Kinodynamic Non-Holonomic Motion Planning for UAVs: A Minimum Energy Approach

Nir Rikovitch

Department of Mechanical Engineering

McGill University

Montréal, Québec

August 2013

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Engineering

DEDICATION

To my family

Congratulations!

Today is your day.

You're off to Great Places!

You're off and away!

You have brains in your head.

You have feet in your shoes

You can steer yourself

any direction you choose.

You're on your own. And you know what you know.

And YOU are the guy who'll decide where to go.

You'll look up and down streets. Look 'em over with care.

About some you will say, "I don't choose to go there."

With your head full of brains and your shoes full of feet,

you're too smart to go down any not-so-good street.

...

OH!

THE PLACES YOU'LL GO!

-Dr. Seuss

ACKNOWLEDGEMENTS

I am enormously grateful to Professor Inna Sharf, my adviser, for her continuous support and guidance throughout this research. Her invaluable scientific advice, knowledge and mentorship have been extremely helpful to me in completing my work and enriching my knowledge. Through our weekly meetings Prof. Sharf has taught me that research is not only about knowing the answers, but about asking the right questions.

I would like to take this opportunity to thank my colleagues. First and foremost to Sven Mikael Persson for his support in general and his advice on probabilistic motion planning in particular. His insight and dedicated help integrating the code into his library has been great help. Second, Adam Harmat for the warm welcome into his world of research. His assistance and friendship through the hurdles of graduate life have been remarkable.

I would also like to thank my other half, best friend and love of my life, my wife Galit, without whose love, encouragement, support and bright outlook on life I would not have finished this thesis. She already has my heart, so I will merely extend a sincere heartfelt gratitude.

The important thing is to not stop questioning.

-Albert Einstein

ABSTRACT

This thesis presents a minimum energy optimal sampling-based motion planning algorithm, MEAQR RRT*, specifically for systems with non-linear dynamics, non-holonomic and actuation constraints. The motion planner developed is intended for unmanned aerial systems (UAS) whether they be fixed or rotary wing. The eventual implementation in this thesis revolves around a quadrotor platform. This work extends the algorithms presented previously [53, 20] by formulating a fixed-final-state-free-final-time open loop state space pseudometric for nearest neighbours search and the appropriate closed loop steering method for the tree extension heuristic. By doing so, it allows the introduction of constraints on actuation magnitude and bandwidth. The controller is formulated by solving a minimization problem for the amount of input energy used to connect two states along a trajectory segment. This thesis argues that this properly tuned pseudometric integrated into a state space exploring algorithm results in a trajectory with minimal energy expenditure, which is extremely beneficial for present-day battery limited capabilities of unmanned aerial systems. The algorithm is demonstrated on two example systems (1) a simple 2D pendulum with actuation constraints and (2) a quadrotor described by a 13-dimensional state-space model. In addition, an environment modelling procedure is designed, implemented and tested for *a-priori* map building required in order to test the algorithm on a real life environment.

ABRÉGÉ

Cette thèse présente un algorithme de planification de mouvement qui minimise l'utilisation d'énergie utilisant une approche d'échantillonnage. Cette algorithme est nommé MEAQR RRT*, il a été conçue pour les systèmes non-linéaire dynamiques, non-holonomique et avec contraintes d'actionnements. Ce planificateur de mouvement a été développé pour les systèmes aériens sans pilotes (UAS) soit avec ailes fixes ou rotationnelles. Ce mémoire pourrait éventuellement planifier des chemins pour une plateforme quadrotor. Ce travail est un extension des algorithmes présenté [53, 20] en formulant un état fixe final et libre dans un espace d'états ouverts afin de trouver le voisin le plus prés et la méthode appropriée pour fermé les boucles de conduite. Ceci permet d'introduire des contraintes d'actionnements sur la grandeur et l'intensité. Ce contrôleur résout le problème de minimiser l'énergie utilisé afin de connecter deux états selon une trajectoire. Il est discuté que ce pseudo-métrique intégré a l'exploration d'états résulte en une trajectoire qui minimise l'utilisation d'énergie. Ceci permet de réduire la consommation d'énergie sur les batteries aux capacités limités d'UAS. Il est démontré la puissance de notre système par l'entremise de deux exemples, un simple pendule avec des contraintes d'activation en deux dimensions et en modélisant un quadrotor avec un espace d'état a 13 dimensions. De plus, une procédure de conceptualisation a été conçu, implémenté et testé afin d'évaluer les besoins d'un plan afin de tester le modèle présenté dans un environnement réel.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

ABSTRACT iv

ABRÉGÉ v

LIST OF TABLES viii

LIST OF FIGURES ix

1 Introduction 1

1.1 Background 1
1.2 Problem Statement 5
1.3 Thesis Overview 6

2 Related Work 9

2.1 Existing Approaches 9
2.1.1 Combinatorial Motion Planning 10
2.1.2 Sampling-Based Motion Planning 10
2.2 Motion Planning Metrics 12
2.2.1 Optimal Control Based Metrics 13
2.3 Optimal Probabilistic Motion Planning 15

3 The Probabilistic Motion Planner - RRT 17

3.1 Rapidly Exploring Random Tree 17
3.2 Optimal Rapidly Exporing Random Tree - RRT* 19

4 Minimum Energy Affine Quadratic Regulator 23

4.1 Affine Quadratic Regulator: Preliminaries 23
4.2 Minimum Energy Pseudometric 26
4.2.1 Pseudometric Formulation 27
4.2.2 Computing the Optimal Time 29
4.3 Minimum Energy Closed-Loop Control 32
4.4 Minimum Energy Affine Quadratic Regulator in RRT* . . 34
4.5 Practical Implementation 39
4.5.1 Weighted Controllability Grammian Positive-Definiteness 40
4.5.2 Integrating Backwards in Time 41

4.5.3 Scaling the MEAQR Pseudometric 42
4.5.4 Actuation Limitation 43
4.5.5 Data Structures 46

5 Simulations 53

5.1 Torque Limited Pendulum 54
5.1.1 Dynamic Model 54
5.1.2 Monte Carlo Results and Discussion 56
5.1.3 Sample Trajectories 59
5.2 Draganflyer X8 62
5.2.1 Single Pillar Room 63
5.2.2 Single Window Room 64

6 Map Building 67

6.1 Laser Aggregator 69
6.2 Map Creation 70
6.3 Real World Examples 72

7 Conclusions and Future Directions 75

7.1 Summary 75
7.2 Improvements and Future Directions 77
7.2.1 Algorithmic Improvements 77
7.2.2 Future Work 78

A Draganfly X8 State Space Model 80

References 82

LIST OF TABLES

Table		page
5–1	Pendulum State Space and Actuation Bounds	55
5–2	Pendulum *Monte-Carlo* Simulation Results	58
5–3	*X8* State Space Bounds	63
5–4	*X8* Actuation Bounds	63
5–5	*X8* MEAQR Pseusdometric Parameters	63
A–1	*X8* Physical Parameters	81

LIST OF FIGURES

Figure		page
1–1	Motion Planning Problem Schematic	6
1–2	Simplified System Architecture	7
2–1	Probabilistic Motion Planner Architecture	11
3–1	RRT Extension Step	18
4–1	Exact MEAQR Cost Function	31
4–2	Approximate MEAQR Cost Function Value	31
4–3	State Space Pseudometric Contours	39
4–4	MEAQR RRT* Algorithm Flowchart	40
4–5	Steering with Actuation Constraints	44
4–6	Kd-tree Metric Space Decomposition	48
4–7	Ball Tree Data Structure	49
4–8	Vp-tree Metric Space Decomposition	50
4–9	Data Structure Performance Comparison for KNN	52
5–1	Inverted Pendulum Free Body Diagram	54
5–2	Pendulum *Monte-Carlo* Simulation Results	57
5–3	Pendulum Energy Optimal Trajectory	60
5–4	Pendulum Energy Non-Optimal Trajectory	61
5–5	The *Draganflyer X8*	62
5–6	*X8* Path in Artificial Environment A	65
5–7	*X8* Path in Artificial Environment B	66
6–1	Motion Planner and Map Building System Architecture	68
6–2	Hokuyo LIDAR	68
6–3	ROS LIDAR Assembler and Filter Nodes	70
6–4	Point Cloud Library Map Creating Process	71

6–5 McGill AML Map Building 73
6–6 McGill McDonald Eng. Building Roof Top Map Building . . . 74

Chapter 1
Introduction

1.1 Background

In recent years unmanned aerial vehicles (UAVs) have gained popularity for various applications, both civilian and military. As technology advanced, UAVs became cheaper, lighter and more readily available than their manned versions. They are utilized for missions where the added value of a human pilot is low while the risk or dullness of the tasks are incommensurately high. Much of the current research in the UAV community aims to modify the system structure to allow the human to over-see the ongoing mission instead of being an integral part of it. The interest is to move the human from the classical position of *in-the-loop* to a more supervisory position *on-the-loop*. This will allow human-robot teams to better perform by liberating the human from constant online mission operation to an upper hierarchy of decision making, which is to monitor the mission and to intervene when needed.

Though a developing market, most UAVs with a limited level of autonomy that are in use today are large fixed-wing commercial-altitude aircraft for military missions. The ability to operate autonomous, small scale ($< 5kg$ in weight), hover-craft UAVs in an urban environment is yet to be conquered. The solution of flying such vehicles above the building line using a straight line trajectory is appealing in its simplicity; however, maintaining such high altitude hinders the mission scope, nor does it exploit the high manoeuvrability of rotary aircraft. On the other hand, flying unmanned vehicles at street level requires that they be able to cope with the proximity to obstacles and the dynamic environment in a safe and efficient manner. Hence, developing

a feasible trajectory planner for an urban environment is key to achieving autonomous UAV flight.

In order to develop such autonomous capabilities, researchers at McGill Aerospace Mechatronics Laboratory, as part of a research contract with DRDC-Suffield, have retrofitted a commercially available octorotor UAV *Draganflyer X8*[1] (hereafter *X8*) with the appropriate sensor suit to allow state estimation, localization and mapping for the vehicle - eventually aimed at autonomous flight. Although the *X8* has eight rotors, it is in an $\times$ configuration with four coaxial pairs of rotors. It can be regarded as a quadrotor since each coaxial pair is treated as a single actuator at the higher level of the control architecture. The availability of a small scale platform, such as the *X8*, allows for short range operations within a cluttered environment below the building height line. At the same time, the aircraft size significantly decreases and constrains its load carrying capabilities. Thus, battery capacity is limited (in today's battery capabilities bounds) and eventually substantially reduces the vehicle's mission endurance and range. Hence, the preferred trajectory for the vehicle to follow is the one that will enable the UAV to operate for as long as possible. This has led to the work described in this thesis which considers minimization of energy as the desired objective along the trajectory generated for the aircraft. This thesis presents an energy optimal kinodynamic sampling based trajectory generator for actuator-limited non-linear systems, specifically the *X8*. The developed motion planner is integrated with *a-priori* environment modelling for a more complete solution.

The field of unmanned aerial systems (UAS) motion planning is a heavily researched area for the reasons alluded to earlier. The majority of existing motion planners were developed for holonomic systems and tend to be very

[1] Draganfly Innovation `http://www.draganfly.com/uav-helicopter/draganflyer-x8/index.php`

reliable and fast for such systems. However, the system at hand is one with non-holonomic constraints. As discussed in chapter 2, the *state-of-the-art probabilistic motion planners* can cope with such systems successfully even though the general motion planning problem is considered PSPACE-hard[1] . The aforementioned constraints arise in systems with fewer actuators than degrees of freedom. The non-holonomic constraints are non-integrable constraints linking the derivatives of the state variables to one another. A physical interpretation of one would be a forward moving quadrotor and its inability to instantaneously move transversely to the forward pointing body axis. In order for a quadrotor to achieve such a maneuver, a direction change is needed first, only later yielding the desired motion. This is due to the axes upon which the forces and torques of the *X8* act - a six degree of freedom system with merely four actuators.

The rapidly exploring random trees (RRT) family of algorithms was chosen as the development frame-work of the motion planner for the *X8* aerial vehicle due to RRTs ability to deal with the kinematic and actuation limitations and to quickly generate a solution. Of the available probabilistic motion planners, the RRT does not require the connecting method (see section 3.1) to be symmetric - hence, it does not require the distance measure between two states to be a true metric. With the ultimate goal of generating an optimal trajectory with respect to a specified cost functional or metric, this thesis uses the variant RRT* for its desirable attributes.

The RRT* is an asymptotically optimal variant of the classical RRT. It differs from the RRT in its ability to rewire the tree in an optimal manner (with respect to the cost functional defined on the state space) and to optimally

[1] A problem classified as PSPACE corresponds to one that can be solved by a Turing machine in polynomial space complexity .

choose the best parent for each newly generated state. This yields asymptotically optimal trajectories [28]. In applying the RRT* to a quadrotor such as the *X8*, two domain-specific components of the motion planning algorithm were addressed in this thesis: the distance metric and the extension heuristic. Early versions of the sampling based motion planners [34, Ch 5] utilized a Euclidean distance metric which failed to capture the state space (SS) manifold structure and resulted in poor SS coverage and slow rate of convergence. As an extension heuristic, random actuation also resulted in slow convergence to an optimal solution. The possibility of using optimal control theory to derive the approximate *cost-to-go* for use as a pseudometric and to define optimal-control-based extension heuristic/steering scheme was suggested by LaValle and Kuffner [35]. Glassman and Tedrake [20] derived a pseudometric based on an affine quadratic regulator for a regular RRT. Webb and Berg [53] utilized the same affine quadratic regulator to produce the means to connect two states in optimal fixed-final-state-free final-time for systems with differential constraints.

In yearning for autonomous flight, a complicated multidisciplinary task indeed, various components must be designed, implemented, integrated, verified and tested. In addition to motion planning, which is the focus of the present thesis, these components include: task assignment, guidance, control, environment modelling, communications, system health monitoring, fault detection and others. Each of these topics require substantial research efforts and are indeed present throughout the literature. With the view to validating and demonstrating the performance of the motion planner developed here, the task of environment modelling was addressed. Environment modelling was implemented with a reconnaissance flight prior to autonomous deployment. The mission flight zone is scanned with a laser to create a map to be used in the motion planing prior to mission execution.

1.2 Problem Statement

In order to properly define the problem, some notation needs an introduction. Let $\chi \subseteq \mathbb{R}^d$ be the *state space* of dimensionality d where the elements of χ are known as *states* denoted as $\boldsymbol{x}$. Let $\mathbb{U} \subseteq \mathbb{R}^{d_u}$ be the *control space* of dimensionality d_u. The space occupied with obstacles is χ_{obs} leaving the rest of the space free χ_{free}. A *path* $\sigma \in \chi$ is a continuous function $\sigma(t) = (\boldsymbol{x}, t)$ connecting two states. It is said to be free iff $\sigma \in \chi_{free}$ and feasible if in addition it satisfies the dynamic and actuation constraints.

The motion planning problem is stated as: Given an initial state $\boldsymbol{x}_{init}$ at t_0, an environment with obstacles χ_{obs} and a goal state $\boldsymbol{x}_{goal}$, the motion planning problem deals with finding a **feasible collision-free** path connecting the initial state to the goal state. The strict goal state condition can be loosened by having a tolerance about it represented by a goal region, denoted as χ_{goal}. Figure 1–1 is an illustration of the generic motion planning problem. The feasibility of the path σ is determined by not only residing in the free space but also governed by the non-linear SS dynamics described by eq. (1.1):

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{1.1}$$

The above is the usual formulation of the motion planning problem which is concerned only with finding a feasible path. However, for many applications such as the one this thesis is aimed at, an optimal path is preferred. If the set of paths is denoted as Σ then $c\colon \Sigma \to \mathbb{R}^+$ is a *cost functional* that maps a candidate free path to a non-negative scalar cost. The *optimal motion planning problem* deals with finding a path σ that does not only adhere to the aforementioned criteria but also minimizes the cost function: $c(\sigma^*) = \inf_{\hat{\sigma} \in \Sigma_{free}} c(\hat{\sigma})$. The optimal path is a once differentiable function C^1 although for actuation feasibility C^2 is required. The time sequence of control inputs $\boldsymbol{u}$ propagating eq. (1.1) forward in time is called the control policy denoted as $\boldsymbol{\pi}$. The

optimal control policy is the sequence yielding the minimal cost value with respect to the defined cost functional c:

$$\boldsymbol{\pi}^* = \underset{\pi}{\operatorname{argmin}}\ c(\boldsymbol{\pi}, \boldsymbol{x}_{init}, \chi_{goal}) \tag{1.2}$$

Finding the optimal control policy $\boldsymbol{\pi}^*$ for the system in eq. (1.1) essentially solves the *optimal motion planning problem*.

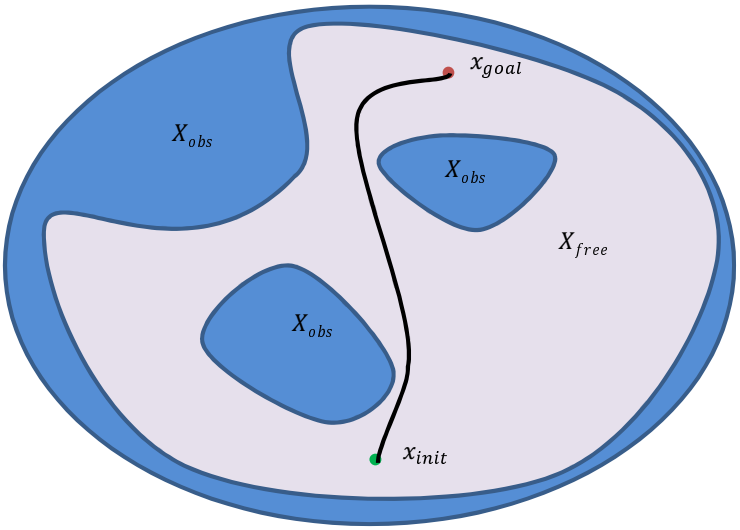


Figure 1–1: The motion planning problem is to find a feasible path $\sigma \in \chi_{free}$ from $\boldsymbol{x}_{init}$ to $\boldsymbol{x}_{goal}$

1.3 Thesis Overview

This work presents an RRT* based motion planner incorporating a minimum energy affine quadratic regulator (MEAQR) as a metric in addition to environment modelling, as depicted in fig. 1–2 for the *X8* vehicle. The map building solution implemented in this work for integration with the proposed motion planner is based on a laser-based sensor scanning the environment, aggregating and projecting all the scans to a world frame and map building a primitive based representation.

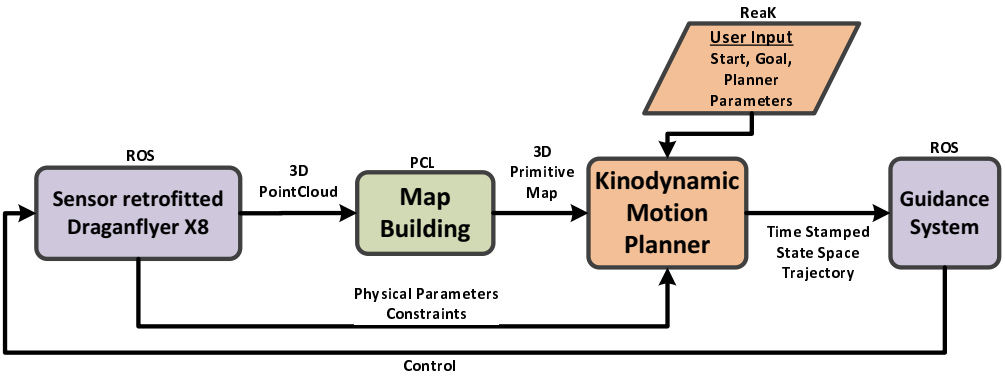


Figure 1–2: A manual flight is executed while laser scans are assembled (ROS)[2] and converted to primitives (PCL)[3] eventually invoking the motion planner off-line. Its output, a time stamped SS trajectory is given to guidance system for execution on the *X8*

The RRT* based motion planner is an extension of the previous publication by the author and Sharf in [46], and builds on the work by Glassman and Tedrake [20], and Webb and Berg [53]. More specifically, the motion planner, linearises the system non-linear dynamics about each sampled state and uses the MEAQR pseudometric to find which vertex in the tree is to be extended while simultaneously computing the optimal feedback controller (a.k.a., steering policy) to connect the two states. This allows the introduction of other dynamic constraints, such as, actuation magnitude and bandwidth limits which characterize real life systems. Collision checking is done as the steering function extends the tree into unexplored SS territory. The motion planner continuously runs, improving on the solution found thus far and can be stopped when the trajectory is needed by other components of the autonomous system or when no substantial cost change is recorded between one solution to the next. This trajectory is then passed to the guidance system

[1] Robot Operating System which provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source BSD license [45].

[2] The Point Cloud Library is a standalone, large scale, open project for 2D/3D image and point cloud processing. PCL is licensed under an open source BSD license [47].

of the aircraft commanding the vehicle to follow the trajectory or otherwise continue to hold, in case other conditions apply. The results of the motion planner are presented for two non-linear systems of different SS dimensionality: (i) a simple pendulum with actuator limits and (ii) a *Draganflyer X8* quadrotor.

The remainder of this thesis is organized as follows. In chapter 2, the related literature is explored, followed by chapter 3 presenting the framework of the probabilistic motion planner based on RRT*. In Chapter 4, the particulars of the proposed pseudometric for minimum energy trajectory generation and the incorporation of this pseudometric into an optimal sampling based motion planner for a non-linear system is elaborated. Then, dynamic models and simulation results for the two chosen robotic systems in artificially created environments are presented in chapter 5. This is followed by environment modelling described in chapter 6. Finally, chapter 7 summarizes the thesis and discusses potential research extensions.

