Kinodynamic Motion Planning for UAVs: A Minimum Energy Approach

Nir Rikovitch* and Inna Sharf†

McGill University, Montreal, Quebec, H3A 2T5, Canada

We present an optimal kinodynamic rapidly exploring random tree, a single query incremental sampling based optimal motion planner for robots with non-linear dynamics, differential constraints and actuation limitations. Our work extends the algorithms presented previously by formulating a fixed-final-state-free-final-time open loop configuration space metric for nearest neighbours search and the appropriate closed loop feedback controller for the tree extension heuristic, allowing us to introduce constraints on actuation magnitude and bandwidth. The controller is formulated by minimizing the amount of energy used to connect two states whereas the trade off is between total trajectory time and weighted actuation norm. We demonstrate the algorithm on (1) a simple 2D pendulum with actuation constraints and (2) a quad rotor 13D model.

Nomenclature

f	State derivative function	c	Cost functional
$\boldsymbol{\tau}$	Torque, Nm	T_{max}	Maximal search horizon for near neighbours, s
$\boldsymbol{x}$	State vector	Q	Quadratic regulator state weight matrix
$\dot{\boldsymbol{x}}$	State vector derivative	R	Quadratic regulator control weight matrix
$\boldsymbol{u}$	Control vector	ρ	Quadratic regulator time weight scalar
d	State space dimension	$\mathbf{T}$	Tree graph
d_u	Control space dimension	σ	Trajectory in state space
A	Jacobian matrix w.r.t the state	σ_i	Trajectory segment i
B	Jacobian matrix w.r.t the control	N	Number of vertices states in tree
C	State derivative value (drift)	$\succ$	Matrix - positive/negative definite
$\boldsymbol{\lambda}$	*Lagrange* multiplier vector	$\succeq$	Matrix - positive/negative semi definite
$\mathbf{m}$	Mass, kg	*Superscript*	
$\mathbb{J}$	Tensor of inertia , kgm^2	$*$	Optimal value
C_{D_v}	Translational drag coefficient matrix, $N(m/s)^{-2}$		
C_{D_ω}	Rotational drag coefficient matrix, $N(m/s)^{-2}$		

I. Introduction

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. The interest is to move the human from the classical position of *in-the-loop*, while actively operating an unmanned system, to a more supervisory position *on-the-loop*. This will allow human-robot teams to better perform by liberating the human from constant online operation to supervisory position, which is to oversee 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 hover-craft UAVs in an urban environment is yet to be conquered. The solution of flying such vehicles above the building line using a

*Graduate Student, Department of Mechanical Engineering, AIAA Student Member

†Associate Professor, Department of Mechanical Engineering, AIAA Member

straight 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 obstacles and proximity to the 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, our research group at McGill Aerospace Mechatronics Lab, as part of a research contract with DRDC-Suffield, has retrofitted a quadrotor UAV *Draganfly X8*[a] (X8 hereafter) with the appropriate sensor suit to allow state estimation, localization and mapping - eventually aimed at autonomous flight. 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. This has led us to consider the energy (control effort) cost functional to minimize along the trajectory generated. We propose an energy optimal kinodyamic sampling based trajectory generator for actuator-limited non-linear systems, specifically the quadrotor.

The field of robotic motion planning is a heavily researched field for the reasons alluded to earlier. The majority of existing motion planners were developed for holonomic systems and tend to be reliable and fast for such systems. However, as later discussed in section II, solving the general motion planning problem is PSPACE-hard. This has led us to consider the probabilistic motion planners which are currently considered the *state-of-the-art*. The probabilistic motion planners can deal with the differential constraints, linking the derivatives of the states to one another, which arise arise in systems with fewer actuators than degrees of freedom. With the view to developing a motion planner for the X8, we have chosen the rapidly exploring random trees algorithm (RRT) due to its ability to quickly generate a solution while not requiring the connecting method to be symmetric - hence, it does not need an actual metric. To generate an optimal trajectory with respect to a cost functional, the algorithmic variant RRT* is an even better fit.

The RRT* is an asymptotic optimality variant of the classical RRT. It differs from the RRT by its ability to rewire the tree in an optimal manner (with respect to the cost functional defined in the state space) and optimally choosing the best parent for each newly generated state. This yields asymptotically optimal trajectories.[3] In applying the RRT* to a quadrotor, we had to address two domain-specific components of the algorithm: the distance metric and the extension heuristic. Early versions of the sampling based motion planners [4, 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 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.[5] Glassman and Tedrake[2] derived a pseudometric based on an affine quadratic regulator for a regular RRT. Webb and Berg[1] 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.

We present a distance pseudometric based on minimum energy affine quadratic regulator (MEAQR) similar to the work by Glassman and Tedrake, and Webb and Berg. As in the aforementioned references, we linearise the system dynamics about each sampled state and use MEAQR pseudometric to find which vertex in the tree is to be extended while simultaneously computing the optimal feedback controller/steering policy to actually connect the two states. This allows us to introduce other dynamics constraints, such as, actuation magnitude and bandwidth limits which are always present in real life systems. Due to those constraints, the actual state at which the extension step ends might not be the target state. The results are presented for two non linear systems of different SS dimensionality: (i) a simple pendulum with actuator limits and (ii) a quadrotor model based on the X8.

The remainder of this paper is organized as follows. In section II, we explore the related literature, followed by section III where we define the particulars of the motion planning problem for a general dynamic system. In Section IV we briefly present the sampling-based RRT* motion planning framework used as the backbone for the proposed pseudometric in section V. Section VI we elaborate on how this pseudometric is incorporated into an optimal sampling based motion planner. Then, simulation results for the two chosen robotic systems are presented in section VII. Finally, section VIII summarizes the simulation results and suggests the conclusions.

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

II. Literature Review

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.[4] In efforts to solve the problem for practical applications, much research has been devoted to probabilistic motion planners, also known as sampling based planners, for instance, the rapidly exploring random trees (RRT)[5] and probabilistic road map (PRM).[6] Both of these methods have been proven to be probabilistically complete - the probability of generating a solution (if such exists) approaches one as the number of samples approaches infinity. In fact, the probability of not finding a solution rapidly decays to zero.[5,7] While the PRM requires the metric used on the space to be a true metric[b], the RRT algorithm does not. This allows for more options to be considered for both the metric and the connecting scheme with the tree. The RRT is efficient in finding an initial solution, although it may contain unnecessary manoeuvres/actuations which are far from optimal. To ameliorate this side-effect, trajectory post-processing is usually conducted by, for example, smoothing[8] or spline fitting[9,10] the solution.

