KISS Post-doctoral Fellowship Final Report

Marin Kobilarov

July 24, 2012

Summary

The KECK Institute fellowship played an instrumental role in my development as a scientist. It gave me freedom to pursue new ideas and work on novel concepts leading to important scientific advancements. I am grateful for having the opportunity to be part of the group of people comprising KISS and hope that these relationship will be long-lasting.

My work consisted of developing new general analytical/computational methods as well as applications in aerospace robotics. The analytical methods were focused on the dynamics, control, and motion planning of autonomous vehicles. These methods were applied to the autonomous control of small spacecraft in the context of the Autonomous Assembly of a Reconfigurable Space Telescope (AAReST) project lead by Professor Sergio Pellegrino.

This report is composed of several papers that describe key results in my research. In particular, the paper titled "Trajectory Planning for Cubesat Proximity Operations" details our approach to autonomous path planning for small satellites such as the ones intended for AAReST. The remaining papers develop general methods that are applicable to any autonomous vehicles such as asatellite, micro aerial vehicles, or wheeled rovers.

Outline. The report includes a list of published papers and invited lectures during the last two years. The remainder is composed of the key journal articles documenting the results of this research.

Publications

Journal Articles

1. M. Kobilarov, S. Pellegrino: Trajectory Planning for Cubesat Proximity Operations, to be submitted, *Journal of Guidance, Dynamics, and Control* (2012).

2. K. Flaßkamp, S. Ober-Blöbaum, M. Kobilarov: Solving Optimal Control Problems by Exploiting Inherent Dynamical Systems Structures, accepted, to appear in *Journal of Nonlinear Science* (2012).

3. M. Kobilarov: Cross-Entropy Sampling-based Motion Planning, accepted, to appear in the *International Journal of Robotics Research*, (2012).

4. M. Kobilarov, J. E. Marsden, G. S. Sukhatme: Global Estimation in Constrained Environments, *International Journal of Robotics Research*, (2011).

5. A. Banaszuk, V. Fonoberov, T. A. Frewen, M. Kobilarov, G. Mathew, I. Mezic, A. Pinto, T. Sahai, H. Sane, A. Speranzon, A. Surana: Scalable Approach to Uncertainty Quantication and Robust Design of Interconnected Dynamical Systems, *IFAC Annual Reviews in Control*, (2011).

6. M. Kobilarov, J. E. Marsden: Discrete Geometric Optimal Control on Lie Groups. *IEEE Transactions on Robotics*, **27** (4), (2011), 641–655.

Refereed Conference Papers

1. M. Kobilarov, Cross-Entropy Randomized Motion Planning, *Robotics: Science and Systems*, (2011).

2. S. Nair and M. Kobilarov: Collision Avoidance Norms In Trajectory Planning, *American Control Conference*, (2011), 4667–4672.

3. T. A. Frewen, H. Sane, M. Kobilarov, S. Bajekal, K. R. Chevva: Adaptive Path Planning in a Dynamic Environment using a Receding Horizon Probabilistic Roadmap: Experimental Demonstration, *Proceedings of the American Helicopter Society Specialists' Meeting on Unmanned Rotorcraft*, (2011).

4. K. Flaßkamp, S. Ober-Blöbaum, M. Kobilarov: Solving Optimal Control Problems Using Inherent Dynamical Properties, *Proceedings in Applied Mathematics and Mechanics*, **10** (1), (2010), 577-578.

Invited Lectures

Selected Lectures

- Applications of Geometric Control, special session on Geometric Mechanics, *AIMS Conference on Dynamical Systems, Differential Equations, and Applications, Arlington*, Orlando, (2012)

- Stochastic optimization of dynamical systems, *Universidad Autonoma de Madrid*, Madrid, (2012)

- Motion Control of Autonomous Systems with Applications to Robotics and Aerospace, *New Trends in Aerospace seminar* , MIT, (2011)
- Motion control of autonomous systems: reconfigurable spacecraft and fast air vehicles, *Information Science and Technology seminar series*, Caltech, (2011)
- Numerical Aspects and Applications of Discrete Geometric Control, *Institute for the Mathematical Sciences*, Madrid, (2011)
- Computational Optimal Control on Lie Groups, *Second Iberoamerican Meeting on Geometry, Mechanics and Control, in honor of Hernán Cendra*, Bariloche, Argentina, (2011)
- Globally-optimal Computational Geometric Control under Uncertainty, *AFOSR Contractors Meeting*, Washington, DC, (2010)

Graduate Course Lectures

- Guidance, Navigation, and Control, *Aerospace Engineering: AE 105*, Caltech, (2010, 2011).

Selected Articles

Trajectory Planning for Cubesat Proximity Operations

Marin Kobilarov and Sergio Pellegrino

–in preparation–

July, 2012

Abstract

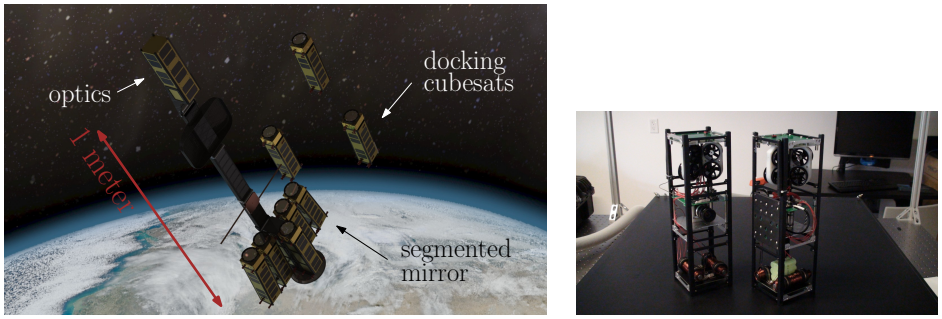
This paper considers motion planning for small (e.g. cubesat-type) satellites operating within a few meters of a target object. The main goal is to develop a principled methodology for handling the coupled effects of orbital dynamics, rotational and translational rigid body dynamics, underactuation and control bounds, and other constraints such as obstacles. The proposed approach is based on constructing a reduced-order representation of the dynamics through dynamics inversion and differential flatness, and on efficient global optimization over a finite dimensional reduced representation. Two simulated scenarios, a satellite reconfiguration maneuver and asteroid surface sampling, are developed to illustrate the approach. In addition, a simple 2-D experimental testbed consisting of an air-bearing table and two cubesat engineering models is developed for partial testing and integration of the proposed methods.

1 Introduction

This paper studies autonomous control of small spacecraft during proximity operations. Autonomous navigation and operation in the proximity of multiple objects in space is becoming increasingly important. It enables critical capabilities such as autonomous on-orbit assembly and inspection, servicing of disabled spacecraft, or debris deorbiting.

This work focuses on autonomous navigation of small and low-cost spacecraft such as cubesats operating within a range of several meters around the target object (e.g. a satellite being serviced, an instrument being assembled, or debris being captured). The cubesat platform is a natural choice for such capability because of its low development and launch costs and unified standard that has accelerated a wide-spread development effort. Even though various small satellite technology are quickly maturing, there are various practical limitations that currently preclude a fully autonomous operation. In particular, robotic operation in space depends on real-time perception, spatial mapping, path planning and trajectory control. The main challenges lie in dealing with limited propulsion due to underactuation, bounded thrust, and finite fuel, and attitude control system (ACS) with bounded torque. In addition, safety constraints such as avoiding collisions with obstacles and perception constraints, such as maintaining sensor field-of-view of target object while avoiding the incident sun angle, must be satisfied. Control methods should also be able to handle high uncertainty in the output response of current cubesat propulsion systems while this technology is maturing.

An example scenario. Consider the *Autonomous Assembly of a Reconfigurable Space Telescope* (AAReST) mission [35] illustrated in Figure 1. The mission is intended for a near-circular orbit at 650km altitude. Table 2 lists the various orbit details. The mission relies on the ability of individual cubesats to autonomously undock, orbit around the mirror, and redock to a different position at the main cluster. Note that under ideal conditions such reconfigurations



a) b)

Figure 1: Autonomous Assembly of a Reconfigurable Space Telescope (AAReST) concept: a) simulated rendezvous and docking of multiple cubesats to form a segmented mirror; each spacecraft perceives the environment using machine vision and autonomously navigates to and docks in a designated configuration. b) a cubesat autonomous reconfiguration testbed using an air table.

Element	Notation	Value	Units	Notes
Dimensions	(d_x, d_y, d_z)	$(0.1, 0.1, 0.3)$	m	
Moments of Inertia	$\mathbb{I}$	$(0.03, 0.03, 0.015)$	$\text{kg} \cdot \text{m}^2$	
Mass	m	3	kg	
Altitude	a_c	650	km	
Inclination	i_c	97.985823	deg.	
Orbit period	T_e	97.728221	min	
Angular rate	ω_c	.00107154	rad/s	
Velocity	v_c	7530.93	m/s	
Relative Perturbations	$\|\|f_{\text{hcw}}\|\|$	$\leq 1.9712 \times 10^{-4}$	N	relative $\|\|v\|\| \leq$ 2cm/s, range $\leq$ 2 m.
Gravity Gradient	$\|\|\tau_{\text{gg}}\|\|$	$\leq 2.5830 \times 10^{-8}$	Nm	max at 45^o
Solar Torque	$\|\|\tau_{\text{SRP}}\|\|$	$\leq 2.7360 \times 10^{-9}$	Nm	COM $\pm[0.02, 0.02, 0.02]$
Solar Force	$\|\|f_{\text{SRP}}\|\|$	$\leq 1.3680 \times 10^{-7}$	N	at max cross area
Drag Torque	$\|\|\tau_{\text{drag}}\|\|$	$\leq 3.3166 \times 10^{-13}$	Nm	COM $\pm[0.02, 0.02, 0.02]$
Drag Force	$\|\|f_{\text{drag}}\|\|$	$\leq 1.6583 \times 10^{-11}$	N	at max cross area

Figure 2: AAReST orbit details.

are accomplished by exploiting relative orbital dynamic forces and thus last around one orbital period. Yet, a simple calculation yields that if the spacecraft are required to operate within close proximity (e.g. less than two meters) then the maximum relative perturbation force does not exceed 1.9712×10^{-4} N (Figure 2). This corresponds to motions no faster than 2 mm/s that can only be achieved with very precise boundary conditions. We expect that Such precision will be very difficult to achieve when interaction between bodies such docking/attaching is involved. In particular, it is realistic to instead assume that relative velocities between interacting bodies in on the order of at least several cm/s. The optimal control strategy in such cases will differ from standard formation flying techniques. For instance, a reconfiguration in the context of AAReST will last several minutes rather than more than one hour and will only partially exploit the orbital dynamics perturbations. In addition, thrusters capable of milli-Newton (variable or high-frequency PWM) thrust will be required.

Approach Motivated by the need for algorithms applicable to shorter time scales and constrained underactuated propulsion we develop techniques for designing approximately optimal trajectories that account for the effects of rigid body dynamics, orbital dynamics, and underactuation. The optimal reconfiguration problem can be regarded as a constrained nonlinear optimal control problem. The general formulation does not have a closed-form or easily computable solution. The two key points of the proposed approach are then: 1) to exploit key properties of the spacecraft dynamics to obtain a reduced order trajectory representation and 2) to employ efficient stochastic global optimization over this reduced space. The case of fully-actuated and

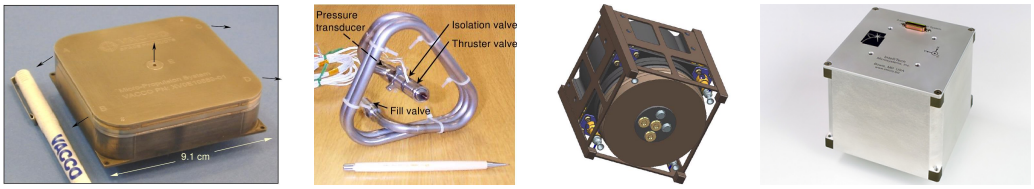
under-actuted systems is handled in a unified way through either simple dynamics inversion or by exploiting *differential flatness*, respectively. Thus the proposed method can be used to compute motions to any desired state for cubesats with minimal actuation, i.e. an ACS and a single thruster (that is realizable with current technology) or with a more advanced conceptual 10-thruster fully actuated propulsion system which could be realizable in the next few years.

Related Work. The reconfiguration problem has been studied from various viewpoints. Traditionally, it is assumed that the spacecraft is a point mass the can be propelled in any given direction subject to distance constraints to other spacecraft or obstacles [28, 30]. In this context, convex programming [1, 32] or mixed-integer linear programming [25] has been successfully used for designing algorithms with provable convergence and run-time properties. Robotic motion planning is used to address the case with full rigid body dynamics and more complex avoidance constraints. The increased dimensionality and related computational burden has been addressed through e.g. randomized methods [23, 11]. The final stage of rendezvous has also been studied through a simple linear model [4]. The problem has also been extensively studied in the context of planned future missions such as Terestial Planet Finder (TPF) [29] and low cost spacecraft testbeds such as the Spheres [27, 3, 34]. The Spheres testbed has also served as a basis for a conceptual mission [19] sharing common aspects with AAReST. Differential flatness[7, 21, 37] is an intrinsic dynamical system property which can be used to transform a highly underactuated system into a system evolving in fully-controllable flat output space. Flatness has been used for spacecraft control purposes but primarily limited to underactuated attitude control problems (see e.g. [33]). An unrelated but interesting application of flatness for handling avoidance constraints appears in [18]. Other methods for handling complex dynamics and underactuation include reducing the dynamics to a driftless system through decoupling vector fields [6] or exploiting symmetries to encode the dynamics through a maneuver automaton [9]. Decoupling vector fields require the system to stop at various points along the motion and often result in motions far from optimal. The correct way to employ a maneuver automaton in the context of symmetries arising from the coupled orbital dynamics and underactuated rigid body dynamics during proximity operations has yet to be studied.

This papers does not assume a particular propulsion model and considers even the case when the spacecraft has a single thruster aligned with the center of mass. Our motion planning methodology leads to high quality open-loop trajectories. Yet, in the underactuated case a major issue is whether these trajectories are robust to noise and can be stably tracked. We will only partially address this issue based on results from backstepping control [37, 10, 5, 2]. Considering the underactuated case is motivated by the recent developments of various propulsion technologies [20]. Existing technologies such as cold gas and pulsed plasma thrusters are directly applicable to cubesats today as long as they can be miniaturized and properly packaged. While there are a number of developments in this direction we mention cold-gas systems such as VACCO's five thrusters MEMS system [14] and Aerojet's CubeSat Hydrazine Adaptable Monopropellant Propulsion System (CHAMPS)[31] as examples of promising technology for the near future. We should note that the MEPSI (Micro-Electromechanical-based Picosat Satellite Inspection Experiment) spacecraft can be considered as precursor to such technologies in its innovative use of smart materials such as photostructurable glass/ceramic. In addition, new technologies such as ion thrusters (see e.g. [24]) and electrospray [ref] result in higher efficiency and delta-V. Only few of these propulsion systems are in near-flight ready state currently. In general, further development is necessary to produce a versatile thruster system that can fully actuate all six degrees of freedom of a cubesat with levels of thrusts/torques enabling agile autonomous navigation.

2 Problem Formulation

The spacecraft is modeled as a single underactuated rigid body with position $\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3$ and orientation matrix $R \in \mathrm{SO}(3)$. The configuration $(\boldsymbol{x}, R)$ is defined with respect to a moving



a) b) c) d)

Figure 3: Cold-gas propulsion systems and an attitude control system: a) VACCO 5-valve MEMS system; b) SSTL heritage Resistojet; c) Aerojet's CubeSat Hydrazine Adaptable Monopropellant Propulsion System (CHAMPS); d) Pumpkin Cubesat MAI-200 ADACS

frame attached to the central body (target), such as an RSW frame [36] where the convention that x-axis is on-track and z-axis points away from Earth is taken. The *body-fixed* angular velocity is denoted by $\boldsymbol{\omega} \in \mathbb{R}^3$. The vehicle has mass m and principal moments of rotational inertia J_x, J_y, J_z forming the inertia tensor $\mathbb{J} = \text{diag}(J_x, J_y, J_z)$. The state space of the vehicle is $S = SE(3) \times \mathbb{R}^6$ with $\boldsymbol{s} = ((R, \boldsymbol{x}), (\boldsymbol{w}, \dot{\boldsymbol{x}})) \in S$ denoting the whole state of the system.

The spacecraft is actuated with control inputs $\boldsymbol{u} \in U$ where $U \subset \mathbb{R}^c$ is a bounded set. The function $\boldsymbol{\tau} : S \times U \to \mathbb{R}^3$ and $\boldsymbol{f} : S \times U \to \mathbb{R}^3$ maps from these control inputs to the resulting torques and forces acting on the body, respectively. The external forces and torques are denoted by the functions $\boldsymbol{\tau}_{\text{ext}} : S \to \mathbb{R}^3$ and $\boldsymbol{f}_{\text{ext}} : S \to \mathbb{R}^3$, respectively. Significant external forces and torques in LEO are due to relative dynamics and gravity gradients (see Figure 2). For near-circular orbit the forces are simplified using the Hill-Clohessy-Wiltshire (HCW) linearized model, i.e.

$$\boldsymbol{f}_{\text{ext}}(\boldsymbol{s}) = m \begin{bmatrix} 2\omega_c \dot{z} \\ -\omega_c^2 y \\ -2\omega_c \dot{x} + 3\omega_c^2 z \end{bmatrix},$$

with constant $\omega_c > 0$ denoting the circular orbit angular rate (e.g. see Figure 2). The external torques due to gravity gradients is defined according to

$$\boldsymbol{\tau}_{\text{ext}}(\boldsymbol{s}) = 3\omega_c^2 R\boldsymbol{e}_3 \times \mathbb{J} R\boldsymbol{e}_3,$$

where $\boldsymbol{e}_3 = (0, 0, 1)$. Note that in the context of very close proximity operations such as in AAReST (Figure 2) assuming that $x, y, z < 2$m and $v_x, v_y, v_z < 2$cm/s the forces and torques do not exceed 1.9712×10^{-4}N and 2.5830×10^{-8} Nm, respectively.

The equations of motion are

$$\dot{R} = R\widehat{\boldsymbol{\omega}}, \tag{1}$$

$$\mathbb{J}\dot{\boldsymbol{\omega}} = \mathbb{J}\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau}_{\text{ext}}(\boldsymbol{s}) + \boldsymbol{\tau}(\boldsymbol{s}, \boldsymbol{u}), \tag{2}$$

$$m\ddot{\boldsymbol{x}} = \boldsymbol{f}_{\text{ext}}(\boldsymbol{s}) + \boldsymbol{f}(\boldsymbol{s}, \boldsymbol{u}), \tag{3}$$

where the map $\widehat{\cdot} : \mathbb{R}^3 \to \mathfrak{so}(3)$ is defined by

$$\widehat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -\omega^3 & \omega^2 \\ \omega^3 & 0 & -\omega^1 \\ -\omega^2 & \omega^1 & 0 \end{bmatrix},$$

Four basic thruster models are detailed next.

Example: full attitude control and one thruster along z-axis (ACS+Z). Consider a cubesat with a 3-axis ACS and a single thruster pointing in the negative z-axis and aligned with the center of mass. Figure 4a shows a model of this configuration while the actual hardware that can be employed to realized this system is shown in Figure 3. The control space is $U \subset \mathbb{R}^4$ where (u_1, u_2, u_3) are the torque inputs while u_4 is the thruster input. The set defined

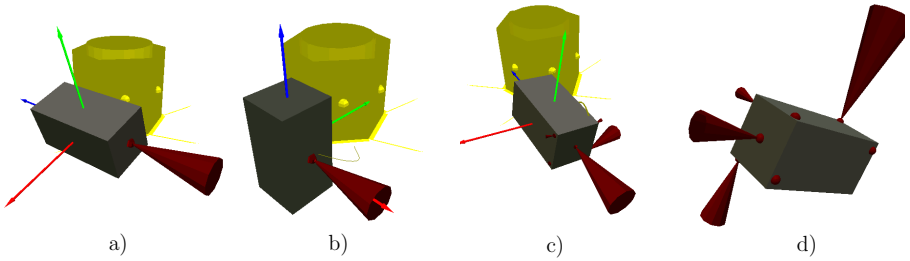


Figure 4: Various thruster configuration studied (drawn cones represent the magnitude of thruster firing): a) single thruster along z-axis (ACS+Z); b) two opposite thrusters along x-axis (ACS±X); c) five-thruster system (VACCO) also shown in Figure 3a; d) conceptual 10-thruster system (T10).

by $U = \{\boldsymbol{u} \in \mathbb{R}^4 \mid |u_i| < u_{\text{maxtorque}},\ \text{for } i = 1,2,3,\ u_{\text{minthrust}} \leq u_4 \leq u_{\text{maxthrust}}\}$ and the forces become

$$\boldsymbol{\tau}(\boldsymbol{s},\boldsymbol{u}) = (u_1,u_2,u_3), \quad \boldsymbol{f}(\boldsymbol{s},\boldsymbol{u}) = u_4 R\boldsymbol{e}_3, \tag{4}$$

where $\boldsymbol{e}_3 = (0,0,1) \in \mathbb{R}^3$. In practice, rather than directly controlling the attitude with given torques, the ACS often operates as closed-loop system for a given desired orientation and slew rate, i.e. the control inputs themselves can be regarded as function of current and desired state $\boldsymbol{u}(\boldsymbol{s},\boldsymbol{s}_d)$. For trajectory optimization purposes our model assumes that attitude is controlled through bounded torque inputs with additional angular velocity constraints corresponding to maximum slew rates achievable by the ACS.

Example: full attitude control and two opposite thrusters along x-axis (ACS±X) Cubesat with an ADC and two opposing thrusters along the x-axis aligned with the center of mass (see Figure 4b). The two thrusters can be modeled as a single input since they are never operated simultaneously. The thruster bound is then simply written as $u_{\text{minthrust}} \leq \|u_4\| \leq u_{\text{maxthrust}}$ and the control forces become

$$\boldsymbol{\tau}(\boldsymbol{s},\boldsymbol{u}) = (u_1,u_2,u_3), \quad \boldsymbol{f}(\boldsymbol{s},\boldsymbol{u}) = u_4 R\boldsymbol{e}_1, \tag{5}$$

where $\boldsymbol{e}_1 = (1,0,0) \in \mathbb{R}^3$.

Example: VACCO system without ACS. Consider the micro-thruster system [14] shown in Figure 3a with simulated model shown in Figure 4c. Assuming no additional ACS the control forces can be expressed as

$$\begin{bmatrix} \boldsymbol{\tau}(\boldsymbol{s},\boldsymbol{u}) \\ \boldsymbol{f}(\boldsymbol{s},\boldsymbol{u}) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix} B\boldsymbol{u}, \tag{6}$$

where $\mathbf{1}$ denotes identity matrix and the thruster allocation matrix takes the form

$$B = \begin{bmatrix} r_y \sin\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & r_z\cos\alpha + r_x \sin\alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & r_y\cos\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\alpha & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sin\alpha \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 & 1 & 0 \\ 1 & -1 & -1 & 1 & 0 \\ 1 & -1 & 1 & -1 & 0 \\ -1 & 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 0 \end{bmatrix},$$

where $r_x, r_y, r_z > 0$ are the offsets of the VACCO thruster valves from the center of mass in the x, y, and z axes respectively, and $\alpha = 15^o$ is the angle with which the four lateral thrusters are pointing in the $+z$-axis.

Note that although there are 5 inputs we have $\text{rank}(B) = 4$ signifying that only four degrees of freedom can be controlled simultaneously. This is evident considering that the second and forth row of the matrix are linearly dependent which physically corresponds to the fact that the VACCO system cannot simultaneously control rotation around the y-axis and translation in x-axis.

Example: fully actuated 10 thruster system (T10). Ideally, all degrees of freedom should be controllable. This can be accomplished by a variety of thruster configurations. One such conceptual configuration appropriate for cubesats is shown in Figure 4d and can be defined similarly to (6) but with matrix

$$B = \begin{bmatrix} 0 & -h & 0 & h & h & 0 & -h & 0 & 0 & 0 \\ h & 0 & -h & 0 & 0 & h & 0 & -h & 0 & 0 \\ a & a & a & a & -a & -a & -a & -a & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}, \tag{7}$$

where $a > 0$ and $h > 0$ are the lateral and vertical offsets of each of the side thrusters.

Constraints. In addition, the vehicle is subject to constraints arising from velocity limits, obstacles in the environment, and avoidance of instrument-sensitive orientations. These constraints are expressed through the m inequalities

$$F_i(\boldsymbol{s}(t), \boldsymbol{u}(t)) \geq 0 \tag{8}$$

for $i = 1, ..., m$. The simplest *velocity constraint* is to maintain a translational velocity below a given magnitude $v_{\max} > 0$ expressed as

$$F_1(\boldsymbol{s}, \boldsymbol{u}) = \|\dot{\boldsymbol{x}}\| - v_{\max}. \tag{9}$$

Obstacle constraints enforce that the vehicle must not collide with obstacles denoted by $\mathcal{O}_1, ..., \mathcal{O}_{n_o}$. Assume that the vehicle is occupying a region $\mathcal{A}(R, \boldsymbol{x}) \subset \mathbb{R}^3$, and let $\mathbf{prox}(\mathcal{A}_1, \mathcal{A}_2)$ be the Euclidean distance between two sets $\mathcal{A}_{1,2}$ that is negative in the case of intersection. Obstacle avoidance constraints in (8) can be written as

$$F_2(\boldsymbol{s}, \boldsymbol{u}) = \min_i \mathbf{prox}(\mathcal{A}(R, \boldsymbol{x}), \mathcal{O}_i), \text{ for all } t \in [0, T]. \tag{10}$$

We use a standard collision checking algorithm (Proximity Query Package [12]) to compute **prox**.

Objective. The goal is to compute the optimal controls $\boldsymbol{u}^*$ and final time T^* driving the system from its initial state $\boldsymbol{s}(0)$ to a given goal region $S_g \subset S$, i.e.

$$\begin{aligned} &(\boldsymbol{u}^*, T^*) = \arg\min_{\boldsymbol{u}, T} \int_0^T C(\boldsymbol{s}(t), \boldsymbol{u}(t))\text{dt}, \\ &\text{s.t. } \boldsymbol{s}(T^*) \in S_g \text{ while satisfying dynamics } (1) - (3) \text{ and constraints } (8), \end{aligned} \tag{11}$$

where $C : S \times U \to \mathbb{R}$ is a given cost function encoding e.g. fuel consumption or total time taken. For instance, a typical cost function includes the total time taken and fuel expended encoded by the L_1 control norm, i.e. $C(\boldsymbol{s}, \boldsymbol{u}) = 1 + \lambda|\boldsymbol{u}|_1$ for some constant $\lambda \geq 0$.

3 Trajectory Generation

The nonlinear optimization problem (11) has no closed form or easily computable solution in general. Nonlinear programming methods can be used to perform the optimization. Yet, such local variational methods are very sensitive to underactuated dynamics and obstacle constraints and require hand-tuned initialization and take long run-times. This precludes real-time performance and direct implementation on-board the spacecraft.

The complexity of the problem can be greatly simplified by exploiting certain properties of the dynamics. In particular, any given trajectory of a fully actuated system (e.g. T10 introduced in §2) can be inverted to compute the required control inputs and is feasibly as long as the inputs are not saturated. Similarly, underactuated systems (e.g. ACS+Z and ACS±X systems introduced in §2) can be handled by exploiting the *differential flatness* property of the control system.

The proposed strategy is to reduce the dimension of the system using such properties of the dynamics. A finite-dimensional trajectory parametrization mapping to the reduced space will then be constructed. The optimal parametrized trajectory will then be computed through global optimization over the reduced parameter space. The final solution will be obtained by mapping the parameters back to the original full trajectory space of the vehicle.

3.1 Underactuated Systems

Underactuated systems cannot track any arbitrary trajectory in their full configuration space. Instead, they can track only as many degrees of freedom simultaneously as the number of their independent control inputs. Differential flatness is an inherent geometric property of the system determining whether and how the remaining (unactuated) degrees of freedom can be controlled.

Differential Flatness. Intuitively, flatness means that the system admits a reduced order representation which uniquely determines the original higher-dimensional system. Thus motion planning and control can be performed in the reduced representation and the resulting solutions mapped backed to the original system. Skipping much of the flatness formalism (see e.g. [37] for an in-depth treatment) this work only considers a particular model – a spacecraft modeled as rigid body with full attitude control and at least one thruster generating force along a line passing through the center of mass.

The systems ACS+Z and ACS±X introduced in §2 are differentially flat in the sense that a given trajectory in position $\boldsymbol{x}(t)$ and a one-degree of freedom of a given orientation $R(t)$ uniquely determines the whole state of the system $\boldsymbol{s}$ and inputs $\boldsymbol{u}$. Intuitively, the spacecraft can execute a given trajectory in position space only by properly orienting itself and firing its single thruster appropriately.

Typically a rotational flat output corresponds to the angle of rotation around the thrust axis. For instance, if the thruster is along the Z-axis then $\boldsymbol{y} = (x, y, z, \psi)$, where ψ is the yaw of the body. Working with Euler angles though introduces singularities. Thus, instead of adopting local coordinates as in the standard approach to flatness the proposed method determines its attitude R to be as close as possible to a desired attitude R_d without coordinates. This is also the approach taken in [10] in the context of backstepping control of helicopters.

The developed control strategy employs the following functions. Recall that the exponential mapping $\exp : \mathbb{R}^3 \to SO(3)$ is defined by

$$\exp(\boldsymbol{\omega}) = \begin{cases} \mathbf{I}_3, & \text{if } \boldsymbol{\omega} = 0 \\ \mathbf{I}_3 + \frac{\sin\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|}\widehat{\boldsymbol{\omega}} + \frac{1-\cos\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|^2}\widehat{\boldsymbol{\omega}}^2, & \text{if } \boldsymbol{\omega} \neq 0 \end{cases}. \tag{12}$$

The right-trivialized tangent [16] map $\mathrm{dexp} : \mathbb{R}^3 \to L(\mathbb{R}^3, \mathbb{R}^3)$ is such that

$$\partial \exp(\boldsymbol{\omega}) \cdot \delta \cdot \exp(\boldsymbol{\omega}) = \widehat{\mathrm{dexp}(\boldsymbol{\omega})\delta} \tag{13}$$

and is defined by

$$\operatorname{dexp}(\boldsymbol{\omega}) = \begin{cases} \mathbf{I}_3, & \text{if } \boldsymbol{\omega} = 0 \\ \mathbf{I}_3 + \left(\frac{1-\cos\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|}\right)\frac{\widehat{\boldsymbol{\omega}}}{\|\boldsymbol{\omega}\|} + \left(1 - \frac{\sin\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|}\right)\frac{\widehat{\boldsymbol{\omega}}^2}{\|\boldsymbol{\omega}\|^2}, & \text{if } \boldsymbol{\omega} \neq 0 \end{cases} \tag{14}$$

With these definitions the procedure for generating trajectories for spacecraft with one thruster and full attitude control is stated as follows.

Proposition 1. *A spacecraft with control torques $\boldsymbol{\tau}(\boldsymbol{s}, \boldsymbol{u}) = \boldsymbol{u}_{1:3}$ and forces $\boldsymbol{f}(\boldsymbol{s}, \boldsymbol{u}) = u_4 R\boldsymbol{e}_i$ for some unit vector $\boldsymbol{e}_i \in \mathbb{R}^3$ can exactly follow a desired trajectory $\boldsymbol{x}_d : [0, T] \to \mathbb{R}^3$ and follow as close as possible a desired attitude $R_d : [0, T] \to SO(3)$ if the controls $\boldsymbol{u} : [0, T] \to \mathbb{R}^4$ satisfy:*

$$\boldsymbol{u}_{1:3} = \boldsymbol{\omega} \times \mathbb{J}\boldsymbol{\omega} + \mathbb{J}\dot{\boldsymbol{\omega}} - \boldsymbol{\tau}_{ext}, \tag{15}$$

$$u_4 = \boldsymbol{\alpha}^T R\boldsymbol{e}_i, \tag{16}$$

$$\boldsymbol{\omega} = \operatorname{dexp}(-\boldsymbol{\rho})\dot{\boldsymbol{\rho}} + \exp(-\boldsymbol{\rho})\boldsymbol{\omega}_d, \tag{17}$$

$$\boldsymbol{\alpha} = m\ddot{\boldsymbol{x}}_d - \boldsymbol{f}_{\text{ext}}, \quad \boldsymbol{b} = (R_d e_i) \times \boldsymbol{\alpha}/\|\boldsymbol{\alpha}\|, \quad \beta = \cos\left((R_d\boldsymbol{e}_i)^T\boldsymbol{\alpha}/\|\boldsymbol{\alpha}\|\right) \tag{18}$$

$$\boldsymbol{\rho} = \begin{cases} \arg\min_{\left\{\beta\frac{\boldsymbol{b}}{\|\boldsymbol{b}\|}, (\beta-\pi)\frac{\boldsymbol{b}}{\|\boldsymbol{b}\|}\right\}} \|\boldsymbol{u}_{1:3}\| & \text{if } \|\boldsymbol{b}\| > 0, \\ \mathbf{0}, & \text{if } \|\boldsymbol{b}\| = 0, \end{cases} \tag{19}$$

$$R = R_d \exp(\boldsymbol{\rho}), \tag{20}$$

where $R : [0, T] \to SO(3)$ denotes the resulting attitude while $\widehat{\boldsymbol{\omega}} = R^T\dot{R}$ and $\widehat{\boldsymbol{\omega}}_d = R_d^T\dot{R}_d$.

Note: the notation $\boldsymbol{\rho} = \arg\min_{\{\boldsymbol{\rho}_1, \boldsymbol{\rho}_2\}} \|\boldsymbol{u}_{1:3}\|$ means that $\boldsymbol{\rho}$ should be set to the one of the two arguments which results in minimum norm of the torque inputs.

Proof. The goal is to use align and fire the thruster so that the total force acting on the body is $m\ddot{\boldsymbol{x}}_d$ for all $t \in [0, T]$. This is accomplished by setting the attitude R so that $R\boldsymbol{e}_i$ is parallel to $\boldsymbol{\alpha} = m\ddot{\boldsymbol{x}}_d - \boldsymbol{f}_{\text{ext}}$. This restricts two degrees of freedom of the matrix R. The remaining one degree of freedom allows R to be chosen as close as possible to the desired R_d. This is accomplished by computing the smallest rotation that aligns $R_d\boldsymbol{e}_i$ with $\boldsymbol{\alpha}$. This rotation is defined by the vector $\boldsymbol{\rho}$ representing the exponential coordinates of the required transformation. Thus it is easy to check that

$$R_d \exp(\boldsymbol{\rho})\boldsymbol{e}_i = \pm\boldsymbol{\alpha},$$

where the $\pm$ sign depends on whether the thrust direction is flipped in order to reduce the required aligning torque (eq. (19)). Noting that

$$u_4 R\boldsymbol{e}_i = (\boldsymbol{\alpha}^T R\boldsymbol{e}_i)R\boldsymbol{e}_i = \left(\boldsymbol{\alpha}^T\frac{\pm\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|}\right)\frac{\pm\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|} = \boldsymbol{\alpha}$$

the total force becomes

$$m\ddot{\boldsymbol{x}} = u_4 R\boldsymbol{e}_i + \boldsymbol{f}_{\text{ext}} = \boldsymbol{\alpha} + \boldsymbol{f}_{\text{ext}} = m\ddot{\boldsymbol{x}}_d.$$

The required torque is then computing by noting that

$$\widehat{\boldsymbol{\omega}} = R^T\dot{R} = R^T\frac{d}{dt}(R_d\exp(\boldsymbol{\rho})) = R^T\dot{R}_d\exp(\boldsymbol{\rho}) + \exp(-\boldsymbol{\rho})\widehat{\operatorname{dexp}(\boldsymbol{\rho})}\dot{\rho}\exp(\boldsymbol{\rho}),$$

using the definition (13). Using the relationships $R\widehat{\boldsymbol{\omega}}R^T = \widehat{R\boldsymbol{\omega}}$ and $\exp(-\boldsymbol{\rho})\widehat{\operatorname{dexp}(\boldsymbol{\rho})}\boldsymbol{\delta}\exp(\boldsymbol{\rho}) = \widehat{\operatorname{dexp}(-\boldsymbol{\rho})\boldsymbol{\delta}}$ (see e.g. [13]) this reduced to

$$\boldsymbol{\omega} = \exp(-\boldsymbol{\rho})\boldsymbol{\omega}_d + \operatorname{dexp}(-\boldsymbol{\rho})\dot{\boldsymbol{\rho}}.$$

Finally, assuming full attitude control the required attitude $R(t)$ is achieved by setting $\boldsymbol{u}_{1:3} = \boldsymbol{\omega} \times \mathbb{J}\boldsymbol{\omega} + \mathbb{J}\frac{d}{dt}\boldsymbol{\omega}$. □

Control bounds. Note that the resulting control law might violate imposed bounds on the inputs. In particular, bounds on the thruster input depend on the accelerations $\ddot{\boldsymbol{x}}$ while bounds on the angular velocity and torques additionally depend on the higher derivatives $\dddot{\boldsymbol{x}}$, $\ddddot{\boldsymbol{x}}$ (see eq. (15) and (17)). The strategy employed in this work is to rescale the time along a given $x(t)$ in order to satisfy control bounds.

3.2 Fully Actuated System

Assume that the the control inputs have the form

$$\begin{bmatrix} \boldsymbol{\tau}(\boldsymbol{s},\boldsymbol{u}) \\ \boldsymbol{f}(\boldsymbol{s},\boldsymbol{u}) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix} B\boldsymbol{u},$$

where $\operatorname{rank}(B) = 6$ and $F = \{B\boldsymbol{u} \mid \boldsymbol{u} \in U\} \subset \mathbb{R}^6$ is an open set containing the origin.

In the fully-actuated case the system can follow any trajectory specified by both position and attitude, i.e. $(\boldsymbol{x}_d, R_d) : [0,T] \to \mathbb{R}^3 \times SO(3)$ as long as the required control inputs are not saturated. It is easy to show that this can be accomplished by setting the controls to

$$\boldsymbol{u} = (B^T B)^{-1} B^T \begin{bmatrix} \boldsymbol{\omega} \times \mathbb{J}\boldsymbol{\omega} + \mathbb{J}\dot{\boldsymbol{\omega}} - \boldsymbol{\tau}_{\text{ext}} \\ R^T(\ddot{\boldsymbol{x}} - \boldsymbol{f}_{\text{ext}}) \end{bmatrix}. \tag{21}$$

If the resulting controls exceed their bounds, the trajectory can be rescaled in time, i.e. T can be increased until $\boldsymbol{u} \in U$ is satisfied.

3.3 Trajectory Parametrization

Trajectories between two given boundary states $\boldsymbol{s}_0$ and $\boldsymbol{s}_g$ will be uniquely generated by selecting m intermediate "waypoints" through which the trajectory will smoothly pass. In the underactuated case the waypoints are the flat configurations, i.e. position $\boldsymbol{x}(t)$ and one degree of freedom of the attitude $R(t)$, while in the fully actuated case the waypoints are the full pose $(\boldsymbol{x}(t), R(t))$. The approach is to connect waypoints with the simplest possible representation ensuring feasible execution. Feasibility imposes certain smoothness conditions that are accomplished through polynomials of an appropriate order. The fully-actuated case is developed next followed by the underactuated case which requires higher-order smoothness conditions.

3.3.1 Fully Actuated Polynomial Curves

The parametrized trajectory space for fully-actuated systems is defined by

$$\mathcal{Z} = (\mathbb{R}^3 \times S^3)^m \subset \mathbb{R}^{7m},$$

with elements $\boldsymbol{z} = (\boldsymbol{q}_1, ..., \boldsymbol{q}_m)$, where $\boldsymbol{q}_i = (\boldsymbol{x}_i, \boldsymbol{r}_i)$ with $\boldsymbol{r} \in S^3$ denoting a unit quaternion corresponding to a given rotation matrix R. Quaternions are used for convenience since interpolation can be performed in the ambient vector space $\mathbb{R}^{7m}$ by initially ignoring the constraints $\boldsymbol{r} \in S^3$ and once the curve is constructed to project it back to $\mathcal{Z}$. Performing interpolation directly in $SO(3)$ is more involved and is avoided to maintain efficiency.

Consider the interpolation of a trajectory between two boundary conditions $(\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0)$ and $(\boldsymbol{q}_g, \dot{\boldsymbol{q}}_g)$ that must pass through intermediate waypoints $(\boldsymbol{q}_1, ..., \boldsymbol{q}_m)$. The boundary conditions, the waypoints, and enforcing second order smoothness at the waypoints (i.e. continuity of $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$) impose a total of $7(4+4m)$ conditions over $m+1$ segments. Thus the minimal representation is a cubic polynomial over each segment. Denoting $\Delta = T/(m+1)$ and $t_i = i\Delta$ the curve is defined by

$$\boldsymbol{q}(t) = C_i \boldsymbol{b}(t - i\Delta), \text{ when } t \in [t_i, t_{i+1}] \tag{22}$$

where $\boldsymbol{b} : [0, \Delta] \to \mathbb{R}^4$ is defined by $\boldsymbol{b}(t) = (t^3, t^2, t, 1)$.

Let $A = [\boldsymbol{b}(0)\ \dot{\boldsymbol{b}}(0)\ \ddot{\boldsymbol{b}}(0)\ \boldsymbol{b}(\Delta)]^{-1}$ and $E = [\boldsymbol{b}(\Delta)\ \dot{\boldsymbol{b}}(\Delta)\ \ddot{\boldsymbol{b}}(\Delta)]$. The coefficients C_i can be found using the following recursion

$$C_0 = [\boldsymbol{q}_0\ \dot{\boldsymbol{q}}_0\ \ddot{\boldsymbol{q}}_0\ \boldsymbol{q}_1]A, \tag{23}$$

$$C_i = [C_{i-1}E\ \ \boldsymbol{q}_{i+1}]A,\ \text{for } i = 1, ..., m, \tag{24}$$

assuming the equivalence $\boldsymbol{q}_{m+1} \equiv \boldsymbol{q}_g$. The initial acceleration $\ddot{\boldsymbol{q}}_0$ is unknown and is found by solving the linear equation

$$C_{m+1}\dot{\boldsymbol{b}}(T) = \dot{\boldsymbol{q}}_g \tag{25}$$

for $\ddot{\boldsymbol{q}}_0$ after which it is then substituted back into each matrix C_i. The unknown coefficients are linear function of the data and can be easily precomputed for any given m. This enables instant trajectory generation for any given boundary conditions and waypoints.

3.3.2 Underactuated Polynomial Curves

The parameters space in the underactuated case is defined by

$$\mathcal{Z} = (\mathbb{R}^3 \times S^1)^m \subset \mathbb{R}^{5m}$$

with elements $\boldsymbol{z} = (\boldsymbol{q}_1, ..., \boldsymbol{q}_m)$, where $\boldsymbol{q}_i = (\boldsymbol{x}_i, \boldsymbol{r}_i)$ where $\boldsymbol{r} \in S^1$ is a unit vector. Thus similarly to the fully-actuated case interpolation is performed in the ambient space $\mathbb{R}^{5m}$ and projected back to $\mathcal{Z}$.

Similarly to the fully-actuated case boundary conditions are given in terms of $\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0$ and $\boldsymbol{q}_g, \dot{\boldsymbol{q}}_g$ and the trajectory must pass through the intermediate spinets $(\boldsymbol{q}_1, ..., \boldsymbol{q}_m)$. Note that it is necessary to enforce third-order smoothness in position (i.e. continuous $\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \dddot{\boldsymbol{x}}$) since by Proposition 1 the resulting angular velocity $\boldsymbol{\omega}$ is a smooth function of $\dddot{\boldsymbol{x}}$. There are a total of $3(4 + 5m)$ position conditions over the $m + 1$ segments. Thus the minimal representation is a cubic polynomial over one of the segments (e.g. the first) and a fourth-order polynomial over all remaining m segments. The rotational degree of freedom is interpolated using a lower order polynomial similarly to the fully-actuated case. The trajectory is expressed as

$$\boldsymbol{x}(t) = \begin{cases} C_0\boldsymbol{b}(t), & \text{when } t \in [0, \Delta], \\ C_i\boldsymbol{b}_4(t - i\Delta), & \text{when } t \in [t_i, t_{i+1}],\ \text{for } i = 1, ..., m, \end{cases} \tag{26}$$

where $\boldsymbol{b}_4(t) = (t^4, t^3, t^2, t, 1)$.

Similarly to §3.3.1 let $A_4 = [\boldsymbol{b}(0)\ \dot{\boldsymbol{b}}(0)\ \ddot{\boldsymbol{b}}(0)\ \dddot{\boldsymbol{b}}(0)\ \boldsymbol{b}(\Delta)]^{-1}$ and $E_4 = [\boldsymbol{b}(\Delta)\ \dot{\boldsymbol{b}}(\Delta)\ \ddot{\boldsymbol{b}}(\Delta)\dddot{\boldsymbol{b}}(\Delta)]$. The coefficients C_i can be found using the following recursion

$$C_0 = [\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0, \ddot{\boldsymbol{x}}_0, \boldsymbol{x}_1]A_3, \tag{27}$$

$$C_1 = [C_0E_3\ \ \boldsymbol{x}_2]A_4, \tag{28}$$

$$C_i = [C_{i-1}E_4\ \boldsymbol{x}_{i+1}]A_4,\ \text{for } i = 2, ..., m, \tag{29}$$

Similarly to §3.3.1 the initial acceleration $\ddot{\boldsymbol{x}}_0$ is unknown and is found by solving the linear equation

$$C_{m+1}\dot{\boldsymbol{b}}_4(T) = \dot{\boldsymbol{x}}_g \tag{30}$$

after which it is then substituted back into the polynomial coefficient matrices C_i.

3.4 Reconstruction

A trajectory represented by a parameter $\boldsymbol{z} \in \mathcal{Z}$ can be mapped to a curve $\boldsymbol{q}(t)$ for all $t \in [0, T]$ using the polynomial interpolations (22) and (26) in the fully and under-actuated cases, respectively. These poses can then be mapped back to the full trajectory and control inputs

$(\boldsymbol{s}, \boldsymbol{u}) : [0, T] \to S \times U$ as described in §3.1 for the underactuated case and in §3.2 for fully-actuated systems. Thus the infinite dimensional space of continuous trajectories has been encoded through a low-order finite dimensional space $\mathcal{Z}$. Next, global optimization will be performed over this space to find the optimal parameter $\boldsymbol{z}^* \in \mathcal{Z}$ that will correspond to an approximately optimal trajectory $\boldsymbol{s}^* : [0, T] \to S$.

4 Motion Planning

The constrained optimal control problem (11) is solved performing global optimization over the trajectory parameter space $\mathcal{Z}$. This is accomplished through the recently proposed cross-entropy motion planning method [15]. The method was chosen for its ability to globally explore the solution space and converge to high-quality solutions, its simple gradient-free implementation, and the possibility to use it in any-time fashion on-board the vehicle. The main limitation of the method is that it cannot handle environments that are very cluttered.

The main idea is to construct a probability distribution over the trajectory space, to sample trajectories from the distribution, evaluate their costs, and adapt the distribution so that it becomes concentrated over high-quality trajectory space regions. The process is repeated iteratively until the distribution has collapsed close to a delta function near the optimum [26]. Note that many of the sampled trajectories will violate the problem constraints (8) such as obstacles. The space of allowed parametrized trajectories that satisfy these constraints is defined by $\mathcal{Z}_{\text{con}} \subset \mathcal{Z}$ and only trajectories $z \in \mathcal{Z}_{\text{con}}$ will be used to update the distribution.

Let Z be a random variable over $\mathcal{Z}$ with density $p(Z; v)$, where $v \in \mathcal{V}$ is the density parameter. Let $n_z = \dim(\mathcal{Z})$. The parameter space is $\mathcal{V} = \left(\mathbb{R}^{n_z} \times \mathbb{R}^{({n_z}^2+n_z)/2}\right)^K \times \mathbb{R}^K$ with elements $v = (\mu_1, \Sigma_1, ..., \mu_K, \Sigma_K, w_1, ..., w_K)$ corresponding to K mixture components with means μ_k, covariance matrices Σ_k (excluding identical elements due to the matrix symmetry) and weights w_k. A mixture of Gaussians is employed defined by

$$p(z; v) = \sum_{k=1}^{K} \frac{w_k}{\sqrt{(2\pi)^{n_z}|\Sigma_k|}} e^{-\frac{1}{2}(z-\mu_k)^T \Sigma_k^{-1}(z-\mu_k)}, \tag{31}$$

where $\sum_{k=1}^{K} w_k = 1$. The number of mixture components K is chosen adaptively (see e.g. [8]). Even the simple case $K = 1$ is sufficient for solving complex multi-extremal problems as long as enough samples are obtained. The complete algorithm adapted to the setting of this work is summarized below (see [15] for complete details).