Chapter 2
Related Work

The problem of planning a dynamically feasible trajectory for a robot is a core problem in modern-day autonomous systems designers' aspirations. Unfortunately, the problem is known to be at least PSPACE-hard, requiring consideration of the algorithm's *completeness* according to the following classification adapted from [34]. An algorithm is considered *complete* if, for any query, it terminates in finite time and either finds a solution or correctly reports that no solution exists. The combinatorial motion planning methods briefly noted in section 2.1.1 will achieve this. Unfortunately, such completeness is not obtained with sampling-based planning. Thus, two weaker notions of completeness are defined: *resolution completeness* and *probabilistic completeness*. An algorithm is *resolution complete* if it is *complete* for a certain parameter value defining the resolution of the underlying search space. An algorithm is *probabilistically complete* if the probability that it finds an existing solution, if one does exist, converges to one as the number of iterations increases. The RRT belongs to the *probabilistically complete* group of algorithms.

2.1 Existing Approaches

There are many algorithms available for solving the motion planning problem in continuous state spaces with numerous assumptions or restrictions. All of these approaches fall into one of the two main categories: *combinatorial motion planning* and *sampling-based motion planning* [34]. As part of this thesis, extensive motion planning surveys were conducted and are publicly available online [39, 40] for the interested reader.

2.1.1 Combinatorial Motion Planning

Combinatorial approaches solve the motion planning problem without resorting to approximations. Due to this property, combinatorial methods are also referred to as *exact*. To solve a motion planning query, all combinatorial approaches construct a *roadmap*. A roadmap provides a discrete representation of the continuous environment without loosing the original connectivity information required to solve the query. Such a query is resolved by connecting the start and goal states to the roadmap and then performing a discrete graph search within it. Some algorithms commence with a cell decomposition step while others create a roadmap directly. The early papers on combinatorial approaches are documented in [48]. The classical robot motion planning book by Latombe [33] covers the majority of combinatorial motion planning algorithms.

2.1.2 Sampling-Based Motion Planning

Sampling-based approaches to motion planning conduct a search that probes the continuous state space with a sampling scheme in order to avoid the explicit construction of the obstacle space (as the case in combinatorial approaches previously mentioned). The concept is depicted in fig. 2–1 adapted from [34]. Exploring the state space is possible with a collision detection module, which the motion planner considers as a "black box". This enables the development of the algorithm independently of the particular world representation and vehicle dynamics/kinematics. The collision detection module can handle any environment representation model such as algebraic sets, 3D triangles (mesh) and even non-convex polyhedra. This philosophy has been very successful in dealing with environments represented by thousands of geometric primitives in many different domains. Such problems would be practically impossible to solve using alternative techniques that implicitly represent the state space.

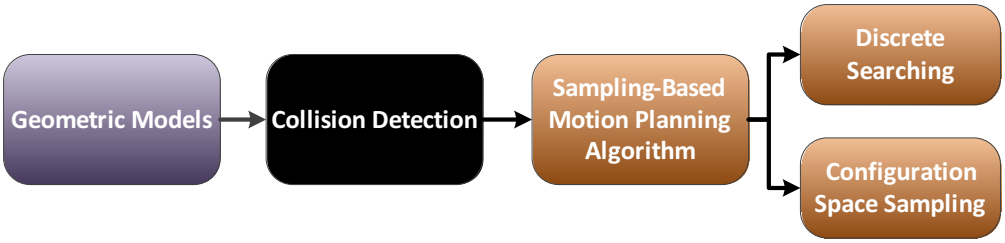


Figure 2–1: The sampling-based motion planning architecture uses a collision detection module as a "black-box". Thus, the planning is separate from the specific geometric world representation and the dynamic/kinematic vehicle model.

The sampling-based motion planning framework features both deterministic and probabilistic approaches. The deterministic sampling based methods include exhaustive search, depth-first search, breadth-first search, Dijaksta's algorithm [13] and the famous A^* [24]. The discussion in this thesis will be limited to the probabilistic sampling based approach presented through the consideration of two main planning methods, the Probabilistic Roadmap (PRM) [29] and the Rapidly-Exploring Random Tree (RRT) [35].

Both PRM and RRT have been proven to be probabilistically complete [35, 30]. Moreover, these methods utilize a space topology χ and a respective metric $\mathcal{D}$ to solve the motion planning problem. The metric used is a measure of proximity required to define state neighbourhoods and quantify the cost of edges connecting states within the space. The PRM framework can offer several routes to the goal in a single query and therefore deals well with wide open spaces. It avoids the computational complexity of the deterministic, complete algorithms (as are the combinatorial methods). It is a multi-query method divided into two planning phases: the off-line preprocessing stage and the on-line query. The preprocessing stage is aimed at constructing a graph of feasible edges connecting different states in the SS making future queries much easier. The on-line phase basically adds the initial and final states to the roadmap and searches the graph for the most suitable trajectory connecting those two. Whereas the PRM requires the metric used on the space to be a

true metric (see section 2.2), the unidirectional classical RRT algorithm does not.

The RRT is a better fit to the motion planning problems where building a roadmap *a-priori* may be irrelevant or simply infeasible, due to its single query nature. This allows for more options to consider for the metric and the tree edge connections. Many variants of RRT have been developed and used in several applications [3, 6, 7, 11, 12, 26, 37, 38, 50, 52], in particular, for autonomous vehicles [16, 31], humanoids [25, 27] and computer animation [21]. The RRT methodology builds a tree as it explores the SS by extending tree edges/branches towards randomly generated states starting from the initial state. The RRT is efficient in finding an initial solution, although it may contain unnecessary detours which are far from optimal. To ameliorate this side-effect, trajectory post-processing is usually conducted by, for example, smoothing [1] or spline fitting [32, 5].

2.2 Motion Planning Metrics

A key component which has a substantial effect on the convergence of any sampling-based motion planner is the distance metric used. This thesis deals with formulating a pseudometric to be used in the state space as an energy optimal trajectory is searched for. Thus, this section is dedicated to explaining what is a metric, how one might choose a suitable one and lists metrics commonly used in motion planning.

Commencing by defining a metric space $(\chi, \mathcal{D})$ which is a topological space χ equipped with a function $\mathcal{D} : \chi \times \chi \to \mathbb{R}^{\geq 0}$ such that for any $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \chi$ the metric holds to the following properties:

Nonnegativity: $\mathcal{D}(\boldsymbol{a}, \boldsymbol{b}) \geq 0$

Reflexivity: $\mathcal{D}(\boldsymbol{a}, \boldsymbol{b}) = 0$ if and only if $\boldsymbol{a} = \boldsymbol{b}$

Symmetry: $\mathcal{D}(\boldsymbol{a}, \boldsymbol{b}) = \mathcal{D}(\boldsymbol{b}, \boldsymbol{a})$

Triangle inequality: $\mathcal{D}(\boldsymbol{a}, \boldsymbol{b}) + \mathcal{D}(\boldsymbol{b}, \boldsymbol{c}) \geq \mathcal{D}(\boldsymbol{a}, \boldsymbol{c})$

A metric can represent linear or angular displacement, control effort or time. The space need not be a vector space, but the discussion here shall be limited to such vector spaces noting that $\chi \subseteq \mathbb{R}^n$. Choosing a metric that best represents the objective of the motion planning problem is a designers choice. Domain specific knowledge is obviously beneficial for choosing the metric which best fits the problem at hand, especially if parameter tuning is needed for the chosen metric. There are multitudes of metrics which are used in non-holonomic motion planning. Two that are related to the one developed in this thesis are:

Total Time For a dynamical system defined by eq. (1.1), the amount of time connecting two states $\boldsymbol{x}(0) = \boldsymbol{a}, \boldsymbol{x}(T_{total}) = \boldsymbol{b}$ can be used as a measure of distance:

$$\mathcal{D}(\boldsymbol{a}, \boldsymbol{b}) = \int_0^{T_{total}} 1 dt$$

It is asymmetric and thus considered a pseudometric. The distance is the time needed to steer the system from one state to the other. This metric can also be formulated as an *minimum-time* optimization problem more generally known as a two-point-boundary-value local optimization problem.

Control Effort Similar to the total time metric for a dynamical system in eq. (1.1), the total actuation effort needed to steer the system from $\boldsymbol{x}(0) = \boldsymbol{a}$ to $\boldsymbol{x}(T_{total}) = \boldsymbol{b}$, can be an asymmetric measure of distance:

$$\mathcal{D}(\boldsymbol{a}, \boldsymbol{b}) = \int_0^{T_{total}} ||\boldsymbol{u}||^2 \, dt$$

Again, this can be formulated as an optimal *minimum-actuation* two point boundery value problem.

2.2.1 Optimal Control Based Metrics

As already alluded, the distance metric is an essential part in SS exploration. The nearest neighbour search and the SS coverage rate are highly

dependent on the chosen metric. An efficient exploration of the SS is obtained only once the metric reflects the true cost-to-go [10] or actual distance between the states. The cost-to-go can be based on optimal control methods for linearised systems [35]. A linear quadratic regulator (LQR) used as a pseudometric was suggested by both Perez et al. [43] and Glassman and Tedrake [20]. Perez et al. suggested an infinite horizon LQR formulation like in eq. (2.1). The solution to this optimal control problem is then used as both a metric as well as an optimal control policy when extending the tree.

$$\begin{aligned} \mathcal{D}(\boldsymbol{a},\boldsymbol{b}) = \min_{u} \quad & \left\{ \int_0^\infty [(\boldsymbol{x}-\boldsymbol{b})^T Q(\boldsymbol{x}-\boldsymbol{b}) + \boldsymbol{u}^T R \boldsymbol{u}] dt \right\} \\ \text{given that} \quad & R = R^T \succ 0 \ , Q = Q^T \succeq 0 \\ \text{subject to} \quad & \dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} + C \\ & \boldsymbol{x}(0) = \boldsymbol{a} \end{aligned} \tag{2.1}$$

The symbols $\succ$ and $\succeq$ denote positive definite (PD) and positive semi-definite (PSD) matrices respectively. The matrices A, B, C are the Jacobians of the non-linear system dynamics model in eq. (1.1) whereas R and Q are the weight matrices in the LQR problem formulation. Although this pseudometric shows promising results in [43], it fails to capture the physical significance of the SS distance. It is difficult to properly incorporate the different elements of the state vector (of such different physical units) with appropriate significance into the pseudometric. In addition, it is not obvious how to scale the regulation matrices R and Q. When the optimal control policy derived from the solution to eq. (2.1) is used as the extension part of the motion planner, it suffers from two drawbacks: first, the state towards which we extend the tree might not be controllable in which case the quadratic regulator problem is ill-posed. Secondly, the infinite horizon controller yields an optimal solution only if the time approaches infinity - whereas the actual time for the extension step does not. This causes the suggested extension heuristic to under-perform, yielding

a sub-optimal trajectory or at best a slower convergence rate to that supposed optimal solution.

The two aforementioned references [43, 20] deal with non-linear systems with differential constraints (accounted for by representing the system in a higher dimensional SS), but do not take into account the actuation magnitude and bandwidth limitations. Glassman and Tedrake's work deals with improving exploration techniques for RRTs by approximating reachability regions but does not demonstrate practical applications on real high dimensional systems. To improve upon the previous work, Webb and Berg [53] introduced a fixed-final-state free-final-time LQR formulation to be used in the local planner as the metric and extension heuristic. Unfortunately, their work does not take into account any actuation constraints. With the view to ensuring dynamic feasibility and adherence to the real system characteristics, these must be introduced. This can be done either by incorporating a regulation term into the objective function which increases computational cost or by the method suggested in this thesis.

2.3 Optimal Probabilistic Motion Planning

To achieve optimal trajectories, the RRT* algorithm [28] was introduced as an extension of RRT that guarantees *almost-sure*[1] convergence to an optimal solution with respect to the defined metric. Although both the RRT and RRT* have been implemented successfully in various practical scenarios [34], a limitation of these probabilistic motion planners is their reliance on the ability to connect two distinct states with a trajectory (optimal at times). As previously mentioned, for holonomic robots, feasible trajectories might be as simple as a straight line along the SS connecting the two states. However, for kinodynamic systems, or robots with non-holonomic constraints, these straight lines

[1] The cost of the minimum cost path in the RRT* converges to the optimal cost c* almost surely, i.e., $Prob(lim_{N\to\infty} c(\sigma) = c*) = 1$

are not feasible trajectories since they do not adhere to the system dynamics. The method introduced in this thesis improves on the previous work by removing restrictive assumptions on the system and generates minimal energy trajectories for a real life platform such as the *X8*.

Chapter 3
The Probabilistic Motion Planner - RRT

This chapter briefly introduces the well-established RRT and RRT* algorithms for sampling-based motion planning with generic algorithmic components for the sake of simplicity.

3.1 Rapidly Exploring Random Tree

Initially presented by Kuffner and LaValle [35], RRT is an incremental sampling-based single query motion planner for holonomic systems. Since inception, it has been further developed and extended; however, the basic building blocks were maintained in all existing variants. In the following, the basic components of RRT are presented. Pseudocode for the algorithm can be found in algorithm 3.1.1 and the interested reader can consult existing literature for further details.

The RRT involves five main components and each can be encapsulated into a function or procedure as follows:

Sample When invoked, the sampling procedure returns a sampled state $\boldsymbol{x} \in \chi_{free}$ from a random distribution (uniform, normal with the goal as mean, or any other pseudorandom sequence as the designer chooses).

Steer Given two states $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \chi_{free}$ the steer procedure attempts to connect these two states. Depending on the specific design, it either returns a boolean indicating connection success and/or a path segment $\sigma \in \chi$ such that $\sigma(0) = \boldsymbol{x}_1$ but not necessarily terminating at $\boldsymbol{x}_2$.

Collision Check Given a path segment $\sigma(t) \in \chi$ or state $\boldsymbol{x} \in \chi$ this procedure returns a boolean true iff σ or $\boldsymbol{x}$ is within χ_{free} in its entirety; that is : $\sigma(t) \in \chi_{free} \ \forall \ t$ or $\boldsymbol{x} \in \chi_{free}$.

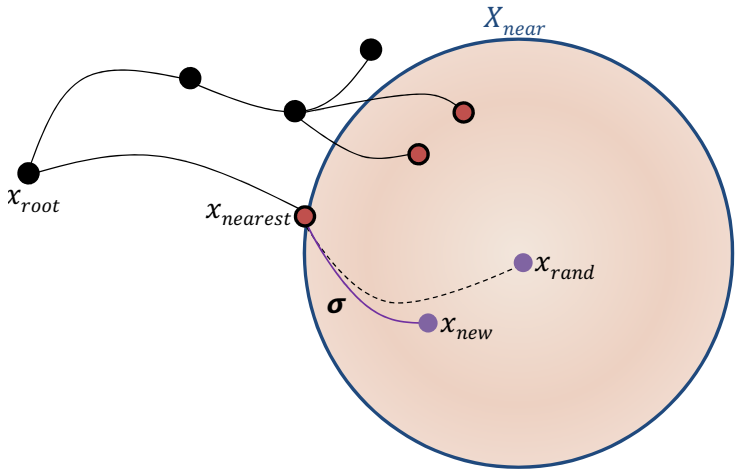


Figure 3–1: RRT extension step of $\boldsymbol{x}_{nearest}$ towards $\boldsymbol{x}_{rand}$. The subset of potential nearest neighbours $\mathcal{V}_{near}$ marked red are chosen from a region χ_{near} depicted as a circle in blue. Nodes in purple have not been added to the tree. The tree node $\boldsymbol{x}_{nearest}$ is steered towards $\boldsymbol{x}_{rand}$. The dotted path segment is the optimal trajectory connecting the two states where the actual output of the steering function is σ. The discrepancy between the dotted path and the steered segment is due to either actuation constraints or insufficient time horizon discussed in section 4.5

Metric A metric (or a pseudometric) is a function $\mathcal{D} : (\boldsymbol{x}_1, \boldsymbol{x}_2) \rightarrow \mathbb{R}^{\geq 0}$ returns a scalar which stands for the "distance" in whichever sense the metric has been defined between the two states. If the two states are not connectible, the metric returns an infinite distance. A true metric along with its properties was defined in section 2.2.

K Nearest Neighbours Given a subset of states $\mathcal{V} \subseteq \chi$ and a state $\boldsymbol{x} \in \chi$, this routine, also known as KNN, returns a subset $\mathcal{V}_{near} \subseteq \mathcal{V}$ of the nearest neighbours to $\boldsymbol{x}$ with respect to the metric $\mathcal{D}$ defined on the space, i.e., $\mathcal{V}_{near} = \left\{ \boldsymbol{x}_i \in \mathcal{V}, (i = 1, \cdots, K) \colon \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{x}) < \alpha \left(\frac{\log N}{N} \right)^{\frac{1}{d}} \right\}$ where d is the SS dimensionality, N is the number of states in the tree, α is a user defined volume parameter, and K is the number of neighbours queried for. These parameters define the neighbourhood in which the search is conducted.

The above components shall be now tied together into the basic RRT procedure. The tree is initialized with the $\boldsymbol{x}_{init}$ state as the root node. Then, a state $\boldsymbol{x}_{rand}$ is sampled from the SS with a predefined probability distribution. The nearest neighbour $\boldsymbol{x}_{near}$ to $\boldsymbol{x}_{rand}$ is searched for among all existing tree nodes (or a subset of those) with respect to the metric $\mathcal{D}$ defined on the SS. Once a neighbour $\boldsymbol{x}_{nearest}$ is chosen, the steering method is called to attempt to connect it to $\boldsymbol{x}_{rand}$ using the dynamics model forward integration with respect to time. The resulting trajectory (branch/edge) is denoted as σ and its end $\boldsymbol{x}_{new}$ might differ from $\boldsymbol{x}_{rand}$ due to collision, limited integration horizon or actuation constraints as per the discussion in section 4.5. If the generated trajectory and states are valid (no collision with workspace and state space obstacles), they are added to the tree. The aforementioned steps of sampling, searching for the nearest neighbour and eventually steering towards the random sample are called together the extension step which is depicted in fig. 3–1. The process is repeated until the number of iterations is exhausted or the goal region χ_{goal} is reached.

Algorithm 3.1.1 RRT

```
1: procedure RRT(x_init,χ_goal,χ_free,N)
2: Initialize(T, x_init)
3:     while i ≤ N do
4:         x_rand ← Sample(χ_free)
5:         x_near ← KNN(x_rand, T)
6:         [x_new, σ] ← Steer(x_near, x_rand)
7:         if CollisionFree(σ, χ_free) then
8:             T.edges ← T_edges ∪ σ
9:             T.vertices ← T_vertices ∪ x_new
10:        end if
11:    end while
12: end procedure
```

3.2 Optimal Rapidly Exporing Random Tree - RRT*

The RRT* is an asymptotically optimal sampling based trajectory generator. The main difference between the RRT* shown in algorithm 3.2.1 and

its predecessor the RRT are the **ChooseParent** and the **Rewire** routines. Those routines guarantee the *almost-sure* convergence to the optimal solution with respect to the metric defined on the space. The **ChooseParent** and the **Rewire** procedures presented in algorithms 3.2.2 and 3.2.3 are adapted from Perez et al. [43].

Algorithm 3.2.1 RRT*

procedure RRT*(x_{init},χ_{goal},χ_{free},N)
Initialize($\mathbf{T}, x_{init}$)
  while $i \leq N$ **do**
    $x_{rand} \leftarrow Sample$ ▷ Sample from the SS
    $x_{near} \leftarrow KNN(x_{rand}, \mathbf{T})$
    $[x_{new}, \sigma] \leftarrow Steer(x_{near}, x_{rand})$
    $\mathcal{V}_{near} \leftarrow KNN(x_{new}, \mathbf{T})$
    $[x_{min}, \sigma_{min}] \leftarrow ChooseParent(V_{near}, x_{new})$
    if $CollisionFree(\sigma, \chi_{free})$ **then**
      $\mathbf{T}.edges \leftarrow \mathbf{T}.edges \cup \sigma$
      $\mathbf{T}.vertices \leftarrow \mathbf{T}.vertices \cup x_{new}$
      Rewire($\mathbf{T}, \chi_{near}, x_{new}$)
    end if
  end while
end procedure

The **ChooseParent** procedure presented in algorithm 3.2.2 is used to ensure the optimality of connections in the tree. For a state $\boldsymbol{x}$ and neighbouring states $\mathcal{V}_{near}$ a candidate $\boldsymbol{x}_i \in \mathcal{V}_{near}$ is steered towards $\boldsymbol{x}$. Then, a search for the best parent, designated as $\boldsymbol{x}_p \in \mathcal{V}_{near}$, via which reaching $\boldsymbol{x}$ has the least cost, is conducted. The search is based on the actual cumulative incurred cost as opposed to the computed cost-to-go in the KNN search. Thus, each new state $\boldsymbol{x}_{new}$ we reach is guaranteed to have its parent state in the tree to be the one with the least actual incurred cost with respect to the metric defined on the SS.

The **Rewire** procedure presented in algorithm 3.2.3 is also used to ensure the optimality of connections in the tree. $\boldsymbol{x}$ is steered towards a candidate $\boldsymbol{x}_i \in \mathcal{V}_{near}$. If they are connected and the cost to reach the $\boldsymbol{x}_i$ via $\boldsymbol{x}$ is less

Algorithm 3.2.2 ChooseParent

```
1: procedure ChooseParent(x,V_near)
2:     MinCost ← ∞
3:     x_min ← ∅
4:     σ_min ← ∅
5:     for ∀x_i ∈ V_near do
6:         [x_new, σ] ← Steer(x_i, x)
7:         if Cost(x_i) + Cost(σ) ≤ MinCost then
8:             MinCost ← Cost(x_i) + Cost(σ)
9:             x_min ← x_i
10:            σ_min ← σ
11:        end if
12:    end for
13: end procedure
```

than the cost to reach $\boldsymbol{x}_i$ via its current parent, we remove the edge connecting $\boldsymbol{x}_i$ to its parent and reconnect it to $\boldsymbol{x}$. This ensures that the connections made are always improving with respect to the cost.