To achieve optimal trajectories the RRT* algorithm[3] was introduced as an extension of RRT that guarantees *almost-sure* 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,[4] a limitation of these probabilistic motion planners is their reliance on the ability to connect two distinct states with a trajectory (optimal at times). For holonomic robots, feasible trajectories might be as simple as straight lines within the SS connecting the two points. However, for kinodymanic systems, or robots with non-holonomic constraints, these straight lines are not feasible trajectories since they do not adhere to the system dynamics. Finding a feasible trajectory that fulfils the differential constraints of the system represents a *two-point boundary value problem*,[4] and it is far from trivial to solve. Well-known methods, such as the shooting method[5] have been employed to solve the problem, however, they result in solutions that are non optimal. Recently, Perez et al. suggested an infinite horizon linear quadratic regulator to be used as a connection method between two states,[11] though the actual extension suffers from two drawbacks; First, the state towards which we extand the tree might not be controllable in which case the algebraic Riccati equation would have no solution (the weighted controllability grammian does not converge). Secondly, the infinite horizon controller yields an optimal solution only if the time approaches infinity - whereas the actual extension step time length is not. This causes the suggested extension heuristic to under-perform, yielding a sub-optimal trajectory.

A key component which has a substantial affect on the convergence of the RRT is the distance metric used. An efficient exploration of the SS is obtained only once the metric reflects the true *cost-to-go*.[12] The *cost-to-go* can be based on optimal control methods for linearised systems.[5] A linear quadratic regulator pseudometric was suggested by Perez et al.[11] and Glassman and Tedrake.[2] The aforementioned references deal with differential constraints and non linear systems, but they do not take into account the actuation magnitude and bandwidth limitations. Moreover, Glassman and Tedrake's work deals with sparse RRTs with reachability regions and does not demonstrate practical applications on real high dimensional systems.

To improve on the previous work, Webb and Berg introduced a fixed-final-state-free-final-time method to be used as the local planner (both the metric and extension heuristic). Unfortunately, their work does not take into account 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 here. Thus, our method improved on previous work by further permitting us to remove restrictive assumptions on the system.

III. Problem Formulation

Let $\boldsymbol{X} \subseteq \mathbb{R}^d$ be the *state space* of dimensionality d where the elements of $\boldsymbol{X}$ are known as *states* denoted as $\boldsymbol{x}$. Let $\boldsymbol{U} \subseteq \mathbb{R}^{d_u}$ be the *control space*. Let $\boldsymbol{X}_{obs}, \boldsymbol{X}_{goal} \subseteq \boldsymbol{X}$ be known as the *obstacle space* and *goal region* respectively. The space defined as $\boldsymbol{X}_{free} \triangleq \boldsymbol{X} \backslash \boldsymbol{X}_{obs}$ is the *free space*. A *path* σ in $\boldsymbol{X}$ is a continuous function $\sigma \colon [0,1] \to \boldsymbol{X} \times \mathcal{T}$. It is said to be free if $\sigma(\tau) \in \boldsymbol{X}_{free} \quad \forall \quad \tau \in [0,1]$. Let τ be non dimensional time and $\mathcal{T}$ be the time space.

[b] A true metric $\mathcal{D}$ on a space $\boldsymbol{X}$ where two elements $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}$ must be non-negative $\mathcal{D}(\boldsymbol{x}, \boldsymbol{y}) \geq 0$, symmetric $\mathcal{D}(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{D}(\boldsymbol{y}, \boldsymbol{x})$,follow the triangle inequality $\mathcal{D}(\boldsymbol{x}, \boldsymbol{z}) \leq \mathcal{D}(\boldsymbol{x}, \boldsymbol{y}) + \mathcal{D}(\boldsymbol{y}, \boldsymbol{z})$ and adhere to the identity of indiscernible $\mathcal{D}(\boldsymbol{x}, \boldsymbol{y}) = 0$ iff $\boldsymbol{x} = \boldsymbol{y}$

Given an initial state $\boldsymbol{x}_{init}$, an environment with obstacles $\boldsymbol{X}_{obs}$ and a goal region $\boldsymbol{X}_{goal}$, the motion planning problem deals with finding a feasible collision-free path $\sigma(\tau) : [0,1] \rightarrow \boldsymbol{X}_{free}$ such that $\sigma(0) = \boldsymbol{x}_{init}$ and $\sigma(1) \in \boldsymbol{X}_{goal}$. The set of possible paths in $\boldsymbol{X}_{free}$ will be denoted as Σ_{free}.

The feasibility of the trajectory σ is determined by not only its existence in $\boldsymbol{X}_{free}$ but also governed by the non linear SS dynamics described by eq. (1) so that $\dot{\sigma}(\tau) = f(\sigma(\tau), u(\tau)) \forall \tau \in [0,1]$:

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

Let $c\colon \Sigma_{free} \rightarrow \mathcal{R}^+$ be a *cost functional* that maps a candidate path $\hat{\sigma} \in \Sigma$ 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 ; $c(\sigma^*) = \inf_{\hat{\sigma} \in \Sigma_{free}} c(\hat{\sigma})$. The optimal path can be represented as a once differentiable function C^1 although for actuation feasibility we shall require C^2. Let $\boldsymbol{\pi}$ be a time sequence of control inputs we shall call policy. The optimal control policy is the sequence yielding the minimal cost value with respect to the defined global functional c, i.e., $\boldsymbol{\pi}^* = \underset{\pi}{\operatorname{argmin}}\ c(\boldsymbol{\pi}, \boldsymbol{x}_{init}, \boldsymbol{X}_{goal})$.

IV. The Optimal Probabilistic Motion Planner - RRT*

This section briefly introduces the well-established RRT* algorithm, which is the framework for the proposed planner, with generic algorithmic components for the sake of simplicity.

Initially presented by Kuffner and LaValle,[5] RRT is an incremental sampling based single query motion planner for holonomic systems. Its optimal variant, the RRT*, in addition to the basic building blocks derived from the RRT, has two additional routines (**ChooseParent** and **Rewire** adapted from Perez et al.[11]) which make it optimal, all of which are outlined below. Pseudocode for the algorithm can be found in algorithm A.1 and the reader can consult existing literature for further details.