Algorithm 4.1. CE Motion Planning

Initialization:

0.1 Compute the optimal trajectory reaching the goal by ignoring the constraints, i.e. $z^* = \min_{z \in \mathcal{Z}} J(z)$

0.2 Set matrix Σ so that the region $\{z \in \mathcal{Z} \mid (z - z^*)^T \Sigma (z - z^*) < 2,\ 0 \le t \le T\} \subset \mathcal{X}$ covers the reachable configuration space of interest

0.3 Choose initial samples $Z_1, ..., Z_N$ from $\texttt{Normal}(z^*, \Sigma)$; set $j = 0$ and $\hat{\gamma}_0 = \infty$

Iteration:

1. Update distribution
$$\hat{v}_j = \operatorname*{argmin}_{v \in \mathcal{V}} \frac{1}{|\mathcal{E}_j|} \sum_{Z_k \in \mathcal{E}_j} \ln p(Z_k; v),$$
over the elite set $\mathcal{E}_j = \{Z_i \mid J(Z_i) \le \hat{\gamma}_j\}$
2. Generate samples $Z_1, ..., Z_N$ from $p(\cdot, v_j)|_{\mathcal{Z}_{\text{con}}}$ and compute the ϱ–th quantile $\hat{\gamma}_{j+1} = J_{\lceil \varrho N \rceil}$
3. If $j > 0$ and $\text{KL}(p(\cdot, v_{j-1}) || p(\cdot, v_j)) < \epsilon$ then finish, otherwise set $j = j + 1$ and goto step (1)

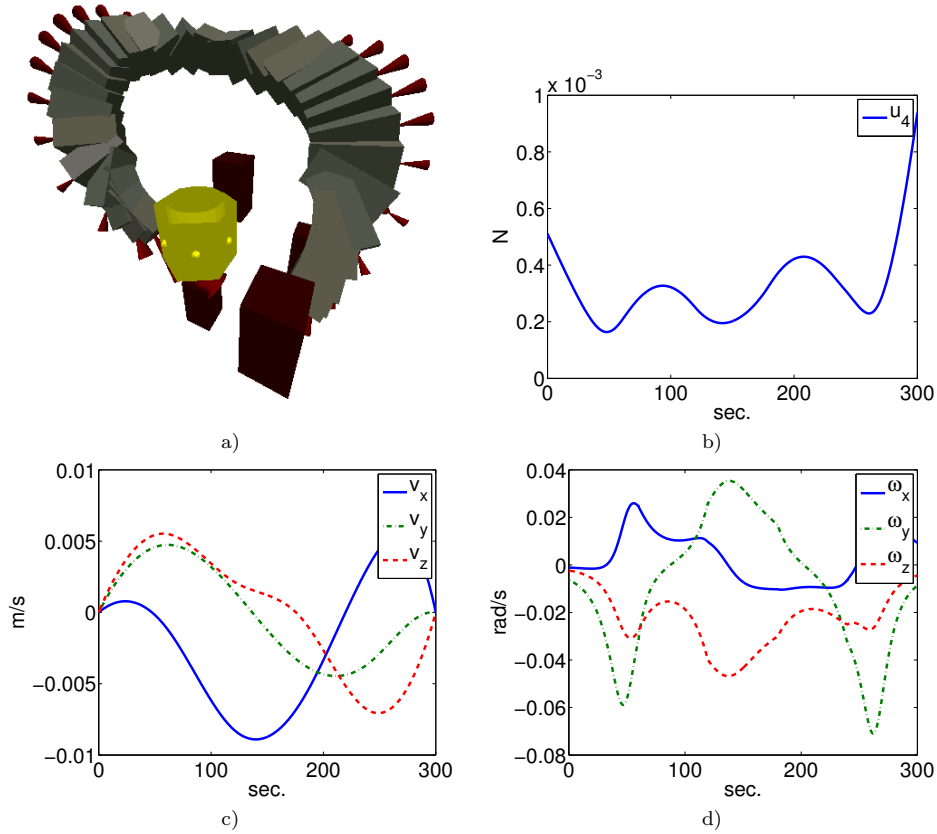


Figure 5: Reconfiguration of the ACS+Z underactuated cubesat: a) resulting path to the goal state on the opposite side of the larger spacecraft in the middle; b) force produced by the single thruster acting along the +Z axis; c) translational velocities along the path; d) angular velocities along the path.

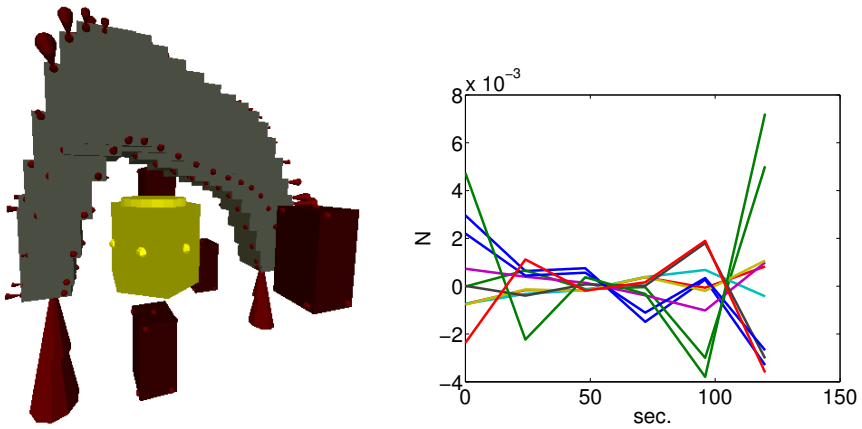
The algorithm performs very efficiently when the environment is not very complex. If the optimal trajectory lies in a narrow passage that is very unlikely to be sampled then the algorithm will be unlikely to find it. Such complex scenarios are handled by combining the proposed method with standard motion planning techniques as detailed in [15].

5 Applications

Two simulated scenarios are developed next in order to test the proposed methodology–a reconfiguration and a surface sampling tasks. In both cases two types of cubesats are employed: an uneractuated with a single thruster and attitude control and a fully-actuated with ten thruster configuration.

5.1 Reconfiguration

The reconfiguration task requires a cubesat to navigate around another larger spacecraft and a few other floating cubesats and arrive optimally to a designated zero-velocity state. Figure 5



a) b)

Figure 6: Reconfiguration of the T10 fully-actuated cubesat: a) resulting path to the goal state on the appositive side of the larger spacecraft in the middle; b) forces produced by the ten thrusters.

shows the resulting trajectory and controls for the underactuated cubesat ACS+Z (defined in (4)).

The cost function used for ACS+Z system is

$$C(\boldsymbol{s}, \boldsymbol{u}) = |u_4| + \|\boldsymbol{u}_{1:3}\|,$$

in order to encode fuel expended by the thruster but also penalize excessive torque required by the ACS. The trajectory required $\Delta V = 0.089$ m/s and momentum change 0.035Nms. A fixed duration of $T = 300$s. was chosen a priori which resulted in forces and torques magnitudes that can be achieved by thrusters such as in Figure 3a-c and ACS (Figure 3d). The computation was based on trajectories with $m = 4$ waypoints and collisions checking was performed every $h = .5$s. along the path using the PQP package. The CE optimization method was run with a single Gaussian and $N = 1000$ samples. The total computation time for this scenario lasted 30 seconds and required 11 iterations.

Similarly, Figure 6 shows the resulting trajectory and controls for the fully actuated cubesat T10 (defined in (7)). The cost function is the total fuel $C(\boldsymbol{s}, \boldsymbol{u}) = |\boldsymbol{u}|$ expended by all ten thrusters over a fixed time horizon of $T = 120$s. Since attitude is controlled by thrusters this maneuver can be executed in a shorter time than the ACS+Z system without violating bounds on the inputs.

5.2 Asteroid Sampling

The second scenario is mock-up sampling maneuver on the sufrace of a small asteroid. Motivated by recent mission such MUSES-C [17] and DAWN [22] the proposed methodology can be used to enhance autonomy and sampling capabilities. For instance, a cubesat can be carried on-board a larger spacecraft and dispatched to various sampling points.

Note that we only consider the near-surface navigation problem rather than the complete asteroid trajectory transfer which can be very complex. Once the spacecraft is in a hovering state a few meters above the surface it can perform autonomous motion planning to various points of interest. Unlike the LEO reconfiguratione example (§5.1) the external disturbances in this example dominated by microgravity, i.e. $\boldsymbol{f}_{\text{ext}}$ should additionally encode the asteroid gravitational effects.

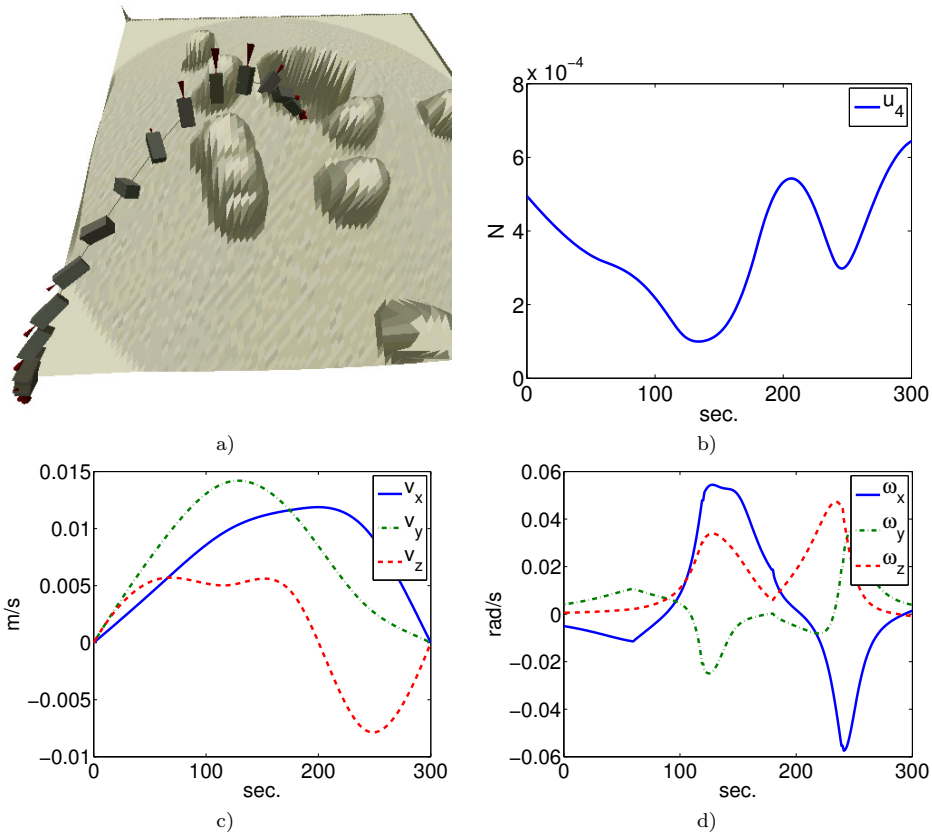


Figure 7: Asteroid crater sampling using the ACS+Z underactuated cubesat: a) optimized path to the bottom of crater; b) resulting force produced by the single thruster along the z-axis; c) resulting translational velocities along the path; d) angular velocities along the path.

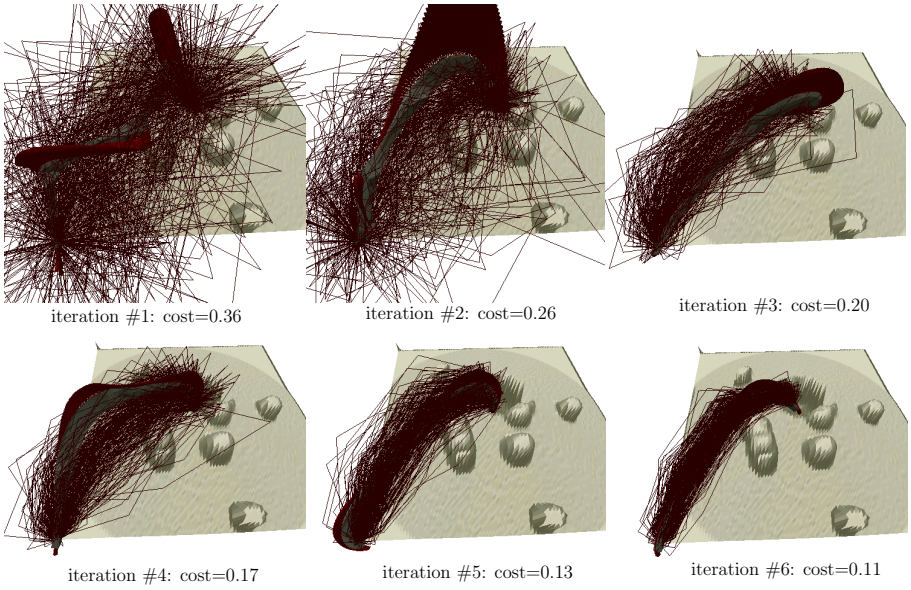


Figure 8: Several iterations of the cross-entropy optimization method (§4) applied to the sampling scenario. The thin trajectories correspond to samples from parametrized trajectory space $\mathcal{Z}$. As the algorithm iterates the distribution and its samples concentrate close to the optimum.

Figure 7 shows the optimized obstacle-free trajectory taking the underactuated spacecraft to the bottom of a small crater with zero final velocity. The algorithm is setup similarly to §**??**. The trajectory shown is optimized for 10 CE iterations taking a total of 45 seconds of computation and resulted in $\Delta V = 0.126 m/s$ and total moment 0.051 Nms. Figure 8 shows the first several iterations of the stochastic optimization method. The random thin lines connect waypoints that compose each sample Z_i. Note that the actual trajectory corresponding to Z_i is smooth and feasible but is not reconstructed and drawn for efficiency. The algorithm typically requires around ten iterations to produce a reasonable solution. It should be noted though that it is difficult to claim how far from the optimal this solution is. The higher N and m are the higher chance is to reach the true optimum but at an increased computational effort.

Similarly, results for the fully actuated cubesat are shown on Figure 9 with a few iterations of the optimization shown on Figure 10. Unlike the underactuated case, the resulting motion for the T10 system can be accomplished without much change in the spacecraft attitude. The scenario was setup similarly to the previous example and required 60 seconds for computing the shown trajectory.

6 Experiments

A simple testbed was developed in order to study the proposed techniques. It consists of an air-hockey type floatation table which supports spacecraft on pucks. Engineering models of cubesats were developed with components necessary for testing reconfiguration maneuvers. The components include an electromagnetic docking system, a visual pose estimation (VPE) system, a mock-up propulsion system, and a high-level computer with wireless communication. Since at the time of developing this project no low-cost propulsion system for cubesats was readily available we designed a system with six micro electric ducted fans that can generate forces down to one milli-Newton. They are used to simulate either underactuated or fully-

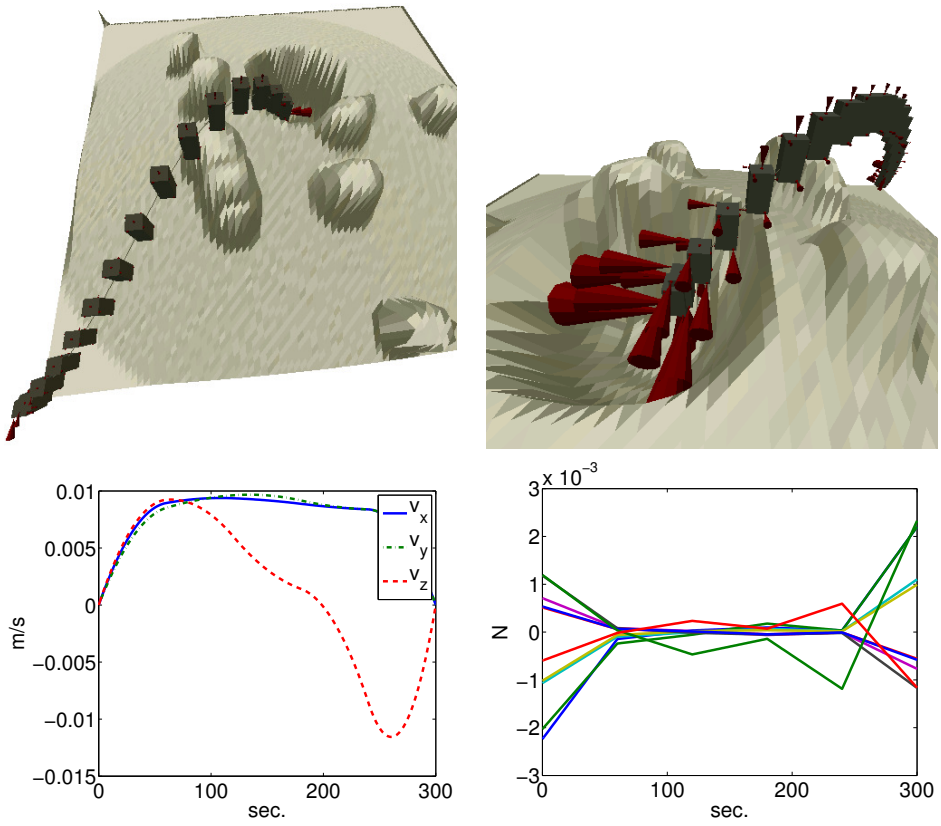


Figure 9: Asteroid crater sampling using the T10 fully actuated cubesat: a) optimized path to the bottom of crater; b) close-up viewl c) resulting translational velocities along the path; d) resulting forces produced by the ten thrusters.

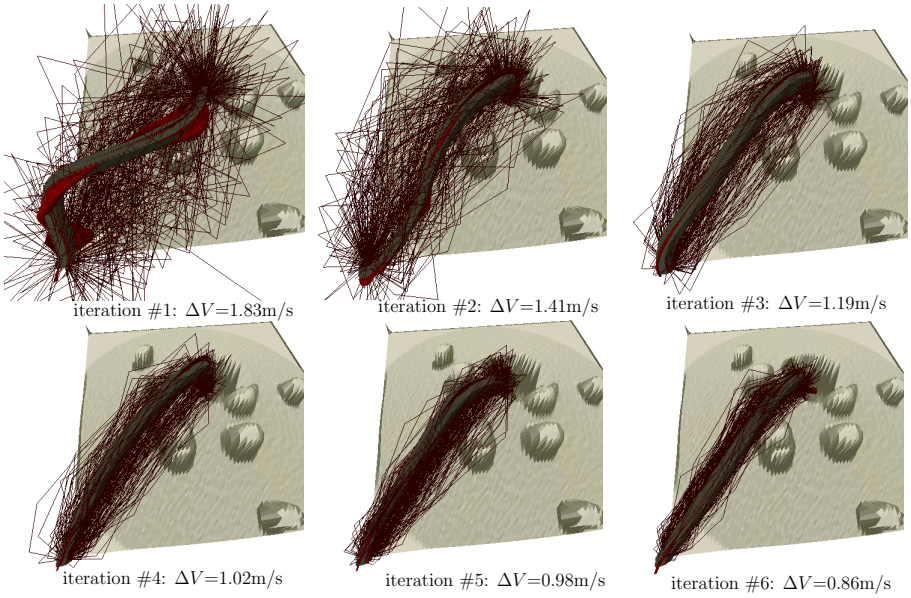


Figure 10: Iterations of the CE method applied to the fully-actuated cubesat performing asteroid sampling.

Component	Qty	Description	Weight	Power
cubesat frame	1	ISIS 3U	200g	n/a
power	1	8.4 NiCad battery or ClydeSpace PSU	100g	n/a
electromagnets	2	custom made at Univ. of Surrey	200g	3W
camera	1	PointGrey FireFly USB	30g	1.5W
fans	6	brushless EDF	12g	0.1 - 10 W
control board	1	controls fans and magnets	20g	0.1W
high-level computer	1	Gumstix Overo 600MHz	8g	1W
comms	1	i2c, wifi+bluetooth, antennas	10g	0.9W

Figure 11: Details of the engineering cubesat model used for performing reconfiguration experiments.

actuated propulsion. The VPE system is based on instrumenting each spacecraft with a unique configuration of bright visual markers (LEDs) and performing full six degrees of freedom pose estimation through LED image coordinates extraction and pattern matching.

A cubesat-compatible control board was designed which interfaces to standard cubesat subsystems through i2c and power interfaces. The board controls the fans, the magnets, the cameras and the visual markers (LEDs), and hosts the high-level computer. Table 11 provides more details about the engineering model.

The proposed motion planning algorithm was implemented and tested on the air table using a mock-up obstacle environment. Figure 13 shows an example setup and a few frames along the computed motion. Here an earlier prototype of the robotic cubesat is shown. Since the environment was simple and the problem in 2-D the proposed algorithm can be implemented very efficiently. It runs in real-time at 20Hz so that there is no need for additional trajectory tracking. Figure 12 shows the assembled robotic surrogate and a few frames along a docking motion using visual feedback from the on-board camera.

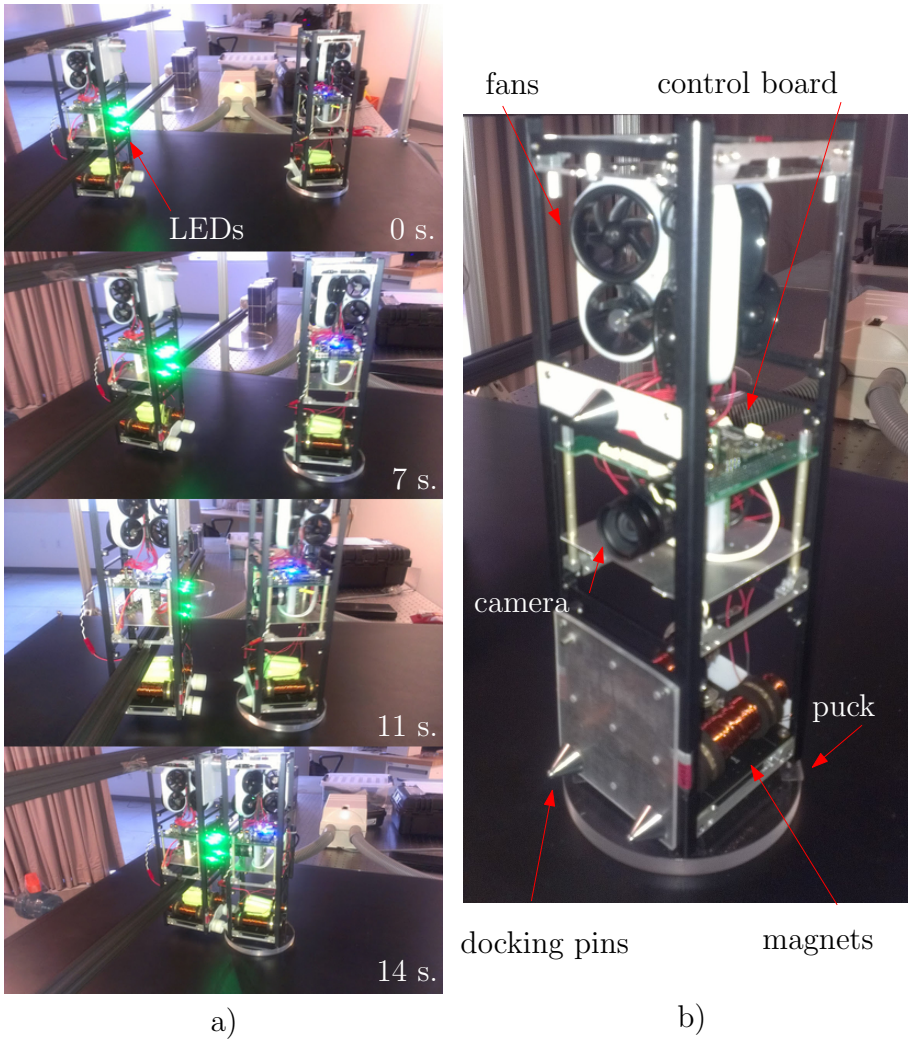


Figure 12: a) frames along a docking motion using visual pose estimation b) a close-up view of the model

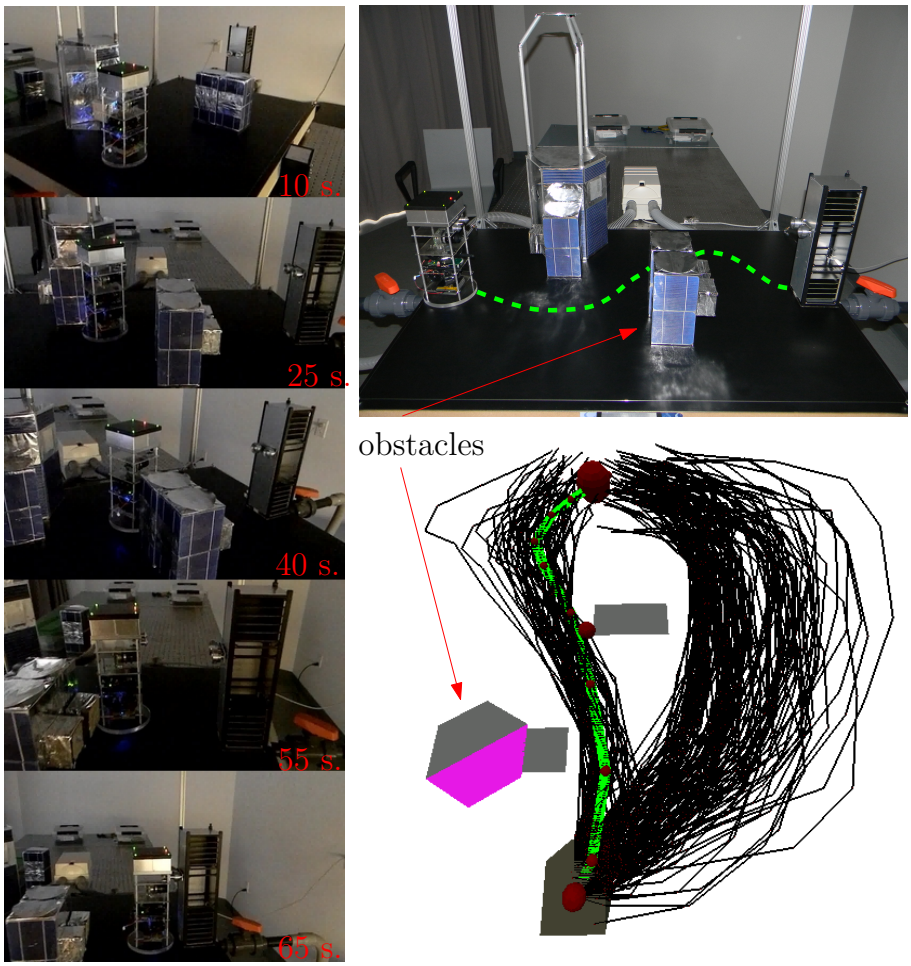


Figure 13: a) frames along a trajectory of a cubesat engineering model that avoids obstacles and performs docking b) the setup (top) and an output of the motion planning algorithm showing the best path to follow (thicker line)

7 Conclusion

This work studied the trajectory planning problem for small spacecraft proximity operations. Motivated by the limitations of current technology the proposed method considers a principled way for computing near-optimal motions that account for underactuation, obstacle avoidance, and orbital and rigid body dynamics. An efficient algorithm capable of computing near-optimal solutions is constructed. It exploits the nature of dynamics to for instant trajectory generation based on reduced-order parametrization and performs stochastic optimization for globally searching for a high-quality trajectory satisfying all given constraints. The main limitations of the method lie in the lack of a formal procedure for selecting the optimal finite dimensional resolution (i.e. number of waypoints) for a given scenario and in its inability to handle very cluttered environments.

The algorithm operates in real-time on-board the experimental cubesats in a simplified 2-D air-table setting. In the context of more complex simulated examples in 3-D, the current implementation requires several seconds to produce high-quality solutions and tens of seconds to converge near a (local) optimum. Through parallelization and more informed sampling it is reasonable that the algorithm complexity can be significantly reduced to enable real-time on-board implementation.

References

[1] B. Acikmese, D. P. Scharf, E. Murray, and F. Y. Hadaegh. A convex guidance algorithm for formation reconguration. In *American Control Conference, 2006. Proceedings of the 2003*, 2003.

[2] A Aguiar and J P Hespanha. Position tracking of underactuated vehicles. *Proceedings Of The American Control Conference*, 3:2–7, 2003.

[3] G. S. Aoude, J. P. How, and D. W. Miller. Reconfiguration maneuver experiments using the spheres testbed onboard the iss. In *Proceedings of the 3rd International Symposium on Formation Flying, Missions and Technologies*, 2008.

[4] L. Breger and J. P. How. Safe trajectories for autonomous rendezvous of spacecraft. *AIAA Journal on Guidance, Control, and Dynamics*, 31(5):1478–1489, 2008.

[5] F. Bullo, N.E. Leonard, and A.D. Lewis. Controllability and motion algorithms for underactuated lagrangian systems on Lie groups. *IEEE Transactions on Automatic Control*, 45(8):1437 – 1454, 2000.

[6] F. Bullo and K. M. Lynch. Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems. *IEEE Transactions on Robotics and Automation*, 17(4):402–412, 2001.

[7] Nadeem Faiz, Sunil Agrawal, and Richard Murray. Differentially flat systems with inequality constraints: An approach to real-time feasible trajectory generation. *Journal of Guidance, Control, and Dynamics*, 24(2):219–227, 2001.

[8] M. A. F. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[9] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, dec 2005.

[10] E. Frazzoli, M.A. Dahleh, and E. Feron. Trajectory tracking control design for autonomous helicopters using a backstepping algorithm. In *Proc. American Control Conf.*, pages 4102–4107, Chicago, IL, June 2000.

[11] Emilio Frazzoli. Quasi-random algorithms for real-time spacecraft motion planning and coordination. *Acta Astronautica*, 53(410):485 – 495, 2003.

[12] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 30:171–180, 1996.

[13] E. Hairer, Ch. Lubich, and G. Wanner. *Geometric Numerical Integration*. Number 31 in Springer Series in Computational Mathematics. Springer-Verlag, 2006.

[14] VACCO Industries. ChEMS micro-propulsion system.

[15] M. Kobilarov. Cross-entropy motion planning. *International Journal of Robotics Research*, 31(7):855–871, 2012.

[16] M. Kobilarov and J. Marsden. Discrete geometric optimal control on Lie groups. *IEEE Transactions on Robotics*, 27(4):641–655, 2011.

[17] Takashi Kubota, Shujiro Sawai, Tatsuaki Hashimoto, Junichiro Kawaguchi, and Akira Fujiwara. Robotics technology for asteroid sample return mission muses-c. In *Proceeding of the 6th International Symposium on Artificial Intelligence and Robotics and Automation in Space: i-SAIRAS*, 2001.

[18] Christophe Louembet. Collision avoidance in low thrust rendezvous guidance using flatness and positive b-splines. In *American Control Conference*, 2011.

[19] David W. Miller, Swati Mohan, and Jason Budinoff. Assembly of a large modular optical telescope (ALMOST). In *SPIE Space Telescopes and Instrumentation*, 2008.

[20] Juergen Mueller, Richard Hofer, and John Ziemer. Survey of propulsion technologies applicable to cubesats. Technical report, Jet Propulsion Laboratory, 2010.

[21] R. M. Murray, M. Rathinam, and W. M. Sluis. Differential flatness of mechanical control systems. In *Proceedings ASME International Congress and Exposition*, 1995.

[22] NASA/JPL. Dawn: http://www.nasa.gov/dawn.

[23] Jeff M. Phillips, L. E. Kavraki, and N. Bedrosian. Probabilistic optimization applied to spacecraft rendezvous and docking. In *13th American Astronomical Society/AIAA - Space Flight Mechanics Meeting*, Puerto Rico, February 2003.

[24] Wirz R. and Conversano R. Cubesat lunar mission using a miniature ion thruster. In *47th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, pages AIAA–2011–6083, 2011.

[25] A. Richards, T. Schouwenaars, J. P. How, and E. Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *AIAA Journal on Guidance, Control, and Dynamics*, 25(4):755–764, July-August 2002.

[26] Reuven Y. Rubenstein and Dirk P. Kroese. *Simulation and the Monte Carlo Method*. Wiley, 2008.

[27] Alvar Saenz-Otero and David W. Miller. Spheres: a platform for formation-flight research. volume 5899, page 58990O. SPIE, 2005.

[28] D. Scharf, F. Hadaegh, , and B. Kang. A survey of spacecraft formation flying guidance. In *Proceedings of the Intl. Symposium Formation Flying*, 2002.

[29] Daniel P. Scharf, Fred Y. Hadaegh, Zahidul H. Rahman, Joel F. Shields, and Gurkipal Singh. An Overview of the Formation and Attitude Control System for the Terrestrial Planet Finder Formation Flying Interferometer. In *International Symposium on Formation Flying Missions and Technologies, 14 Sep. 2004, Washington, DC, United States*, 2004.

[30] D.P. Scharf, F.Y. Hadaegh, and S.R. Ploen. A survey of spacecraft formation flying guidance and control (part 1): guidance. In *American Control Conference, 2003. Proceedings of the 2003*, volume 2, pages 1733 – 1739, jun 2003.

[31] Derek T. Schmuland, Robert K. Masse, and Charles G. Sota. Hydrazine propulsion module for cubesats. In *Small Satellite Conference*, 2011.

[32] Cornel Sultan, Sanjeev Seereram, and Raman K. Mehra. Deep space formation flying spacecraft path planning. *Int. J. Rob. Res.*, 26(4):405–430, April 2007.

[33] Panagiotis Tsiotras. Feasible trajectory generation for underactuated spacecraft using differential flatness. In *AAS/AIAA Astrodynamics Specialist Conference*, 1999.

[34] Brent E. Tweddle, Alvar Saenz-Otero, and David W. Miller. Design and development of a visual navigation testbed for spacecraft proximity operations. In *AIAA SPACE 2009 Conference and Exposition*, 2009.

[35] C. Underwood and S. Pellegrino. Autonomous assembly of a reconfigurable space telescope (aarest) for astronomy and earth observation. In *8th IAA Symposium on Small Satellites for Earth Observation*, 2011.

[36] David Vallado. *Fundamentals of Astrodynamics and Applications*. Primis, 1997.

[37] Michiel J Van Nieuwstadt and Richard M Murray. Real time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control*, 8(11):995–1020, 1998.

ijrr

The International Journal of
Robotics Research
31(7) 855–871

DOI: 10.1177/0278364912444543
ijr.sagepub.com
SAGE

Cross-entropy motion planning

Marin Kobilarov

Abstract

This paper is concerned with motion planning for non-linear robotic systems operating in constrained environments. A method for computing high-quality trajectories is proposed building upon recent developments in sampling-based motion planning and stochastic optimization. The idea is to equip sampling-based methods with a probabilistic model that serves as a sampling distribution and to incrementally update the model during planning using data collected by the algorithm. At the core of the approach lies the cross-entropy method for the estimation of rare-event probabilities. The cross-entropy method is combined with recent optimal motion planning methods such as the rapidly exploring random trees (RRT) in order to handle complex environments. The main goal is to provide a framework for consistent adaptive sampling that correlates the spatial structure of trajectories and their computed costs in order to improve the performance of existing planning methods.*

Keywords

motion planning, nonlinear control, importance sampling, cross-entropy, stochastic optimization, RRT

1. Introduction

Consider an agile robotic vehicle navigating in a natural environment. The vehicle motion is constrained due to its kinematics and dynamics and due to obstacles in the environment. The task is to compute an open-loop trajectory that reaches a desired goal region optimally under the assumption that perfect models of the robot and the environment are available. Motion planning is a key requirement for autonomous systems and solving this problem is of central importance in robotics.

In general, the problem cannot be solved in closed form since both the dynamics and constraints can be nonlinear. Gradient-based optimization is not suitable unless a good starting guess is chosen since the constraints impose many local minima. In addition, constraints corresponding to arbitrary obstacles can be non-smooth and require special differentiation (Clarke et al. 1998) to guarantee convergence. An alternative is to discretize the vehicle state space, e.g. using a grid and generate candidate paths by transitioning between adjacent cells. Such an approach is computationally intractable if the state space has more than a few dimensions and is limited to systems with very simple dynamics, e.g. an unconstrained point mass in the plane. This is due to the exponential (both in state dimension and planning horizon) size of the search space, also known as the 'curse of dimensionality'.

Since the motion planning problem is computationally NP-complete in general (LaValle 2006) one has to resort to approximation algorithms. Sampling-based motion planning has become an established methodology in this context. The basic idea is to construct a graph structure with nodes corresponding to states and with edges satisfying the dynamics and constraints. In essence, the graph is regarded as a finite approximation of the infinite set of feasible trajectories. The optimal control problem is then solved approximately through graph search. The two main families of such methods are rapidly exploring dense trees (RDT) (LaValle 2006) and probabilistic roadmaps (PRM) (Choset et al. 2005). Such sampling-based methods are probabilistically complete, i.e. the probability of failing to find a solution, if it exists, approaches zero at least asymptotically in the number of iterations.

The property of sampling-based methods of interest in this work is optimality. Standard rapidly exploring random trees (RRT) methods can quickly explore the space but typically provide a solution far from optimal while standard PRM methods asymptotically approach the optimal

California Institute of Technology, 2543 Wellesley Avenue, Los Angeles, CA

Corresponding author:
Marin Kobilarov, California Institute of Technology, 2543 Wellesley Avenue, Los Angeles, CA 90064, USA.
Email: marin@cds.caltech.edu

trajectory but at an exponentially slow rate (due to the increased amount of vertices and edges) which in higher dimensions becomes computationally intractable. The optimal RRT (RRT*) and optimal rapidly exploring random graph (RRG*) were recently proposed (Karaman and Frazzoli 2011) to overcome some of these limitations while retaining probabilistic completeness. The basic idea is to maintain the standard RRT exploratory properties while rewiring the structure until it satisfies local dynamic programming conditions on the set of vertices which result in asymptotic optimality.

The performance of sampling-based methods, regarding optimality, can be further improved from a different viewpoint. In particular, typically much effort is wasted in sampling nodes from parts of the state space that are unlikely to improve the current solution. Instead, it is possible to sample from a probabilistic model that incrementally identifies promising regions of the state space using the costs of trajectories that have already reached the goal. The purpose of this paper is to construct a method that combines sampling-based methods with such an adaptive sampling approach in order to exploit the collected information about the optimality of trajectories.

Employing adaptive or biased sampling is not new in motion planning. Such ideas have lead to more efficient algorithms using guided sampling to reduce the chance of colliding edges and accelerate state space exploration in order to find solutions more efficiently (e.g. Burns and Brock 2005b; Hsu et al. 2006; Kalisiak and van de Panne 2007; Li and Bekris 2010; Knepper and Mason 2011). In addition, several methods have achieved significant improvements by exploiting workspace information (Kurniawati and Hsu 2004), learning from previous motion planning runs (Hsu et al. 2005), or exploiting rather than discarding colliding samples (Denny and Amato 2011). A related viewpoint is to adaptively balance between exploring the space and exploiting collected information about collisions (Ladd and Kavraki 2005; Rickert et al. 2008). The unifying idea behind most of these methods is the construction of a deterministic or a probabilistic model (Burns and Brock 2005a; Zucker et al. 2008) that can be adjusted before and during execution to improve planning performance.

At the same time, probabilistic models also serve as a basis for stochastic optimization (Spall 2003). Among the many flavors and applications of stochastic optimization we mention methods based on global models incorporating past data (e.g. Moore and Schneider (1995) and Atkeson et al. (1997) in the context of plant optimization) or local stochastic gradient-based algorithms (Powell 2007) which include several recently developed methods for robotic trajectory optimization among obstacles (Ratliff et al. 2009; Theodorou et al. 2010; Kalakrishnan et al. 2011).

The idea behind the methods that we propose is to employ stochastic optimization of probabilistic models in the context of sampling-based planning. For this purpose we employ the cross-entropy (CE) method (Rubinstein and Kroese 2004; Kroese et al. 2006) originally developed for estimation of rare-event probabilities and later employed as a general optimization framework. The CE method is a stochastic optimization technique that can be either local or global depending on the chosen model and prior. It is widely applicable and is used to solve complex combinatorial problems such as the minimum graph cut or the traveling salesman problems. The basic idea behind applying the CE approach to motion planning is to recursively iterate the two steps:

1. generate samples from a distribution and compute their costs;
2. update the distribution using a subset of 'high-quality' samples;

until the set of samples becomes concentrated around the optimum, or equivalently until the distribution has approached a delta function. The scheme is general and is expected to converge to an optimum assuming that enough feasible trajectories can be sampled. Yet, the exact number of samples required for approaching a global optimum is difficult to determine. Even though general theoretical convergence of the CE method has been shown (Homem-de Mello 2007; Margolin 2005; Costa et al. 2007; Hu et al. 2007; Hu and Hu 2009) actual rates of convergence, sample complexity, or precise performance guarantees remain open problems.

1.1. Contributions

This work builds upon recent developments in optimal sampling-based planning and stochastic optimization to develop a new method aimed at producing lower cost trajectories through optimally estimated adaptive sampling distribution. From the point of view of motion planning, one can consider the CE method as a way to optimize the distribution for sampling vertices. From the point of view of stochastic optimization, the motion planning component provides feasible samples (trajectories) that are otherwise prohibitively expensive to generate due to complex constraints. Thus, the proposed combined technique utilizes the strengths of both methods to compute trajectories that have lower costs after fewer iterations. The practical contribution of this paper is to spell out the details of two new algorithms combining the RRT* method (Karaman et al. 2011) with the CE method for robotic trajectory optimization (Kobilarov 2011). The difference between the two new methods is whether sampling occurs in the state space or in the space of parametrized trajectories. We provide empirical results of the performance of each approach and demonstrate marked improvement over existing methods. The major limitations of the approach lie in the increased computational time, the lack of a systematic procedure for choosing the best statistical model in view of the system and environment for a given problem, and the fact that the method is only useful in

scenarios in which multiple trajectories to the goal can be computed during the algorithm execution.

1.2. Organization

The motion planning problem is formulated in Section 2. An overview of the probabilistic techniques required for the CE method is given in Section 3.1 followed by a quick background on optimal sampling-based methods, in particular RRT*, in Section 3.2. The methods in Section 3.2 include a slight modification of the original version required by the proposed approach. The CE method applied to trajectory optimization is given in Section 4. The new CE motion planning algorithms are developed in Section 5 and illustrated with simple examples. A more detailed empirical analysis and comparisons using a double integrator in three dimensions and a simple aerial vehicle are given in Section 6.

2. Problem formulation

Consider a robotic vehicle with state trajectory $x : [0, T] \to \mathcal{X}$ controlled using actuator inputs $u : [0, T] \to \mathcal{U}$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ denotes the set of controls, and $T > 0$ is the final time of the trajectory. The state and controls at time $t > 0$ are denoted by $x(t) \in \mathcal{X}$ and $u(t) \in \mathcal{U}$, respectively. The vehicle dynamics satisfies the ordinary differential equation (ODE)

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

which is used to evolve the vehicle state forward in time. In addition, the vehicle is subject to constraints arising from actuator bounds and obstacles in the environment. These constraints are expressed through the vector of inequalities

$$F(x(t)) \geq 0, \tag{2}$$

for all $t \in [0, T]$. The goal is to compute the optimal controls u^* and time T^* driving the system from its initial state $x_0 \in \mathcal{X}$ to a given goal region $\mathcal{X}_g \subset \mathcal{X}$, i.e.

$$\begin{aligned} (u^*, T^*) = \operatorname*{argmin}_{u,T} & \int_0^T C(u(t), x(t))\, \mathrm{dt}, \\ \text{subject to } & \dot{x}(t) = f(x(t), u(t)), \\ & F(x(t)) \geq 0,\ x(0) = x_0,\ x(T) \in \mathcal{X}_g \end{aligned} \tag{3}$$

for all $t \in [0, T]$ and where $C : \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ is a given cost function. A typical cost function includes a time component and a control effort component, i.e. $C(u(t), x(t)) = 1 + \lambda \|u(t)\|^2$ where $\lambda \geq 0$ is a chosen weight.

3. Background

The proposed methodology is based on two main ingredients: the CE method and optimal sampling-based motion planning. Therefore, we first describe the general CE method and then a slightly modified version of RRT* to fit in our framework.

3.1. Cross-entropy method

The CE method can be regarded as an importance sampling solution to the problem of estimating rare events. Let Z denote a random variable defined over a space $\mathcal{Z}$. The rare event of interest in this work is finding a parameter z with a real-valued cost $J(z)$ which happens to be very close to the cost of an optimal parameter z^*. Therefore, as will be explained below, the rare-event estimation is equivalent to the global optimization of $J(z)$. Our development follows closely (Rubenstein and Kroese 2008).

3.1.1. Importance sampling Consider the estimation of the following expression

$$\ell = \mathbb{E}_p[H(Z)] = \int H(z) p(z)\, dz, \tag{4}$$

where $H : \mathcal{Z} \to \mathbb{R}$ is some non-negative performance metric and p is the probability density of Z. Assume that there is another dominating[1] probability density q which is easy to evaluate and sample from, such as a Gaussian. The integral (4) can be expressed as

$$\ell = \int H(z) \frac{p(z)}{q(z)} q(z)\, dz = \mathbb{E}_q \left[H(z) \frac{p(z)}{q(z)} \right]. \tag{5}$$

The density q is called the 'importance density' and can be used to evaluate the integral using independent and identically distributed (i.i.d.) random samples $Z_1, \ldots, Z_N$ from q so that

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} H(Z_i) \frac{p(Z_i)}{q(Z_i)} \tag{6}$$

is an unbiased estimator of ℓ. An important question is then how to select a good density q. The most natural choice is the density that minimizes the variance of the estimator $\hat{\ell}$, i.e.

$$q^* = \arg\min_q \mathbb{V}_q \left(H(Z) \frac{p(Z)}{q(Z)} \right),$$

the solution to which is

$$q^*(z) = \frac{H(z) p(z)}{\ell} \tag{7}$$

since $\mathbb{V}_{q^*}(\ell) = 0$. The density q^* is called the 'optimal importance sampling density'. Of course, this density is only hypothetical and cannot be implemented in practice since it involves the value of ℓ which is what is being estimated in the first place.

A natural way to find a density q that is closest to q^* is in the Kullback–Leibler (KL) sense, i.e. with minimum CE distance between q^* and q. The KL distance between any two given distributions q and p is defined by

$$\mathrm{KL}(p \parallel q) = \int p(z) \ln p(z)\, dz - \int p(z) \ln q(z)\, dz \tag{8}$$

and the required q solves the following optimization

$$\min_q \mathrm{KL}(\, q^* \parallel q)\,. \tag{9}$$

We next consider the case when Z has a probability density function (pdf) $p(\,\cdot;\bar{v})$ belonging to some parametric family $\{p(\,\cdot;v)\,, v \in \mathcal{V}\}$ where $\bar{v}$ is the true or nominal parameter. For instance, this could be a mixture of Gaussians. It is natural to consider an importance density q from the same family. Its optimal parameter v is found through the parametric optimization

$$\min_v \mathrm{KL}(\, q^* \parallel p(\,\cdot, v)\,)$$

This is equivalent to maximizing with respect to v

$$\int H(z)\, p(z,\bar{v}) \ln p(z,v)\; dz,$$

which is obtained using (7) and (8). In other words, the optimal importance density parameter v^* can be found as

$$v^* = \operatorname*{argmax}_v \mathbb{E}_{\bar{v}}[H(Z) \ln p(Z,v)\,]. \tag{10}$$

Finally, the optimal parameter can be approximated numerically by

$$\hat{v}^* = \operatorname*{argmax}_{v\in\mathcal{V}} \frac{1}{N}\sum_{i=1}^{N} H(Z_i) \ln p(Z_i,v)\,, \tag{11}$$

where $Z_1,\ldots,Z_n$ are i.i.d. samples from $p(\,\cdot,\bar{v})$.

3.1.2. Estimation of rare-event probabilities Consider the estimation of the probability that a parameter $z \in \mathcal{Z}$ sampled from $p(\,\cdot;\bar{v})$ has an associated cost $J(z)$ smaller than a given constant γ. It is defined as

$$\ell = \mathbb{P}_{\bar{v}}(J(Z) \le \gamma) = \mathbb{E}_{\bar{v}}[I_{\{J(Z)\le\gamma\}}], \tag{12}$$

where $I_{\{\cdot\}}$ is the indicator function. This can be computed approximately using (6) according to

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} I_{\{J(Z_i)\le\gamma\}} \frac{p(Z_i;\bar{v})}{p(Z_i;v)},$$

where $Z_1,\ldots,Z_N$ are i.i.d. samples from $p(\,\cdot,v)$. In order to determine the optimal v for this computation we can employ (11) to obtain

$$\hat{v}^* = \operatorname*{argmax}_{v\in\mathcal{V}} \frac{1}{N}\sum_{i=1}^{N} I_{\{J(Z_i)\le\gamma\}} \ln p(Z_i,v)\,, \tag{13}$$

where $Z_1,\ldots,Z_N$ are i.i.d. samples from $p(\,\cdot,\bar{v})$. The problem is that when $\{J(Z)\le\gamma\}$ is a rare event, this approximation is meaningless because there will be almost no samples z with $J(z)\le\gamma$ and $\hat{\ell}$ will be incorrectly estimated as zero.