Algorithm 3.2.3 Rewire

```
1: procedure Rewire(T,V_near,x)
2:     for ∀x_i ∈ V_near do
3:         [x_new, σ] ← Steer(x, x_i)
4:         if Cost(x) + Cost(σ) ≤ Cost(x_i) then
5:             if CollisionFree(σ) then
6:                 x_parent ← Parent(x_i)
7:                 T.edges ← T.edges \ {x_parent, x_i}
8:                 T.edges ← T.edges ∪ σ
9:             end if
10:        end if
11:    end for
12: end procedure
```

It is important to note that although the steering function is not required to make a connection in general, when it is called upon from within the **ChooseParent** and **Rewire** routines it must. This is why both a boolean indicator as well as a path segment is returned from the steering routine. If a connection cannot be made, the routine iterates to the next potential connection.

The RRT* presented in this chapter is used as the framework for the motion planner proposed in this thesis. Next, the SS topology and its essential components for solving the motion planning problem are presented.

Chapter 4
Minimum Energy Affine Quadratic Regulator

This chapter introduces the Minimum Energy Affine Quadratic Regulator, how it is used as a pseudometric as well as a closed loop control for steering and how it is incorporated into the motion planner.

4.1 Affine Quadratic Regulator: Preliminaries

With the aim of generating an optimal trajectory, the first issue addressed is the specific cost by which the trajectory's optimality will be measured. This cost will be formulated as an objective function to be minimized along the generated trajectory, similar to the optimal control metrics used in non-holonomic motion planning mentioned in section 2.2.1. One of the barriers to autonomous aerial systems' capabilities is the limited mission endurance due to short battery life mentioned earlier in chapter 1. As is the case for small unmanned aerial vehicles, the chosen cost is one that provides a measure of the total energy consumed during a mission. It is well-known that for electromechanical systems, the total energy used for actuation is proportional to the integral of the norm of the control effort [2]. Therefore, the cost considered for the proposed motion planner is defined as:

$$c(\boldsymbol{\pi}, \boldsymbol{x}_{init}, \boldsymbol{x}_{goal}, T_{total}) = \int_0^{T_{total}} \frac{1}{2} \boldsymbol{u}^T R \boldsymbol{u} dt \ ; \ R \succ 0 \tag{4.1}$$

As alluded in chapter 2, computing the optimal control policy $\boldsymbol{\pi}^*$ represents a global optimization problem. One of the basic principles behind the RRT family of algorithms is to "divide and conquer" the global motion planning problem into smaller local sub-problems. Each is treated as an optimal control problem, that is, to optimally connect two states every extension step using a linear approximation of the system's dynamics. In the context of the

RRT* motion planner, each sampled state acting as a target state $\boldsymbol{x}_{rand}$ is essentially the linearisation point denoted in this chapter as $\boldsymbol{x}_0$. Hence, for a non-linear system such as in eq. (1.1), both the cost and the dynamics are reformulated about that target state such that $\boldsymbol{x} = \boldsymbol{x}_0 + \hat{\boldsymbol{x}}$ and $\boldsymbol{u} = \boldsymbol{u}_0 + \hat{\boldsymbol{u}}$ to yield:

$$\begin{aligned} c &= \sum_{i=1}^{n} \int_0^{T_i} \frac{1}{2}(\boldsymbol{u}_0 + \hat{\boldsymbol{u}})^T R(\boldsymbol{u}_0 + \hat{\boldsymbol{u}})dt \\ &= \sum_{i=1}^{n} \int_0^{T_i} \frac{1}{2}[\boldsymbol{u}_0{}^T R\boldsymbol{u}_0 + 2\boldsymbol{u}_0{}^T R\hat{\boldsymbol{u}} + \hat{\boldsymbol{u}}^T R\hat{\boldsymbol{u}}]dt \end{aligned}$$

where n is the number of segments that comprise the entire trajectory. Since $\boldsymbol{x}_0$ is an arbitrary state, which might not be a set point, the drift term $f(\boldsymbol{x}_0, \boldsymbol{u}_0)$ in eq. (4.4a) (C in eq. (4.4b)) is either minimized by solving the minimization problem:

$$\boldsymbol{u}_0 = \underset{\boldsymbol{u}}{\operatorname{argmin}} ||\boldsymbol{f(x_0, u)}|| \tag{4.2}$$

or simply set to zero and compensated for as described in section 4.3. The cross term $\boldsymbol{u}_0{}^T R\hat{\boldsymbol{u}}$ is neglected for the purposes of the approximation aided by the assumption that the relative magnitude and directions of $\boldsymbol{u}_0$ and $\hat{\boldsymbol{u}}$ yield a negligible term. Thus, the cost associated with segment i of eq. (4.2) can then be written as:

$$c_i = \int_0^{T_i} \rho_i + \frac{1}{2}\hat{\boldsymbol{u}}_i^T R\hat{\boldsymbol{u}}_i dt \;\; ; \;\; \rho_i \in \mathbb{R}^+, \;\; R \succ 0 \tag{4.3}$$

where ρ_i can be interpreted as half the generalized norm of the control signal needed to maintain the state at $\boldsymbol{x}_0$. To linearise the dynamics, a standard application of the Taylor expansion to eq. (1.1) gives:

$$\dot{\hat{\boldsymbol{x}}} = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_0, \boldsymbol{u}_0} \hat{\boldsymbol{x}} + \left.\frac{\partial f}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}_0, \boldsymbol{u}_0} \hat{\boldsymbol{u}} + f(\boldsymbol{x}_0, \boldsymbol{u}_0) \tag{4.4a}$$

or, using the state-space notation:

$$\dot{\hat{\boldsymbol{x}}} = A\hat{\boldsymbol{x}} + B\hat{\boldsymbol{u}} + C \tag{4.4b}$$

Following the *Bellman principle of optimality* [4], the optimality of each of the segments shall guarantee the optimality of the entire trajectory. To simplify the presentation, $\hat{()}$ notation is now dropped such that $\boldsymbol{x} = \hat{\boldsymbol{x}}$, $\boldsymbol{u} = \hat{\boldsymbol{u}}$ hereafter. Therefore, the local planning problem is to generate a trajectory segment σ_i connecting $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$ such that $\sigma_i(0) = \boldsymbol{x}_1$ and $\sigma_i(T_i) = \boldsymbol{x}_2$, while incurring minimal cost with respect to the objective function in eq. (4.3). Omitting the segment index i, the local planning problem is stated as:

$$\begin{aligned} \mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \min_{u} \quad & \left\{ \int_0^T [\rho + \frac{1}{2}\boldsymbol{u}^T R \boldsymbol{u}] dt \right\} \\ \text{given that} \quad & R = R^T \succ 0,\ \rho \in \mathbb{R}^+ \\ \text{subject to} \quad & \dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} + C \\ & \boldsymbol{x}(0) = \boldsymbol{x}_1 \\ & \boldsymbol{x}(T) = \boldsymbol{x}_2 \end{aligned} \tag{4.5}$$

The solution to the minimization problem on J (the argument of the minimzation function in eq. (4.5)) is used as a state-space distance pseudometric $\mathcal{D}$, and cost-to-go for the tree expansion step (closed loop control) as explained in section 4.3. It is referred to as a pseudometric because it violates the symmetry property of a true metric as defined in section 2.2 and is called the minimum energy affine quadratic regulator pseudometric (MEAQR) hereafter. However, it does conform to the triangle inequality due to the minimization operation yielding the "shortest" path in the metric sense. This formulation is similar to the optimal control metrics mentioned in section 2.2.1 as they are integrated with respect to time and the system dynamics are incorporated into the optimization problem.

To derive the optimal feedback control law, *Pontryagin principle of optimality* [36] is called upon, commenced by defining the Hamiltonian for the optimization problem in eq. (4.5):

$$H(\boldsymbol{\lambda}, \boldsymbol{u}, \boldsymbol{x}) = \rho + \frac{1}{2}\boldsymbol{u}^T R\boldsymbol{u} + \boldsymbol{\lambda}^T[A\boldsymbol{x} + B\boldsymbol{u} + C] \tag{4.6}$$

where the dynamics of the systems are incorporated in the Hamiltonian via the *Lagrange* multiplier $\boldsymbol{\lambda}$. The necessary conditions of the optimal trajectory and control are:

$$\dot{x} = \frac{\partial H}{\partial \boldsymbol{\lambda}} = A\boldsymbol{x} + B\boldsymbol{u} + C \tag{4.7a}$$

$$-\dot{\boldsymbol{\lambda}} = \frac{\partial H}{\partial \boldsymbol{x}} = A^T\boldsymbol{\lambda} \tag{4.7b}$$

and the optimal (open-loop) control policy can be computed from the stationary condition:

$$0 = \frac{\partial H}{\partial \boldsymbol{u}} = R\boldsymbol{u} + B^T\boldsymbol{\lambda}$$

$$\boldsymbol{u} = -R^{-1}B^T\boldsymbol{\lambda} \tag{4.8}$$

Initial and final conditions for the differential eq. (4.7b) will be derived in sections 4.2 and 4.3, depending on the specific usage.

4.2 Minimum Energy Pseudometric

This section deals with the actual computation of the cost incurred by connecting two states with respect to the suggested objective function in eq. (4.5) which is regarded as the MEAQR pseudometric. Moreover, the optimal time to complete that trajectory is also computed. This enables an approximation of the cost-to-go from a chosen tree state to any other state in unexplored SS. It is worth mentioning that the discrepancy between the actual system dynamics and the linearised system make the computation of the cost-to-go in section 4.2 depend on the proximity of the two states for which the pseudodistance is computed. Moreover, the pseudometric does not take into account the

actuation constraints which might affect the ability to connect the two states. This is dealt with in section 4.5.4.

4.2.1 Pseudometric Formulation

The *Lagrange* multiplier vector $\boldsymbol{\lambda}$ determined by its dynamics in eq. (4.7b) is independent of the state and a closed-form solution for it can be obtained by analytically integrating eq. (4.7b) back in time:

$$\boldsymbol{\lambda}(t) = exp[A^T(T-t)]\boldsymbol{\lambda}(T) \tag{4.9}$$

Before the above can be used to compute the optimal control policy of eq. (4.8) and subsequently the cost-to-go, the final condition $\boldsymbol{\lambda}(T)$ is required. The following derivation outlines a solution procedure for $\boldsymbol{\lambda}(T)$.

Substituting eq. (4.9) into eq. (4.4) and then into eq. (4.8) and integrating for the final state $\boldsymbol{x}_2$ gives:

$$\begin{aligned} \boldsymbol{x}(T) &= \boldsymbol{x}_2 \\ &= exp[AT]\boldsymbol{x}_1 - \int_0^T exp[A(T-\tau)]BR^{-1}B^T exp[A^T(T-\tau)]\boldsymbol{\lambda}(T)d\tau + \\ &\quad \int_0^T Cexp[A(T-\tau)]d\tau \end{aligned} \tag{4.10}$$

Introducing the *weighted controllability grammian* as:

$$G(t_0, T) = \int_{t_0}^T exp[A(T-\tau)]BR^{-1}B^T exp[A^T(T-\tau)]d\tau \tag{4.11}$$

which is a positive-definite (PD) symmetric matrix and can be found as the solution to the Lyapunov matrix differential equation in eq. (4.12) [36]:

$$\dot{P} = AP + PA^T + BR^{-1}B^T \tag{4.12}$$

with $P(t_0) = 0$, and accordingly, $G(t_0, t) = P(t)$. To simplify eq. (4.10), it is helpful to define two auxiliary variables:

$$\dot{D}(t) = AD(t) + C \quad ; \quad D(0) = 0 \tag{4.13a}$$

and

$$ZIR(\boldsymbol{x}_1, t) = exp[At]\boldsymbol{x}_1 + D(t) \tag{4.13b}$$

With the use of eqs. (4.12) and (4.13) in eq. (4.10), a concise form for $\boldsymbol{x}_2$ is obtained:

$$\boldsymbol{x}_2 = ZIR(\boldsymbol{x}_1, T) - P(T)\boldsymbol{\lambda}(T) \tag{4.14}$$

Adapting the missile guidance terminology for *zero-effort-miss* denoted by:

$$\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T) = \boldsymbol{x}_2 - ZIR(\boldsymbol{x}_1, T) \tag{4.15}$$

and using it in eq. (4.14), a solution for $\boldsymbol{\lambda}(T)$ can be immediately written as:

$$\boldsymbol{\lambda}(T) = -P^{-1}(T)\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T) \tag{4.16}$$

The above development, in particular, eqs. (4.9) and (4.16) allows writing the optimal control policy merely as function of the boundary states and the time horizon:

$$\boldsymbol{u}(t) = R^{-1}B^T exp[A^T(T-t)]P^{-1}(T)\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T) \tag{4.17}$$

The control policy in eq. (4.17) can be used to estimate the cost-to-go of a specific vertex from $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$ by substituting eq. (4.17) into the definition of J in eq. (4.5) and remembering the integral definition of the *weighted controllability grammian* in eq. (4.11) to give:

$$J(\boldsymbol{x}_1, \boldsymbol{x}_2, T) = \rho T + \frac{1}{2}\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T)^T P^{-1}(T)\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T) \tag{4.18}$$

For a specific initial and final condition $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ respectively, the cost-to-go function J is given in terms of the terminal time (arrival time). The time interval in which the trajectory segment is executed is denoted:

$$T^* = \underset{T}{\text{argmin}}\ J(\boldsymbol{x}_1, \boldsymbol{x}_2, T) \tag{4.19}$$

Once obtained, it allows the computation of the pseudodistance such that:

$$\mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_2) = J(\boldsymbol{x}_1, \boldsymbol{x}_2, T^*) \tag{4.20}$$

4.2.2 Computing the Optimal Time

The optimal T^* time is found by computing $J(\boldsymbol{x}_1, \boldsymbol{x}_2, T)$ from $T = 0$ until a predetermined time horizon $T_{horizon}$. As mentioned when first introduced, the matrix P is positive-definite and thus so is its inverse P^{-1}. However, as eqs. (4.12) and (4.13a) are integrated for an increasingly larger time intervals, several types of behaviours can occur for the weighted controllability grammian P. The matrix P can either be unbounded or can converge to some limiting solution $P(\infty)$. Extensive research has been conducted to formulate the necessary and sufficient conditions under which one could expect the Riccati differential equation (RDE) to converge (the Lyapunov matrix differential equation is a particular, simpler case of RDE). Since the cost-to-go in eq. (4.18) is comprised of a linear term in time and a quadratic term weighted by P^{-1}, it would be preferable to see the grammian $P \to \infty$ so that the second term in eq. (4.18) converges to zero. The convergence depends on the properties of the dynamic system, the linearisation point and the objective function weighting matrix R. The limiting behaviour of the grammian is determined by:

- non-uniqueness of the solution;
- insufficient time horizon - $T_{horizon}$;
- the non stabilizability[1] of the state and control matrices (A, B);
- the non detectability[2] of $(A, \sqrt{Q})$; There is no Q matrix in eq. (4.5).

[1] Stabilizability is a weaker notion of controllability - a system is considered to be stabilizable when all uncontrolable states have stable dynamics

[2] Detectability is a weaker notion of observability - a system is considered to be detectable if and only if all unobservable states have stable dynamics

The time $T_{horizon}$ for which the matrix ODE is integrated is finite, bounded and determined at each iteration by the probabilistic sampler as elaborated in section 4.4. Although the time horizon can be enlarged and the sampled states can be limited to stabilizable ones (limiting the RRT SS exploration capability), the non detectability of the system hinders any further attempts at guaranteeing the preferred behaviour of the RDE. However, regardless of the limiting behaviour of the grammian, there might still be a minimum value for the cost-to-go objective function for some $t \in [0, T_{horizon}]$; $T_{horizon} < \infty$. This is what makes the MEAQR method better than the infinite horizon affine quadratic regulator (where solution of the algebraic Riccati equation is attempted). The interested reader is referred to the works by Callier, Frank and Park [8, 15, 42] for studies on the RDE behaviour for various systems.

The reader's attention is directed to the fact that the cost-to-go is comprised of a linear term in T and another positive term so that the time dependant lower boundary for the cost is ρT. At every time step the cost $J(T)$ is compared to the minimal cost found thus far denoted by J^* with its corresponding T^*. Once $\rho T \geq J^*$ the search can be terminated at T_{stop} since no lower cost than the one already found may be obtained, J^* is then used as the pseudodistance value of $\mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_2)$. The relationship between J and time with the corresponding optimal values are graphically depicted in fig. 4–1.

In the unfortunate case where no minimal cost is found within the limited time horizon $T_{horizon}$, the projected cost is approximated based on the present minimal computed cost J_{min} and its corresponding time T_{min} as depicted in fig. 4–2. Since the pseudodistance is bounded from below by the term ρT, and the cost converges to that linear function (oscillations are caused by the matrix differential equation in eq. (4.12) for P), it is used as a reference

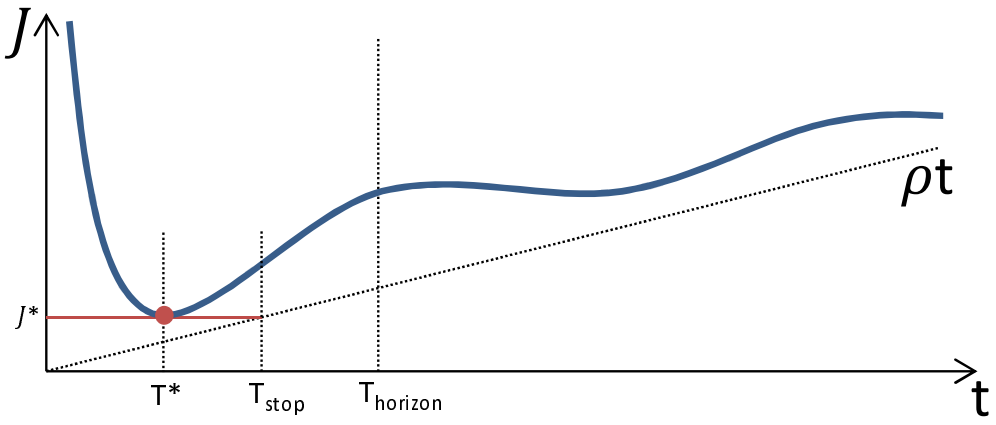


Figure 4–1: An illustration of the cost function vs. time during the search for optimal time and minimal pseudodistance. When the allowed time horizon is sufficiently large such that $T^* < T_{horizon}$ an exact value of J^* can be obtained

value. First, $T_1 = \frac{J_{min}}{\rho}$ is computed, then the time corresponding to the midpoint denoted as $T^* = \frac{1}{2}(T_1 + T_{horizon})$ and its corresponding optimal cost value $J^* = \frac{1}{2}(J_{min} + \rho T^*)$.

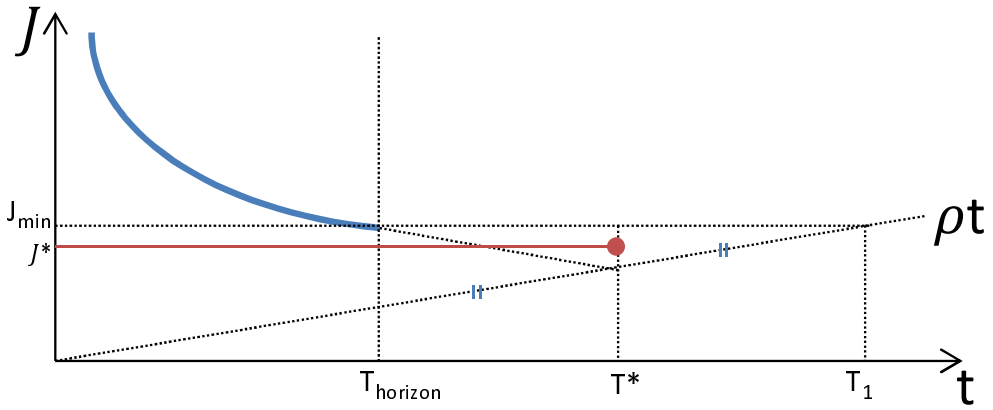


Figure 4–2: An illustration of the cost function versus time during the search for optimal time and minimal pseudodistance. When the time horizon is insufficiently large eventually resulting in an approximation of both the optimal cost J^* and optimal time T^*. T_{min} is equal to $T_{horizon}$ in this case.

Finally, the optimal time T^*, in whichever way it was obtained, is used to compute the closed loop control policy as presented in section 4.3. For practical implementation purposes, the closed loop gain matrix and bias vector presented in section 4.3 are computed simultaneously with the cost-to-go

estimation presented here. This is possible due to the dependence of the differential equations solely on the linearisation point as well as their resemblance to one another (eqs. (4.12) and (4.25)). This issue is further elaborated on in section 4.5.

4.3 Minimum Energy Closed-Loop Control

The closed loop policy is employed once the optimal time T^* has been found for connecting $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$, as described in section 4.2.2. As in real life systems, actuation is limited in magnitude as well as in bandwidth. Formulating a closed loop extension heuristic as opposed to an open loop policy enables us to account for the actuation constraints as will be detailed in section 4.5.4.

