a new vertex (algorithm 7, line 13).

If the best available solution cost c_i after i iterations is finite (indicating that a sub-optimal solution has been discovered), redundant exploration can be avoided by focusing the search on the Informed or Relevant Region set. As the Informed Set may be ineffective in focusing the search for general cost-space problems, LES generates new samples in the Relevant Region $\mathcal{X}_{\mathrm{rel}}^{\epsilon}$ [69], defined in Equation 4.5. The value of ϵ in Equation 4.5 is set

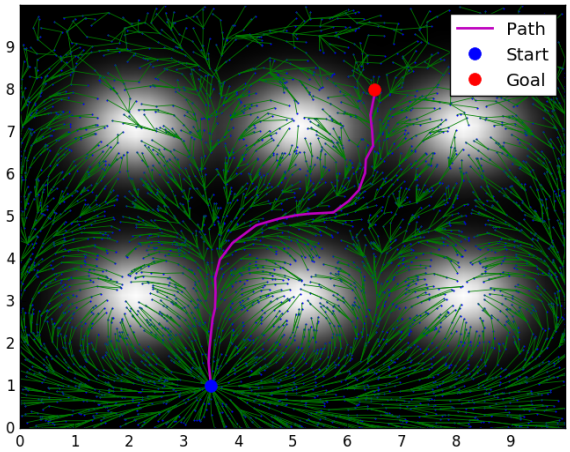


Figure 5.4: Planning with the proposed LES algorithm on a potential cost-map. The robot incurs a higher cost if it travels in the white regions.

to $\epsilon = 1.5\eta$, where η is the range parameter in SBMP algorithms [54], which controls the maximum edge-length in $\mathcal{G}$. The procedure for selecting a vertex (chooseVertex), is same as the implementation in chapter 2 [69]. It assigns a weight $q_{\mathbf{v}}$ for each $\mathbf{v} \in V_{\text{rel}}$ and uses a binary heap data-structure for sorting. Start, goal and leaf vertices (vertices with no children) are ignored by the chooseVertex function.

Note that, a closed-form solution to the optimization objective $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v})$ cannot be obtained in general. Hence, LES proceeds by numerically calculating the gradient of $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v})$ and moving an appropriate step-size in the direction of the gradient. The procedure to calculate the gradient direction $\hat{\mathbf{e}}$ is given in algorithm 8. First, a random subset of children $\widehat{V}_{\mathbf{v}}$ is obtained. This is done by randomly sampling a start integer $s_{\text{ind}} \sim U\big[0, |V_{\mathbf{v}}| - 1\big]$ and an end integer $e_{\text{ind}} \sim U\big[s_{\text{ind}}, |V_{\mathbf{v}}|\big]$ Then, all children vertices with indices s_{ind} to e_{ind} and their descendants upto a certain depth are considered for constructing $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v})$. The gradient of $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v})$ with respect to $\mathbf{v}$ is calculated numerically using the symmetric difference formula (algorithm 8, line 3). Having obtained the gradient direction $\hat{\mathbf{e}}$, the algorithm to calculate the step-size is given in algorithm 9. As finding the optimal step-size γ^* by solving $\arg\min_\gamma \widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v} - \gamma\hat{\mathbf{e}})$ is intractable, approaches such as backtracking line search [70] have been suggested. However, executing backtracking line search is computationally not viable for the current application, as it requires a higher number of expensive calls to

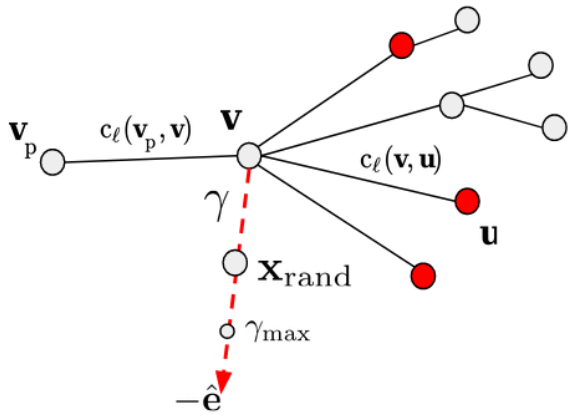


Figure 5.5: A schematic for the LES algorithm. The gradient direction (red) is calculated by considering the cost-to-come value of randomly selected children vertices (red) and their descendants.

calculate $\widehat{J}_{\mathcal{T},\widehat{V}_\mathbf{v}}$. Instead, LES uses a procedure given in algorithm 9, which is similar to the Hit-and-Run Sampler implemented in [51]. First, given a vertex $\mathbf{v}$ and the travel direction $-\hat{\mathbf{e}}$, the procedure in [69] is used to calculate the maximum step-size γ_rel. This ensures that a candidate $\mathbf{v} - \gamma\hat{\mathbf{e}} \in \mathcal{X}^\epsilon_\mathrm{rel}$ for any $\gamma \in (0, \gamma_\mathrm{rel})$. Variable γ_max is set to γ_rel. Next, a random step-size γ is sampled from the interval $(0, \gamma_\mathrm{max})$. Please see Figure 5.5. The exponent of $1/d$ in algorithm 9, line 5 biases γ towards γ_max. If the candidate $\mathbf{v} - \gamma\hat{\mathbf{e}}$ results in an improvement for $\widehat{J}_{\mathcal{T},\widehat{V}_\mathbf{v}}$, step-size γ is returned. Else, γ_max is updated to γ. Thus, the search interval is sequentially reduced until a suitable step-size is discovered. Theoretically, a travel of infinitesimal magnitude in the direction of the gradient always results in an improvement. However, if γ_max is less than a small quantity $\delta \ll \eta$, then a random γ in the interval $(0, \gamma_\mathrm{rel})$ is returned (Algorithm. algorithm 9, line 11-12) to avoid clumping of new vertices around $\mathbf{v}$.

5.5 Experiments and Discussion

Numerical experiments consider three variations of the proposed LES algorithm, namely LES-2, LES-8 and LES-∞. LES-2 and LES-8 consider descendants up to a maximum depth of 2 and 8 respectively, whereas LES-∞ considers full depth up to the leaf nodes.

Benchmarking was performed against Informed sampler and Relevant Region sampler described in [12] and [69] respectively. Note that LES and Relevant Region sampler share a similar chooseVertex procedure. However, the Relevant Region sampler only generates random samples in $\mathcal{X}_{\text{rel}}^{\epsilon}$ and does not consider the optimization problem corresponding to Equation 5.3 or Equation 5.4. All the algorithms were implemented using C++/OMPL [54]. Data was gathered over 100 trials for each experiment using the standardized OMPL benchmarking tools [62]. All experiments were performed on a 64 bit laptop running Ubuntu 16.04 OS, with 16 GB RAM and an Intel i7 processor. The parameter p_{LES} and an analogous parameter p_{rel} for Relevant Region sampler were both set to 0.5. The parameter δ was set to 10^{-4}. All sampling strategies used a goal bias of 5% and were paired with RRT$^{\#}$'s global rewiring for graph-processing. A description of the different benchmarking environments is given below.

Potential Cost-map: This environment, illustrated in Figure 5.4, has the state-cost function

$$C(\mathbf{x}) = 1 + 9 \sum_{i} \exp\big(- \|\mathbf{x}_i^{\text{c}} - \mathbf{x}\|_2^2 \big). \tag{5.7}$$

Here, $\mathbf{x}_i^{\text{c}}$ represent the center points of the high cost white regions. The objective for the robot is to plan a path to the goal while avoiding these soft obstacles. The range parameter η was set to $0.4, 0.6$ and 1.5 for the 2D, 4D and 6D versions of environment respectively.

Robot Manipulator: A planning problem for a 7 DOF Panda and a 14 DOF Baxter arm is illustrated in Figure 5.6. The objective was to find the minimum length path ($C(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathcal{X}$) in the configuration-space with strict joint limits ($\mathcal{X} \subset \mathbb{R}^7$ for Panda, $\mathcal{X} \subset \mathbb{R}^{14}$ for Baxter). These joint limits and collision checking calculations were implemented with the help of MoveIt! [63]. The range parameter η was set to 1.2 and 2 for the Panda and Baxter experiments respectively.

Many Homotopy Classes World: This case, similar to one studied in [12], is illustrated in Figure 5.7. The objective is to find a length-optimal path in this cluttered environment. This world has multiple non-optimal homotopy classes, but only two optimal ones. Experiments

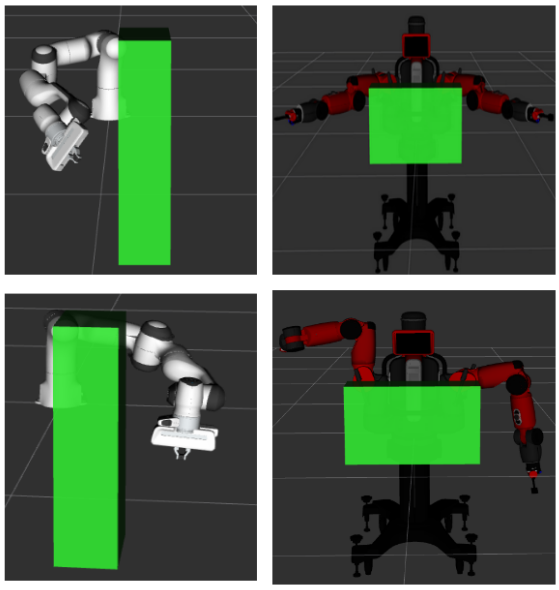


Figure 5.6: Planning in the joint space of Panda and Baxter manipulator arms. The start and goal positions for both robots are indicated in the top and bottom figures respectively. A video of full robot plan can be found here: https://youtu.be/J4B5_L2Ghrs

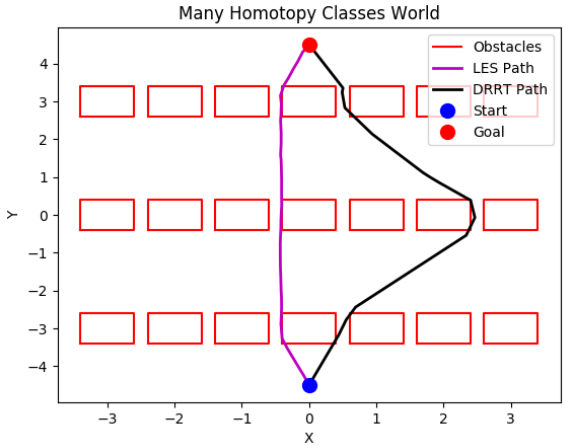
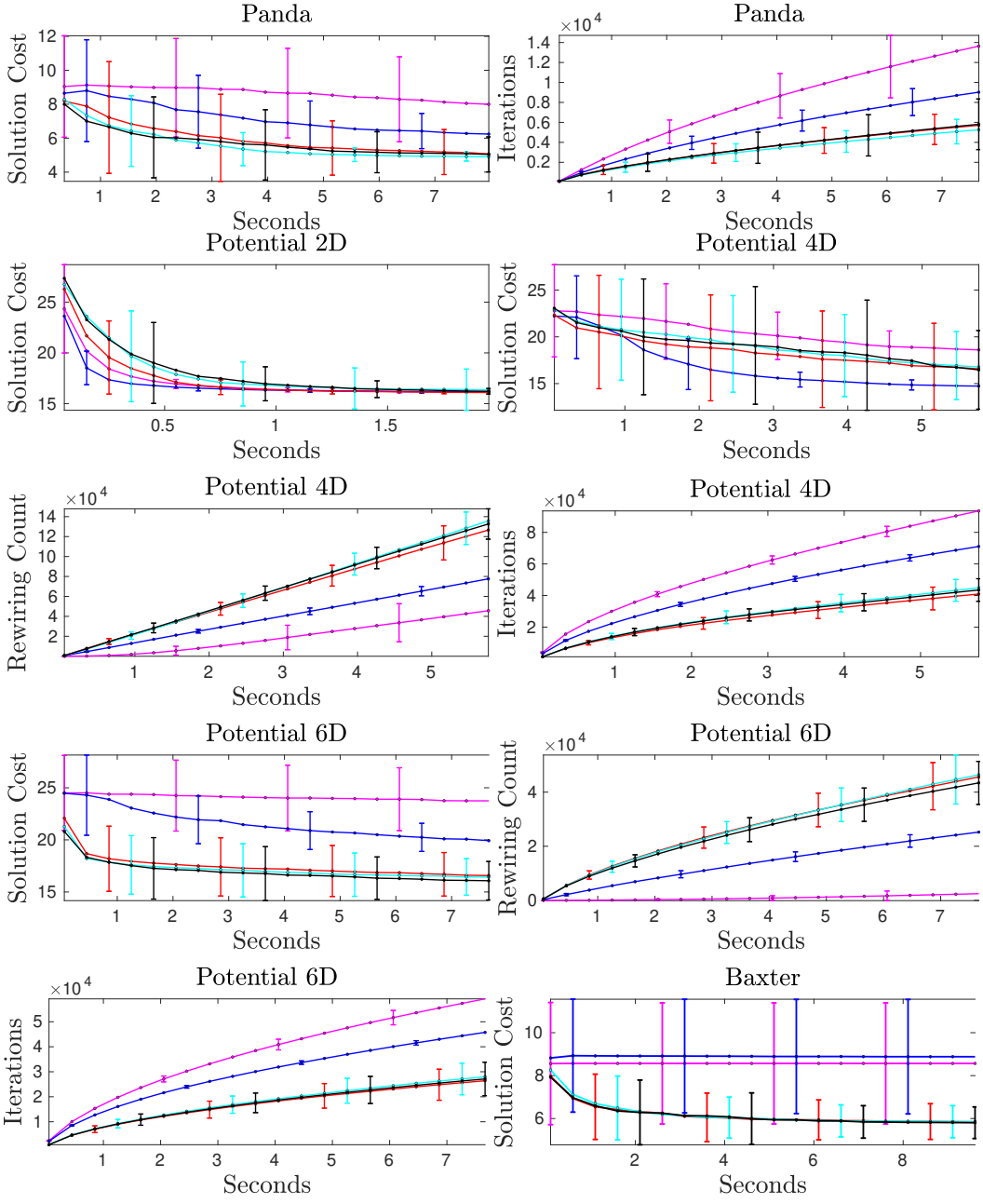


Figure 5.7: Planning with the proposed LES-RRT$^{\#}$ and DRRT [24] in a world with multiple homotopy classes. The computationally costlier DRRT (black path) can get "stuck" in a wrong homotopy class. For the above trial run, LES-RRT$^{\#}$ and DRRT sampled 9414 and 573 graph vertices respectively in 1 second of planning time.

were performed for a 4D and 6D version of this environment with the range parameter η set to 1.5 and 2.5 respectively. Benchmarking for this case also considered two variants of the DRRT planner, namely DRRTd and DRRT0.3 [24]. The DRRTd algorithm delays the computationally expensive, vertex optimization procedure until an initial solution is discovered, whereas DRRT0.3 calls that procedure only with a probability of 0.3.

Results from the numerical experiments are illustrated in Figure 6.5. The proposed LES variations outperform Informed and Relevant Region samplers in higher dimensional settings (Potential 6D, Panda, Baxter) in terms of cost convergence. While the performance of all three LES variants is similar, LES-∞ and LES-8 perform slightly better than LES-2 in case of Potential 6D and Panda. LES also initiates a larger number of rewirings in $\mathcal{T}$. However, similar performance gains are not seen in the lower dimensional environments (Potential 2D, 4D). The Relevant Region sampler, with its focusing properties, performs better than Informed sampling. LES incurs a higher computational cost due to the numerical gradient calculations in algorithm 8 and expensive function evaluations of $\widehat{J}_{\mathcal{T},\widehat{V}_\text{v}}$ in algorithm 9. Thus, the application of LES leads to a lesser number of iterations executed in a given time period compared to the other two methods. This might slow down convergence in lower dimensions. However, random search techniques are affected by the "curse of dimensionality" as illustrated in the analysis above. This justifies the computationally more costly procedures of LES which lead to an accelerated convergence in higher dimensions.