The idea behind the CE method is to employ a multilevel approach using a sequence of parameters $\{v_j\}_{j\ge0}$ and levels $\{\gamma_j\}_{j\ge1}$. At the end the sequence converges to the optimal v^* which then can be used to estimate the integral $\hat{\ell}$ correctly. The procedure starts by drawing N samples $Z_1,\ldots,Z_N$ using an initial parameter v_0, for instance $v_0=\bar{v}$. Let ϱ be a small number, e.g. $10^{-2}\le\varrho\le10^{-1}$. The value γ_1 is set to the ϱth quantile of $H(Z)$, i.e. γ_1 is the largest real number for which

$$\mathbb{P}_{v_0}(H(Z)\le\gamma_1) = \varrho.$$

The level γ_1 can be computed approximately by sorting the costs of the samples $J(Z_1),\ldots,J(Z_N)$ in an increasing order, say $J_1\le\cdots\le J_N$, and setting $\hat{\gamma}_1 = J_{\lceil\varrho N\rceil}$. The optimal parameter v_1 for level $\hat{\gamma}_1$ is then estimated using (11) by replacing γ with $\hat{\gamma}_1$.

Note that the samples with costs $J_1,\ldots,J_{\lceil\varrho N\rceil}$ will also be the samples used to estimate v_1. They form the 'elite set', i.e. the ϱ-fraction of the N samples with the best costs. The procedure then iterates to compute the next γ_i and v_i and terminates when $\gamma_i\le\gamma$. At this point we set $v=v_i$ as the optimal parameter corresponding to the originally given level γ and the probability of $J(Z)\le\gamma$ is computed using v. In summary, each iteration of the algorithm perform two steps, starting with v_0.

1. *Sampling and updating of γ_j*: Sample $Z_1,\ldots,Z_n$ from $p(\,\cdot,\hat{v}_{i-1})$ and compute the ϱth quantile $\hat{\gamma}_t$.
2. *Adaptive updating of v_j*: Compute $\hat{v}_j$ such that

$$\hat{v}_j = \operatorname*{argmin}_{v\in\mathcal{V}} \frac{1}{|\mathcal{E}_j|}\sum_{Z_k\in\mathcal{E}_j} \ln p(Z_k;v)\,, \tag{14}$$

where $\mathcal{E}_j$ is the 'elite' set of samples, i.e. samples Z_k for which $J(Z_k)\le\hat{\gamma}_j$.

3.1.3. CE optimization The idea behind the CE method is to treat the optimization of $J(z)$ as an estimation problem of rare-event probabilities. Define the cost function optimum γ^* by

$$\gamma^* = \min_{z\in\mathcal{Z}} J(z)\,.$$

Finding the optimal trajectory then amounts to iterating the rare-event simulation steps defined in Section 3.1.2 until the cost γ_j approaches γ^*. Typically, after a finite number of iterations $p(\,\cdot,v_j)$ will approach a delta distribution and all samples Z_i will become almost identical. This signifies that the optimum has been found and z^* is set to the sample with lowest cost. Note that the term 'optimal' should be used with caution because although the method explores the state space globally it might still converge to a local solution if, for instance, no samples were obtained near the true global value.

3.2. Sampling-based motion planning

The methods developed in this paper are based on the RRT*/RRG* algorithms. The development will be restricted

to RRT but can be analogously applied in the RRG setting as well. The main point is to perform sampling which exploits all information collected about the costs of trajectories during the algorithm execution. Following Karaman et al. (2011) we construct a RRT* algorithm and augment it with a simple extension: a connection is attempted between all newly added nodes and the goal in order to generate as many trajectories reaching the goal region $\mathcal{X}_g$ as possible. The costs of these trajectories will comprise the data used for adaptive sampling.

Algorithm 1: $\mathcal{T} \leftarrow \texttt{RRT*}(\eta_0, \mathcal{X}_g)$

1 $\mathcal{T} \leftarrow \texttt{InitializeTree}()$
2 $\mathcal{T} \leftarrow \texttt{InsertNode}(\emptyset, \eta_0, \mathcal{T})$
3 $\mathcal{N}_g \leftarrow \emptyset$
4 **for** $i = 1 : N$ **do**
5 $\quad \eta_{\text{rand}} \leftarrow \texttt{Sample}(i, \mathcal{N}_g)$
6 $\quad \eta_{\text{nearest}} \leftarrow \texttt{Nearest}(\mathcal{T}, \eta_{\text{rand}})$
7 $\quad (x_{\text{new}}, u_{\text{new}}, T_{\text{new}}) \leftarrow \texttt{Steer}(\eta_{\text{nearest}}, \eta_{\text{rand}})$
8 $\quad$ **if** $\texttt{ObstacleFree}(x_{new})$ **then**
9 $\quad\quad \mathcal{N}_{\text{near}} \leftarrow \texttt{Near}(\mathcal{T}, \eta_{\text{new}}, |V|)$
10 $\quad\quad \eta_{\text{min}} = \texttt{ChooseParent}(\mathcal{N}_{\text{near}}, \eta_{\text{nearest}}, x_{\text{new}})$
11 $\quad\quad \mathcal{T} \leftarrow \texttt{InsertNode}(\eta_{\text{min}}, \eta_{\text{new}}, \mathcal{T})$
12 $\quad\quad \mathcal{T} \leftarrow \texttt{Rewire}(\mathcal{T}, \mathcal{N}_{\text{near}}, \eta_{\text{min}}, \eta_{\text{new}})$
13 $\quad\quad (x_{\text{g}}, u_{\text{g}}, T_{\text{g}}) \leftarrow \texttt{Steer}(\eta_{\text{new}}, \mathcal{X}_{\text{g}})$
14 $\quad\quad$ **if** $\texttt{ObstacleFree}(x_g)$ *and* $x_g(T_g) \in \mathcal{X}_g$ **then**
15 $\quad\quad\quad \mathcal{T} \leftarrow \texttt{InsertNode}(\eta_{\text{new}}, \eta_{\text{g}}, \mathcal{T})$
16 $\quad\quad\quad \texttt{UpdateCostToGo}(\eta_{\text{new}}, \texttt{Cost}(x_{\text{g}}))$
17 $\quad\quad\quad \texttt{UpdateCostToCome}(\eta_{\text{g}}, \texttt{CostToCome}$
18 $\quad\quad\quad (\eta_{\text{new}}) + \texttt{Cost}(x_g))$
19 $\quad\quad\quad \mathcal{N}_g \leftarrow \mathcal{N}_g \cup \{\eta_g\}$
20 **return** $\mathcal{T}$

The RRT* algorithm. The RRT* algorithm maintains a tree $\mathcal{T}$ of nodes. Each node $\eta \in \mathcal{T}$ contains a state $x \in \mathcal{X}$ and pointers to its parent node and the set of children nodes through the functions $\texttt{Parent}(\eta)$ and $\texttt{Children}(\eta)$. The functions $\texttt{CostToCome}(\eta)$ and $\texttt{CostToGo}(\eta)$ maintain the cost from the start to the node and from the node to the goal set $\mathcal{X}_g$, and are set to ∞ initially. A new node η_{new} is inserted into the tree to become a child of an existing node η_{current} using a function $\texttt{InsertNode}(\eta_{\text{current}}, \eta_{\text{new}}, \mathcal{T})$ which creates an edge between the two nodes and also updates $\texttt{CostToCome}(\eta_{\text{new}})$. The function $\texttt{Steer}(\eta_1, \eta_2)$ generates an optimal trajectory $x : [0, T] \rightarrow \mathcal{X}$ that attempts to drive the system between the two given states x_1 and x_2 for time T. Complete details of the algorithm can be found in Karaman et al. (2011) and Karaman and Frazzoli (2011).

A simple extension. For the purposes of this work the key points are to generate trajectories that reach the goal and

Algorithm 2: $\eta_{\text{min}} \leftarrow \texttt{ChooseParent}(\mathcal{N}_{\text{near}}, \eta_{\text{nearest}}, x_{\text{new}})$

1 $\eta_{\text{min}} \leftarrow \eta_{\text{nearest}}$
2 $c_{\text{min}} \leftarrow \texttt{CostToCome}(\eta_{\text{nearest}}) + \texttt{Cost}(x_{\text{new}})$
3 **for** $\eta_{near} \in \mathcal{N}_{near}$ **do**
4 $\quad (x', u', T') \leftarrow \texttt{Steer}(\eta_{\text{near}}, \eta_{\text{new}})$
5 $\quad$ **if** $\texttt{ObstacleFree}(x')$ *and* $x'(T') = \eta_{new}$ **then**
6 $\quad\quad c' = \texttt{CostToCome}(\eta_{\text{near}}) + \texttt{Cost}(x')$
7 $\quad\quad$ **if** $c' < c_{min}$ **then**
8 $\quad\quad\quad \eta_{\text{min}} \leftarrow \eta_{\text{near}}$
9 $\quad\quad\quad c_{\text{min}} \leftarrow c'$
10 **return** η_{min}

Algorithm 3: $\mathcal{T} \leftarrow \texttt{Rewire}(\mathcal{T}, \mathcal{N}_{\text{near}}, \eta_{\text{min}}, x_{\text{new}})$

1 **for** $\eta_{near} \in \mathcal{N}_{near} \backslash \{\eta_{min}\}$ **do**
2 $\quad (x', u', T') \leftarrow \texttt{Steer}(\eta_{\text{new}}, \eta_{\text{near}})$
3 $\quad$ **if** $\texttt{ObstacleFree}(x')$ *and* $x'(T') = \eta_{near}$ *and*
4 $\quad\quad \texttt{CostToCome}(\eta_{new}) + \texttt{Cost}(x') <$
$\quad\quad \texttt{CostToCome}(\eta_{near})$ **then**
5 $\quad\quad \mathcal{T} \leftarrow \texttt{Reconnect}(\eta_{\text{new}}, \eta_{\text{near}}, \mathcal{T})$
6 $\quad\quad \texttt{UpdateCostToCome}(\eta_{\text{near}}, \texttt{CostToCome}$
7 $\quad\quad (\eta_{\text{new}}) + \texttt{Cost}(x'))$
8 **return** $\mathcal{T}$

to update the cost-to-come and cost-to-go parameters while the tree grows and rewires itself. This is accomplished by attempting to connect every newly added node not only to the existing tree but also to the goal region. Lines 13–18 in Algorithm 1 were added to accomplish this. The leaf nodes of all paths connecting to the goal are stored in a list $\mathcal{N}_g$ which will be used for adaptive sampling (see Section 5) and are passed as an argument to the function `Sample` on line 5. Two new procedures `UpdateCostToGo` (Algorithm 4) and `UpdateCostToCome` (Algorithm 5) are included which are called during rewiring and every time the goal is reached. The minimum cost of trajectories that reach the goal and pass through a given vertex η can then be computed by $\texttt{CostToCome}(\eta) + \texttt{CostToGo}(\eta)$. This value will be use used in the CE method for adaptive sampling in Section 5.

Algorithm 4: $\texttt{UpdateCostToGo}(\eta, c)$

1 **while** $\texttt{CostToGo}(\eta) > c$ **do**
2 $\quad \texttt{CostToGo}(\eta) \leftarrow c$
3 $\quad (\eta, x) \leftarrow \texttt{Parent}(\eta)$
4 $\quad$ **if** $\eta = \emptyset$ **then**
$\quad\quad$ **return**
5 $\quad c = c + \texttt{Cost}(x)$

Algorithm 5: `UpdateCostToCome`(η, c)

```
if CostToCome(η) > c then
    CostToCome(η) ← c
    N = Children(η)
    foreach η_child ∈ N do
        UpdateCostToCome(η_child, c + Cost((η, η_child)))
```

4. CE trajectory optimization

We next present the CE method for robotic trajectory optimization. In addition to describing the core approach (Kobilarov 2011) we focus on the key points of the algorithm required for successful implementation. In addition, we mention its shortcomings that motivate the development of the CE RRT methods.

The CE method is suitable for non-linear and high-dimensional systems operating in relatively uncluttered obstacle environments such as car-like or helicopter robots navigating among buildings or canyons. The method is based on sampling in parametrized trajectory space; while this has resulted in robust and efficient optimization in various settings, it also imposes important limitations. In particular, more complicated obstacles such as those associated with narrow passages significantly shrink the feasible regions in trajectory space and thus, in the absence of a good prior, can render the method intractable due to the large number of rejected samples. This is also one of the motivations for developing the main methods in this work (Section 5) that combining CE with optimal motion planning.

4.1. Trajectory parametrization

A trajectory recording the controls and states over the time interval $[0, T]$ is denoted by the function $\pi : [0, T] \to \mathcal{U} \times \mathcal{X}$, i.e. $\pi(t) = (u(t), x(t))$ for all $t \in [0, T]$. A given control curve $u : [0, T] \to \mathcal{U}$ determines a unique state trajectory $x : [0, T] \to \mathcal{X}$ by evolving the dynamics from an initial state $x_0 \in \mathcal{X}$ (under standard regularity conditions on the ODE (1)). Let $\tau(\pi)$ denote the duration of a given trajectory π. The space of all trajectories originating at point x_0 and satisfying the dynamics is denoted by

$$\mathcal{P} = \{\pi : t \in [0, T] \to (u(t), x(t)) \mid \dot{x}(t) = f(x(t), u(t)), \; x(0) = x_0, \; T > 0\}.$$

Consider a finite-dimensional parametrization of trajectories in terms of vectors $z \in \mathcal{Z}$ where $\mathcal{Z} \subset \mathbb{R}^{n_z}$ is the parameter space. Assume that the parametrization is given by the function $\varphi : \mathcal{Z} \to \mathcal{P}$ according to

$$\pi = \varphi(z).$$

The (control, state) tuples along a trajectory parametrized by z are written as $\pi(t) = \varphi(z, t)$. In addition, it is useful to define the functions $\varphi_x(z, t)$ and $\varphi_u(z, t)$ which extract only the state $x(t)$ or the control $u(t)$, respectively, for given parameter z and time t. The time duration of a trajectory parametrized by z is given by $\tau(z)$. In addition, there is a function $\psi : \mathcal{P} \to \mathcal{Z}$ which extracts a set of parameters from a given trajectory, i.e. $z = \psi(\pi)$. Note that φ and ψ are not necessarily the inverse of each other since there can be many different parametrization choices yielding the same trajectory.

The constrained parameter space $\mathcal{Z}_{\text{con}} \subset \mathcal{Z}$ is the set of parameters satisfying the boundary conditions and constraints and is defined by

$$\mathcal{Z}_{\text{con}} = \{z \in \mathcal{Z} \mid F(\varphi_x(z, t)) \geq 0, \varphi_x(z, T) \in \mathcal{X}_g.\}, \quad (15)$$

for some $T > 0$ and all $t \in [0, T]$. Define the 'cost function' $J : \mathcal{Z} \to \mathbb{R}$ according to

$$J(z) = \int_0^T C(\varphi(z, t)) \, dt. \quad (16)$$

Problem (3) can now be solved approximately by finding T^* and $(x^*, u^*) = \varphi(z^*)$ such that

$$z^* = \arg\min_{z \in \mathcal{Z}_{\text{con}}} J(z). \quad (17)$$

This optimization will be solved through the CE optimization described in Section 3.1.

The correct choice of parametrization φ in general is non-trivial and depends on the problem. The basic requirement is to retain reachability and controllability properties of the original system. There are two general types of parametrizations: through primitives and through a finite number of states along the trajectory connected using a steering method.

Primitives. One general approach is to use $z = (u_1, \tau_1, \ldots, u_m, \tau_m)$ where each u_i, for $1 \leq i \leq m$, is a constant control input applied for duration τ_i. This is an example of a simple 'primitive', e.g. a motion with constant control u_i. Conditions for resolution completeness of planning with such primitives have been established (Yershov and LaValle 2011). Yet, many systems, for instance with inherently unstable dynamics, cannot be directly captured in this representation and require a more complicated type of primitives termed 'maneuvers' to achieve transitions between simpler primitives (Frazzoli et al. 2005). In addition, certain systems have special structural properties of the dynamics such as stable/unstable manifolds or limit cycles that could also be abstracted through special parameters in addition to control inputs.

States. Exact and near-optimal steering methods exist for many robotic locomotion systems, i.e. it is possible to compute a unique curve that satisfies given boundary conditions. This can be accomplished with the help of primitives, or by exploiting properties such as differential flatness

(see e.g. Murray et al. 1995; Murray and Sastry 1993). For such systems a trajectory can be parametrized directly as a sequence of states, e.g. $z = (x_1, \ldots, x_m)$ since the curves between x_0 and x_1, x_1 and x_2, etc. are computed using steering and thus the whole trajectory originating from x_0, passing through $x_1, \ldots, x_m$, and reaching $\mathcal{X}_g$ is uniquely determined by z.

Mixed parametrization. For certain systems steering can be solved through inverse kinematics of a properly chosen sequence of primitives. Thus, it is possible to employ either primitives or states and internally reconstruct one from the other. The helicopter example is one such system.

Examples of each of these three different representation are given in Section 4.5. In general, there is a complicated trade-off between the dimensionality of $\mathcal{Z}$, its resolution completeness, optimality gap due to quantization, the ability to handle complicated environments, and the resulting efficiency of the optimization. Determining provably good parametrization requires an in-depth study beyond the empirical evidence that we provide.

4.2. Choosing a probabilistic model

The CE method falls in the category of global optimization method based on probabilistic models (Zhigljavsky and Zilinskas 2008). The key point is that at every iteration it transforms the model to shift probability mass in low-cost regions. The underlying importance density we choose is a Gaussian mixture model (GMM) since it is a compact way to encode multiple trajectories across multiple homotopy classes. A GMM is chosen for computational convenience since the CE density estimation can be performed using well-established expectation–minimization (EM). On the other hand, a GMM is limited since it can only capture as many local regions in the space as the number of the components in its mixture. Yet, a favorable property is that each Gaussian component can be regarded as an approximation of a local second-order model of the objective function centered at each mean (i.e. by regarding the covariance as the inverse Hessian of the cost). This could explain its fast convergence in the vicinity of a local optimum observed in our examples (in the absence of obstacles).

In addition, the choice of Gaussian distributions is reinforced by the close links between the CE method and two other families of recent stochastic optimization methods, in particular[2] covariance matrix adaptation (CMA) evolution strategies (Igel et al. 2006) and estimation of distribution algorithms (EDA) (Larrañaga and Lozano 2002; Pelikan et al. 2002), as well as related ideas in earlier machine learning literature (e.g. De Bonet et al. 1996). While CE, CMA, and EDA can have different flavors depending on the chosen models and parameter values, their practical implementation is nearly identical in the single Gaussian case.

Algorithm 6: CE motion planning.

Initialization:

0.1 Compute the optimal trajectory reaching the goal by ignoring the constraints, i.e. $z^* = \min_{z \in \mathcal{Z}} J(z)$ such that $\varphi_x(z^*, \tau(z^*)) \in \mathcal{X}_g$

0.2 Set matrix Σ so that the region $\{\varphi_x(z,t) \mid (z - z^*)^T \Sigma (z - z^*) < 2,\ 0 \le t \le \tau(z)\} \subset \mathcal{X}$ covers the reachable state space of interest

0.3 Choose initial samples $Z_1, \ldots, Z_N$ from $\texttt{Normal}(z^*, \Sigma)$; set $j = 0$ and $\hat{\gamma}_0 = \infty$

Iteration:

1. Update $\hat{v}_j$ using (14) (e.g. by EM) over the elite set $\mathcal{E}_j = \{Z_i \mid J(Z_i) \le \hat{\gamma}_j\}$
2. Generate samples $Z_1, \ldots, Z_N$ from $p(\cdot, v_j)\,|_{\mathcal{Z}_{\text{con}}}$ and compute the ϱth quantile $\hat{\gamma}_{j+1} = J_{\lceil \varrho N \rceil}$
3. If $j > 0$ and $\text{KL}(p(\cdot, v_{j-1}) \,||\, p(\cdot, v_j)) < \epsilon$ then finish, otherwise set $j = j + 1$ and goto step (1)

4.3. A parametric density algorithm

A general trajectory optimization algorithm based on the CE method is next constructed. The parameter space is $\mathcal{V} = (\mathbb{R}^{n_z} \times \mathbb{R}^{(n_z{}^2 + n_z)/2})^K \times \mathbb{R}^K$ with elements $v = (\mu_1, \Sigma_1, \ldots, \mu_K, \Sigma_K, w_1, \ldots, w_K)$ corresponding to K mixture components with means μ_k, covariance matrices Σ_k (excluding identical elements due to the matrix symmetry), and weights w_k. The density is defined as

$$p(z; v) = \sum_{k=1}^{K} \frac{w_k}{\sqrt{(2\pi)^{n_z} |\Sigma_k|}} e^{-\frac{1}{2}(z-\mu_k)^T \Sigma_k^{-1} (z-\mu_k)}, \qquad (18)$$

where $\sum_{k=1}^{K} w_k = 1$. The number of mixture components K can be fixed or chosen adaptively (see e.g. Figueiredo and Jain 2002). In the absence of complex constraints even the simplest case with $K = 1$ is capable of solving complex multi-extremal problems.

The complete algorithm is summarized in Algorithm 6.

There are several important points determining the success of the approach.

Initialization. The initial set of samples should be generated to achieve a good coverage of $\mathcal{X}$. We assume that no prior knowledge about the problem is available. Therefore, initial sampling is achieved by ignoring the obstacles and computing an optimal trajectory z^* (step 0.1). A normal distribution with covariance Σ chosen to cover the reachable space of interest (step 0.2) is centered at z^* to obtain the initial sample set (step 0.3). When condition (0.2) cannot be easily solved the covariance can be determined by trial and error until the environment is covered.

Parameter update. The optimal parameter update (step (1)) is accomplished using an EM algorithm (McLachlan and Peel 2002) for $K \geq 2$. In the case $K = 1$ the components of $\hat{v}_j$ are updated simply as

$$\mu = \frac{1}{|\mathcal{E}_j|} \sum_{Z_k \in \mathcal{E}_j} Z_k, \quad \Sigma = \frac{1}{|\mathcal{E}_j|} \sum_{Z_k \in \mathcal{E}_j} (Z_k - \mu)(Z_k - \mu)^T .$$

As the algorithm iterates, the uncertainty volume (i.e. the determinants of the covariances Σ_k) is shrinking towards a delta distribution. This might happen prematurely before reaching a good quality solution if, for instance, too few samples were used. To prevent such degeneracy it is useful to inject a small amount of noise with variances $\nu \in \mathbb{R}^{n_z}$ (see Botev and Kroese 2004), i.e. by setting $\Sigma_k = \Sigma_k + \text{diag}(\nu)$ for each $k = 1, \ldots, K$.

Sampling. The sampling step (2) in Algorithm (6) is accomplished using a standard accept–reject argument and is given below only for convenience.

Generating Samples over $\mathcal{Z}_{con}$

1. Compute $A_k = \sqrt{\Sigma}_k$ for all $k = 1, \ldots, K$ and set $i = 1$
2. Choose $k \in \{1, \ldots, K\}$ proportional to w_k
3. Sample $r \sim \mathcal{N}_{n_z}(0, 1)$ and set $Z_i = \mu_k + A_k r$
4. if $Z_i \in \mathcal{Z}_{con}$ then go to line (3)
5. if $i = N$ then finish; otherwise set $i = i + 1$ and goto line (2)

Note that the only expensive operation is the square root (using e.g. Cholesky decomposition) on line (1) but it is performed only K times before drawing all N samples. It is also possible to handle constrained sampling more efficiently through a local search to detect the boundary of $\mathcal{Z}_{\text{con}}$ and escape from $\mathcal{Z} \backslash \mathcal{Z}_{\text{con}}$.

Termination. The algorithm ends after the change in distribution (measured by KL-divergence) between iterations has become less than a given small constant ϵ. Alternatively, it can be terminated if the GMM covariances have nearly shrunk to zero (measured by their determinant). Very irregular cost functions will prevent such convergence and require separately monitoring decrease in the current optimum in order to terminate.

4.4. GMMs and stochastic optimization

There is an interesting link between employing GMMs in the CE setting and other stochastic trajectory optimization based on fixed local 'exploratory' distribution. In case when very few elite samples are available a classical GMM EM algorithm will fail to estimate correct means and covariances. The EM implementation employed in this work was therefore augmented to avoid degeneracies by adding small artificial noise to the estimated covariances after each iteration. As a result even with few samples the algorithm succeeds and could assign as low as a single sample per Gaussian. In such cases the added noise serves the role of the exploratory distribution used to perform local stochastic variations for gradient descent at the next iteration. Another requirement for successful EM is to initialize the components with slightly overlapping covariance ellipsoids so that their union covers the state space of interest.

4.5. Examples

We illustrate the CE method with three examples developed by Kobilarov (2011). The simple car (Figure 1) is an example of parametrization using primitives with constant forward and turning velocities. The resulting trajectories are not constrained to reach the goal; this condition is instead enforced by a penalty term in the cost function. In contrast, the double integrator example (Figure 2) is based on parametrization using states along the path the curves between which can be computed in closed form. Finally, the helicopter example (Figure 3) illustrates a more complex system evolving in 3D with dynamics abstracted using a sequence of trim primitives and maneuvers. The parametrization is computed through inverse kinematics so that all trajectories reach the goal; the cost function is then simply the time of flight.

The car, point mass, and helicopter scenarios take approximately 6, 10, and 30 seconds, respectively, to produce the solutions shown using a PC with Core i7-920XM, 2.0 GHz. The computation is based on rejecting all sampled trajectories that collide with obstacles. For instance, in the helicopter example approximately 85% of all sampled trajectories are rejected. Thus, the computational times can be significantly improved through: (1) parallelized sampling and parallelized collision checking, (2) constructing a probabilistic parametrized trajectory space obstacle model and using it to tilt the CE distribution away from obstacles. Such directions are not explored in this work. Instead, complex constraints such as obstacles will be handled by using a sampling-based motion planning tree as a basis for stochastic optimization, as described next.

5. CE motion planning

We next develop the CE motion planning methodology that combines optimal sampling-based motion planning and global stochastic optimization. These two concepts are linked through a probabilistic model that is being iteratively optimized while at the same time serving as a motion planning sampling distribution. Our particular approach utilizes the RRT* algorithm for planning and the CE method for adaptive sampling. The basic steps are summarized as follows:

Algorithm overview: trajectory-cross-entropy (TCE) motion planning

0. Expand RRT/PRM and attempt to connect to goal region

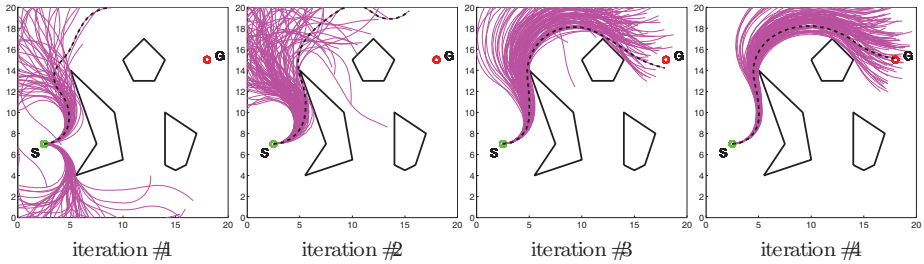


Fig. 1. The first four iterations of the CE algorithm 6 applied to a simple car model. The upper plots show the sampled trajectories $\varphi(Z_1), \ldots, \varphi(Z_N)$ and the current optimal path $\varphi(z^*)$ (dashed). A total of $m = 6$ primitives were used. Iteration #1 shows the resulting set of trajectories after the initialization steps 0.1–0.3 of the CE algorithm.

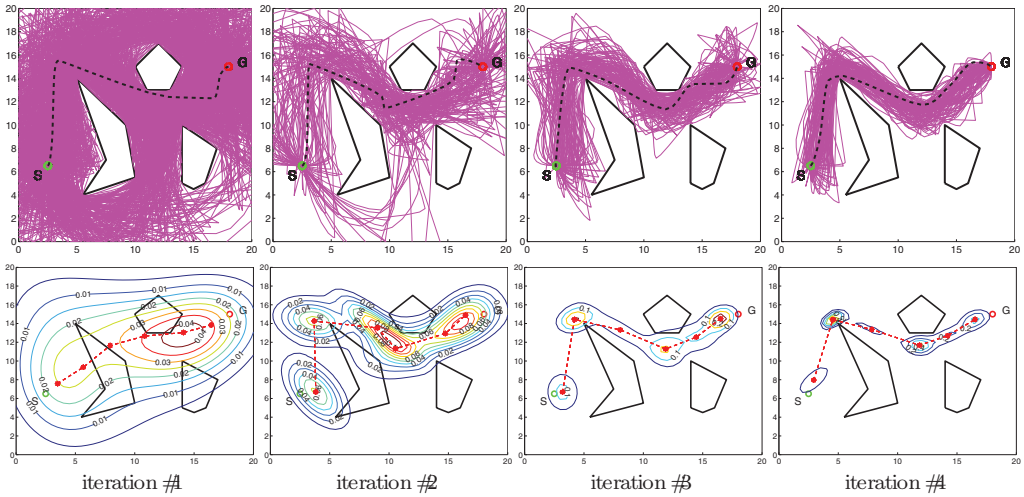


Fig. 2. The first four iterations of the CE algorithm to a double integrator. The upper plots show the sampled trajectories $\varphi(Z_1), \ldots, \varphi(Z_N)$ and the current optimal path $\varphi(z^*)$ (dashed). The lower plots visualize $p(\cdot, \hat{\nu}_j)$ as the level sets of an induced density over the (x, y)-position space which encodes the lowest cost of trajectories passing through it.

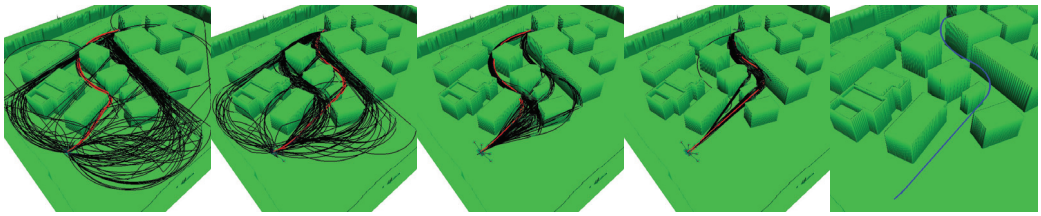


Fig. 3. Several iterations of the CE algorithm applied to the helicopter scenario. As the optimization proceeds the set of samples concentrates in the homotopy classes with minimum trajectory cost.

1. Obtain all RRT/PRM trajectories $\{\pi_i\}_{i=1}^N$ reaching the goal
2. Construct parametrized trajectories $Z_i = \psi(\pi_i)$
3. Update $p_{\mathcal{Z}}$ using the elite subset of these parameters
4. Sample a trajectory $Z \sim p_{\mathcal{Z}}$
5. Select one or more states $X = \varphi(Z, t)$ for a random t and add to RRT/PRM
6. Repeat from either (0) or (1) with some probability. Stop on a termination condition.

The algorithm is based on a CE update of a density $p_{\mathcal{Z}}$ defined over a space of trajectory parameters $z \in \mathcal{Z}$. This density is then used to sample trajectories from which states are extracted and used as RRT/PRM vertices. This can be equivalently regarded as inducing another density on the state space as will be explained below. A simplified version[3] is also developed which bypasses trajectory parametrization and performs density estimation and sampling directly in a state space density $p_{\mathcal{X}}$:

Algorithm overview: state-cross-entropy (SCE) motion planning

0. Expand RRT/PRM and attempt to connect to goal region

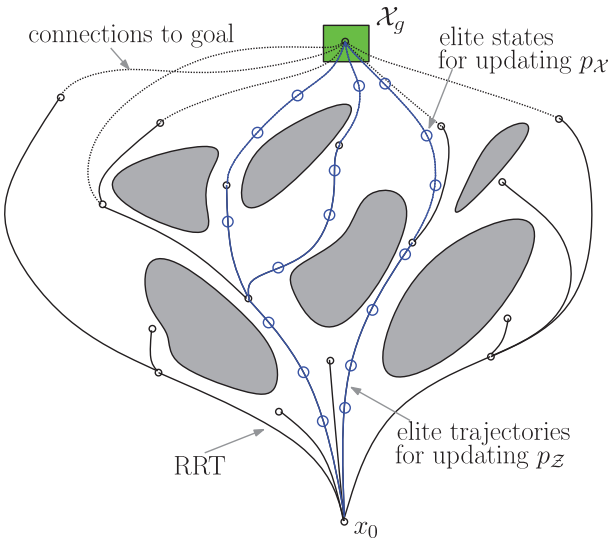


Fig. 4. A motion planning tree with nodes connected to the goal region $\mathcal{X}_g$. Elite trajectories are drawn with thicker lines. Elite states are obtained by discretizing these trajectories, e.g. with a constant time step, and extracting the corresponding states (drawn as circles).

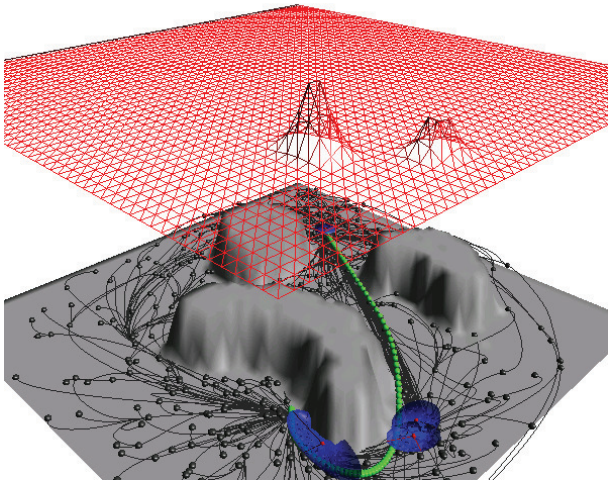


Fig. 5. A tree constructed using the SCE-RRT* algorithm. Gaussian mixture model ellipsoids of the currently updated density are shown. The top graph shows the likelihood $p_{\mathcal{X}}$ (restricted to the planar Euclidean coordinates for better visualization). The peaks identify salient state space regions the are likely to result in low cost trajectories. These regions will be sampled at the next RRT iteration.

1. Obtain all RRT/PRM trajectories $\{\pi_i\}_{i=1}^N$ reaching the goal
2. Discretize each trajectory π_i into a set of states
3. Update $p_{\mathcal{X}}$ using the elite subset of all states of discretized trajectories
4. Sample a state $X \sim p_{\mathcal{X}}$ and add to RRT/PRM
5. Repeat from either (0) or (1) with some probability. Stop on a termination condition.

Figure 4 shows the distinction between using elite states and elite trajectories to update different distributions.

The two methods can be unified by regarding them as optimizations of a special cost function $J(x)$ over the state space $\mathcal{X}$. To make this precise, define the subspace of trajectories $\mathcal{P}_{\text{con}} \subset \mathcal{P}$ which satisfy the constraints and reach the goal, i.e.

$$\mathcal{P}_{\text{con}} = \{(x,u) \in \mathcal{P} \mid \exists T > 0 \text{ s.t. } F(x(t)) > 0 \text{ and } x(T) \in \mathcal{X}_g \text{ for all } t > 0\}.$$

In addition, extend the cost function definition according to $J : \mathcal{P} \cup \mathcal{Z} \cup \mathcal{X} \to \mathbb{R}$ giving the cost of either a trajectory $J(\pi)$, a trajectory parameter $J(z)$, or a single state $J(x)$ depending on the argument. The first two were already defined (see (3) and (16)) by

$$J(\pi) = \int_0^T C(\pi(t))\, dt \quad \text{and} \quad J(z) = \int_0^T C(\varphi(z,t))\, dt,$$

while the new induced cost over $\mathcal{X}$ becomes

$$J(x) = \min_{\pi \in \mathcal{P}_{\text{con}}} \{J(\pi) \mid \exists t \geq 0 \text{ s.t. } \pi(t) = x, \pi(T) \in \mathcal{X}_g, T \geq t\},$$

i.e. the minimum cost over all trajectories that pass through x and reach the goal region. Clearly, we have

$$\min_{x \in \mathcal{X}} J(x) = \min_{\pi \in \mathcal{P}_{\text{con}}} J(\pi) \leq \min_{z \in \mathcal{Z}_{\text{con}}} J(z).$$

From the point of view of stochastic optimization one can regard the problem as either optimizing $J(x)$ using a model $p_{\mathcal{X}}$ (corresponding to the SCE algorithm) or to optimizing $J(\pi)$ using a model $p_{\mathcal{Z}}$ (corresponding to the TCE algorithm). In both cases the data is provided using a motion planning method such as RRT*. The motion planning method uses a sampling density which in SCE case is precisely the same $p_{\mathcal{X}}$, while in the TCE case is a distribution on $\mathcal{X}$ that is implicitly induced by $p_{\mathcal{Z}}$. We next detail the resulting algorithms based on these ideas.

5.1. The SCE RRT* algorithm

The SCE method is a straightforward extension to the modified RRT* algorithm outlined in Section 3.2. It is implemented as a special `Sample` function (Algorithm 7) which has access to the set of tree nodes reaching the goal. This function is called from the main `RRT*` routine (Algorithm 1).

The function performs CE sampling with probability r_{ce}, otherwise reverts to standard uniform sampling over the space of interest in $\mathcal{X}$, denoted by $X \sim \texttt{Uniform}(\mathcal{X})$. The function $\tau_{\min}(\mathcal{N}_g)$ returns the smallest time among the nodes $\mathcal{N}_g$. This time is used to determine the time step h with which trajectories reaching the goal will be quantized to extract states for updating the sampling distribution. The minimum required number of such states $N_{\min}^{\mathcal{X}}$ should either result in at least $2n$ elite samples (where $n = \dim(\mathcal{X})$) or

Algorithm 7: $X \leftarrow$ `Sample`$(i, \mathcal{N}_g)$

parameters: r_{ce}–CE sampling ratio, m–path discretization, k–GMM components

1 **if** `rand`$() < r_{ce}$ **then**
2 $\quad N^{\mathcal{X}}_{\min} = \max(2n/\rho, 2nk)$
3 $\quad h = \tau_{\min}(\mathcal{N}_g)/m$
4 $\quad X \leftarrow$ `SCE_Sample`$(\mathcal{N}_g, h, N^{\mathcal{X}}_{\min})$
5 $\quad$ **if** $X \neq \emptyset$ **then return** X
repeat
6 $\quad$ sample $X \sim$ `Uniform`$(\mathcal{X})$
until `ObstacleFree`(X)
7 **return** X

Algorithm 8: $X \leftarrow$ `SCE_Sample`$(\mathcal{N}_g, h, N_{\min})$

1 $\mathfrak{X} \leftarrow \emptyset$
2 **foreach** $\eta_g \in \mathcal{N}_g$ **do**
3 $\quad$ backtrack path π connecting η_0 and η_g
4 $\quad$ **for** $t = h;\ t < \tau(\pi);\ t = t + h$ **do**
5 $\quad\quad \mathfrak{X} \leftarrow \{\mathfrak{X}, (\pi_x(t), \texttt{CostToCome}(\eta_g))\}$
6 **if** $|\mathfrak{X}| > N_{min}$ **then**
7 $\quad p_{\mathcal{X}} \leftarrow$ `CE_Estimate`$(\mathfrak{X})$
$\quad$ **repeat**
8 $\quad\quad$ sample $X \sim p_{\mathcal{X}}$
$\quad$ **until** `ObstacleFree`(X)
$\quad$ **return** X
else
$\quad$ **return** $\emptyset$

Algorithm 9: $p^* \leftarrow$ `CE_Estimate` $\left(\{Z_i, \gamma_i\}_{i=1}^N\right)$

parameters: ρ—fraction of elite samples

1 $\gamma \leftarrow$ `Quantile`$\left(\{\gamma_i\}_{i=1}^N\right), \rho)$
2 $\hat{v}^* \leftarrow \underset{v \in \mathcal{V}}{\operatorname{argmax}} \dfrac{1}{N} \displaystyle\sum_{i=1}^{N} I_{\{\gamma_i < \gamma\}} \ln p(Z_i, v)$
3 **return** $p(\cdot; \hat{v}^*)$

be proportional to the required number of GMM components. Note that these are guidelines that have resulted in good performance in practice but other choices for $N^{\mathcal{X}}_{\min}$ are possible as well.

Once the list of states and the costs of optimal trajectories passing through them have been assembled in the set $\mathfrak{X}$, it is used to update the distribution using a generic `CE_Estimate` routine (Algorithm 9). Note that this routine will be used for updating both distributions over $\mathcal{X}$ and $\mathcal{Z}$ but for generality is defined on the space $\mathcal{Z}$.

An illustration of the density $p_{\mathcal{X}}$ and its likelihood is given in Figure 5 for a double integrator system with

Algorithm 10: $X \leftarrow$ `Sample`$(i, \mathcal{N}_g)$

parameters: r_{ce}—CE sampling ratio, m—path discretization, k—GMM component

1 **if** `rand`$() < r_{ce}$ **then**
2 $\quad N^{\mathcal{Z}}_{\min} = 2mk$
3 $\quad X \leftarrow$ `TCE_Sample`$(\mathcal{N}_g, N^{\mathcal{Z}}_{\min})$
4 $\quad$ **if** $X = \emptyset$ **then**
5 $\quad\quad N^{\mathcal{X}}_{\min} = max(2n/\rho, 2nk)$
6 $\quad\quad h = \tau_{\min}(\mathcal{N}_g)/m$
7 $\quad\quad X \leftarrow$ `SCE_Sample`$(\mathcal{N}_g, h, N^{\mathcal{X}}_{\min})$
8 $\quad$ **if** $X \neq \emptyset$ **then return** X
repeat
9 $\quad$ sample $X \sim$ `Normal`$(\mathcal{X})$
until `ObstacleFree`(X)
10 **return** X

bounded acceleration (Karaman et al. 2011) moving optimally in a natural terrain. Further details and empirical analysis are given in Section 6.

5.2. The TCE RRT* algorithm

The motivation behind the TCE method is to fully exploit not only individual states in the state space but also correlation between states along whole trajectories. Search over parametrized trajectories has proven key in the stochastic optimization techniques (see Sections 1 and 4.2). Yet, as explained in Section 4 the basic CE method does not easily extend to highly constrained settings. The proposed TCE algorithm overcomes these issues by integrating the CE trajectory optimization (Section 4) with RRT* for handling complex environments.

The TCE algorithm is implemented through the `Sample` (Algorithm 10) function called from the main `RRT*` routine (Algorithm 1). If not enough data is available then the algorithm reverts to SCE sampling (line 4). The parametrization we choose is based on a finite set of states along the path connected with optimal trajectories (as explained in Section 4.1), i.e. $\mathcal{Z} = \mathcal{X}^m$ and

$$z = (x_1, \ldots, x_m),$$

where m is a chosen number. The parametrization then becomes

$$\varphi_x(z) = (x_0, x_1)(x_1, x_2) \cdots (x_{m-2}, x_{m-1}) \\ (x_{m-1}, x_m), (x_m, \mathcal{X}_g)$$

where (x_a, x_b) is an optimal obstacle-free trajectory between two states x_a and x_b computed using the `Steer` command defined in Section 3.2. The function extracting parameters from a given trajectory $\pi = (x, u)$ is then

$$\psi(\pi) = (x(h), x(2h), \ldots, x(mh)),$$

Algorithm 11: $X \leftarrow$ `TCE_Sample`$(\mathcal{N}_g, N_{\min})$

$\mathfrak{Z} \leftarrow \emptyset$
foreach $\eta_g \in \mathcal{N}_g$ **do**
  backtrack path π connecting η_0 and η_g
  convert to parameters $Z \leftarrow \psi(\pi)$
  $\mathfrak{Z} \leftarrow \{\mathfrak{Z}, (Z, \texttt{CostToCome}(\eta_g))\}$
if $|\mathfrak{Z}| < N_{min}$ **then**
  return $\emptyset$
else
  $p_{\mathcal{Z}} \leftarrow \texttt{CE\_Estimate}(\mathfrak{Z})$
repeat
  sample $Z \sim p_{\mathcal{Z}}$
  sample $t \sim \texttt{Uniform}([0, \tau(Z)])$
  $X \leftarrow \varphi(Z, t)$
until `ObstacleFree`(X)
return X

where $h = T/(m+1)$ and T is the shortest time among all trajectories ending in $\mathcal{N}_g$.

This representation was chosen since it naturally fits into the RRT framework through the edge creation process. It is also meaningful considering the L_2 metric on trajectories since

$$\int_0^T \|x_a(t) - x_b(t)\|^2 \, dt \approx \sum_{i=1}^{m} \|x_a(ih) - x_b(ih)\|^2 = \|z_a - z_b\|^2, \tag{19}$$

which approaches equality as $m \to \infty$. This is also the type of metric employed by parametric density estimators such as EM. On the other hand, other metrics such as L_∞ (see e.g. Yershov and LaValle 2011) or Hausdorff (Knepper et al. 2010) distances can also be considered in a more general, i.e. non-parametric setting.

With such a choice of parametrization the algorithm extracts trajectories reaching the goal, converts them into parameter vectors, and updates the distribution using their elite subset. A new trajectory is then sampled and a state along that trajectory is selected to join the tree as long as the whole trajectory is free of obstacles.

The resulting density can be visualized in Figure 6. The density $p_{\mathcal{Z}}$ is difficult to display so an approximation of the 'induced density' $p_{\mathcal{X}}$ is used, formally defined by

$$p_{\mathcal{X}}(X) = \eta \cdot \max_{Z \in \mathcal{Z}_{\text{con}}} \{p_{\mathcal{Z}}(Z) \mid X = \varphi_x(Z, t) \text{ for some } 0 < t < \tau(Z)\}, \tag{20}$$

where $\eta > 0$ is a normalizing constant. The approximate density shown in Figure 6 was built by partitioning the subspace of planar Euclidean coordinates into a grid, drawing 10,000 trajectory samples from $p_{\mathcal{Z}}$, tracing the corresponding trajectories and retaining the maximum likelihood value in each traced cell. The unvisited cells are set to 0. Further

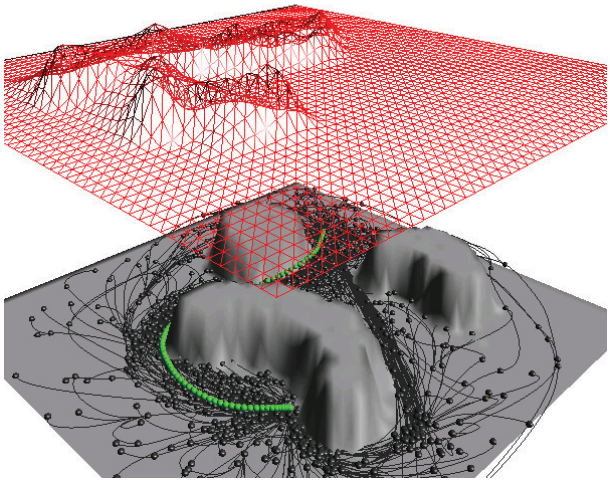


Fig. 6. An RRT* built using TCE sampling. The top graph visualizes the likelihood $p_{\mathcal{Z}}$ transformed into another likelihood $p_{\mathcal{X}}$ that encodes the cost of the optimal trajectory passing through a given state. Although only three Gaussian mixture components in trajectory parameter space $\mathcal{Z}$ were used, the induced likelihood in state space $\mathcal{X}$ has a non-trivial landscape. It corresponds to salient states that are likely to be sampled in the next iteration.

examples and empirical analysis of the algorithm will be given in Section 6.

5.3. User parameters

The algorithm depends on several user-defined parameters that require further investigation (Table 1). The elite fraction ρ typically require minor tuning since the listed default values have proven effective for various classes of problems (Rubenstein and Kroese 2008). We set the CE sampling ratio r_{ce} to 0.5 in order to balance equally between exploring the space and exploiting the collected information about trajectory costs. In the absence of enough information the algorithm automatically reverts to exploration. This parameter can be optimized on-line using more sophisticated techniques. Path discretization parameter m determines how much information is extracted from each trajectory. A very fine discretization results in redundant information that is unnecessary and will not improve the density estimates. On the other hand, very complicated maze-like environments would require long trajectories and more segments. A rule of thumb is to keep track of the average number of tree edges along paths reaching the goal and set m to be at least as high. The optimal number of GMM components is problem dependent and currently no general procedure is available to select k. In theory, each component should 'learn' the most promising state space regions or, respectively, trajectory homotopy classes. In practice, for the examples studied in this work more than four components did not improve the resulting trajectory cost. This could also be explained through the fact that it is non-trivial

Table 1. User-defined parameters affecting the algorithms.