To formulate the closed loop control policy, $\boldsymbol{\lambda}$ is now assumed to be a function of the state $\boldsymbol{x}$ rather than an autonomous variable (as eq. (4.7b) might suggest). This is aimed at developing a feedback law which assumes an affine relationship between the *Lagrange* multiplier $\boldsymbol{\lambda}(t)$ and the state $\boldsymbol{x}(t)$ [19], that is:

$$\boldsymbol{\lambda}(t) = M(t)\boldsymbol{x}(t) + \boldsymbol{\eta}(t) \tag{4.21}$$

such that at the final time eq. (4.21) satisfies $\boldsymbol{\lambda}(T) = M(T)\boldsymbol{x}(T)$ inferring that $\boldsymbol{\eta}(T) = \mathbf{0}$. As for the matrix M, its final condition shall be derived shortly. Differentiating eq. (4.21) and substituting eqs. (4.4) and (4.8) into it yields:

$$\dot{\boldsymbol{\lambda}} = \dot{M}\boldsymbol{x} + M\dot{\boldsymbol{x}} + \dot{\boldsymbol{\eta}} = \dot{M}\boldsymbol{x} + M[A\boldsymbol{x} - BR^{-1}B^T(M\boldsymbol{x} + \boldsymbol{\eta}) + C] + \dot{\boldsymbol{\eta}}$$

equating the above result with eq. (4.7b) :

$$[\dot{M} + MA + A^TM - MBR^{-1}B^TM]\boldsymbol{x} + [\dot{\boldsymbol{\eta}} + A^T\boldsymbol{\eta} - MBR^{-1}B^T\boldsymbol{\eta} + MC] = 0 \tag{4.22}$$

The reader's attention is now directed to the two parts of eq. (4.22). The second term in square brackets describes the dynamics of $\boldsymbol{\eta}$ which, as an extra

degree of freedom, must adhere to the following differential equation:

$$\dot{\boldsymbol{\eta}} + [A^T - MBR^{-1}B^T]\boldsymbol{\eta} + MC = 0 \quad ; \quad \boldsymbol{\eta}(T) = 0 \tag{4.23}$$

The first bracketed term of eq. (4.22) must also equate identically to zero, that is:

$$\dot{M} + MA + A^T M - MBR^{-1}B^T M = 0 \quad ; \quad M(T) = \infty \tag{4.24}$$

This is a backwards integrated RDE, where the final condition for M arises from the fixed-final-state. This condition, however, makes the integration of eq. (4.24) backwards in time from T^* impractical, and eq. (4.24) is therefore replaced by the matrix differential equation for M^{-1}, using the relationship $\dot{M}^{-1} = -M^{-1}\dot{M}M^{-1}$:

$$\dot{M}^{-1} = AM^{-1} + M^{-1}A^T - BR^{-1}B^T \quad ; \quad M^{-1}(T) = 0 \tag{4.25}$$

To summarize, once the closed loop control policy is called upon in the motion planner, the following system is recalled from memory, as it was already integrated backwards in time during the pseudometric computation stage, according to section 4.5.2 :

$$\begin{bmatrix} \dot{\boldsymbol{\eta}} \\ \dot{M}^{-1} \end{bmatrix} = \begin{bmatrix} \left(A^T - MBR^{-1}B^T\right)\boldsymbol{\eta} + MC \\ AM^{-1} + M^{-1}A^T - BR^{-1}B^T \end{bmatrix} \; ; \; \begin{bmatrix} \boldsymbol{\eta} \\ M^{-1} \end{bmatrix}_{t=T} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.26}$$

The solution of eq. (4.26) for the optimal time duration of the segment T^* allows computation of the optimal closed-loop control policy for the entirety of the segment from eqs. (4.8) and (4.21).

4.4 Minimum Energy Affine Quadratic Regulator in RRT*

Now that the RRT* algorithm has been explained as well as the suggested pseudometric, the implementation scheme that integrates all the previous elements that comprise the proposed kinodynamic motion planner can be presented. Algorithm 4.4.1 is the pseudocode and fig. 4–4 is the flowchart of the entire algorithm with the MEAQR as the KNN search pseudometric as well as the MEAQR closed loop control law for extending the tree.

Algorithm 4.4.1 RRT* with MEAQR

procedure RRT*(x_{init},X_{goal},X_{free},N)
Initialize($\mathbf{T}, x_{init}$)
  while $i \leq N$ **do**
    $x_{rand} \leftarrow Sample$ ▷ Sample from the SS
    $x_{near} \leftarrow KNN(x_{rand}, \mathbf{T})$
    while $\mathcal{D}(x_{near}, x_{rand}) > \bar{\mathbb{E}}$ **do**
      $x_{rand} \leftarrow 0.5(x_{near} + x_{rand})$ ▷ Brought closer in L_2 sense
      $x_{near} \leftarrow KNN(x_{rand}, \mathbf{T})$
    end while
    $[x_{new}, \sigma] \leftarrow Steer(x_{near}, x_{rand})$
    $\mathcal{V}_{near} \leftarrow KNN(x_{new}, \mathbf{T})$
    $[x_{min}, \sigma_{min}] \leftarrow ChooseParent(\mathcal{V}_{near}, x_{new})$
    if $CollisionFree(\sigma_{min}, X_{free})$ **then**
      $\mathbf{T}.edges \cup \sigma_{min}$ ▷ Added iff $c(\sigma_{min}) < BestCost$ thus far
      $\mathbf{T}.vertices \cup x_{new}$ ▷ Added iff $c(\sigma_{min}) < BestCost$ thus far
      $\mathcal{V}_{near} \leftarrow KNNRewire(x_{new}, \mathbf{T})$
      Rewire($\mathbf{T}, \mathcal{V}_{near}, x_{new}$)
    end if
  end while
end procedure

Sampling. The SS representation is comprised of a vector with position and velocity states (might also be orientation and angular velocity where applicable). The sampling scheme utilized in this work is uniform on the position and orientation subspace (pose hereafter), meaning that only the pose states are sampled uniformly and the remaining states are set to zero. This fits within the characteristics of the sampling framework for sampling based planning for the class of problems dealt with in this thesis, since the initial

and goal regions both lie on the zero velocity manifold. Each random state sampled $\boldsymbol{x}_{rand}$ is used as the linearisation point so that the initial condition of the optimization problem of the MEAQR in eq. (4.5) is $\boldsymbol{x}_1 = \boldsymbol{x}_{near} - \boldsymbol{x}_{rand}$ and the ideal final state is $\boldsymbol{x}_2 = \mathbf{0}$. Sampling is step #1 in fig. 4–4.

Nearest Neighbours. Potential neighbouring states are searched for within a time horizon of $T_{horizon} = T_{max}\left(\frac{\log N}{N}\right)^{1/d}$ where N and d have already been defined in section 3.1 and T_{max} is a user-defined time threshold. This vicinity should be carefully selected and it is a function of the tree density [30]: too large a vicinity radius would render the linearisation invalid. Thus, special attention should be devoted to defining T_{max}, which in the context of the MEAQR pseudometric essentially bounds the allotted energy sphere radius $\bar{E}$ in which the KNN search in conducted. Once computed, P and $\boldsymbol{d}$ are used to determine which state is the nearest neighbour along with its corresponding optimal time T^*. If the pseudodistance of $\boldsymbol{x}$ to $\boldsymbol{x}_{rand}$ is $\mathcal{D}(\boldsymbol{x}, \boldsymbol{x}_{rand}) > 0.75\rho T_{horizon}$, $\boldsymbol{x}_{rand}$ is iteratively brought closer, until it is close enough, along the straight line (Euclidean sense) connecting the two states and denoted $\boldsymbol{x}_{rand_i}$ where i stands for the contraction iteration. In the rare case where the dynamics model of the system at hand is linear, it is beneficial to pre-compute all of the aforementioned time dependant matrices, P, D, M^{-1} and $\boldsymbol{\eta}$, prior to query. The nearest neighbour search is step #2 in fig. 4–4. The MEAQR KNN pseudocode is presented in algorithm 4.4.2.

Steering. Once the state $\boldsymbol{x}_{rand_i}$ is "close-enough", a steering attempt is made. Steering is regarded as acceptable only if $c(\sigma) > 0.1\mathcal{D}(\boldsymbol{x}, \boldsymbol{x}_{rand_i})$ to ensure minimal length path segments. The end of the path segment σ is referred to it as $\boldsymbol{x}_{new}$ hereafter. The closed loop control matrices M and $\boldsymbol{\eta}$ in eq. (4.26) are computed during the KNN search simultaneously with P and D in eqs. (4.12) and (4.13a) and saved associated with $\boldsymbol{x}_{new}$. If the optimal time T^* was the result of the approximation (as in fig. 4–2) and is

Algorithm 4.4.2 MEAQR K-Nearest-Neighbours

procedure KNN($\mathbf{T}$,T_{max},K,N)
  $k \leftarrow 0$ ▷ Initialize neighbour counter
  $T_{horizon} \leftarrow T_{max} \left(\frac{\log N}{N}\right)^{1/d}$
  $Candidates \leftarrow$ All vertices in $\mathbf{T}$
  while $t \leq T_{horizon}$ **do**
    $BestCost = \infty$
    for all $Candidates$ **do**
      $\mathcal{D} \leftarrow \rho t + \frac{1}{2}\boldsymbol{d}^T(t)P^{-1}(t)\boldsymbol{d}(t)$
      if $\mathcal{D} \leq BestCost(i)$ **then** ▷ Improvement in cost found
        $BestCost(i) \leftarrow J$
        $OptimalTime(i) \leftarrow t$
      end if
      if $BestCost(i) \leq \rho t$ **then** ▷ Cost threshold reached
        $Neighbours(k) \leftarrow Candidate(i)$ ▷ Register neighbour
        $Candidates \leftarrow Candidates \setminus$ Node(i)
        $k \leftarrow k + 1$
      end if
      if $k = K$ **then return** $Neighbours$
      end if
    end for
  end while
end procedure

larger than $T_{horizon}$, the steering method uses the initial entries of the gain and bias matrices M and $\boldsymbol{\eta}$ for the time interval of $t \in [0, T^* - T_{horizon}]$. Since these matrices are obtained by backwards integration as elaborated on in section 4.5.2, these are essentially the final entries prior to reversing the ordering. Steering is step #3 in fig. 4–4.

Collision Checking. Throughout the steering process, collision checking is done every few integration steps, depending on a predefined SS resolution. Meaning, once the state has been propagated some pseudodistance, the collision check is invoked. If collision is detected, the integration is terminated and the trajectory is truncated of the invalid part, returning the portion of the segment residing in χ_{free}. The specific realization of the collision checking is, as mentioned in 2.1.2, independent of the environment representation. In this thesis, a simple intersection check between the state (or trajectory segment) in

question and all of the obstacles primitives (representative of the environment) as well as SS bounds, is used as the collision checking procedure. Collision checking is step #4 in fig. 4–4.

Choose Parent. The neighbourhood from which the parent is chosen for $\boldsymbol{x}_{new}$ is computed with respect to that node since the pseudometric matrices differ with each linearisation point. Figure 4–3 illustrates this for the SS of inverted pendulum, where contours representing equal pseudodistances from three chosen states are shown; $\boldsymbol{x}_{center}$ is equidistant from those states in the L_2-norm sense but there is no one distinct point that is equidistant from all three states with respect to the MEAQR pseudometric. This is why for the purposes of neighbour or parent search, a metric based data structure as opposed to a simple linear search scheme is preferable, as will be addressed in section 4.5.5. Finally, the optimal parent is chosen and the corresponding edge is added to the tree. The optimal parent might be the original state $\boldsymbol{x}_{nearest}$ from which $\boldsymbol{x}_{new}$ was steered if it in indeed still within the neighbourhood, or any other state in the subset $\mathcal{V}_{near}$. The **ChooseParent** routine is step #5 in fig. 4–4.

Rewire. The neighbourhood of $\boldsymbol{x}_{new}$ used for rewiring is of high importance to the optimality of the tree and thus for the eventual trajectory generated by the motion planner. Unlike the extension stage, in the rewiring procedure the tree is extended *from* $\boldsymbol{x}_{new}$ *towards* its neighbours and the pseudodistance is to be computed using the **target** state $\boldsymbol{x}_{near} \in \mathcal{V}_{near}$ pseudodistance matrices P and $\boldsymbol{d}$ as opposed to the newly added node $\boldsymbol{x}_{new}$ matrices. The nearest neighbour search for the rewire process is depicted in algorithm 4.4.3 and the actual rewiring process was presented in section 3.2 and algorithm 3.2.3. The **Rewire** routine is step #6 in fig. 4–4.

Pruning. Once a solution is found for the first time, the cost is updated from ∞ (no path connecting the root to the goal region) to the computed

Algorithm 4.4.3 KNN REWIRE

procedure KNNREWIRE($\mathbf{T}$,T_{max},K)
  $Candidates \leftarrow$ All vertices in $\mathbf{T}$
  while $t \leq T_{max}$ **do**
    $BestCost \leftarrow \infty$
    for all $\boldsymbol{x} \in Candidates$ **do**
      $[P, d] \leftarrow \boldsymbol{x}.LoadMatrices(t)$
      $J \leftarrow \rho t + \frac{1}{2}\boldsymbol{d}^T P^{-1}\boldsymbol{d}$
      if $J \leq BestCost(i)$ **then** ▷ Improvement in cost found
        $BestCost(i) \leftarrow J$
        $OptimalTime(i) \leftarrow t$
      end if
      if $BestCost(i) \leq \rho t$ **then** ▷ Cost threshold reached
        $Neighbours \leftarrow Neighbours \cup \boldsymbol{x}$
        $Candidates \leftarrow Candidates \setminus \boldsymbol{x}$
      end if
      if $size(Neighbours) = K$ **then return** $Neighbours$
      end if
    end for
  end while
end procedure

value now called $MinCost$. In order to reduce computational complexity, tree *pruning* is done by discarding all states with a cost higher than $MinCost$.

Branch and Bound. As the tree continues to grow after an initial solution, the solution is further optimized and only states with cost lower than $MinCost$ are added to the tree - this is a *branch and bound* method, employed to minimize redundant states in the tree. One disadvantage of this method worth mentioning is that all subsequent solutions generated will be of the same homotopical class, thus restricting the solution space. Therefore, if the initial solution is not within the same homotopical class as the optimal solution, the latter will never be found.

Goal Bias. To shorten the time till an initial solution is generated, the sampling scheme can be biased towards the goal. There are multiple ways of inducing a bias: one possibility is to sample $\boldsymbol{x}_{rand} \in \chi_{goal}$ a specified percentage of the samples while the rest of the SS is sampled regularly. Another option

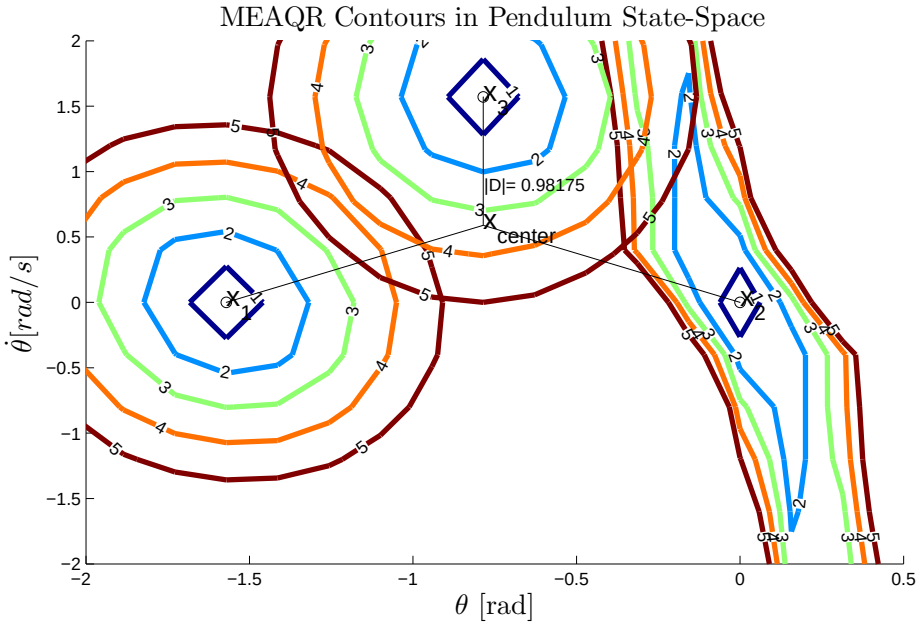


Figure 4–3: Inverted pendulum state space pseudometric contours which are highly dependent on the linearisation points $\boldsymbol{x}_i$, as can be seen from their form and values. All three states are equidistant (in the L_2 sense $\mathcal{D} = 0.98175$) from $\boldsymbol{x}_{center} = \left[\frac{\pi}{4}, \frac{3\pi}{16}\right]$. However, the MEAQR pseudodistance of $\boldsymbol{x}_{center}$ from each varies and equals to $7.388, 91.63, 2.82$ J respectively when $\rho = 1$ and $R = \frac{8}{3}$.

is to sample a Gaussian distribution with $\boldsymbol{x}_{goal}$ as its mean. The algorithm behaviour is sensitive to the bias parameter which should be adjusted with care. The disadvantage of a strong bias can be the resulting large regions of unexplored SS which affect the convergence of the RRT* to a solution and the optimal solution in particular. The extreme case of solely sampling the goal might yield the fastest solution for an environment without obstacles. On the other hand, the RRT* might get stuck, unable to expand if an obstacle is right in the way between the tree and the goal region (local minimum).

4.5 Practical Implementation

The algorithm presented in this paper was developed with the view to a specific practical application—motion planning for quadrotors; thus, the restrictions that arise for such are addressed in this section.

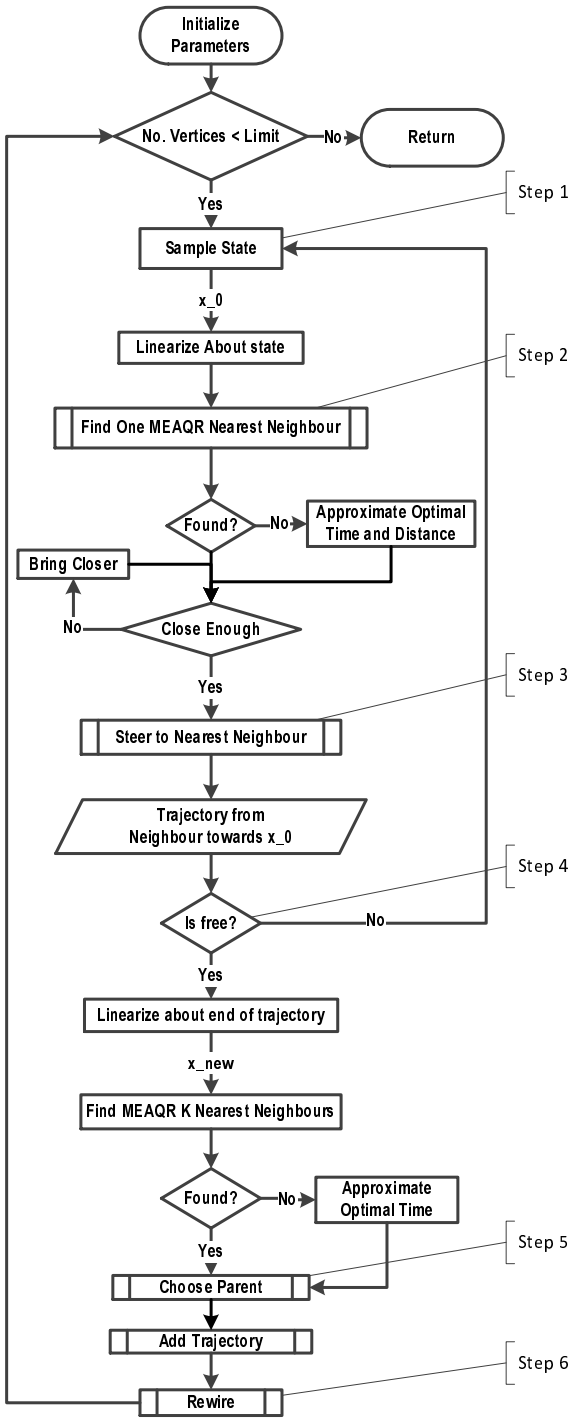


Figure 4–4: MEAQR RRT* Algorithm Flowchart

4.5.1 Weighted Controllability Grammian Positive-Definiteness

As the weighted controllability grammian P is integrated backwards in time, it is imperative it maintains its positive definiteness. Another concern is the inversion of the grammian required for computing the cost-to-go

in eq. (4.18). The initial condition for $P(0) = 0$ is merely theoretical and does not allow inversion, therefore, the matrix $P(0)$ is defined with small elements on its diagonal (numerical damping). Another concern is that the accumulated integration errors might result in deviation of P from maintaining its positive-definiteness. In order to ensure the PD of P as well as its counterpart M^{-1}, the *Cholesky* decomposition [49] is used in this implementation. Thus,

$$P = LL^T \tag{4.27a}$$

$$\dot{P} = \dot{L}L^T + L\dot{L}^T \tag{4.27b}$$

where L is lower triangular. Substituting eq. (4.27b) into the RDE in eq. (4.12) gives:

$$\dot{L}L^T + L\dot{L}^T = ALL^T + LL^TA^T + BR^{-1}B^T \tag{4.28}$$

and pre-multiplying by L^{-1} and post-multiplying by L^{-T}, yields:

$$\underbrace{L^{-1}\dot{L}}_{\text{Lower Triangular}} + \underbrace{\dot{L}^TL^{-T}}_{\text{Upper Triangular}} = L^{-1}AL + L^TA^TL^{-T} + L^{-1}BR^{-1}B^TL^{-T} \tag{4.29}$$

The left hand side of eq. (4.29) is a sum of a lower triangular matrix and an upper triangular matrix. Denoting this sum by $F = F_{LT} + F_{UT}$ and computing F from eq. (4.29), $\dot{L}$ is then obtained from the lower triangular part F_{LT} as:

$$\dot{L} = LF_{LT}$$

and is integrated to yield L. As the integration process proceeds, P can be computed from eq. (4.27a) as needed.