Sample: Returns a sampled state $\boldsymbol{x} \in \boldsymbol{X}_{free}$ from a random distribution.

Steer: Connects two states $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \boldsymbol{X}_{free}$. Returns either a boolean indicating connection success and/or a path segment $\sigma \in \boldsymbol{X}$ such that $\sigma(0) = \boldsymbol{x}_1$ but not necessarily terminating at $\boldsymbol{x}_2$.

Collision Check: For a path segment or state this procedure returns a boolean true iff $\sigma(t) \in \boldsymbol{X}_{free}\ \forall\ t$ or $\boldsymbol{x} \in \boldsymbol{X}_{free}$[a].

Metric: A scalar function $\mathcal{D} : (\boldsymbol{x}_1, \boldsymbol{x}_2) \rightarrow \mathbb{R}^{\geq 0}$ which stands for the "distance" in whichever sense the metric has been defined. If the two states are not connectible, the metric returns an infinite distance.

K Nearest Neighbours (KNN): Given a subset of states $\mathcal{V}$ and a state $\boldsymbol{x}$, this routine 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. Pseudocode for KNN is given in algorithm A.2.

ChooseParent: For a state $\boldsymbol{x}$ and neighbouring states $Neighbours$ we attempt to steer a candidate $Neighbour_i$ towards $\boldsymbol{x}$. $Neighbour_i$ via which reaching $\boldsymbol{x}$ has the least cost, is designated as the chosen parent. Thus, each new state $\boldsymbol{x}_{new}$ reached, is guaranteed to have its parent state in the tree to be the one with the least incurred cost with respect to the metric defined on the SS. The pseudocode for this procedure is algorithm A.3.

Rewire: Steers $\boldsymbol{x}$ towards a candidate $Neighbour_i \in Neighbours$. If they are connected and the cost to reach the $Neighbour_i$ via $\boldsymbol{x}$ is less than the cost to reach $Neighbour_i$ via its current parent, the edge connecting $Neighbour_i$ to its parent is removed and reconnect it to $\boldsymbol{x}$. This ensures that the connections made are always improving with respect to the cost. The pseudocode for this procedure is algorithm A.4.

[a] Collision checking is done during the steering routine every few integration steps, depending on a predefined SS resolution. Meaning, once the state has been propagated some workspace "distance", the collision check is invoked. If collision is detected, the integration is terminated and the trajectory is truncated of the invalid part.

V. Minimum Energy Affine Quadratic Regulator

This section 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.

A. Affine Quadratic Regulator: Preliminaries

As one of the barriers to autonomous systems' capabilities is the limited mission endurance due to short battery life, as is the case for small UAVs, we choose a cost 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,[13] thus we consider the following objective function:

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

As alluded in section III, computing the optimal control policy $\boldsymbol{\pi}^*$ represents a global optimization problem. Each RRT* path segment 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. Hence, for a non-linear system such as in eq. (1), we reformulate the cost about a nominal point $\boldsymbol{u} = \boldsymbol{u}_0 + \hat{\boldsymbol{u}}$, $\boldsymbol{x} = \boldsymbol{x}_0 + \hat{\boldsymbol{x}}$ to yield:

$$c = \sum_{i=1}^{n} \int_0^{T_i} \frac{1}{2}(\boldsymbol{u}_0 + \hat{\boldsymbol{u}})^T R(\boldsymbol{u}_0 + \hat{\boldsymbol{u}})dt = \sum_{i=1}^{n} \int_0^{T_i} \frac{1}{2}[{\boldsymbol{u}_0}^T R\boldsymbol{u}_0 + 2{\boldsymbol{u}_0}^T R\hat{\boldsymbol{u}} + \hat{\boldsymbol{u}}^T R\hat{\boldsymbol{u}}]dt \tag{3}$$

where $\boldsymbol{x}_0$ is an arbitrary state and $\boldsymbol{u}_0$ is assumed zero, dealt with later in section C. A segment i of eq. (3) can then be approximated as eq. (4):

$$c_i = \int_0^{T_i} \rho_i + \frac{1}{2}\hat{\boldsymbol{u}}_i^T R\hat{\boldsymbol{u}}_i dt \;\; ; \;\; \rho_i \in \mathbb{R}^+, \;\; R \succ 0 \tag{4}$$

Following the *Bellman principle of optimality*,[14] the optimality of all segments shall guarantee the optimality of the entire trajectory. Therefore, our local planning problem in the linear state-space, 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 $\sigma_i(1) = \boldsymbol{x}_2$, while incurring minimal cost with respect to the objective function in eq. (4). Omitting the segment index i as we all the $\hat{()}$ notation, the local planning problem is stated as:

$$\mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \min_u \left\{ \int_0^T [\rho + \frac{1}{2}\boldsymbol{u}^T R\boldsymbol{u}]dt \right\} \;\; ; \;\; R = R^T \succ 0, \;\; \rho \in \mathbb{R}^+ \tag{5a}$$

$$\text{subject to} \quad \dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} + C \tag{5b}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_1 \tag{5c}$$

$$\boldsymbol{x}(T) = \boldsymbol{x}_2 \tag{5d}$$

The minimization of J, the argument of eq. (5), is used as a state-space distance pseudometric $\mathcal{D}$. It is a pseudometric because it violates the symmetry property of a true metric [b] and adheres to the triangle inequality only in the linearised state-space. This pseudometric will be referred to as the minimum energy affine quadratic regulator pseudometric (MEAQR) hereafter.

The optimal feedback control law is derived as per Pontryagin principle of optimality. Defining the Hamiltonian for the optimization problem in eq. (5):

$$H(\boldsymbol{\lambda}, \boldsymbol{u}, \boldsymbol{x}) = \rho + \frac{1}{2}\boldsymbol{u}^T R\boldsymbol{u} + \boldsymbol{\lambda}^T[A\boldsymbol{x} + B\boldsymbol{u} + C] \tag{6}$$

The necessary conditions of the optimal trajectory and control are eq. (5b) and:

$$-\dot{\boldsymbol{\lambda}} = \frac{\partial H}{\partial \boldsymbol{x}} = A^T\boldsymbol{\lambda} \tag{7}$$

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

$$0 = \frac{\partial H}{\partial \boldsymbol{u}} = R\boldsymbol{u} + B^T\boldsymbol{\lambda} \rightarrow \boldsymbol{u} = -R^{-1}B^T\boldsymbol{\lambda} \tag{8}$$

Final conditions for the differential eq. (7) will be derived later, as needed.