The DRRT planner can show tremendous performance gains, in terms of cost convergence, for higher dimensional, relatively less cluttered environments. However, for cases such as Figure 5.7, the DRRT planner can find an initial solution in a non-optimal homotopy class. Its computationally expensive, vertex movement procedure requires additional collision-checking and nearest-neighbor calculations to maintain feasibility and graph neighborhoods. As a consequence, it generates significantly lesser number of vertices compared to the other methods (see Figure 6.5). Hence, optimizing the graph comes at a steep cost of reduced exploration and finding solution paths in other homotopy classes


Panda
Solution Cost
Seconds
Panda
Iterations
Seconds
Potential 2D
Solution Cost
Seconds
Potential 4D
Solution Cost
Seconds
Potential 4D
Rewiring Count
Seconds
Potential 4D
Iterations
Seconds
Potential 6D
Solution Cost
Seconds
Potential 6D
Rewiring Count
Seconds
Potential 6D
Iterations
Seconds
Baxter
Solution Cost
Seconds

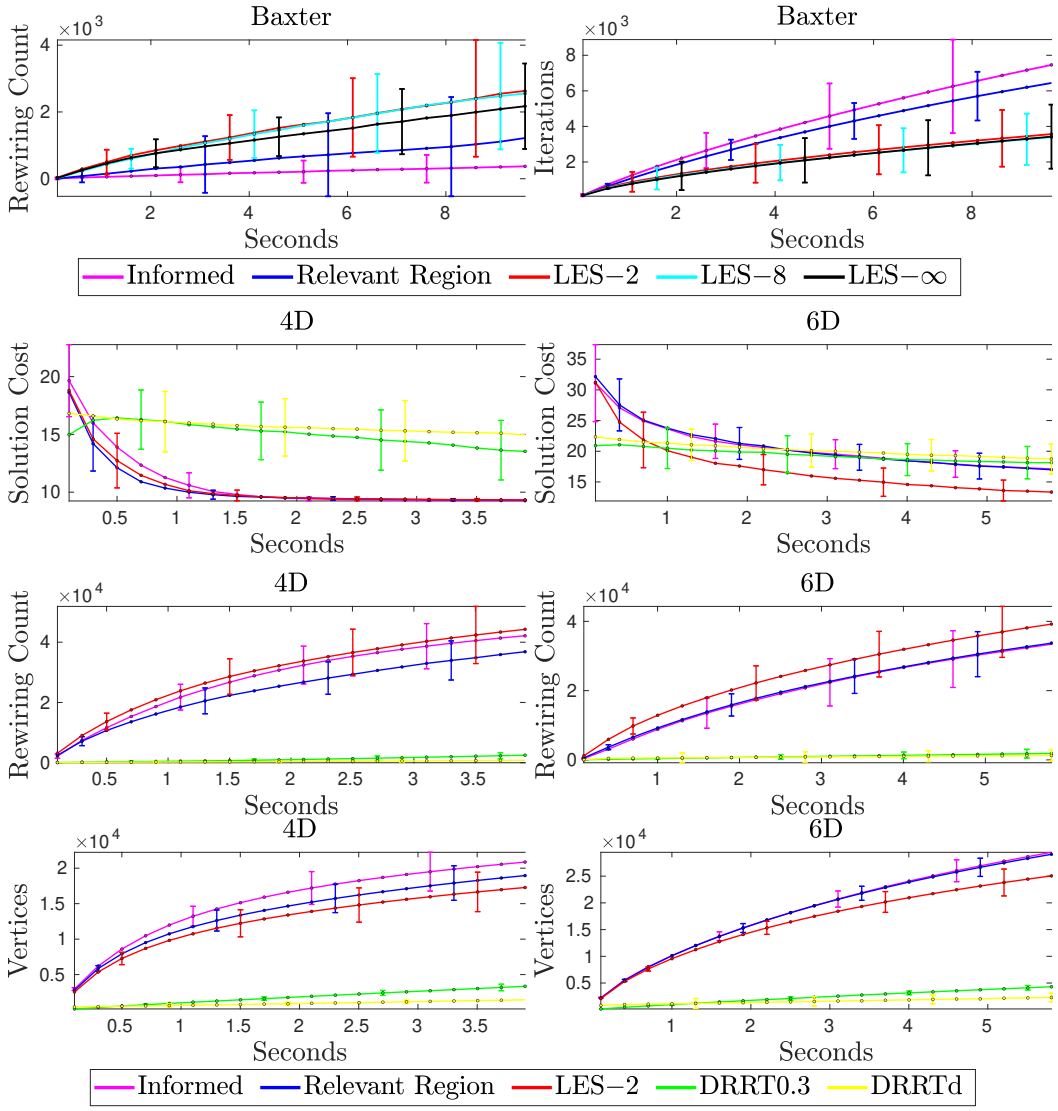


Figure 5.8: Benchmarking plots for the numerical experiments. Solid lines indicate the value averaged over 100 trials and the error bars represent standard deviation. Application of the proposed LES (red, cyan, black) leads to a faster convergence and a larger number of tree rewirings in higher dimensions. However, it incurs a higher computational cost and hence executes a lesser number of iterations compared to Informed (magenta) and Relevant Region (blue) sampling.

for DRRT. It can thus get "stuck" in the wrong homotopy class. While LES also leverages the gradient information, it is much more computationally tractable than DRRT. It generates enough vertices to explore all relevant homotopy classes while also initiating higher number of rewirings to improve the cost-to-come values. Thus, LES strikes the right balance between exploration and exploitation and outperforms all other methods in a higher dimensional case such as 6D in Figure 6.5.

The previous three chapters explore a family of intelligent exploration algorithms that focus search for geometric planners. These planners ignore the kino-dynamic constraints of the robot and connect any two points in the search-space using a straight line. The next chapter tackles a challenging problem of informed exploration for kino-dynamic planning.

CHAPTER 6
TIME-INFORMED EXPLORATION

6.1 Motivation

While the previous chapters discussed the problem of intelligent exploration for geometric sampling-based planners, this chapter looks into the kino-dynamic case. As delineated in the Chapter 2, a significant progress has been made in the area of sampling-based kino-dynamic planners, with efficient methods such as SST [31]. However, developing intelligent exploration strategies to complement these kino-dynamic planners still remains a challenging problem. Uniform random sampling results in a rapid exploration of the search-space and is effective for finding a first solution. However, after an initial solution is found, exploration can be focused on a subset of the search-space that can potentially further improve the current solution. For the case of geometric, length-optimal planning, Gammell et al [12] introduced the "L_2-Informed Set" that contains all the points that can *potentially* improve the current solution. This set is a prolate hyper-spheroid with focii at the start and the goal states and its transverse diameter is equal to the current best solution cost. The direct Informed Sampling (IS) technique proposed in [12] provides a scalable approach to focus search, and shows dramatic convergence improvements in higher dimensions compared to the other state-of-the-art heuristic methods.

However, as discussed in [50], [51] deriving a parameterized representation or direct sampling of such Informed Sets for systems with differential constraints is a challenging problem. In this work, we propose an analogue to the Informed Set for the case of *time-optimal* kino-dynamic planning using ideas from reachability analysis [71, 72]. Given a feasible (but perhaps sub-optimal) solution trajectory with time cost $T > 0$, we define a Time-Informed Set (TIS) as the set that contains all the trajectories with time cost less

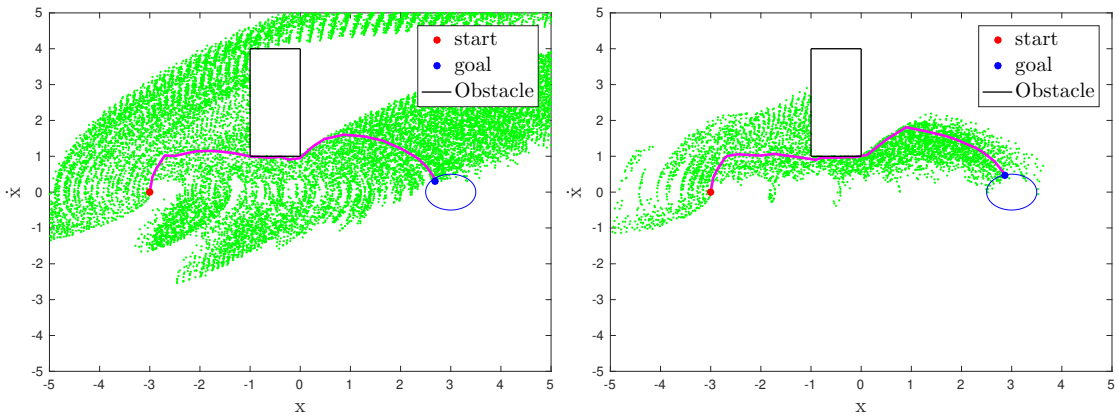


Figure 6.1: Time-optimal planning for a 2D system using the SST algorithm with uniform exploration (left) and the proposed strategy (right). The tree vertices generated are represented in green. Using the proposed strategy leads to a focused search.

than or equal to T. The planner can thus avoid redundant exploration outside the TIS. The proposed exploration algorithm can be applied to a variety of systems, even if a tractable TPBVP solver may not be available.

6.2 Time-Informed Set

This section defines a "Time-Informed Set" to focus search in the case of time-optimal kino-dynamic planning. This problem is defined below.

$$T^* = \min_{\mathbf{u}} \; T \tag{6.1a}$$

$$\text{subject to: } \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \tag{6.1b}$$

$$\mathbf{x}(0) = \mathbf{x}_\text{s}, \; \mathbf{x}(T) \in \mathcal{X}_\text{g}, \tag{6.1c}$$

$$\mathbf{x}(t) \in \mathcal{X}_\text{free}, \; \mathbf{u}(t) \in \mathcal{U} \quad \text{for all } t \in [0, T]. \tag{6.1d}$$

Now, consider the set of points that can be reached at time t, starting from $\mathbf{x}_\text{s}$ at time $t_0 < t$, using admissible controls,

$$\begin{aligned} \mathcal{X}_f[t_0, t] = \{\mathbf{z} \in \mathcal{X} \mid \exists\, \mathbf{u} : [t_0, t] \to \mathcal{U}, \; \mathbf{x} : [t_0, t] \to \mathcal{X}, \\ \text{s.t } \mathbf{x}(t_0) = \mathbf{x}_\text{s}, \mathbf{x}(t) = \mathbf{z}, \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))\}. \end{aligned} \tag{6.2}$$

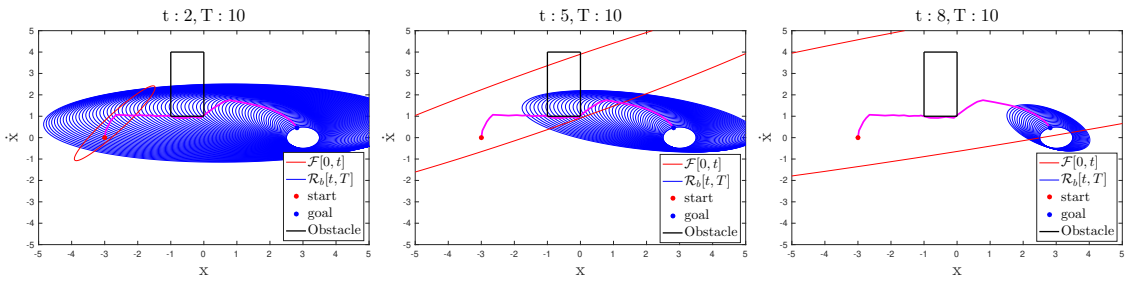


Figure 6.2: Evolution of the forward reachable set $\mathcal{F}[0,t]$ and the backward reachable tube $\mathcal{R}_b[t,T]$ for the 2D Toy system at time $t = 2, 5, 8$. Note that $\Omega(T)$ comprises of the intersections $\mathcal{F}[0,t] \cap \mathcal{R}_b[t,T]$.

Let $\mathcal{F}[t_0,t]$ be an over-approximation of $\mathcal{X}_f[t_0,t]$, i.e., $\mathcal{X}_f[t_0,t] \subseteq \mathcal{F}[t_0,t]$. Similarly, the set of points starting at time t that can reach $\mathcal{X}_\text{g}$ at time $t_f > t$ using admissible controls can be defined as,

$$\begin{aligned}\mathcal{X}_b[t,t_f] = \{\mathbf{z} \in \mathcal{X} \mid \exists\, \mathbf{u} : [t,t_f] \to \mathcal{U},\ \mathbf{x} : [t,t_f] \to \mathcal{X},\\ \text{s.t } \mathbf{x}(t) = \mathbf{z}, \mathbf{x}(t_f) \in \mathcal{X}_\text{g}, \dot{\mathbf{x}}(t) = f(\mathbf{x}(t),\mathbf{u}(t))\}.\end{aligned} \tag{6.3}$$

Let $\mathcal{B}[t,t_f]$ be an over-approximation of $\mathcal{X}_b[t,t_f]$, i.e., $\mathcal{X}_b[t,t_f] \subseteq \mathcal{B}[t,t_f]$. Note that state constraints ensuring collision-free trajectories are not imposed while defining the above sets. The (over-approximated) backward reachability *tube* over the interval $[t,t_f]$ includes the set of all points starting at time t, that can reach $\mathcal{X}_\text{g}$ at any time $\tau \in [t,t_f]$

$$\mathcal{R}_b[t,t_f] = \bigcup_{t \le \tau \le t_f} \mathcal{B}[t,\tau]. \tag{6.4}$$

Assume that a feasible (perhaps sub-optimal) solution to problem Equation 6.1 with time cost $T > 0$ is available. Consider the following definition of the Time-Informed Set (TIS)

$$\Omega(T) = \bigcup_{0 \le t \le T} \mathcal{F}[0,t] \cap \mathcal{R}_b[t,T]. \tag{6.5}$$

Intuitively, $\Omega(T)$ contains all the points $\mathbf{x} \in \mathcal{X}$ that can be reached from $\mathbf{x}_\text{s}$ at a time t, where $0 \le t \le T$, i.e., $\mathbf{x} \in \mathcal{F}[0,t]$ and then can reach the goal at time τ, $t \le \tau \le T$, i.e.,

$\mathbf{x} \in \mathcal{R}_b[t, T]$. Please see Figure 6.2 and the attached video[1] for a visualization of $\Omega(T)$.

The following theoretical arguments formally prove that given a sub-optimal solution with time cost T, the set $\Omega(T)$ contains all the trajectories with time cost T or less.

Lemma 4. *Given a feasible solution with cost $T > 0$, $\mathcal{F}[0, t] \cap \mathcal{B}[t, T] \neq \varnothing$ for all $t \in [0, T]$.*

Proof. Consider the solution trajectory with time cost T, $\zeta : [0, T] \to \mathcal{X}$, where $\zeta(0) = \mathbf{x}_{\mathrm{s}}$ and $\zeta(T) = \mathbf{x}_{\mathrm{g}}$. For any point $\mathbf{x}$ on this trajectory, there exists $t \in [0, T]$ such that $\mathbf{x} = \zeta(t)$. Thus, $\mathbf{x} \in \mathcal{F}[0, t]$ and $\mathbf{x} \in \mathcal{B}[t, T]$. It follows that, $\mathbf{x} \in \mathcal{F}[0, t] \cap \mathcal{B}[t, T]$. Therefore, $\mathcal{F}[0, t] \cap \mathcal{B}[t, T] \neq \varnothing$. □

Lemma 5. *$\mathcal{R}_b[t, T_1] \subset \mathcal{R}_b[t, T_2]$ for any $T_2 > T_1 > t > 0$.*

Proof. Note from the definition Equation 6.4,

$$\mathcal{R}_b[t, T_2] = \bigcup_{t \leq \tau \leq T_2} \mathcal{B}[t, \tau] = \Big(\bigcup_{t \leq \tau \leq T_1} \mathcal{B}[t, \tau] \Big) \bigcup \Big(\bigcup_{T_1 \leq \tau \leq T_2} \mathcal{B}[t, \tau] \Big).$$

Since $\mathcal{R}_b[t, T_1] = \bigcup_{t \leq \tau \leq T_1} \mathcal{B}[t, \tau]$ it follows that $\mathcal{R}_b[t, T_1] \subset \mathcal{R}_b[t, T_2]$. □

Theorem 6. *The set $\Omega(T)$ contains all trajectories with time cost exactly T.*

Proof. Consider any solution trajectory $\zeta : [0, T] \to \mathcal{X}$ with time cost $T > 0$, where $\zeta(0) = \mathbf{x}_{\mathrm{s}}$, $\zeta(T) = \mathbf{x}_{\mathrm{g}}$. For any point $\mathbf{x}$ on this trajectory, there exists $t \in [0, T]$ such that $\mathbf{x} = \zeta(t)$. Then, $\mathbf{x} \in \mathcal{F}[0, t]$ and $\mathbf{x} \in \mathcal{B}[t, T]$. This implies that $\mathbf{x} \in \mathcal{F}[0, t] \cap \mathcal{B}[t, T]$ and hence $\mathbf{x} \in \mathcal{F}[0, t] \cap \mathcal{R}_b[t, T]$. Thus, $\mathbf{x} \in \Omega(T)$. Since t is arbitrary, if follows that $\zeta(t) \in \Omega(T)$ for all $t \in [0, T]$. □

Theorem 7. *$\Omega(T_1) \subset \Omega(T_2)$ for any $T_2 > T_1 > 0$.*

Proof. Recall that the set $\Omega(T_2)$ is defined by

$$\Omega(T_2) = \bigcup_{0 \leq t \leq T_2} \mathcal{F}[0, t] \cap \mathcal{R}_b[t, T_2], \tag{6.6}$$

[1] https://www.youtube.com/watch?v=dnMHb7uFEGw

which can be re-written as

$$\Omega(T_2) = \Big(\bigcup_{0 \leq t \leq T_1} \mathcal{F}[0,t] \cap \mathcal{R}_b[t, T_2] \Big) \bigcup \Big(\bigcup_{T_1 \leq t \leq T_2} \mathcal{F}[0,t] \cap \mathcal{R}_b[t, T_2] \Big).$$

From Lemma Theorem 5, it follows that $\mathcal{R}_b[t, T_1] \subset \mathcal{R}_b[t, T_2]$.

Hence, $\Omega(T_1) = \bigcup_{0 \leq t \leq T_1} \mathcal{F}[0,t] \cap \mathcal{R}_b[t, T_1] \subset \bigcup_{0 \leq t \leq T_1} \mathcal{F}[0,t] \cap \mathcal{R}_b[t, T_2]$. Thus, $\Omega(T_1) \subset \Omega(T_2)$. □

Corollary 8. *Given a solution to Equation 6.1 with time cost T, the set $\Omega(T)$ defined in Equation 6.5 contains all the trajectories with cost less than or equal to T. Conversely, any trajectory that is not contained inside $\Omega(T)$ has time cost $T^{'} > T$*

Proof. From Theorem Theorem 6, it follows that $\Omega(T)$ contains all trajectories with time cost exactly T. Theorem Theorem 7 implies that $\Omega(T)$ is a superset of all the sets containing trajectories with time cost less than T. Thus, $\Omega(T)$ also contains all the trajectories with cost less than or equal to T. □

After a, perhaps sub-optimal, solution with cost T is found, any state that lies on an improved solution path necessarily lies inside the TIS. The search can thus be focused onto the TIS. This can avoid redundant computations and accelerate convergence, especially for higher dimensional problems.