Description	Variable	Range	Default value
Elite fraction	ρ	$[0.01, 0.1]$	0.1
CE sampling ratio	r_{ce}	$[0, 1]$	.5
Path discretization	m	> 0	8
GMM components	k	> 0	4

CE, cross-entropy; GMM, Gaussian mixture model.

to populate all homotopy classes of trajectories reaching the goal even with fast exploration methods such as RRT.

5.4. Limitations

The proposed CE motion planning methods are applicable to problems for which a single or multiple paths to the goal can be computed during the algorithm execution. If only one path was computed then the proposed approach behaves similarly to existing local stochastic optimization techniques. The greater the number of paths that become available, the better the model that can be constructed and exploited for further sampling. The GMM probabilistic model was chosen for computational convenience and due to empirical evidence as well as links to other successful methods. Precise guidelines for selecting a representation that optimally trades off efficiency and optimality are still lacking. In this context, it would be also useful to consider non-parametric representations such as locally weighted learning techniques (Atkeson et al. 1997) due to their incremental nature and computational efficiency, or sparse Gaussian process (GP) classification (Rasmussen and Williams 2005) with available formal bounds (Seeger 2003) that could be useful for establishing performance guarantees. Another issue that could effect performance and requires further study is the relation between the chosen model and the environment complexity. Finally, the proposed methods are useful in higher dimensions where dense sampling is computationally intractable. In problems with up to a few dimensions we expect that standard methods can simply be run longer to obtain comparable results.

6. Examples

We use two simple simulated examples to test the proposed methods. They correspond to commonly used models and were chosen since optimal steering procedures required by the RRT algorithms are readily available.

6.1. Double integrator

We consider a double integrator system moving in a constrained three-dimensional environment. This is a higher-dimensional version of the example considered by Karaman et al. (2011). The system has state space $\mathcal{X} = \mathbb{R}^3 \times \mathbb{R}^3$ with state $x = (q, v)$ consisting of the position q and velocity v.

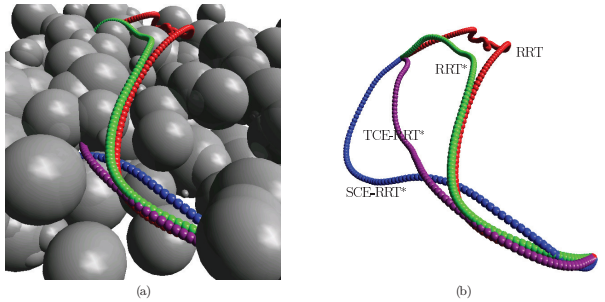


Fig. 7. (a) A box environment with randomly placed spheres and computed optimal trajectories by the four methods using 5,000 samples. (b) The same trajectories with obstacles removed for clearer view. The resulting trajectory costs are: RRT, 21.09; RRT*, 13.73; SCE-RRT*, 11.39; TCE-RRT*, 10.70. While the random sampling seed is identical for all four methods, the optimal solution computed by the three RRT* methods lie in different homotopy classes. The CE computations were performed with $m = 8$ for trajectory quantization and $k = 4$ Gaussian mixture components.

The control space $\mathcal{U} \subset \mathbb{R}^3$ consists of the accelerations u. The dynamics is

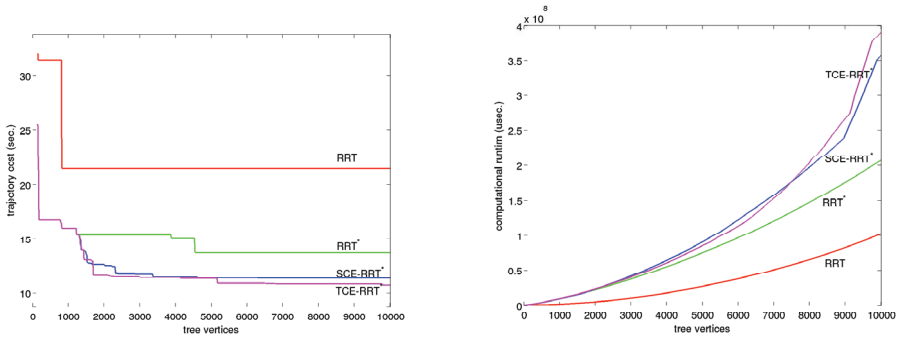
$$\dot{q} = v, \quad \dot{v} = u, \quad \|u\| < u_{max},$$

where $u_{\max} > 0$ is a given scalar bound. We consider the time-optimal planning problem, i.e. the cost defined in (3) is

$$C(x, u) = 1.$$

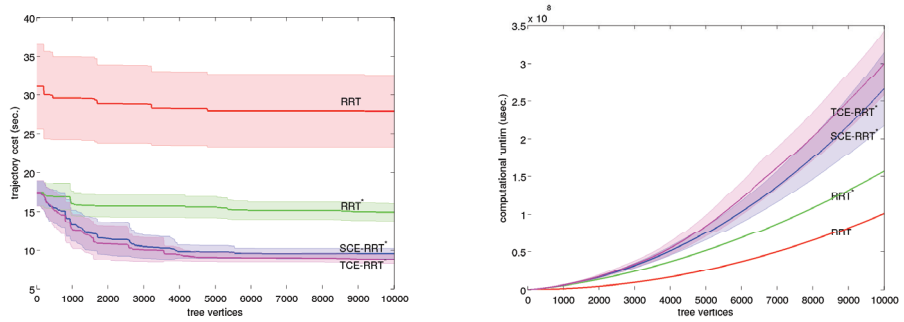
The `Steer`(x_a, x_b) function computes the time-optimal trajectory between the two states. A time-optimal profile of a one-dimensional double integrator can be computed in closed form since it consists of two intervals of constant acceleration. To extend to multiple dimensions the time-optimal acceleration profiles are computed for each dimension, the one with highest resulting time is retained, and the other profiles are then recomputed under that final time constraint. This is accomplished directly through the solution of quadratic equations in closed form with some bookkeeping.

We construct an environment bounded by a box in three dimensions, populated with spheres with random centers inside the box and random radiuses of up to half the smallest dimension of the box. Figure 7 shows an example box environment with dimensions $50 \times 50 \times 10$ meters populated with 300 spheres. The spheres can intersect and create non-trivial concave obstacles and multiple homotopy classes of paths. We study the performance of four sampling-based methods: the standard RRT (LaValle 2006); the RRT* (Karaman et al. 2011); and the two adaptive sampling methods proposed in this work abbreviated by SCE-RRT* (Section 5.1) and TCE-RRT* (Section 5.2). The RRT* algorithms were constructed using nearest neighbor set with size $|\mathcal{N}_{\text{near}}| = \gamma \log(|\mathcal{T}|)$. We set $\gamma = 10$ since in our experiments smaller values resulted in poorer solutions by all three RRT*-based methods.



(a) (b)

Fig. 8. Computation details for the scenario in Figure 7: (a) resulting trajectory costs as a function of vertices; (b) computational time taken as the number of vertices increases.



(a) (b)

Fig. 9. Double integrator trajectory costs (a) and computational times (b) computed by the four algorithms: RRT, RRT*, SCE-RRT*, TCE-RRT*. Results are averaged over 20 Monte Carlo runs.

Even though the double integrator is a simple system, the devised six-dimensional scenario cluttered with obstacles is a challenging planning problem. As our results illustrate sampling-based methods including RRT* can greatly benefit from an informed adaptive sampling. The proposed CE sampling methods compute lower-cost solutions quicker (see Figure 8). The required computational effort is greater than existing methods but is offset by the ability to find more superior solutions using a smaller number of nodes. Note that these are only empirical observations. Establishing convergence rates and sample complexity formally remains an open problem. Figure 9 provides averaged results from multiple randomized runs.

6.2. Simple air vehicle

We employ the simple fixed wing vehicle model (Karaman and Frazzoli 2010) consisting of decoupled models of the Dubins car in the plane and a double integrator in altitude. The state space is $\mathcal{X} = \mathrm{SE}(2) \times \mathbb{R}^2$ with state $x = (\theta, x, y, z, v_z)$. The control space is $\mathcal{U} \subset \mathbb{R}^2$ with controls $u = (\omega, a_z)$ such that $|\omega| < \tan(\phi_{\max})$ and $|a_z| \leq a_{\max}$, where $\phi_{\max}$ is the maximum steering angle and $a_{\max}$ is the maximum altitude acceleration. The vehicle dynamics is defined according to

$$\dot{\theta} = \omega, \quad \dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{z} = v_z, \quad \dot{v}_z = a_z,$$

where $v > 0$ is the constant forward velocity. We are interested in time optimal motions. Following Karaman and Frazzoli (2010), we employ a steering procedure which initially computes decoupled optimal Dubins curves in the plane and optimal double integrator curve in altitude. If the double integrator curve takes less time than Dubins, then it is recomputed using the Dubins time as a final time constraint. Otherwise the Dubins velocities and times are scaled to match the longer time required by the double integrator.

Density estimation is performed in the ambient space of $\mathcal{X}$ (in the SCE case) and $\mathcal{Z}$ (in the TCE case). For instance, on $\mathcal{X} = \mathrm{SE}(2) \times \mathbb{R}^2 \sim S^1 \times \mathbb{R}^2 \times \mathbb{R}^2$, the angle θ is regarded as a vector so that $x := (\cos(\theta), \sin(\theta), x, y, z, v_z)$ and the CE update is performed in $\mathbb{R}^6$ without internally enforcing $(x^1, x^2) \in S^1$. Once the estimation (i.e. using EM) completes the estimated vectors $(\hat{x}^1, \hat{x}^2)$ are projected onto the circle. This corresponds to using the von Mises distribution to approximate a Gaussian on the circle. A more generally applicable approach is to consider direct estimation on SE(2) using Lie group methods (Chirikjian 2012),

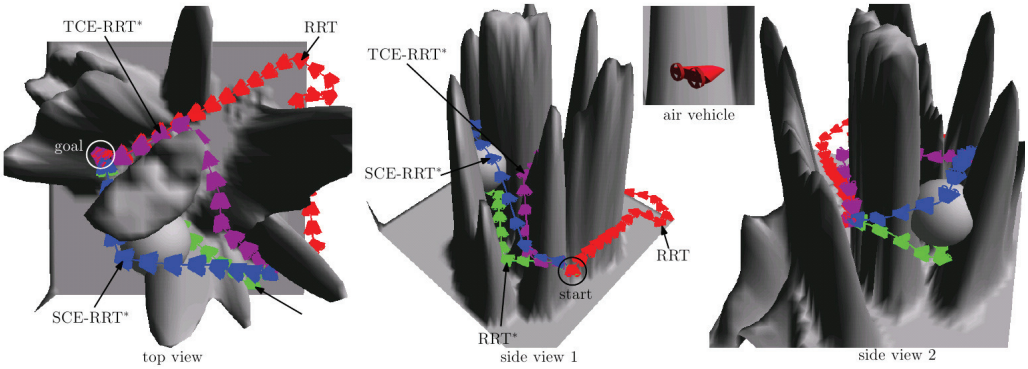


Fig. 10. Air vehicle paths through a digital terrain, computed by the four algorithms: RRT, RRT*, SCE-RRT*, TCE-RRT* using 1,000 vertices. Only a few states are rendered along the paths for better visibility.

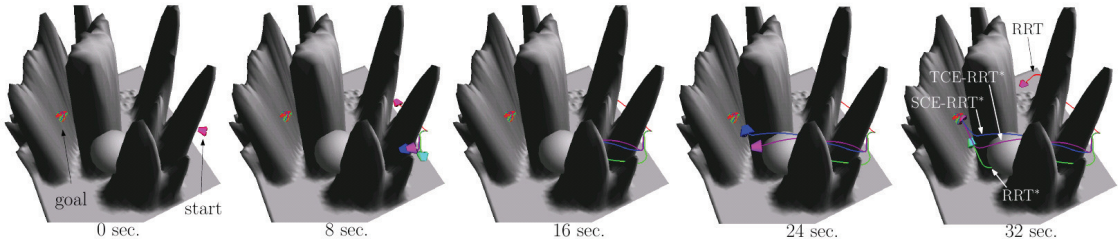


Fig. 11. Snapshots in time of the solution trajectories computed by the four methods. TCE-RRT* and SCE-RRT* compute a 32 second path; the RRT* path lasts 37 seconds; RRT takes 47 seconds to reach the goal.

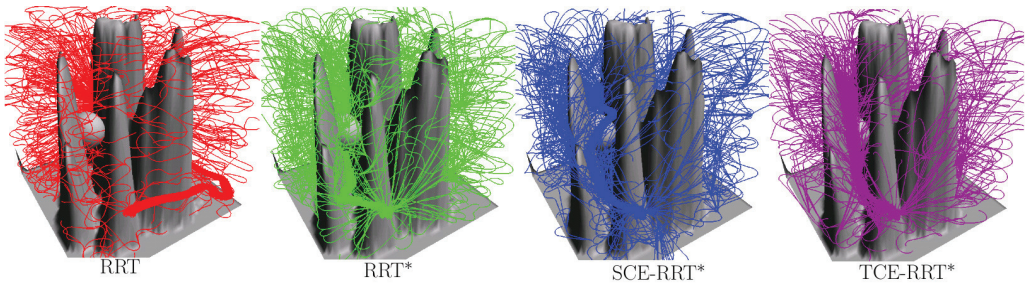


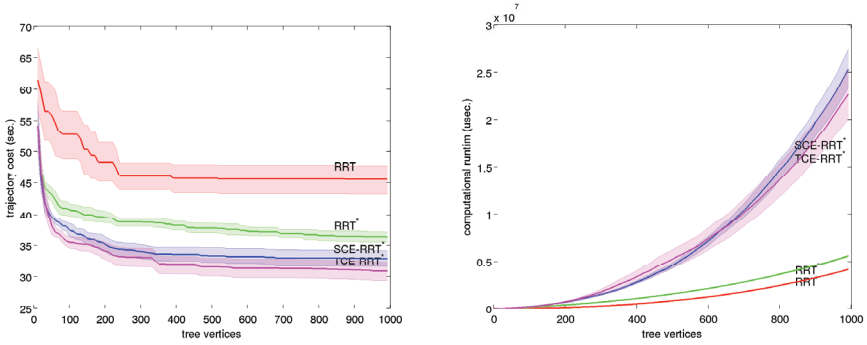
Fig. 12. Roadmaps with 1,000 vertices used to compute the trajectories shown in Figure 11.

which would be especially useful in higher dimensions, e.g. for systems consisting of rigid bodies.

A scenario with uneven terrain depicted in Figure 10 is used to compare the performance of the proposed adaptive sampling methods to RRT and RRT*. In this example we used $k = 4$ Gaussian components and $m = 8$ discrete trajectory states. Figure 11 shows one particular solution to illustrate the type of paths computed by each method. Results from 20 Monte Carlo runs are shown on Figure 13. The CE sampling methods are able to reduce the computed costs even after a few hundred iterations. Yet, as the number of iterations increases their required computational time becomes an order of magnitude higher than RRT and RRT*. This is because the number of trajectories reaching the goal (and, hence, the accumulated information about trajectory costs) is increasing and requires more processing. This is a drawback of our basic CE implementation based on a fixed quantile and iterative GMM processing. The issue can be remedied though for instance by employing incremental, localized, and/or sparsified models as discussed in Section 5.4.

7. Conclusion

In this paper we have proposed a methodology for improving the performance of sampling-based motion planning, i.e. for computing trajectories with lower costs more efficiently. The key point is to use a probabilistic model in either state space and parametrized trajectory space that



(a) (b)

Fig. 13. Air vehicle trajectory cost (a) and computational times (b) using the environment depicted in Figure 10 taken by the four algorithms: RRT, RRT*, SCE-RRT*, TCE-RRT*. Results are averaged over 20 Monte Carlo runs.

is estimated based on the cost of generated paths reaching the desired goal region. The CE method for stochastic optimization is used to adapt the model towards regions of progressively lower cost. The approach is coupled with the optimal rapidly-exploring random tree (RRT*) which uses the model as a sampling distribution and at the same time updates it with newly explored trajectories. Empirical evidence suggests that such adaptive sampling produces lower cost solutions with fewer iterations. Future work will evaluate model representations in view of system and environment complexity and the resulting trade-off between efficiency and optimality. Establishing formal convergence guarantees and sample complexity is another important aspect that must be addressed.

Notes

1. The density q dominates Hp when $q(z) = 0 \Rightarrow H(z)\, p(z) = 0$.
2. The author thanks C. Atkeson for pointing out the closely related CMA method.
3. The author thanks A. Bagnell for a discussion that lead to including this version of the algorithm.

Funding

The author was partially supported by the Keck Institute for Space Studies, Caltech.

Acknowledgements

The author thanks the reviewers for the useful directions for improving the paper.

References

Atkeson CG, Moore AW and Schaal S (1997) Locally weighted learning. *Artificial Intelligence Review* 11: 11–73.

Botev Z and Kroese DP (2004) Global likelihood optimization via the cross-entropy method with an application to mixture models. In *Winter Simulation Conference*, pp. 529–535.

Burns B and Brock O (2005a) Sampling-based motion planning using predictive models. In *IEEE International Conference on Robotics and Automation*, April 2005, pp. 3120–3125.

Burns B and Brock O (2005b) Toward optimal configuration space sampling. In *Robotics: Science and Systems*.

Chirikjian GS (2012) Applied and numerical harmonic analysis. *Stochastic Models, Information Theory, and Lie Groups. Volume II: Analytic Methods and Modern Applications*. Basel: Birkhäuser.

Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE, et al. (2005) *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press.

Clarke FH, Ledyaev YS, Stern RJ and Wolenski PR (1998) *Nonsmooth Analysis and Control Theory*. Berlin: Springer.

Costa A, Jones OD and Kroese D (2007) Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters* 35: 573–580.

De Bonet JS, Isbell CL Jr and Viola PA (1996) Mimic: Finding optima by estimating probability densities. In *Proceedings of NIPS'96*, pp. 424–430.

Denny J and Amato NM (2011) Toggle PRM: Simultaneous mapping of C-free and C-obstacle - a study in 2D. In *Proceedings IEEE IROS2011*, pp. 2632–2639.

Figueiredo MAF and Jain AK (2002) Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24: 381–396.

Frazzoli E, Dahleh MA and Feron E (2005) Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics* 21: 1077–1091.

Homem-de Mello T (2007) A study on the cross-entropy method for rare-event probability estimation. *INFORMS Journal on Computing* 19: 381–394.

Hsu D, Latombe J-C and Kurniawati H (2006) On the probabilistic foundations of probabilistic roadmap planning. *The International Journal of Robotics Research* 25: 627–643.

Hsu D, Sanchez-Ante G and Sun Z (2005) Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA 2005)*, pp. 3874–3880.

Hu J, Fu MC and Marcus SI (2007) A model reference adaptive search method for global optimization. *Operations Research* 55: 549–568.

Hu J and Hu P (2009) On the performance of the cross-entropy method. In *Winter Simulation Conference (WSC '09)*, pp. 459–468.

Igel C, Suttorp T and Hansen N (2006) A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation GECCO*. New York: ACM Press, pp. 453–460.

Kalakrishnan M, Chitta S, Theodorou E, Pastor P and Schaal S (2011) STOMP: stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Kalisiak M and van de Panne M (2007) Faster motion planning using learned local viability models. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA'07)*, pp. 2700–2705.

Karaman S and Frazzoli E (2010) Optimal kinodynamic motion planning using incremental sampling-based methods. In *IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, December 2010.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research (to appear)*, 2011.

Karaman S, Walter MR, Perez A, Frazzoli E and Teller S. Anytime motion planning using the RRT*. In *IEEE Conference on Robotics and Automation (ICRA)*, April 2011.

Knepper RA and Mason MT (2011) Realtime informed path sampling for motion planning search. In *15th International Symposium on Robotics Research (ISRR)*.

Knepper RA, Srinivasa S and Mason MT (2010) An equivalence relation for local path sets. In Latombe J-C, Hsu D, Isler V and Lin MC (eds), *The Ninth International Workshop on the Algorithmic Foundations of Robotics*. Heidelberg: Springer.

Kobilarov M (2011) Cross-entropy randomized motion planning. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, June 2011.

Kroese D, Porotsky S and Rubinstein R (2006) The cross-entropy method for continuous multi-extremal optimization. *Methodology and Computing in Applied Probability* 8: 383–407.

Kurniawati H and Hsu D (2004) Workspace importance sampling for probabilistic roadmap planning. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, vol. 2, pp. 1618–1623.

Ladd AM and Kavraki LE (2005) *Fast Tree-Based Exploration of State Space for Robots with Dynamics*. Berlin: Springer, pp. 297–312.

Larrañaga P and Lozano JA (eds) (2002) *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Dordrecht: Kluwer Academic Publishers.

LaValle SM (2006) *Planning Algorithms*. Cambridge: Cambridge University Press.

Li Y and Bekris KE (2010) Balancing state-space coverage in planning with dynamics. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3246–3253.

Margolin L (2005) On the convergence of the cross-entropy method. *Annals of Operations Research* 134: 201–214.

McLachlan G and Peel D (2002) *Finite Mixture Models*. New York: John Wiley & Sons, Inc.

Moore A and Schneider J (1995) Memory-based stochastic optimization. In *Neural Information Processing Systems 8*.

Murray RM, Rathinam M and Sluis WM (1995) Differential flatness of mechanical control systems. In *Proceedings ASME International Congress and Exposition*.

Murray RM and Sastry S (1993) Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control* 38: 700–716.

Pelikan M, Goldberg DE and Lobo FG (2002) A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications* 21: 5–20.

Powell WB (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (*Wiley Series in Probability and Statistics*). New York: John Wiley & Sons, Inc.

Rasmussen CE and Williams CKI (2005) *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. Cambridge, MA: The MIT Press.

Ratliff N, Zucker M, Bagnell JA and Srinivasa S (2009) Chomp: Gradient optimization techniques for efficient motion planning. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA'09)*, pp. 489–494.

Rickert M, Brock O and Knoll A (2008) Balancing exploration and exploitation in motion planning. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA 2008)*, pp. 2812–2817.

Rubinstein RY and Kroese DP (2004) *The Cross-entropy Method: A Unified Approach to Combinatorial Optimization*. New York: Springer.

Rubenstein RY and Kroese DP (2008) *Simulation and the Monte Carlo Method*. New york: John Wiley & Sons, Inc.

Seeger M (2003) Pac–Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research* 3: 233–269.

Spall JC (2003) *Introduction to Stochastic Search and Optimization*. New York: John Wiley & Sons, Inc..

Theodorou E, Buchli J and Schaal S (2010) A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research* 11: 3137–3181.

Yershov D and LaValle S (2011) Sufficient conditions for the existence of resolution complete planning algorithms. In Hsu D, Isler V, Latombe J-C and Lin M (eds), *Algorithmic Foundations of Robotics IX* (*Springer Tracts in Advanced Robotics*, vol. 68). Berlin: Springer, pp. 303–320.

Zhigljavsky A and Zilinskas A (2008) *Stochastic Global Optimization*. New York: Springer.

Zucker M, Kuffner J and Bagnell JA (2008) Adaptive workspace biasing for sampling-based planners. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA'08)*, pp. 3757–3762.

J Nonlinear Sci
DOI 10.1007/s00332-012-9140-7

Journal of Nonlinear Science

Solving Optimal Control Problems by Exploiting Inherent Dynamical Systems Structures

Kathrin Flaßkamp · Sina Ober-Blöbaum · Marin Kobilarov

Received: 31 October 2011 / Accepted: 23 June 2012

Abstract Computing globally efficient solutions is a major challenge in optimal control of nonlinear dynamical systems. This work proposes a method combining local optimization and motion planning techniques based on exploiting inherent dynamical systems structures, such as symmetries and invariant manifolds. Prior to the optimal control, the dynamical system is analyzed for structural properties that can be used to compute pieces of trajectories that are stored in a motion planning library. In the context of mechanical systems, these motion planning candidates, termed primitives, are given by relative equilibria induced by symmetries and motions on stable or unstable manifolds of e.g. fixed points in the natural dynamics. The existence of controlled relative equilibria is studied through Lagrangian mechanics and symmetry reduction techniques. The proposed framework can be used to solve boundary value problems by performing a search in the space of sequences of motion primitives connected using optimized maneuvers. The optimal sequence can be used as an admissible initial guess for a post-optimization. The approach is illustrated by two numerical examples,

Communicated by Oliver Junge.

K. Flaßkamp · S. Ober-Blöbaum (✉)
Computational Dynamics and Optimal Control, Department of Mathematics, University of Paderborn, Paderborn, Germany
e-mail: sinaob@math.upb.de

K. Flaßkamp
e-mail: kathrinf@math.upb.de

M. Kobilarov
Control and Dynamical Systems, California Institute of Technology, Pasadena, USA
e-mail: marin@cds.caltech.edu

Published online: 13 July 2012

the single and the double spherical pendula, which demonstrates its benefit compared to standard local optimization techniques.

Keywords Lagrangian mechanics · Optimal control · Symmetries · Invariant manifolds

Mathematics Subject Classification 37J15 · 49M37 · 70Q05

1 Introduction

This work combines classical methods from dynamical systems theory and recently developed numerical optimal control methods. The underlying motivation is to overcome one of the major limitations of current numerical control methods, namely the restriction to local optimality for optimal control solutions. The basic idea of the presented approach is to exploit the inherent dynamical properties of the system under consideration. These structures can be revealed by an analysis of the natural dynamics on the one hand, and the system's motion under the influence of specific controls on the other hand.

Optimal Control of Dynamical Systems Optimal control theory goes back to the work of Pontryagin on necessary optimality conditions in the last century, whereas its roots are attributed to Bernoulli because of his work on the brachistochrone problem dating from more than 300 years ago (Sussmann and Willems 1997). To this day, it is an important field of research, based on the question of how to force a system into a desired behavior in an optimal way. A typical problem formulation is as follows: which path of the system's dynamical states, forced by an admissible control trajectory, minimizes a given cost functional? Here, the space of all admissible state and control trajectories is infinite dimensional and additionally constrained by fixed boundary values and possibly further restrictions on the system's states or control input. Therefore, solving optimal control problems most often relies on numerical means. The developed methods can be divided into two classes (cf. Binder et al. 2001). Indirect methods apply the Pontryagin maximum principle to obtain a system of necessary optimality conditions and then solve these boundary value problems. In contrast, direct methods (cf. Betts 1998) begin with a discretization that transforms the optimal control problem to a nonlinear constrained optimization problem. Nonlinear programming methods, such as SQP (sequential quadratic programming, cf. e.g. Gill et al. 2000) can then be applied. However, typically, these are local solvers that cannot guarantee global optimality and require good initial guesses. A number of global optimization methods (e.g. Neumaier 2004; Zhigljavsky and Zilinskas 2008) have been developed to overcome these limitations.

This work develops a global approach that exploits special system structures related to dynamical symmetries. Preservation of such structures will also play a key role in the numerical approximation methods that we will develop. This is related to recent approaches in optimal control for mechanical systems that are structure preserving, i.e. the discretization preserves the system structures, e.g. symmetry or conserved quantities such as momenta, by using discrete variational mechanics (cf. Marsden and West 2001; Ober-Blöbaum et al. 2011). The structure exploiting

motion planning approach that we present provides state and control sequences that can serve as an admissible sophisticated initial guess to a post-optimization, e.g. by a local SQP method.

Applications of optimal control theory are numerous in many different areas of research. Optimal control methods have been successfully applied to electric power systems (Christensen et al. 1987) as well as to many different mechanical systems, e.g. in aerodynamics (Naldi and Marconi 2011) and space mission design (Dellnitz et al. 2009), in bio medicine, robotics (Leyendecker et al. 2009), and automotive engineering (Gerdts 2005).

Motion Planning In the last decades, there has been a growing importance of mechatronic systems as mechanical systems with embedded electronics and digital control units. This has led to multidisciplinary research on mechatronic systems as well, in particular regarding control issues. While classical control theory focuses on stability and feedback or on open loop control, the influence of planning methods from the field of artificial intelligence, i.e. discrete methods such as e.g. decision processes (cf. LaValle 2006) give rise to new kinds of motion planning approach that combine continuous and discrete methods. In this work, the term motion planning is used in the sense of generating open loop trajectories for dynamical systems. This will be accomplished by combining optimal control methods with discrete planning techniques based on search trees. Motion planning by *motion primitives* fits into this category of hybrid motion planning approaches. The idea is to solve the complex control problem by constructing a finite sequence of simple motion termed motion primitives. Frazzoli et al. (2005) explain that this approach can be deduced from the intuitive way in which human pilots steer helicopters, that is, by recurrent simple steering modes with short intermediate control maneuvers.

Following the idea of Frazzoli et al. (2005), we quantize the space of state and control trajectories by representative small pieces of solution curves which can be combined into various sequences. These motion primitives are stored in a *motion planning library*. The problem is thus reduced to searching for the optimal sequence out of all admissible sequences in the library which can be solved using global search methods. Problems with state constraints such as obstacles in the environment can be handled with the help of probabilistic roadmap methods (LaValle 2006; Choset et al. 2005). Candidates for the motion primitives can be obtained by the inherent dynamical properties of the system under consideration, such as motion along relative equilibria (cf. Sect. 3.2) or motions on stable or unstable manifolds of the natural dynamics (cf. Sect. 3.3).

Mechanical Systems and Symmetry The proposed motion planning approach is general and can be applied to arbitrary dynamical systems. However, we focus on optimal control of mechanical systems, because these systems exhibit well-studied structural properties (cf. e.g. Abraham and Marsden 1987; Marsden and Ratiu 1999; Bloch 2003; Bullo and Lewis 2004).

In geometric mechanics, mechanical systems are modeled by a variational approach. Hamilton's least action principle is based on the Lagrangian of the system and can be extended to systems underlying external forcing by the Lagrange–d'Alembert principle. This leads to the well known forced Euler–Lagrange equations as the sys-

tem's equations of motion. Variational mechanics can be directly discretized using discrete variations, e.g. for numerical simulation techniques or optimal control methods (e.g. Marsden and West 2001; Ober-Blöbaum et al. 2011). These so called geometric integrators are of great interest, because they preserve properties of the continuous system, such as symplecticity or conserved momenta induced by symmetry. In addition, they exhibit long-time stable energy behavior. An optimal control method for mechanical systems based on variational integrators is DMOC (*Discrete Mechanics and Optimal Control*, cf. (Ober-Blöbaum et al. 2011)) which will be described in Sect. 2.3.

In this work, we will study continuous symmetries that can be described by a Lie group action. For physical systems, this means the invariance of the Lagrangian with respect to translational or rotational motions. These properties are important in control, because a solution trajectory that has been designed for one specific situation, e.g. a turn maneuver for a helicopter, is suitable in many other cases as well, because it does not explicitly depend on the absolute position in space. More precisely, we will call two pairs of state and control trajectories of a symmetric system *equivalent*, if the states are related by a Lie group element and the pairs by a time shift, i.e. a spatiotemporal symmetry equivalence. Continuous symmetries in mechanical systems correspond to the conservation of momenta and to the existence of motions that are solely induced by symmetry, i.e. relative equilibria (cf. Sect. 3.2). For Hamiltonian and Lagrangian systems, relative equilibria can be determined analytically by symmetry reduction procedures (Marsden and Ratiu 1999; Marsden et al. 2000; Marsden 1993). Whereas relative equilibria and symmetry reduction for Hamiltonian systems have been comprehensively studied for several decades (see e.g. the textbooks of Marsden and Ratiu 1999; Marsden 1993 and for more recent work, e.g. Bullo and Lewis 2007; Roberts et al. 2002), reduction procedures directly on Lagrangian systems have gained less attention (cf. Marsden and Scheurle 1993). Related work studies symmetry properties of relative equilibria and design feedback control laws directly based on a symmetry splitting of the state space (Simo et al. 1991; Bloch et al. 2000). Families of relative equilibria in Hamiltonian systems can be numerically computed by path following methods to study e.g. bifurcation phenomena (Wulff and Schilder 2009).

There also exists an intensive research in mechanical (control) systems on Lie groups, i.e. if the overall state space has Lie group structure (see e.g. Bullo and Lewis 2004 or Kobilarov and Marsden 2011). To avoid confusion, in our setting the configuration manifold is an arbitrary manifold Q on which a Lie group G operates by symmetry actions $\Phi : G \times Q \to Q$. Typically the invariance is found only in some coordinates, thus the remaining coordinates have to be left unchanged by the symmetry action. If the Lagrangian of a mechanical system does not explicitly depend on a coordinate (but on the corresponding velocities), it is called *cyclic*. Reduction of cyclic Lagrangian was considered by Routh (see e.g. Marsden and Ratiu 1999), who called relative equilibria *steady motions* since they are the equilibrium points of the reduced Euler–Lagrange equations. In Sect. 5.1 the simple spherical pendulum is considered as an example of a cyclic Lagrangian system. In contrast, the double spherical pendulum is not cyclic and therefore has to be addressed by the extended *Lagrangian reduction* (see Marsden and Scheurle 1993 and Sects. 3.2 and 5.2).

Invariant Manifolds in Natural Dynamics In the control of mechanical, electrical or mechatronic systems, minimizing the energetic effort is often of particular importance. Thus it is obvious that trajectories of the natural, i.e. uncontrolled dynamics that are free of cost, should be used whenever the planning scenario allows for it. However, even the natural dynamics of nonlinear systems are typically quite complicated such that an analysis, by analytical and / or numerical means, is necessary to identify promising candidates for planning scenarios. In this work, we study the use of trajectories on (un)stable manifolds of the natural dynamics for motion planning purposes. The manifolds arise at invariant objects, in the simplest case an equilibrium or a periodic orbit. Near these critical objects, the manifolds are tangent to the eigenspaces of the system's linearization. Eigenvalues with zero real part give rise to so called center manifolds. Conversely, if the spectrum is hyperbolic, i.e. it has no eigenvalues on the imaginary axis, the stable and unstable invariant subspaces span the entire state space. The stable manifold consists of all points that tend to the critical object under the system's flow; points of the unstable manifold show the same behavior in backward time (see any textbook on dynamical systems, e.g. Guckenheimer and Holmes 1983 or Abraham and Marsden 1987 for a focus on Hamiltonian and Lagrangian systems). We will go into this in more detail in Sect. 3.3.

Since the studies of orbit structures in celestial mechanics performed in Conley (1968), McGehee (1969), invariant manifolds have been exploited in this spirit for a variety of space mission trajectories for the energy efficient transport between different planets and their nearby orbits (see e.g. Gómez et al. 2004; Koon et al. 2001 among numerous others). This concept has been extended in such a way that (un)stable manifolds of several different systems are used as partial orbits that are concatenated by appropriately controlled maneuvers (see e.g. Koon et al. 2000 and related work of these authors or Dellnitz et al. 2009).

Throughout the present work, all (uncontrolled) systems are assumed to be autonomous, i.e. not explicitly time-dependent except for the control force that is a function of time. However, in a non-autonomous setting, which arises e.g. when studying fluid dynamics or ocean flow dynamics, organizing structures that are related to (un)stable manifolds, e.g. Lagrangian coherent structures, have been detected (Haller 2001; Haller and Yuan 2000) and studied in a number of preceding works (see e.g. Froyland and Padberg 2009 for a comparing description of computational techniques).

Contributions This work extends recent results on motion planning for systems with symmetries (Frazzoli et al. 2005) to include new kinds of motion primitive—trajectories along stable or unstable manifolds of equilibria or periodic orbits. Thus, the resulting solutions exploit the structure of the system dynamics to a higher degree. On the theoretical side, the study of relative equilibria in geometric mechanics is extended to mechanical systems with a special kind of control forces. This enables us to identify candidates that satisfy the definition of trim primitives. In addition, maneuvers which serve as transition in the motion library are designed using structure-respecting optimal control that provably preserves symmetries and motion invariants. We develop a numerical framework based on (un)stable manifolds, relative equilibria, and optimized maneuvers and apply it to a nontrivial example, the double spherical pendulum.

The outline is as follows: In Sect. 2 a short introduction in optimal control formulations, variational mechanics and the optimal control method DMOC is given. Next, we analyze the inherent dynamical structures of mechanical systems that can be exploited for optimal control in Sect. 3, i.e. symmetry, relative equilibria and (un)stable manifolds. In Sect. 4 it is explained in detail, how we perform the motion planning with primitives. Numerical results for two examples, a simple and a double spherical pendulum are presented in Sect. 5 which have been partly presented in a short version of this work in Flaßkamp and Ober-Blöbaum (2012). Finally, Sect. 6 concludes by pointing out several directions of further research.

2 Preliminaries

In this work, optimal control problems for complex nonlinear systems are studied and solved by numerical methods. Here, we focus on systems that can be modeled by Lagrangian mechanics. In this section, we briefly introduce the framework our research is based upon.

2.1 Optimal Control

Consider a system with time-dependent state $x(t) \in X$ controlled using time-dependent actuator input $u(t) \in U$, where X is the state space and U denotes the set of controls. The dynamics is described by the function $f : X \times U \to TX$ defined by

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

which is used to evolve the state forward in time. In addition, the system is subject to constraints arising from actuator bounds and forbidden regions in the state space. These constraints are expressed through the vector of inequalities

$$h\big(x(t), u(t)\big) \geq 0, \tag{2}$$

for all $t \in [0, t_\mathrm{f}]$, where $t_\mathrm{f} > 0$ is the final time of the trajectory. The goal is to compute the optimal controls u^* and time t_f^* driving the system from its initial state $x_0 \in X$ to a given goal region $\mathcal{X}_\mathrm{f} \subset X$, i.e.

$$\begin{aligned} \big(u^*, t_\mathrm{f}^*\big) &= \arg\min_{u, t_\mathrm{f}} \int_0^{t_\mathrm{f}} C\big(x(t), u(t)\big)\,\mathrm{d}t, \\ \text{subject to} \quad & \dot{x}(t) = f\big(x(t), u(t)\big), \\ & h\big(x(t), u(t)\big) \geq 0, \quad x(0) = x_0,\ x(t_\mathrm{f}) \in \mathcal{X}_\mathrm{f} \end{aligned} \tag{3}$$

for all $t \in [0, t_\mathrm{f}]$ and where $C : X \times U \to \mathbb{R}$ is a given cost function. A typical cost function includes a time component and a control effort component, e.g. $C(x(t), u(t)) = \lambda_1 \cdot 1 + \lambda_2 \cdot \|u(t)\|^2$ where $\lambda_{1,2} \geq 0$ are chosen weights and $\|\cdot\|$ is the 2-norm. The problem (3) has no closed-form solution since both the dynamics (1) and constraints (2) are nonlinear.

There exist a number of different approaches for numerically solving optimal control problems, for a good overview we recommend Binder et al. (2001) and the references therein. The solution methods can be divided into indirect and direct methods. While indirect methods generate and then solve a boundary value problem according to the necessary optimality conditions of the Pontryagin maximum principle (cf. e.g. Binder et al. 2001), direct methods start with a discretization of the problem (3). Thus one obtains a nonlinear optimization problem that can be addressed by appropriate state of the art techniques such as sequential quadratic programming (SQP, cf. e.g. Gill et al. 2000). These methods require derivative information of the constraints and the objective, which can be either approximated by finite differences, or provided by analytical expressions if at hand, or computed by algorithmic differentiation (cf. Griewank and Walther 2008).

However, gradient-based optimization is not suitable unless a good starting guess is chosen since typically there are many local minima. Thus, instead of solving (3) numerically using a black box nonlinear programming tool such as SQP we reformulate the problem by exploiting any existing structure in the dynamics. This is accomplished by considering symmetries and invariant manifolds described in Sect. 3.

2.2 Variational Mechanics

An important class of dynamical systems that are rich in inherent structural properties are mechanical systems. The study of mechanical systems from the perspective of differential geometry has a long history (cf. e.g. Abraham and Marsden 1987; Marsden and Ratiu 1999; Marsden and West 2001). However, geometric mechanics is an active field of research, in particular regarding optimal control problems (Bloch 2003; Bullo and Lewis 2004; Ober-Blöbaum et al. 2011). The following descriptions are mainly taken from Flaßkamp and Ober-Blöbaum (2012).

Let Q be an n-dimensional configuration manifold with tangent bundle TQ and cotangent bundle T^*Q. Consider a mechanical system with time-dependent configuration vector $q(t) \in Q$ and velocity vector $\dot{q}(t) \in T_{q(t)}Q$, $t \in [0, t_\mathrm{f}]$, whose dynamical behavior is described by the Lagrangian $L : TQ \to \mathbb{R}$. Typically, the Lagrangian L consists of the difference of the kinetic and potential energy. In addition, there is a force $f : TQ \times U \to T^*Q$ acting on the system. This force depends on a time-dependent control parameter $u(t) \in U \subseteq \mathbb{R}^m$ that influences the system's motion. The equations of motion can be described via a variational principle. Define the *action map* $\mathfrak{S} : C^2([0, t_\mathrm{f}], Q) \to \mathbb{R}$ as

$$\mathfrak{S}(q) = \int_0^{t_\mathrm{f}} L\big(q(t), \dot{q}(t)\big)\,\mathrm{d}t.$$

Then the *Lagrange–d'Alembert principle* seeks curves $q \in C^2([0, t_\mathrm{f}], Q)$ with fixed initial value $q(0)$ and fixed final value $q(t_\mathrm{f})$ satisfying

$$\delta \int_0^{t_\mathrm{f}} L(q, \dot{q})\,\mathrm{d}t + \int_0^{t_\mathrm{f}} f(q, \dot{q}, u) \cdot \delta q\,\mathrm{d}t = 0 \tag{4}$$

for all variations $\delta q \in T_q C^2([0, t_\mathrm{f}], Q)$, where the second integral in (4) is the *virtual work* acting on the mechanical system via the force f. This yields the *forced Euler–Lagrange equations*

$$\frac{\partial L}{\partial q}(q, \dot{q}) - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}}(q, \dot{q})\right) + f(q, \dot{q}, u) = 0. \tag{5}$$

For the unforced Euler–Lagrange equations ($f = 0$), we denote the *Lagrangian vector field* by X_L and by $F_\mathrm{L} : TQ \times [0, t_\mathrm{f}] \to TQ$ its *flow*. If we fix the time t, we write $F_\mathrm{L}^t : TQ \to TQ$. In case there is external forcing acting on the system, (5) implicitly defines a family of *forced Lagrangian flows* F_L^u for fixed curves $u : [0, t_\mathrm{f}] \to U$. The *forced Lagrangian vector field* is then denoted by X_L^u.

2.3 Discrete Mechanics and Optimal Control

To formulate the optimal control problem for controlled Lagrangian systems, we replace in Sect. 2.1 the state space X by the tangent bundle TQ by setting $x(t) = (q(t), \dot{q}(t))$ and replace the differential equation in the optimal control problem (3) by the forced Euler–Lagrange equations (5). Recently, the direct optimal control method DMOC (*Discrete Mechanics and Optimal Control*, Ober-Blöbaum et al. 2011) was developed to numerically solve optimal control problems of Lagrangian systems and thereby taking the special structure of mechanical systems into account. Using concepts from discrete variational mechanics, DMOC is based on a direct discretization of the Lagrange–d'Alembert principle of the mechanical system. The goal of discrete variational mechanics is to derive discrete approximations of the solutions of the forced Euler–Lagrange equations that inherit the same qualitative behavior as the continuous solution.

The continuous optimal control problem is transformed into a finite dimensional constrained optimization problem using a global discretization of the states and the controls. The state space TQ is replaced by $Q \times Q$ and the discretization grid is defined by $\Delta t = \{t_k = kh \mid k = 0, \ldots, N\}$, $Nh = t_\mathrm{f}$, where N is a positive integer and h is the step size. The path $q : [0, t_\mathrm{f}] \to Q$ is replaced by a discrete path $q_\mathrm{d} : \{t_k\}_{k=0}^N \to Q$, where $q_k = q_\mathrm{d}(kh)$ is an approximation to $q(kh)$ (Marsden and West 2001; Ober-Blöbaum et al. 2011). Similarly, the control path $u : [0, t_\mathrm{f}] \to U$ is replaced by a discrete one. To this end, a refined grid, $\Delta\tilde{t}$, is generated via a set of control points $0 \le c_1 < \cdots < c_s \le 1$ and $\Delta\tilde{t} = \{t_{k\ell} = t_k + c_\ell h \mid k = 0, \ldots, N-1; \ell = 1, \ldots, s\}$. The discrete control path is defined to be $u_\mathrm{d} : \Delta\tilde{t} \to U$. The intermediate control samples $u_k = (u_{k1}, \ldots, u_{ks}) \in U^s$ on $[t_k, t_{k+1}]$ are defined to be the values of the control parameters guiding the system from $q_k = q_\mathrm{d}(t_k)$ to $q_{k+1} = q_\mathrm{d}(t_{k+1})$, where $u_{kl} = u_\mathrm{d}(t_{kl})$ for $l \in \{1, \ldots, s\}$.

To construct a discrete version of the Lagrange–d'Alembert principle, a *discrete Lagrangian* $L_\mathrm{d} : Q \times Q \to \mathbb{R}$ is defined that approximates the action integral along the curve segment between two adjacent points q_k and q_{k+1} as

$$L_\mathrm{d}(q_k, q_{k+1}) \approx \int_{kh}^{(k+1)h} L\big(q(t), \dot{q}(t)\big)\,\mathrm{d}t. \tag{6}$$

The *discrete action* is given by the sum of the discrete Lagrangian on each adjacent pair $\mathfrak{S}_\mathrm{d}(q_\mathrm{d}) = \sum_{k=0}^{N-1} L_\mathrm{d}(q_k, q_{k+1})$. Similarly, the virtual work can be approximated via

$$\begin{aligned}&\sum_{k=0}^{N-1} f_\mathrm{d}^-(q_k, q_{k+1}, u_k) \cdot \delta q_k + f_\mathrm{d}^+(q_k, q_{k+1}, u_k) \cdot \delta q_{k+1} \\ &\quad \approx \int_0^{t_\mathrm{f}} f\big(q(t), \dot{q}(t), u(t)\big) \cdot \delta q(t)\,\mathrm{d}t \end{aligned} \tag{7}$$

with the left and right discrete forces $f_\mathrm{d}^\pm(q_k, q_{k+1}, u_k) := f_k^\pm$. Based on these discrete objects, the *discrete Lagrange–d'Alembert principle* seeks discrete curves of points $\{q_k\}_{k=0}^N$ satisfying

$$\delta\mathfrak{S}_\mathrm{d} + \sum_{k=0}^{N-1} f_k^- \cdot \delta q_k + f_k^+ \cdot \delta q_{k+1} = 0 \tag{8}$$

for all variations δq_k vanishing at the endpoints. With $\delta q_0 = \delta q_N = 0$, (8) is equivalent to the *discrete forced Euler–Lagrange equations*

$$D_1 L_\mathrm{d}(q_k, q_{k+1}) + D_2 L_\mathrm{d}(q_{k-1}, q_k) + f_k^- + f_{k-1}^+ = 0 \tag{9}$$

for each $k = 1, \ldots, N-1$, where D_i denotes the derivative w.r.t. the ith argument. For given control values u_k, (9) provides a time stepping scheme for the simulation of the mechanical system which is called a *variational integrator* (cf. Marsden and West 2001). Since these integrators, derived in a variational way, are structure preserving, important properties of the continuous system are preserved (or change consistently with the applied forces), such as symplecticity or momentum maps induced by symmetries. In addition, they have an excellent long-time energy behavior. However, rather than solving initial value problems, an optimal control problem has to be solved, which involves the minimization of a cost functional $J(x, u) = \int_0^{t_\mathrm{f}} C(x(t), u(t))\,\mathrm{d}t$. Thus, in the same manner, an approximation of the cost functional generates the discrete cost functions C_d and J_d, respectively. The resulting nonlinear restricted optimization problem reads

$$\min_{q_\mathrm{d}, u_\mathrm{d}} J_\mathrm{d}(q_\mathrm{d}, u_\mathrm{d}) = \min_{q_\mathrm{d}, u_\mathrm{d}} \sum_{k=0}^{N-1} C_\mathrm{d}(q_k, q_{k+1}, u_k) \tag{10}$$

subject to the discrete forced Euler–Lagrange equations (9) and optionally discretized boundary and (in-)equality constraints for states and / or controls. Thus, the discrete forced Euler–Lagrange equations serve as equality constraints for the optimization problem which can be solved by standard optimization methods like SQP.

The approximation order of the optimal control scheme depends on the quadrature rule used to approximate the relevant integrals in (6) and (7). In general, one uses polynomial approximations to the trajectories and numerical quadrature to approximate the integrals. Then the order of the discrete Lagrangian and the discrete

forces is given by the order of the quadrature rule in use (e.g. second order using a midpoint rule approximation and assuming constant control parameters on each time interval with $l = 1$ and $c_1 = \frac{1}{2}$).