B. Minimum Energy Pseudometric

The *Lagrange* multiplier vector $\boldsymbol{\lambda}$ is independent of the state, and a closed-form solution for it can be obtained by integrating eq. (7) backwards in time:

$$\boldsymbol{\lambda}(t) = exp[A^T(T-t)]\boldsymbol{\lambda}(T) \tag{9}$$

Before the above can be used to compute the optimal control policy of eq. (8), the final condition $\boldsymbol{\lambda}(T)$ is required as derived next. Substituting eq. (9) into eq. (5b) and integrating for the final state $\boldsymbol{x}_2$ gives:

$$\boldsymbol{x}(T) = \boldsymbol{x}_2 = exp[AT]\boldsymbol{x}_1 - \int_0^T exp[A(T-\tau)]BR^{-1}B^T exp[A^T(T-\tau)]\boldsymbol{\lambda}(T)d\tau + \int_0^T Cexp[A(T-\tau)]d\tau \tag{10}$$

Introducing the *Weighted Controllability Grammian* as:

$$G(t_0, T) = \int_{t_0}^T exp[A(T-\tau)]BR^{-1}B^T exp[A^T(T-\tau)]d\tau \tag{11}$$

it can be found as the solution to the Lyaponov differential Equation (12),[15]

$$\dot{P} = AP + PA^T + BR^{-1}B^T \;\; ; \;\; P(t_0) = 0 \tag{12}$$

such that $G(t_0, t) = P(t)$. By using the ZIR (zero-input-response) in eq. (10) we obtain a concise form for $\boldsymbol{x}_2 = ZIR(\boldsymbol{x}_1, T) - P(T)\boldsymbol{\lambda}(T)$. Adapting the missile guidance terminology for *zero-effort-miss* denoted by

$$\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T) = \boldsymbol{x}_2 - ZIR(\boldsymbol{x}_1, T) \tag{13}$$

we can immediately write a solution for $\boldsymbol{\lambda}(T)$ as:

$$\boldsymbol{\lambda}(T) = -P^{-1}(T)\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T) \tag{14}$$

The above development, in particular, eqs. (9) and (14) allows us to write the optimal control policy merely as function of the boundary states and the time horizon:

$$\boldsymbol{u}(t) = R^{-1}B^T exp[A^T(T-t)]P^{-1}(T)\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T) \tag{15}$$

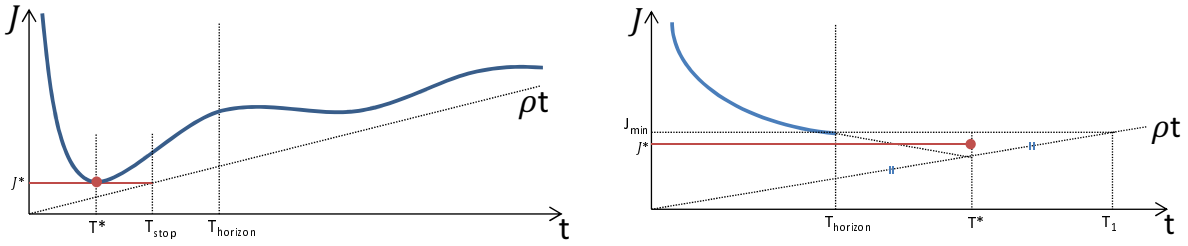
This control policy can be used to estimate the *cost-to-go* of a specific vertex from $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$ by substituting eq. (15) into eq. (5):

$$J(\boldsymbol{x}_1, \boldsymbol{x}_2, T) = \rho T + \frac{1}{2}\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T)^T P^{-1}(T)\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2, T) \tag{16}$$

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). In order to find a minimum time interval upon which the trajectory is executed we shall note:

$$T^* = \underset{T}{\text{argmin}} \;\; J(\boldsymbol{x}_1, \boldsymbol{x}_2, T) \tag{17}$$

The optimal T^* time is found by computing $J(\boldsymbol{x}_1, \boldsymbol{x}_2, T)$ from $T = 0$ until a predetermined time horizon $T_{horizon}$. The cost-to-go is comprised of a linear term in time and a quadratic term weighted by P^{-1}, thus, the lower boundary for the cost is ρT. The cost $J(T)$ is continuously compared to the minimal cost found thus far J^* with its corresponding T^*. Once $\rho T \geq J^*$ the search is 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 $\mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_2)$. This is graphically depicted in fig. 1(a).



(a) Exact MEAQR Cost Function

(b) Approximate MEAQR Cost Function

Figure 1: MEAQR Cost Function : An illustration of the cost function vs. time during the search for optimal time. (a) When the allowed time horizon is sufficiently large such that $T^* < T_{horizon}$ an exact value of J^* can be obtained. (b) When the time horizon is not sufficiently 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.

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. 1(b). 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 $J^* = \frac{1}{2}(J_{min} + \rho T^*)$.

Finally, the optimal time T^*, in whichever way it was obtained, is used to compute the closed loop control policy as presented in section C simultaneously with the closed loop gain matrix and bias vector.

As eq. (12) is integrated for an increasingly larger time intervals, several types of behaviours can occur for the weighted controllability grammian P affecting the computation of the objective function. 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 (Lyapunov matrix differential equation is a particular simpler case of RDE). Due to the structure of the cost-to-go in eq. (16) it would be preferable to see the grammian $P \to \infty$ so that the second term in eq. (16) would converge to zero. The convergence depends on the properties of the dynamic system, the linearisation point and the objective function weighting matrix R. 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 solving of the algebraic Riccati equation is attempted). The interested reader is referred to the works by Callier, Frank and Park[17,18,19] for studies on the RDE behaviour for various systems.