6.3 Time-Informed vs L_2-Informed Set

This section examines the relationship between the TIS defined in Equation 6.5 and the L_2-Informed Set from [12] for a special case of a linear single integrator system. The purpose of this investigation is to show that the TIS is a generalization of the L_2-Informed Set approach in [12]. Consider the case of single-integrator dynamics $\dot{\mathbf{x}}(t) = \mathbf{u}(t)$, for which

$$\begin{aligned} \mathcal{F}[t_0, t] &= \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}_\mathrm{s}\|_2 \leq u_{\max}(t - t_0)\} \\ \mathcal{B}[t, t_f] &= \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}_\mathrm{g}\|_2 \leq u_{\max}(t_f - t)\}. \end{aligned} \tag{6.7}$$

Here, $\|.\|_2$ represents the L_2-norm. As the set $\mathcal{U}$ is compact, there exists a $u_{\max} > 0$, so that $\|\mathbf{u}(t)\|_2 \leq u_{\max}$ for all t. Note that, for this special case, the forward and backward reachable sets defined in Equation 6.7 are concentric circles. Then, for a given $t < t_f$, we have $\mathcal{B}[t, t_f] = \mathcal{R}_b[t, t_f]$ and hence $\Omega(T) = \bigcup_{0 \leq t \leq T} \mathcal{F}[0, t] \cap \mathcal{B}[t, T]$. Thus, for any $\mathbf{x} \in \Omega(T)$, we have $\|\mathbf{x} - \mathbf{x}_{\mathrm{s}}\|_2 \leq u_{\max} t$ and $\|\mathbf{x} - \mathbf{x}_{\mathrm{g}}\|_2 \leq u_{\max}(T - t)$. Adding the two inequalities we get,

$$\Omega(T) = \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}_{\mathrm{s}}\|_2 + \|\mathbf{x} - \mathbf{x}_{\mathrm{g}}\|_2 \leq u_{\max} T\} \tag{6.8}$$

The TIS in Equation 6.8 in this case has the same prolate hyper-spheroid form as the L_2-Informed Set [12]. Thus, the TIS can be seen as a generalization of the L_2-Informed Set.

6.4 Time-Informed Exploration Algorithm

6.4.1 Ellipsoidal Approximations

Although obtaining the exact reachable sets defined in Equation 6.2, Equation 6.3 may not be computationally tractable, various techniques have been proposed to obtain tight over-approximations of these sets. These include application of polytopes and zonotopes [73], ellipsoidal calculus [72] and formulating reachability problem as a Hamilton-Jacobi-Bellman (HJB) PDE [71]. In this work, we use the ellipsoidal technique which provides a scalable framework for reachability analysis of robots with linear-affine dynamics. However, as discussed later on, the HJB reachability formulation can be used to extend the algorithms proposed in this work for general cost-functions and non-linear systems.

Consider the special case of linear kino-dynamic systems. Concretely, the constraint Equation 6.1b is $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$, with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$. Then, $\mathcal{X}_f[t_0, t]$ and

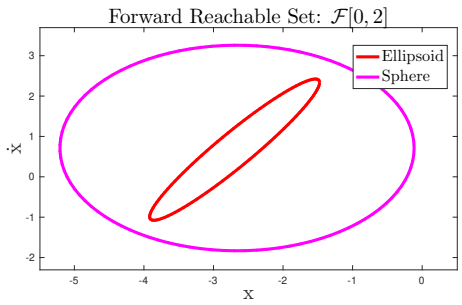


Figure 6.3: Comparing the forward reachable set $\mathcal{F}[0,t]$ for the 2D system at $t=2$ using the hyper-sphere and ellipsoidal approximation.

$\mathcal{X}_b[t,t_f]$ can be defined as

$$\begin{aligned}\mathcal{X}_f[t_0,t] &= \{\mathbf{x}\in\mathcal{X} \mid \exists\ \mathbf{u}:[t_0,t]\to\mathcal{U}, \text{s.t}\ \mathbf{x}=\mathrm{e}^{A(t-t_0)}\mathbf{x}_\mathrm{s}+\int_{t_0}^{t}\mathrm{e}^{A(t-\tau)}B\mathbf{u}(\tau)\,\mathrm{d}\tau\},\\ \mathcal{X}_b[t,t_f] &= \{\mathbf{x}\in\mathcal{X} \mid \exists\ \mathbf{u}:[t,t_f]\to\mathcal{U}, \text{s.t}\ \mathbf{x}=\mathrm{e}^{-A(t_f-t)}\mathbf{x}_\mathrm{g}-\int_{t}^{t_f}\mathrm{e}^{-A(\tau-t)}B\mathbf{u}(\tau)\,\mathrm{d}\tau\}.\end{aligned} \tag{6.9}$$

Here, $\mathbf{x}_\mathrm{g}\in\mathcal{X}_\mathrm{goal}$. A hyper-sphere over-approximation to the above sets can be constructed as follows [73],

$$\begin{aligned}\mathcal{F}[t_0,t] &= \{\mathbf{x}\in\mathcal{X}\ |\|\mathbf{x}-\mathrm{e}^{A(t-t_0)}\mathbf{x}_\mathrm{s}\|_2\le r(t_0,t,u_\mathrm{max})\},\\ \mathcal{B}[t,t_f] &= \{\mathbf{x}\in\mathcal{X}\ |\|\mathbf{x}-\mathrm{e}^{-A(t_f-t)}\mathbf{x}_\mathrm{g}\|_2\le r(t,t_f,u_\mathrm{max})\},\\ r(t_1,t_2,&u_\mathrm{max}) = (\mathrm{e}^{\|A\|_2(t_2-t_1)}-1)\|B\|u_\mathrm{max}/\|A\|_2.\end{aligned} \tag{6.10}$$

Here, $\|M\|_2$ represents the induced two norm (maximum singular value) for a matrix M. However, the above over-approximation might be too conservative for the current application. See Figure 6.3. If the reachable sets are overtly conservative and $\lambda(\Omega(T))\approx\lambda(\mathcal{X})$, then TIE may result in little or no focus of the search.

In contrast, the ellipsoidal technique [72] approximates the reachable sets as ellipsoids,

$$\mathcal{E}(\mathbf{x}_c,Q)=\{\mathbf{x}\in\mathbb{R}^n|\langle\mathbf{x}-\mathbf{x}_c,Q^{-1}(\mathbf{x}-\mathbf{x}_c)\rangle\le 1\}. \tag{6.11}$$

Here, $\mathbf{x}_c$ is the center and Q is the positive definite shape matrix of the ellipsoid. Forward

and backward reachable sets, $\mathcal{F}[0,t], \mathcal{B}[t,T]$ can be obtained by solving an ordinary differential equation (ODE) for the center and shape matrix. Please see the Ellipsoidal Toolbox[2] documentation for a brief overview. Note that the boundary conditions for the forward and backward reachable set ODE are the start and goal ellipsoids respectively. From the problem definition in Equation 6.1, the start ellipsoid is encoded as a hyper-sphere with negligible radius around the center $\mathbf{x}_{\mathrm{s}}$. The goal set $\mathcal{X}_{\mathrm{g}}$ is represented also as a hyper-sphere with a set radius around a center $\mathbf{x}_{\mathrm{g}} \in \mathcal{X}_{\mathrm{g}}$. The ODE for the shape matrix can be solved and stored off-line. An analytical solution for the ODE describing the center's trajectory can also be constructed. Thus, a "library" of reachable sets $\mathcal{F}[0,t], \mathcal{B}[t,T]$ can be created off-line to be used in the sampling and vertex inclusion algorithm described below. This library stores the value of center vector $\mathbf{x}_c$ and matrices Q, L of the forward and backward reachable sets. Here L is obtained using the Cholesky decomposition of Q, $Q = LL^{\mathsf{T}}$ and is used for generating samples inside $\mathcal{E}(\mathbf{x}_c, Q)$ [12]. Please see Figure 6.2 for a visualization of $\mathcal{F}[0,t]$ and $\mathcal{B}[t,T]$ constructed using the ellipsoid technique.

6.4.2 Sampling Algorithm

The algorithm 10 describes a procedure to generate a new sample $\mathbf{x}_{\mathrm{rand}}$ in $\Omega(T)$. Notice from Equation 6.5 that $\Omega(T)$ consists of a union over the intersections of sets. Devising a direct sampling technique to generate uniform random samples in $\Omega(T)$ (as done for the L_2 Informed Set in [12]) is hence a challenging task. The proposed algorithm proceeds by first sampling a time t in the interval $(0,T)$ according to a probability distribution $p_{[0,T]}(t)$ (line 2). Ideally, to generate uniform random samples in $\Omega(T)$ with respect to the Lebesgue measure, this distribution needs to be $p_{[0,T]}(t) = \lambda(\mathcal{F}[0,t] \cap \mathcal{R}_b[t,T])/\lambda(\Omega(T))$. However, calculating and sampling from this distribution may not be tractable for general higher dimensional systems. Hence, for the sake of simplicity, we choose $p_{[0,T]}(t)$ to be uniform over the interval $[0,T]$. Given t, the sets $\mathcal{F}[0,t], \mathcal{B}[t,T]$ can then be obtained from the

[2] http://systemanalysisdpt-cmc-msu.github.io/ellipsoids/doc/main_manual.html

Algorithm 10: Sampling Algorithm

```
1  generateSample (T) :
2      t ~ p_[0,T](t);
3      for i = 1 : n_s do
4          if λ(F[0,t]) < λ(B[t,T]) then
5              x_cand ← sampleUniform(F[0,t]);
6              if x_cand ∈ B[t,T] then
7                  x_rand ← x_cand;
8                  return x_rand;
9          else
10             x_cand ← sampleUniform(B[t,T]);
11             if x_cand ∈ F[0,t] then
12                 x_rand ← x_cand;
13                 return x_rand;
14     x_rand ← sampleUniform(X);
15     return x_rand;
```

library of stored reachable sets as discussed in the previous section. We leverage the fact that $\mathcal{F}[0,t] \cap \mathcal{B}[t,T] \neq \varnothing$ from Lemma Theorem 4 to generate a $\mathbf{x}_{\text{rand}} \in \mathcal{F}[0,t] \cap \mathcal{B}[t,T]$. If the Lebesgue measure of $\mathcal{F}[0,t]$ is less than $\mathcal{B}[t,T]$, a uniform sample is generated in $\mathcal{F}[0,t]$ and checked if it belongs to $\mathcal{B}[t,T]$, otherwise, $\mathcal{B}[t,T]$ is sampled and checked if it belongs to $\mathcal{F}[0,t]$ (lines 4-13). Notice from Figure 6.2 that $\lambda(\mathcal{F}[0,t])$ increases and $\lambda(\mathcal{B}[t,T])$ decreases as t varies from 0 to T. An efficient algorithm for generating uniform samples inside a hyper-ellipsoid is discussed in [12]. If no $\mathbf{x}_{\text{rand}} \in \mathcal{F}[0,t] \cap \mathcal{B}[t,T]$ can be generated in n_s attempts, the algorithm returns a uniform random sample from the search-space $\mathcal{X}$ (line 14-15).

6.4.3 Vertex Inclusion Algorithm

The vertex inclusion procedure, described in algorithm 11, accepts a candidate vertex if it lies in $\Omega(T)$. Consider a candidate vertex $\mathbf{v}$ with cost-to-come t, i.e., the cost of trajectory from $\mathbf{x}_{\text{s}}$ to $\mathbf{v}$ is t. Since the cost-to-come is t, we have $\mathbf{v} \in \mathcal{F}[0,t]$. Thus, if $\mathbf{v} \in \mathcal{R}_b[t,T]$, then $\mathbf{v} \in \Omega(T)$. The proposed algorithm discretizes the interval $[t,T]$ with a step-size δ. A

Algorithm 11: Vertex Inclusion Algorithm

```
1 includeVertex (v, t, T) :
2     if t > T then
3         return false ;
4     foreach τ ∈ {t + δ, t + 2δ, ... T} do
5         if v ∈ B[t, τ] then
6             return true ;
7     return false;
```

vertex is accepted if it lies in any $\mathcal{B}[t, \tau]$, for $\tau \in \{t + \delta, t + 2\delta, \dots T\}$ (line 4-6). The sets $\mathcal{B}[t, \tau]$ are again obtained from the stored library of reachable sets.

In order to maintain the theoretical guarantees of TIE, an over-estimate of the solution cost T is required. This over-estimate can be obtained (and updated) after the planner discovers (and then improves) an initial, sub-optimal solution. Also, learning-based methods similar to [49] can be used to obtain an estimate of the solution cost given a planning environment. In this work, the above algorithms are called only after an initial solution is discovered.

6.5 Experiments and Discussion

Benchmarking experiments were performed by pairing different exploration strategies with the SST planner [31]. All algorithms were implemented in C++ using the OMPL framework [54], and the tests were run using OMPL's standardized benchmarking tools [62]. The data was recorded over 100 trials for all the cases on a 64-bit laptop PC with 16 GB RAM and an Intel i7 Processor, running Ubuntu 16.04 OS. The performance of the proposed exploration strategy was benchmarked against uniform sampling (Uni) and uniform sampling combined with Informed propagation (IP). Informed propagation essentially rejects expansion vertices with cost-to-come $t > T$, if there exists a sub-optimal solution with cost T. If $t < T$, then forward propagation from the vertex is done for at most $T - t$ duration. Thus, Informed Propagation (IP) prohibits exploration outside the set $\bigcup_{0 \leq \tau \leq T} \mathcal{F}[0, \tau]$. The

proposed Time-Informed exploration (TIE) algorithm uses the sampling and vertex inclusion procedures described in Algorithms algorithm 10 and algorithm 11 with $n_s = 10$ and $\delta = 0.1$. The SST planner parameters, namely, the selection and pruning radius were set to standard OMPL values of 0.2 and 0.1 respectively. The L_2-norm was used as the distance function. A description of different case-studies is given below.

2D System: Consider a 2D kino-dynamic system $\dot{\mathbf{x}} = A_{2\times2}\mathbf{x} + B_{2\times1}u$ with, $\mathbf{x} = [x, \dot{x}]^\mathsf{T}$ and

$$A_{2\times2} = \begin{bmatrix} 0.0 & 0.5 \\ -0.1 & 0.2 \end{bmatrix}, \quad B_{2\times1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{6.12}$$

The set-up of the planning problem is illustrated in Figure 6.1, with $\mathbf{x}_\mathrm{s} = [-3\ 0]^\mathsf{T}, \mathbf{x}_\mathrm{g} = [3\ 0]^\mathsf{T}, \mathcal{X}_\mathrm{g} = \mathcal{E}(\mathbf{x}_\mathrm{g}, 0.25\ \mathrm{I}_2), u \in [-0.5\ 0.5]$. Here, I_2 represents the 2×2 identity matrix.

8D System: The 2D system described above is extended to a 8D system $\dot{\mathbf{x}} = A_{8\times8}\mathbf{x} + B_{8\times4}\mathbf{u}$, with $\mathbf{x} = [x_1\ \dot{x}_1\ x_2\ \dot{x}_2\ x_3\ \dot{x}_3\ x_4\ \dot{x}_4]^\mathsf{T}, \mathbf{u} = [u_1\ u_2\ u_3\ u_4]^\mathsf{T}$ and

$$\begin{aligned} A_{8\times8} &= \mathrm{blkdiag}[A_{2\times2}, A_{2\times2}, A_{2\times2}, A_{2\times2}], \\ B_{8\times4} &= \mathrm{blkdiag}[B_{2\times1}, B_{2\times1}, B_{2\times1}, B_{2\times1}]. \end{aligned} \tag{6.13}$$

The single obstacle in 2D case was extended to 8D by adding a length of 2 units symmetrically in the extra dimensions. Also, $\mathbf{x}_\mathrm{s} = [-2\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^\mathsf{T}, \mathbf{x}_\mathrm{g} = [2\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^\mathsf{T}, \mathcal{X}_\mathrm{g} = \mathcal{E}(\mathbf{x}_\mathrm{g}, \mathrm{I}_8), u_i \in [-1\ 1], i \in \{1, 2, 3, 4\}$.

Moon-lander Robot: A simplified version of a planar "moon-lander robot" is illustrated in Figure 6.4. The robot has three thrusters F_l, F_r and F_t acting in the left, right and up direction respectively. In the absence of upwards thrust, the robot falls under gravity. The

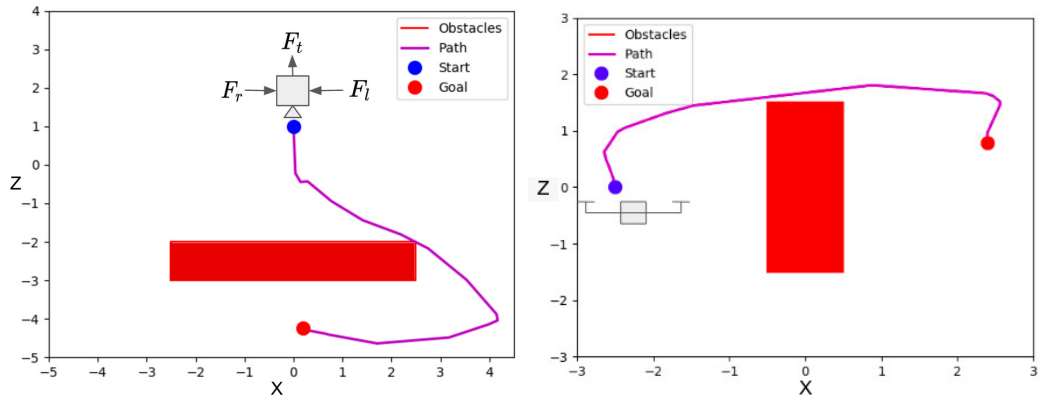


Figure 6.4: A schematic for the moon-lander robot (left) and quadrotor (right) simulation cases with sample solution paths found by the proposed algorithm after 40 sec of planning time.

dynamics of the robot is assumed to be as follows.

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x \\ z \\ \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ z \\ \dot{x} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} F_l \\ F_r \\ F_t \end{bmatrix}. \tag{6.14}$$

The start, goal and admissible control space were set as follows: $\mathbf{x}_{\mathrm{s}} = [0\ 1\ 0\ -2]^{\mathsf{T}}, \mathbf{x}_{\mathrm{g}} = [0\ -4\ 0\ 0]^{\mathsf{T}}, \mathcal{X}_{\mathrm{g}} = \mathcal{E}(\mathbf{x}_{\mathrm{g}}, 0.25\ \mathrm{I}), F_l \in [0\ 1], F_r \in [0\ 1], F_t \in [-2\ 2]$. The objective is to land the robot in time-optimal fashion.