4.5.2 Integrating Backwards in Time

To integrate eqs. (4.23), (4.25) and (4.26) backwards in time, a generic ODE is assumed with a final condition formulated as follows:

$$\dot{y} = f(t, y(t)) \quad ; \quad y(t_f) = y_f$$

Introducing a change of variable for time τ:

$$\tau = t_f - t \quad \text{such that} \quad \frac{d\tau}{dt} = -1$$

the final condition ODE is reformulated as an initial value problem:

$$\frac{dy}{d\tau} = \frac{dy}{dt}\frac{dt}{d\tau} = -f(\tau, y(\tau)) \quad ; \quad y(\tau = 0) = y_f$$

The backwards integration process can be summarized in the following two steps:

1. Integrate the original ODE with a preceding minus sign, forward in time $\tau \in [0, t_f]$
2. Time reverse the solution such that $y(\tau = 0)$ becomes $y(t_f)$

4.5.3 Scaling the MEAQR Pseudometric

The MEAQR pseudometric must be scaled in an appropriate way to allow the pseudodistance computed to correspond to a physical quantity. If the cost connecting two states is the *energy* consumed along the trajectory in Joules, then the search for neighbours within specified $T_{horizon}$ is essentially bounding the allowed energy used to steer the system from one state to the next. With the units of the integrand as watts (power), the weighing parameter designated by ρ shall represent the power used to maintain the vehicle at the linearisation point $\boldsymbol{x}_0$ (assumed to be stabilizable). This parameter changes with $\boldsymbol{x}_0$ and can be approximated point-wise or for regions of the SS before the planner is queried and stored in a look-up table. For simplicity, ρ is assumed constant for the entire SS and is equal to the power needed to maintain the system idle. The second term in the integrand of eq. (4.5) is $\boldsymbol{u}^T R \boldsymbol{u}$, and hence for units of the control signal taken as Newton for thrust or Newton $\cdot$ meter for torque, R should have the units of $\frac{\text{Velocity}}{\text{Force}}$ or $\frac{\text{Angular velocity}}{\text{Torque}}$ respectively. If maximal values of velocity (angular velocity) and force (torque) are used (for the respective actuation axes), the term $\boldsymbol{u}^T R \boldsymbol{u}$ is equivalent to the fraction

of the available maximal power used in applying the control at that instance. This interpretation is both physical and intuitive. Additional scalar weights can be introduced in the matrix R at the designers' discretion to weigh the different actuation axes differently.

4.5.4 Actuation Limitation

In order to incorporate the actuation constraints into the formulation of the optimal trajectory, it is imperative to first identify them mathematically.

Magnitude Limit The control signal of the system is limited in magnitude due to electromechanical constraints for the two application examples employed in this thesis. An inverted pendulum is limited in motor torque and the *X8* vehicle is limited in thrust and moments about each of its axes. These constraints are not necessarily symmetrical and can be written as:

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max} \quad \forall t \in [0, T^*] \tag{4.30}$$

Bandwidth Limit The rate at which the control signal can change depends on the motor dynamics. Since the motion planner does not take the motor model into account explicitly, a limitation on the time derivative of the element-wise control signal is formulated as:

$$||\dot{\boldsymbol{u}}(\boldsymbol{t})|| \leq \dot{\boldsymbol{u}}_{max} \quad \forall t \in [0, T^*] \tag{4.31}$$

These constraints are defined on the control signal $\boldsymbol{u}$ associated with the path segment σ for the optimal time interval T^* computed in section 4.2.

Two methods are available to incorporate these limitations into the motion planning framework. These shall be called the *approximate* and the *iterative* methods. The *approximate* method is based on the fact that in infinite time linear quadratic regulators for stabilizable systems, the control signals and their time derivatives monotonically converge to zero. The explicit form

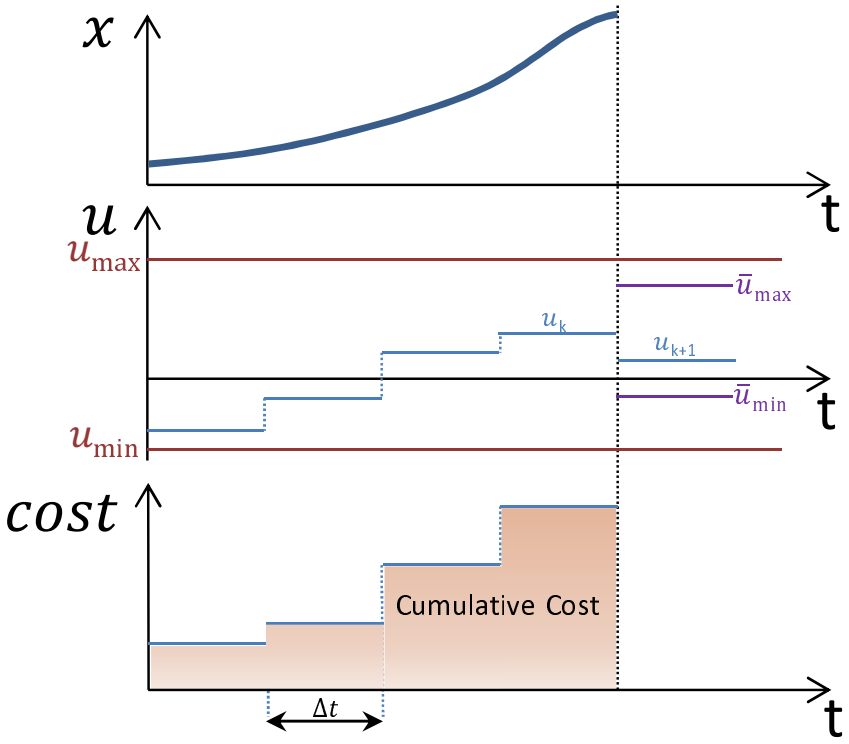


Figure 4–5: Illustration of the state, control and cost profiles versus time. The magnitude and bandwidth constraints are marked with red and purple lines respectively

of the control derivative $\dot{\boldsymbol{u}}$ can be obtained from eq. (4.17) as:

$$\dot{\boldsymbol{u}}(t)_{approx} = -R^{-1}B^T A^T exp[A^T(T^* - t)]P^{-1}(T^*)\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T^*) \qquad (4.32)$$

As can be seen from eqs. (4.17) and (4.32), the matrices R, B, A, P^{-1} and $\boldsymbol{d}$ are all constant and do not depend on time (T^* was already determined in section 4.2). Thus, the only term varying in time is $exp[A^T(T^* - t)]$ which obtains its maximal value when $t = 0$ if the system is indeed stabilizable. This *approximate* method can be used within the K-nearest-neighbours search when the pseudodistance is computed; if the magnitude or bandwidth constraints are violated for a given candidate neighbour, it is discarded as it is unreachable from the starting state and the routine continues to the next candidate.

The *iterative* method verifies that the control signal conforms to both constraints at every integration step. The minimal and maximal values, $\boldsymbol{u}_{max}$

and $\boldsymbol{u}_{min}$, are derived from the systems' physical properties whereas the time dependant bounds $\bar{\boldsymbol{u}}_{min}$ and $\bar{\boldsymbol{u}}_{max}$ are computed by:

$$\begin{aligned} \bar{\boldsymbol{u}}_{max} &= \boldsymbol{u}_k + \dot{\boldsymbol{u}}_{max}\Delta t \\ \bar{\boldsymbol{u}}_{min} &= \boldsymbol{u}_k - \dot{\boldsymbol{u}}_{max}\Delta t \end{aligned} \tag{4.33}$$

Both are depicted by dark and light red lines in fig. 4–5 which illustrates the state, actuation and cost versus time as the steering takes place. At each time step both the actuation and its approximate first derivative in eq. (4.34):

$$\dot{\boldsymbol{u}}_{iter} = \frac{\boldsymbol{u}_{k+1} - \boldsymbol{u}_k}{\Delta t} \tag{4.34}$$

are tested to verify their conformity to the bounds defined. If a violation is detected, the appropriate signal is saturated accordingly and the integration continues normally. While the *iterative* method is more computationally expensive, it is less conservative in discarding potential nearest neighbour compared to the *approximate* method. Therefore the *iterative* is the chosen method used in this thesis.

These added non-linearities in the system, undermine the validity of the optimal time computed in the open loop part of the algorithm. If the actuation limit is reached, it is highly probable that the system will not reach its target state by the designated time T^*. However, as the tree grows and its reachability region does as well, there will be fewer occasions where chosen neighbour states are unreachable. Hypothetically, it is possible to allow the steering function to propagate the system forward past the optimal time towards the target state until it is within a predefined proximity measure. However, this time interval needs to be determined prior to calling the steering function since the matrices M and $\boldsymbol{\eta}$ are highly dependent on it (due to the backwards integration in time) and the propagated edge can "over-shoot" its target. Thus, as already noted earlier, we shall only extend the tree for T^* presuming that as the tree grows, its

unreachable region will shrink and occurrences of non-target reaching segments will diminish.

4.5.5 Data Structures

As alluded to earlier, the KNN routine is of substantial computational complexity and there is much benefit in minimizing the amount of computations needed to find the subset of neighbouring states. The data structure is crucial from a memory management and complexity perspectives. The availability of this data saves redundant re-computations and enables all state-dependent operations on the tree to be easily executed.

As the RRT* grows and explores the SS, each state within it holds the following attributes that shall be noted as *State Data Entry*:

- State vector $\boldsymbol{x}$
- Linear state-space matrices A, B, C for the system linearised about $\boldsymbol{x}$
- Pseudometric time dependant matrices P, D
- Closed loop time dependant control matrices $M, \boldsymbol{\eta}$
- Cost from root state
- Pointers to parent state

All of the RRT* *State Data Entries* together are called the *Data Set*. The *State Data Entry* is constructed as follows: the state vector is registered once the state is sampled where as the linearisation, pseudometric and closed loop matrices are computed and logged immediately after. The cost from the root is updated along with the parent pointer once the state is added to the tree. The cost is also updated within the **Rewire** routine as connections are broken and others are made - ensuring a reduction overall.

Many data structures that can be used to accommodate the motion planner are available. Whichever structure is used, it is preferable that it be readily available from software libraries used for implementation and should also enable the following operations on it:

1. Easy Update - Minimal structural disruption when a new state is added to the data structure
2. Fast Query - When a state is queried for its nearest neighbours from among the SS tree, the number of operations needed to search down the structure (and their complexity) should be minimized.

The simplest approach to accommodate the tree requirements and desired operations is a linear search among the tree states organized as a list (also known as the naive approach). The structure update is easily done since it is merely a concatenation of a corresponding state data entry to the array. As for the query, as its name might suggest, the entire list is iterated while the K states yielding the lowest cost-to-go to the arbitrary state $\boldsymbol{x}_0$ are pointed to in a neighbours pointers list. This method has a running time of $\mathcal{O}(Nd)$, where its recalled that N is the number of states in the tree and d is the SS dimensionality. The linear search approach was used as a benchmark to improve upon when choosing the data structure eventually used in this thesis. Other data structures considered can be mainly classified into projective and metric methods and are briefly discussed below. These are tree-based data structures that arrange the SS states in a binary structure where each decision juncture is called a node.

Projective methods categorize the *Data Set* based on each entry's projection on some lower-dimensional space. The Kd-Tree [17] of Friedman, Bentley and Finkel has emerged as a useful tool in Euclidean space of moderate dimensions. It is a multidimensional generalization of a binary search tree. It can be constructed in $\mathcal{O}(N \log d)$ time. The Kd-Tree is built by recursively bisecting the database using a single SS coordinate position cuts as depicted in fig. 4–6. For a given SS axis, the data set is cut at the median of its projection onto that axis. An optimized Kd-Tree results by choosing the cutting axis to be that whose distribution exhibits the most variance. Similarly, the PCA-Tree

is not axes aligned but rather uses Principal Component Analysis to find the eigenvector along which the data set has the highest variance. This direction is then used for bisection. For SS of higher dimension ($d < 10$), both of these methods tend to visit nearly every data set element, approaching linear query time, and thus rendering these methods irrelevant [54]. Furthermore, these methods appear to ignore the underlying data structure hierarchy, and thus are generally more useful for smoother and abundant data sets which is not the case in the motion planning domain.

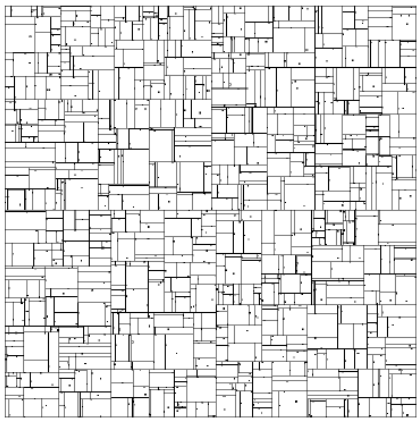
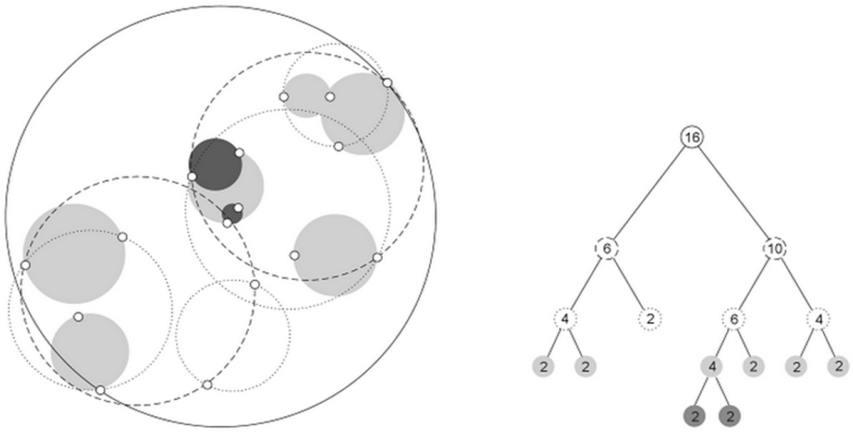


Figure 4–6: Kd-tree space decomposition method. The data set is marked as small dark dots and the coordinate-aligned fragmentation as straight lines is noticeable [56]

Metric methods are data structures that partition the space (organize the state data entries) based on any metric that can be computed for any pair of points. Thus, they do not require points to be finite-dimensional or even in vector space. A ball tree is a complete binary tree in which a ball (hyper sphere in the Euclidean sense) is associated with each partitioning node. Each node partitioning is binary in that all entries in the data set inside the ball are assigned to one child node (interior), and all externals are assigned to the other (exterior). An interior node's ball is the smallest to contain the balls of all its children. Unlike the node regions in Kd-Trees, the sibling hyper-spheres in ball-trees are allowed to intersect and need not partition the entire space.

Five methods of construction of ball-tree data structures are described in [41] and an example of a bottom-up construction method is depicted in fig. 4–7.



(a) Ball Tree Node Projection (b) Ball Tree Structure

Figure 4–7: Ball Tree Data Structure : A two-dimensional data set marked in small white circles, organized in a ball tree data structure. Balls of lower hierarchy are darker in shade [55]

Vantage point (Vp) trees [56] are similar to ball trees in their in/out and left/right decomposition, but rather than using a Euclidean hyper ball, the Vp-Tree employs a pseudometric from a selected vantage point taking advantage of symmetry and triangle inequality. State data entries which are *near* the vantage point, make up the *left/inside* subspace while the *right/outside* consists of *far* data entries as seen in fig. 4–8. Proceeding recursively, a binary tree is formed. Each of its nodes (a carefully selected state data entry) is considered a vantage point, and contains the boundaries of the subspace for its child nodes.

Given a metric space such as the SS defined in section 1.2, and a subset $\mathcal{V}_D$, it is beneficial to organize the data set so that the K nearest neighbours are more efficiently located. For a query $\boldsymbol{x}_0 \in \chi$ the K nearest neighbours operator within the energy hypersphere $\bar{E}$ is denoted $KNN|_{\bar{E}}(\boldsymbol{x}_0, \mathcal{V}_D, K)$. Each element of the SS in a sense has a *perspective* on the entire space, as perceived by it via the pseudodistance to all the other elements, making it a potential vantage

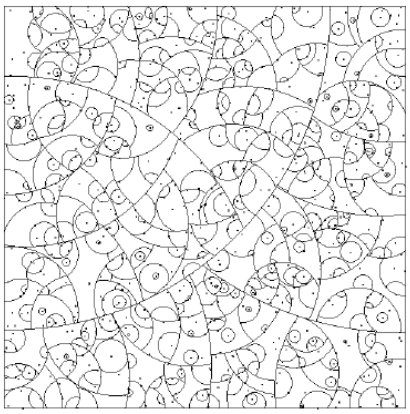


Figure 4–8: Vp-tree space decomposition. The data set is marked as small dark dots and the arcs are representative of the circles of the in/out subspace division [56]

point (a Vp-Tree node if chosen). This perspective is adapted for this thesis from [56] and best formulated mathematically:

Definition 1. *Let* $(\chi, \mathcal{D})$ *be a* $[0,1]^3$ *bounded metric space. Given a distinguished element* $p \in \chi$*, the following are defined for all* $a, b \in \chi$*:*

1. $\Pi_p : \chi \to [0,1]$ *is given by :* $\Pi_p(a) = \mathcal{D}(a,p)$.
2. $\mathcal{D}_p : \chi \times \chi \to [0,1]$ *is given by :*

   $\mathcal{D}_p(a,b) = |\Pi_p(a) - \Pi_p(b)| = |\mathcal{D}(a,p) - \mathcal{D}(b,p)|$

The function Π_p is best thought of as a projection of χ onto $[0,1]$ from the perspective of p ; that is, how χ is seen from p through the pseudometric perspective. The function now defined as $\mathcal{D}_p$ is not a metric since it is not symmetric but it does satisfy the triangle inequality. Relying on the previously asserted triangle inequality for the MEAQR pseudometric in section 4.2:

$$\mathcal{D}(a,b) \geq |\mathcal{D}(a,p) - \mathcal{D}(b,p)| = \mathcal{D}_p(a,b) \tag{4.35}$$

[3] Unbounded metrics can be easily scaled by $\bar{\mathcal{D}}(a,b) = \frac{\mathcal{D}(a,b)}{1+\mathcal{D}(a,b)}$

Hence, pseudo-distances tend to *shrink* when viewed from p by $\mathcal{D}_p$. A useful implication is:

$$\mathcal{D}_p(a,b) \geq \bar{E} \Longrightarrow \mathcal{D}(a,p) \geq \bar{E} \tag{4.36}$$

Therefore, if during a search, state $\boldsymbol{x}$ that is a pseudodistance $\bar{E}$ from $\boldsymbol{x}_0$ is encountered, there is no need to consider any element for which $\mathcal{D}_p(\boldsymbol{x}_0, \boldsymbol{x}) \geq \bar{E}$. Since the MEAQR pseudometric is indeed state dependent when incorporated into the RRT* framework, that is, as if every state is a perspective, it makes perfect sense to use the perspective-dependence property of the Vp-Tree. To deal with the asymmetry of the MEAQR pseudometric, the actual value used to represent the cost is the minimum of the two pseudodistance values such that:

$$\mathcal{D}(a,b) = \min\{\mathcal{D}(a,b), \mathcal{D}(b,a)\} \tag{4.37}$$

This is a conservative method ensuring that potential near neighbours are not discarded easily yet inducing an increase in computational cost. This thesis makes use of the *Dynamic Vp-Tree* (dVp) structure described in [18] that enables fast update operations and is available in ReaK library [44] developed by M. Persson. As can be seen from fig. 4–9, compared to the linear search, the performance of dVp-tree is substantially better in execution time of the KNN search with the MEAQR pseudometric, for trees of all sizes.

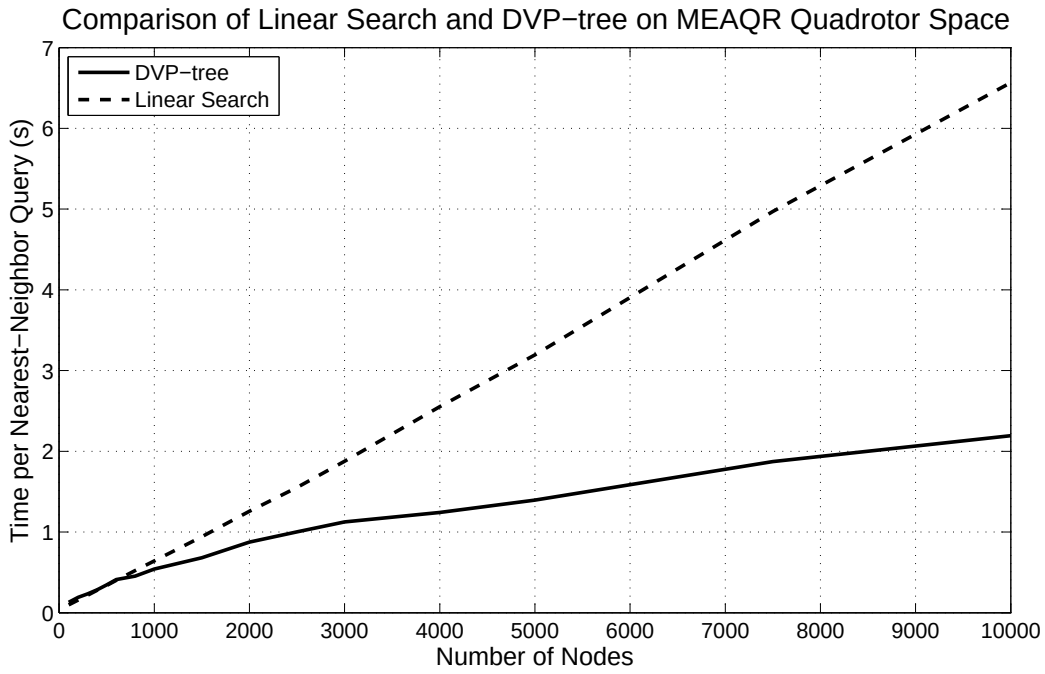


Figure 4–9: A log-log graph of the KNN execution time of the MEAQR pseudometric in a *X8* topology (13D) with linear search compared to a dVp-Tree structure [44]

Chapter 5
Simulations

The suggested algorithm was tested in simulation on two different kinodynamic non-linear systems: an inverted pendulum with actuation constraints (2D SS) and a quadrotor in 3D work space (13D SS) which is also non-holonomic. The two systems and the corresponding motion planning scenarios are discussed in section 5.1 and section 5.2 respectively. The inverted pendulum was chosen for its model simplicity as well as its low dimensionality while it is still a non-linear system. The *X8* is the target system for the contract this thesis is a part of; thus, a proof of concept is the main aim in designing a motion planner for use by that system. The *Monte-Carlo* simulations for the inverted pendulum were conducted on a 64-bit PC with Intel i7-2600 3.4Ghz with 8GB RAM running Mathworks Matlab© 2012b and simulations for the *X8* were conducted using C++ Reak [44] by M. Persson.