C. 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 in real life systems, actuation is limited in magnitude as well as in bandwidth. Formulating a feedback extension heuristic as opposed to an open loop policy enables us to account for those constraints. We therefore assume that $\boldsymbol{\lambda}$ is a function of the state $\boldsymbol{x}$ rather then an autonomous variable (as eq. (7) 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 $\boldsymbol{x}(t)$,[16] that is:

$$\boldsymbol{\lambda}(t) = M(t)\boldsymbol{x}(t) + \boldsymbol{\eta}(t) \quad ; \quad \boldsymbol{\eta}(T) = 0 \tag{18}$$

Differentiating eq. (18) and substituting eqs. (5b) and (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. (7), we get :

$$[\dot{M} + MA + A^T M - MBR^{-1}B^T M]\boldsymbol{x} + [\dot{\boldsymbol{\eta}} + A^T\boldsymbol{\eta} - MBR^{-1}B^T\boldsymbol{\eta} + MC] = 0 \tag{19}$$

from which two differential equations follow:

$$\dot{\boldsymbol{\eta}} + [A^T - MBR^{-1}B^T]\eta + MC = 0 \quad ; \quad \boldsymbol{\eta}(T) = 0 \tag{20a}$$

$$\dot{M} + MA + A^T M - MBR^{-1}B^T M = 0 \quad ; \quad M(T) = \infty \tag{20b}$$

where the final condition for M arises from the fixed-final-state. This condition, however, makes the integration of eq. (20b) backwards in time from T^* impractical, and we replace the above by the differential equation for M^{-1}, using the relationship $\dot{M}^{-1} = -M^{-1}\dot{M}M^{-1}$, with $M^{-1}(T) = 0$. In summary, once the closed loop control policy is called upon, we shall integrate the following system backwards in time:

$$\begin{bmatrix} \dot{\boldsymbol{\eta}} \\ \dot{M}^{-1} \end{bmatrix} = \begin{bmatrix} \left(A^T - MBR^{-1}B^T\right)\boldsymbol{\eta} + MC \\ AM^{-1} + M^{-1}A^T - BR^{-1}B^T \end{bmatrix} \quad ; \quad \begin{bmatrix} \boldsymbol{\eta} \\ M^{-1} \end{bmatrix}_{t=T} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{21}$$

Solution of eqs. (18) and (21) allows us to compute the optimal closed-loop control policy from eq. (8).

D. Practical Implementation

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

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 through the numerical integration process for computing the closed loop control matrices in eq. (21). The initial condition for $P(0) = 0$ is merely theoretical, but its inversion is needed; therefore, the matrix is defined initially with small elements on its diagonal (numerical damping), which can result in deviation of P from maintaining its positive-definiteness. To ensure the PD of P, the *Cholesky* decomposition[20] is used. Thus,

$$P = LL^T \tag{22}$$

where L is lower triangular. Substituting eq. (22) and its derivative into the RDE in eq. (12) gives:

$$\dot{L}L^T + L\dot{L}^T = ALL^T + LL^T A^T + BR^{-1}B^T \tag{23}$$

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

$$\underbrace{L^{-1}\dot{L}}_{\text{Lower Triangular}} + \underbrace{\dot{L}^T L^{-T}}_{\text{Upper Triangular}} = L^{-1}AL + L^T A^T L^{-T} + L^{-1}BR^{-1}B^T L^{-T} \tag{24}$$

The left hand side of eq. (24) is a sum of a lower triangular matrix and an upper triangular matrix denoted as $F = F_{LT} + F_{UT}$. Computing F from eq. (24), $\dot{L}$ 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. (22) as needed.

2. *Scaling the 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. (5) is $\boldsymbol{u}^T R\boldsymbol{u}$, and hence for units of the control signal taken as Newtons 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.

3. 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 paper. An inverted pendulum is limited in motor torque and the *X8* vehicle is limited in thrust and moments around each of its axes. These constraints are not necessarily symmetrical and can be written as : $\boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max} \quad \forall t \in [0, T^*]$.

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 control is formulated as : $||\dot{\boldsymbol{u}}(t)|| \leq \dot{\boldsymbol{u}}_{max} \quad \forall t \in [0, T^*]$.

At each integration step both the actuation and its approximate first order derivative $\dot{\boldsymbol{u}}_{approx} = [\boldsymbol{u}_{k+1} - \boldsymbol{u}_k] / \Delta t$ are tested to verify their conformity to the bounds defined. The global minimal and maximal values, $\boldsymbol{u}_{max}$ and $\boldsymbol{u}_{min}$, are inherent in the systems' dynamics whereas the time dependant bounds are computed by:

$$\bar{u}_{max} = \boldsymbol{u}_k + \dot{\boldsymbol{u}}_{max} \Delta t \quad ; \quad \bar{u}_{min} \quad = \boldsymbol{u}_k - \dot{\boldsymbol{u}}_{max} \Delta t \tag{25}$$

If a violation is detected, the appropriate signal is saturated accordingly and the integration continues normally. 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.

VI. Kinodynamic Minimum Energy RRT*

Now that the RRT* algorithm has been explained as well as the suggested pseudometric, we shall present the implementation scheme that integrates all the previous elements that comprise the proposed kinodynamic motion planner shown in algorithm VI.1. As presented in section V, we are interested in the general case of a motion planning for a non-linear system.

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. (5) is $\boldsymbol{x}_i = \boldsymbol{x}_{near} - \boldsymbol{x}_{rand}$ and the final state is $\boldsymbol{x}_f = \boldsymbol{0}$. The sampling scheme is uniform on the zero-velocity manifold, as are the initial and goal states for the two example systems, thus minimizing the non-stabilizable states attempted to be steered towards.

The state feedback control matrices M & $\boldsymbol{\eta}$ in eq. (19) are computed during the KNN search simultaneously with P in eq. (11) for a time horizon of $T_{horizon} = T_{max} \left(\frac{\log N}{N} \right)^{1/d}$ where N is the current number of vertices in the tree and d is the SS dimensionality. The former are used to steer whichever neighbour was chosen towards $\boldsymbol{x}_{rand}$. If the dynamic model is linear it is beneficial to pre-compute all of the aforementioned time dependant matrices for some predefined T_{max} then use those to determine the nearest neighbour as well as for the steering with the appropriate initial and final conditions.

To shorten the time till an initial solution is generated we use branch and bound as well as goal biasing. Branch and bound allows only adding states which have a lower cost than the presently minimal cost. Goal biasing is done by attempting to connect newly added state at the end of each new segment - to the goal, given is it within a predefined vicinity.

VII. Simulation Results

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 A and section B 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 simulations were conducted on a 64-bit PC with Intel i7-2600 3.4Ghz with 8GB RAM running Mathworks Matlab© 2012b.