Planar Quadrotor model: A linearized quadrotor model for longitudinal flight based on [74] can be written as $\dot{\mathbf{x}} = A_{6\times6}\mathbf{x} + B_{6\times2}\mathbf{u}$, with $\mathbf{x} = [x\ z\ u\ w\ q\ \theta]^{\mathsf{T}}, \mathbf{u} = [f_t\ \tau_y]^{\mathsf{T}}$ and

$$A_{6\times6} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -g \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} B_{6\times2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1/m & 0 \\ 0 & 1/I_y \\ 0 & 0 \end{bmatrix} \tag{6.15}$$

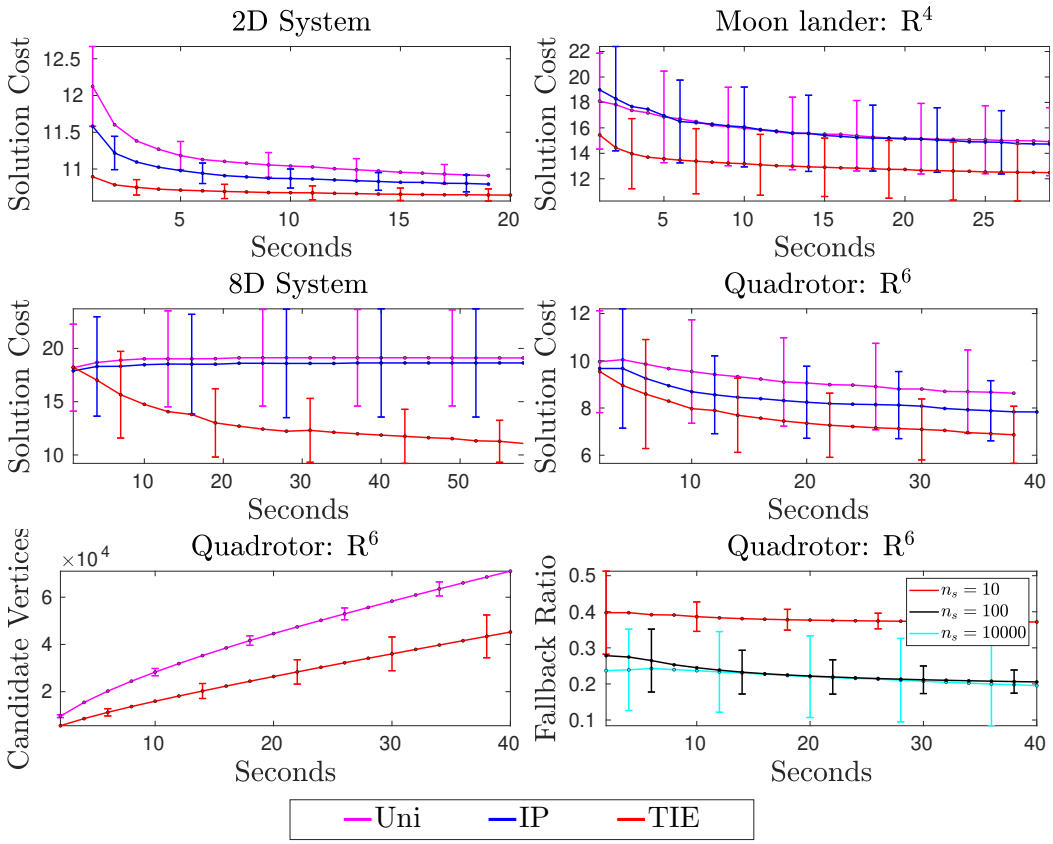


Figure 6.5: Convergence plots for the numerical experiments. Using the proposed TIE leads to a faster convergence in all cases (red plot). The bottom left figure illustrates number of candidate vertices generated using uniform and TIE exploration method. The bottom right figure plots the fallback ratio for different values of n_S. Solid lines indicate the value averaged over 100 trials and the error bars represent the standard deviation.

The start, goal and admissible control space were set as follows: $\mathbf{x}_{\mathrm{s}} = [-2.5\,0\,0\,0\,0\,0]^{\mathsf{T}}$, $\mathbf{x}_{\mathrm{g}} = [2.5\,0\,0\,0\,0\,0]^{\mathsf{T}}$, $\mathcal{X}_{\mathrm{g}} = \mathcal{E}(\mathbf{x}_{\mathrm{g}}, \mathrm{I})$, $f_t \in [-1\ 1]$, $\tau_y \in [-1\ 1]$. The set-up for time-optimal planning problem is shown in Figure 6.4.

The results of the numerical simulations are illustrated in Figure 6.5. It can be seen that Informed Propagation (blue) performs better than the naïve uniform exploration (magenta). However, using the proposed TIE strategy, a combination of algorithm 10 and algorithm 11, outperforms the other methods in all cases. Note that for a planner such as SST, the sampling procedure influences the vertex to be selected for forward propagation. Generating random samples $\mathbf{x}_{\mathrm{cand}} \in \Omega(T)$ biases the selection of vertices in the TIS for expansion. After a vertex is selected, expansion is performed by forward propagating the

system dynamics to generate a new candidate vertex $\mathbf{v}$. The vertex inclusion algorithm then ensures that the candidate vertex $\mathbf{v} \in \Omega(T)$. Thus, the combination of the proposed sampling and inclusion algorithm avoids redundant exploration focuses search, and leads to a faster convergence in all cases. In order to study the computational cost incurred by the TIE procedure, the quadrotor simulation was run without obstacles. Compared to uniform sampling, TIE generates a lower number of feasible, candidate vertices for inclusion in the planner tree, as illustrated in Figure 6.5. Future work will explore leveraging GPUs and operating on batch of samples and reachable sets in parallel. Parameter n_s controls the maximum number of attempts made to generate a new sample $\mathbf{x}_{\text{cand}} \in \Omega(T)$ before a uniform random sample is returned. In order to analyze the effect of n_s, the "fallback ratio" is defined as,

$$\text{Fallback Ratio} = \frac{\text{Number of Fallbacks to Uniform Sampling}}{\text{Number of Calls to the TIE Sampler}}$$

The fallback ratio was found to be negligible for the lower dimensional 2D system. It is relatively higher for the 6D quadrotor simulation run in a no-obstacle environment (see Figure 6.5). This ratio can be decreased by increasing n_s. However, a large value of n_s corresponds to a larger amount of computations invested in the sampling procedure, which can adversely impact the convergence of solution cost. Note that while the sampling algorithm may return $\mathbf{x}_{\text{cand}}$ outside the TIS if n_s attempts are exhausted, the vertex inclusion procedure ensures that a candidate vertex $\mathbf{v}$ is incorporated in the planner tree only if it lies in the TIS.

CHAPTER 7

NON-TRIVIAL QUERY SAMPLING FOR LEARNING TO PLAN

7.1 Motivation

The previous chapters explored various "heuristic" based techniques for SBMP algorithms. These methods leverage various sources of information available, such as heuristic functions, graph state and collision checking information to improve the performance of planners. Techniques such as Informed Sampling, Relevant Region and LES accelerate the convergence of planners while still maintaining some theoretical guarantees. Although such methods can improve the performance of motion planning algorithms, many require handcrafted heuristics for efficacy. Also, these methods do not leverage prior experience or the data gathered from expert demonstrations offline. Deep learning based approaches for motion planning address these two limitations by generating a dataset of high quality paths in various environments. In the offline phase, this dataset is used to train a deep neural network (DNN) model to predict quantities of interest, such as cost-to-go [75], next point along the optimal path [76] or a sampling distribution [77]. The DNN model can then be used online to focus search and dramatically increase the efficiency of planning algorithms. However, note that these learning-based methods may not offer similar theoretical guarantees as the heuristic-based methods.

The data generation process for learning-based methods involves sampling and solving a set of queries (start, goal pairs) in a given environment. For many applications, the map/environment in which the robot needs to operate may be fixed or given apriori. However, the query sampling distribution can be modified in order to extract more informative paths beneficial to the learning process. Conventionally, uniform random sampling is used to generate the start and goal states. Many queries in this uniformly sampled dataset

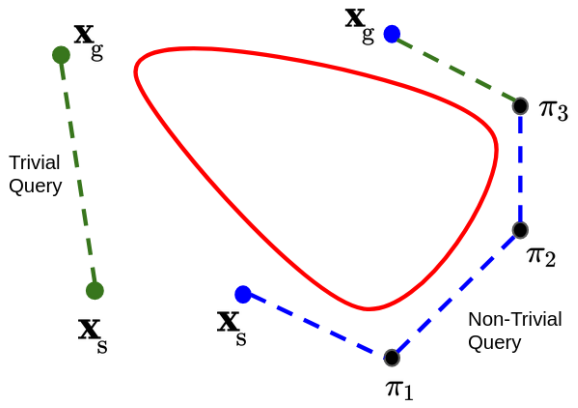


Figure 7.1: Schematic for the proposed data generation method. Instead of trivial queries that can be solved by a greedy connection, more non-trivial queries are added. In addition, the data pruning procedure filters out the trivial part $(\pi_3, \mathbf{x}_g)$ of the path.

can be solved by greedily connecting the start and goal state using a steering function (if available). Such a steering function gives the optimal path between any two states after relaxing the collision constraint. This work proposes adding more "non-trivial" queries to the dataset, which cannot be solved by a simple greedy connection. The efficiency of the neural planners can be boosted by training deep models on this dataset comprising of relatively more complex paths. This is demonstrated by creating datasets with different degrees of non-triviality and benchmarking the performance of the neural planner on various robotic planning tasks.

7.2 Related Work

Exciting progress has been made at the intersection of learning and motion planning in recent years. To improve the performance of discrete space planners, methods that leverage reinforcement learning [78] and imitation learning [79], [80] have been proposed. Neural A* [81] presents a data-driven approach that reformulates conventional A* as a differentiable module. Huh et al present a higher-order function network in [82] that can predict cost-to-go values to guide the A* search. For sampling-based planning, techniques such as [83] and [39] learn a local sampling strategy for global planning. Zhang et al present a deep learning based rejection sampling scheme for SBMP in [84]. Ichter et al [77] train

a conditional variational autoencoder (CVAE) model to learn sampling distributions and generate samples along the solution paths. NEXT [75] learns a local sampling policy and cost-to-go function using "meta self-improving learning". Kuo et al [85] input the planner state and local environment information into a deep sequential model to guide search during planning. Approaches such as [86], [87] focus on identifying critical or bottleneck states in an environment to generate a sparse graph while planning. While the above techniques may differ in terms of their model architectures or outputs, they do not tackle the problem of improving the data-generation process via query sampling for increasing the efficacy of neural planners.

In [88], Huh et al extend their previous work to present a query sampling technique for non-holonomic robots. However, this technique is purely based on dyamics of the robot, does not utilize obstacle information and is only applicable for car-like robots. Recently proposed OracleNet [89] and Motion Planning Networks (MPNet) algorithm [76] learn a deep model to recursively predict the next state along the solution path, given a query. Authors in [76] use "Active Continual Learning (ACL)" to improve the data-efficiency of the training process. Similar to the DAGGER algorithm [90] the ACL process involves training a MPNet deep model on a set of expert demonstrations for a $N_c > 0$ number of initial iterations. The ACL algorithm then finds the cases where the MPNet planner fails and invokes the expert planner only to solve them. The solutions generated by the expert planner are stored in a replay buffer to be used in training. ACL also leverages the "Gradient Episodic Memory (GEM)" technique [91] to alleviate the problem of catastrophic forgetting during the learning process. While more data-efficient, the ACL process can be tricky to implement and computationally more expensive, as it involves interleaving the training process with running the neural planner multiple times. The performance of ACL models can also be worse than that of batch-offline models in many cases [76]. In contrast, this work proposes a modified query sampling procedure for data-generation and trains all models in a batch-offline manner.

Algorithm 12: Neural Planner

NeuralPlanner ($\boldsymbol{x}_{\mathrm{s}}, \boldsymbol{x}_{\mathrm{g}}, \mathcal{X}_{\mathrm{obs}}$) **:**
  $\pi \leftarrow \{\mathbf{x}_{\mathrm{s}}\}$;
  for $i = 1 : N_{\mathrm{plan}}$ **do**
    if steerTo$(\pi_{\mathrm{end}}, \boldsymbol{x}_{\mathrm{g}}, \mathcal{X}_{\mathrm{obs}})$ **then**
      $\pi \leftarrow \pi \cup \{\mathbf{x}_{\mathrm{g}}\}$;
      break;
    else
      $\mathbf{x}_{\mathrm{new}} \leftarrow$ PNet$(\pi_{\mathrm{end}}, \mathbf{x}_{\mathrm{g}})$;
      $\pi \leftarrow \pi \cup \{\mathbf{x}_{\mathrm{new}}\}$;
  if $\boldsymbol{x}_{\mathrm{g}} \notin \pi$ **then**
    return $\emptyset$;
  if Feasible(π) **then**
    return π;
  else
    return Replan(π);

7.3 Supervised Learning for Planning

Learning based methods use the data gathered from successful plans to train models in the offline phase. In the online phase, this learned model can be used to solve (Equation 2.3) or assist the classical planners. The data generation process involves creating an environment and sampling a set of $K_{\mathrm{train}} > 0$ queries or (start, goal) pairs in $\mathcal{X}_{\mathrm{free}}$. Let $\mathcal{Q} = \mathcal{X}_{\mathrm{free}} \times \mathcal{X}_{\mathrm{free}}$ denote the "query-space", where $\times$ represents the Cartesian product between two sets. A classical planner is then used to solve the path planning problem (Equation 2.3) for each of the K_{train} queries and obtain a good quality solution. A quantity of interest to be learned as output (such as cost-to-go, next state on the optimal path etc) is extracted from these solution paths. The objective of the training process is to learn a function f_θ (usually a deep neural network), on the dataset $\mathcal{D}$ by minimizing an empirical loss with respect to weight parameters θ

The MPNet procedure involves learning a planning network PNet, that predicts the next state on the optimal path, given a current state, a goal state and environment information as

the input. A neural planning algorithm, in line with the one described in [76], is illustrated in Algorithm algorithm 12. Given a query $\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{g}$, the path π to be returned is initialized with the start-state. This path π can be represented as an ordered list of length $L \geq 2$, $\pi = \{\pi_1, \pi_2 \dots \pi_L\}$. Let π_end denote the last point on the path π. At each iteration, the neural planner attempts a greedy connection to the goal using the steerTo function. For length-optimal or geometric planning case, the steerTo connects any two points using a straight line. For car-like robots, this steering function can use the Dubins curves for dynamically feasible connections [92]. To probe the feasibility of this greedy connection, the steerTo procedure discretizes the path and collision-checks the points on it. If the greedy connection π_end to $\mathbf{x}_\mathrm{g}$ is valid, the goal-state is appended to the path and the planning loop terminates. Else, the learned planning network PNet, is used to predict the next state on the optimal path. If the path π is infeasible, a neural replanning procedure is performed on this coarse path in an attempt to repair it. Please see [76] for more details about these procedures.

Planning with the MPNet neural planner, given in algorithm 12, involves forward passes through a neural network. This computational operation is known to have a constant time complexity since it is a matrix multiplication of a fixed size input vector with network weights to generate a fixed size output vector. In contrast, many SBMP algorithms such as RRT* are known to have $O(n \log n)$ complexity [17], where n is the number of samples. Thus, a well trained MPNet planner can outperform state-of-the-art SBMP methods such as BIT*, especially in higher dimensions, as shown in [76].

7.4 Non-trivial Query Sampling

Conventionally, uniform random sampling is used in the data generation process to obtain a query $\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{g} \in \mathcal{Q}$. However, this may result in the inclusion of "trivial" queries in the dataset, for which $\mathsf{steerTo}(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{g}) = \mathsf{True}$. In case of such trivial queries, the neural planning algorithm 12 terminates in the first iteration after processing lines 4-6. Thus, a key

Algorithm 13: Data Generation Algorithm

1 $\mathcal{D} \leftarrow \emptyset$;
2 $\mathcal{X}, \mathcal{X}_{\text{obs}} \leftarrow \mathsf{createEnvironment}()$;
3 **for** $j = 1 : K_{\text{train}}$ **do**
4 $\quad u_{\text{rand}} \sim U[0, 1]$;
5 $\quad$ **if** $u_{\text{rand}} < p_{\text{nt}}$ **then**
6 $\quad\quad \mathbf{x}_{\text{s}}, \mathbf{x}_{\text{g}} \leftarrow \mathsf{nonTrivialQuerySampling}(\mathcal{X}_{\text{free}})$;
7 $\quad$ **else**
8 $\quad\quad \mathbf{x}_{\text{s}}, \mathbf{x}_{\text{g}} \leftarrow \mathsf{uniformSampling}(\mathcal{X}_{\text{free}})$;
9 $\quad \pi \leftarrow \mathsf{solveQuery}(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{g}}, \mathcal{X}_{\text{obs}})$;
10 $\quad \mathcal{D} \leftarrow \mathsf{includeData}(\mathcal{D}, \pi)$;
11 **return** $\mathcal{D}$

observation is that steerTo procedure in algorithm 12, line 4 performs an implicit classification of queries, so that only "non-trivial" queries are passed over to the PNet. Concretely, the set of non-trivial queries can be defined as

$$\mathcal{Q}_{\text{nt}} = \{\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{g}} \in \mathcal{Q} \mid \mathsf{steerTo}(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{g}}) = \mathsf{False}\}. \tag{7.1}$$

This motivates the proposed data generation algorithm 13, which aims to increase the number of non-trivial data samples in $\mathcal{D}$. After an environment $\mathcal{X}, \mathcal{X}_{\text{obs}}$ is created, data is generated by solving a total of K_{train} queries. With probability p_{nt},
the proposed nonTrivialQuerySampling procedure is used to obtain $\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{g}} \in \mathcal{Q}_{\text{nt}}$. Else, conventional uniformSampling returns a query in $\mathcal{Q}$. A classical planner such as A* or BIT* then solves this query and outputs a good quality solution path π. Samples from this path π are appended to the dataset $\mathcal{D}$ with the proposed data inclusion procedure includeData.