In Ober-Blöbaum et al. (2011), a detailed analysis of DMOC resulting from this discrete variational approach is given. The optimization scheme is symplectic-momentum consistent, i.e. the symplectic structure and the momentum maps corresponding to symmetry groups are consistent with the control forces for the discrete solution independent on the step size h. Thus, the use of DMOC leads to a reasonable approximation to the continuous solution, also for large step sizes, i.e. a small number of discretization points. In this work, DMOC will be used to compute the short controlled maneuvers mentioned above as well for a post-optimization of the found sequence. DMOC maneuvers combined with trims have been successfully used before in Kobilarov (2008) to build up a motion planning library for the optimal control of an autonomous helicopter.

3 Structures in Mechanical Systems

The structure of mechanical systems is now studied in more detail. Each of the listed properties below can be advantageously used in the motion planning approach for optimal control problems. While symmetry exploiting methods in motion planning using trim primitives has been already proposed in Frazzoli et al. (2005), the incorporation of trajectories on (un)stable manifolds in this framework came up quite recently (Flaßkamp et al. 2010; Flaßkamp and Ober-Blöbaum 2012), motivated by successful applications of (un)stable manifolds in space mission design (e.g. Serban et al. 2002).

3.1 Symmetry

Symmetries are present in a variety of mechanical systems and have been extensively studied and analyzed during the last years. In this section the general setting of symmetries in mechanical systems is introduced. We mainly follow the concept of Marsden and West (2001). For a detailed formulation and analysis of symmetries in unforced Lagrangian systems we also refer to Marsden (1994), Marsden and Ratiu (1999), Bloch (2003), Marsden and Scheurle (1993).

Assume a *Lie group* G with Lie algebra $\mathfrak{g}$ acts on the configuration manifold Q by a *left-action* $\Phi : G \times Q \to Q$. For each $g \in G$ we denote by $\Phi_g : Q \to Q$ the diffeomorphism defined by $\Phi_g := \Phi(g, \cdot)$. Let $\Phi^{TQ} : G \times TQ \to TQ$ for $(q, v) \in TQ$ be the tangent lift of the action given by $\Phi_g^{TQ}(q, v) = T(\Phi_g) \cdot (q, v)$. The symmetry of the unforced mechanical system corresponds to the invariance of the Lagrangian under the group action, i.e. $L \circ \Phi_g^{TQ} = L$ for all $g \in G$. One also says: the group action is a symmetry of the Lagrangian. The presence of a symmetry leads to the notion of equivalent trajectories in the following way.

Definition 3.1 (Equivalence of trajectories) Two trajectories $\pi_1 : t \in [t_{\mathrm{i},1}, t_{\mathrm{f},1}] \mapsto (q_1(t), \dot{q}_1(t), u_1(t))$ and $\pi_2 : t \in [t_{\mathrm{i},2}, t_{\mathrm{f},2}] \mapsto (q_2(t), \dot{q}_2(t), u_2(t))$ of (5) are equivalent, if we have

(i) $t_{\mathrm{f},1} - t_{\mathrm{i},1} = t_{\mathrm{f},2} - t_{\mathrm{i},2}$ and
(ii) there exist a $g \in G$ and a $T \in \mathbb{R}$, such that $(q_1, \dot{q}_1)(t) = \Phi_g^{TQ}((q_2, \dot{q}_2)(t - T))$ and $u_1(t) = u_2(t - T)\ \forall t \in [t_{\mathrm{i},1}, t_{\mathrm{f},1}]$.

Thus, equivalent trajectories can be constructed by a group action and a time shift only. All equivalent trajectories can be summed up in an equivalent class. By a slight abuse of notation, we call the equivalent class, but also its representative, a *motion primitive* (cf. Frazzoli et al. 2005). The number of candidates for the motion planning library can be immensely reduced by exploiting the system's invariance. Only one representative has to be stored and can then be used in different regions of the state space by a transformation of the lifted action.

Invariance and Lagrangian Flows Since G leaves the set of solutions of the variational principle invariant, the group action commutes with the Lagrangian flow F_{L} (Marsden et al. 1998). Furthermore, the invariance of the Lagrangian leads to the preservation of specific quantities by the Lagrangian flow.

For $\xi \in \mathfrak{g}$, let $\Phi^{\xi} : \mathbb{R} \times Q \to Q$ be the $\mathbb{R}$-action given by $\Phi^{\xi}(t, q) = \Phi(\exp(t\xi), q)$. The *infinitesimal generator* defined as $\xi_Q(q) = \frac{\mathrm{d}}{\mathrm{d}t}|_{t=0} \Phi(\exp(t\xi), q)$ is a vector field on Q while $\Phi(\exp(t\xi), \cdot) : Q \to Q$ is the corresponding flow on Q.

Assume that $L(q, v) = T(q, v) - V(q)$, where $V(q)$ is a G-invariant potential. G acts by isometries on the kinetic energy term, which can be written as $T(q, v) = \frac{1}{2} v^{\mathrm{T}} M(q) v = \frac{1}{2} \langle\langle v, v \rangle\rangle$ with mass matrix M and $\langle\langle \cdot, \cdot \rangle\rangle$ its induced inner product. The *Lagrangian momentum map* for a G-invariant Lagrangian L is defined by $\langle J(q, v), \xi \rangle = \langle \frac{\partial L}{\partial \dot{q}}(q, v), \xi_Q(q) \rangle = \langle\langle v, \xi_Q(q) \rangle\rangle$. Here, $\langle \cdot, \cdot \rangle$ denotes the natural pairing between elements of $T_q Q$ and its dual $T_q^* Q$.

The symmetry of Lagrangian systems leads to preservation of the associated momentum map as stated by Noether's theorem (see e.g. Marsden and West 2001).

Theorem 3.2 (Noether's Theorem) *Let $L : TQ \to \mathbb{R}$ be invariant under the lift of the action $\Phi : G \times Q \to Q$ as defined above, then the corresponding Lagrangian momentum map $J : TQ \to \mathfrak{g}^*$ is a conserved quantity for the flow, i.e. $J \circ F_{\mathrm{L}}^t = J$ for all times t.*

In general, arbitrary forcing would destroy the symmetry of Lagrangian systems since it breaks the conservation of the momentum map. However, as the forced Noether's theorem states, forcing that is orthogonal to the group action preserves symmetry (Marsden and West 2001; Ober-Blöbaum et al. 2011).

Theorem 3.3 (Forced Noether's Theorem) *Let the Lagrangian L and the symmetry action Φ be as in Theorem 3.2. Consider a force $f_{\mathrm{L}} : TQ \times U \to T^*Q$ such that $\langle f_{\mathrm{L}}(q, \dot{q}, u), \xi_Q(q) \rangle = 0$ for all $(q, \dot{q}) \in TQ$, $u(t) \in U\ \forall t$ and all $\xi \in \mathfrak{g}$. Then the Lagrangian momentum map $J : TQ \to \mathfrak{g}^*$ is preserved by the forced Lagrangian flow, i.e. $J \circ (F_{\mathrm{L}}^u)^t = J$ for all t.*

3.2 Relative Equilibria and Trim Primitives

The presence of symmetry gives rise to the existence of a special kind of trajectories, i.e. motions that are solely generated by the symmetry action. These group orbits are therefore called *relative equilibria*.

Definition 3.4 (Relative equilibrium) A point $x_e = (q_e, v_e) \in TQ$ is called a relative equilibrium, if $X_L(x_e) \in T_{x_e}(G \cdot x_e)$, i.e. the Lagrangian vector field X_L at x_e points in the direction of the group orbit $G \cdot x_e = \{x = (q, v) | (q, v) = \Phi_g^{TQ}(q_e, v_e) \text{ for } g \in G\}$.

Finding relative equilibria is closely related to reduction processes, since relative equilibria correspond to fixed points of reduced equations of motion (Marsden 1993). Roughly speaking, the conservation of momentum maps (as stated by Noether's theorem) can be used to reduce the system's dynamic equations by constraining them to a fixed momentum value. In the following we give a brief overview of Lagrangian reduction techniques.

Lagrangian Reduction The symmetry reduction method for Lagrangian systems is called the *Lagrangian reduction method* and can be seen as the counterpart of the common *symplectic reduction method* or *energy-momentum method* for Hamiltonian systems. Lagrangian reduction (cf. Marsden and Scheurle 1993) is a generalization of the classical Routh reduction (see e.g. Bloch 2003) for cyclic variables. In the following, we recall some results of Marsden and Scheurle (1993) that equip us with a method to compute relative equilibria for mechanical systems and that can be extended to Lagrangian systems with forcing.

For each $q \in Q$ the *locked inertia tensor* $\mathbb{I} : \mathfrak{g} \to \mathfrak{g}^*$ is defined by $\langle \mathbb{I}(q)\eta, \zeta \rangle = \langle\langle \eta_Q(q), \zeta_Q(q) \rangle\rangle$ with η_Q, ζ_Q being the infinitesimal generators to $\eta, \zeta \in \mathfrak{g}$. It can be interpreted as the inertia tensor of a system which moves only in the direction of the infinitesimal generators of the symmetry action, as e.g. a multi-body system that has been locked to a rigid structure. The corresponding angular velocity is then given by $\alpha(q, v) = \mathbb{I}(q)^{-1} J(q, v)$, called the *mechanical connection*. For each $\mu \in \mathfrak{g}^*$, it leads to the definition of a one form on Q, denoted by α_μ with $\langle \alpha_\mu(q), v \rangle = \langle \mu, \alpha(q, v) \rangle$. The *amended potential* is defined by $V_\mu(q) = V(q) + \frac{1}{2}\langle \mu, \mathbb{I}(q)^{-1}\mu \rangle$ (cf. Marsden and Scheurle 1993) and plays an important role in reduction processes. For a given value $\mu \in \mathfrak{g}^*$ of the momentum map, the Routhian $R^\mu : TQ \to \mathbb{R}$ is defined as $R^\mu(q, v) = L(q, v) - \langle \alpha(q, v), \mu \rangle$. Fixing the level set of the momentum map, i.e. $J(q, \dot{q}) = \mu$ it can be shown (see Marsden and Scheurle 1993 for the variational derivation) that the original Euler–Lagrange equations are equivalent to the Euler–Lagrange equations of the Routhian R^μ with an additional gyroscopic forcing term, reading

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial R^\mu}{\partial \dot{q}} - \frac{\partial R^\mu}{\partial q} = \dot{q}^{\mathrm{T}} \beta.$$

Here, β is the *magnetic two form* on Q, $\beta(q) : T_qQ \times T_qQ \to \mathbb{R}$, defined by $\beta = \mathbf{d}\alpha_\mu$, i.e. in coordinates, $\beta_{ij} = \frac{\partial \alpha_j}{\partial q^i} - \frac{\partial \alpha_i}{\partial q^j}$. (Recall that α_μ is a one form on Q, so in coordinates, $\alpha_\mu = \alpha_i \, \mathrm{d}q^i$ with $\mathrm{d}q^i$ being the basis vectors for T_q^*Q.) Based on

a splitting of the configuration manifold into the symmetry group G and the shape space $S = Q/G$, each vector $(q, v) \in T_q Q$ can be decomposed into its horizontal and its vertical part, $v = \mathrm{hor}_q v + \mathrm{ver}_q v$, where $\mathrm{ver}_q v = [\alpha(q, v)]_Q(q)$ and $\mathrm{hor}_q v = v - \mathrm{ver}_q v$. That means, the vertical part belongs to the vertical space of the connection and consists of all points that are mapped to zero under the projection from Q to S. These are the infinitesimal generators. The horizontal part is an element of the space that is orthogonal to the G-orbits, given by $\mathrm{hor}_q = \{(q, v) | J(q, v) = 0\}$. In Marsden and Scheurle (1993), it is shown that for fixed level sets of J, the Routhian can be reduced to $R^\mu = \frac{1}{2}\|\mathrm{hor}(q, v)\|^2 - V_\mu$. Hence, the Euler–Lagrange equations for R^μ can be reduced as well and the following statement can be deduced (cf. e.g. Marsden and Scheurle 1993, Prop. 3.5):

Proposition 3.5 *A point $x_\mathrm{e} = (q_\mathrm{e}, v_\mathrm{e})$ is a relative equilibrium if and only if q_e is a critical point of the amended potential V_μ with $\mu = J(q_\mathrm{e}, v_\mathrm{e})$.*

For control purposes it makes sense to generalize the definition of relative equilibria to forced systems by allowing constant control values. We will use the terminology of Frazzoli et al. (2005) and call them *trim primitives* originated from *trimmed motions*.

Definition 3.6 (Trim primitives) A point $x_\mathrm{e} = (q_\mathrm{e}, v_\mathrm{e})$ together with some control value $u_\mathrm{e} \in U$ is called a *trim primitive* (or shortly a *trim*), if we have $X_\mathrm{L}^{u_\mathrm{e}}(x_\mathrm{e}) \in T_{x_\mathrm{e}}(G \cdot x_\mathrm{e})$ with the forced Lagrangian vector field $X_\mathrm{L}^{u_\mathrm{e}}$.

In other words, trims generate solutions $(q(t), \dot{q}(t))$ on $[0, t_\mathrm{f}]$ of the forced Euler–Lagrange equations with control $u(t)$ for a G-invariant Lagrangian L and forcing f_L, which can be written as $(q, \dot{q})(t) = \Phi^{TQ}(\exp(t\xi), (q_\mathrm{e}, v_\mathrm{e}))$, $u(t) = u_\mathrm{e} = \mathrm{const.}$ $\forall t \in [0, t_\mathrm{f}]$ with $\xi \in \mathfrak{g}$ and $\exp : \mathfrak{g} \to G, \xi \mapsto \exp(t\xi) \in G$. Trims are uniquely defined by their initial value $(q_0, \dot{q}_0, u_0)$ and the Lie algebra element ξ, which makes them easy to store and handle in a library of motion primitives. A second benefit of trims is that they are simply parametrized by time, i.e. their duration need not be fixed in advance, but can be adjusted during the sequencing (cf. Frazzoli et al. 2005).

In the following we will introduce the concept of controlled potentials that provides a method to construct trim primitives based on the computation of relative equilibria.

Controlled Potentials We augment the original potential $V(q)$ by a parameter-dependent term, representing potential forces, i.e. a special kind of forcing that is defined by a potential (cf. Bullo and Lewis 2004). That means, we replace $V : Q \to \mathbb{R}$ by $V^u : Q \to \mathbb{R}$, $V^u(q) = V(q) - \nu(q)$ with $\nu : Q \to \mathbb{R}$ having the property that $\frac{\partial}{\partial q}\nu(q) = u$ for some control value $u \in U$, where we assume that $U \subseteq \mathbb{R}^n$.

This type of control is intrinsically restricted to depend on configurations, so cannot be used to model dissipative, i.e. velocity dependent forces. However, many examples of control forces on mechanical systems fit into this structure.

The following theorem describes how a trim primitive for a controlled Lagrangian system can be computed by means of the concept of controlled potentials.

Theorem 3.7 *Let $L = T - V$ be a G-invariant Lagrangian and $V^u(q) = V(q) - \nu(q)$ the augmented, G-invariant controlled potential. The critical points of the amended controlled potential V^u_μ are relative equilibria of the forced Lagrangian vector field, X^u_{L}, i.e. trim primitives according to Definition 3.6.*

Proof Amending the controlled potential V^u leads to the amended controlled potential $V^u_\mu = V^u + \frac{1}{2}\langle \mu, \mathbb{I}(q)^{-1}\mu\rangle = V(q) - \nu(q) + \frac{1}{2}\langle \mu, \mathbb{I}(q)^{-1}\mu\rangle = V_\mu - \nu(q)$. Since we assume V^u to be G-invariant, Proposition 3.5 can be applied to the modified system given by the Lagrangian $L^u = T - V^u$, i.e. relative equilibria are given by the critical points of V^u_μ:

$$\frac{\partial}{\partial q} V^u_\mu = 0 \quad \Leftrightarrow \quad \frac{\partial}{\partial q}\big(V_\mu - \nu(q)\big) = 0 \quad \Leftrightarrow \quad \frac{\partial}{\partial q} V_\mu = u.$$

In other words, if a pair $(x_{\mathrm{e}}, u_{\mathrm{e}}) = ((q_{\mathrm{e}}, v_{\mathrm{e}}), u_{\mathrm{e}})$ satisfies $\frac{\partial}{\partial q} V_\mu(q_{\mathrm{e}}) = u_{\mathrm{e}}$ with $\mu = J(q_{\mathrm{e}}, v_{\mathrm{e}})$, the definition of a relative equilibrium, $X_{L^u}(x_{\mathrm{e}}) \in T_{x_{\mathrm{e}}}(G \cdot x_{\mathrm{e}})$, is fulfilled. The Euler–Lagrange equations of L^u read as follows: $\frac{\partial}{\partial q}(T(q,\dot q) - V^u(q)) - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial}{\partial \dot q} T(q,\dot q) = \frac{\partial}{\partial q}(T(q,\dot q) - V(q)) - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial}{\partial \dot q} T(q,\dot q) + u = 0$ and hence are equal to the forced Euler–Lagrange equations for L with forcing $f(q,\dot q, u) = \frac{\partial}{\partial q}\nu(q) = u$. Thus, the vector fields $X_{L^u} = X^u_{\mathrm{L}}$ coincide and therefore, $X^{u_{\mathrm{e}}}_{\mathrm{L}}(x_{\mathrm{e}}) \in T_{x_{\mathrm{e}}}(G \cdot x_{\mathrm{e}})$, i.e. $(x_{\mathrm{e}}, u_{\mathrm{e}})$ is a trim primitive as defined in Definition 3.6. □

Note that in Theorem 3.7 the condition that the controlled potential is G-invariant implicitly gives restrictions on ν and thus on the control u. The forced Noether's Theorem 3.3 suggests candidates for trim primitives, namely all trajectories with such controls that act orthogonal to the group action. Indeed the following corollary states that this orthogonality condition is in fact necessary for the construction of trim primitives.

Corollary 3.8 *If $x_{\mathrm{e}} = (q_{\mathrm{e}}, v_{\mathrm{e}})$ with control u_{e} is a trim primitive of a Lagrangian system with symmetry group G and G-invariant controlled potential $V^{u_{\mathrm{e}}} = V(q) - \nu(q)$ with $\frac{\partial}{\partial q}\nu(q) = u_{\mathrm{e}}$, it necessarily holds that $u_{\mathrm{e}} \cdot \xi_Q(q_{\mathrm{e}}) = 0$, with $\cdot$ denoting the standard scalar product. Here, ξ_Q is the infinitesimal generator of $\xi \in \mathfrak{g}$ such that $(q,\dot q)(t) = \Phi^{TQ}(\exp(t\xi), (q_{\mathrm{e}}, v_{\mathrm{e}}))$, $u(t) \equiv u_{\mathrm{e}}$ is a solution of the forced Euler–Lagrange equations.*

Proof It follows from the G-invariance of L that the original V is G-invariant, because we assume G to act by isometries and the kinetic energy is given in terms of a metric. Then, from the G-invariance of $V^{u_{\mathrm{e}}}$, i.e. $V^{u_{\mathrm{e}}}(\Phi(g,q)) = V^{u_{\mathrm{e}}}(q)$, it can be deduced that

$$V\big(\Phi(g,q)\big) - \nu\big(\Phi(g,q)\big) = V(q) - \nu(q) \quad \Leftrightarrow \quad \nu\big(\Phi(g,q)\big) - \nu(q) = 0.$$

As g is a point in the one-parameter subgroup $\mathbb{R} \ni s \to \exp(s\xi) \in G$ generated by $\xi \in \mathfrak{g}$, we can replace g by $\exp(s\xi)$, set q to the trim primitive value x_{e} and then

differentiate with respect to s and evaluate at $s = 0$:

$$0 = \frac{\mathrm{d}}{\mathrm{d}s}\big(v\big(\Phi\big(\exp(s\xi), q_\mathrm{e}\big)\big) - v(q_\mathrm{e})\big)\Big|_{s=0}$$
$$= \frac{\partial}{\partial q} v\big(\Phi\big(\exp(s\xi), q_\mathrm{e}\big)\big) \cdot \frac{\mathrm{d}}{\mathrm{d}s}\Phi\big(\exp(s\xi), q_\mathrm{e}\big)\Big|_{s=0} = u_\mathrm{e} \cdot \xi_Q(q_\mathrm{e}). \qquad \square$$

Hence, we received a necessary condition on u_e to be admissible for a trim primitive that can be used to compute trim primitives in example systems.

Note that depending on the system under consideration, it is not guaranteed to identify all trim primitives by numerically finding the zeros of the gradient of the amended potential. However, restricting to only some of all existing trim primitives does not make the motion planning approach fail. However, it does in fact reduce the number of controllable states.

3.3 Invariant Manifolds in Natural Dynamics

In this section, we analyze the system's *natural dynamics*, i.e. the unforced case of e.g. a mechanical system. Nonlinear dynamical systems may exhibit complicated structures, e.g. local attractors or invariant manifolds (Guckenheimer and Holmes 1983) that separate the state space. These structures are not at all obvious up to a careful and systematic analysis. However, there may be motions of the unforced system that can be of great interest in control problems when searching for energy efficient solutions. Stable manifolds are introduced in a number of textbooks on dynamical systems (e.g. Guckenheimer and Holmes 1983; Katok and Hasselblatt 1995 among others or Abraham and Marsden 1987 for mechanical systems). The following definitions are basically taken from the latter with a slightly different notation at some points.

Consider a vector field X on a manifold with its corresponding flow F^t, e.g. a Lagrangian vector field X_L on the tangent bundle TQ with flow $F_\mathrm{L}^t : TQ \to TQ$. A *critical element* is either an *equilibrium*, i.e. a point $\bar{x} \in TQ$ such that $X_\mathrm{L}(\bar{x}) = 0$ and, hence, $F_\mathrm{L}^t(\bar{x}) = \bar{x}$ for all $t \in \mathbb{R}$, or a *closed orbit*, i.e. the orbit of a periodic point ($F_\mathrm{L}^t(\bar{x}) = F_\mathrm{L}^{t+\tau}(\bar{x})$ with $\tau > 0$ being the smallest value that satisfies this condition).

Given an equilibrium point $\bar{x}$, we are interested in the eigenvalues of $X'_\mathrm{L}(\bar{x})$, i.e. the linearization of X_L at $\bar{x}$, $X'_\mathrm{L}(\bar{x}) : T_{\bar{x}}(TQ) \to T_{\bar{x}}(TQ)$ defined by $X'_\mathrm{L}(\bar{x}) \cdot v = \frac{\mathrm{d}}{\mathrm{d}\lambda}(TF_\mathrm{L}^\lambda(\bar{x}) \cdot v)|_{\lambda=0}$. In coordinates, the matrix $X'_\mathrm{L}(\bar{x})$ is given by $(\frac{\partial X_\mathrm{L}^i}{\partial x^j})|_{x=\bar{x}}$. It is a well known stability criterion that a system is asymptotically (un)stable, if all eigenvalues have strictly negative (resp. positive) real parts. In the following, we will study the case where there are eigenvalues on both sides of the imaginary axis. A critical point is called *hyperbolic*, if none of the corresponding linearization eigenvalues has zero real part.

To investigate the dynamic behavior near closed orbits, the Poincaré map of a transversal section S is studied. A *transversal section* of X_L at a point x on the orbit is a submanifold $S \subset TQ$ of codimension one with $x \in S$ and for all $s \in S$, $X_\mathrm{L}(s)$ is not contained in T_sS. Then, roughly speaking, the *Poincaré map* of a closed orbit γ is a diffeomorphism Θ between neighborhoods of x in S that assigns to each

neighboring point $s \in S$ the point, where the orbit $F_{\mathrm{L}}^{\rho(s)}(s)$ intersects S again for the first time. Here, $\rho(s)$ is the corresponding return time. (For a detailed description we refer to Abraham and Marsden 1987 or another textbook on dynamical systems.) For a closed orbit γ of a vector field X_{L}, the *characteristic multipliers* of X_{L} at γ are the eigenvalues of $T_x\Theta$ for any Poincaré map Θ at any $x \in \gamma$. γ is called *hyperbolic*, if none of the characteristic multipliers has modulus one. Analogous to the stability criterion of Lyapunov for equilibria, a period orbit is asymptotically (un)stable, if the modulus of all characteristic multipliers is less (resp. greater) than one.

Theorem 3.9 (cf. Abraham and Marsden 1987) *If $\gamma \subset TQ$ is a critical element of X_{L}, there exist submanifolds of TQ, i.e. local stable ($W^{\mathrm{s}}_{\mathrm{loc}}$), center-stable ($W^{\mathrm{cs}}_{\mathrm{loc}}$), center ($W^{\mathrm{c}}_{\mathrm{loc}}$), center-unstable ($W^{\mathrm{cu}}_{\mathrm{loc}}$), and unstable ($W^{\mathrm{u}}_{\mathrm{loc}}$) manifolds, respectively, with the following properties*:

(i) *each submanifold is invariant under X_{L} and contains γ,*
(ii) *For $x \in \gamma$, $T_x(W^{\mathrm{s}}_{\mathrm{loc}})$ is the sum of the eigenspace in $T_x(TQ)$ of the characteristic multipliers of modulus <1 and the subspace $T_x\gamma$; $T_x(W^{\mathrm{cs}}_{\mathrm{loc}})$ (resp. $T_x(W^{\mathrm{c}}_{\mathrm{loc}})$, $T_x(W^{\mathrm{cu}}_{\mathrm{loc}})$, $T_x(W^{\mathrm{u}}_{\mathrm{loc}})$) is the sum of the eigenspace in $T_x(TQ)$ of the characteristic multipliers of modulus ≤ 1 (resp. $=1$, ≥ 1, >1) and the subspace $T_x\gamma$.*
(iii) *If $x \in W^{\mathrm{s}}_{\mathrm{loc}}$, then the ω-limit, given by $\omega(x) = \bigcap_{T=0}^{\infty} \overline{(\bigcup_{t \geq T} F_{\mathrm{L}}^t(x))}$ is equal to γ. If $x \in W^{\mathrm{u}}_{\mathrm{loc}}$, then the α-limit is γ, with $\alpha(x) = \bigcap_{T=0}^{-\infty} \overline{(\bigcup_{t \leq T} F_{\mathrm{L}}^t(x))}$.*
(iv) *$W^{\mathrm{s}}_{\mathrm{loc}}$ and $W^{\mathrm{u}}_{\mathrm{loc}}$ are locally unique.*

Thus, all points of the local stable manifold $W^{\mathrm{s}}_{\mathrm{loc}}$ tend to the critical element under the evolution. Conversely, the local unstable manifold $W^{\mathrm{u}}_{\mathrm{loc}}$ consists of all points in TQ which show this behavior if time runs backwards. The dynamics on the center manifold is subject to a further analysis (see e.g. Abraham and Marsden 1987 and the references therein) but out of the scope for this work.

Remark 3.10 In case of a critical point, i.e. an equilibrium $\gamma = \bar{x}$, the tangent space is trivial, $T_{\bar{x}}\gamma = \{0\}$ and therefore, $T_{\bar{x}}(W^{\mathrm{s}}_{\mathrm{loc}})$ equals the eigenspace in $T_{\bar{x}}(TQ)$ of the characteristic multipliers of modulus <1. Further, for $\gamma = \bar{x}$, the characteristic multipliers have to be interpreted as the eigenvalues of $T_{\bar{x}}F_{\mathrm{L}}^t$, i.e. $\mathrm{e}^{t\mu_1}, \ldots, \mathrm{e}^{t\mu_n}$ where $\mu_1, \ldots, \mu_n$ are the eigenvalues of $X'_{\mathrm{L}}(\bar{x})$ (also called characteristic exponents). In other words, the stable manifold $W^{\mathrm{s}}_{\mathrm{loc}}$, for example, is defined by the eigenvalues that lie in the strict left plane ($\mathfrak{Re}(\mu_i) < 0$). In contrast, for γ being a closed orbit, $T_{\bar{x}}\gamma$ is the subspace generated by $X(\bar{x})$ that is included in all of the submanifolds defined above.

Of special interest is the hyperbolic case, where there are no center eigenspaces. Then the nearby orbits of γ behave qualitatively like the linear case, i.e. for a hyperbolic critical point, the flow nearby looks like that of the linearization at γ.

Corollary 3.11 (Global stable manifold theorem of Smale, cf. Abraham and Marsden 1987) *If γ is hyperbolic, then the stable manifold, $W^{\mathrm{s}}(\gamma) = \{x \in TQ | \omega(x) \subset \gamma\}$ and the unstable manifold, $W^{\mathrm{u}}(\gamma) = \{x \in TQ | \alpha(x) \subset \gamma\}$ are immersed submanifolds.*

Also, $\gamma \subset W^{\mathrm{s}}(\gamma) \cap W^{\mathrm{u}}(\gamma)$ *and for* $x \in \gamma$, $T_x W^{\mathrm{s}}(\gamma)$ *and* $T_x W^{\mathrm{u}}(\gamma)$ *generate* $T_x(TQ)$. *If* n_s *is the number of characteristic multipliers of* γ *of modulus* <1, *and* n_u *the number of modulus* >1, *then the dimension of* $W^{\mathrm{s}}(\gamma)$ (*resp.* $W^{\mathrm{u}}(\gamma)$) *is* n_s (*resp.* n_u) *if* γ *is a critical point, or* $n_s + 1$ (*resp.* $n_u + 1$) *if* γ *is a closed orbit.*

That means, the local (un)stable manifolds defined in Theorem 3.9 can be uniquely expanded to global manifolds by applying the flow of the vector field.

So far, we have not covered all of the structure of critical points of Lagrangian systems. Since a regular Lagrangian system can be transformed into a Hamiltonian system by the Legendre transformation, the eigenvalue spectrum of a critical point can be characterized even further. It is a well known result (see e.g. Abraham and Marsden 1987) that the linearization of a Hamiltonian system is a linear Hamiltonian system and therefore, if μ is an eigenvalue of $X'_H(\bar{x})$, then so are $\bar{\mu}, -\mu, -\bar{\mu}$. Therefore, stable and unstable manifolds of a critical point always have the same dimension and the center manifold, if existent, is even dimensional. Additionally, for a Lagrangian that equals kinetic minus potential energy, solely the second-order partial derivatives of the potential, i.e. $\frac{\partial^2}{\partial q^2} V$ determine the spectral characteristics. From the Lagrange–Dirichlet stability criterion (see e.g. Abraham and Marsden 1987; Marsden 1993), it follows that the system is stable, if the matrix $\frac{\partial^2}{\partial q^2} V$ evaluated at the equilibrium is positive definite. Then the eigenvalues lie on the imaginary axis. Otherwise, the system is unstable, because there has to be at least one eigenvalue with positive real part giving rise to an unstable manifold.

For another extension of the preceding theory for critical points, let us assume that the strongest stable and unstable eigenvalues, i.e. $\mu^{\mathrm{ss}} := \min_{\mu \in \sigma} \mathfrak{Re}(\mu)$ and $\mu^{\mathrm{uu}} := \max_{\mu \in \sigma} \mathfrak{Re}(\mu)$ where σ denotes the eigenvalue spectrum of the linearization at the equilibrium $\bar{x}$, are unique and real. Then we define the *strong (un)stable manifolds* $W^{\mathrm{ss}}(\bar{x})$ and $W^{\mathrm{uu}}(\bar{x})$ as the submanifold of the (un)stable manifold that are tangent to the eigenspace corresponding to the strongest (un)stable eigenvalues μ^{ss} and μ^{uu} (see e.g. Osinga et al. 2004).

However, in most cases it is not possible to compute these global invariant manifolds analytically. For that reason, a number of numerical techniques for approximating (un)stable manifolds has been developed in the last decades (see Krauskopf et al. 2005 for an overview of existing approaches and a comprehensive comparison of the methods for the example of the Lorenz system). The different methods all share the idea to successively grow the (un)stable manifold from a local neighborhood of the equilibrium. Among these techniques is GAIO (*Global Analysis of Invariant Objects*), a set oriented method (cf. Dellnitz et al. 2001) that we use for our numerical examples.

As discussed in Sect. 1, invariant manifolds have been used in many applications in astrodynamics to generate low energy trajectories. In this setting, trajectories along invariant manifolds provide pieces of maneuver sequences that are solutions of optimal control problems in space mission design (cf. e.g. Koon et al. 2000 or Dellnitz et al. 2006, 2009 for approaches based on set oriented computations by GAIO).

4 Motion Planning Using Primitives

Now we go into more detail describing the computational aspects of motion planning with primitives. First it is shown how the identified dynamical structures can be used to generate motion primitives. The maneuver automaton is introduced to organize the primitives in a library. Secondly, trajectory generation and the computation of motion plans are presented.

4.1 Maneuver Automaton

A library of primitives constitutes a *maneuver automaton*, i.e. a finite-state machine with states and transitions corresponding to motion primitives (cf. Frazzoli et al. 2005). In our work, the states correspond not only to trim primitives, but also to (un)stable manifolds. The automaton's transitions between different states correspond to maneuvers, i.e. short motions satisfying the boundary conditions imposed by the initial and final states that they connect.

A library is constructed by selecting a discrete set of trim primitives and trajectories on (un)stable manifolds. Let $\mathcal{E}$ denote the set of trims, chosen for instance by uniformly gridding a bounded subspace of the Lie algebra (see Definition 3.6), or alternatively by quantizing the space of internal (shape) variables and control input. For example, the elements of $\mathcal{E}$ can be computed by the critical points of the controlled amended potential (cf. Sect. 3.2). It is sufficient to select and store only a trim's initial value $\alpha(0) := (x_\alpha(0), u_\alpha(0))$, because the orbit can be constructed by the flow, $\alpha : t \in [0, t_\mathrm{f}] \mapsto (\Phi^{TQ}(\exp(t\xi), x_\alpha(0)), u_\alpha)$ with $\xi \in \mathfrak{g}$ and a constant control $u_\alpha \equiv u_\alpha(0)$.

Analogously, a finite set $\mathcal{O}$ of orbits on (un)stable manifolds, $\mathcal{O} \ni O : t \in [0, t_\mathrm{f}] \mapsto F_\mathrm{L}(x_O, t)$ has to be defined with some initial value x_O on the manifold. For motion planning purposes it is advantageous to select orbits with fast dynamical transition. Such motions correspond to trajectories on the *strong* (un)stable manifolds, because these are the directions of the most contraction to (expansion from, respectively) a critical element. Such choices are reasonable since we are interested in energy minimal or time minimal solutions (cf. Sect. 2.1). Because the strong (un)stable manifolds are one-dimensional (assuming simple strong (un)stable eigenvalues as in Osinga et al. 2004), orbits $O \in \mathcal{O}$ can be computed by choosing a starting point x_O in the close neighborhood and evolving the uncontrolled flow for some finite time t_f; in forward time for unstable, and in backward time for stable manifolds, respectively.

A maneuver is then designed to connect pairs of trim primitives and pairs of orbits on manifolds as well as pairs of a trim primitive and a manifold orbit. A fully connected automaton graph would thus require $n_\mathrm{t}(n_\mathrm{t}-1)+n_\mathrm{o}(n_\mathrm{o}-1)+n_\mathrm{t}n_\mathrm{o}$ maneuvers, where $n_\mathrm{t} = \dim(\mathcal{E})$ and $n_\mathrm{o} = \dim(\mathcal{O})$.

4.2 Maneuvers

We next describe the construction of maneuvers. Let the map $\varpi : X \to X\backslash G$ subtract out the invariant coordinates from a given state according to the system's symmetry

equivalence. Each maneuver is computed through nonlinear optimization of a trajectory whose start and end correspond to either a trim primitive or a manifold. This is defined through the following procedure:

$$\textit{Compute}: \quad t_\mathrm{f}; x : [0, t_\mathrm{f}] \to X;\ u : [0, t_\mathrm{f}] \to U \tag{11}$$

$$\textit{minimizing}: \quad J(x, u, t_\mathrm{f}) = \int_0^{t_\mathrm{f}} \left(\lambda_1 \cdot 1 + \lambda_2 \cdot \left\|u(t)\right\|^2\right) \mathrm{d}t, \tag{12}$$

$$\textit{subject to}: \quad \text{dynamics equation (5) for all } t \in [0, t_\mathrm{f}] \tag{13}$$

and *one* of the following boundary conditions:

$$\text{from trim } x_\alpha \text{ to trim } x_\beta: \quad \varpi\left(x(0)\right) \in \varpi(x_\alpha), \varpi\left(x(t_\mathrm{f})\right) \in \varpi(x_\beta), \tag{14}$$

$$\text{from trim } x_\alpha \text{ to manifold } O_\beta: \quad \varpi\left(x(0)\right) \in \varpi(x_\alpha), \varpi\left(x(t_\mathrm{f})\right) \in \varpi(O_\beta), \tag{15}$$

$$\text{from manifold } O_\alpha \text{ to trim } x_\beta: \quad \varpi\left(x(0)\right) \in \varpi(O_\alpha), \varpi\left(x(t_\mathrm{f})\right) \in \varpi(x_\beta), \tag{16}$$

$$\text{from manifold } O_\alpha \text{ to manifold } O_\beta: \quad \varpi\left(x(0)\right) \in \varpi(O_\alpha), \varpi\left(x(t_\mathrm{f})\right) \in \varpi(O_\beta), \tag{17}$$

with $\lambda_1, \lambda_2 \in \mathbb{R}_0^+$ and where $\varpi(x_\alpha)$ should be understood as a pointwise evaluation. In essence, the optimization is performed by not enforcing a given final group displacement or by allowing a maneuver to start and end anywhere on the initial and final manifold orbits, respectively.

Definition 4.1 (Maneuvers) A *maneuver* is a solution pair $\pi := (x^*, u^*)$, $\pi : t \in [0, t_\mathrm{f}] \mapsto (q^*(t), \dot{q}^*(t), u^*(t))$ to (11)–(17) that connects two automaton states, i.e. trim primitives or (un)stable manifolds.

More generally, the boundary constraints can adapted based on the problem. For instance, using the identity instead of ϖ in (14)–(17), the points on the primitives are completely fixed (including the invariant coordinates). This is important, if one wants to control the group displacement of the maneuver (cf. Sect. 4.3). In addition, a boundary point on a manifold orbit can either be a fixed point on the orbit, or an analytic expression of the entire orbit (or an approximation of it, e.g. by splines) can be used as a boundary constraint. As another design parameter, the weighting $\lambda_{1,2}$ has to be chosen to prioritize one objective over the other. Naturally in most applications, energy efficiency is contradictory to time optimality, rendering a multiobjective optimal control problem. That means that different values of $\lambda_{1,2}$ correspond to different optimal compromises of the concurring objectives—the so called *Pareto optimal solution* (Ehrgott 2005) of the optimal control problem. The prioritization of objectives leads to a scalarization of the vector valued cost functional. If this cannot or is not desired to be done in advance, several optimal control maneuvers for the same boundary conditions, but different values of $\lambda_{1,2}$ can be computed and stored in the motion planning library.

In our applications, the optimizations were performed offline such that all maneuvers are organized and saved in a library which is loaded at run-time providing instant

look-up during planning. The continuous optimal control formulation was computationally solved through the discrete mechanics methodology DMOC (Ober-Blöbaum et al. 2011; Marsden and West 2001; Kobilarov 2008) which is particularly suitable for systems with nonlinear state spaces and symmetries.

The key property of motion primitives is that they can be concatenated to create more complex motion sequences. Following Frazzoli et al. (2005), a concatenation of two motion primitives $\pi_1 : t \in [0, t_{\mathrm{f},1}] \mapsto (x_1(t), u_1(t))$ and $\pi_2 : t \in [0, t_{\mathrm{f},2}] \mapsto (x_2(t), u_2(t))$ on the time interval $[0, t_{\mathrm{f},1} + t_{\mathrm{f},2}]$ is defined by

$$\pi_1\pi_2(t) := \begin{cases} (x_1(t), u_1(t)) & \text{if } t \leq t_{\mathrm{f},1}, \\ (\Phi^{TQ}(g_{12}, x_2(t - t_{\mathrm{f},1})), u_2(t - t_{\mathrm{f},1})) & \text{otherwise,} \end{cases}$$

if there exists a group element g_{12} such that the second motion can be shifted compatibly, i.e. it holds $x_1(t_{\mathrm{f},1}) = \Phi^{TQ}(g_{12}, x_2(0))$. Furthermore, a trajectory π connecting two trims x_α and x_β by means of a motion along an (un)stable manifold orbit $O \in \mathcal{O}$ can be regarded as an *extended maneuver*.

Definition 4.2 (Extended Maneuvers) Let x_α, x_β $(\alpha \neq \beta)$ be trims and let $O \in \mathcal{O}$ be an (un)stable manifold orbit. Let π_1 be the maneuver of duration t_1 connecting x_α and O, and π_2 the maneuver connecting O and x_β. Define $\kappa : t \in [t_1, \tilde{t}] \mapsto (F_{\mathrm{L}}(x_1(t_1), t), 0)$, $\tilde{t} \geq t_1$ as that piece of the trajectory on O (with zero control) that starts at the final state of maneuver π_1 with duration $\tilde{t} - t_1$, s.t. $x_2(0) = F_{\mathrm{L}}(x_1(t), \tilde{t} - t_1)$. Then an extended maneuver π is defined as

$$\pi = \pi_1\kappa\pi_2.$$

Recall that the dynamical system is assumed to be autonomous, so time shifts are well defined. Controllability of the maneuver automaton is proved in Frazzoli et al. (2005) and still holds in the same sense for our motion planning approach applied to the trims and extended maneuvers.

4.3 Trajectory Generation

Consider the task of generating a trajectory from a given state $x_0 \in X$. It is typical to assume that this is either an equilibrium or periodic motion corresponding to a trim primitive. Denote the primitive by α_0 with initial state $x_{\alpha_0}(0)$. Then we have $x_0 = \Phi^{TQ}(g_0, x_{\alpha_0}(0))$ for some $g_0 \in G$. A trim primitive can be parametrized by its time duration, called the *coasting time* τ, leading to a family of trims, $\alpha(\tau) : t \in [0, \tau] \mapsto (\Phi^{TQ}(\exp(t\xi_\alpha, x_\alpha(0))), u_\alpha)$.

Consider a sequence of trim primitives $\alpha_0, \alpha_1, \ldots, \alpha_N$ with coasting times $\tau_0, \tau_1, \ldots, \tau_N$ and connecting maneuvers $\pi_0, \ldots, \pi_{N-1}$. These maneuvers can be either regular or extended. They form the trajectory ρ starting from x_0, defined by

$$\rho = \alpha_0(\tau_0)\pi_0\alpha_1(\tau_1)\pi_1 \cdots \alpha_N(\tau_{N-1})\pi_{N-1}\alpha_N(\tau_N). \tag{18}$$

The states along ρ are expressed, for $k \geq 0$, by

$$\rho(t) = \begin{cases} (\Phi^{TQ}(g_k \exp((t-t_k)\xi_{\alpha_k}), x_{\alpha_k}(0)), u_{\alpha_k}), & t \in [t_k, t_k + \tau_k], \\ (\Phi^{TQ}(g_k \exp(\tau_k \xi_{\alpha_k}), x_{\pi_k}(t')), u_{\pi_k}(t')), & t \in [t_k + \tau_k, t_{k+1}], \end{cases} \tag{19}$$

where $g_k = g_0 \prod_{i=0}^{k-1} \exp(\tau_i \xi_{\alpha_i}) g_{\pi_i}$, $t_k = \sum_{i=0}^{k-1} (\tau_i + |\pi_i|)$, with duration $|\pi_i|$ of maneuver π_i, and $t' = t - t_k - \tau_k$. The group elements $\exp(\tau_i \xi_{\alpha_i})$ are trim displacements, whereas g_{π_i} are the displacements of the maneuvers π_i. In addition, the total group displacement along ρ is

$$g_\rho = g_N \exp(\tau_N \xi_{\alpha_N}). \tag{20}$$

4.4 Computing Motion Plans

Next, consider the task of finding a sequence of primitives driving the system from its initial state x_0 to a given final state $x_\mathrm{f} \in \mathcal{X}_\mathrm{f}$. Let α_0 and α_f denote the given boundary trims with initial states $x_{\alpha_0}(0)$ and $x_{\alpha_\mathrm{f}}(0)$. Then we have $x_0 = \Phi^{TQ}(g_0, x_{\alpha_0}(0))$ and $x_\mathrm{f} = \Phi^{TQ}(g_\mathrm{f}, x_{\alpha_\mathrm{f}}(0))$ for some group elements $g_0, g_\mathrm{f} \in G$.

Computing a motion from x_0 to x_f amounts to finding a proper sequence of trim primitives $\alpha_0, \ldots, \alpha_N, \alpha_\mathrm{f}$, coasting times $\tau_0, \tau_1, \ldots, \tau_N, \tau_\mathrm{f}$, and connecting maneuvers $\pi_0, \ldots, \pi_N$ some of which include motions along (un)stable manifold orbits $O \in \mathcal{O}$. The sequence will form the trajectory ρ defined by

$$\rho = \alpha_0(\tau_0)\pi_0\alpha_1(\tau_1)\pi_1 \cdots \alpha_N(\tau_N)\pi_N\alpha_\mathrm{f}(\tau_\mathrm{f}). \tag{21}$$

The total group displacement along ρ is

$$g_\rho = \left[\prod_{i=0}^{N} \exp(\tau_i \xi_{\alpha_i}) g_{\pi_i} \right] \exp(\tau_\mathrm{f} \xi_{\alpha_\mathrm{f}}) \tag{22}$$

and computing a motion from x_0 to x_f amounts to finding a motion plan ρ such that

$$g_\rho = g_0^{-1} g_\mathrm{f}. \tag{23}$$

An optimal sequence of primitives and manifolds orbits should minimize the cost function $J(\rho)$. Although this is generally a complex combinatorial optimization with nonlinear constraints, it is much easier to solve than the original optimal control problem. The first reason is that the length of the sequence of required primitives is usually known in advance. For instance, in addition to the initial and final trim in general at least $\max(n-2, 0)$ additional (intermediate) trims are required to exactly satisfy any boundary conditions for an n-dimensional group G. In addition, in many cases the condition (23) can be solved analytically by computing the required trim coasting times using kinematic inversion.

Our implementation is based on a search tree (see Fig. 1) which expands all possible sequences of trims and manifold orbits and connecting maneuvers. The tree is grown in depth-first manner so that each trajectory contains $\max(n, 2)$ trim primitives. The search space is pruned using bounds on the optimal cost that is updated

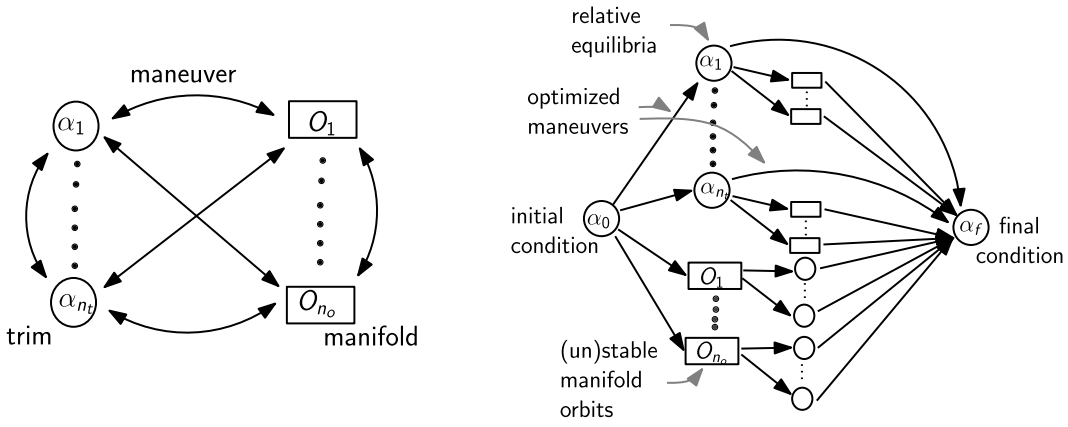


Fig. 1 *Left*: An automaton with states corresponding to relative equilibria and (un)stable manifolds and transitions corresponding to maneuvers. *Right*: An example of a search tree expanding paths of various sequences of primitives. This particular tree has three trim primitives and each trajectory must end at trim state α_f. The goal is to find a sequence and the coasting times along its relative equilibria so that the group displacement is satisfied and the total cost is minimized

Table 1 Combinations of primitives for $n = 1, 2, 3$

dim(G)	sequence	# of trims	# of maneuvers (depth)
1	$\alpha_0\pi_0\alpha_\mathrm{f}$	2	1
	$\alpha_0\pi_0 O_1\pi_1\alpha_\mathrm{f}$		2
2	$\alpha_0\pi_0\alpha_\mathrm{f}$	2	1
	$\alpha_0\pi_0 O_1\pi_1\alpha_\mathrm{f}$		2
3	$\alpha_0\pi_0\alpha_1\pi_1\alpha_\mathrm{f}$,	3	2
	$\alpha_0\pi_0 O_1\pi_1\alpha_1\pi_2\alpha_\mathrm{f}$		3
	$\alpha_0\pi_0\alpha_1\pi_2 O_1\pi_3\alpha_\mathrm{f}$		3
	$\alpha_0\pi_0 O_1\pi_1\alpha_1\pi_2 O_2\pi_3\alpha_\mathrm{f}$		4

during the algorithm operation. Thus, the number of primitives along trajectories in the tree varies from $2n - 1$ (when no manifolds are visited) to $4n - 3$ (when alternating between visiting trims and manifolds). For instance, for $n = 3$ the shortest sequence is $\alpha_0\pi_0\alpha_1\pi_1\alpha_\mathrm{f}$ while the longest is $\alpha_0\pi_0 O_1\pi_1\alpha_1\pi_2 O_2\pi_3\alpha_\mathrm{f}$. Table 1 lists the combinations of primitives up to $n = 3$.