Algorithm VI.1 RRT* with MEAQR

procedure RRT*(x_{init},X_{goal},X_{free},N)
Initialize($\mathbf{T}, x_{init}$)
  while $i \leq N$ **do**
    $x_{rand} \leftarrow Sample$
    $x_{near} \leftarrow KNN(x_{rand}, \mathbf{T})$
    while $\mathcal{D}(x_{near}, x_{rand}) > \bar{\mathbb{E}}$ **do**
      $x_{rand} \leftarrow 0.5(x_{near} + x_{rand})$
      $x_{near} \leftarrow KNN(x_{rand}, \mathbf{T})$
    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**
      $\mathbf{T}.edges \cup \sigma_{min}$
      $\mathbf{T}.vertices \cup x_{new}$
      $\mathcal{V}_{near} \leftarrow KNNRewire(x_{new}, \mathbf{T})$
      Rewire($\mathbf{T}, \mathcal{V}_{near}, x_{new}$)
    end if
  end while
end procedure

As a benchmark for algorithm performance, the L_2-norm of the actuation signal $\mathbb{E}_{in} = \int_0^{T_{total}} ||u||^2 dt$ 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.

A. Torque Limited Pendulum

The torque limited inverted pendulum is depicted in fig. 2 and its dynamics with viscous frictional damping term are described with $\mathcal{I}\ddot{\theta} = -mgl\cos\theta - b\dot{\theta} + \tau$.

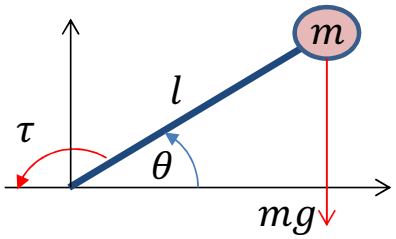


Figure 2: Inverted Pendulum Free Body Diagram.

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

Table 1: Pendulum State Space and Actuation Bounds

The state vector is defined as $\boldsymbol{x} = [\theta, \dot{\theta}]^T$ and the parameters of the model are: moment of inertia $\mathcal{I} = ml^2$, mass $m = 1kg$, length $l = 1m$, viscous friction coefficient $b = 0.1Nms$ and $g = 9.81ms^{-2}$. The environment in which the pendulum manoeuvres is obstacle free. The pendulum state space bounds arise from dynamic and geometric constraints as listed in table 1 along with the actuation magnitude $||\tau||$ and bandwidth (rate) $||\dot{\tau}||$ constraints.

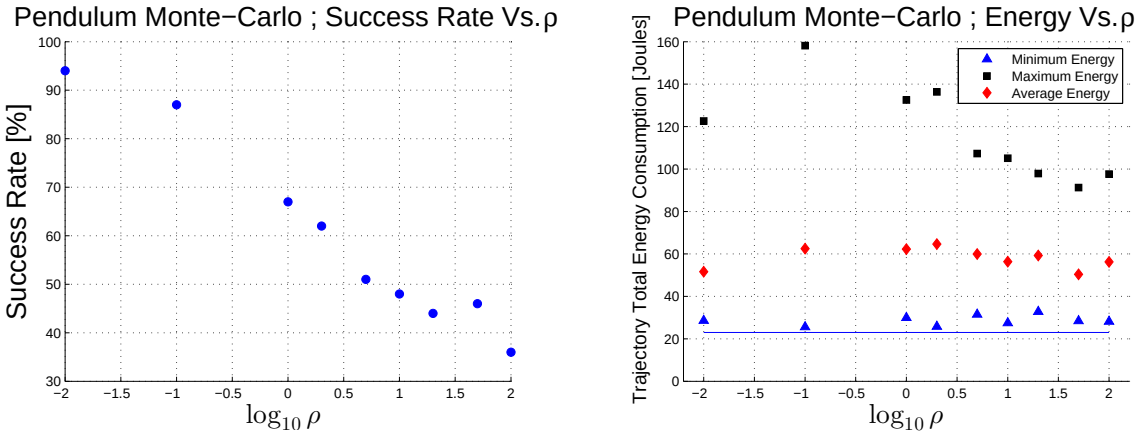
The trajectory generated by the motion planner originates from the downward equilibrium state $\boldsymbol{x}_{init} = [-\frac{\pi}{2}, 0]^T$ searching a trajectory to the inverted equilibrium state $\boldsymbol{x}_{goal} = [\frac{\pi}{2}, 0]^T$. The change in energy, in the absence of non-conservative forces, needed to quasi-statically bring the pendulum from the initial to the

goal state is:

$$\Delta\mathbb{E} = mgl(1 + \sin\left(\frac{\pi}{2}\right)) = 19.82 \;\; J \tag{26}$$

The work done by the viscous friction depends on a sample trajectory such as the one presented in fig. 4 is $W_f = -3.14\,J$. This suggests that an approximate lower bound of input energy needed for the trajectory, which shall serve as a benchmark for the motion planner performance: $\underline{\mathbb{E}}_{in} = \Delta\mathbb{E} - W_f \approx 23J$

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. Figure 3 report the results of *Monte-Carlo* simulations.



(a) Success Rate Vs. ρ

(b) Energy Vs. ρ

Figure 3: Pendulum *Monte-Carlo* Simulation Results : (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 $\underline{\mathbb{E}}_{in} = 23J$ is also shown as a blue line. Values are generated by running 100 *Monte-Carlo* simulations for each value of ρ.

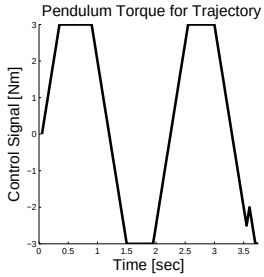
The results in fig. 3 show that changing the pseudometric parameter ρ does not yield a difference in the total *minimum* energy used on a given trajectory, which correspond to the aforementioned lower bound. From above, the energy value is unbounded since there might be a case for which no trajectory is generated hence the infinite cost. A run for which no trajectory was generated is discarded, and counted as a failure (hence the success rate).

The decreasing trend in the maximum energy used, is due to the decreasing success rate. As the success rate decreases, there are significantly less trials to represent the true underlying distribution of energy consumption for the trajectories. If infinitely many Monte-Carlo simulations were attempted, the maximal energy levels recorded will be, as mentioned, unbounded.