Although the classical SS sampling scheme is uniform in nature on all dimensions, a different sampling strategy was employed in the implementation for the two aforementioned systems as was elaborated in section 4.4. As for the goal biasing, instead of sampling the goal region every few samples, all newly added tree nodes $\boldsymbol{x}_{new}$ are attempted to connect to the goal region as long as those are within a predefined energy threshold. This solves the extreme bias problem by maintaining a uniform tree expansion while shortening the time elapsed till an initial solution is generated. As a benchmark for algorithm performance, the L_2-norm of the actuation signal $\mathbb{E}_{in} = \int_0^{T_{total}} ||\boldsymbol{u}||^2 dt$ is computed as well as the total trajectory time T_{total} for a constant ρ based on *Monte-Carlo* runs. A valid assumption is that there exists a minimal amount of electromechanical energy to bring a system from its initial state to the final

one. The successful planner will generate, with high probability, a trajectory corresponding to that minimal energy level, regardless of its parameters, which will be demonstrated in the following.

5.1 Torque Limited Pendulum

5.1.1 Dynamic Model

The torque limited inverted pendulum is depicted in fig. 5–1 and its dynamics with viscous frictional damping term are described with eq. (5.1):

$$\mathcal{I}\ddot{\theta} = -mgl\cos\theta - b\dot{\theta} + \tau \tag{5.1}$$

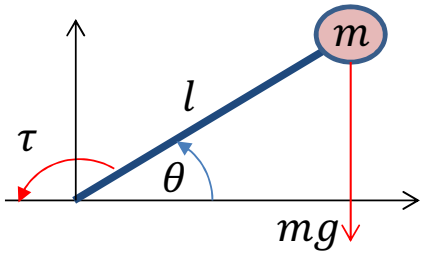


Figure 5–1: Inverted Pendulum Free Body Diagram

The state vector is defined as $\boldsymbol{x} = \left[\theta, \dot{\theta}\right]^T$ and the parameters of the model are: moment of inertia $\mathcal{I} = ml^2$, mass $m = 1kg$, length $l = 1m$, damping coefficient $b = 0.1Nms$ and $g = 9.81ms^{-2}$. In SS formulation, the dynamics model is:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}(2) \\ \frac{1}{ml^2}\left(\tau - b\boldsymbol{x}(2) - mgl\cos\boldsymbol{x}(1)\right) \end{bmatrix} \tag{5.2}$$

The pendulum SS bounds arise from dynamic and geometric constraints as listed in table 5–1 along with the actuation magnitude $\|\tau\|$ and bandwidth (rate) $\|\dot{\tau}\|$ constraints. The pendulum is referred to as under-powered since the torque required to maintain it at the zero state $\boldsymbol{x} = [0, 0]^T$ is:

$$\tau_{zero} = mgl = 9.81Nm$$

Table 5–1: Pendulum State Space and Actuation Bounds

State	Units	Lower Bound	Upper Bound	Maximal Rate
τ	N m	-3	3	10
θ	rad	$-\pi$	π	-
$\dot{\theta}$	rad/sec	-8	8	-

whereas there are only $\pm 3Nm$ available from the actuator. The set-points in the SS are those for which both states are zero:

$$\boldsymbol{x}_{set} = \begin{bmatrix} 0 \\ \arccos\left(\frac{\tau}{-mgl}\right) \end{bmatrix} \tag{5.3}$$

as represented in the SS by merely a line along the $\dot{\theta} = 0$ axis on the interval $\theta \in [-0.31, 0.31]$, symmetric about the downward equilibrium state of the pendulum as expected.

The environment in which the pendulum manoeuvres is obstacle free. The sampling scheme is of the form $\boldsymbol{x}_{rand} = [\theta, 0]^T$ where $\theta \sim \mathcal{U}(-\pi, \pi)$. This scheme avoids redundant sampling and increases the probability of sampling a stabilizable point (a set-point) which improves the behaviour of the proposed pseudometric as discussed in section 4.5.1. The trajectory generated by the motion planner originates from the downward equilibrium state $\boldsymbol{x}_{init} = \left[-\frac{\pi}{2}, 0\right]^T$ searching a trajectory to the inverted equilibrium state $\boldsymbol{x}_{goal} = \left[\frac{\pi}{2}, 0\right]^T$. The change in energy, in the absence of non-conservative forces needed to quasi-statically bring the pendulum from the initial zero-velocity state to the goal zero-velocity state is simply the change in the gravitational potential energy of the pendulum, that is:

$$\Delta\mathbb{E} = mgl(1 + \sin\left(\frac{\pi}{2}\right)) = 19.82 \ J \tag{5.4}$$

The input energy $\mathbb{E}_{in}$ and the frictional forces are the non-conservative forces, and the work-energy balance is therefore:

$$\Delta\mathbb{E} = \mathbb{E}_{in} + W_f \tag{5.5}$$

The work done by the viscous friction depends on the specific trajectory (as for non-conservative forces), and is computed numerically for the sample optimal trajectory presented in fig. 5–3 as:

$$W_f = \int_{\theta} \tau_f d\theta = -\int_0^{T_{total}} b\dot{\theta}^2 dt = -3.14 \ J \tag{5.6}$$

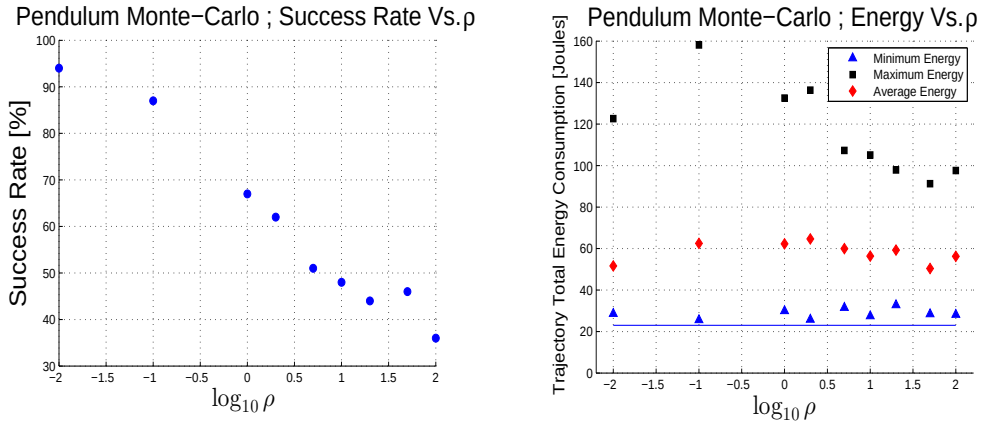
This suggests that an approximate lower bound for the pendulum system, which shall serve as a theoretical benchmark for the motion planner performance is:

$$\mathbb{E}_{in} = \Delta\mathbb{E} - W_f \approx 23 \ J \tag{5.7}$$

5.1.2 Monte Carlo Results and Discussion

Montre-Carlo simulations are conducted with the proposed MEAQR RRT* motion planner. It is queried 100 times for each value of ρ with $R = \frac{8}{3}$, and $T_{max} = 5[s]$ limiting the number of tree states to $N_{max} = 2000$ and running time to 10 minutes. Table 5–2 and fig. 5–2 report the results of *Monte-Carlo* simulations for nine different values of ρ in the range 0.01 to 100.

As can be seen from table 5–2, the average values of energy for varying ρ range between $50 - 65J$ with a large standard deviation of approximately 40% on average. This is far from the energy consumption for the optimal trajectory, leading to the conclusion that for many runs, the planner had not sufficient time (as well as sufficient states) to converge to the optimal solution. Over all, such high standard deviation ($> 100\%$) means there is no true meaning to the average value. Thus, a more clear picture of the Monte-Carlo simulation results is depicted in fig. 5–2 which in addition to the average values shows the minimum and maximum energy trajectories.



(a) Success Rate Vs. ρ

(b) Energy Vs. ρ

Figure 5–2: Motion planner (a) success rate for generating a trajectory and (b) total actuation energy input (minimum, maximum, average) for varying values of ρ. The approximate lower bound $\mathbb{E} = 23J$ is also shown as a blue line. Values are generated by running 100 *Monte-Carlo* simulations for each value of ρ.

Theoretically, the energy value needed for a trajectory is bounded from below by the global minimum, approximated in eq. (5.7), which is needed to steer the pendulum from start to goal. The results in fig. 5–2 along with the energy bound $\underline{\mathbb{E}}_{in}$ show that changing the pseudometric parameter ρ does not yield a significant difference in the computed minimum energy used on a given trajectory, which is a desirable finding. From above, the energy value is unbounded since there may be cases when no trajectory is generated by the motion planner, hence, the infinite cost.

A run for which no trajectory was generated is discarded, and counted as a failure (hence the success rate). The reason for monotonic decrease in the success rate as ρ increases is due to the way ρ affects the local planner through its representation in the pseudometric. As ρ increases, the optimal time T^* is shortened (the termination condition of $MinCost < \rho T$ is reached sooner), resulting in shorter trajectory segments. Thus, for given N (number of states in the tree), the branches (and therefore growth and region of reachability) are much smaller relative to a tree with a lower ρ value.

Table 5–2: Pendulum *Monte-Carlo* simulation results. The mean and standard deviation of the energy $\mathbb{E}_{in}$, trajectory time T_{total} and computation time $T^{1^{st}}$ till a solution is available.

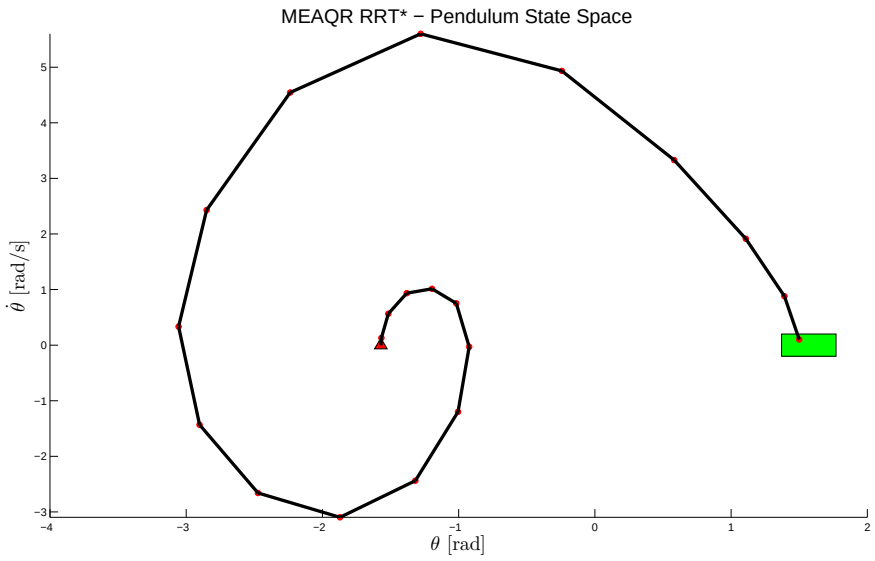
ρ	$\mu_{\mathbb{E}} \pm \sigma_{\mathbb{E}}[Joules]$	$\mu_{T_{total}} \pm \sigma_{T_{total}}[s]$	$\mu_{T^{1^{st}}} \pm \sigma_{T^{1^{st}}}[s]$	Success Rate [%]
0.01	51.63 ± 19.27	5.89 ± 2.19	71.83 ± 102.1	94
0.1	62.48 ± 27.30	7.18 ± 3.15	111.4 ± 171.9	87
1	62.27 ± 22.76	7.38 ± 2.5	61.81 ± 89.79	67
2	64.62 ± 24.40	7.76 ± 2.7	103.76 ± 136.81	62
5	59.95 ± 18.60	7.17 ± 2.12	107.97 ± 132.52	51
10	56.37 ± 17.97	6.69 ± 2.08	135.3 ± 152.43	48
20	59.27 ± 17.14	6.92 ± 1.85	95.84 ± 166.3	44
50	50.40 ± 15.72	5.86 ± 1.79	70.41 ± 109.45	46
100	56.26 ± 17.55	6.47 ± 1.95	91.9 ± 141.19	36

The decreasing trend in the maximum energy used, is due to the decreasing success rate, as there are significantly fewer trials to represent the true underlying distribution of energy consumption for the generated trajectories. If infinitely many Monte-Carlo simulations were attempted, the maximal energy levels recorded will be, as mentioned, unbounded. Probabilistically speaking, for each set of pseudometric parameters, there might be at least one simulation that is able to generate a highly inefficient (*nearly infinite cost*) trajectory right before the time limit is reached and the simulation is terminated.

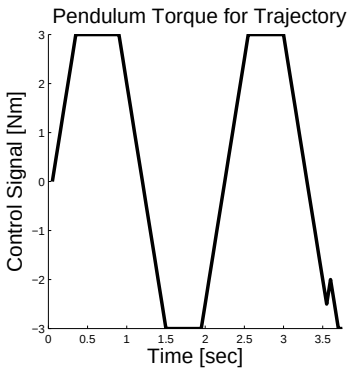
As claimed earlier, if an optimal trajectory exists, for which the energy consumption is minimal, the algorithm shall converge to it, regardless of the parameters ρ and R. The parameters do however, affect the rate of convergence of the algorithm to the optimum. Given a sufficient amount of time and a higher upper bound on the number of states allowed in the tree, trials should eventually converge to that optimal trajectory. The closer the approximation of the pseudometric to the actual cost at each linearisation point, the faster is the convergence.

5.1.3 Sample Trajectories

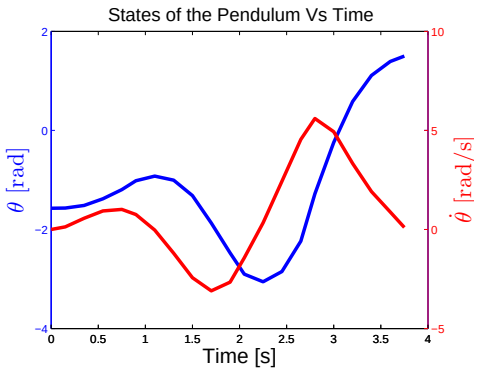
Sample energy optimal and suboptimal trajectories along with their actuation signal and states versus time are presented in figs. 5–3 and 5–4 respectively. The optimal trajectory reached the goal region within two swings across the downward equilibrium point applying the maximal available torque in synchronization with the sign of $\dot{\theta}$, maximizing the positive work done by the actuator. In contrast, the suboptimal trajectory swings multiple times, applying torque not in phase with $\dot{\theta}$. During these intervals, the actuator does negative work thus reduces the amount of mechanical energy in the system, wasting valuable power and resulting in a sub-optimal trajectory. The saturation and bandwidth limits are apparent in the actuation plot versus time in figs. 5–3(b) and 5–4(b).



(a) Pendulum State Space

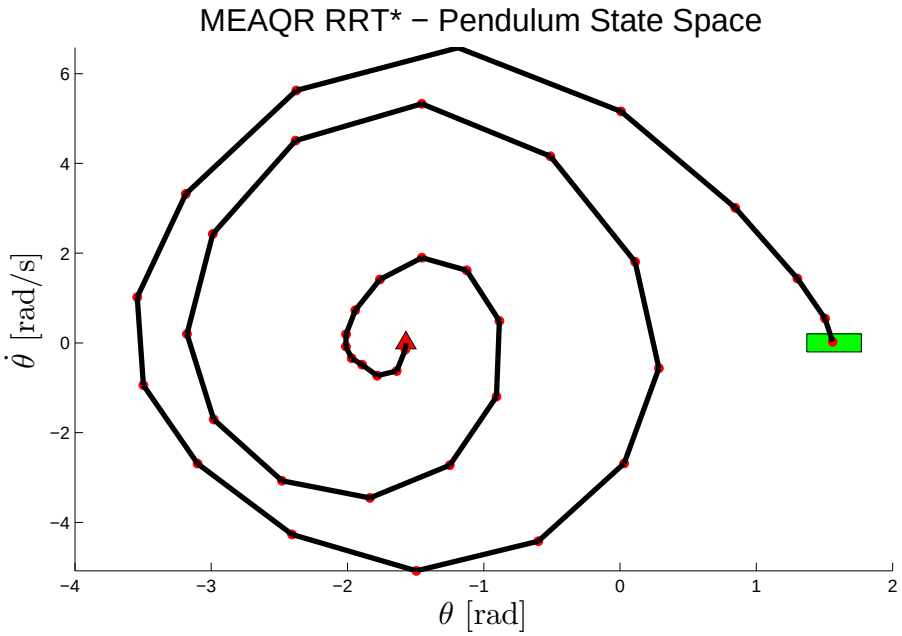


(b) Actuation

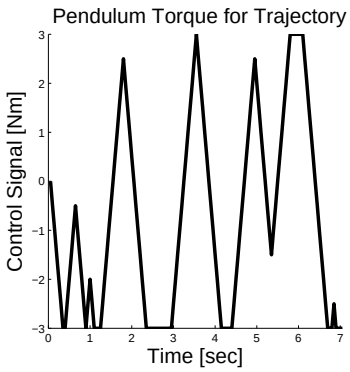


(c) States Vs. Time

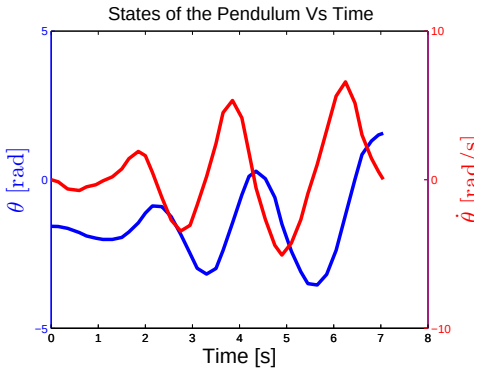
Figure 5–3: An energy optimal trajectory of the inverted pendulum with the MEAQR pseudometric parameters: $\rho = 1, R = \frac{8}{3}$. The total energy needed is $\mathbb{E}_{in} = 25.2\ J$ where as the work done by the viscous friction is $W_f = -3.14\ J$



(a) Pendulum State Space



(b) Actuation



(c) States Vs. Time

Figure 5–4: A suboptimal trajectory of the inverted pendulum with the MEAQR pseudometric parameters: $\rho = 1, R = \frac{8}{3}$. The total energy needed is $\mathbb{E}_{in} = 48.96\ J$ where as the work done by the viscous friction is $W_f = -1.98\ J$

5.2 Draganflyer X8

The quadrotor simulation is based on the previously mentioned *Draganflyer X8* vehicle shown in fig. 5–5. Due to the complexity of the vehicle and the absence of a comprehensive model for this specific vehicle, it is modelled in appendix A as a four rotor helicopter (quadrotor) with approximated aerodynamic body drag and no gyroscopic affects. The state vector $\boldsymbol{x}$ of the *X8* is comprised of the position $\boldsymbol{p} = [x, y, z]$, velocity $\boldsymbol{v} = [u, v, w]$, a quaternion rotation representation $\boldsymbol{q} = [q_0, q_1, q_2, q_3]$ (q_0 is the scalar) and angular rates $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$ and is formally defined in eq. (A.2) with the non-linear state-space model given in eq. (A.3).

Figure 5–5: The *Draganflyer X8* retrofitted with LIDAR, beam directing mirrors and a GoPro camera.

The quadrotor state bounds arise from dynamic and geometric constraints which are listed in table 5–3. The actuation pair $\boldsymbol{u} = (\mathcal{F}, \boldsymbol{\tau})$ of dimensionality 4, is comprised of the total thrust $\mathcal{F}$ and $\boldsymbol{\tau} = [\mathcal{L}, \mathcal{M}, \mathcal{N}]$ which are torques in body axes. It is limited in magnitude $\|\boldsymbol{u}\|$ and in bandwidth (rate) $\|\dot{\boldsymbol{u}}\|$ as listed in table 5–4. The non-holonomic constraints are derived and shown in eq. (A.4) without aerodynamic drag terms. Those constraints represent an explicit relationship between the orientation (represented for human readability by two Euler angles θ and ϕ) and the inertial acceleration and the yaw angle ψ. The non-holonomic constraints can be incorporated into the sampling scheme of the planner, allowing to compute θ and ϕ as a function of $\ddot{\boldsymbol{p}}$ and

ψ. However, by doing so, the SS representation must include the acceleration terms, which leads to a SS of higher dimensionality than the original formulation. Instead, the steering function propagating the dynamics model forward in time ensures adherence to those constraints. The workspace bounds listed in table 5–3 are scenario dependent, and are identical for the two artificial environments considered in this thesis.