Such sequences are automatically created by the tree-expansion algorithm for a given Lie group G. Note that we have assumed that all trims are non-equilibrium. Boundary conditions corresponding to equilibrium states (i.e. zero velocity) are handled by creating a sequence with an additional maneuver to or from a non-zero trim primitive.

By the construction of the maneuver automaton, the sequence ρ is a dynamically feasible solution for the optimal control problem from the initial state x_0 to the final state x_f. Thus, it can be used as a good initial guess for a post-optimization, e.g. performed by DMOC again. If the maneuver automaton is small, i.e. the gridding of trim and manifold state space is rough and the number of different connecting maneuvers

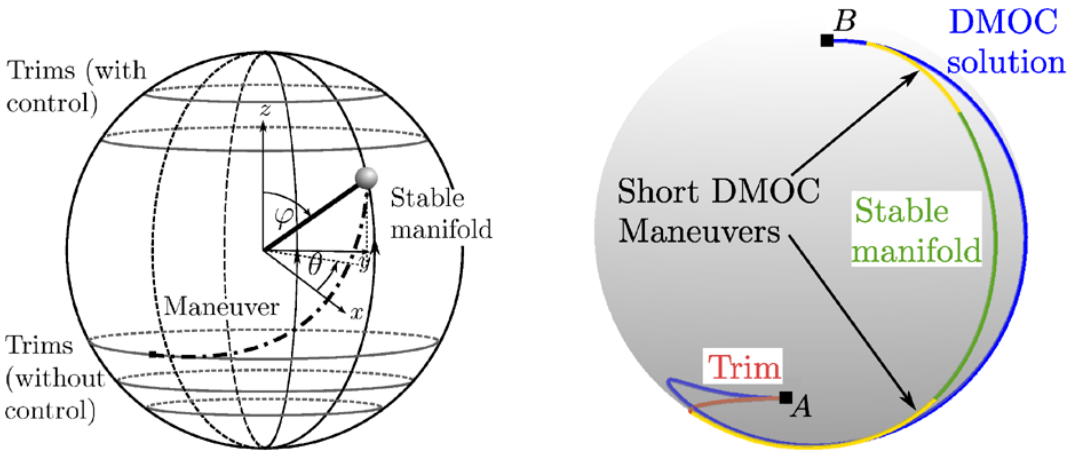


Fig. 2 *Left*: Model of the simple spherical pendulum with sketched motion primitives in configuration space: trims are purely horizontal motions ($\dot{\varphi} = 0$), whereas the (un)stable manifold are vertical motions ($\dot{\theta} = 0$). *Right*: Optimal sequence for a scenario from trim A to trim B consisting of the trims, two controlled maneuvers and a trajectory on the stable manifold in between. The *blue curve* is the solution of a DMOC optimization with the sequence as initial guess (Color figure online)

is small, a post-optimization is useful to smooth out the changes between controlled and uncontrolled pieces of the sequence trajectory. Furthermore, post-optimization is also required if the weighing of the (concurrent) objectives has to be updated or adjusted.

5 Numerical Examples

The simple spherical pendulum, i.e. a mass point moving in 3D constrained on a sphere, is a popular example to study symmetries of a mechanical system (see, among others Marsden 1993; Abraham and Marsden 1987; Bullo and Lewis 2004) and can be also used to demonstrate our motion planning approach (cf. Flaßkamp et al. 2010). From the application point of view, spherical pendula can be seen as idealizations of industrial robots, for example a double spherical pendulum is a simplified two-link manipulator. Therefore, optimal control of spherical pendula is of great importance.

5.1 The Spherical Pendulum

The pendulum consists of a point mass with mass m that is firmly connected by a massless rod of length r to the ground. Thus, the configuration space of this two degree of freedom system is a sphere. In coordinates, it can be described by a vertical angle, denoted by φ and a horizontal angle, denoted by θ (cf. Fig. 2).

Invariance and Symmetry The Lagrangian is given by $L(\varphi, \dot{\theta}, \dot{\varphi}) = K(\varphi, \dot{\theta}, \dot{\varphi}) - V(\varphi) = \frac{1}{2}mr^2(\dot{\varphi}^2 + \dot{\theta}^2 \sin^2(\varphi)) - mgr(\cos(\varphi) + 1)$. It can be easily seen that L is independent of θ, which is therefore called a *cyclic coordinate* (cf. e.g. Bloch 2003). Thus, it follows directly from $\frac{\partial L}{\partial \theta} = 0$ that the corresponding Euler–Lagrange equations simplify to $\frac{\partial L}{\partial \dot{\theta}} = \text{const}$. In other words, the system is symmetric w.r.t. rotations

about the vertical axis and the symmetry group is $G = S^1$, acting by addition only in the horizontal coordinate. Therefore, the conserved quantity equals $p_\theta = \frac{\partial L}{\partial \dot{\theta}}$, that is, the momentum map $J = p_\theta = mr^2 \sin^2(\varphi)\dot{\theta}$. The Hamiltonian as the system's energy is given by $E(q, \dot{q}) = K(q, \dot{q}) + V(q)$. The amended potential is then given by $V_\mu(q) = V(q) + \frac{1}{2}\langle \mu, \mathbb{I}^{-1}\mu \rangle = mgr(\cos(\varphi) + 1) + \frac{1}{2}\mu^2(mr^2 \sin^2(\varphi))^{-1}$. Relative equilibria can be computed as critical points of V_μ and fulfill $\dot{\theta}^2 = -\frac{g}{r \cdot \cos(\varphi)}$, i.e. they are purely horizontal rotations ($\dot{\varphi} = 0$) in the lower hemisphere. A discrete set Ξ of uncontrolled trims (cf. Sect. 4.1) for $\varphi \in \{\frac{2}{3}\pi, \frac{3}{4}\pi, \frac{5}{6}\pi\}$ for example, can be defined by the rotational velocity, i.e. the Lie group elements $\Xi = \{\pm\sqrt{2\frac{g}{r}}, \pm\sqrt{\sqrt{2}\frac{g}{r}}, \pm\sqrt{\frac{2\sqrt{3}}{3}\frac{g}{r}}\}$. If we add control in φ-direction, the rotational velocity and the height of a trim can be chosen arbitrarily with $u_\varphi = -mgr \sin(\varphi) - mr^2 \sin(\varphi)\cos(\varphi)\dot{\theta}^2$.

(Un)stable Manifolds of the Upper Equilibrium The *planar pendulum* exhibits a hyperbolic equilibrium in the upper fixed point. This gives rise to one-dimensional stable and unstable manifolds; together they form the separatrix in the well known phase portrait of a simple pendulum. For purely vertical initial conditions ($\dot{\theta} = 0$), the spherical pendulum behaves like a planar pendulum. This, together with the horizontal symmetry, explains why the stable and unstable manifold of the upper equilibrium of the spherical pendulum are given by $W^{\mathrm{u,s}}(\bar{x}) = \{(q, \dot{q}) \in TQ \mid J(q, \dot{q}) = 0, E(q, \dot{q}) = V(\bar{x}) = 2mgr\} = \{(\theta, \varphi, \dot{\theta}, \dot{\varphi}) | \theta = \text{const.}, \dot{\theta} = 0, \dot{\varphi}^2 = 2\frac{g}{r}(1 - \cos(\varphi))\}$, i.e. the manifolds of the planar pendulum with an arbitrary, but fixed horizontal angle.

Motion Planning Trim primitives are uniform rotations in horizontal planes, whereas trajectories on the (un)stable manifolds are purely vertical motions. Choosing a discretization in both angles (plus a discretization of the rotational velocity for trim primitives in case of non-zero control) gives the motion primitives for the library (see Fig. 2 (left) for a sketch of the motion primitives). For numerical computations, all parameters are normalized to one. The connecting maneuvers are computed by DMOC. Here we allow forcing in both coordinate directions and search for solutions that minimize $J(x, u) = \int_0^{t_\mathrm{f}} (u_\theta(t)^2 + u_\varphi(t)^2)\,\mathrm{d}t$. As an exemplary scenario we choose a starting point A and a final point B on trims ($\varphi_A = \frac{13}{16}\pi$ uncontrolled, $\varphi_B = \frac{1}{8}\pi$ controlled s.t. $\dot{\theta} = -\pi$) and search for sequences with minimal control effort that connect these trims with a trajectory on the stable manifold to the upper equilibrium by maneuvers. The resulting trajectory (cf. Fig. 2 (right)) has the costs $J = 3.2211$ and the final time $t_\mathrm{f} = 4.3335$, which is the sum of the time spent on the trims, the fixed durations of the maneuvers and the time that the sequence stays on the manifold orbit. The sequence is then used as an initial guess for a post-optimization by DMOC that reduces the costs of the sequence to $J = 1.3821$. This is compared to optimal solutions of naive, direct optimizations with simple linearly interpolated initial guesses, i.e. we interpolate each coordinate between its initial and final point on an equidistant time grid. Such an initial guess can be constructed without any knowledge of the dynamical system, however, the resulting curve is by no means an admissible solution. It turns out that the information about the duration of the optimal trajectory that we obtain from the sequencing approach is important for finding energy efficient maneuvers: direct solutions for $t_\mathrm{f} = 2$ or $t_\mathrm{f} = 12$ have much higher costs

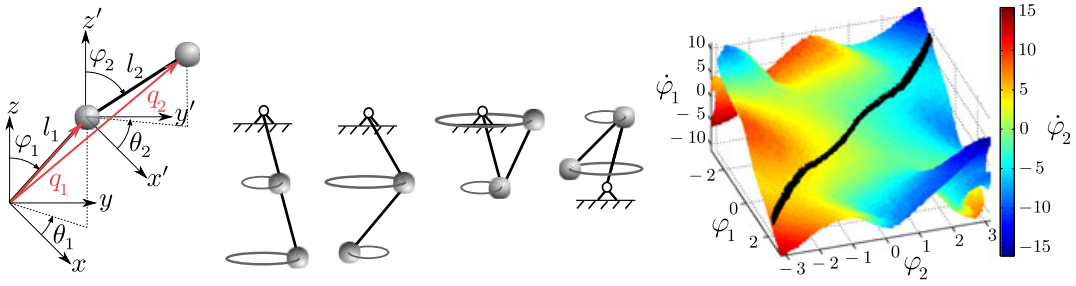


Fig. 3 *Left*: Model of the double spherical pendulum. *Middle*: Shape of the relative equilibria without control. Adding constant control in vertical direction also allows for trim primitives with both pendula pointing upwards and an arbitrary jointly rotational velocity (not shown). *Right*: Approximation of the stable manifold of the upper equilibrium and the strong stable manifold in black, computed by GAIO (Color figure online)

of $J = 6.3427$ and $J = 2.6084$. For the time t_f defined by the sequence, the direct solutions are similar in cost and qualitative behavior compared to the sequence. In more complicated systems, such as a double spherical pendulum, it is much harder to find any reasonable, admissible solution without choosing a sophisticated initial guess.

5.2 The Double Spherical Pendulum

In case of a double spherical pendulum, a mixture of analytical and computational methods have to be applied to compute the motion primitives. In the following, we will present candidates for a motion planning library and afterwards, show numerical results for specific optimal control scenarios.

Euler–Lagrange Equations The configuration space of two 3D pendula, idealized as mass points m_1 and m_2 on massless rods, is $Q = S^2_{l_1} \times S^2_{l_2}$, where $S^2_{l_{1,2}}$ denotes the two dimensional sphere of radius $l_{1,2}$. As a minimal set of coordinates, we choose horizontal and vertical angles ($q = (\theta_1, \theta_2, \varphi_1, \varphi_2)$), such that the mass points positions are given by (cf. Fig. 3)

$$q_1 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} l_1 \cos(\theta_1) \sin(\varphi_1) \\ l_1 \sin(\theta_1) \sin(\varphi_1) \\ l_1 \cos(\varphi_1) \end{pmatrix},$$

$$q_2 = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + \begin{pmatrix} l_2 \cos(\theta_2) \sin(\varphi_2) \\ l_2 \sin(\theta_2) \sin(\varphi_2) \\ l_2 \cos(\varphi_2) \end{pmatrix}.$$

The Lagrangian as the difference of kinetic and potential energy can be written as $L(q(t), \dot{q}(t)) = K(q(t), \dot{q}(t)) - V(q(t))$, where $V(q(t)) = (m_1 + m_2) g l_1 (\cos(\varphi_1) + 1) + m_2 g l_2 (\cos(\varphi_2) + 1)$, and $K(q(t), \dot{q}(t)) = \frac{1}{2} \dot{q}^T(t) M(q(t)) \dot{q}(t)$ with the symmetric mass matrix $M = (m_{ij})$ with $m_{11} = (m_1 + m_2) l_1^2 \cdot \sin^2(\varphi_1)$, $m_{12} = m_2 l_1 l_2 \cdot \cos(\theta_1 - \theta_2) \cdot \sin(\varphi_1) \sin(\varphi_2)$, $m_{13} = 0$, $m_{14} = -m_2 l_1 l_2 \sin(\theta_1 - \theta_2) \cdot \sin(\varphi_1) \cos(\varphi_2)$, $m_{22} = m_2 l_2^2 \sin^2(\varphi_2)$, $m_{23} = m_2 l_1 l_2 \cdot \sin(\theta_1 - \theta_2) \cdot \cos(\varphi_1) \sin(\varphi_2)$, $m_{24} = 0$, $m_{33} = (m_1 + m_2) l_1^2$, $m_{34} = m_2 l_1 l_2 ((\cos(\theta_1 - \theta_2) \cdot \cos(\varphi_1) \cos(\varphi_2)) \sin(\varphi_1) \sin(\varphi_2))$, $m_{44} = m_2 l_2^2$.

Hence, the Euler–Lagrange equations for the double spherical pendulum without forcing are

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\varphi}_{1,2}} - \frac{\partial L}{\partial \varphi_{1,2}} = 0, \qquad \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\theta}_{1,2}} - \frac{\partial L}{\partial \theta_{1,2}} = 0. \tag{24}$$

Symmetry and Reduction The symmetry group is $G = S_1$, acting by rotation of both pendula about the z-axis: $\Phi : G \times Q \to Q$, $\Phi(g, (\theta_1, \theta_2, \varphi_1, \varphi_2)) = (g + \theta_1, g + \theta_2, \varphi_1, \varphi_2)$ with tangent lift to TQ by $\Phi_g^{TQ}(q, v) = (\Phi(g, q), \dot{\theta}_1, \dot{\theta}_2, \dot{\varphi}_1, \dot{\varphi}_2)$. Then the infinitesimal generator can be determined to be $\xi_Q(q) = (\xi, \xi, 0, 0)^{\mathrm{T}}$ with $\xi \in \mathbb{R}$. Hence, the conserved quantity is the total angular momentum about the z-axis

$$J(q, v) = \frac{\partial L}{\partial \dot{\theta}_1} + \frac{\partial L}{\partial \dot{\theta}_2}, \tag{25}$$

and the locked inertia tensor (cf. Sect. 3.2) equals

$$\begin{aligned}\mathbb{I}\big(q(t)\big) = (m_1 + m_2) l_1^2 \sin^2(\varphi_1) + m_2 l_2^2 \sin^2(\varphi_2) \\ + 2 m_2 l_1 l_2 \cos(\theta_1 - \theta_2) \cdot \sin(\varphi_1) \sin(\varphi_2).\end{aligned}$$

The mechanical connection $\alpha : TQ \to \mathfrak{g}$ can be easily computed by $\alpha(q, v) = \mathbb{I}^{-1}(q) \cdot J(q, v)$, assigning to each (q, v) the angular velocity of the locked system (cf. Marsden and Scheurle 1993). The amended potential can be computed by

$$V_\mu\big(q(t)\big) = V\big(q(t)\big) + \frac{\mu^2}{2\mathbb{I}(q)}.$$

Trims Trims of the uncontrolled system, i.e. relative equilibria, are classified in (Marsden and Scheurle 1993) in an elegant way by introducing two shape defining parameters and then computing the critical points of the amended potential (cf. the Lagrangian reduction in Sect. 3.2). Besides the four true equilibria (each pendulum either pointing straight upwards or downwards), all relative equilibria are given by a one-parameter curve and they look similar to one of the four sketched types in Fig. 3 (middle). According to Definition 3.6, non-zero constant control values are allowed, if they do not influence the conservation of the angular momentum J (cf. (25)). Hence, we add forcing in φ_1- and φ_2-direction in (24). This leads to a controlled potential and we can therefore solve $\frac{\partial}{\partial q} V_\mu = -u$ for constant $u = (0, 0, u_{\varphi_1}, u_{\varphi_2})^{\mathrm{T}}$ as proposed by Theorem 3.7. This additionally admits trims with both pendula pointing upwards as well as arbitrary rotating velocities in all shapes.

Manifolds For this example, we are interested in the (un)stable manifold of the upper equilibrium, i.e. both pendula pointing upwards ($\bar{x} = (\bar{q}, \dot{\bar{q}}) = 0_{8\times 1}$). In this point, the system's energy equals $E_{\bar{x}} := V(\bar{q})$ while the angular momentum is zero. Hence the manifolds are part of the set $\{x \in TQ \mid E(x) = E_{\bar{x}}, J(x) = 0\}$. This includes in particular the motion on (un)stable manifolds of a planar double pendulum, to which we have restricted our computations so far. Since the manifolds are two dimensional, we still have to choose concrete trajectories that are stored in the motion planning

library. Here we use the strong (un)stable manifolds (cf. Sect. 4.1). In Fig. 3 (right) the black line corresponds to the approximation of the strong stable manifold in the stable manifold of the upper equilibrium, which has been computed with GAIO (cf. Sect. 3.3).

Numerical Results for Motion Planning Scenarios As mentioned before, we use the optimal control method DMOC (cf. Sect. 2.3) to compute connecting maneuvers between trims and orbits on manifolds. For numerical computations, we choose the following parameter values: $m_1 = m_2 = 1$ kg, $l_1 = l_2 = 1$ m, and $g = 9.81 \frac{\mathrm{m}}{\mathrm{s}^2}$. The nonlinear optimization problem is solved by an SQP method (cf. Sect. 2.1) of NAG[1] (*Numerical Algorithms Group*). To improve the accuracy of the derivatives that have to be provided, the implementation of the DMOC method has been combined with ADOL-C[2] (*Automatic Differentiation by OverLoading in C++*), an automatic differentiation technique, in Ober-Blöbaum and Walther (2010). For the connecting maneuvers, we allow an arbitrary boundary point on the specific trims (cf. (14)–(17)), i.e. the point is fixed except for the horizontal coordinates, which have to fulfill $\theta_1 = \theta_2$ for an arbitrary angle θ_1. Since the double spherical pendulum is modeled in minimal coordinates that are not globally valid, we are faced with singularities in our numerical computations. If one of the pendulum's vertical angle equals 0 or π (or multiplicities of that), the horizontal angle becomes meaningless. The NAG algorithm is able to perform the optimization for our scenarios. Nevertheless, to overcome this problem in principle, a global system description by e.g. differential algebraic models could be used in future work.

The motion planning is performed for the following scenario: the starting point is chosen to lie on an uncontrolled trim ($\varphi_1 = 2.4087$, $\varphi_2 = 2.2532$), where the double pendulum is outstretched. The final point is the upper equilibrium, i.e. both pendula pointing upwards. We consider the fully actuated system ($u = (u_{\theta_1}, u_{\theta_2}, u_{\varphi_1}, u_{\varphi_2})^{\mathrm{T}}$) and choose the control effort as the cost functional, i.e.

$$J\big(x(t), u(t)\big) = \int_0^{t_\mathrm{f}} u(t)^2 \,\mathrm{d}t \quad \text{with } u(t) \in \mathbb{R}^4.$$

According to the defined scenario, a sequence of depth 2 (cf. the definition of a search tree in Sect. 4.4) is searched for, consisting of a maneuver from the trim to the orbit of the strong stable manifold of the upper equilibrium and then a second, very short maneuver to bridge the gap from the orbit's endpoint to the equilibrium itself. Figure 4 shows a resulting sequence with duration $t_\mathrm{f} = 3.28$ and costs $J = 548.76$. The durations of the maneuvers have been fixed in advance, such that the entire duration depends on how long the sequence stays on the manifold orbit. The dashed lines refer to the results of a post-optimization performed by DMOC, which reduces the costs to $J = 296.51$. In comparison, when DMOC is directly applied to the problem with a simple, linearly interpolated initial guess, the retained optimal solution has much higher costs of $J = 5.85 \cdot 10^3$.

[1] www.nag.co.uk.

[2] https://projects.coin-or.org/ADOL-C.

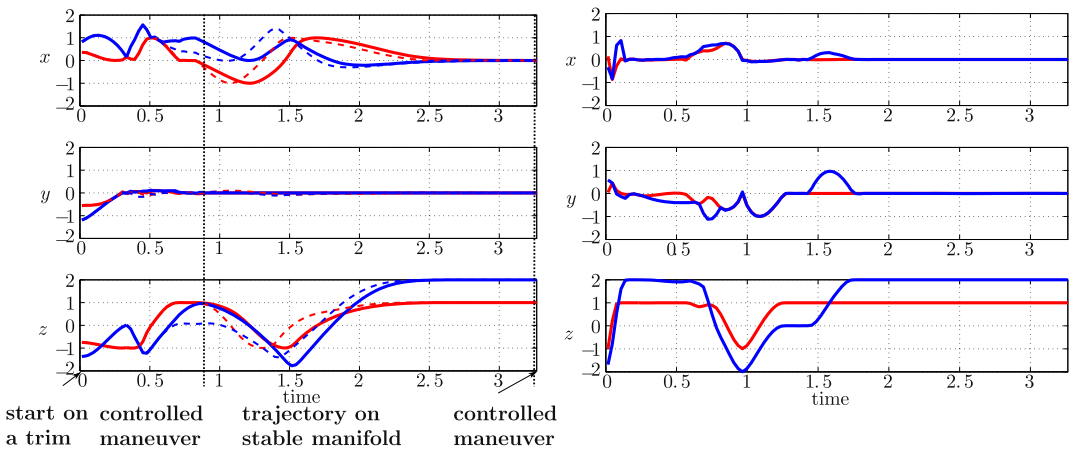


Fig. 4 *Left*: Sequence for an example scenario, presented in Cartesian coordinates of both pendula (*solid lines*; inner pendulum *red*, outer pendulum *blue*) and resulting optimal trajectory for a post-optimization with DMOC (*dashed lines*). *Right*: A maneuver resulting from an optimization by DMOC with simple initial guess (Color figure online)

In this scenario, we considered sequences involving only one manifold and therefore restricted to the stable manifold of the upper equilibrium. However, it might be possible that a sequence of higher depth including other manifolds as well would even lead to further improvement. This has to be studied in future work.

6 Conclusion and Outlook

This work proposes a motion planning strategy based on motion primitives encoding inherent dynamical system properties. We extend the approach of Frazzoli et al. (2005) by including motion primitives on (un)stable manifolds of critical elements of the uncontrolled dynamics. Such primitives are useful for finding energy efficient solutions, experimentally confirmed by the numerical results for our example optimal control scenarios. We study the motion primitives induced by symmetries in more detail, focusing on mechanical systems. Trim primitives for arbitrary mechanical systems are identified using Noether's theorem on conserved angular momenta through a symmetry reduction process. In addition to trim primitives and orbits on (un)stable manifolds, connecting maneuvers are computed by the optimal control method DMOC and stored in the motion planning library. The maneuver automaton of Frazzoli et al. (2005) is extended to include orbits on manifolds and finally we develop a tree search algorithm in this new automaton for motion planning. The application of the approach to the optimal control of a double spherical pendulum clearly shows that optimization using initial guesses, obtained by exploiting the key structural properties of the system, results improved solutions (w.r.t. to cost) compared to standard initialization.

Future work will apply this optimal control policy to more complex systems such as multi-body systems with holonomic (Leyendecker et al. 2009) or nonholonomic constraints (Kobilarov et al. 2010). For example, the structural properties of a rigid body pendulum that are revealed in Chaturvedi et al. (2011) could be used for optimal

control scenarios as well. In higher dimensional systems it will be more challenging to identify symmetries, i.e. Lie groups and admissible controls for trim primitives. If the computation of (un)stable manifolds gets numerically too expensive, invariant objects in a reduced system (e.g. obtained by Lagrangian reduction) could be considered. Alternatively, one could restrict oneself to the one-dimensional strong (un)stable manifolds.

Studying hybrid Lagrangian systems is another natural extension. A control sequence resulting from a motion planning procedure is already hybrid in the sense that different types of control trajectory are concatenated. Hybrid dynamics can also occur if different Lagrangian are valid in different regions of state space or because of impacts, i.e. instantaneous jumps in the states. Symmetry and reduction of hybrid Lagrangian systems has been already studied in e.g. Ames and Sastry (2006). Thus, it is desirable and conceptually possible to extend the motion planning approach to symmetric hybrid systems. If the motion planning library is designed to include many admissible sequences, our tree search can be augmented with a more efficient sampling strategy. A possible approach is to employ adaptive sampling used in the context of randomized motion planning (Kobilarov 2011).

Thinking further ahead, optimal control problems of complex dynamical systems will typically include not only one or two, but several, competing objectives. Additionally, a real world motion planning problem will include several boundary constraints, e.g. given by different operation points that have to be reached one after the other. These subproblems as well as the prioritized objective may change during an operation. For scenarios like this, a motion planning library that has been generated offline in advance seems appropriate for an online optimization.

Acknowledgements Jerry Marsden has been a great inspiration to us to work on this topic. We thank him for fruitful discussions and collaborations during the last years. This contribution was partly developed and published in the course of the Collaborative Research Centre 614 "Self-Optimizing Concepts and Structures in Mechanical Engineering" funded by the German Research Foundation (DFG) under grant number SFB 614.

M. Kobilarov was supported by the Keck Institute for Space Studies, Caltech.

References

Abraham, R., Marsden, J.E.: Foundations of Mechanics. Addison-Wesley, Redwood City (1987)

Ames, A.D., Sastry, S.: Hybrid Routhian reduction of Lagrangian hybrid systems. In: American Control Conference, 2006, June 2006. 6 pp.

Betts, J.T.: Survey of numerical methods for trajectory optimization. AIAA J. Guid. Control Dyn. **21**(2), 193–207 (1998)

Binder, T., Blank, L., Bock, H.G., Bulirsch, R., Dahmen, W., Diehl, M., Kronseder, T., Marquardt, W., Schlöder, J.P., von Stryk, O.: Introduction to model based optimization of chemical processes on moving horizons. In: Grötschel, M., Krumke, S.O., Rambau, J. (eds.) Online Optimization of Large Scale Systems: State of the Art, pp. 295–340. Springer, Berlin (2001)

Bloch, A.M.: Nonholonomic Mechanics and Control. Springer, Berlin (2003)

Bloch, A.M., Leonard, N.E., Marsden, J.E.: Controlled Lagrangians and the stabilization of mechanical systems. I. The first matching theorem. IEEE Trans. Autom. Control **45**(12), 2253–2270 (2000)

Bullo, F., Lewis, A.D.: Geometric Control of Mechanical Systems. Texts in Applied Mathematics, vol. 49. Springer, New York (2004)

Bullo, F., Lewis, A.: Reduction, linearization, and stability of relative equilibria for mechanical systems on Riemannian manifolds. Acta Appl. Math. **99**, 53–95 (2007)

Chaturvedi, N., Lee, T., Leok, M., McClamroch, N.: Nonlinear dynamics of the 3D pendulum. J. Nonlinear Sci. **21**, 3–32 (2011)
Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge (2005)
Christensen, G.S., El-Hawary, E., Soliman, S.A.: Optimal Control Applications in Electric Power Systems. Mathematical Concepts and Methods in Science and Engineering, vol. 35. Plenum, New York (1987)
Conley, C.: Low energy transit orbits in the restricted three-body problem. SIAM J. Appl. Math. **16**, 732–746 (1968)
Dellnitz, M., Froyland, G., Junge, O.: The algorithms behind GAIO—set oriented numerical methods for dynamical systems. In: Fiedler, B. (ed.) Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems, pp. 145–174. Springer, Berlin (2001)
Dellnitz, M., Junge, O., Post, M., Thiere, B.: On target for Venus—set oriented computation of energy efficient low thrust trajectories. Celest. Mech. Dyn. Astron. **95**, 357–370 (2006)
Dellnitz, M., Ober-Blöbaum, S., Post, M., Schütze, O., Thiere, B.: A multi-objective approach to the design of low thrust space trajectories using optimal control. Celest. Mech. Dyn. Astron. **105**, 33–59 (2009). doi:10.1007/s10569-009-9229-y
Ehrgott, M.: Multicriteria Optimization, 2nd edn. Springer, Berlin (2005)
Flaßkamp, K., Ober-Blöbaum, S.: Energy efficient control for mechanical systems based on inherent dynamical structures. In: American Control Conference (ACC), June 2012, Montréal, Canada, pp. 2609–2614 (2012)
Flaßkamp, K., Ober-Blöbaum, S., Kobilarov, M.: Solving optimal control problems by using inherent dynamical properties. PAMM **10**(1), 577–578 (2010)
Frazzoli, E., Dahleh, M.A., Feron, E.: Maneuver-based motion planning for nonlinear systems with symmetries. IEEE Trans. Robot. **21**(6), 1077–1091 (2005)
Froyland, G., Padberg, K.: Almost-invariant sets and invariant manifolds—connecting probabilistic and geometric descriptions of coherent structures in flows. Physica D **238**(16), 1507–1523 (2009)
Gerdts, M.: Solving mixed-integer optimal control problems by branch & bound: a case study from automobile test-driving with gear shift. Optim. Control Appl. Methods **26**(1), 1–18 (2005)
Gill, P.E., Jay, L.O., Leonard, M.W., Petzold, L.R., Sharma, V.: An SQP method for the optimal control of large-scale dynamical systems. J. Comput. Appl. Math. **120**, 197–213 (2000)
Gómez, G., Koon, W.S., Lo, M.W., Marsden, J.E., Masdemont, J., Ross, S.D.: Connecting orbits and invariant manifolds in the spatial three-body problem. Nonlinearity **17**, 1571–1606 (2004)
Griewank, A., Walther, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, 2nd edn. SIAM, Philadelphia (2008)
Guckenheimer, J., Holmes, P.: Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields. Applied Mathematical Sciences, vol. 42. Springer, Berlin (1983)
Haller, G.: Distinguished material surfaces and coherent structures in 3d fluid flows. Physica D **149**, 248–277 (2001)
Haller, G., Yuan, G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. Physica D **147**, 352–370 (2000)
Katok, A., Hasselblatt, B.: Introduction to the Modern Theory of Dynamical Systems. Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge (1995)
Kobilarov, M.: Discrete geometric motion control of autonomous vehicles. PhD thesis, University of Southern California, USA (2008)
Kobilarov, M.: Cross-entropy randomized motion planning. In: Proceedings of Robotics: Science and Systems, Los Angeles, CA, USA, June 2011
Kobilarov, M., Marsden, J.E.: Discrete geometric optimal control on Lie groups. IEEE Trans. Robot. **27**(4), 641–655 (2011)
Kobilarov, M., Marsden, J.E., Sukhatme, G.S.: Geometric discretization of nonholonomic systems with symmetries. Discrete Contin. Dyn. Syst., Ser. S **3**(1), 61–84 (2010)
Koon, W.S., Lo, M.W., Marsden, J.E., Ross, S.D.: Shoot the Moon. Spacefl. Mech. **105**(2), 1017–1030 (2000)
Koon, W.S., Lo, M.W., Marsden, J.E., Ross, S.D.: Low energy transfer to the Moon. Celest. Mech. Dyn. Astron. **81**(1–2), 63–73 (2001)
Krauskopf, B., Osinga, H.M., Doedel, E.J., Henderson, M.E., Guckenheimer, J., Vladimirsky, A., Dellnitz, M., Junge, O.: A survey of methods for computing (un)stable manifolds of vector fields. Int. J. Bifurc. Chaos Appl. Sci. Eng. **15**(3), 763–791 (2005)
LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)

Leyendecker, S., Ober-Blöbaum, S., Marsden, J.E., Ortiz, M.: Discrete mechanics and optimal control for constrained systems. In: Optimal Control, Applications and Methods, 2009

Marsden, J.E.: Lectures on Mechanics. London Mathematical Society Lecture Note Series, vol. 174. Cambridge University Press, Cambridge (1993)

Marsden, J.E.: Geometric Mechanics, Stability, and Control, pp. 265–291. Springer, New York (1994)

Marsden, J.E., Ratiu, T.S.: Introduction to Mechanics and Symmetry, 2nd edn. Text in Applied Mathematics, vol. 17. Springer, Berlin (1999)

Marsden, J.E., Scheurle, J.: Lagrangian reduction and the double spherical pendulum. Z. Angew. Math. Phys. **44** (1993)

Marsden, J.E., West, M.: Discrete mechanics and variational integrators. Acta Numer. **10**, 357–514 (2001)

Marsden, J.E., Patrick, G.W., Shkoller, S.: Multisymplectic geometry, variational integrators, and nonlinear PDEs. Commun. Math. Phys. **199**, 351–395 (1998)

Marsden, J.E., Ratiu, T.S., Scheurle, J.: Reduction theory and the Lagrange–Routh equations. J. Math. Phys. **41**, 3379–3429 (2000)

McGehee, R.: Some homoclinic orbits for the restricted three-body problem. PhD thesis, University of Wisconsin (1969)

Naldi, R., Marconi, L.: Optimal transition maneuvers for a class of V/STOL aircraft. Automatica **47**(5), 870–879 (2011)

Neumaier, A.: Complete search in continuous global optimization and constraint satisfaction. Acta Numer. **13**, 271–369 (2004)

Ober-Blöbaum, S., Walther, A.: Computation of derivatives for structure preserving optimal control using automatic differentiation. PAMM **10**(1), 585–586 (2010)

Ober-Blöbaum, S., Junge, O., Marsden, J.E.: Discrete mechanics and optimal control: an analysis. Control Optim. Calc. Var. **17**(2), 322–352 (2011)

Osinga, H.M., Rokni Lamooki, G.R., Townley, S.: Numerical approximations of strong (un)stable manifolds. Dyn. Syst. **19**(3), 195–215 (2004)

Roberts, M., Wulff, C., Lamb, J.S.W.: Hamiltonian systems near relative equilibria. J. Differ. Equ. **179**(2), 562–604 (2002)

Serban, R., Koon, W.S., Lo, M.W., Marsden, J.E., Petzold, L.R., Ross, S.D., Wilson, R.S.: Halo orbit mission correction maneuvers using optimal control. Automatica **38**, 571–583 (2002)

Simo, J.C., Lewis, D., Marsden, J.E.: Stability of relative equilibria. Part I: the reduced energy-momentum method. Arch. Ration. Mech. Anal. **115**, 15–59 (1991)

Sussmann, H.J., Willems, J.C.: 300 years of optimal control: from the Brachystochrone to the maximum principle. IEEE Control Syst. **17**(3), 32–44 (1997)

Wulff, C., Schilder, F.: Numerical bifurcation of Hamiltonian relative periodic orbits. SIAM J. Appl. Dyn. Syst. **8**(3), 931–966 (2009)

Zhigljavsky, A., Zilinskas, A.: Stochastic Global Optimization. Springer Optimization and Its Applications. Springer, Berlin (2008)

The International Journal of
Robotics Research
31(1) 24–41

DOI: 10.1177/0278364911423558
ijr.sagepub.com

Global estimation in constrained environments

Marin Kobilarov[1], Jerrold E Marsden[1] and Gaurav S Sukhatme[2]

Abstract

This article considers the optimal estimation of the state of a dynamic observable using a mobile sensor. The main goal is to compute a sensor trajectory that minimizes the estimation error over a given time horizon taking into account uncertainties in the observable dynamics and sensing, and respecting the constraints of the workspace. The main contribution is a methodology for handling arbitrary dynamics, noise models, and environment constraints in a global optimization framework. It is based on sequential Monte Carlo methods and sampling-based motion planning. Three variance reduction techniques–utility sampling, shuffling, and pruning–based on importance sampling, are proposed to speed up convergence. The developed framework is applied to two typical scenarios: a simple vehicle operating in a planar polygonal obstacle environment and a simulated helicopter searching for a moving target in a 3-D terrain.

Keywords

Aerial robotics, motion planning, estimation, search and rescue robots

1. Introduction

Consider a mobile sensor (a *vehicle*) estimating the state of an observable (a *target*) with uncertain dynamics through noisy measurements. The vehicle and target motions are constrained due to their natural kinematics and dynamics and due to obstacles in the environment. The task is to compute an open-loop vehicle trajectory over a given time horizon resulting in a target state estimate with lowest uncertainty. This sort of capability is required, for instance, for time-critical *surveillance* or *search-and-rescue* missions.

The problem is formally defined through a hidden Markov model (HMM) of a stochastic process $\{(X_k, Y_k)\}_{0\leq k\leq N}$ where X_k denotes the hidden target state and Y_k denotes the observation at the kth time epoch. The process evolution is studied over a horizon of N epochs. The respective state and observation realizations are denoted by $x_k \in \mathcal{X} \subset \mathbb{R}^{n_x}$ and $y_k \in \mathcal{Y} \subset \mathbb{R}^{n_y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are vector spaces. The HMM is defined by

$$X_{k+1} = f(X_k, \Omega_k), \tag{1a}$$

$$Y_k = g(X_k, V_k; \mu_k), \tag{1b}$$

where Ω_k and V_k are independent and identically distributed (i.i.d.) noise terms and $\mu_k \in \mathcal{M}$ denotes the vehicle state[1] at time k. The manifold $\mathcal{M}$ need not be a vector space. A trajectory of states between two epochs i and j, where $0 \leq i < j \leq N$, is denoted by $x_{i:j} := \{x_i, x_{i+1}, \ldots, x_{j-1}, x_j\}$.

The vehicle trajectory $\mu_{0:N}$ is subject to dynamical constraints, for example, arising from discretized Euler–Lagrange equations of motion, expressed through the equality $h_d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^{n_d}$

$$h_d(\mu_k, \mu_{k+1}) = 0, \text{ for all } 0 \leq k < N. \tag{2}$$

In addition the vehicle must avoid obstacles and is subject to velocity and actuator bounds, jointly encoded through the inequality constraints $h_c : \mathcal{M} \to \mathbb{R}^{n_c}$

$$h_c(\mu_k) \geq 0, \text{ for all } 0 \leq k \leq N. \tag{3}$$

A trajectory that satisfies the constraints (2) and (3) is termed *feasible*.

The functions h_d and h_c are typically *non-convex* and in some cases *non-smooth*. On the one hand, complicated non-linear dynamics and obstacles induce multiple homotopy classes of vehicle trajectories that preclude convexity. On the other, function (2) can have singularities (and, hence, might not be smooth everywhere) when the dynamics is underactuated or non-holonomic (Choset et al. 2005; LaValle 2006). In addition, the distance-to-obstacle

[1]California Institute of Technology, Pasadena, CA, USA
[2]University of Southern California, Los Angeles, CA, USA

Corresponding author:
Marin Kobilarov, 2543 Wellesley Avenue, Los Angeles, California 90064, US.
Email: marin@cds.caltech.edu

function [encoded in (3)] might not be differentiable, for example, at sharp obstacle corners, requiring either ad hoc smoothing or special non-smooth techniques (Clarke et al. 1998) such as *generalized gradients* (Choset et al. 2005).

The vehicle numerically computes the target state distribution, also referred to as the *filtering distribution*, denoted by $\pi_k(dx|y_{1:k};\mu_{0:k})\,dx := \mathbb{P}(X_k \in dx|y_{1:k};\mu_{0:k})$ with respect to some standard measure dx assuming the vehicle has moved along trajectory $\mu_{0:k}$ and obtained a sequence of measurements $y_{1:k}$.

1.1. Objective

The goal is to control the vehicle to obtain a high-quality estimate of the target state during the *future* N epochs. Typically, only a subset of the target coordinates are of interest. An appropriately chosen function $\varphi : \mathcal{X} \to \mathbb{R}^{n'_x}$, where $n'_x \leq n_x$, selects and weighs a combination of these coordinates. For instance, φ can pick out only the position of a moving target and ignore its velocity and heading. The optimization problem is to compute the optimal future vehicle trajectory $\mu^*_{1:N}$, which minimizes the target estimate *uncertainty* defined by

$$\mu^*_{1:N} = \arg\min_{\mu_{1:N}} \mathbb{E}\left[\left\|\varphi(X_N) - \int \varphi(x)\,\pi_N(x|Y_{1:N};\mu_{1:N})\,dx\right\|^2\right], \tag{4}$$

subject to the dynamics (2) and constraints (3). The expectation in (4) is taken over all future realizations of the states $X_{0:N}$ and the measurements $Y_{1:N}$ while π_N is the posterior density after filtering these measurements given (1).

The cost function in (4) is equivalent to the *trace of the covariance* of $\varphi(X_N)$. While it is possible to use other measures, such as entropy or the covariance determinant, this metric is chosen since its value can be interpreted in meaningful units (e.g. see Mihaylova et al. 2003a). For instance, the special case $\varphi(x) = M^{\frac{1}{2}}x$ for some weighting matrix M corresponds to a well-established tolerance-weighted error or L-optimal design (de Geeter et al. 1998).

1.2. Simple example

These definitions can be illustrated with a simple example of a target modeled as a unit-mass particle moving in a plane. A vehicle with fixed constant velocity $v \in \mathbb{R}^2$ takes relative position measurements and must avoid a circular obstacle with center $o \in \mathbb{R}^2$ and radius d. In this situation we have $\mathcal{X} = \mathcal{Y} = \mathcal{M} = \mathbb{R}^2$ with motion function $f(x,\omega) = x + \omega$, observation function $g(x,\nu;\mu) = x - \mu + \nu$, and vehicle dynamics and constraints given by $h_d(\mu_k,\mu_{k+1}) = \mu_{k+1} - \mu_k - v$ and $h_c(\mu) = \|\mu - o\| - d$, respectively. The noise terms ω and ν are realizations of, for example, white random processes. Setting $\varphi(x) = x$ is then equivalent to minimizing the sum of the variances of the two planar coordinates. Therefore, if x were measured in meters then the square root of the right-hand side of (4) is also in meters, which is convenient for establishing meaningful tolerances.

1.3. Related work

The optimization (4) corresponds to the *optimal sensor scheduling* problem (Tremois and Le Cadre 1999; Singh et al. 2007), which is of central importance for the target-tracking community. It is also highly relevant to the problem of *active sensing* studied in robotics (Grocholsky et al. 2003; Mihaylova et al. 2003a; Thrun et al. 2005) where the vehicle estimates its own state (Paris and Le Cadre 2002; He et al. 2008) and in some cases refines its knowledge about the environment (Sim and Roy 2005; Stachniss et al. 2005).

One approach is to solve the problem approximately by discretizing the vehicle and target state spaces. Such techniques, for example, based on regular grids (e.g. Chung and Burdick 2007) or shaping functions (Lavis et al. 2008), are too restrictive when non-trivial dynamics and sensing are considered. They are more appropriate for higher-level decision making. For instance, a policy search in an information space (LaValle 2006) or a Markov decision process (MDP)-based search [e.g. Bethke et al. (2008)] would typically be based on such representations. We emphasize that our main interest is in high-dimensional problems dominated by fast non-linear and underactuated dynamics and sensing. In this context techniques exploiting convexity [respectively submodularity (Krause and Guestrin 2007; Hollinger et al. 2009)] are not suitable since such approximations are either too coarse or will violate the dynamics and result in solutions that the original system cannot realistically execute.

Most existing techniques, beyond discrete methods, have one or more of the following limitations: they are based on models with linear or Gaussian structure; they are limited to myopic one-step optimal decision making; or the state space is unconstrained [i.e. no constraints of the form (3) are considered]. Several recent works have addressed some of these issues but not all. For instance, simulation-based stochastic gradient optimization is proposed by Singh et al. (2007) in order to handle arbitrary motion and sensor models. The resulting method is provably convergent and it exploits the problem structure through control variates to reduce variance. A related approach (Martinez-Cantin et al. 2009) aimed at on-line active sensing employs Bayesian optimization, that is, using Gaussian process cost-function approximation to speed up the search. Several recent works with application to unmanned aerial vehicles (UAVs) also address some of the listed limitations but are still restricted to either stationary targets (Tisdale et al. 2009), one-step planning (Cole et al. 2008; Bryson and Sukkarieh 2009; Hoffmann and Tomlin 2010), or unconstrained scenarios (Geyer 2008; Ryan 2008).

1.4. Overview of contributions and approach

The distinctive feature of this paper is the treatment of the constraints (2) and (3) in the optimization (4). In particular, gradient-based optimization as in Paris and Le Cadre (2002), Mihaylova et al. (2003b), and Singh et al. (2007) is

not suitable unless a good starting guess is chosen since the constraints impose many local minima. In addition, special differentiation (Clarke et al. 1998) is required to guarantee convergence due to the non-smooth nature of the constraints.

To overcome these issues we instead employ a methodology based on global exploration of the solution space of vehicle trajectories. This is achieved through a *random tree* of feasible trajectories. Such a tree is constructed following ideas from sampling-based motion planning (LaValle 2006). The key property of motion-planning trees relevant to this paper is that the tree is guaranteed to reach asymptotically close to any reachable state in the state space as the algorithm iterates. Yet, the problem (4) is more difficult than a typical motion-planning problem because the cost is based on uncertainty that depends on the whole trajectory. In essence, the problem cannot be cast as a graph or tree search typically employed in motion planning, for example, to solve shortest-path problems, because the cost function (4) is not derived from a local metric and is not additive over separate trajectory segments. Additional tools are necessary. The solution proposed in this paper is to perform stochastic optimization over a solution space encoded through a dynamically adaptive trajectory tree.

The advantage of using a tree is that it provides a computationally efficient way to encode multiple solution trajectories and to propagate probability distributions recursively. While a uniformly random tree can asymptotically reach an optimal solution this might be an infinitely slow process in practice. Therefore, as with most Monte Carlo methods (Rubinstein and Kroese 2008) it is essential to exploit the problem structure in order to speed up the search. We employ three variance-reduction techniques to guide and accelerate the optimization:

1. The first, termed *biased sampling*, chooses tree nodes based on the expected utility of improving the target estimate in order to focus tree exploration into more 'promising' parts of the state space.
2. We then introduce a technique termed *shuffling*, which randomly modifies the tree structure in an attempt to lower the optimal cost. This is achieved by disconnecting a subtree from its parent and connecting it to a different part of the tree. The tree parts to be modified are chosen probabilistically.
3. The third technique introduced in the paper, termed *randomized pruning*, removes existing nodes probabilistically according to their performance.

While biased sampling has been widely used to speed up regular (i.e. deterministic) motion-planning algorithms (Choset et al. 2005), pruning and shuffling have not been previously employed in the context of sampling-based motion planning under uncertainty. These proposed methods result in a significant computational speed-up compared to a random baseline algorithm. Yet, currently, under general regularity conditions and no additional assumptions about the structure of the HMM (1) and constraints (2) and (3), formally only asymptotic convergence rates (i.e. as the number of iterations tends to infinity) are possible. Nevertheless, there is a sound reason why the combination of these three techniques is effective. A successful optimization methodology must address the exploration–exploitation trade-off paradigm (see e.g. Powell 2007) and do so with computational efficiency by dynamically adjusting the search space. The proposed optimization algorithm accomplishes that. In particular, the basic random tree expansion achieves exploration of the state space. The biased sampling and shuffling steps exploit information known a priori and collected during the algorithm operation to focus the search on more promising parts of the state space. Pruning is critical for maintaining a balance between the size and quality of the search space in order to achieve computational efficiency. These properties are summarized in Table 1 and will be developed in detail in the paper.

Table 1. Variance-reduction techniques.