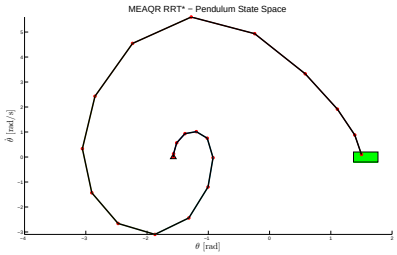
As claimed earlier, if an optimal trajectory exists, for which the energy consumption is minimal, the algorithm shall converge to it, regardless of the parameters ρ and R. The parameters do however, affect the rate of convergence of the algorithm to the optimum. Given a sufficient amount of running time 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 the convergence. A sample energy optimal and suboptimal trajectories along with their actuation signal and states versus time are presented in figs. 4 and 5 respectively. The saturation and bandwidth limits are apparent in the actuation plot versus time in figs. 4(a) and 5(a).

B. Quadrotor

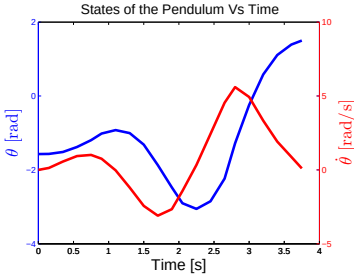
The quadrotor simulation is based on the previously mentioned *Draganflyer X8* vehicle shown in fig. 6. Due to the complexity of the vehicle and the absence of a comprehensive model for this specific vehicle, it is modelled in section A as a four rotor helicopter with approximated aerodynamic body drag 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) Actuation

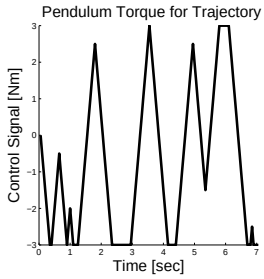


(b) Pendulum State Space

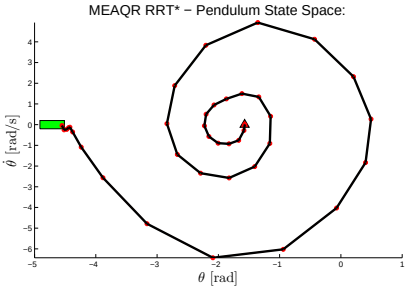


(c) States Vs. Time

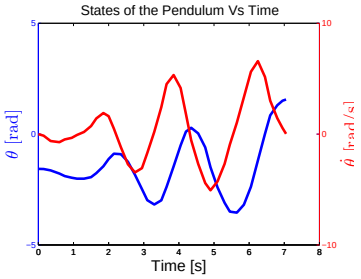
Figure 4: Pendulum Energy Optimal Trajectory with the MEAQR pseudometric parameters: $\rho = 1, R = \frac{8}{3}$. The total energy needed is $\mathbb{E}_{in} = 25.5$ *Joul*, the work done by the viscous friction $W_f = -3.14$ *Joul* and $T_{total} = 3.7s$.



(a) Actuation



(b) Pendulum State Space



(c) States Vs. Time

Figure 5: Pendulum Energy Non-Optimal Trajectory with the MEAQR pseudometric parameters: $\rho = 1, R = \frac{8}{3}$. The total energy needed is $\mathbb{E}_{in} = 30.65$ *Joul*, the work done by the viscous friction $W_f = -6.04$ *Joul* and $T_{total} = 7.05s$.

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

Name	Units	Lower Bound	Upper Bound	Maximal Rate
$\mathcal{F}$	N	0	35	20
$\boldsymbol{\tau}$	N m	-5	5	10
$[x, y, z]$	m	0	5	-
$\|\|\boldsymbol{v}\|\|$	m/sec	0	6	-
$\|\|\boldsymbol{\omega}\|\|$	rad/sec	0	3	-

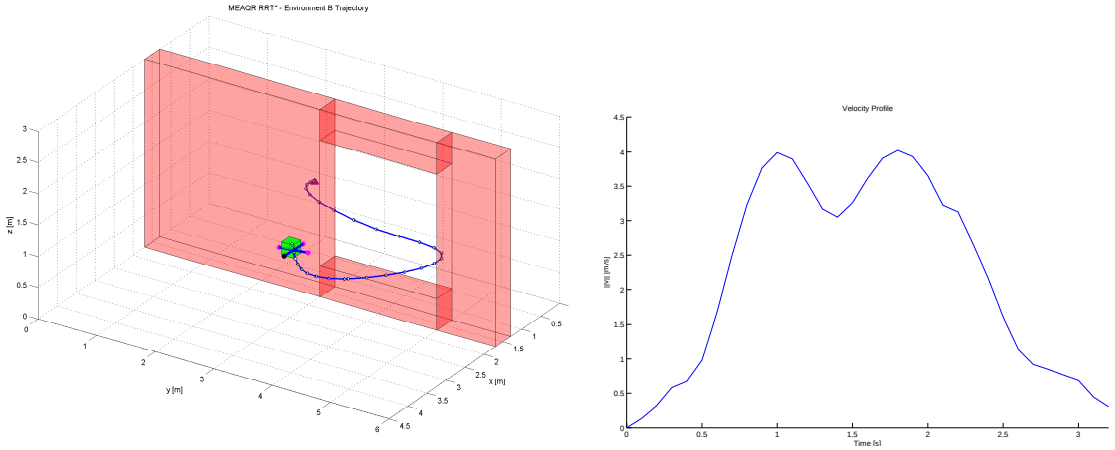
Table 2: *X8* actuation and state-space bounds

The quadrotor state bounds arise from dynamic and geometric constraints whereas the actuation pair $\boldsymbol{u} = (\mathcal{F}, \boldsymbol{\tau})$ is limited in magnitude $\|\boldsymbol{u}\|$ and in bandwidth (rate) $\|\dot{\boldsymbol{u}}\|$ due to motor capabilities. Both are listed in table 2.

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

Table 3: *X8* MEAQR pseusdometric parameters

The MEAQR RRT* algorithm is queried with the pseudometric and planner properties in table 3. The motion planner is invoked to determine a path for the *X8* in a room the size of $5m \times 5m \times 5m$ divided by a windowed wall, the obstacles of which are primitive-based. The initial and target positions queried are $\boldsymbol{p}_{init} = [0.3, 2, 1]$ and $\boldsymbol{p}_{goal} = [4, 4, 2]$ respectively. Figure 7 depicts the resulting trajectory in the workspace and its corresponding speed profile. The energy needed to complete this trajectory is $\mathbb{E}_{in} = 2818\, Joul$ and the time needed to complete it is $T_{traj} = 3.3\, s$. For comparison purposes, it takes approximately $760 J$ to maintain the X8 in hover the same amount of time based on $P_{hover} = 230\, W$. It takes, on average, approximately an hour to generate this trajectory.