Table 5–3: *X8* state space bounds

State	Units	Lower Bound	Upper Bound
$[x, y, z]$	m	0	5
$\|\|\boldsymbol{v}\|\|$	m/sec	0	6
$\|\|\boldsymbol{\omega}\|\|$	rad/sec	0	3

Table 5–4: *X8* actuation bounds

Input Name	Units	Lower Bound	Upper Bound	Maximal Rate
$\mathcal{F}$	N	0	35	20
$\mathcal{L}$	N m	-5	5	10
$\mathcal{M}$	N m	-5	5	10
$\mathcal{N}$	N m	-3	3	5

The MEAQR RRT* algorithm is queried with the pseudometric and planner properties in table 5–5. The planner is tested on two artificial environments in which the *X8* manoeuvres; both contain obstacles in the form of geometrical primitives.

Table 5–5: *X8* MEAQR pseusdometric parameters

Parameter	ρ	R	$T_{max}[s]$	N_{max}
Value	20	$diag(\frac{6}{35}, \frac{3}{5}, \frac{3}{5}, 1)$	5	2000

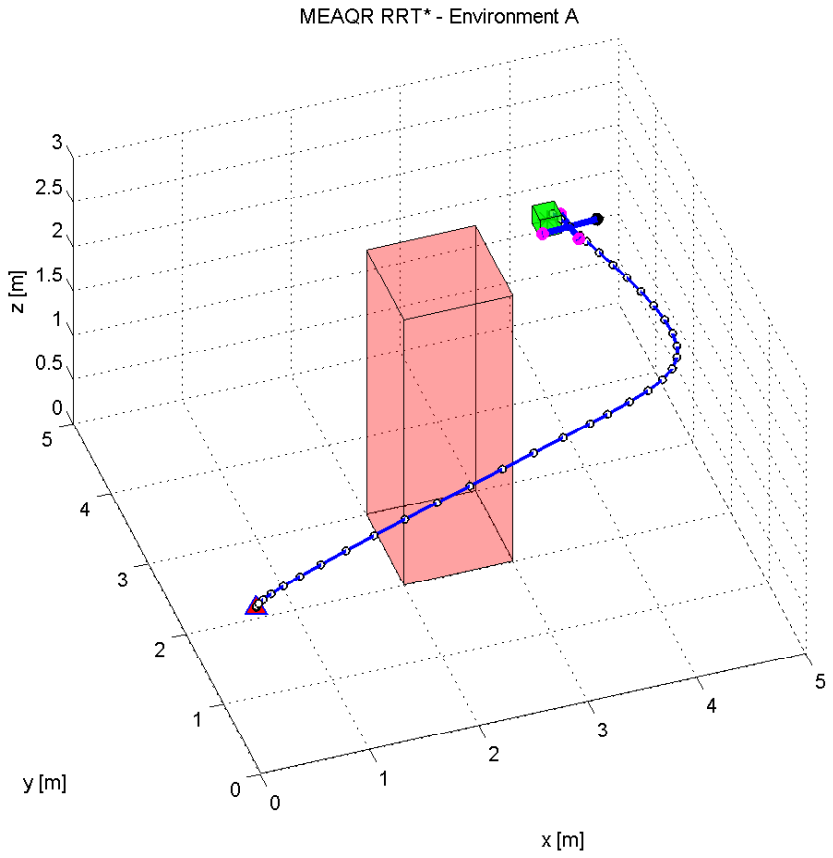
5.2.1 Single Pillar Room

In this scenario the motion planner is invoked to determine a path for the *X8* in a $5m \times 5m \times 5m$ space with a prism-like obstacle in the center (see fig. 5–6(a)). The initial and target positions queried are $\boldsymbol{p}_{init} = [0.75, 0.75, 1]$ and

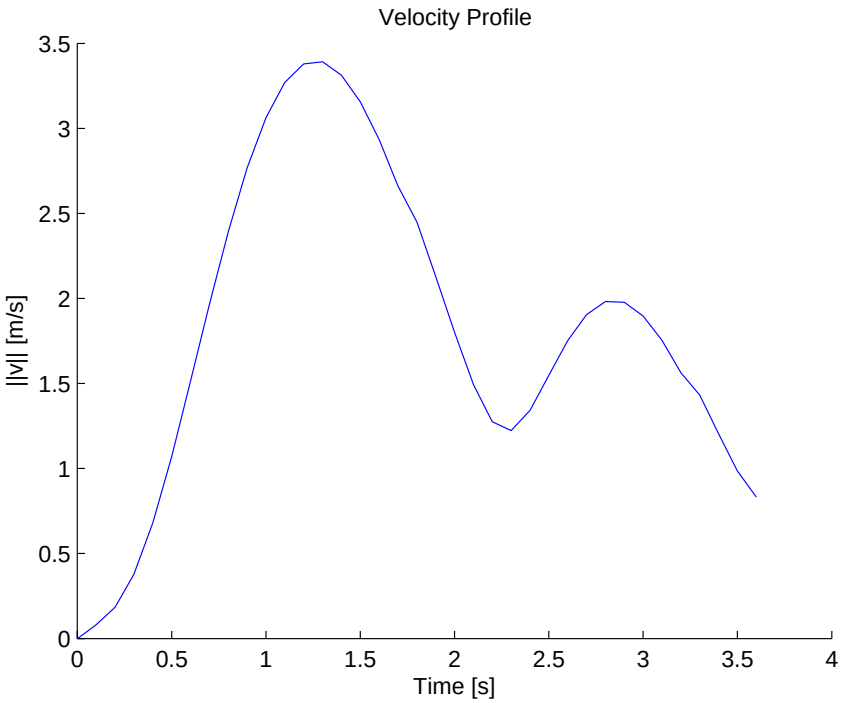
$\boldsymbol{p}_{goal} = [4, 4, 3]$ respectively. Figure 5–6(a) depicts the trajectory in the workspace. It averts the obstacle located at the center of the room while maintaining a smooth trajectory. Figure 5–6(b) shows the speed profile of the trajectory which clearly satifsies the velocity magnitude constraints and is smooth, befitting a physical system. The energy needed to complete this trajectory is $\mathbb{E}_{in} = 2650\,J$ and the trajectory duration is $T_{traj} = 3.7\,s$. It takes, on average, approximately an hour to generate this trajectory. For comparison purposes, the amount of energy needed to maintain the *X8* in steady hover for the same length of time can be computed. Based on power consumption of $P_{hover} = 230\,Watt$, the total energy needed to maintain hover for the duration of the trajectory is $851\,J$.

5.2.2 Single Window Room

A more challenging environment to manoeuvre in is depicted in fig. 5–7 with the MEAQR RRT* optimal trajectory and velocity profile. The initial and target positions queried are $\boldsymbol{p}_{init} = [0.3, 2, 1]$ and $\boldsymbol{p}_{goal} = [4, 4, 2]$ respectively. The energy needed to complete this trajectory is $\mathbb{E}_{in} = 2818\,J$ and the time needed to complete it is $T_{traj} = 3.2\,s$. It takes, on average, approximately an hour to generate this trajectory.

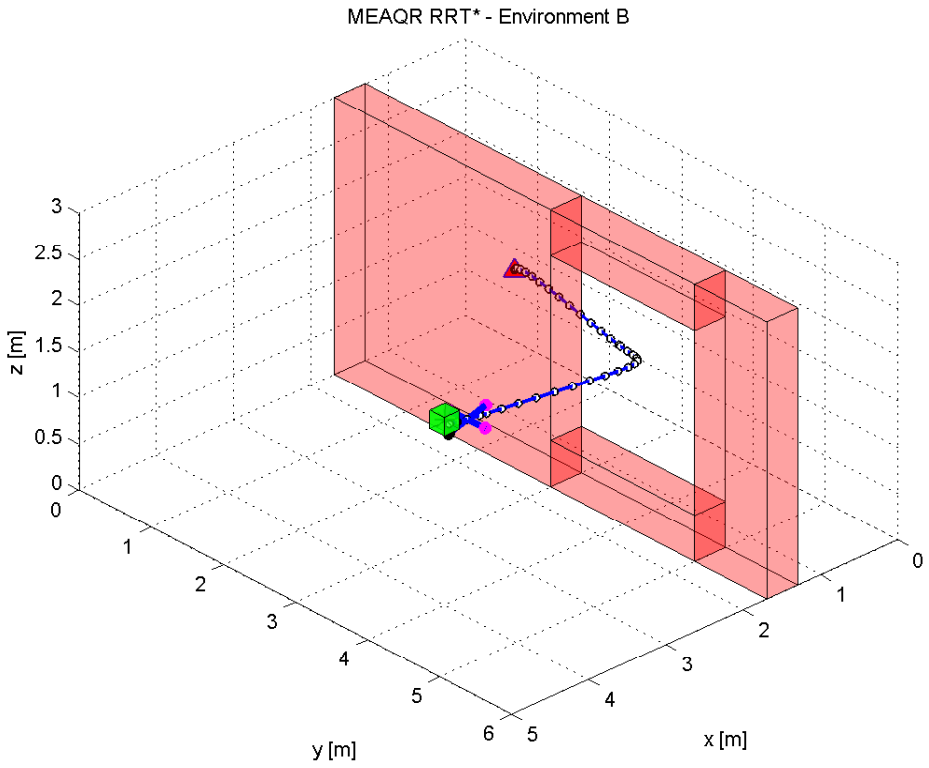


(a) Workspace 3D view

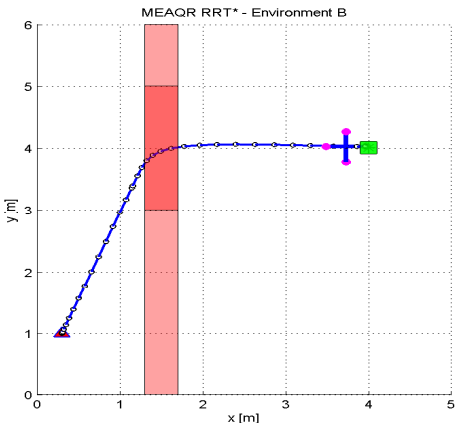


(b) Speed Profile

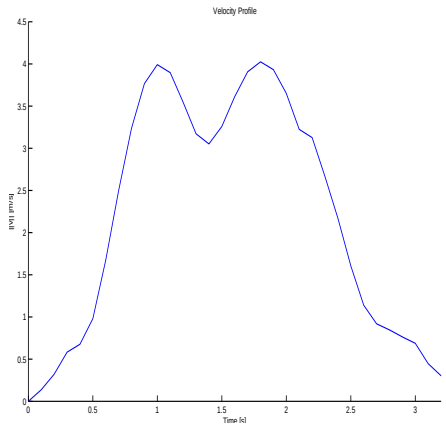
Figure 5–6: The projection of the trajectory on the *3D* workspace and the speed profile for environment A.



(a) Workspace 3D view



(b) Workspace 2D view



(c) Speed Profile

Figure 5–7: The projection of the trajectory on the *3D* workspace and the speed profile for environment B.

Chapter 6
Map Building

In order to apply the motion planner on a real environment, a map representing the workspace is required prior to invoking the planner. In a scenario where a map of the mission environment is not available a-priori, a manual *reconnaissance* flight is conducted prior to the autonomous flight. This chapter presents the different parts of the map building process, the parts of which are depicted in fig. 6–1, that was developed and implemented for the *X8* vehicle and accordingly, makes use of the shortly presented sensors, available on the *X8*. Of course, if the environment map is already available from other sources, the motion planner can be applied directly to it.

In the course of the previous work in AML, conducted as part of a research contract with DRDC-Suffield, the *X8* has been retrofitted with the appropriate sensor suit to allow state estimation, localization and mapping. This sensor suit is comprised of an inertial measurement unit (IMU, available on stock vehicle), an Hokuyo laser range finder (LIDAR)[1] , a sonar sensor and a multi-camera vision system.

The Hokuyo LIDAR, mounted atop the *X8*, continuously scans its region of interest as depicted in fig. 6–2. Each sweep of the plane contains 1080 samples or a circular sector of 270°. Each sample represents a reflected laser ray treated by the system as an obstacle at that point in space. Atop the LIDAR, two down reflecting mirrors are strategically placed to allow measurement of distance to the surface above which the flight is conducted.

[1] `http://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html`

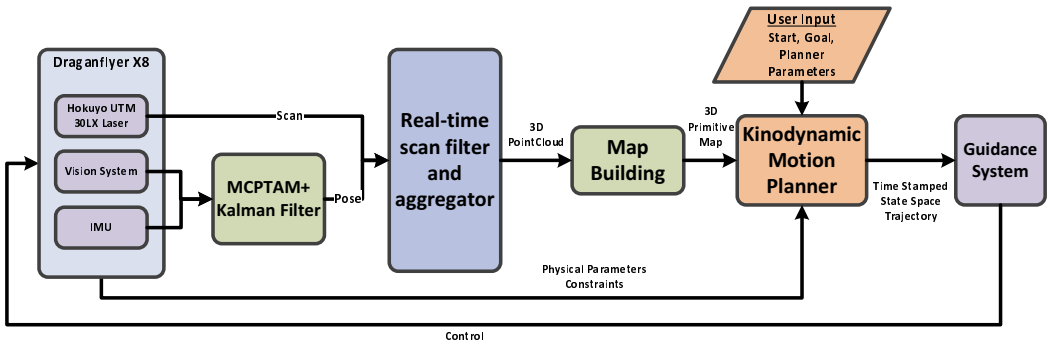


Figure 6–1: The sensor retrofitted *Draganflyer X8* is manually flown in the target environment while laser scans are aggregated and projected to the world frame, segmented, clustered, bounded and exported as a primitive based map for the motion planner.

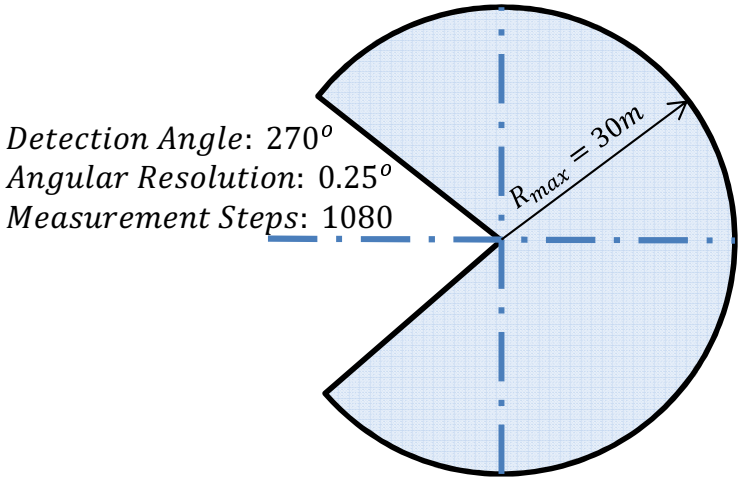


Figure 6–2: Hokuyo LIDAR sampling region. Adapted from product specifications [1]

The multi-camera vision system runs an MCPTAM [23] (by A. Harmat of AML) algorithm which utilizes the features detected by the vision system, fused with IMU data, passed through a Kalman filter, to estimate the pose of the *X8* and to localize it relative to an arbitrary global frame of reference. The pose and location estimates are used in ROS [45] to create a transformation tree (tf tree[2]) relating the different frames of reference used, such as vision,

[2] tf is a ROS package that lets the user keep track of multiple coordinate frames over time. tf maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

sonar, IMU, body and LIDAR to one another. As depicted in fig. 6–1, it is on the basis of this transformation information that the LIDAR aggregated scans are projected to the global frame of reference to create a point cloud. The point cloud is processed using PCL [47] to build the eventually exported primitive based map of the target environment.

6.1 Laser Aggregator

To construct the map of the target environment with the *X8* vehicle, a laser assembler ROS node[3] queues the incoming scans through the following filters:

Shadow Filter Readily available from ROS, this filter removes laser scan samples that are most likely caused by the veiling effect when the edge of an object is being scanned and returns multiple samples from that edge. These redundant samples are removed by the shadow filter in the following manner: Let O denote the origin of the laser scanner and two samples are P_1 and P_2. If the angle $\angle OP_1P_2$ is less than a particular minimum or greater than a particular maximum, all neighbouring samples further away than P_1 are removed. The user defines the minimal and maximal angle, and the number of neighbouring samples that are to be removed. The interested reader is directed to ROS documentation.

Mirror Filter The environment mapping procedure does not attempt to explicitly recognize the ground; thus, the samples associated with the down-reflecting mirrors are filtered. These samples do not hold relevant information for the obstacle registration process.

Rig Obstruction Filter Samples representative of objects closer than a predefined distance (1 meter in this case) are also ignored. The distance

[3] A node is a process or independent routine in ROS that performs a computation.

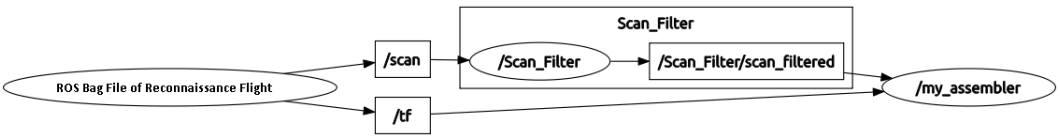


Figure 6–3: As LIDAR scans and the tf tree are published (from the ROS Bag file), the laser assembler and the filter nodes subscribe to those massages. Scan are queued for filtering and are projected using the *tf tree*. Eventually, all transformed and filtered scans are published as a point cloud file for further processing in PCL.

used is one meter since it is not intended for the *X8* to fly in such proximity to obstacles. Moreover, there are physical components on the *X8* which obstruct the laser beam path thereby generating false samples and, should not be mapped.

The filter chain through which the laser scan are passed, is depicted in fig. 6–3. Each scan is associated with a time stamp and so are the transformations in the tf tree. Once filtered, the scans are transformed, by the corresponding transformation from the tf tree, to the world frame. The x, y axes of the world frame are arbitrary in their direction, where the z direction (gravity based) is approximated from the Kalman filtered IMU data. The laser scan assembler aggregates the projected scans together to a $3D$ point cloud representative of the environment.

6.2 Map Creation

The proposed motion planner receives a primitive based environment map as input. To create this representation, the generated point cloud is processed using PCL [47] using both readily available algorithms it offers, as well as custom written ones.

Due to the high sampling rate of the LIDAR, there might be multiple samples in a given location in space. These are regarded as redundant; thus, with the aim of reducing the cloud size, down-sampling is conducted using the

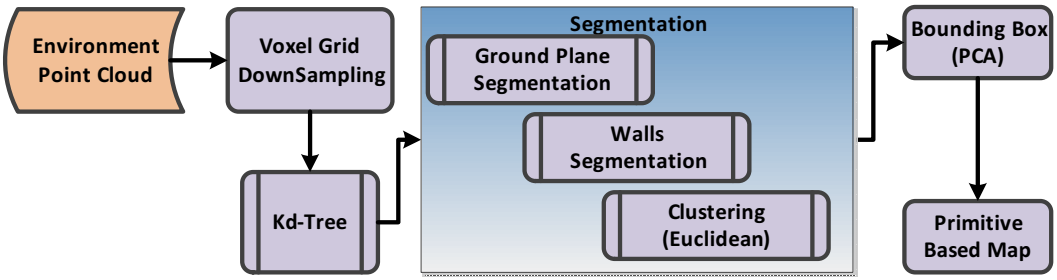


Figure 6–4: A block diagram through which the point cloud is processed as the environment map is created

PCL *voxel filter*[4] . Then, a PCL *Kd Tree* data structure is built to reduce computational complexity of nearest neighbour queries in the following stages.

To identify the obstacle scene, the ground plane is segmented first in order to limit the flight altitude. The ground is assumed to be a plane of the form:

$$n_x x + n_y y + n_z z + D = 0 \tag{6.1}$$

where $\hat{\boldsymbol{n}} = [n_x, n_y, n_z]^T$ is a normal unit vector. The segmentation is done using model based RANSAC[5] [14] assuming an upward pointing normal vector $\hat{\boldsymbol{n}}$ such that:

$$\arccos\left(\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{z}}\right) < 15^\circ$$

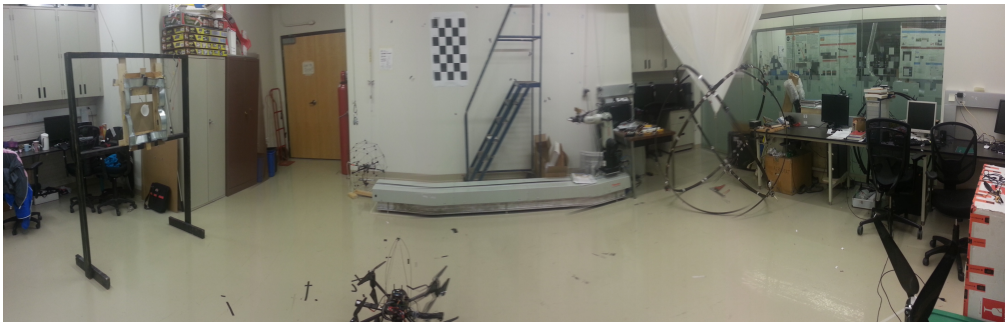
A similar segmentation scheme is utilized to identify the walls (if those exist), also modelled as planes, with a normal vector perpendicular to the ground normal. Lastly, Euclidean clustering is conducted on the remaining points in the cloud. Each cluster cloud is then fitted with the *minimal bounding box* using principal component analysis representative of the space occupied by this obstacle. The resulting bounding boxes are registered and exported to the motion planner. This process is depicted in the block diagram in fig. 6–4.

[4] The space is divided by a grid of user chosen voxel resolution. All points within a voxel are replaced by a single point in the voxel geometric center, otherwise, it is left empty.