Technique	Exploration	Exploitation	Computational efficiency
Tree expansion	√		
Biased sampling	√	√	
Shuffling		√	√
Pruning			√

1.5. Links to evolutionary computing

The resulting approach has close links to evolutionary computing. In particular, biased sampling and pruning based on an importance function correspond to *selection* using the 'fitness' criteria employed in genetic algorithms. Shuffling is related to *cross-over* and *migration* used in genetic programming (Langdon and Poli 2001) since a shuffle generates new trajectories by combining existing segments. A standard genetic algorithm could be used to perform the optimization (4) but will have difficulty managing the constraints (2) and (3) (e.g. see Michalewicz and Schoenauer 1996). Standard techniques, such as penalty functions or infeasible path rejection, employed in works such as Xiao et al. (1997), Vaidyanathan et al. (2001), Hocaoglu and Sanderson (2001), and Erinc and Carpin (2007) depend on parametrized paths and on cost-function tuning parameters. It is not clear how their performance scales as the environment becomes more cluttered. In contrast, sampling-based trees are specifically developed to handle systems with complicated dynamics and obstacle constraints. Therefore, this paper employs a general motion tree to automatically encode feasible candidate paths and avoid problem-specific parameter tuning. The stochastic optimization then amounts to dynamically adapting the tree structure so that there is convergence to an optimal trajectory. While shuffling and pruning might seem akin to standard genetic operation,

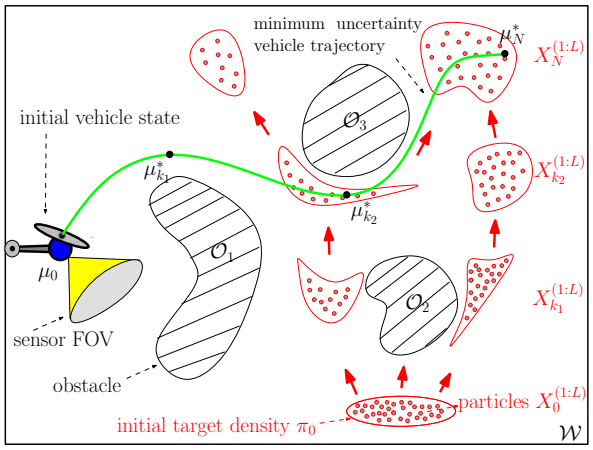


Fig. 1. A scenario with a vehicle (depicted as a small helicopter) at state $\mu_0 \in \mathcal{M}$ and a target with initial distribution π_0 diffusing north. Both target and vehicle avoid obstacles denoted $\mathcal{O}_i$. The set of possible target motions is approximated by L sampled trajectories $X_{0:N}^{(\ell)}$ for $\ell = 1, \ldots, L$. The figure shows the sampled states (particles) at the beginning $k = 0$, at two intermediate times $0 \le k_1 \le k_2 \le N$, and at the horizon $k = N$. We wish to find the vehicle trajectory $\mu_{0:N}^*$, which minimizes the expected target state estimate uncertainty. The vehicle sensor typically has a small field of view (FOV) relative to the environment size.

there is an important distinction–they are designed to operate over a 'population' encoded as a tree of trajectories rather than as separate paths as in a standard genetic algorithm. In that sense the proposed techniques are unique and make a bridge between evolutionary algorithms and randomized motion-planning methods.

2. An example scenario

Consider the scenario depicted in Figure 1. The vehicle and target operate in a workspace (i.e. an environment) denoted by $\mathcal{W}$, where $\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$ (Latombe 1991). The workspace contains a number of *obstacles* denoted by $\mathcal{O}_1, \ldots, \mathcal{O}_{n_o} \subset \mathcal{W}$, which the vehicle must avoid.

The vehicle state is defined as $\mu = (r, v) \in \mathcal{C} \times \mathbb{R}^{n_v}$ consisting of its configuration $r \in \mathcal{C}$ and velocity $v \in \mathbb{R}^{n_v}$. $\mathcal{C}$ is the vehicle configuration space describing, for example, the position, orientation, and joint angles of the system. Assume that the vehicle occupies a region $\mathcal{A}(r) \subset \mathcal{W}$ and that the function $\mathbf{prox}(\mathcal{A}_1, \mathcal{A}_2)$ returns the closest Euclidean distance between two sets $\mathcal{A}_{1,2} \subset \mathcal{W}$ and is negative if they intersect. One of the constraints defined in (3) is to avoid obstacles, generally expressed as

$$h_c^1((r, v)_k) = \min_i \mathbf{prox}(\mathcal{A}(r_k), \mathcal{O}_i), \text{ for all } 0 \le k \le N. \tag{5}$$

The framework developed in the paper will be applied to two types of vehicles. The first has a simple first-order model and operates in a polygonal obstacle environment [i.e. $\dim(\mathcal{W}) = 2$]–a setting suitable for measuring the algorithm performance compared to an idealized scenario. The second scenario is based on a low-flying underactuated UAV operating in a mountainous terrain in 3-D [i.e. $\dim(\mathcal{W}) = 3$)]. The simpler model is presented next while the helicopter application will be developed in Section 7.2.

2.1. A simple vehicle

Consider a point-mass vehicle moving in a plane. Its state space is $\mathcal{M} = \mathbb{R}^2 \times \mathbb{R}^2$ with state $\mu = (r, v)$ consisting of the position $r := (r_x, r_y) \in \mathbb{R}^2$ and velocity $v := (v_x, v_y) \in \mathbb{R}^2$. It evolves according to the simple dynamics

$$r_{k+1} = r_k + \tau v_k, \tag{6}$$

which is encoded by the function h_d defined in (2). The constant τ is the *time step*, that is, the sampling period, measured in seconds. The velocity v_k can be directly controlled but is bounded $\|v_k\| < v_{\max}$. For instance, in the scenario (Figure 1) a bound of $v_{\max} = 8$ m s^{-1} is chosen to create a problem that can be solved optimally only by a particular type of trajectory known in advance for a time horizon of 30 s.

2.2. Target dynamics

The target is modeled as a point mass on the ground with position $r = (r_x, r_y) \in \mathbb{R}^2$ and velocity $v = (v_x, v_y) \in \mathbb{R}^2$ forming the state $x = (r, v)$ with $\mathcal{X} = \mathbb{R}^2 \times \mathbb{R}^2$.

The target dynamics is governed by a general control law including a proportional term, such as arising from a goal attraction, a damping term in order to constrain the target speed, and obstacle-avoidance forcing. In addition, there is a white-noise acceleration component Ω with standard deviations σ_x and σ_y. The model is

$$\begin{aligned} r_{k+1} &= r_k + \tau v_k, \\ v_{k+1} &= v_k + \tau \Big(\omega_k + K_p(r_g - r_k) - K_d v_k \\ &\qquad + \begin{bmatrix} 0 & -k_o/d_k \\ k_o/d_k & 0 \end{bmatrix} v_k \Big), \\ \omega_k &\sim \text{Normal}(0, \text{diag}(\sigma_x^2, \sigma_y^2)), \end{aligned} \tag{7}$$

where τ is the time step; $K_p > 0$ and $K_d > 0$ are the potential and dissipative matrices, respectively; $k_o > 0$ is an obstacle steering scalar gain; $r_g \in \mathbb{R}^2$ is a constant goal location; and $d_k = \|g_k\|$ with $g_k := r_o - r_k$, where r_o is the closest point on the obstacle set. The model (7) corresponds to the function f defined in (1a).

Such a model is chosen in order to add realism to the target motion as if it were executing an actual navigation task. The amount of knowledge of this task can be tuned using the parameters. For instance, in the numerical scenarios studied in this paper we use

$$K_p = \text{diag}(0, 0.03), \; K_d = \text{diag}(0, 0.6), \; r_g = (0, 200), \\ k_o = 50, \text{ if } |\beta_k| < \pi/2, \text{ else } k_o = 0,$$

where β_k is the angle between v_k and g_k. Assuming the target starts at the origin, the dynamics would drive it north, avoiding obstacles if too close, diffusing in uncertain directions to east or west, and finally ending up and staying around the boundary line $r_y = 200$.

2.3. Sensor model

The vehicle is equipped with sensors, which provide the relative range and bearing to the target; hence, $\mathcal{Y} = \mathbb{R}^2$. The target is observed only if its line of sight is not obstructed by obstacles and if it falls within the *sensing distance* d_s of the vehicle. This is formally expressed through the target *visibility area* $\mathcal{V}(r') \subset \mathcal{W}$ for given vehicle position $r' = (r'_x, r'_y) \in \mathbb{R}^2$ defined by

$$\mathcal{V}(r') := \big\{ r \in \mathcal{W} \mid \|r - r'\| < d_s \text{ and } h^1_c\big((r' + \alpha(r - r'), \cdot)\big) \geq 0,\ \forall \alpha \in [0,1] \big\}.$$

In order to define the sensor function (1b) for a target with position $r = (r_x, r_y)$ first define the *perfect* sensor function

$$g^*((r,v);(r',v')) = \begin{bmatrix} \|r - r'\| \\ \arctan((r_y - r'_y),(r_x - r'_x)) \end{bmatrix}. \tag{8}$$

This function is only valid if the target reading originated from its visibility region. In addition, there is a small probability $P_f \in [0,1]$ of a false reading uniformly distributed over the visibility region. The actual sensor function is then given by

$$g((r,v);(r',v'),V) = \begin{cases} \left\{ \begin{array}{ll} g^*((r,v);(r',v')) + \frac{\|r-r'\|}{d_s} V, & r \in \mathcal{V}(r') \\ \emptyset, & r \notin \mathcal{V}(r') \end{array} \right\} & \text{if } u_f \geq P_f \\ g^*((r^f,v);(r',v')),\ r^f \sim \mathcal{U}(\mathcal{V}(r')) & \text{if } u_f < P_f, \end{cases} \tag{9}$$

with noise $V := (V_d, V_b) \sim \text{Normal}(0, \text{diag}(\sigma_d^2, \sigma_b^2))$ where σ_d and σ_b define the range and bearing standard deviations, respectively, and u_f is a uniform sample from $[0,1]$.

2.4. Cost function

Finally, the vehicle is interested in minimizing the uncertainty in the target position estimate. This is encoded by simply setting

$$\varphi(x) = (r_x, r_y) \in \mathbb{R}^2$$

when performing the optimization (4).

3. Problem formulation

An alternative way to express the HMM (1) is through the known densities

$$X_0 \sim \pi_0, \tag{10a}$$
$$X_k \sim p(\cdot|X_{k-1}), \qquad k > 0 \tag{10b}$$
$$Y_k \sim q(\cdot|X_k;\mu_k), \qquad k > 0 \tag{10c}$$

where π_0 is the initial distribution. Note that the expectation operator $\mathbb{E}[\cdot]$ used throughout the paper is applied with respect to these densities, unless noted otherwise. The filtering density employed in the computation of the cost (4) is then expressed recursively (e.g. Robert and Casella 2004) according to

$$\pi_k(x|y_{1:k};\mu_{1:k}) = \frac{q(y_k|x;\mu_k)\int p(x|x')\,\pi_{k-1}(x'|y_{1:k-1};\mu_{1:k-1})\,dx'}{\int q(y_k|x;\mu_k)\int p(x|x')\,\pi_{k-1}(x'|y_{1:k-1};\mu_{1:k-1})\,dx'dx}. \tag{11}$$

In addition, the tree optimization algorithm will require the definition of the *prediction density* at time $k+i$, for $i > 0$, after receiving measurements only during the first k epochs. It is denoted $\pi_{k+i|k}$ and defined by

$$\pi_{k+i|k}(x|y_{1:k};\mu_{1:k}) = \int p(x|x')\,\pi_{k+i-1|k}(x'|y_{1:k};\mu_{1:k})\,dx', \tag{12}$$

with $\pi_{k|k} \equiv \pi_k$. The estimate of $\varphi(X_N)$ after collecting a sequence of measurements $y_{1:k}$ obtained from a vehicle trajectory $\mu_{1:k}$ is denoted $\Phi_{N|k} : \mathcal{Y}^k \times \mathcal{M}^k \to \mathbb{R}^{n'_x}$ and defined by

$$\Phi_{N|k}(y_{1:k};\mu_{1:k}) := \mathbb{E}[\varphi(X_N) \mid y_{1:k};\mu_{1:k}] = \int \varphi(x_N)\,\pi_{N|k}(x|y_{1:k};\mu_{1:k})\,dx. \tag{13}$$

The objective function in (4) or, equivalently, the expected *uncertainty cost* at time N given a vehicle trajectory $\mu_{0:k}$, for $k \leq N$, is denoted by $J_{N|k} : \mathcal{M}^k \to \mathbb{R}$ and defined as

$$J_{N|k}(\mu_{1:k}) = \mathbb{E}\left[\|\varphi(X_N) - \Phi_{N|k}(Y_{1:k};\mu_{1:k})\|^2\right] = \int \|\varphi(x_N) - \Phi_{N|k}(y_{1:k};\mu_{1:k})\|^2\, p(x_{0:N}, y_{1:k}|\mu_{1:k})\,dx_{0:N}dy_{1:k}. \tag{14}$$

The expectation over states and measurements in (14) is taken with respect to the density $p(x_{0:N}, y_{1:k}|\mu_{1:k})$, which, for Markov models in the form (10), can be decomposed [see e.g. Robert and Casella (2004)] as

$$p(x_{0:N}, y_{1:k}|\mu_{1:k}) = \pi_0(x_0)\prod_{i=1}^{N} p(x_i|x_{i-1})\prod_{i=1}^{k} q(y_i|x_i;\mu_i). \tag{15}$$

The cost of a complete trajectory $\mu_{1:N}$ is denoted for brevity by $J := J_{N|N}$. The goal (4) is then expressed in short as

$$\mu^*_{1:N} = \arg\min_{\mu_{1:N}} J(\mu_{1:N}), \tag{16}$$

subject to the dynamics and constraints.

4. Sampling-based approximation

The filtering densities (11) and (12) generally cannot be computed in closed form since they are based on non-linear or non-Gaussian models. Therefore, following Singh et al. (2007), we employ a particle-based approximation using L delta distributions, placed at state *samples* $X_k^{(j)} \in \mathcal{X}$ with positive weight functions $w_k^{(j)} : \mathcal{Y}^k \times \mathcal{M}^k \to \mathbb{R}_+$, that is

$$\pi_k(x|y_{1:k};\mu_{1:k}) \approx \hat{\pi}_k(x|y_{1:k};\mu_{1:k}) := \sum_{j=1}^{L} w_k^{(j)}(y_{1:k};\mu_{1:k})\,\delta_{X_k^{(j)}}(x), \quad (17)$$

where δ_y denotes the Dirac delta mass at point y.

A simple way to construct such a representation is to sample L independent trajectory realizations $\{X_{0:k}^{(j)}\}_{j=1}^{L}$ using the prior (10a) and target motion model (10b) and to compute the weights, for given measurements $y_{1:k}$ obtained at vehicle states $\mu_{1:k}$, according to

$$\bar{w}_k^{(j)}(y_{1:k};\mu_{1:k}) := \prod_{i=1}^{k} q(y_i|X_i^{(j)};\mu_i), \quad (18)$$

$$w_k^{(j)} = \frac{\bar{w}_k^{(j)}}{\sum_{\ell=1}^{L} \bar{w}_k^{(\ell)}}, \quad (19)$$

so that the weights are normalized, that is, $\sum_{j=1}^{L} w_k^{(j)} = 1$. This is equivalent to a sequential importance sampling (SIS) scheme with importance distribution $\pi_0(x_0)\prod_{i=1}^{k} p(x_i|x_{i-1})$. Note that while more sophisticated sampling methods have been developed, for example, that additionally account for measurements to reduce variance (Doucet et al. 2001; Robert and Casella 2004), this paper follows the basic choice for simplicity. Figure 1 depicts a subset of possible evolutions of such particles in the helicopter search scenario.

With this representation it is straightforward to show that (12) is approximated simply according to

$$\pi_{k+i|k}(x|y_{1:k};\mu_{1:k}) \approx \hat{\pi}_{k+i|k}(x|y_{1:k};\mu_{1:k}) := \sum_{j=1}^{L} w_k^{(j)}(y_{1:k};\mu_{1:k})\,\delta_{X_{k+i}^{(j)}}(x). \quad (20)$$

The estimate (13) is then approximated by

$$\Phi_{N|k}(y_{1:k};\mu_{1:k}) \approx \hat{\Phi}_{N|k}(y_{1:k};\mu_{1:k}) := \sum_{j=1}^{L} \varphi(X_N^{(j)})\,\hat{\pi}_{N|k}(X_N^{(j)}|y_{1:k};\mu_{1:k}). \quad (21)$$

Note that updating the cost along a vehicle trajectory has computational complexity $O(L^2)$ per time step. Yet, due to particle independence the computation can be parallelized using special hardware up to a factor of $O(L)$ and sped up significantly.

As the time N increases the approximation (21) degrades since the probability mass becomes concentrated in a decreasing number of particles (Robert and Casella 2004). A standard remedy is to include a resampling step (Doucet et al. 2001) to redistribute the samples equalizing the weights. While it is possible to perform sequential importance resampling (SIR) in the proposed framework it is avoided for computational reasons specific to the tree structure employed for uncertainty propagation. The drawback is that the method is limited to small time horizons, such as $N < 30$. The distinct advantage though is that the simpler SIS scheme permits a computationally efficient update of the density (17) and estimate (21) during optimization. The idea (described in detail in the following sections) is that SIS can be implemented as a *simple and fast* parallel weights rescaling in a dynamically changing tree of vehicle trajectories that explores the solution space.

Finally, the error $J_{N|k}$ is approximated through importance sampling of the integrand in (14), that is by drawing $\left(X_{0:N}^{(\ell)}, Y_{1:k}^{(\ell)}\right)$ from $p(x_{0:N}, y_{1:k}|\mu_{1:k})$. It is natural to use the i.i.d. state particles $X_{0:N}^{(\ell)}$ already sampled for the approximation of the density (17). Measurement sequences $Y_{1:k}^{(\ell)}$ are then sampled by drawing $Y_i^{(\ell)} \sim q(\cdot|X_i^{(\ell)};\mu_i)$ for all $i = 1,\ldots,k$. As long as the densities (10) can be directly sampled from, which is valid for common models used in robotics (e.g. Thrun et al. 2005), then the approximation simplifies to the Monte Carlo or the stochastic counterpart, that is

$$J_{N|k}(\mu_{1:k}) \approx \hat{J}_{N|k}(\mu_{1:k}) := \frac{1}{L}\sum_{\ell=1}^{L} \left\| \varphi\left(X_N^{(\ell)}\right) - \hat{\Phi}_{N|k}\left(Y_{1:k}^{(\ell)};\mu_{1:k}\right)\right\|^2, \quad (22)$$

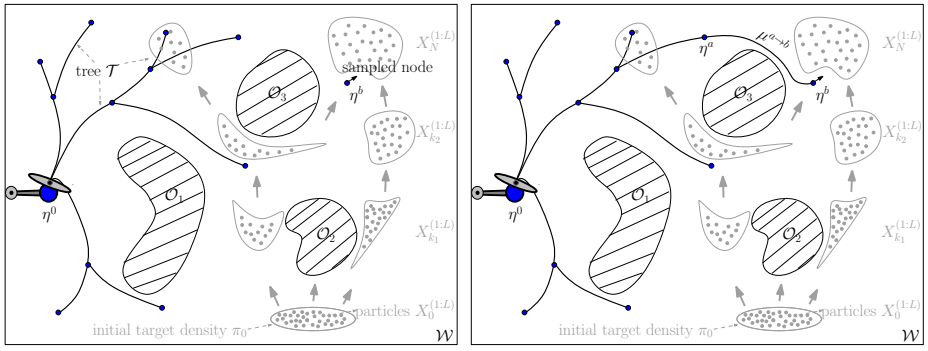
with $\hat{J} := \hat{J}_{N|N}$ denoting the approximate cost of a whole trajectory $\mu_{1:N}$. The global optimization algorithms developed in this paper will be based on the approximate estimate (21) and cost (22), that is, they will solve

$$\hat{\mu}_{0:N}^{*} = \arg\min_{\mu_{0:N}} \hat{J}(\mu_{0:N}). \quad (23)$$

In this sense only an approximate solution will be obtained. Yet, by the law of large numbers (Del Moral 2004) $\hat{\mu}_{0:N}^{*}$ will approach the true solution $\mu_{0:N}^{*}$ by increasing the number of simulations L.

5. Random tree optimization

The nature of the constraints (2) and (3) renders gradient-based methods unsuitable for solving (23). An alternative is to discretize the vehicle state space $\mathcal{M}$, for example, using a grid and generating candidate paths by transitioning between adjacent cells. Such an approach is generally limited to a few dimensions, such as $\dim(\mathcal{M}) \leq 3$, and to



(a) A tree $\mathcal{T}$ and a newly sampled node η^b (b) Connecting η^b to the tree

Fig. 2. A tree-expansion step implemented by **Expand** (Section 5.1): (a) a node η^b is sampled; (b) it is then connected to a randomly chosen node $\eta^a \in \mathcal{T}$.

systems with very simple dynamics, for example an unconstrained point mass in the plane. This is due to the exponential (both in state dimension and trajectory epochs) size of the search space, also known as the *curse of dimensionality*.

In this paper we also employ a tree-based search but unlike a standard discrete search, the nodes of the tree are *sampled* from the original continuous space $\mathcal{M}$ and the edges correspond to trajectories satisfying any given dynamics (2) and general constraints (3). Our approach is based on a recent methodology under active development in the robotics community known as *sampling-based motion planning*, which includes the *rapidly exploring random tree* (RRT) (LaValle 2006) and the *probabilistic road map* (PRM) (Choset et al. 2005). Unlike these motion-planning algorithms, the trees employed in this paper are not expanded based on a 'distance' metric between nodes. Instead, the connections are made probabilistically, nodes and edges can be added, swapped, or deleted during the algorithm operation. This section considers the basic tree expansion that explores the state space, while Section 6 introduces the variance-reduction techniques that complete the overall approach.

5.1. Tree expansion

The set of nodes is denoted by $\mathcal{N} := \mathbb{N} \times \mathcal{M} \times \mathbb{R}^{L\times L} \times \mathbb{R}_+ \times \mathbb{N}$. Each *node* is defined by the tuple

$$\eta = (k, \mu, W, \hat{J}, \rho) \in \mathcal{N},$$

consisting of the epoch index $0 \le k \le N$; the vehicle state μ; the particle weights matrix W, which gives a convenient way to compute the density $\hat{\pi}$; the target state estimate uncertainty cost $\hat{J} \ge 0$; and the tree parent index ρ. Nodes and their subelements are indexed by superscripts, that is, η^a has state μ^a and its parent node is η^{ρ^a}. The *root* of the tree that contains the starting vehicle state is denoted $\eta^0 = (0, \mu_0, W^0, \hat{J}^0, \cdot)$, where the matrix elements $W^0_{\ell j} = 1/L$ for all $\ell, j = 1, \ldots, L$. A *trajectory* between two nodes, η^a and η^b, is denoted $\mu^{a\to b}$ and a state at time k along this trajectory is denoted $\mu_k^{a\to b}$ where $k^a \le k \le k^b$.

A tree $\mathcal{T} \subset \mathcal{N}$ is a set of nodes connected by feasible trajectories. The tree structure guarantees that there is a unique trajectory leading from the root to each node $\eta^a \in \mathcal{N}$, which is denoted $\mu^{0\to a}$. An overview of the elements comprising each node is given in Table 2. Their exact computation is detailed next.

A tree is constructed by assuming that a local controller is available (LaValle 2006) that attempts to drive the vehicle between two given nodes η^a and η^b. For instance, if the states μ^a and μ^b were close enough and no obstacles between them were present then a trajectory $\mu^{a\to b}$ would be produced, otherwise the connection fails. Such a controller is abstractly represented by the function **Connect**, that is

$$\textbf{Connect}(\eta^a, \eta^b) \Rightarrow \begin{cases} \mu^{a\to b}, & \text{if path found} \\ \emptyset, & \text{otherwise.} \end{cases} \tag{24}$$

The tree is constructed by sampling and connecting nodes. Assume that a function **Sample** is available, which returns a new node, that is

$$\eta^b = \textbf{Sample}(). \tag{25}$$

The default choice is to sample (state, time) pairs (μ, k) uniformly from $\mathcal{M} \times \{1, \ldots, N\}$ and discard samples that violate the constraints (3), for example, which lie inside obstacles. Next define the set $\mathcal{T}^{\to b} \subset \mathcal{T}$ of all existing tree nodes for which a feasible trajectory to the newly sampled node can be found, that is

$$\mathcal{T}^{\to b} = \{\eta \in \mathcal{T} \mid \textbf{Connect}(\eta, \eta^b) \ne \emptyset\}.$$

One of these nodes denoted $\eta^a \in \mathcal{T}^{\to b}$ is selected uniformly at random to become the parent of η^b linked with trajectory $\mu^{a\to b}$, that is $\rho^b = a$. Figure 2 illustrates the construction.

After a new node η^b is added to the tree, the target filtering density $\hat{\pi}$ (20) is propagated along the newly added trajectory segment $\mu^{a\to b}$ for all sampled target paths $X^{(\ell)}_{k^a:k^b}$ by simulating measurements

$$Y_k^{(\ell)} \sim q(\cdot | X_k^{(\ell)}; \mu_k^{a\to b}), \quad \text{for } k = k^a, \ldots, k^b, \tag{26}$$

Table 2. Description of the elements of a node $\eta^b = (k^b, \mu^b, W^b, c^b, \hat{J}^b, \rho^b) \in \mathcal{T}$.

Variable	Type	Element description
k^b	integer ≥ 0	time epoch index
μ^b	$\dim(\mathcal{M}) \times 1$ vector	vehicle state at node
W^b	$L \times L$ matrix	weights $W^b_{\ell j} := w^{(j)}_{k^b}(Y^{(\ell)}_{1:k^b}; \mu^{0\to b})$, for $\ell, j = 1, \ldots, L$
$\hat{J}^b$	scalar > 0	uncertainty cost $\hat{J}^b := \hat{J}_{N\mid k^b}(\mu^{0\to b})$
ρ^b	integer ≥ 0	parent-node index

for all $\ell = 1, \ldots, L$. A row in the matrix W^b, that is, W^b_ℓ for any $1 \leq \ell \leq L$, corresponds to the resulting weights for each measurement sequence, that is $W^b_{\ell j} := w^{(j)}_{k^b}(Y^{(\ell)}_{k^b}; \mu^{0\to b})$, where $w^{(j)}_{k^b}$ is defined in (18). The weights are computed *incrementally* using the parent weights $W^a_{\ell j}$ through

$$\bar{W}^b_{\ell j} = W^a_{\ell j} \cdot U_{\ell j}, \text{ where } U_{\ell j} := \prod_{k=k^a}^{k^b} q(Y^{(\ell)}_k | X^{(j)}_k; \mu^{a\to b}_k), \tag{27a}$$

$$W^b_{\ell j} = \frac{\bar{W}^b_{\ell j}}{\sum_{j=1}^{L} \bar{W}^b_{\ell j}}. \tag{27b}$$

The error (22) of the complete trajectory $\mu^{0\to b}$, denoted by $\hat{J}^b$, is

$$\hat{J}^b := \hat{J}_{N\mid k^b}(\mu^{0\to b}) = \sum_{\ell=1}^{L} \left\| \varphi(X^{(\ell)}_N) - \sum_{j=1}^{L} W^b_{\ell j} \varphi(X^{(j)}_N) \right\|^2. \tag{28}$$

Note again that $\hat{J}^b$ represents the uncertainty measure at the end of the time horizon N but only based on measurements collected along $\mu^{0\to b}$, that is, up to time k^b. If $\hat{J}^b < \hat{J}^*$, where $\hat{J}^b$ is computed using (28) and $\hat{J}^*$ is the current best cost, then the current best node is reset, that is, $\eta^* = \eta^b$. The updated optimal vehicle trajectory can be backtracked from η^b to the root η^0.

Let $s \sim \mathcal{U}(\mathcal{S})$ denote uniform sampling of an element s from a finite set $\mathcal{S}$. The complete tree-expansion algorithm can now be summarized as

Expand
1. $\eta^b = \textbf{Sample}()$
2. $\eta^a \sim \mathcal{U}(\mathcal{T}^{\to b})$
3. $\mu^{a\to b} = \textbf{Connect}(\eta^a, \eta^b)$
4. $\mathcal{T} = \mathcal{T} \cup \{\eta^b\}$; $\rho^b = a$
5. compute W^b and $\hat{J}^b$ using (27) and (28)
6. if $\hat{J}^b < \hat{J}^*$ then $\eta^* = \eta^b$

The expansion is repeated $n-1$ times in order to produce a tree with n nodes. Initially, the tree contains only the root, that is, $\mathcal{T} = \{\eta^0\}$, and $\eta^* = \eta^0$.

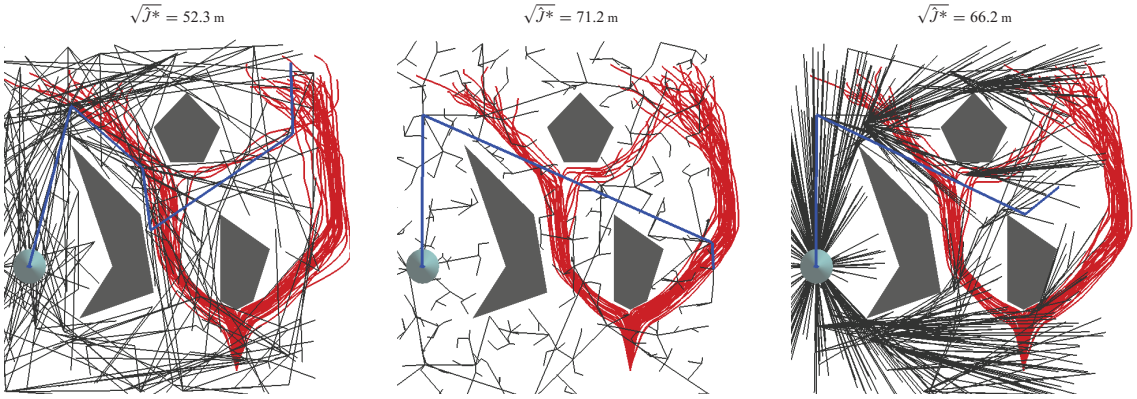
5.1.1. Computational saving It is important to stress that the computation (28) is accomplished through an *incremental propagation* of the filtering density weights along the newly added trajectory $\mu^{a\to b}$ from parent η^a to child node η^b rather than the complete trajectory $\mu^{0\to b}$. This is the advantage of using a tree rather than a naive enumeration of vehicle trajectories in order to explore the solution space $\mathcal{M}^k$. For instance, assume that the tree were a complete binary tree with n nodes and, hence, with depth $d = \log(n+1) - 1$. Then it encodes $n/2$ different trajectories since each leaf can be backtracked to generate a unique trajectory $\mu_{0:N}$. If each edge lasts on average N/d time epochs then the density computation (27) must be performed $\frac{n}{d}N$ times for the tree compared to $\frac{n}{2}N$ times if the $n/2$ trajectories were enumerated. In other words, the tree provides an $O(\log(n))$ saving factor on average.

5.2. Example: simple vehicle

Consider a simple vehicle with dynamics (6). Since there is no bound on accelerations a trajectory between two nodes is simply a line segment with constant velocity. The function **Connect** introduced in Section 5.1 takes the form:

Connect(η^a, η^b)
1. if $k^b = \infty$,
2. $\quad k^b = k^a + \left\lceil \frac{\|r^b - r^a\|}{\tau v_{\max}} \right\rceil$
3. $v = \frac{r^b - r^a}{\tau(k^b - k^a)}$
4. for $k = k^a : k^b$
5. $\quad r_k = r^a + \frac{k-k^a}{k^b-k^a}(r^b - r^a)$; $\mu^{a\to b}_k = (r_k, v)$
6. $\quad$ if $h^1_c(\mu^{a\to b}_k) < 0$ return $\emptyset$
7. return $\mu^{a\to b}$

It begins by checking whether the trajectory to be computed must have a fixed final time k^b. When the final time epoch k^b is set to ∞ (line 1) then any k^b such that $k^a < k^b \leq N$ is allowed. This occurs when η^b is a uniform sample[2] in which case the trajectory $\mu^{a\to b}$ is generated using the maximum permitted velocity $v_{\max}$ and k^b is set to the resulting time (line 2). The constant velocity along the trajectory is computed in line 3. The points along the trajectory are then linearly interpolated (line 5). If the trajectory intersects any obstacles then the connection fails (line 6).



(a) Random tree (RND) (b) Tree with local metric d (RRT) (c) Tree with cost-to-come metric $\bar{d}$ (iPRM)

Fig. 3. Three types of search trees used to explore the vehicle trajectory space corresponding to the scenario in Section 2. The vehicle has simple dynamics and a circular sensing radius shown as a disk at its starting state. A subset of the target paths $X_{0:N}^{(1:L)}$ are shown diffusing from the bottom right to the top of the environment. The trees are: (a) a random tree (RND) constructed using **Expand** (Section 5.1); (b) a rapidly exploring random tree (RRT) using the nearest-neighbor metric d (29); (c) an incremental tree-based probabilistic road map (iPRM) expanded based on the cost-to-come distance $\bar{d}$ (30). Each tree has 500 nodes. While the optimal (i.e. with minimum target position variance) trajectories of all trees are quite different (shown as thicker lines), they all yield very similar costs $\hat{J}^*$. It is evident that these solutions are of *poor quality* since the square root of the variance is large (i.e. > 52.3 m) relative to the environment size (200×200 m).

The optimal estimation algorithm is tested in a polygonal obstacle environment mimicking the scenario in Section 2. A tree built after calling **Expand** 500 times is shown in Figure 3(a). It takes a few milliseconds of computation to generate such a tree. In practice, a tree will contain tens of thousands of vertices. Figure 4 shows a few frames of the resulting motion along an optimal trajectory obtained by a denser tree with 10,000 nodes which took 5 s to compute. More detailed computational studies are performed in Section 7.

5.3. Other expansion methods

There are alternative methods of constructing an exploration tree. For instance, instead of connecting nodes at random, a newly sampled node can be connected to an existing node in the tree based on some deterministic criteria. Classical planning trees such as an RRT employ *nearest*-neighbor connections. Nearest should be understood with respect to a predefined pseudo-*distance metric* $d : \mathcal{N} \times \mathcal{N} \to \mathbb{R}$.[3] Typical metrics include the Euclidean distance $d(\eta^a, \eta^b) = \|\mu^b - \mu^a\|$ (assuming $\mathcal{M}$ is a vector space) or the time of travel between nodes

$$d(\eta^a, \eta^b) = k^b - k^a. \tag{29}$$

Using such a local cost d has the advantage of quickly exploring the state space. It also has a major drawback: the cumulative cost of a path (e.g. its length or total time) can significantly deviate from an existing shortest path. More intuitively, even though the tree would approximate all states in the domain, most of the paths will be jagged and circuitous. For instance, Figure 3(b) shows such a tree computed for a scenario mimicking the setup of Figure 1.

This can be remedied by considering the cost-to-come c defined by

$$c^b = c^a + d(\eta^a, \eta^b),$$

with the cost of the root $c^0 = 0$. The nearest neighbor is then chosen according to a modified distance $\bar{d}$ defined by

$$\bar{d}(\eta^a, \eta^b) = c^a + d(\eta^a, \eta^b). \tag{30}$$

A tree based on $\bar{d}$ will contain 'straighter' trajectories, see Figure 3(c), which take a minimum time to reach the reachable points in the state space. Such a tree is termed an *incremental probabilistic road map* (iPRM) to distinguish it from the standard RRT. These issues are discussed in Frazzoli et al. (2002) who propose a general motion-planning algorithm based on a combined metric of d and $\bar{d}$.

More formally, the nearest existing node $\eta^a \in \mathcal{T}$ to a sampled node η^b is given by

$$\eta^a = \arg \min_{\eta \in \mathcal{T}^{\to b}} d(\eta, \eta^b). \tag{31}$$

Such nearest-neighbor expansion is achieved by replacing line 2 in routine **Expand** with (31).

While the RRT and iPRM are standard choices for motion-planning problems, as we will show they are not a good option when optimizing uncertainty-based cost functions, that is, solving (4). The issue is that the cost J of a path cannot be expressed as a summation over the costs of its individual edges. In that respect, the random algorithm RND turns out to be more effective.

5.4. Probabilistic completeness

Under certain assumptions a random tree can reach asymptotically close to any state that is reachable. Recall the following lemma adapted to the settings of this paper:

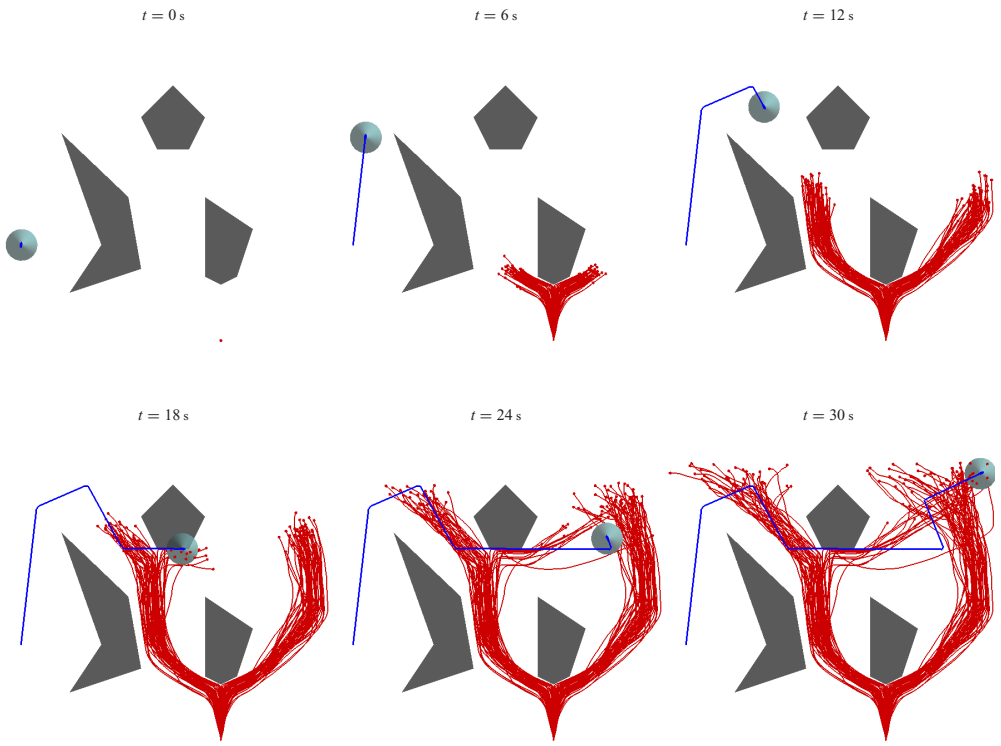


Fig. 4. The optimal vehicle path computed using algorithm **Expand** (Section 5.1) with $L = 100$ particles, time horizon $T = 30$ s, and time step $\tau = 250$ ms. The consecutive frames show the evolution of the sampled target trajectories $X_{0:N}^{(1:L)}$ and the vehicle trajectory $\mu_{0:N}^*$. The computed cost is $\sqrt{\hat{J}^*} = 43.9$ m.

Lemma 5.1. *(See e.g. Ladd and Kavraki 2004.) After selecting n nodes, the probability of failing to find a path from the root η^0 to a uniformly sampled node $\eta^a \in \mathcal{N}$ reduces exponentially in n. More formally,*

$$P(\nexists \mu^{0\to a}) < \ell(1-c)^n,$$

for some constants $\ell, c \in (0,1)$.

The main assumption in Lemma (5.1) is that each node can be connected to a sufficiently large number of other nodes using the **Connect** routine. The collection of these nodes is called the *reachable set* and its size depends on the constraints (2) and (3), for example, it shrinks if a vehicle is slow or if the number of obstacles in the environment increases. The intuition behind Lemma (5.1) then is that, as long as every node has large enough reachable space (the volume of which is proportional to c) and, under the assumption that the path has a finite length (related to the constant ℓ), then adding more nodes would exponentially increase the probability of finding the path. The precise technical conditions that render Lemma (5.1) applicable to the scenario of this paper are developed in Hsu et al. (2002) and Frazzoli et al. (2002).

6. Variance-reduction techniques

The advantage of constructing the exploration tree described in Section 5 is that it asymptotically reaches arbitrarily close to any state in the state space $\mathcal{M}$ assuming that it is bounded. To guarantee that formally (as described in Section 5.4) the tree is constructed using uniform node sampling and random connection of new nodes. Note that the expansion is agnostic to the uncertainty cost $\hat{J}$ along trajectories that we actually seek to minimize. This is problematic because it might take infinitely long to explore a reasonable fraction of trajectories with low $\hat{J}$. Therefore, in the spirit of Monte Carlo optimization based on importance sampling (Rubinstein and Kroese 2008) as well as genetic computation, we propose three techniques that retain the probabilistic completeness of random trees but at the same time drastically speed up the optimization.

6.1. Utility sampling of nodes

The difficulty of optimization in a complicated high-dimensional landscape can in practice be alleviated by incorporating problem-specific knowledge. For instance, the set of nodes considered during randomized motion planning can be chosen in a biased way, for example, proportional to some *utility* function known to reduce the trajectory cost (see e.g. Burns and Brock 2005).

This paper employs a similar approach dictated by the fact that an optimal vehicle trajectory $\mu_{0:N}^*$, that is, with lowest uncertainty cost J, is likely to pass close to states

with a high observation likelihood. Thus, a sample μ_k^{sample} is chosen so that

$$\mu_k^{\text{sample}} = \arg\max_{\mu} q(\, Y_k^{\ell} | X_k^{\ell}; \mu), \tag{32}$$

where $(X_k^{(\ell)}, Y_k^{(\ell)})$ is a single particle selected by sampling ℓ uniformly from $\{1, \ldots, L\}$. The optimal state μ in (32) is usually straightforward to compute. For instance, the optimal vehicle position in the example scenario from Section 2 will coincide (on average) with the target position at X_k^{ℓ}, so according to (32) one can simply set $\mu_k^{\text{sample}} = X_k^{\ell}$.

It is also possible to sample μ by minimizing the joint likelihood over all particles. Since this results in a complex multimodal optimization subject to sensor visibility constraints [such as (9)] we choose to use the simplest form (32) as it does not add extra complexity to the overall algorithm.

The function **Sample** introduced in Section 5.1 is specified as follows. It samples a state μ in two ways: 1) based on the utility (32) and 2) uniformly in the space $\mathcal{M}$. It selects the former with probability P_U, otherwise it selects the latter at every tree expansion. The routine is summarized as

Sample
1. with probability P_U,
2. $\quad k \sim \mathcal{U}(\{0, \ldots, N\})$; $\ell \sim \mathcal{U}(\{1, \ldots, L\})$
3. $\quad \mu = \arg\max_{\mu'} q(\, Y_k^{\ell} | X_k^{\ell}; \mu')$, where $Y_k^{\ell} \sim q(\cdot | X_k^{\ell}; \mu')$
4. otherwise
5. $\quad k = \infty$
6. $\quad \mu \sim \mathcal{U}(\mathcal{M})$
7. repeat Sample if $h_c(\mu) < 0$
8. return $\eta = (k, \mu, \ldots)$

A sampled node is accepted as long as it satisfies the constraints (line 7). A utility node is sampled by choosing its time epoch k (line 2) while a uniform node has time $k = \infty$ (line 5). This is related to the way the function **Connect**(η^a, η^b) links two nodes η^a and η^b. Whenever $k^b = \infty$, **Connect** is allowed to produce a trajectory $\mu^{a \to b}$ with any final time $k^b \leq N$ that it chooses. In such cases **Connect** is typically designed to compute a time-optimal trajectory so that k^b is minimized. In contrast, when k^b is set to a specific value (line 2) then **Connect** produces a trajectory that arrives at μ^b at time epoch k^b exactly.

6.2. Tree shuffling

Shuffling is the process of probabilistically selecting a branch of the tree, detaching it from its parent and attaching it to another branch. The first step is to choose a node η^a at random. Then n_r other existing nodes are selected from the tree according to a 'fitness' function. Each of these nodes, denoted $\eta^b \in \mathcal{T} \backslash \{\eta^a \cup \eta^0\}$, are then disconnected from their current parents η^{ρ^b} and connected to η^a instead, as long as this switch lowers the resulting uncertainty cost of the subtree attached at η^b (see Figure 6).

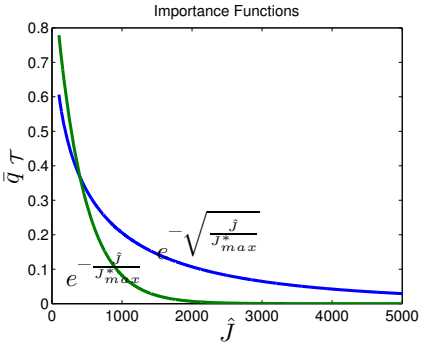


Fig. 5. Two importance density functions $\bar{q}_{\mathcal{T}}$ used to sample nodes during shuffling (with $J^*_{\max} = 400$). The function with the 'fatter' tail [defined by (33)] is the proper choice to guarantee exploration of the state space.

The *fitness density* over a given set $\mathcal{T}$ is denoted $q_{\mathcal{T}} : \mathcal{T} \to [0, 1]$ and defined by

$$q_{\mathcal{T}}(\eta^b) = \frac{\bar{q}_{\mathcal{T}}(\eta^b)}{\sum_{\eta \in \mathcal{T}} \bar{q}_{\mathcal{T}}(\eta)}, \text{ where } \bar{q}_{\mathcal{T}}(\eta^b) = e^{-\sqrt{\frac{\hat{J}^b}{J^*_{\max}}}}, \tag{33}$$

where $J^*_{\max}$ is a constant denoting the upper bound of an acceptable optimal cost that the algorithm is expected to yield. Sampling from the fitness function biases the selection of more capable nodes but without completely disregarding nodes with lower performance. This is achieved by a distribution with a fat tail as shown in Figure 5.

Let the *subtree* rooted at η^b be denoted $\mathcal{T}^b \subset \mathcal{T}$. Define the combined trajectory connecting node η^a to node η^b and node η^b to node η^c, denoted $\mu^{a \to b \to c}$, by

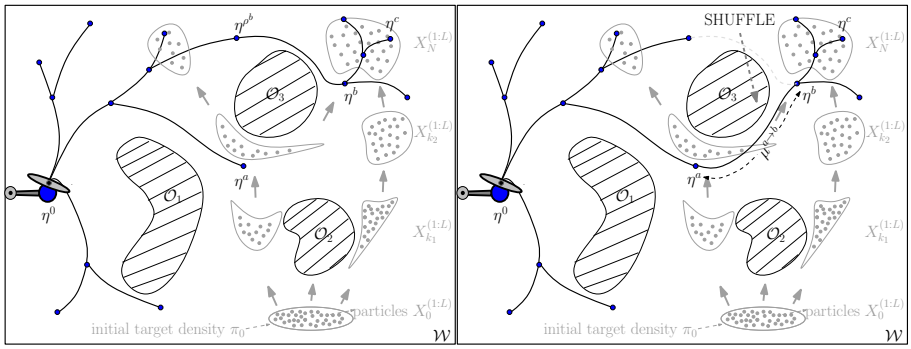
$$\mu^{a \to b \to c} := \mu^{a \to b} \cup \mu^{b \to c}.$$

More precisely, a *shuffle*, that is, a parent switch $\rho^b = a$, occurs in two cases (see also Figure 6). The obvious case is when the current optimal uncertainty cost $\hat{J}^*$ can be improved by a trajectory $\mu^{0 \to a \to b \to c}$ in the modified tree. The second case is heuristic: a switch occurs only if the cost can be lowered on average across all nodes in the subtree. These conditions are expressed as

$$\text{if} \left\{ \begin{array}{c} \min_{\eta^c \in \mathcal{T}^b} \hat{J}_{N|k^c}(\mu^{0 \to a \to b \to c}) < \hat{J}^* \\ \text{or} \\ \sum_{\eta^c \in \mathcal{T}^b} \left(\hat{J}_{N|k^c}(\mu^{0 \to a \to b \to c}) - \hat{J}^c \right) < 0 \end{array} \right\} \text{then } \rho^b = a. \tag{34}$$

Note that $\hat{J}^c$ in (34) should be understood as the present cost in the unmodified tree, that is, $\hat{J}^c := \hat{J}_{N|k^c}(\mu^{0 \to \rho^b \to b \to c})$.