A rejection sampling algorithm to generate new queries in $\mathcal{Q}_{\text{nt}}$ is given in algorithm 14. For N_{nt} number of attempts, uniform sampling is used to first generate a valid query $\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{g}} \in \mathcal{Q}$. The steerTo module then validates the connection between start and goal state. If found invalid, the corresponding non-trivial query is returned. Thus, this procedure intends to filter out trivial paths while maintaining the exploratory/coverage property

Algorithm 14: Non-trivial Query Sampling

```
nonTrivialQuerySampling ( X_obs ) :
    for i = 1 : N_max do
        (x_s, x_g) ← uniformSampling(X_free);
        valid ← steerTo(x_s, x_g);
        if not valid then
            return (x_s, x_g);
return (x_s, x_g);
```

of uniform sampling. Please see Figure 7.2 and Figure 7.5 for a visualization of queries generated using the proposed non-trivial sampling procedure.

The data inclusion algorithm 15 iterates over the segments of path π and logs the current state, goal state $(\pi_i, \pi_{\text{end}})$ as the input and the next state (π_{i+1}) as the output/label. However, if the flag $\mathsf{pruneData} = \mathsf{True}$, the algorithm skips including the data-sample $\{(\pi_i, \pi_{\text{end}}), \pi_{i+1}\}$ if $\pi_i, \pi_{\text{end}} \notin \mathcal{Q}_{\text{nt}}$. Thus, $\mathsf{pruneData} = \mathsf{True}$ ensures that only the non-trival segments of π are incorporated in $\mathcal{D}$. Please see Figure 7.1 for an illustration of this.

Depending on the topology of $\mathcal{X}_{\text{obs}}$, the neural planner may find it relatively harder to predict feasible paths in certain environments. The notion of non-trivial queries can be used to define a metric that captures this level of difficulty. Consider a "non-triviality ratio", which can be defined as,

$$\gamma_{\text{nt}} = \frac{\text{\# Non-trivial queries}}{\text{\# Uniformly sampled queries}}. \tag{7.2}$$

Thus, γ_{nt} is the ratio of number of non-trivial queries found in a (large enough) set of uniformly sampled queries. This ratio will be high for complex, cluttered and narrow-passage type environments and low for relatively simpler, single-obstacle type environments.

Algorithm 15: Data Inclusion Procedure

```
includeData ( 𝒟, π ) :
    for i = 1 : L − 1 do
        if pruneData then
            valid ← steerTo(π_i, π_end);
            if valid then
                continue;
        𝒟 ← 𝒟 ∪ {(π_i, π_end), π_{i+1}}
return 𝒟;
```

7.5 Experiments and Discussion

In order to benchmark the proposed data generation algorithm, the following procedure was implemented for each planning environment. First, four datasets with different parameter settings were created. These were as follows: $\mathcal{D}_0$ ($p_{\text{nt}} = 0,$ pruneData $=$ False), $\mathcal{D}_1$ ($p_{\text{nt}} = 0.5,$ pruneData $=$ False), $\mathcal{D}_2$ ($p_{\text{nt}} = 1.0,$ pruneData $=$ False), $\mathcal{D}_3$ ($p_{\text{nt}} = 1.0,$ pruneData $=$ True). Thus, $\mathcal{D}_0$ represents the dataset generated using the conventional uniform query sampling, whereas $\mathcal{D}_3$ is created using the proposed non-trivial query sampling and data pruning procedure. Four deep models, $\mathsf{PNet}_0, \mathsf{PNet}_1, \mathsf{PNet}_2, \mathsf{PNet}_3$ were then trained on their respective datasets. The neural network shape, size and the training parameters were held constant while learning all four models. Performance of the neural planner algorithm 12 using these four models was evaluated on 1) K_{test} number of new uniform queries and 2) K_{test} number of new non-trivial queries. Two performance metrics, namely, success ratio and cost ratio were considered. Success ratio gives the number of times (out of K_{test} in total) the neural planner was successful in finding a feasible (collision-free) solution. Cost ratio denotes the ratio of neural planner's solution cost to that of classical planner, averaged over K_{test} trials. Model training and evaluation process was performed using the Python PyTorch API on a 64 bit, 16 GB RAM laptop with Intel i7 processor and a NVIDIA GeForce RTX 2060 GPU. A description of robotic planning tasks along with a discussion of results is given below.

Point Robot: Four different 20×20 environments, illustrated in Fig. Figure 7.2, were considered for the case of point robot planning. Four datasets $\{\mathcal{D}_i\}_{i=0}^{3}$, as described above, were generated for each environment. A total of $K_{\text{train}} = 3000$ number of queries were sampled for each dataset. An A* planner, followed by post-processing/smoothening, was used to solve these queries and obtain length-optimal paths. A small padding of 0.8 units around the obstacles was propagated during the data generation step, and was relaxed during the final performance evaluation step. This was found to greatly boost the success ratio of the neural planner, while making slight compromise in the cost ratio metric. The performance metrics were logged by solving $K_{\text{test}} = 500$ number of unseen uniform and non-trivial queries with the four learned PNet models.

Rigid Body Planning: Figure 7.3 shows the instance of planning for a rigid robot in four 10×10 environments. A total of $K_{\text{train}} = 5000$ queries were considered to create each of the four datasets $\{\mathcal{D}_i\}_{i=0}^{3}$. All the queries were solved in the $\mathrm{SE}(2)$ space using OMPL's [54] implementation of the BIT* planner. An obstacle-padding of 0.4 units was propagated during the data-generation phase. The learned PNet models predicted a three dimensional $[x, y, \theta]$ vector representing the robot's pose. These models were evaluated on $K_{\text{test}} = 500$ unseen uniform and non-trivial queries. The BIT* planner was allowed a run-time of 3 seconds during the data-generation phase and 2 seconds during the evaluation phase.

n-link Manipulator Planning: To observe performance of the neural planner in higher dimensions, a planning problem for $2, 4$ and 6-link manipulator robot was considered. Please see Figure 7.5. The joint angles were constrained to lie between $-\pi$ and π. Four datasets $\{\mathcal{D}_i\}_{i=0}^{3}$ were created for each of the $2, 4$ and 6-link case by considering a total of $3000, 4000$ and 5000 queries respectively. These queries were solved using OMPL's BIT* planner with an padding of 0.8 units around the workspace obstacles. The final performance evaluation was done by solving $K_{\text{test}} = 500$ new queries with the neural planner. The BIT* planner was run for $2, 4, 6$ seconds during the data-generation stage and $1, 2, 4$ seconds during the evaluation stage for the $2, 4$ and 6-link planning respectively.

General trends seen from the results in Table 7.1, Table 7.2, Table 7.4 are as follows. In most cases, PNet_2 and PNet_3 outperform PNet_0 in terms of success ratio, where as the performance of PNet_1 is more sporadic. The cost ratio is relatively lower for the rigid body and 6-link case compared to others, as the BIT* planner may not find a good quality solution in the given planning time for these challenging cases. The success ratio for all PNet models is naturally higher over uniform test queries rather than non-trivial test queries. The success ratio also generally has an inverse relation with γ_{nt}, as seen strongly in the case of rigid body and n-link manipulator planning. For the case of point robot planning, all models perform well with a success rate of over 90% (see Table 7.1). However, slight performance gains due to the proposed method can be seen for Environments $0, 2, 3$. These gains are much more noticeable for the rigid body planning case (see Table 7.2). The PNet_3 model has the highest success ratio in all cases except one (Environment 3, Non-trivial Query), where its performance is comparable to PNet_1. The gradation in performance due to dimensionality and γ_{nt} can be seen clearly in the n-link planning case (see Table 7.4). The success ratio over uniform queries is in the range of $0.9, 0.8$ and 0.7 for the case of $2, 4$ and 6-link planning case respectively. For the relatively simpler 2-link planning case with $\gamma_{\text{nt}} = 0.225$, only small gains in the success ratio over non-trivial queries can be seen. However, the improvement in performance is much more evident for the higher dimensional 4 and 6-link cases. The PNet_3 model shows about a 25% increase in the success ratio over PNet_0 for the 6-link (non-trivial queries) case.

The neural planning algorithm 12 and the corresponding results discussed above assume the availability of a steering function. While this is readily available for cases such as geometric or non-holonomic (car-like) planning [92], it may not be computationally tractable others. To analyze the performance of PNet models without the $\mathsf{steerTo}$ function, simulations were performed by only executing the lines 8 and 9 of the neural planner algorithm 12 for maximum N_{plan} iterations. Instead of lines 4-6 in algorithm 12, the following termination condition was implemented, $\|\pi_{\text{end}} - \mathbf{x}_{\text{g}}\|_2 \leq \delta$, with a small $\delta > 0$.

As illustrated in Figure 7.4, the PNet_3 model, which has no trivial sample in its training dataset, naturally cannot solve a trivial query. Numerical results for the rigid body planning without the steerTo function and $\delta = 1.0$ are tabulated in Table 7.3. The success ratio of all PNet models is adversely affected in this case. The PNet_0 model performs the best on uniform queries in all environments, whereas the performance of PNet_3 is the worst. However, PNet_1 or PNet_2 show better performance over non-trivial queries in some cases. Thus, without a steering function, a uniformly sampled training dataset might best choice if the test queries are uniformly distributed too. However, a model trained over a dataset with an appropriate value of p_{nt} may perform better over non-trivial test queries. This makes a case for an ensemble model, as elaborated in the conclusions chapter.

Table 7.1: Point robot planning

Environment 0, $\gamma_{\text{nt}} = 0.311$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.970	1.004	0.894	1.006
PNet_1	0.980	1.002	0.954	1.007
PNet_2	0.974	1.003	0.918	1.010
PNet_3	0.980	1.002	0.930	1.007

Environment 1, $\gamma_{\text{nt}} = 0.402$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.992	1.031	0.986	1.075
PNet_1	0.976	1.029	0.940	1.064
PNet_2	0.986	1.027	0.966	1.063
PNet_3	0.992	1.029	0.988	1.067

Environment 2, $\gamma_{\text{nt}} = 0.414$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.966	1.051	0.880	1.117
PNet_1	0.960	1.051	0.906	1.125
PNet_2	0.950	1.044	0.916	1.120
PNet_3	0.962	1.056	0.900	1.113

Environment 3, $\gamma_{\text{nt}} = 0.628$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.944	1.056	0.864	1.156
PNet_1	0.958	1.069	0.896	1.167
PNet_2	0.956	1.063	0.898	1.163
PNet_3	0.974	1.095	0.912	1.191

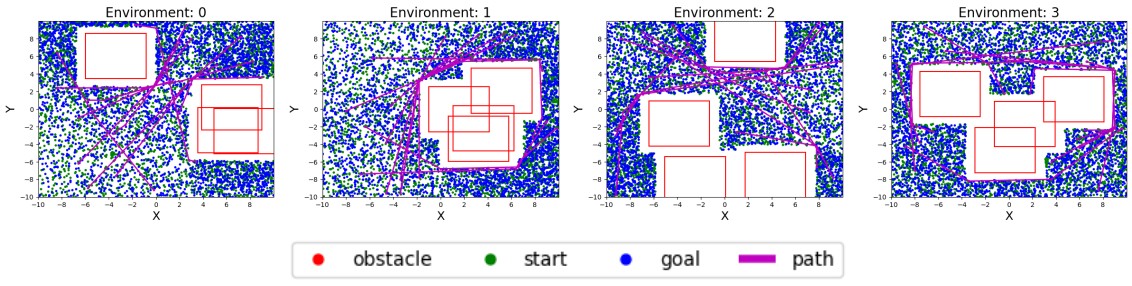


Figure 7.2: Four different environments for the point robot planning task. Data generated using the proposed non-trivial query sampling results in following (start, goal) distribution. Some of the length-optimal paths solved using a classical planner are illustrated in magenta

Table 7.2: Rigid body planning

Environment 0, $\gamma_{\text{nt}} = 0.503$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.828	0.985	0.656	0.972
PNet_1	0.842	0.985	0.622	0.972
PNet_2	0.834	0.979	0.698	0.960
PNet_3	0.856	0.978	0.784	0.958

Environment 1, $\gamma_{\text{nt}} = 0.659$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.788	0.948	0.714	0.874
PNet_1	0.784	0.948	0.686	0.878
PNet_2	0.792	0.971	0.638	0.889
PNet_3	0.856	0.950	0.776	0.865

Environment 2, $\gamma_{\text{nt}} = 0.556$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.862	0.965	0.770	0.900
PNet_1	0.898	0.963	0.776	0.907
PNet_2	0.902	0.963	0.806	0.908
PNet_3	0.922	0.970	0.876	0.912

Environment 3, $\gamma_{\text{nt}} = 0.690$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.722	0.985	0.606	1.000
PNet_1	0.714	0.991	0.660	1.042
PNet_2	0.756	0.994	0.650	0.996
PNet_3	0.772	0.981	0.652	1.006

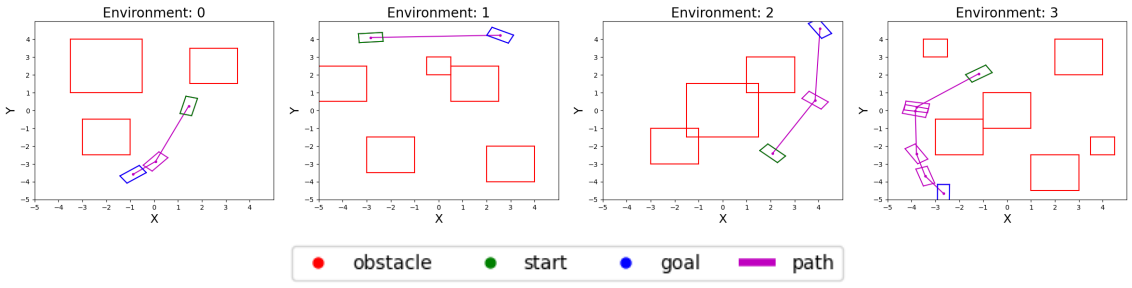


Figure 7.3: Four different environments for the rigid body planning task.

Table 7.3: Rigid Body without steerTo

Environment 0, $\gamma_{\text{nt}} = 0.503$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.624	0.982	0.414	0.957
PNet_1	0.592	0.983	0.372	0.979
PNet_2	0.564	0.977	0.392	0.960
PNet_3	0.228	0.985	0.212	0.978

Environment 1, $\gamma_{\text{nt}} = 0.659$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.498	0.971	0.380	0.936
PNet_1	0.476	0.965	0.430	0.923
PNet_2	0.456	0.974	0.372	0.946
PNet_3	0.266	0.970	0.208	0.968

Environment 2, $\gamma_{\text{nt}} = 0.556$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.684	0.943	0.514	0.872
PNet_1	0.684	0.944	0.506	0.889
PNet_2	0.630	0.929	0.550	0.890
PNet_3	0.284	0.937	0.270	0.931

Environment 3, $\gamma_{\text{nt}} = 0.690$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.514	0.986	0.374	0.990
PNet_1	0.504	0.996	0.426	1.005
PNet_2	0.502	1.004	0.396	1.004
PNet_3	0.302	1.001	0.196	0.979

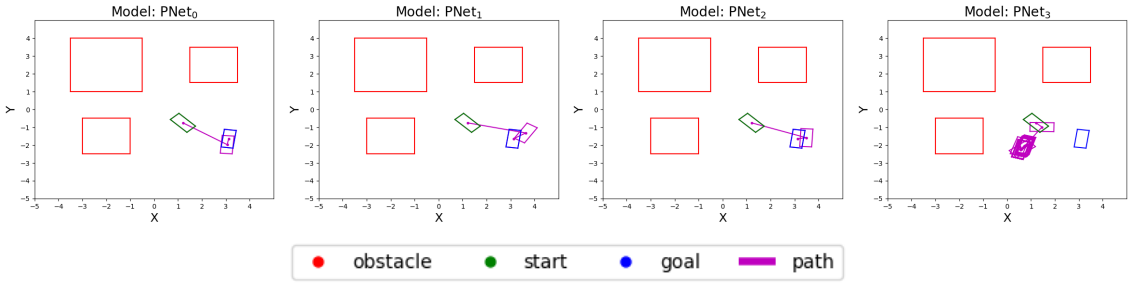


Figure 7.4: Solving a trivial query with the four learned models without the steerTo function. Models PNet_0,PNet_1,PNet_2 can successfully solve the query. However, the PNet_3 model, which does not have any trivial sample in its training dataset, is unable to solve it.