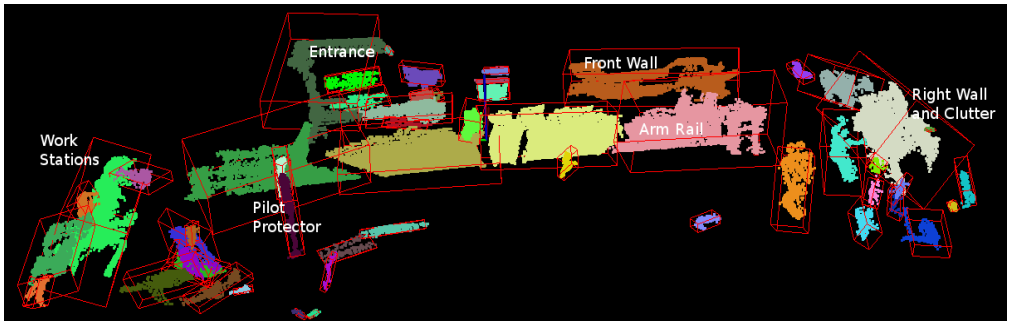
[5] RANdom SAmple Consensus is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers.

6.3 Real World Examples

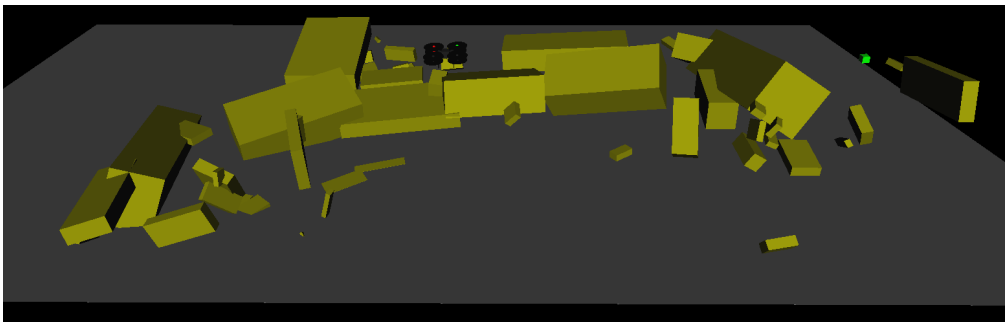
Now that the motion planner and the map building process have been presented, a test for their integration and performance on a real environment is conducted. The McGill Aerospace Mechatronics Laboratory and the McDonald Engineering building were used as testing environments. The *X8* was manually flown in both environments with LIDAR and vision data collected for the map building procedure. The picture of the environment, the point cloud after segmentation, clustering and bounding, and the primitive based ReaK representation are shown in fig. 6–5 for the AML laboratory. Figure 6–6 shows the same for the McDonald Engineering building rooftop.



(a) McGill AML Picture



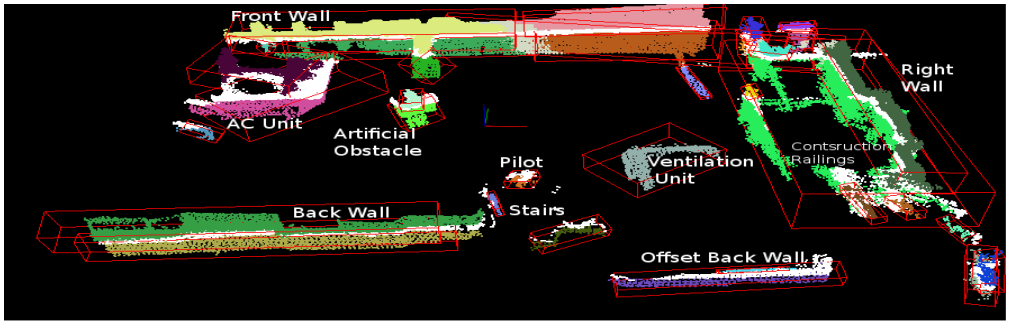
(b) McGill AML in PCL



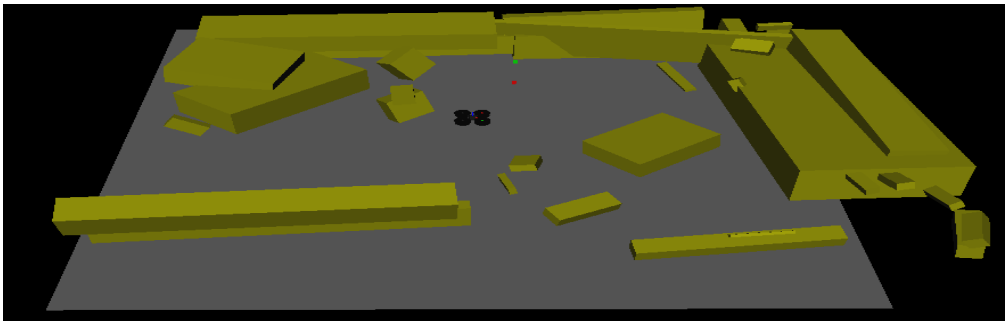
(c) McGill AML in ReaK

Figure 6–5: Map generation of the *McGill Aerospace Mechatronics Laboratory*. (a) A panoramic picture of the lab. Note the point cloud clusters in different colors (b) with their corresponding bounding boxes which are then used as obstacle primitives in ReaK [44] as shown in figure (c)

(a) McGill McDonald Engineering Building Rooftop Picture



(b) McGill McDonald Engineering Building Rooftop in PCL



(c) McGill McDonald Engineering Building Rooftop in ReaK

Figure 6–6: Map generation of the McGill McDonald engineering building (a) A panoramic picture of the roof. Note the point cloud clusters in different colors (b) with their corresponding bounding boxes which are then used as obstacle primitives in ReaK [44] as shown in figure (c)

Chapter 7
Conclusions and Future Directions

This thesis presented a kinodynamic minimum energy RRT* based motion planner-an incremental sampling based approach that extends the RRT* for non-holonomic systems with actuation constraints for a-priori known environments to achieve minimal energy consumption along the generated trajectory while maximizing the utility of the flight envelop. The MEAQR pseudometric for the KNN search and a closed-loop optimal steering method for tree extension for state connections were formulated. The MEAQR pseudometric is representative of the electromechanical energy needed to steer the system. Hence, by minimizing the trajectory cost with respect to this pseudometric, a minimal energy trajectory is obtained. This approach is asymptotically optimal, as it takes advantage of the guaranteed optimality of the RRT* *almost-sure* convergence. The proposed algorithm was tested on two systems: the inverted pendulum, a system with 2D SS representation and the *Draganflyer X8*, an octocopter in an $\times$ configuration with a 13D SS representation and complex actuation constraints. In the following, the approach is summarized, its contributions and limitations are reviewed, and lastly, possible improvements and future research directions are conveyed.

7.1 Summary

Minimum energy. The MEAQR RRT* planner is based on the premise that there exists such an optimal, minimal energy trajectory connecting the initial state to the goal region. It was built on the concatenation of optimal local path segments, yielding a global optimal trajectory, which was mathematically validated for the simple, yet non-linear, and under-powered, inverted pendulum system through *Monte-Carlo* simulations. Moreover, the planner

was demonstrated on the high dimensional, under-actuated, non-holonomic *Draganflayer X8* system, by generating smooth, feasible, energy optimal trajectories. The cost functional formulated for the linearised SS used as a pseudometric represents the local approximation of the global objective to be minimized, which is the total energy consumed along the trajectory generated, resulting in a minimal energy trajectory. In systems where this pseudometric misrepresents the energy costs, the convergence rate of the proposed algorithm to the minimal energy trajectory shall be substantially reduced.

Flight envelope and actuation constraints. The planner maximizes the utility of the flight envelope, while ensuring the feasibility of the trajectory by maintaining the actuation signals used to steer the systems dynamic model forward in the tree extensions step, within their physical bounds.

Complex local planner. Most other motion planning approaches are concerned only with collision detection and goal reaching on a simplified dynamic model. Moreover, the accepted approach for probabilistic motion planning (roadmap building) is based on the simplicity of the local planner for configuration (or state) space connections. The MEAQR RRT* planner, in contrast to these approaches, uses a complex local planner and pseudometric, with high computational complexity involving long running times for trajectory generation, rendering this method not feasible for real-time application, given present day computing ability.

Environment modelling. The proposed planner assumes *a-priori* knowledge of the static environment in which the autonomous flight shall be conducted. To test the proposed planner on an actual vehicle, a map building procedure was developed in ROS and PCL utilizing the *X8* sensor suit. A primitive based environment representation was created using a set of segmentation, clustering and bounding routines.

7.2 Improvements and Future Directions

7.2.1 Algorithmic Improvements

Sampling scheme. A wisely chosen alternative sampling scheme, such as goal centred Gaussian, shall shorten the elapsed time till an initial solution is available and increase the convergence rate of the algorithm to the optimal solution. However, the planner may get stuck in a local minima under unfortunate circumstances, depending on the environment configuration. Another option is a dynamic sampling region. Regions that are already explored are no longer of interest to steer towards. Mapping the reachability region of the tree with respect to the metric, similar to [51], and sampling outside that region, reduces redundant sampling and steering towards areas which are already within the RRTs reach.

Increasing the convergence rate. Once a solution is found, and the planner is allowed to continue running and improve on that generated trajectory, sampling about that solution shall contribute to the convergence to the optimal trajectory. However, this restricts the eventual trajectory to a specific homotopic class of trajectories.

Reducing overall running time. In this work, the *Runge-Kutta* integrator was found to be the most computationally efficient (least running time) among those offered by both Matlab© and ReaK library. However, energy balanced integrators (symplectic [9] or in particular Verlet [22]) which are used in orbital trajectories or Hamiltonian mechanical systems, shall be more useful in planners where the actuation is explicitly known, such as the one presented in this thesis. Those are based on conservation of either energy or momentum or a known incremental change in those quantities.

7.2.2 Future Work

The area of operating autonomous aerial vehicles shall continue to develop. There are many future research directions in this field from which the MEAQR RRT* planner can benefit.

Bi-Directional RRT. Simultaneously growing an RRT from the initial as well as the goal state substantially shortens the search time till an initial solution is available and reduces the chance of converging to local minimas. A Bi-Directional RRT approach is possible if the MEAQR can be formulated to represent the cost connecting two states backwards in time.

Energy Functional. The pseudometric this thesis makes use of is based on the fact that the electromechanical energy is proportional to the L_2-norm of the actuation. Incorporating a machine-learning approach to create a more accurate mapping from state space trajectories to their energy consumption profiles will enable the planner to reach even closer to the true global minimum energy trajectory. It will also enable lower dimensional planning, representing the kinematics of the vehicle only, since each segment can be mapped to its energy consumption profile.

Planning with a kinematic system representation. The framework employed in the proposed planner, is equivalent to solving for a PID controller (optimal control) for the system as many time as there are connections in the tree. If the closed-loop response of the system is known (after the controller design), planning in the kinematic space as opposed to the open loop dynamic space, shall prove to be much faster. To pursue this, a mapping of an initial state and a reference signal to the state space is needed, enabling collision detection and cost evaluation. This method can maintain its energy considerations if trajectory segments can be efficiently mapped to their energy consumption costs.

Dynamic Environment. Though operation in a static environment was assumed in this thesis, it is imperative that in future work, other aerial or ground vehicles during low altitude flight shall be accounted for as obstacles.

Larger Mission Spaces. Mapping data structures used in this thesis are sufficient for very small missions (rooms under 500_{m^3} in volume). However, more efficient representations and algorithms are required to plan a trajectory in a possible long range mission.

Trajectory Following Controllers. The appropriate trajectory following controllers, shall be used to demonstrate the generated trajectory on the *X8*. Such controllers can be developed with any one of existing traditional techniques such as LQR or virtual target following.

Appendix A
Draganfly X8 State Space Model

The *Draganfly X8* state space model fits the state derivative formulation:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{A.1}$$

Where the state is defined as the *13D* vector:

$$\boldsymbol{x}(t) = [\boldsymbol{p}(t), \boldsymbol{v}(t), \boldsymbol{q}(t), \boldsymbol{\omega}(t)] \tag{A.2}$$

$\boldsymbol{p}(t) = [p_x, p_y, p_z]^T$	Position for the vehicle center of mass in the global frame of reference
$\boldsymbol{q}(t) = [q_0, q_1, q_2, q_3]^T$	Unit quaternion representing the rotation of the body fixed frame in $SO(3)$, with q_0 being the real part
$\boldsymbol{v}(t) = [v_x, v_y, v_z]^T$	Linear velocity in the global frame of reference
$\boldsymbol{\omega}(t) = [\omega_x, \omega_y, \omega_z]^T$	Angular velocity

The equations of motion follow:

$$\begin{bmatrix} \dot{\boldsymbol{p}}(t) \\ \dot{\boldsymbol{v}}(t) \\ \dot{\boldsymbol{q}}(t) \\ \dot{\boldsymbol{\omega}}(t) \end{bmatrix} = \begin{pmatrix} \boldsymbol{v}^I(t) \\ \boldsymbol{g}^I - \frac{R(\boldsymbol{q})}{\mathbf{m}} \left(C_{D_v}(R(\boldsymbol{q})^T \boldsymbol{v}^I(t))) \| R(\boldsymbol{q})^T \boldsymbol{v}^I(t) \| + \mathcal{F}^B \right) \\ \frac{1}{2} \boldsymbol{q}(t) \cdot \hat{\boldsymbol{\omega}}^B(t) \\ \mathbb{J}^{-1} \left(\boldsymbol{\tau}^B(t) - \boldsymbol{\omega}^B(t) \times \mathbb{J} \boldsymbol{\omega}^B(t) - C_{D_\omega} \boldsymbol{\omega}^B(t) \| \boldsymbol{\omega}^B(t) \| \right) \end{pmatrix} \tag{A.3}$$

where $\hat{\boldsymbol{\omega}}(t) = [0, \omega_x, \omega_y, \omega_z]^T$. The rotation matrix $R(\boldsymbol{q})$ and its transpose $R(\boldsymbol{q})^T$ are obtained by converting the unit quaternion $\boldsymbol{q}(t)$ to its equivalent

matrix representation. Since the unit quaternion constraint is enforced, the SS is equivalent to *12D* SS. $\mathcal{F}$ and $\boldsymbol{\tau}$ are the actuation forces and torques. $\boldsymbol{g}^I = [0, 0, g]^T$ is the gravity vector. Note that the superscript I and B refers to vector representation in the global and body fixed frame of references respectively. C_{D_v} and C_{D_ω} are translational and rotational aerodynamic drag coefficient matrices. Both are $\mathbb{R}^{3\times3}$ and assumed to be diagonal to simply model body cross-sectional drag.

Due to symmetry the inertia matrix $\mathbb{J}$ is assumed to be diagonal. The translational and rotational drag coefficients are assumed to be located on the diagonal as well. The numeric values of the parameters used in the model are:

Table A–1: *X8* Physical Parameters

Parameters	Units	Value
$\mathbf{m}$	kg	2.025
$\mathbb{J}_{11} = \mathbb{J}_{22}$	kgm^2	0.0613
$\mathbb{J}_{33}$	kgm^2	0.1115
C_{D_v}	kg/m	$0.5I_{3\times3}$
C_{D_ω}	kg/m	$0.5I_{3\times3}$

The non-holonomic constraints presented here without aerodynamic drag with Euler angles for readability:

$$\tan(\theta) = \frac{\ddot{x}\cos\psi + \ddot{y}\sin\psi}{\ddot{z} - g} \tag{A.4a}$$

$$\sin(\phi) = \frac{-\ddot{x}\sin\psi + \ddot{y}\cos\psi}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} - g)^2}} \tag{A.4b}$$

References

[1] F. Andert, F. Adolf, L. Goormann, and J. Dittrich. Mapping and path planning in complex environments: An obstacle avoidance approach for an unmanned helicopter. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 745 –750, may 2011.

[2] M. Athans and P.L. Falb. *Optimal control: an introduction to the theory and its applications*. McGraw-Hill New York, 1966.

[3] Kostas E Bekris, Brian Y Chen, Andrew M Ladd, Erion Plaku, and Lydia E Kavraki. Multiple query probabilistic roadmap planning using single query planning primitives. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 656–661. IEEE, 2003.

[4] R. Bellman. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences of the United States of America*, 42(10):767, 1956.

[5] Y. Bouktir, M. Haddad, and T. Chettibi. Trajectory planning for a quadrotor helicopter. In *Control and Automation, 2008 16th Mediterranean Conference on*, pages 1258 –1263, june 2008.

[6] Michael S Branicky, Michael M Curtiss, Joshua A Levine, and Stuart B Morgan. Rrts for nonlinear, discrete, and hybrid planning and control. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1, pages 657–663. IEEE, 2003.

[7] James Bruce and Manuela Veloso. Real-time randomized path planning for robot navigation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2383–2388. IEEE, 2002.

[8] F. Callier and J. Willems. Criterion for the convergence of the solution of the riccati differential equation. *Automatic Control, IEEE Transactions on*, 26(6):1232–1242, 1981.

[9] P J Channell and C Scovel. Symplectic integration of hamiltonian systems. *Nonlinearity*, 3(2):231, 1990.

[10] Peng Cheng and S.M. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 43 –48 vol.1, 2001.

[11] Peng Cheng and Steven M LaValle. Resolution complete rapidly-exploring random trees. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 267–272. IEEE, 2002.

[12] MJ de Smith. *Distance and Path: The Development, Interpretation and Application of Distance Measurement in Mapping and Modelling*. PhD thesis, PHD THESIS, University College, 2003.

[13] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[14] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[15] M.C. FRANK, J. WINKIN, and L.W. JACQUES. Convergence of the time-invariant riccati differential equation and lq-problem: mechanisms of attraction. *International Journal of Control*, 59(4):983–1000, 1994.

[16] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[17] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[18] Ada Wai-chee Fu, Polly Mei-shuen Chan, Yin-Ling Cheung, and Yiu Sang Moon. Dynamic vp-tree indexing for n-nearest neighbor search given pairwise distances. *The VLDB Journal—The International Journal on Very Large Data Bases*, 9(2):154–173, 2000.

[19] V. Garber. Optimum intercept laws for accelerating targets. *AIAA Journal*, 6:2196–2198, November 1968.

[20] E. Glassman and R. Tedrake. A quadratic regulator-based heuristic for rapidly exploring state space. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5021–5028. IEEE, 2010.

[21] Jared Go, Thuc D Vu, and James J Kuffner. Autonomous behaviors for interactive vehicle animations. *Graphical models*, 68(2):90–112, 2006.

[22] E. Hairer, C. Lubich, G. Wanner, et al. Geometric numerical integration illustrated by the stormer-verlet method. *Acta Numerica*, 12:399–450, 2003.

[23] Adam Harmat, Inna Sharf, and Michael Trentini. Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle. In *Intelligent Robotics and Applications*, pages 421–432. Springer, 2012.

[24] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[25] Satoshi Kagami, J Kuffner, K Nishiwaki, K Okada, M Inaba, and H Inoue. Humanoid arm motion planning using stereo vision and rrt search. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2167–2172. IEEE, 2003.

[26] M Kallman and Maja Mataric. Motion planning using dynamic roadmaps. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4399–4404. IEEE, 2004.

[27] Marcelo Kallmann, Amaury Aubel, Tolga Abaci, and Daniel Thalmann. Planning collision-free reaching motions for interactive object manipulation and grasping. In *Computer Graphics Forum*, volume 22, pages 313–322. Wiley Online Library, 2003.

[28] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, June 2011.

[29] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics & Automation*, 12(4):566–580, June 1996.

[30] L.E. Kavraki, M.N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 4, pages 3020 –3025 vol.4, apr 1996.

[31] Jongwoo Kim and James P Ostrowski. Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2200–2205. IEEE, 2003.

[32] E. Koyuncu and G. Inalhan. A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3d environments. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 815 –821, sept. 2008.

[33] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[34] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[35] S.M. LaValle and Jr. Kuffner, J.J. Randomized kinodynamic planning. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 473 –479 vol.1, 1999.

[36] Frank L. Lewis, Draguna Vrabie, and Vassilis L. Syrmos. *Optimal Control*. Wiley, 1995.

[37] Stephen R Lindemann and Steven M LaValle. Incrementally reducing dispersion by increasing voronoi bias in rrts. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3251–3257. IEEE, 2004.

[38] Stephen R Lindemann and Steven M LaValle. Steps toward derandomizing rrts. In *Robot Motion and Control, 2004. RoMoCo'04. Proceedings of the Fourth International Workshop on*, pages 271–277. IEEE, 2004.

[39] Rikovitch N. Motion planning methods survery, 2012.

[40] Rikovitch N. Uav motion planning survey, 2012.

[41] Stephen Malvern Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[42] PooGyeon Park and T. Kailath. Convergence of the dre solution to the are strong solution. *Automatic Control, IEEE Transactions on*, 42(4):573 –578, apr 1997.

[43] A. Perez, R. Platt, G.D. Konidaris, L.P. Kaelbling, and T. Lozano-Perez. Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2537–2542, May 2012.

[44] Sven Mikael Persson. GTS: GNU ReaK C++ library. `https://github.com/mikael-s-persson/ReaK`, 2011.

[45] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[46] N. Rikovitch and I. Sharf. Kinodynamic motion planning for uavs: A minimum energy approach. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Boston, MA, USA, August 2013 (AIAA-2013-1662944).

[47] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[48] Micha Sharir. Jack schwartz and robotics: The roaring eighties. In *From Linear Operators to Computational Biology*, pages 87–104. Springer, 2013.

[49] GW Stewart. *AFTERNOTES*. SIAM.

[50] Morten Strandberg. Augmenting rrt-planners with local trees. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3258–3262. IEEE, 2004.

[51] R. Tedrake. Lqr-trees: Feedback motion planning on sparse randomized trees. 2009.

[52] Chris Urmson and Reid Simmons. Approaches for heuristically biasing rrt growth. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1178–1183. IEEE, 2003.

[53] Dustin J. Webb and Jur van den Berg. Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints. *CoRR*, abs/1205.5088, 2012.

[54] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, pages 194–205, 1998.

[55] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[56] Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 311–321. Society for Industrial and Applied Mathematics, 1993.