(a) Workspace 3D view

(b) Velocity Profile

Figure 7: *X8* Path in Artificial Environment B : The projection of the trajectory on the *3D* workspace is shown in red in a small $5m \times 5m \times 5m$ room with a windowed wall.

VIII. Conclusion

We have presented a kinodynamic RRT* minimum energy motion planner, an incremental sampling-based approach that extends the RRT* for systems with differential and actuation constraints. The proposed approach achieves asymptotic optimality as guaranteed by the RRT* *almost-sure* convergence by formulating an open-loop pseudometric for the KNN search and a closed-loop optimal controller to connect two states. 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. The proposed algorithm was simulated 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.

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.

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. Nonetheless, this pseudometric can be used as a benchmark for the minimal energy needed to transverse an environment, and compared to the output of other planners.

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:

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	$0.5I_{3\times3}$

Table 4: *X8* Physical Parameters

B. Algorithms

Algorithm A.1: RRT*

```
1: procedure RRT*(x_init,X_goal,X_free,N)
2: Initialize(T, x_init)
3:     while i ≤ N do
4:         x_rand ← Sample
5:         x_near ← KNN(x_rand, T)
6:         [x_new, σ] ← Steer(x_near, x_rand)
7:         X_near ← KNN(x_new, T)
8:         [x_min, σ_min] ← ChooseParent(X_near, x_rand)
9:         if CollisionFree(σ, X_free) then ▷ [a]
10:            T.Edges ← T.Edges ∪ σ
11:            T.Vertices ← T.Vertices ∪ x_new
12:            Rewire(T, X_near, x_new)
13:        end if
14:    end while
15: end procedure
```

Algorithm A.2: K-Nearest-Neighbours

```
1: procedure KNN(T,T_max,K)
2:     k ← 0        ▷ Initialize neighbour counter
3:     Candidates ← All vertices in T
4:     while t ≤ T_max do
5:         BestCost = ∞
6:         for all Candidates do
7:             J ← ρt + ½ d^T P^{-1} d
8:             if J ≤ BestCost(i) then
9:                 BestCost(i) ← J
10:                OptimalTime(i) ← t
11:            end if
12:            if BestCost(i) ≤ ρt then
13:                Neighs(k) ← Candidate(i)
14:                Candidates ← Candidates \ Node(i)
15:            end if
16:            if #Neighs = K then return Neighs
17:            end if
18:        end for
19:    end while
20: end procedure
```

Algorithm A.3: ChooseParent

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

Algorithm A.4: Rewire

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

Acknowledgments

This work has been carried out with financial support from the research contract with DRDC-Suffield, contract number W7702-11-5122.out, represented by Michael Trentini. The first author thanks Mikael Persson of *McGill University - CIM* for his support in general and his advice on probabilistic motion planning in particular. This work also benefited from an initial code contribution by Steve Levine of *MIT - CSAIL*.

References

[1]Webb, D. J. and van den Berg, J., "Kinodynamic RRT*: Optimal Motion Planning for Systems with Linear Differential Constraints," *CoRR*, Vol. abs/1205.5088, 2012.

[2]Glassman, E. and Tedrake, R., "A quadratic regulator-based heuristic for rapidly exploring state space," *Robotics and*

Automation (ICRA), 2010 IEEE International Conference on, IEEE, 2010, pp. 5021–5028.

[3]Karaman, S. and Frazzoli, E., "Sampling-based Algorithms for Optimal Motion Planning," *International Journal of Robotics Research*, Vol. 30, No. 7, June 2011, pp. 846–894.

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

[5]LaValle, S. and Kuffner, J.J., J., "Randomized Kinodynamic Planning," *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, Vol. 1, 1999, pp. 473 –479 vol.1.

[6]Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H., "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics & Automation*, Vol. 12, No. 4, June 1996, pp. 566–580.

[7]Kavraki, L., Kolountzakis, M., and Latombe, J.-C., "Analysis of probabilistic roadmaps for path planning," *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, Vol. 4, apr 1996, pp. 3020 –3025 vol.4.

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

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

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

[11]Perez, A., Platt, R., Konidaris, G., Kaelbling, L., and Lozano-Perez, T., "LQR-RRT*: Optimal Sampling-Based Motion Planning with Automatically Derived Extension Heuristics," *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2012, pp. 2537–2542.

[12]Cheng, P. and LaValle, S., "Reducing metric sensitivity in randomized trajectory design," *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, Vol. 1, 2001, pp. 43 –48 vol.1.

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

[14]Bellman, R., "Dynamic programming and Lagrange multipliers," *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 42, No. 10, 1956, pp. 767.

[15]Lewis, F. L., Vrabie, D., and Syrmos, V. L., *Optimal Control*, Wiley, 1995.

[16]Garber, V., "Optimum intercept laws for accelerating targets." *AIAA Journal*, Vol. 6, Nov. 1968, pp. 2196–2198.

[17]Callier, F. and Willems, J., "Criterion for the convergence of the solution of the Riccati differential equation," *Automatic Control, IEEE Transactions on*, Vol. 26, No. 6, 1981, pp. 1232–1242.

[18]FRANK, M., WINKIN, J., and JACQUES, L., "Convergence of the time-invariant Riccati differential equation and LQ-problem: mechanisms of attraction," *International Journal of Control*, Vol. 59, No. 4, 1994, pp. 983–1000.

[19]Park, P. and Kailath, T., "Convergence of the DRE solution to the ARE strong solution," *Automatic Control, IEEE Transactions on*, Vol. 42, No. 4, apr 1997, pp. 573 –578.

[20]Stewart, G., *AFTERNOTES*, SIAM.

[21]Tedrake, R., "LQR-Trees: Feedback motion planning on sparse randomized trees," 2009.

[22]Channell, P. J. and Scovel, C., "Symplectic integration of Hamiltonian systems," *Nonlinearity*, Vol. 3, No. 2, 1990, pp. 231.

[23]Hairer, E., Lubich, C., Wanner, G., et al., "Geometric numerical integration illustrated by the Stormer-Verlet method," *Acta Numerica*, Vol. 12, 2003, pp. 399–450.