Table 7.4: n-link manipulator planning

2-link planning, $\gamma_{\text{nt}} = 0.225$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.984	1.002	0.924	1.061
PNet_1	0.990	1.004	0.956	1.060
PNet_2	0.984	1.003	0.962	1.062
PNet_3	0.988	1.004	0.972	1.068

4-link planning, $\gamma_{\text{nt}} = 0.366$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.862	1.005	0.722	1.003
PNet_1	0.876	0.997	0.718	0.991
PNet_2	0.890	0.994	0.772	1.003
PNet_3	0.928	0.860	0.82	0.992

6-link planning, $\gamma_{\text{nt}} = 0.608$

Model	Uniform Query		Non-trivial Query	
	success ratio	cost ratio	success ratio	cost ratio
PNet_0	0.716	0.982	0.512	0.972
PNet_1	0.650	0.980	0.454	0.966
PNet_2	0.712	0.984	0.528	0.971
PNet_3	0.776	0.975	0.644	0.958

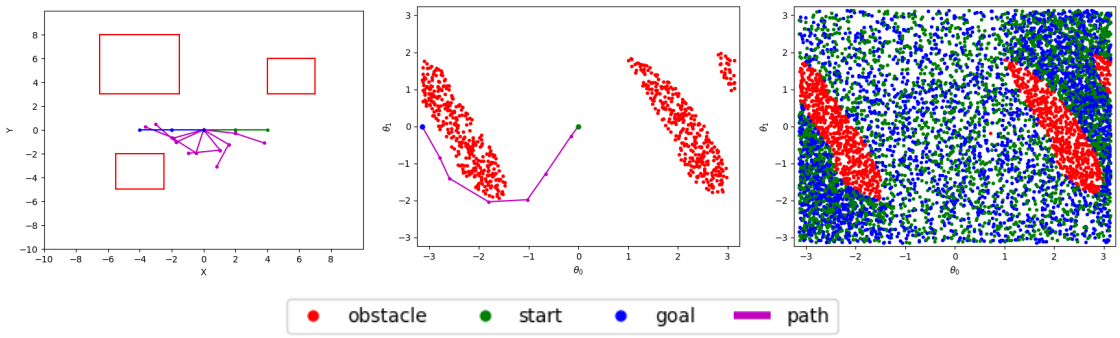


Figure 7.5: A schematic for n-link manipulator planning. Left figure shows the obstacles and solution path in the work-space of the robot. The corresponding obstacles and solution path in the configuration-space (middle). The (start, goal) distribution produced by the proposed non-trivial query sampling in the configuration-space (right).

CHAPTER 8

RACECAR ROBOT PLATFORM

The RACECAR robot (see Figure 8.1) is a powerful platform for robotics research and education. Initially developed by MIT [93], it is based on the 1/10 th scale Traxxas RC Rally Car. The objective of this work is to develop the hardware and software capabilities of the RACECAR robot in order to implement and test the planning algorithms proposed in this thesis. Concretely, this thesis provides a guide for assembling this platform, noting down parts that of the build that are unique to Georgia Tech/DCSL lab. Secondly, this thesis develops a Robotic Operating System (ROS) [1] based software stack for autonomous navigation. The software stack consists of a LiDAR based obstacle avoidance system for map exploration and a full SLAM, planning and control module for point to point navigation. Lastly, this thesis tests the software stack in ROS simulation and also on the physical hardware.

8.1 Platform Overview

An illustration of the RACECAR robot, along with its sensors and processing unit is given in Figure 8.1. A brief description of these relevant hardware components is given below.

- Motors and Speed Controllers: The robot has one electric motor to drive the back wheels and one servo motor to steer the front wheels. An open-source Electronic Speed Controller (ESC), called the VESC is used to program the firmware of the motor drivers. A measurement of the vehicle speed is extracted from the VESC module.

- NVIDIA Jetson TX2: The main processing unit of the RACECAR is the NVIDIA

[1] https://www.ros.org/

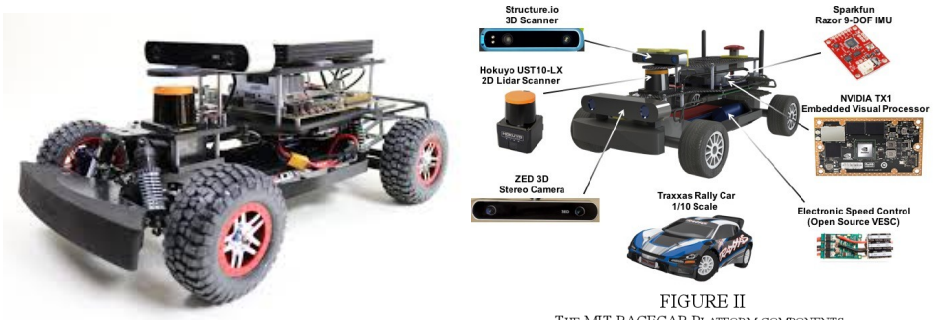


Figure 8.1: RACECAR robot hardware and its relevant components. Picture credits [93]

Jetson Tegra X2 (Jetson TX2). It has a 256-core GPU and a 8 GB of memory. It runs on a electrical power of mere 7.5 Watts. This makes it ideal for high compute robotic applications.

- Hokuyo UST10-LX Laser Range Finder: The Hokuyo LiDAR rotates at a 40 Hz speed, providing 270-degree field of view at 1/4 degree resolution.
- The Stereo Camera: Stereolab's ZED Camera provides a synchronized video feed from two cameras. These two images can be used to reconstruct depth information using the stereo matching techniques.
- IMU: RACECAR comes equipped with a Sparkfun Razor 9-DOF Interial Measurement Unit (IMU).
- Structure Sensor: The Structure.io sensor provides RGB-D data using the strucutre light method, which is similar to the Microsoft Kinect. This consists of a 640×480 pixel image, delivered at a rate of 40 frames per second. It also perceives depth in the range of 0.4 to 3.5 meters.

A step-by-step guide for the assembly of this platform is given in the docs folder of the racecar github repo [2]

[2]https://github.gatech.edu/DCSL/racecar_control/blob/master/docs/RACECARJ_Manual.docx

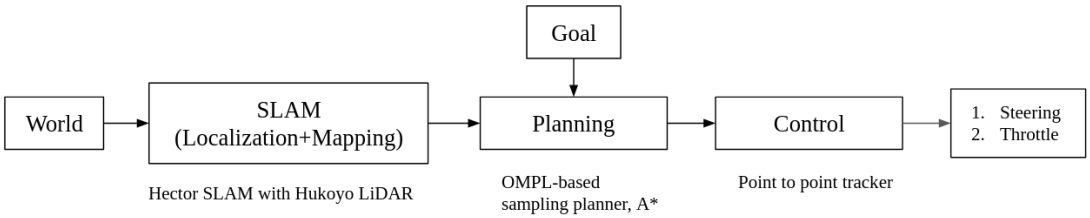


Figure 8.2: Architecture of the ROS autonomous navigation stack for the RACECAR robot.

8.2 Autonomous Navigation Stack

The Jetson computer runs an Ubuntu Linux operating system, on which ROS can be installed. Using ROS, each software component can be separated into modules called "nodes". Each sensor, actuator, mapping, localization, planning and control module has an associated ROS node. These nodes use "ROS messages" to exchange information with each other. Additionally, ROS provides tools to visualize data and simulation environments to rapidly test the algorithms. A ROS based autonomous navigation stack was developed for the RACECAR robot. The instructions to set-up and compile this package are given here[3]. A brief description of different modules in this stack is given below.

Obstacle Avoidance: This module can be used by the RACECAR robot to autonomously explore an environment while avoiding collisions. This light-weight script is also an efficient way to verify the status of robot's LiDAR sensor, motor actuators and corresponding ROS nodes. This module takes as an input, the distance to obstacle readings from LiDAR for every angle $[-135°, 135°]$, and outputs a steering angle and speed command for a collision free navigation. The LiDAR data processing component of this module filters out noisy readings and calculates the direction and distance to the nearest obstacle for the robot. Based on this angle and distance, the robot either 1) goes forward with no steering input 2) goes forward with a steering input to avoid obstacles 3) stops, then reverses if its too close to an obstacle.

Point-to-point navigation: Unlike the obstacle avoidance algorithm described above, this

[3]https://github.gatech.edu/DCSL/racecar_control/blob/master/README.md

module executes full SLAM + planning + control to drive the robot from its current state to a user defined goal state on the map (see Figure 8.2).

- SLAM: The RACECAR robot executes simultaneous localization and mapping (SLAM) using the LiDAR data. This thesis uses the Hector-SLAM algorithm [94], which is available as an open-source ROS package. This generates a map and pose for the robot with their respective ROS topics. Hector-SLAM uses a robust scan matching method for mapping, which is the process of aligning laser scans with each other or with an existing map. An Extended Kalman Filter (EKF) algorithm is used for localization. The map is encoded as a 2D occupancy grid. Through this data-structure, the probability information of a grid cell being in the obstacle space can be queried.

- Planning: Given a map the pose information from the SLAM module, the planning block takes in a user specified goal point and outputs a collision free path from the start state of the robot. An interactive marker, created by a ROS server, can be manipulated by the user to input a goal state on the map. The path planning problem is then solved using discrete-space algorithms such as A*, or any OMPL based planners. The solution path is outputted as a series of way-points. A path smoothing algorithm based on "short-cutting" [54] is can be employed as post-processing. This iterative algorithms tries to shorten the length of the path while maintaining its feasibility. Finally, the first way-point to track along the computed path is sent to the control/path-tracking module. As the robot moves through the world and gathers more information, the map data might change. A ROS callback function updates the occupancy grid data-structure, following which a collision check is initiated for the current planned path. If found infeasible, the planning algorithm is called again and a new path is calculated.

- Way-point tracker: The tracking algorithm flow is described in algorithm 16. Given a user defined goal state $\mathbf{x}_g$, it is transformed to the base link frame, giving $\mathbf{x}_{go}$. Please

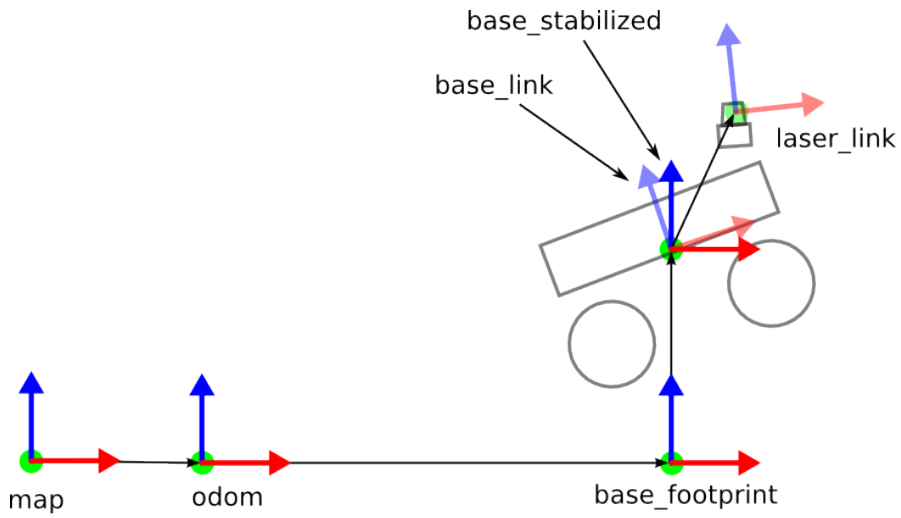


Figure 8.3: Coordinate frames for the RACECAR robot.

see Figure 8.3 and here[4] for further information. The distance to the goal and angle is then calculated and a proportional controller is used to reach is goal (essentially make the distance to goal $d = 0$). If the angle to the goal is less than a set critical angle, the mode is operation is termed as "normal" and the car proceeds in forward direction. Else, the car goes into a "uturn" mode described in the following section. Note that the uturn mode is essential due to the non-holonomic nature of the car. For the normal mode of the operation, the desired speed is set proportional to the distance d to the goal. Similarly, the desired steering angle δ is set proportional to the angle to the goal θ. A set of "filtering coefficients" α_v, α_δ etc are used to avoid abrupt fluctuations in the key quantities. The throttle input u is set proportional to the difference between desired and the filtered speed. If the magnitude of angle to the goal is greater than the critical θ_c, the car goes into the uturn mode. If this change is made from the normal mode, the distance to the closest obstacle and the current position is recorded. The uturn mode ensures that the car stays within a circle of radius r_o with center at $\mathbf{x}_c$. The direction of the steering wheel and throttle is initialized as in line 20. if the car starts to escape this ball of radius r_o, the direction of steering and throttle are flipped. The radius r_o is proportional to the distance to the closest obstacle. This prevents the

[4]http://wiki.ros.org/hector_slam/Tutorials/SettingUpForYourRobot

car from colliding with obstacles while in the uturn mode.

Algorithm 16: Waypoint Tracker Algorithm Flow

Input: Goal $\mathbf{x}_g$;
while *True* **do**
  $\mathbf{x}_{go} \leftarrow$ GetGoalInBaseFrame($\mathbf{x}_g$);
  $d \leftarrow \|\mathbf{x}_{go}\|, \theta \leftarrow \arctan(\mathbf{x}_{go})$;
  $d_f \leftarrow (\alpha_d)d + (1-\alpha_d)d_f$;
  $v \leftarrow (d_f - d_p)/\Delta T$;
  $v_f \leftarrow (\alpha_v)v + (1-\alpha_v)v_f$;
  $d_p \leftarrow d_f$;
  if $|\theta| < \theta_c$ **then**
    MODE $\leftarrow$ NORMAL;
    $v_d \leftarrow k_d d$;
    $u \leftarrow k_s(v_d - v_f)$;
    $u_f \leftarrow (\alpha_u)u + (1-\alpha_u)u_f$;
    $\delta \leftarrow k_\delta \theta$;
    $\delta_f \leftarrow (\alpha_\delta)\delta + (1-\alpha_\delta)\delta_f$
  else
    if *MODE=NORMAL* **then**
      $d_o \leftarrow$ DistToClosestObstacle();
      $r_o \leftarrow k_o d_o$;
      $e_\delta \leftarrow \theta/|\theta|, e_u \leftarrow 1$;
      $\mathbf{x}_c \leftarrow$ GetCurrentPosition();
    MODE $\leftarrow$ UTURN;
    $u \leftarrow k_u e_u$;
    $u_f \leftarrow (\alpha_u)u + (1-\alpha_u)u_f$;
    $\delta \leftarrow k_\delta e_\delta$;
    $\delta_f \leftarrow (\alpha_\delta)\delta + (1-\alpha_\delta)\delta_f$;
    $\mathbf{x} \leftarrow$ GetCurrentPosition() ;
    if $\|\boldsymbol{x} - \boldsymbol{x}_c\| \geq r_o$ **then**
      $e_\delta \leftarrow (-1)e_\delta, e_u \leftarrow (-1)e_u$;

8.3 Experiments and Discussion

The autonomous navigation stack described above was tested in ROS simulation as well as on the real hardware. The ROS Gazebo[5] simulator provides a detailed physics-engine and

[5]http://gazebosim.org/

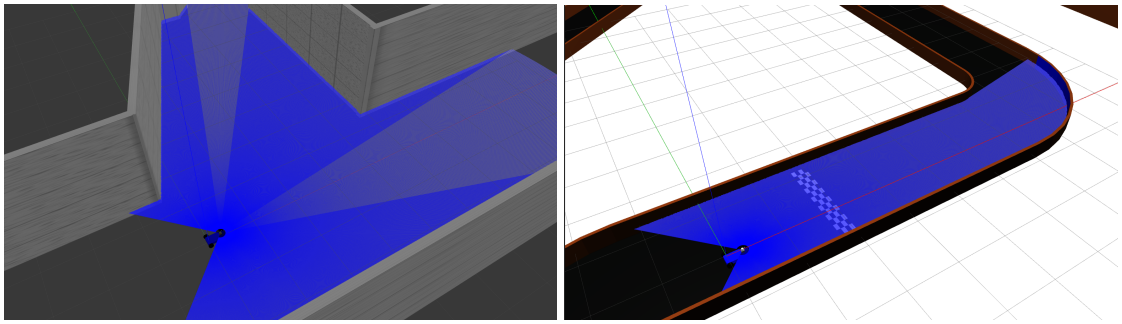


Figure 8.4: Different simulation environments for the RACECAR robot in ROS Gazebo.

good quality graphics. Different environments can be created in Gazebo for simulating the navigation software stack. Please see Figure 8.4 for an illustration. The RACECAR robot can be spawned in Gazebo using the following commands.

```
roscore
roslaunch racear_gazebo some_world_name.launch
```

The obstacle avoidance module can then be launched with,

```
rosrun racecar_control laser_navigate.py
```

A short video of the robot using the obstacle avoidance module can be found here[6].

The ROS RViz GUI can be used to visualize data from different sensors, display robot's pose and planned path, monitor the explored parts of map and send goal commands to the robot. Please see Figure 8.5. The point-to-point navigation stack can be launched in simulation and visualized in RViz with the following commands

```
roscore
roslaunch racear_control planner_simu.launch
```

Different parameters, such as the Gazebo world name, constants used by the SLAM, planning and path-tracking nodes, can be altered in the ROS launch file. A short video of the autonomous stack in action using the Relevant Region planner, proposed in Chapter 4, can be found here[7].

[6] https://www.youtube.com/watch?v=cD3Ww1h0AkE
[7] https://www.youtube.com/watch?v=u5Zc4rvt2UE

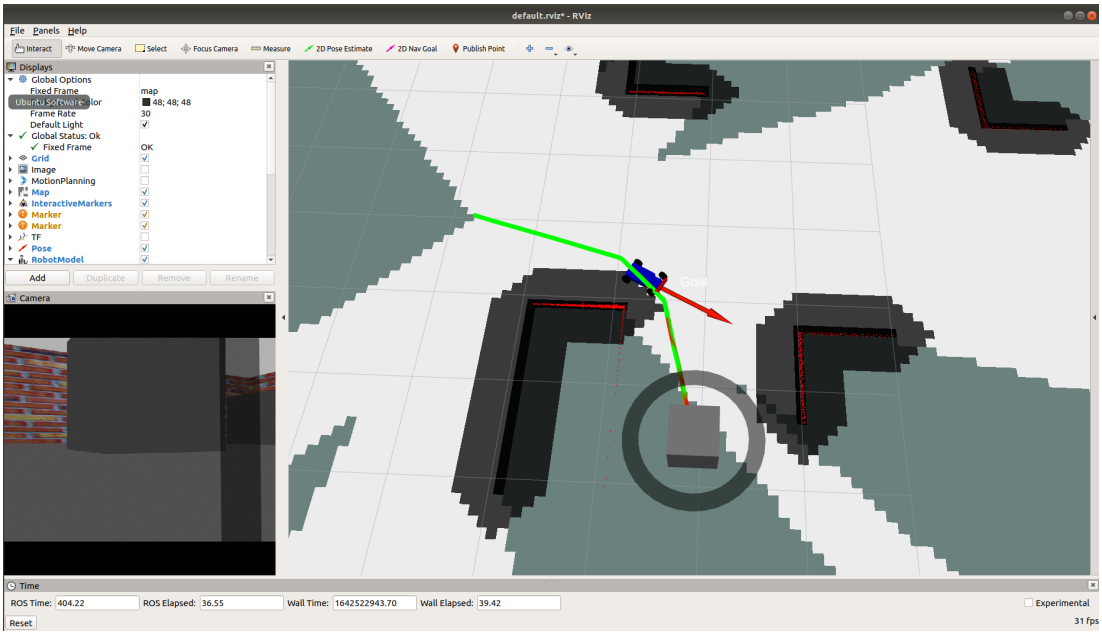


Figure 8.5: Visualizing different ROS topics in RViz. The red arrow illustrates the robot pose obtained from the Hector-SLAM. White and black areas denote the free and obstacle space respectively, whereas lighter black represents inflated areas around the obstacles. The ringed-cube is the goal marker, that can be manipulated by the user to set a goal state on the map. The path calculated by the Relevant Region planner is given in green. Other ROS topics, such as the feed from ZED camera can also be visualized.

To launch the autonomous stack on actual RACECAR hardware, communication first needs to be established between the robot and a remote laptop. The procedure for this is given here[8]. After a connection is established, SSH can be used to run commands on the robot hardware via a remote laptop. In simulation only mode, the RACECAR package launches "fake" motor-controller and sensor ROS nodes. To start the VESC and sensors on the actual RACECAR hardware, following commands can be executed.

```
roscore
roslaunch racecar bringup_teleop.launch
```

This launches the VESC and sensors, starts Hector-SLAM and puts the robot in a "teleop" mode. The RACECAR can now be moved around using a gaming controller. This teleop mode can be used to generate a map of the room and save it. The map can then be loaded at

[8]https://github.gatech.edu/DCSL/racecar_control/blob/master/docs/RACECAR_Launch_Instructions.pdf

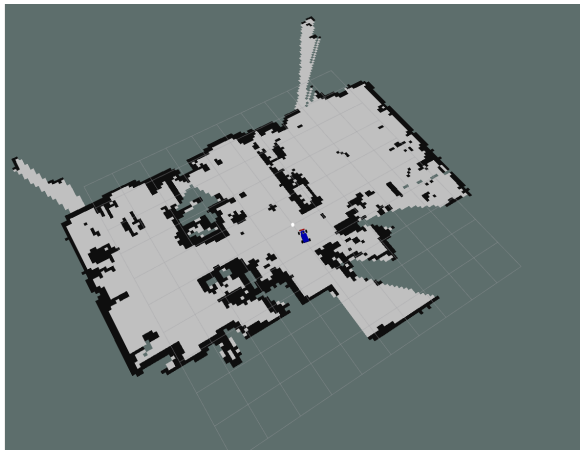


Figure 8.6: A map of the ESM G13 room in the AE department, Georgia Tech, generated using the Hector-SLAM algorithm on the RACECAR robot.

a later stage if needed and used in the planning stage. Please find a map of a AE department room generated by the robot's SLAM module in Figure 8.6.

The obstacle avoidance module can be initiated on the hardware with,

```
rosrun racecar_control laser_navigate_racecar.py
```

Please see Figure 8.7 and a short video here[9]. As seen in this experiment, the robot tries to avoid the wall and boxes using the LiDAR data. However, in some cases, the Jetson computer may not process the information fast enough, resulting in the RACEAR robot getting too close to an obstacle. This problem can be alleviated if the speed of the robot is low and exacerbated if it is high. In the case where the robot comes too close to an obstacle, it backs up until it is a threshold distance away before starting the forward motion again.

If the user instead wants to launch the point-to-point navigation stack, it can be done with

```
roslaunch racear_control planner_racecar.launch
```

This starts the SLAM, planning and path-tracking nodes. RViz can be used to send goal commands to the robot and monitor the explored map. Please see Figure 8.8 and a short video here[10].

[9] https://www.youtube.com/watch?v=bnLuKOhOU_I
[10] https://www.youtube.com/watch?v=BO5B2XXrntc

Figure 8.7: Running the obstacle avoidance module on RACECAR hardware

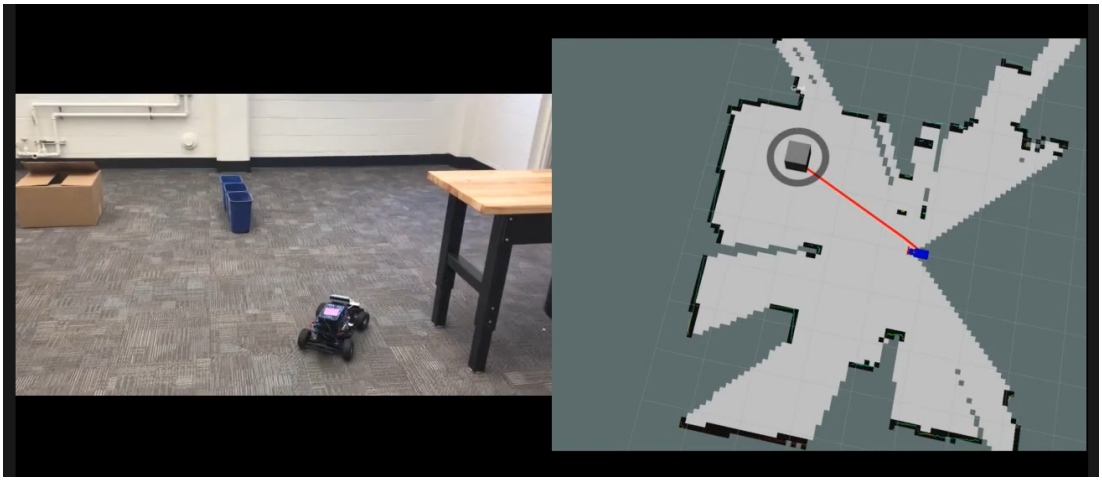


Figure 8.8: Running the point-to-point navigation stack on the RACECAR hardware (Left). Visualizing the robot's pose, planned path, goal state and the explored map in RViz (Right).

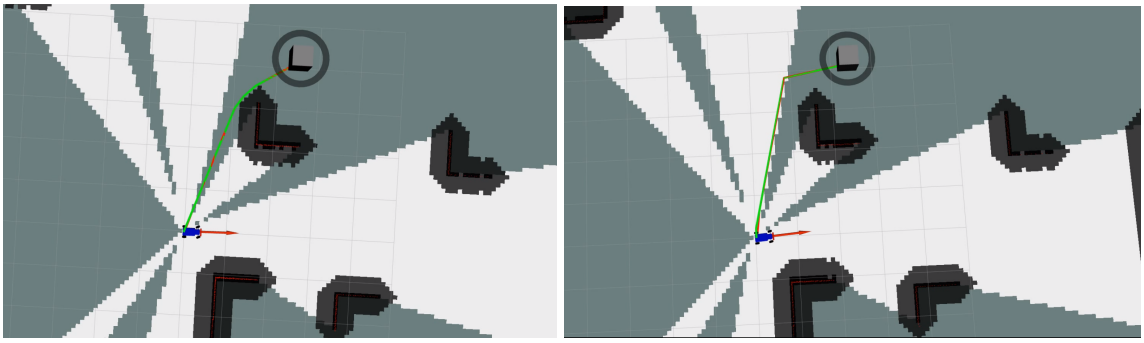


Figure 8.9: Length-optimal planning with the Relevant Region planner (left). Planning with the Relevant Region planner on the occupancy cost-map (right). In the latter case, the planner produces a path that stays more in the explored free space (white region), whereas the length-optimal path passes through the unexplored grey areas which may belong to the obstacle-space.

As seen in this experiment video, the robot plans paths to a series of goals and tracks them. As the robot moves through the world, the SLAM module updates the map. These map updates are sent to the planner module to verify the feasibility of the solution paths. Minor ROS communication errors or goal infeasibility can lead to the robot getting "stuck" at times. This can be solved in real time by just re-initializing the goal command. Similar to the obstacle avoidance controller case described above, increasing speed of the RACECAR can result in a computation lag and some overshoots in the robot motion. A major processing bottleneck for this navigation stack is the storage and update of the SLAM map. One possible solution to address this bottleneck is to lower the map resolution. This can free up more computational resources for the planning and control modules. The RACECAR robot can potentially be run at higher speeds then. However, lowering the map resolution too much can result in a coarse map and poor planning, especially in cluttered environments.

The Relevant Region algorithm proposed in this thesis can be used in conjunction with RACECAR's occupancy cost-map for "cautious" navigation. The SLAM module gives a probability of occupancy value for each cell in the grid, which can be used to define a cost-map. Unexplored cells with unknown occupancy are given a value of 0.5, whereas cells in the obstacle and free space are given values of 1.0 and 0.0 respectively. The Relevant

Region planner can take this information to plan paths that try to stay in the explored part of the free-space. Planning paths with such a cost-map can help the robot avoid collisions due to surprise obstacles that may spring up during map update. Please see Figure 8.9 for an illustration of this case. Please find a full video of the length optimal planning here[11] and the cost-map based planning here[12]

[11] https://www.youtube.com/watch?v=YO2Duj2cCaE
[12] https://www.youtube.com/watch?v=ULqDrw3hyKs

CHAPTER 9
CONCLUSION AND FUTURE DIRECTIONS

Randomized algorithms have become a prime choice for solving complex, higher dimensional robot motion planning problems due to their scalability and ease in handling the constraints. The sampling strategy used by these planners plays a critical role in dictating their performance. An oracular sampling strategy will generate points along the optimal path for a given query, leading to immediate convergence. However, realistically, sampling strategies need to balance two fundamental behaviors, namely, exploration and exploitation. This is because the planner needs to search the space to find the optimal homotopy class (explore) and then focus on it to improve the quality of the solution returned (exploit). The basic uniform random sampling can be classified as a pure exploration strategy. The implicit Voronoi bias its creates is essential for the fast exploration and to find an initial solution quickly. However, naively searching the entire space can adversely affect the quality of solution returned by the planner, as the sampling strategy invests very less computational effort on exploitation. This thesis endeavors to addresses the exploration-exploitation problem by proposing a family of strategies that leverage different sources of information available during planning time. These include the heuristics, collision data, planner's tree structure information, gradient information and robot dynamics. While the proposed techniques address many limitations of the state-of-the-art algorithms in the literature, a brief discussion on the scope for future work and some concluding remarks are given below.

Informed Sampling provides a scalable approach to focus search for the case of length-optimal planning, by direct sampling the interior of a prolate hyperspheroid (the L_2 Informed Set). However, it has to rely on uniform rejection sampling until an initial solution is found. Also, Informed Sampling can generate a large fraction of samples in the obstacle

space if the ratio $\lambda(\mathcal{X}_{\text{obs}} \cap \mathcal{X}_{\text{inf}})/\lambda(\mathcal{X}_{\text{inf}})$ is high. These issues are addressed in this thesis by proposing a non-parametric Informed Sampling (NP-Informed Sampling) technique. It leverages heuristics and planner collision checking data to prioritize search and avoid generating samples in the obstacle space. One of the drawbacks of the NP-Informed sampling is the computational complexity associated with choosing and generating samples from a kernel vertex. This results in a lesser number of vertices produced compared to Informed Sampling. However, NP-Informed sampling can compensate by finding first a solution of better quality to attain a better convergence profile. The proposed approach presents many promising directions for future research. An efficient balance between exploration using Informed Sampling and exploitation using NP-Informed sampling can be attained. An example of this is illustrated in Figure 3.4, which shows NP-Informed sampling with two different sets of weights, signifying different behavior after an initial solution is found. The black plot with bias for exploitation shows better convergence than the pure exploratory NP-Informed sampling. This can be extended to have a reward system for kernel vertices that produce "good" samples. A recent work by Mandalika et al [95] implements ideas along these lines

This thesis proposes a novel algorithm to sample the Relevant Region, a subset of the Informed Set, for SBMP. While the Informed Set has a 100% recall (it includes all the points that can improve the current solution), the Relevant Region can have a higher precision. Using a combination of Informed and Relevant Region sampling thus leads to a exploration-exploitation balance. Note that Informed Sampling uses a purely heuristic estimate of the solution cost, which may not be effective in general cost-space environments. The Relevant Region set on the other hand, considers the topology of $\mathcal{X}_{\text{free}}$, reduces the dependence on heuristics, and effectively focuses the search to accelerate convergence. The Relevant Region framework presents many avenues for future work. A simulated annealing-like procedure can be implemented to balance Relevant Region and Informed Sampling to eventually focus the search to the Relevant Region. Data from past iterations

can also be used to infer the nature of cost-map for intelligent exploration.

Extending the Relevant Region framework, this thesis proposes a "Locally Exploitative Sampling" algorithm, that generates new samples to improve the cost-to-come value of vertices in a neighborhood. LES numerically calculates the gradient of an objective function and decides an appropriate step-size to obtain a new sample. Although computationally costlier than other methods, LES adds an "exploitative-bias" that can accelerate convergence of SBMP algorithms, especially in higher dimensions. LES generates new samples in the Relevant Region to avoid redundant exploration after an initial solution is discovered. As discussed earlier, Informed Sampling is a necessary condition to improve the current solution. However, it is not sufficient, as an "Informed sample" is not guaranteed to bring about improvements in the current solution or the cost-to-come value of vertices. LES can be seen as a way to address this limitation of Informed Sampling. For future work, LES can be extended to kino-dynamic settings and be used with planners such SST [31]. Ideas from [39] can also be used to have an "obstacle-aware" version of LES.

This thesis uses ideas from reachability analysis to define a "Time-Informed Set" (TIS), to focus exploration for time-optimal kino-dynamic planning after an initial solution is found. We prove that exploring the TIS is a necessary condition to improve the current solution. The proposed method can be applied to a variety of systems for which an efficient local steering module may not be available, but (over-)approximations of the reachable sets can be constructed. It should be noted that the L_2-Informed set is *sharp* [12], i.e., it uses a heuristic estimate which gives the exact cost-to-come and cost-to-go for any point in the absence of obstacles. The TIS is not so, as it is constructed using *over-approximations* of the reachable sets. Hence, finding tight approximations of the reachable sets is critical for the efficacy of the proposed approach. In order to apply TIE for sampling-based planning, the reachability library needs to constructed offline. Creating, storing and accessing this library should be computationally efficient for higher dimensional systems to be of use in practice. The ellipsoidal reachable sets used in this work satisfy these criteria. The

HJB reachability toolboxes [71] can be potentially used to create this library for a general non-linear systems. These frameworks solve the value function PDE by discretizing the state space. However, the computational cost of these methods scale exponentially with the dimension. In order to address this curse of dimensionality, application of deep-learning frameworks for reachability, such as [96], [97], can be explored. Recent works such as DeepReach [97] avoid gridding the state space and use deep neural networks (DNN) to learn a parameterized approximation of the value function. These DNNs can be stored and used to classify or generate new samples in the TIS.

The prospect of leveraging ideas from deep learning for motion planning is promising. Many of the previous techniques in the literature have focused on exploring different deep architectures for planning, while using a uniformly sampled dataset for training. This thesis on the other hand, investigates the problem of improving the data-generation process while holding the model architecture and planning algorithm constant. The proposed query sampling and data pruning procedure adds more complicated paths in the dataset. Numerical experiments show that the success rate of the neural planner can be boosted using the deep models trained on such non-trivial datasets. This work presents many openings for future research. An ensemble model can be constructed by combining predictions from different models trained on datasets with varying degrees of non-triviality. This can potentially further increase the success rate of neural planner. Instead of a Boolean pruneData flag, calling the pruning procedure with a probability of γ_{nt} can be explored. This can prevent excessive pruning and drastic reduction in the size of the dataset for relatively less cluttered environments.

Finally, this thesis developed a autonomous navigation software framework to deploy some of the proposed planning algorithms on the RACECAR robot. The software stack consists of a LiDAR based SLAM module that outputs a map and robot pose, a planning module that generates feasible paths given a user defined goal, and a path tracking controller that outputs a steering and velocity command to be sent to the robot's motors. This

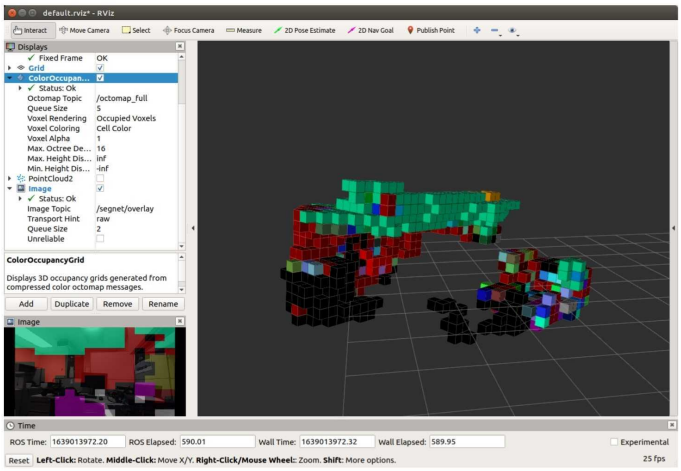


Figure 9.1: Semantic octomap generated for a room using the ZED camera on the RACECAR platform.

thesis tested the developed framework in different ROS based simulation environments and also on the actual hardware. The capabilities of this platform can be extended in several ways. Currently, the software stack only uses the LiDAR to perceive the environment. However, the ZED cameras can also be used to run the ORB-SLAM algorithm [98]. Off the shelf deep learning packages can be utilized to create a semantic octomap of the environment [99]. Please see Figure 9.1. Information extracted from this octomap can then be used for contextual planning. While the current planning library includes the standard OMPL-based randomized planners and A*, this collection can be expanded to incorporate several niche planners such as COA* [100] and L-GLS [101]. Lastly, the current software stack performs a point-to-point tracking with a simple, proportional controller. More sophisticated techniques such as differential dynamic programming or MPPI [102] can be implemented to devise a better controller module.

The algorithms proposed in thesis are most suited for solving robotic planning problems in higher dimensional spaces with a good amount of clutter. As shown in Chapter 5, the computational cost of the proposed methods such as LES is only justified in higher dimensions. For 2D environments, discrete-space planners such as A* may outperform sampling-based methods, even after incorporating the intelligent exploration techniques

proposed in this thesis. For relatively less cluttered environments, optimization based "local" methods such as CHOMP [66] and TrajOpt [103] and sampling-based methods such as DRRT can be very effective. This is because, if a candidate solution path is initialized in the right homotopy class, these methods can rapidly improve on it using the gradient updates. For many single-obstacle environments, DRRT was found to outperform many existing sampling-based planners. However, the performance of these methods can deteriorate in many-obstacle or cluttered environments, where a good degree of exploration is required to find the globally optimal solution. In such cases, the cost function may have local minima that can cause the optimization-based methods to get "stuck". While LES presents a good balance between pure exploration and local optimizer type methods, future research can further investigate a framework to modulate and balance these behaviors. As an example, a simulated annealing type of approach can be used to start planning with pure exploration and then incrementally increase the local-optimizer type behavior.

Dynamic environments with moving obstacles can add an additional layer of complexity while planning. RRT^x [104], a sampling-based planner, performs graph repairing operation through rewiring to account for moving and unpredictable environments. A potential direction of future work can be to investigate the integration of the proposed exploration techniques into the planners such as RRT^x, for efficient planning in dynamic settings.

The primary focus of this thesis was on developing intelligent exploration algorithms to speed up sampling-based planners in deterministic settings. However, considering different sources of uncertainties while planning might be necessary for a safe and reliable operation of the robots during run time. Various sources of uncertainty include inaccuracies in the motion model, actuation or sensor noise, uncertain or dynamic obstacles etc. In order to address this "belief-space planning" problem, several techniques such as [105], [106], [107], [108], [109] have been proposed. While these techniques represent significant advances in this area, ideas from this thesis can be utilized to further improve their performance through intelligent exploration. One approach to achieve this might be to incorporate an

additional “uncertainty-cost” (as done in [105]) into the original path or edge-cost. Then, an analogue of Relevant Region or LES can be proposed to focus search. The ellipsoidal toolbox also allows construction of reachable sets in presence of ellipsoidal disturbances. This functionality can be used to extend the TIE framework for planning in presence of disturbances.

REFERENCES

[1] S. Thomas, G. Song, and N. M. Amato, "Protein folding by motion planning", *Physical Biology*, vol. 2, no. 4, S148, 2005.

[2] Y. Liu and N. I. Badler, "Real-time reach planning for animated characters using hardware acceleration", in *Proceedings 11th IEEE international workshop on program comprehension*, IEEE, 2003, pp. 86–93.

[3] P. W. Finn and L. E. Kavraki, "Computational approaches to drug design", *Algorithmica*, vol. 25, no. 2, pp. 347–371, 1999.

[4] J. H. Reif, "Complexity of the mover's problem and generalizations", in *20th Annual Symposium on Foundations of Computer Science (SFCS)*, IEEE Computer Society, 1979, pp. 421–427.

[5] J. T. Schwartz and M. Sharir, "On the "piano movers'" problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers", *Communications on pure and applied mathematics*, vol. 36, no. 3, pp. 345–398, 1983.

[6] T. Lozano-Perez, "Spatial planning: A configuration space approach", in *Autonomous robot vehicles*, Springer, 1990, pp. 259–271.

[7] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs", *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[9] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality", *Advances in neural information processing systems*, vol. 16, pp. 767–774, 2003.

[10] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*", *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[11] A. Stentz, "Optimal and efficient path planning for partially known environments", in *Intelligent unmanned ground vehicles*, Springer, 1997, pp. 203–220.

[12] 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, Aug. 2018.

[13] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps", *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, Aug. 2010.

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

[15] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning", *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[16] J. Barraquand, L. Kavraki, J.-C. Latombe, R. Motwani, T.-Y. Li, and P. Raghavan, "A random sampling scheme for path planning", *The International Journal of Robotics Research*, vol. 16, no. 6, pp. 759–774, 1997.

[17] 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, Jun. 2011.

[18] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning", in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 2421–2428.

[19] ——, "Incremental sampling-based motion planners using policy iteration methods", in *IEEE 55th Conference on Decision and Control*, Las Vegas, NV, Dec. 2016, pp. 5004–5009.

[20] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs", in *IEEE International Conference on Robotics and Automation*, Seattle, WA, May 2015, pp. 3067–3074.

[21] 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, May 2015.

[22] M. P. Strub and J. D. Gammell, "Advanced BIT*(ABIT*): Sampling-based planning with advanced graph-search techniques", in *International Conference on Robotics and Automation (ICRA)*, IEEE, Paris, France, 2020, pp. 130–136.

[23] ——, "Adaptively informed trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics", in *International Conference on Robotics and Automation (ICRA)*, IEEE, Paris, France, 2020, pp. 3191–3198.

[24] F. Hauer and P. Tsiotras, "Deformable rapidly-exploring random trees.", in *Robotics: Science and Systems*, Cambridge, MA, Jul. 2017.

[25] T. Lai, F. Ramos, and G. Francis, "Balancing global exploration and local-connectivity exploitation with rapidly-exploring random disjointed-trees", in *International Conference on Robotics and Automation (ICRA)*, IEEE, Montreal, Canada, 2019, pp. 5537–5543.

[26] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods", in *49th IEEE Conference on Decision and Control*, Atlanta, GA, Dec. 2010, pp. 7681–7687.

[27] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics", in *IEEE International Conference on Robotics and Automation*, Saint Paul, MN, 2012, pp. 2537–2542.

[28] D. J. Webb and J. Van Den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics", in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 2013, pp. 5054–5061.

[29] E. Schmerling, L. Janson, and M. Pavone, "Optimal sampling-based motion planning under differential constraints: The drift case with linear affine dynamics", in *IEEE Conference on Decision and Control (CDC)*, Osaka, Japan, Dec. 2015, pp. 2574–2581.

[30] J. hwan Jeon, S. Karaman, and E. Frazzoli, "Optimal sampling-based feedback motion trees among obstacles for controllable linear systems with linear constraints", in *IEEE International Conference on Robotics and Automation*, Seattle, Washington, May 2015, pp. 4195–4201.

[31] Y. Li, Z. Littlefield, and K. E. Bekris, "Sparse methods for efficient asymptotically optimal kinodynamic planning", in *Algorithmic Foundations of Robotics XI*, Springer, 2015, pp. 263–282.

[32] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, Sep. 2011, pp. 2640–2645.

[33] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth", in *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, vol. 2, Las Vegas, NV, Oct. 2003, pp. 1178–1183.

[34] R. Diankov and J. Kuffner, "Randomized statistical path planning", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, Oct. 2007, pp. 1–6.

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

[36] ——, "Single-query entropy-guided path planning", in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 2124–2129.

[37] M. Rickert, O. Brock, and A. Knoll, "Balancing exploration and exploitation in motion planning", in *IEEE International Conference on Robotics and Automation*, Pasadena, CA, May 2008, pp. 2812–2817.

[38] R. Alterovitz, S. Patil, and A. Derbakova, "Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning", in *IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3706–3712.

[39] T. Lai, P. Morere, F. Ramos, and G. Francis, "Bayesian local sampling-based planning", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1954–1961, 2020.

[40] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, "An obstacle-based rapidly-exploring random tree", in *IEEE International Conference on Robotics and Automation*, Orlando, FL, May 2006, pp. 895–900.

[41] J. Denny, E. Greco, S. Thomas, and N. M. Amato, "MARRT: Medial axis biased rapidly-exploring random trees", in *IEEE International Conference on Robotics and Automation*, Hong Kong, China, Jun. 2014, pp. 90–97.

[42] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces", in *IEEE International Conference on Robotics and Automation*, vol. 3, Albuquerque, NM, Apr. 1997, pp. 2719–2726.

[43] J. M. Phillips, N. Bedrossian, and L. E. Kavraki, "Guided expansive spaces trees: A search strategy for motion-and cost-constrained state spaces", in *IEEE International Conference on Robotics and Automation*, New Orleans, LA, Apr. 2004, pp. 3968–3973.

[44] S. M. Persson and I. Sharf, "Sampling-based A* algorithm for robot path-planning", *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, Oct. 2014.

[45] Z. Littlefield and K. E. Bekris, "Informed asymptotically near-optimal planning for field robots with dynamics", in *Field and Service Robotics*, Springer, 2018, pp. 449–463.

[46] ——, "Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions", in *IEEE International Conference on Intelligent Robots and Systems*, Madrid, Spain, Oct. 2018, pp. 1–9.

[47] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints", in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 2859–2865.

[48] S. D. Pendleton *et al.*, "Numerical approach to reachability-guided sampling-based motion planning under differential constraints", *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1232–1239, Jul. 2017.

[49] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "RL-RRT: Kinodynamic motion planning via learning reachability estimators from rl policies", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4298–4305, 2019.

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

[51] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 2018, pp. 7063–7070.

[52] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees", in *IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 2007, pp. 3307–3312.

[53] M. Muja and D. G. Lowe, "FLANN, fast library for approximate nearest neighbors", in *International Conference on Computer Vision Theory and Applications*, vol. 3, Lisbon, Portugal, Feb. 2009.

[54] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library", *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, Dec. 2012.

[55] O. Arslan and P. Tsiotras, "Dynamic programming guided exploration for sampling-based motion planning algorithms", in *IEEE International Conference on Robotics and Automation*, Seattle, WA, May 2015, pp. 4819–4826.

[56] J. Mainprice, E. A. Sisbot, L. Jaillet, J. Cortés, R. Alami, and T. Siméon, "Planning human-aware motions using a sampling-based costmap planner", in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 5012–5017.

[57] L. Jaillet, F. J. Corcho, J.-J. Pérez, and J. Cortés, "Randomized tree construction algorithm to explore energy landscapes", *Journal of Computational Chemistry*, vol. 32, no. 16, pp. 3464–3474, 2011.

[58] D. Devaurs, T. Siméon, and J. Cortés, "Enhancing the transition-based RRT to deal with complex cost spaces", in *IEEE International Conference on Robotics and Automation*, Karlsrühe, Germany, May 2013, pp. 4120–4125.

[59] D. Berenson, T. Siméon, and S. S. Srinivasa, "Addressing cost-space chasms in manipulation planning", in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 4561–4568.

[60] D. Devaurs, T. Siméon, and J. Cortés, "Optimal path planning in complex cost spaces with sampling-based algorithms", *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 415–424, 2015.

[61] O. Arslan and P. Tsiotras, "Machine learning guided exploration for sampling-based motion planning algorithms", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, Sep. 2015, pp. 2646–2652.

[62] M. Moll, I. A. Sucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization", *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 96–102, 2015.

[63] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ROS topics]", *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[64] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning", in *IEEE International Conference on Robotics and Automation.*, vol. 2, San Franciso, CA, Apr. 2000, pp. 995–1001.

[65] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, "Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning", in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 4207–4214.

[66] M. Zucker *et al.*, "CHOMP: Covariant Hamiltonian optimization for motion planning", *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[67] D. Kim, M. Kang, and S.-E. Yoon, "Volumetric tree*: Adaptive sparse graph for effective exploration of homotopy classes", in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, 2019, pp. 1496–1503.

[68] J. Watt, R. Borhani, and A. K. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, and Applications*, 1st. USA: Cambridge University Press, 2016, ISBN: 1107123526.

[69] S. S. Joshi and P. Tsiotras, "Relevant region exploration on general cost-maps for sampling-based motion planning", in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, Las Vegas, NV, Oct. 2020.

[70] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[71] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances", in *IEEE 56th Annual Conference on Decision and Control (CDC)*, Brisbane, Australia, Dec. 2017, pp. 2242–2253.

[72] A. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis", in *International Workshop on Hybrid Systems: Computation and Control*, Springer, 2000, pp. 202–214.

[73] A. Girard, C. Le Guernic, and O. Maler, "Efficient computation of reachable sets of linear time-invariant systems with inputs", in *International Workshop on Hybrid Systems: Computation and Control*, Springer, 2006, pp. 257–271.

[74] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed", *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[75] B. Chen, B. Dai, Q. Lin, G. Ye, H. Liu, and L. Song, "Learning to plan in high dimensions via neural exploration-exploitation trees", in *International Conference on Learning Representations*, 2020.

[76] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners", *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.

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

[78] M. Zucker, J. Kuffner, and A. J. Bagnell, "Adaptive workspace biasing for sampling-based planners", in *IEEE International Conference on Robotics and Automation*, Pasadena, CA, May 2008, pp. 3757–3762.

[79] M. Bhardwaj, S. Choudhury, and S. Scherer, "Learning heuristic search via imitation", in *Conference on Robot Learning*, PMLR, 2017, pp. 271–280.

[80] S. Choudhury *et al.*, "Data-driven planning via imitation learning", *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1632–1672, 2018.

[81] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki, "Path planning using neural a* search", in *International Conference on Machine Learning*, PMLR, 2021, pp. 12 029–12 039.

[82] J. Huh, G. Xing, Z. Wang, V. Isler, and D. D. Lee, "Learning to generate cost-to-go functions for efficient motion planning", *arXiv preprint arXiv:2010.14597*, 2020.

[83] C. Chamzas, A. Shrivastava, and L. E. Kavraki, "Using local experiences for global motion planning", in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8606–8612.

[84] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning", in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, 2018, pp. 3654–3661.

[85] Y.-L. Kuo, A. Barbu, and B. Katz, "Deep sequential models for sampling-based planning", in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 6490–6497.

[86] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 9535–9541.

[87] R. Kumar, A. Mandalika, S. Choudhury, and S. Srinivasa, "Lego: Leveraging experience in roadmap generation for sampling-based planning", *International Conference on Intelligent Robots and Systems (IROS)*, pp. 1488–1495, 2019.

[88] J. Huh, D. D. Lee, and V. Isler, "Learning continuous cost-to-go functions for nonholonomic systems", *arXiv preprint arXiv:2103.11168*, 2021.

[89] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3965–3972.

[90] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning", in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[91] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning", in *Advances in Neural Information Processing Systems*, 2017, pp. 6467–6476.

[92] J. J. Johnson, L. Li, F. Liu, A. H. Qureshi, and M. C. Yip, "Dynamically constrained motion planning networks for non-holonomic robots", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 6937–6943.

[93] S. Karaman *et al.*, "Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at mit", in *2017 IEEE Integrated STEM Education Conference (ISEC)*, 2017, pp. 195–203.

[94] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable SLAM system with full 3d motion estimation", in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, IEEE, Nov. 2011.

[95] A. Mandalika, R. Scalise, B. Hou, S. Choudhury, and S. S. Srinivasa, "Guided incremental local densification for accelerated sampling-based motion planning", *arXiv preprint arXiv:2104.05037*, 2021.

[96] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone, "A machine learning approach for real-time reachability analysis", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, Sep. 2014, pp. 2202–2208.

[97] S. Bansal and C. Tomlin, "DeepReach: A Deep Learning Approach to High-Dimensional Reachability", *arXiv e-prints: 2011.02082*, Nov. 2020. arXiv: 2011.02082 [cs.RO].

[98] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular slam system", *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[99] A. Asgharivaskasi and N. Atanasov, "Active bayesian multi-class mapping from range and semantic segmentation observations", in *International Conference on Robotics and Automation (ICRA)*, IEEE, Xian, China, 2021, pp. 1–7.

[100] J. Lim and P. Tsiotras, "A generalized a* algorithm for finding globally optimal paths in weighted colored graphs", in *International Conference on Robotics and Automation (ICRA)*, IEEE, Xian, China, 2021, pp. 7503–7509.

[101] J. Lim, S. Srinivasa, and P. Tsiotras, "Lazy lifelong planning for efficient replanning in graphs with expensive edge evaluation", *arXiv preprint arXiv:2105.12076*, 2021.

[102] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control", in *International Conference on Robotics and Automation (ICRA)*, IEEE, Stockholm, Sweden, 2016, pp. 1433–1440.

[103] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking", *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[104] M. Otte and E. Frazzoli, "Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning", *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[105] B. Luders, M. Kothari, and J. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty", in *AIAA guidance, navigation, and control conference*, 2010, p. 8160.

[106] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements", *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.

[107] C. K. Verginis, D. V. Dimarogonas, and L. E. Kavraki, "Sampling-based motion planning for uncertain high-dimensional systems via adaptive control", in *International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2020, pp. 159–175.

[108] D. Zheng, J. Ridderhof, P. Tsiotras, and A.-a. Agha-mohammadi, "Belief space planning: A covariance steering approach", *arXiv preprint arXiv:2105.11092*, 2021.

[109] D. Zheng and P. Tsiotras, "Batch belief trees for motion planning under uncertainty", *arXiv preprint arXiv:2110.00173*, 2021.

VITA

Sagar Suhas Joshi was born in the Pune, India. In 2017, he earned a bachelors degree in Engineering Design from the Indian Institute of Technology(IIT)-Madras, with a top rank in his batch. During his time at IIT-Madras, he worked on autonomous planning and control of unmanned ground vehicles. Sagar is currently a PhD candidate in robotics at the Institute for Robotics and Intelligent Machines (IRIM), Georgia Tech, USA. His research focuses on intelligent exploration for sampling-based motion planning algorithms. This area is at the exciting intersection of topics such as artificial intelligence, machine learning and optimal control. During his PhD, Sagar also did an internship at Aurora Innovation, Pittsburgh, where he worked on planner strategy ranking and scoring for safe autonomous driving.