6.2.1. Computational savings The step (34) requires the computation of the uncertainty cost of all trajectories $\mu^{0 \to a \to b \to c}$ obtained from the original $\mu^{0 \to \rho^b \to b \to c}$ by replacing the segment $\mu^{\rho^b \to b}$ with $\mu^{a \to b}$. Since all trajectories in the subtree at η^b are affected this could be an expensive operation. In addition, it seemingly requires that the



(a) Current tree showing a path $\mu^{0\to b\to c}$ (b) Shuffled tree by deleting $\mu^{\rho^b\to b}$ and adding $\mu^{a\to b}$

Fig. 6. A tree *shuffling* iteration: (a) a node $\eta^a \in \mathcal{T}$ has been chosen at random; another node $\eta^b \in \mathcal{T}$ is then selected with probability inversely proportional to its current uncertainty cost $\hat{J}^b$; (b) η^b is disconnected from its parent η^{ρ^b} and connected to η^a after checking that the uncertainty cost of the newly formed trajectory $\mu^{0\to a\to b\to c}$ either improves the global optimum J^* or on average improves the costs in the subtree $\mathcal{T}^b$ [see (34)].

densities (27) at node η^a be re-propagated along the complete and potentially long new trajectory segment $\mu^{a\to b\to c}$. In the SIS framework though it is not necessary to perform the whole propagation. In particular, only the incremental weight update U_{ij} along the new segment $\mu^{a\to b}$ must be computed [using (27)] and then the weights at all subtree nodes $\eta^c \in \mathcal{T}^b$ are updated directly through the simple *weight rescaling* formula

$$\bar{W}_{ij}(\mu^{0\to a\to b\to c}) = W_{ij}^c \frac{W_{ij}^a}{W_{ij}^b} U_{ij}, \tag{35}$$

where the weights W_{ij}^a are the existing weights at node η^a, respectively b and c. After computing the unnormalized weights (35) the cost $\hat{J}_{N|k^c}(\mu^{0\to a\to b\to c})$ used in the shuffling (34) is computed through (27b) and (28).

In summary, a shuffling step computes the incremental weight update along $\mu^{a\to b}$ and simply rescales the existing weights at all affected child nodes $\mathcal{T}^b$. It is summarized as

Shuffle
1. choose η^a from $\mathcal{T}$ at random
2. for $i = 1 : n_r$
3. sample $\eta^b \sim q_{\mathcal{T}}(\cdot)$
4. execute (34)
5. if $\hat{J}^c < \hat{J}^*$ then $\eta^* = \eta^c$

The number of nodes to be tried for a parent switch, n_r, during the shuffling step, can be constant but it is more reasonable to increase it as the tree becomes denser. Hence, the default choice used is $n_r = \log(\dim(\mathcal{T}))$.

6.3. Randomized pruning

Shuffling, Section 6.2, dynamically rebuilds the tree by removing and adding edges. A complementary operation can be considered, which dynamically adds and removes nodes based on their accumulated performance.

Denote the set of *leaf* nodes in a tree by $\mathcal{L}_{\mathcal{T}} \subset \mathcal{T}$, that is,

$$\mathcal{L}_{\mathcal{T}} := \{\eta^a \in \mathcal{T} \mid \nexists b \text{ s.t. } \rho^b = a\}.$$

Nodes are removed sequentially from the 'bottom' of the tree, starting with leaf nodes. The procedure is summarized as

Prune
1. for $i = 1, \ldots, n_p$
2. $\mathcal{L}' = \mathcal{L}_{\mathcal{T}} \backslash \{\eta^*, \eta^0\}$
3. $\eta \sim 1 - q_{\mathcal{L}'}(\cdot)$
4. $\mathcal{T} = \mathcal{T} \backslash \{\eta\}$

The probability of choosing nodes for pruning is inversely proportional to their fitness density (line 3). Empirically, as shown in Section 7, pruning is an effective strategy [again in the spirit of importance (re)sampling] for obtaining improved solutions more quickly. Yet, the optimal choice for a number of nodes to be pruned n_p at every iteration is difficult to determine. In our tests we prune a small fraction of the total nodes n, that is, $n_p = n/5$.

6.4. Rate of convergence

The success of the proposed algorithm depends on two key issues–whether it converges to an optimum and the rate at which it converges. A basic requirement is to obtain asymptotic convergence, that is, to reach the optimum as the number of algorithm iterations tends to infinity.

The random expansion algorithm (RND) (Section 5.1) samples points and connects them uniformly at random. This is equivalent to generating random trajectories covering the search space. The motion-planning approach ensures that the tree trajectories satisfy the dynamics and constraints. In principle, the algorithm will find a trajectory close to the optimum as the number of iterations increases. Yet, this might be an infinitely slow process in practice, a fact also confirmed by simulations in Section 7.

Utility-based importance sampling (Section 6.1) can provide a good solution more quickly by biasing trajectories to pass through more promising parts of the state space. Further work is necessary to establish non-asymptotic convergence rates by assuming a particular problem structure

and regularity conditions. Currently, the advantage of this proposed technique is observed empirically for specific problems.

The idea of shuffling (Section 6.2) is to approximate an optimum not by generating completely new trajectories (as RND does) but instead by executing local modifications to the existing tree structure. Assume that the optimal trajectory, which the algorithm aims to compute, consists of $m+1$ nodes $\eta^0, \bar{\eta}^1, \ldots, \bar{\eta}^m$. As the number of sampled nodes in the tree $\mathcal{T}$ increases there will be m nodes $\hat{\eta}^i \in \mathcal{T}$ approaching asymptotically close to each of the unknown optimal nodes $\bar{\eta}^i$, $i = 1, \ldots, m$. The problem is that there is a very small probability that all $\hat{\eta}^i$ will in fact be connected by a physical path contained in the tree, that is a path which itself is approaching the optimal path. The purpose of shuffling is then to remove and add edges in an attempt to discover this path.

Shuffling becomes less effective as the number of nodes increases since the number of pairs of old and new subtree parents to be tested for shuffling grows quadratically. The purpose of pruning (Section 6.3) is, then, to remove less promising nodes in order to reduce the search space. In essence, the size of the solution space depends on the number of nodes and on the edges connecting these nodes. Shuffling allows control over the set of edges while sampling and pruning dynamically control the set of nodes. Empirical studies of the convergence rates are studied next.

7. Numerical tests

Numerical studies based on the simple vehicle are presented first followed by a more complex helicopter search example.

7.1. Simple vehicle

The methods were tested through multiple simulation runs in the simulated scenario defined in Section 2 with the tree node connection defined in Section 5.2. Four algorithms were developed and analyzed in order to compare the proposed baseline algorithm and variance-reduction techniques. The algorithms are:

- RND: baseline random expansion algorithm (Section 5.1)
- RND+UTIL: RND augmented with utility-based sampling (Section 6.1)
- RND+UTIL+SHUFFLE: with the addition of a shuffling step (Section 6.2)
- RND+UTIL+SHUFFLE+PRUNE: the final algorithm including pruning (Section 6.3).

Figure 7 shows the resulting averaged results of the performance of each algorithm, including a comparison with a standard RRT expansion. For completeness, more detailed plots are also given in Figure 9. The simulations shows that a random search tree (RND) is better than a standard motion-planning tree for obtaining convergence to an optimal trajectory. Yet convergence is very slow. This is remedied by the combination of the proposed variance-reduction techniques. The complete algorithm RND+UTIL+SHUFFLE+PRUNE computes a solution with an acceptable performance within the allotted computation time of 60 s. Several snapshots of the dynamic tree and the resulting optimal trajectory as the algorithm progresses are shown in Figure 8. Note that all reported results are based on a global open-loop optimization with simulated future measurements rather than in a receding horizon fashion.

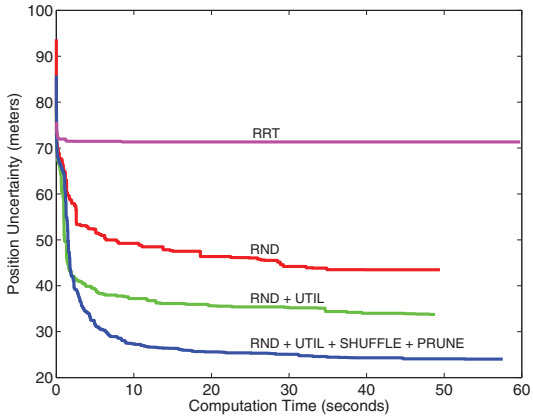


Fig. 7. Comparison of several tree-search algorithms. The y-axis plots the square root of the total variance $\sqrt{\tilde{J}^*}$, which can be interpreted as the combined error for the two-position coordinates. A standard RRT algorithm is not suitable for optimal planning since it quickly converges to and remains at a low-quality solution. The algorithm RND converges asymptotically but the rate decreases with computation time. Utility-based sampling (RND+UTIL) speeds up convergence but not drastically. The final complete algorithm, including shuffling and pruning, provides the best performance providing an acceptable error below 25 m.

Table 3. Parameters for the complete algorithm RND+UTIL+SHUFFLE+PRUNE

Parameter	Description	Default
n_i	# of nodes to be added at iteration i [i.e. $\dim(\mathcal{T}) = \sum n_i$]	10
P_U	utility-sampling probability	.5
n_r	# of nodes checked for shuffling at iteration i	$\log(\dim(\mathcal{T}))$
n_p	# of nodes to be pruned at iteration i	$\dim(\mathcal{T})/5$

Finally, it should be noted that the complete algorithm requires several parameters, which must be selected in advance. These are listed in Table 3.

The optimal values of these parameters are difficult to determine and might vary as the optimization runs. This is a problem that requires further study. The recommended default values are chosen to provide a balance between the baseline random tree that explores the state space and the variance-reduction steps to focus and speed up the search.

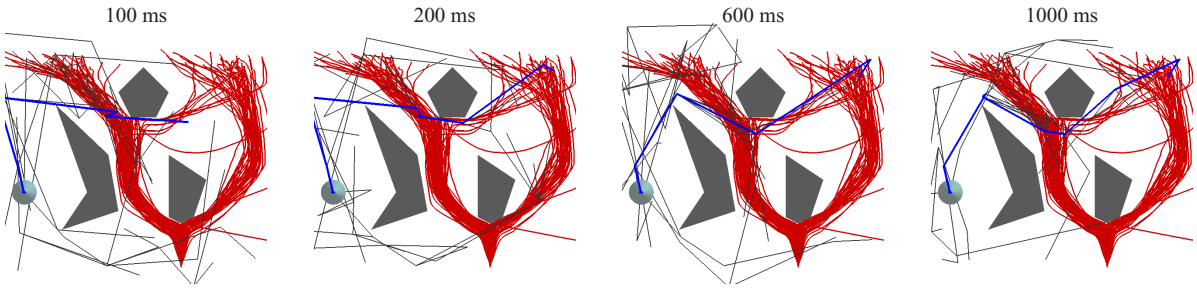


Fig. 8. Several frames showing the current tree and optimal trajectory computed by algorithm RND+UTIL+SHUFFLE+PRUNE at different computation times.

7.2. A helicopter search scenario

Consider a small autonomous helicopter as depicted in Figure 10 operating in a 3-D terrain. The vehicle is modeled as a single underactuated rigid body with position $r \in \mathbb{R}^3$ and orientation $R \in \mathrm{SO}(3)$ where $\mathrm{SO}(3)$ denotes the space of right-handed coordinate frames described by three orthogonal vectors (i.e. by a 3×3 orthogonal matrix with positive determinant). Its *body-fixed* angular and linear velocities are denoted by $\omega \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$, respectively. The vehicle has mass m and principal moments of rotational inertia J_1, J_2, J_3 forming the inertia tensor $\mathbb{J} = \mathrm{diag}(J_1, J_2, J_3)$.

The vehicle is controlled through a *collective* u_c (lift produced by the main rotor) and a *yaw* u_ψ (force produced by the rear rotor), while the direction of the lift is controlled by tilting the main blades forward or backward through a *pitch* γ_p and sideways through a *roll* γ_r. The four control inputs then consist of the two forces $u = (u_c, u_\psi)$ and the two shape variables $\gamma = (\gamma_p, \gamma_r)$. The state space of the vehicle is $\mathcal{M} = SO(3) \times \mathbb{R}^3 \times \mathbb{R}^6 \times \mathbb{R}^2$ with $\mu = ((R, p), (\omega, v), \gamma)$.

The equations of motion are

$$\begin{bmatrix} \dot{R} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} R\,\widehat{\omega} \\ R\,v \end{bmatrix}, \tag{36}$$

$$\begin{bmatrix} \mathbb{J}\,\dot{\omega} \\ m\dot{v} \end{bmatrix} = \begin{bmatrix} \mathbb{J}\,\omega \times \omega \\ mv \times \omega + R^T(0, 0, -9.81\ \mathrm{m}) \end{bmatrix} + F(\gamma)\,u, \tag{37}$$

where the map $\widehat{\cdot}: \mathbb{R}^3 \to \mathfrak{so}(3)$ is defined by

$$\widehat{\omega} = \begin{bmatrix} 0 & -\omega^3 & \omega^2 \\ \omega^3 & 0 & -\omega^1 \\ -\omega^2 & \omega^1 & 0 \end{bmatrix},$$

while the control matrix is defined as

$$F(\gamma) = \begin{bmatrix} d_t \sin\gamma_r & 0 \\ d_t \sin\gamma_p \cos\gamma_r & 0 \\ 0 & d_r \\ \sin\gamma_p \cos\gamma_r & 0 \\ -\sin\gamma_r & -1 \\ \cos\gamma_p \cos\gamma_r & 0 \end{bmatrix}.$$

The local motion-planning method corresponding to **Connect** is based on sequencing the precomputed motion primitives that satisfy the dynamics, (36) and (37). This is accomplished using a maneuver automaton described in Appendix B. In essence, the set of primitives abstracts away the complex dynamics and reduces the edge creation problem to an optimization in the discrete set of primitives and the space of translations and planar rotations—$\mathrm{SE}(2) \times \mathbb{R}^1$. A trajectory consisting of a given sequence of a minimum number of primitives can then be computed instantly in closed form through inverse kinematics. Figure 10(b) shows an example of such a sequence of primitives connected in order to exactly solve the boundary-value problem. The terrain is represented using a digital elevation map loaded from a file. Collision checking and avoidance is performed using the Proximity Query Package (PQP) (Gottschalk et al. 1996), which computes the closest distance between arbitrary polyhedra and is used to implement the function **prox** defined in (5).

The algorithm is tested using a scenario similar to that in Section 2, now extended to 3-D. The helicopter is not permitted to fly above obstacles. Figure 11 shows the resulting helicopter trajectory [see also the video in Extension 1 (Section A)], a view of the constructed road map, and a close-up along the optimal path within the road map.

8. Conclusion

This paper deals with optimal estimation for systems with non-linear dynamics subject to non-convex constraints. The approach is based on a random enumeration of trajectories generated from a tree that compactly approximates the reachable space and efficiently propagates probability distributions through recursion. The randomly sampled tree nodes approach any reachable state with exponentially (for the number of iterations) high probability and, therefore, encode a versatile road map of solution trajectories. Yet, without assuming any special structure known a priori, random search alone does not result in an efficient algorithm due to the high dimensionality of the problem. This issue is alleviated through variance-reduction techniques similar to importance sampling for stochastic optimization and to cross-over in evolutionary algorithms. While these methods show a marked improvement in solution quality and run-time efficiency, no formal non-asymptotic convergence

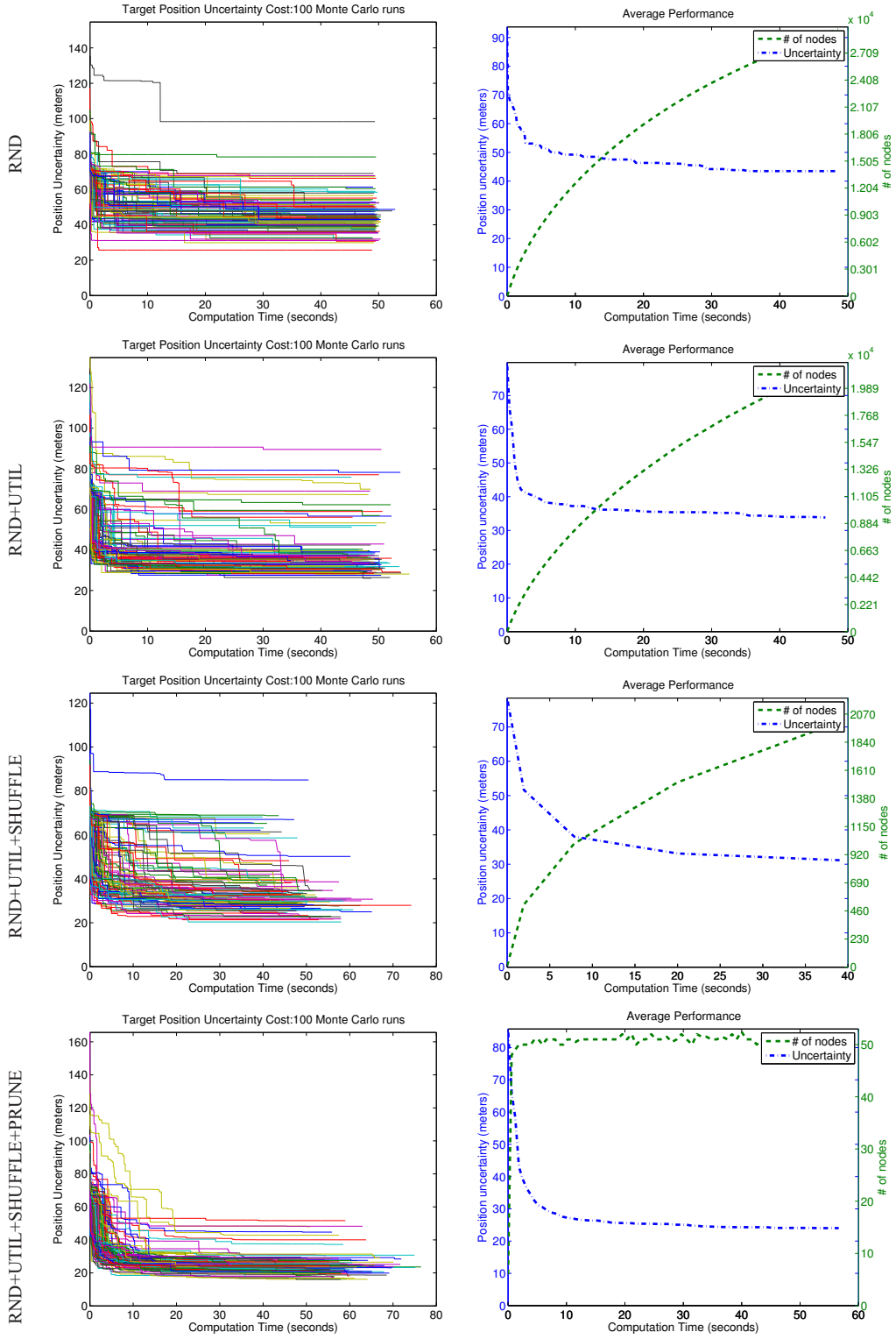


Fig. 9. Monte Carlo analysis of the the proposed techniques applied to the scenario in Section 2. The performance metrics are the resulting optimal uncertainty cost $\sqrt{\hat{J}^*}$ (which can be regarded roughly as the standard deviation of the distance to the true value) and the corresponding number of tree nodes. These two metrics are expressed in terms of the required computation time (shown on the x-axis). The left plots show all 100 Monte Carlo runs while the right plots shows the averaged results.

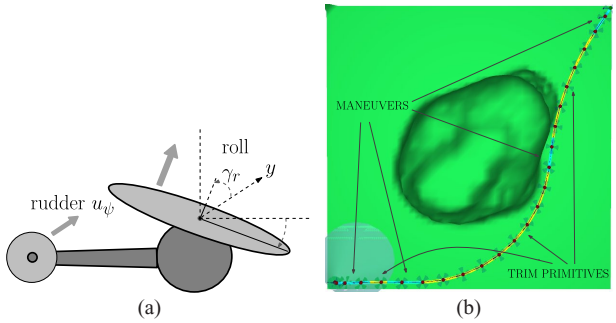


Fig. 10. (a) Simplified helicopter model used in our tests. (b) Example of a computed sequence of four maneuvers and three trim primitives, connecting two zero-velocity states in the corners of the workspace and avoiding an obstacle in the center. The trim velocities satisfy the invariance conditions defined in Section B.1 while the maneuvers are computed as outlined in Section B.2.

rates have been established. A possible future direction is to address this issue by assuming certain regularity conditions on the models involved. A related direction is to combine the proposed approach with the *cross-entropy* (CE) optimization method (Rubinstein and Kroese 2004; Celeste et al. 2007), which is designed to explicitly identify structure in the solution space by maintaining and optimally adapting an importance sampling distribution. Guiding the random tree expansion through a CE-type method would provide a consistent exploration–exploitation approach [for initial developments see Kobilarov (2011)] that optimally accounts for the sampled data during optimization. Finally, even though formally fast convergence rates are absent in our general setting, this paper provides a simple particle-based algorithm applicable to general types of dynamics and uncertainty models, which is easy to implement and performs well in practice.

Notes

1. A *state* denotes both the configuration and velocity of the vehicle (i.e. 'vehicle state') or the target (i.e. 'target state'); when used simply as 'state' its meaning will be clear from the context.
2. A uniform sample is sampled uniformly over the state space; additional sampling choices will be given in Section 6.1.
3. d is not required to be a true distance metric, i.e. it does not need to be symmetric or necessarily satisfy the triangle inequality (LaValle 2006).

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflict of interest statement

None declared.

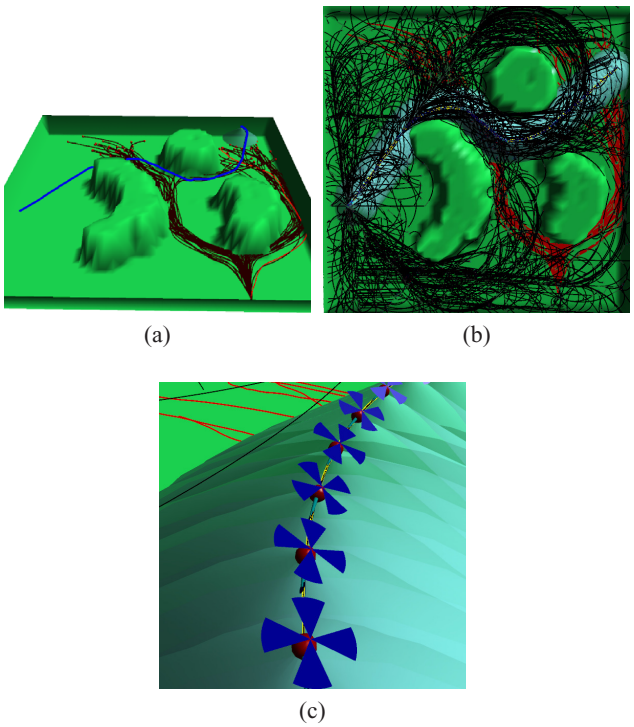


Fig. 11. The algorithm applied to the helicopter example described in Section 7.2 showing (a) the optimal helicopter trajectory; (b) the constructed road map and sensor footprint along a path; and (c) the close-up view along an edge of the road map.

Acknowledgement

The authors thank the reviewers for their comprehensive review and advice for improving this paper.

References

Bethke B, Bertuccelli LF, and How JP (2008) Experimental demonstration of adaptive MDP-based planning with model uncertainty. In: *AIAA Guidance, Navigation and Control Conference*.

Bryson M and Sukkarieh S (2009) Architectures for cooperative airborne simultaneous localisation and mapping. *Journal of Intelligent and Robotic Systems, Special Issue on Airborne SLAM*, 55: 267–297.

Burns B and Brock O (2005) Toward optimal configuration space sampling. In: *Robotics: Science and Systems*.

Celeste F, Dambreville F and Cadre J-PL (2007) Optimal path planning using cross-entropy method. In: *2006 9th International Conference on Information Fusion*, 1–8.

Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE and Thrun S (2005) *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press.

Chung TH and Burdick JW (2007) A decision-making framework for control strategies in probabilistic search. In: *Intl. Conference on Robotics and Automation*. ICRA, 4386–4393.

Clarke FH, Ledyaev YS, Stern RJ and Wolenski PR (1998) *Nonsmooth Analysis and Control Theory*. Springer.

Cole DT, Goktogan AH and Sukkarieh S (2008) The demonstration of a cooperative control architecture for UAV teams. In:

Experimental Robotics. Berlin, Heidelberg: Springer, vol. 39, 501–510.

de Geeter J, de Schutter J, Bruyninckx H, van Brussel H and Decreton M (1998) Tolerance-weighted L-optimal experiment design for active sensing. In: *Proc. Conf. IEEE/RSJ Int Intelligent Robots and Systems*, vol. 3, 1670–1675.

Del Moral P (2004) *Feynman-Kac Formulae. Genealogical and Interacting Particle Systems with Applications.* Springer-Verlag.

Doucet A, de Freitas N and Gordon N (eds) (2001) *Sequential Monte Carlo Methods in Practice*.

Erinc G and Carpin S (2007) A genetic algorithm for nonholonomic motion planning. In: *Proc. IEEE Int Robotics and Automation Conf*, 1843–1849.

Frazzoli E, Dahleh MA and Feron E (2002) Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics* 25: 116–129.

Frazzoli E, Dahleh MA and Feron E (2005) Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics* 21: 1077–1091.

Geyer C (2008) Active target search from UAVs in urban environments. In: *Proc. IEEE International Conference on Robotics and Automation*. ICRA, 2366–2371.

Gottschalk S, Lin MC and Manocha D (1996) OBBTree: A hierarchical structure for rapid interference detection. *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 30: 171–180.

Grocholsky B, Makarenko A and Durrant-Whyte H (2003) Information-theoretic coordinated control of multiple sensor platforms. In: *Proc. IEEE International Conference on Robotics and Automation*. ICRA, vol. 1, 1521–1526.

He R, Prentice S and Roy N (2008) Planning in information space for a quadrotor helicopter in a GPS-denied environments. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. ICRA, 1814–1820.

Hocaoglu C and Sanderson AC (2001) Planning multiple paths with evolutionary speciation. *IEEE Transactions on Evolutionary Computing* 5: 169–191.

Hoffmann GM and Tomlin CJ (2010) Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control* 55(1): 32–47.

Hollinger G, Singh S, and Kehagias A (2009) Efficient, guaranteed search with multi-agent teams. In: *Robotics: Science and Systems*.

Hsu D, Kindel R, Latombe J and Rock S (2002) Randomized kinodynamic motion planning with moving obstacles. *Int. J. Robotics Research* 21: 233–255.

Kobilarov M (2008) *Discrete Geometric Motion Control of Autonomous Vehicles*. PhD thesis, University of Southern California.

Kobilarov M (2011) Cross-entropy randomized motion planning. In: *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA.

Krause A and Guestrin C (2007) Nonmyopic active learning of Gaussian processes: an exploration-exploitation approach. In: *International Conference on Machine Learning (ICML)*, Corvallis, Oregon.

Ladd A and Kavraki L (2004) Measure theoretic analysis of probabilistic path planning. *IEEE Transactions on Robotics and Automation* 20: 229–242.

Langdon WB and Poli R (2001) *Foundations of Genetic Programming*. Springer.

Latombe J-C (1991) *Robot Motion Planning*. Kluwer Academic Press.

LaValle SM (2006) *Planning Algorithms*. Cambridge, UK: Cambridge University Press.

Lavis B, Furukawa T and Whyte HFD (2008) Dynamic space reconfiguration for Bayesian search and tracking with moving targets. *Auton. Robot* 24: 387–399.

Marsden J and West, M. (2001). Discrete mechanics and variational integrators. *Acta Numerica* 10: 357–514.

Martinez-Cantin R, de Freitas N, Brochu E, Castellanos J and Doucet A (2009) A Bayesian exploration–exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Auton. Robot* 27: 93–103.

Michalewicz Z and Schoenauer M (1996) Evolutionary algorithms for constrained parameter optimization problems. *Evol. Comput.* 4: 1–32.

Mihaylova L, Lefebvre T, Bruyninckx H, Gadeyne K and Schutter JD (2003a) A comparison of decision making criteria and optimization methods for active robotic sensing. In: *Lecture Notes in Computer Science*, vol. LNCS 2542, 316–324.

Mihaylova L, Schutter JD and Bruyninckx H (2003b) A multisine approach for trajectory optimization based on information gain. *Robotics and Autonomous Systems* 43: 231–243.

Paris S and Le Cadre J-P (2002) Planning for terrain-aided navigation. In: *Proc. 5th Int Information Fusion Conf.*, vol. 2, 1007–1014.

Powell W (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics.

Robert CP and Casella G (2004) *Monte Carlo Statistical Methods*. Springer.

Rubinstein RY and Kroese DP (2008) *Simulation and the Monte Carlo Method*. Wiley.

Rubinstein RY and Kroese DP (2004) *The cross-entropy method: a unified approach to combinatorial optimization*. Springer.

Ryan A (2008) Information-theoretic tracking control based on particle filter estimate. In: *AIAA Guidance, Navigation, and Control Conf.*

Sim R and Roy N (2005) Global A-optimal robot exploration in SLAM. In: *Proc. IEEE Int. Conf. Robotics and Automation*. ICRA, 661–666.

Singh SS, Kantas N, Vo B-N, Doucet A and Evans RJ (2007) Simulation-based optimal sensor scheduling with application to observer trajectory planning. *Automatica* 43: 817–830.

Stachniss C, Grisetti G and Burgard W (2005) Information gain-based exploration using Rao–Blackwellized particle filters. In: *Robotics: Science and Systems*.

Thrun S, Burgard W and Fox D (2005) *Probabilistic Robotic*. MIT Press.

Tisdale J, Kim Z and Hedrick J (2009) Autonomous UAV path planning and estimation. *IEEE Robotics & Automation Magazine* 16: 35–42.

Tremois O and Le Cadre J-P (1999) Optimal observer trajectory in bearings-only tracking for maneuvering sources. *IEEE Proceedings – Radar, Sonar and Navigation* 146: 31–39.

Vaidyanathan R, Hocaoglu C, Prince TS and Quinn RD (2001) Evolutionary path planning for autonomous air vehicles using multi-resolution path representation. In: *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, vol. 1, 69–76.

Xiao J, Michalewicz Z, Zhang L and Trojanowski K (1997) Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation* 1: 18–28.

Appendices

A. Index to multimedia extensions

The multimedia extensions to this article are at: http://www.ijrr.org.

Extension	Type	Description
1	Video	Helicopter Search Scenario

B. Helicopter primitives

The local motion-planning method is based on computing a sequence of *motion primitives* that exactly satisfy the boundary conditions, that is, one that exactly reaches a sampled node. The symmetry in the system dynamics allows us to employ a *maneuver automaton* to produce sequences of continuously parametrizable motions (trim primitives) connected with maneuvers. This general framework, developed by Frazzoli et al. (2005), is suitable for systems such as UAVs or ground robots if one ignores pose-dependent external forces, such as varying wind or ground friction that is a function of position.

Let the vehicle rotation be described by its roll ϕ, pitch θ, and yaw ψ. Denote the linear velocity by $v = (v_x, v_y, v_z) \in \mathbb{R}^3$, and the angular velocity by $\omega = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$. Denote the whole configuration by $g \in SE(3)$, the whole velocity by $\xi_b \in \mathfrak{se}(3)$, where $SE(3)$ and $\mathfrak{se}(3)$ denote the Euclidean group (translations and rotations) and its set of body-fixed velocities, respectively. The elements are defined by

$$g = \begin{bmatrix} R & r \\ \mathbf{0} & 1 \end{bmatrix}, \quad \xi_b = \begin{bmatrix} \widehat{\omega} & v \\ \mathbf{0} & 0 \end{bmatrix}. \tag{38}$$

By defining the map $\mathbb{I} = \mathrm{diag}(J_1, J_2, J_3, m, m, m)$ the dynamics can be expressed in more general form as

$$\mathbb{I}\dot{\xi}_b = \mathrm{ad}^*_{\xi_b} \mathbb{I}\,\xi_b + f_u + \mathrm{Ad}^*_g f_{\mathrm{ext}},$$

where f_u is the control force, corresponding to $F(\gamma)u$ in (37), while $f_{\mathrm{ext}} = (0, 0, 0, 0, 0, -9.81\ \mathrm{m}) \in \mathfrak{se}(3)^*$ is the gravity force. Since gravity is the only configuration-dependent term in the dynamics and is invariant for translations and rotations about the z-axis, then the dynamics symmetry group can be set as $G = SE(2) \times \mathbb{R}$.

The motion-planning problem is solved in closed form through inverse kinematics of a minimal number of primitives. A total of five is employed when moving between non-equilibrium states and an extra maneuver is added to connect from/to an equilibrium (zero velocity) state. Our particular design for the trims and maneuvers used is described next. An example sequence is shown in Figure 10.

B.1. Trim primitives

Denote the transformation corresponding to roll and pitch only by $g(\phi, \theta) \in SE(3) \backslash G$. The G-symmetry corresponds to velocity $\xi \in \mathfrak{se}(2) \times \mathbb{R}$, which corresponds to the velocity vector $(0, 0, \omega_z, v_x, v_y, v_z)$, for which $\xi_b = \mathrm{Ad}_{g(\phi,\theta)^{-1}} \xi$ is a relative equilibrium for the whole system on $SE(3)$, that is, $\dot{\xi}_b = 0$ and $g(t) = g(0)\exp(t\xi_b)$. This velocity is obtained by satisfying this invariance condition, or equivalently

$$\mathrm{ad}^*_{\xi_b} \mathbb{I}\,\xi_b + f_u + \mathrm{Ad}^*_{g(\phi,\theta)} f_{\mathrm{ext}} = 0 \tag{39}$$

since f_{ext} is G-invariant.

Condition (39) can be simplified if one assumes that the moments of rotational inertia around the y- and z-axes are identical. In this case the invariance conditions simplify to:

$$\begin{aligned} &\theta = 0, \quad u_y = 0, \quad \gamma_p = 0, \quad \gamma_r = 0, \\ &\phi = \arctan(-w_z v_x / 9.81), \\ &u_c = m(\cos\phi\ 9.81 - w_z v_x \sin\phi). \end{aligned} \tag{40}$$

In order to design trim primitives, one can pick desired velocities $(\omega_z, v_x, v_y, v_z)$ and use (40) to compute the required constant roll, pitch, and control inputs for motion along that trim. Since control inputs and shape variables have bounds, equations (40) can also be inverted to compute the maximum sustainable trim velocities.

B.2. Maneuvers

Maneuvers are computed to connect two trim motions. The parameters of a trim primitive are its roll, pitch, velocities, and shape variables. Let the map $\pi : X \to X \backslash G$ subtract out the invariant coordinates from a given state. Then given two trims, the first one ending with state x_1 and the second one starting with state x_2, we compute a maneuver trajectory x using the following optimization procedure:

Compute: $T;\ x : [0, T] \to X;\ u : [0, T] \to U$

minimizing: $J(x, u, T) = \int_0^T 1 + \lambda \|u(t)\|^2 dt$

subject to: $\pi(x(0)) = x_1, \pi(x(T)) = x_2,$
dynamics equations (36) and (37)
for all $t \in [0, T]$.

The parameter λ controls the importance of minimizing the control effort. An alternative strategy is to fix T in the above formulation and to search for the minimal T yielding a feasible control–effort optimal problem. We calculate this by using a binary search for T over the real line by solving a fixed-T optimal control problem in each iteration.

The continuous optimal control formulation is computationally solved through the discrete mechanics methodology (Marsden and West 2001), which is particularly suitable for systems with non-linear state spaces and symmetries. A geometric structure preserving the optimizer, developed by (Kobilarov 2008, Section 2.7), is used to perform the computations. The computations are performed off-line with the resulting optimal maneuvers assembled in a library offering an instant look-up during run-time.

Discrete Geometric Optimal Control on Lie Groups

Marin Kobilarov and Jerrold E. Marsden

***Abstract*—We consider the optimal control of mechanical systems on Lie groups and develop numerical methods which exploit the structure of the state space and preserve the system motion invariants. Our approach is based on a coordinate-free variational discretization of the dynamics leading to structure-preserving discrete equations of motion. We construct necessary conditions for optimal trajectories corresponding to discrete geodesics of a higher order system and develop numerical methods for their computation. The resulting algorithms are simple to implement and converge to a solution in very few iterations. A general software implementation is provided and applied to two example systems: an underactuated boat and a satellite with thrusters.**

I. Introduction

We consider the optimal control of mechanical systems evolving on a finite dimensional Lie group. Our primary motivation is the control of autonomous vehicles modeled as rigid bodies. The goal is to actuate the system to move from its current state to a desired state in an optimal way, e.g. with minimum control effort or time.

The standard way to solve such problems is to first derive the continuous nonlinear equations of motion, for example using a variational principle such as Lagrange-d'Alembert. Two general methods, termed direct and indirect, are then available to compute a minimum-cost trajectory [1]. In the direct method, the differential equations are discretized (or represented using a finite set of parameters) and enforced as algebraic constraints in a nonlinear optimization program. Such a formulation is then computationally solved using a package such as sequential quadratic programming. The indirect method is to derive necessary conditions for optimality, i.e. using Pontryagin-maximum principle by formulating another variational problem based on the original continuous equations and cost function. The necessary conditions are expressed through the evolution of additional adjoint variables satisfying a set of ordinary differential and transversality equations. These equations are then discretized and solved iteratively, e.g. using Newton's method, in order to compute an approximate numerical solution.

The framework proposed in this work uses a different computational strategy. It employs the theory of *discrete mechanics* based on a discrete variational formulation. In particular, we employ a discrete Lagrange-d'Alembert (DLA) variational principle yielding a set of discrete trajectories that approximately satisfy the dynamics and that respect the state space structure. Among these trajectories we find the extremal one without any further discretization or approximation. This is achieved by formulating a higher order variational problem to be solved through a repeated application of the DLA principle to obtain the optimal control trajectory. Such a construction enables the preservation of important properties of the mechanical system–group structure, momentum, and symplectic structure–and results in algorithms with provable accuracy and stability.

Our approach is designed for accurate and numerically stable computation and is especially suited for nonlinear systems that require iterative optimal control solvers. We construct a general optimization framework for systems on Lie groups and demonstrate its application to rigid body motion groups as well as to any real matrix subgroup. Finally, in addition to developing a computational control theory for Lie groups, we spell out the details necessary for a practical implementation.

The Lagrangian Mechanical System

We consider mechanical systems evolving on an n-dimensional Lie group G. The fundamental property of Lie groups is that each tangent vector on the manifold can be generated by translating a unique tangent vector at the identity using the group operation. More formally, each vector $\dot{g} \in T_gG$ at configuration $g \in G$ corresponds to a unique vector $\xi \in \mathfrak{g}$ through $\dot{g} = g\xi$, where $\mathfrak{g} := T_eG$ denotes the Lie algebra and $e \in G$ is the group identity.

In view of this structure the dynamics can be derived through the *reduced Lagrangian* $\ell : G \times \mathfrak{g} \to \mathbb{R}$ defined by $\ell(g, \xi) = L(g, g\xi)$ where $L : TG \to \mathbb{R}$ is the standard Lagrangian. As we will show working with the reduced state (g, ξ) has important implications for constructing numerical optimization schemes.

In this work we employ general Lagrangians of the form

$$\ell(g, \xi) = K(\xi) - V(g), \tag{1}$$

where $K : \mathfrak{g} \to \mathbb{R}$ and $V : G \to \mathbb{R}$ are given kinetic and potential energy functions. The kinetic energy Hessian $\partial^2_\xi K$ is assumed non-singular over the control problem domain.

The system is actuated using a control force $f(t) \in \mathfrak{g}^*$ defined in the body reference frame[1]. We will begin our development with fully actuated systems, i.e. such that f can act in any direction of the linear space $\mathfrak{g}^*$. We will then consider underactuated systems with control parameters, i.e. systems such that $f = \sum_{i=1}^{c} f^i(\phi)u^i$, where $f^i \in \mathfrak{g}^*$ define the allowed control directions (covectors) which depend on the controllable parameters $\phi \in \mathbb{M} \subset \mathbb{R}^m$. Here, $u \in \mathbb{U} \subset \mathbb{R}^c$ denotes the control inputs. The control input set $\mathbb{U}$ and the control shape space $\mathbb{M}$ are bounded vector spaces. The reader

Marin Kobilarov and Jerrold Marsden are with the Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA, 91125 USA (e-mail: marin@cds.caltech.edu, marsden@cds.caltech.edu).

Manuscript received June 11, 2010; revised March 29, 2011.

[1] In the Lagrangian setting a force is an element of the Lie algebra dual $\mathfrak{g}^*$, i.e. a one-form $\langle f, \cdot \rangle$ that pairs with velocity vectors to produce the total work $\int_0^T \langle f, \xi \rangle dt$ done by the force along a path between $g(0)$ and $g(T)$.

can also consult e.g. [2], [3], [4] regarding standard notation as well as more formal introduction to Lie groups and robot dynamics.

The Optimization problem

The system is required to move from a fixed initial state $(g(0), \xi(0))$ to a fixed final state $(g(T), \xi(T))$ during a time interval $[0, T]$. The problem is to find the optimal control $u^* = \operatorname{argmin}_u J(u, T)$ where the cost function J is defined by

$$J(u,T) = \frac{1}{2}\int_0^T \|u(t)\|^2 \mathrm{dt}, \tag{2}$$

subject to the dynamics and boundary state conditions.

Related work

Our approach is based on recently developed *structure-preserving* numerical integration and optimal control methods. While standard optimization methods are based on shooting, multiple shooting, or collocation techniques, recent work on Discrete Mechanics and Optimal Control (DMOC) [5] employs *variational integrators* [6], [7] that are derived from the discretization of variational principles such as Hamilton's principle for conservative systems or the Lagrange-D'Alembert principle for dissipative systems. Momentum preservation and symplecticity are automatically enforced, avoiding numerical issues (like numerical dissipation) that generic algorithms often possess.

Structure preservation plays an especially important role for systems on manifolds such as Lie groups which has led to a number of *geometric integration* methods for ordinary differential equations [8]. *Symplectic-momentum integrators* on Lie groups [9], [10] are a particular class of such methods that were combined with ideas developed in the context of *Lie group methods* [11] to construct more general and higher order integrators on Lie groups [12], [13], [14].

The optimal control problem considered in this work has a rich history both in the analytical exploration of its interesting geometric structure as well as in its numerical treatment. In particular, finding trajectories extremizing an action similar to (2) can be equivalently stated as computing geodesics for a higher-order system known as *Riemannian cubics* [15]. Riemannian cubics are generalizations of straight lines on a manifold for which, roughly speaking, the higher-order system velocity corresponds to the acceleration of the original system. When the manifold is a Lie group, such cubics can be reduced by symmetry to Lie quadratics (since the resulting curves are quadratic in the Lie algebra, while the cubics are cubic in the manifold tangent bundle). General optimality conditions as well as insightful geometric invariants have been derived (e.g. [15], [16], [17]) with particular attention to rigid body rotation problems on $\mathrm{SO}(3)$ while other works [18], [19], [20] have focused on a more practically computable approach applicable to SE(3).

Note that such works focused on the deriving optimality conditions of the two-state boundary value problem in the standard continuous setting. In contrast, we focus on developing numerical algorithms for computing high-quality solutions. Existing numerical implementations were mainly restricted to simple systems, e.g. ones possessing bi-invariant metrics or fully actuated ones.

Necessary conditions resulting from (Pontryagin's) optimality principle have been derived for simple mechanical systems [4] and for systems on Lie groups [21]. Our work reformulates these problems through a *discrete geometric framework* in order to directly obtain an algorithm for computing optimal solutions with provable numerical properties. Our approach, partially documented in [22], is based on necessary conditions formulated in a manner similar to [23], [24] which study optimal control of rigid bodies. The main difference is that our framework is applicable to any Lie group (not just the Euclidean groups) and offers greater flexibility by allowing different numerical parametrizations as well as underactuation. This generality is accomplished through the formulation of discrete necessary conditions in the spirit of the Riemannian cubics [15] employed in the continuous setting. In essence, optimal trajectories are derived as discrete geodesics of a higher-order action. Following this approach, the numerical formulation requires only the very basic ingredients–the Lagrangian, group structure, control basis, and external forces–and can automatically obtain a solution. Thus, both the regular and higher-order problems are solved using the same general discrete variational approach leading to structure-preserving dynamics and symplectic necessary conditions. Our approach is also linked to the symplectic derivation of optimal control studied in [25] to address the more generic case of an explicit discrete control system evolving in a vector space.

Finally, we point out that group structure and symmetries play an important role in robotic dynamics and motion control [26]. Variational integrators have been used in an interesting way [27] to derive the dynamics of complex multi-body systems through recursive differentiation rather than explicitly computing equations of motion. Various control methods have been developed, e.g. [28], [29], to numerically compute optimal trajectories for systems such as the snakeboard or the robotic eel. In relation to such methods, our proposed approach is unique since it builds on a unified discrete variational framework for both deriving the dynamics as well as computing the optimal controls. Note that while this work deals with systems evolving on Lie groups, it can be extended to multi-body systems with nonholonomic constraints following the construction proposed in [30].

Contributions

This paper provides a simple numerical recipe for computing optimal controls driving a mechanical system between two given boundary conditions on pose and velocity. The framework is general and can automatically generate optimal trajectories for any system on a given Lie group by providing its Lagrangian, group structure, and description of acting forces. There are several practical benefits over existing standard methods:

- the algorithm does not require choosing coordinates and avoids issues with expensive chart switching that cause sudden jumps or singularities, e.g. due to gimbal lock, that prevent convergence in iterative optimization;

- the optimization is based on minimum reduced dimension and does not require Lagrange multipliers enforcing, e.g. matrix orthogonality constraints or quaternion unit norms;
- the *discrete mechanics* approach guarantees symplectic structure and momentum preservation as well as energy approximation which is close to the true energy. The combination of these factors leads to an accurate and numerically robust approximation of the dynamics and optimality conditions as a function of the time step (formal and numerical comparisons to standard methods can be found in [7], [31], [14], [32], [30], [27]);
- predictable computation even at lower resolutions allows increased run-time efficiency.

Since the dynamics are nonlinear, an optimal solution generally does not exist in closed-form and must be computed using iterative root-finding. Numerical tests show that our proposed iterative scheme converges to a solution in surprisingly few iterations irrespective of the chosen resolution, even when applied to an underactuated system with external non-conservative forces.

Outline.

After a quick review of regular variational integrators in §II, we present in §III a formal, general treatment of the discrete variational principle used to formulate the numerical optimal control problem for systems on Lie groups. We then present the resulting optimal control algorithms first for fully actuated systems (§IV) and then for underactuated systems with control parameters (§V). The explicit expressions necessary for implementation are given in §VI for any system on the groups SE(2), SO(3), or SE(3), as well as on any general real matrix subgroup. Specific cases of a boat and a satellite are detailed as concrete examples in §VII. We also point out issues related to controllability in the underactuated case and on numerical implementation in §VIII.

II. Background on Variational Integrators

A mechanical integrator advances a dynamical system forward in time. Such numerical algorithms are typically constructed by directly discretizing the differential equations that describe the trajectory of the system, resulting in an *update rule* to compute the next state in time. In contrast, variational integrators [7] are based on the idea that the update rule for a discrete mechanical system (i.e., the time stepping scheme) should be derived *directly* from a variational principle rather than from the resulting differential equations. This concept of using a unifying principle from which the equations of motion follow (typically through the calculus of variations [33]) has been favored for decades in physics. Chief among the variational principles of mechanics is *Hamilton's principle* which states that the path $q(t)$ (with endpoint $q(t_0)$ and $q(t_1)$) taken by a mechanical system extremizes the *action integral* $\int_{t_0}^{t_1} L(q,\dot{q})dt$, i.e., the state variables $(q,\dot{q})$ evolve such that the time integral of the *Lagrangian* L of the system (equal to the kinetic minus potential energy) is extremized. A number of properties of the Lagrangian have direct consequences on the mechanical system. For instance, a *symmetry* of the system (i.e., a transformation that preserves the Lagrangian) leads to a momentum preservation.

Although this variational approach may seem more mathematically motivated than numerically relevant, integrators that respect variational properties exhibit improved numerics and remedy many practical issues in physically based simulation and animation. First, variational integrators automatically preserve (linear and angular) momenta exactly (because of the invariance of the Lagrangian with respect to translation and rotation) while providing good energy conservation over exponentially long simulation times for non-dissipative systems. Second, arbitrarily accurate integrators can be obtained through a simple change of quadrature rules. Finally, they preserve the *symplectic structure* of the system, resulting in a much-improved treatment of damping that is essentially independent of time step [31].

Practically speaking, variational integrators based on Hamilton's principle first approximate the time integral of the continuous Lagrangian by a *quadrature* rule. This is accomplished using a "discrete Lagrangian," which is a function of two consecutive states q_k and q_{k+1} (corresponding to time t_k and t_{k+1}, respectively):

$$L_d(q_k, q_{k+1}) \approx \int_{t_k}^{t_{k+1}} L(q(t),\dot{q}(t))\mathrm{d}t.$$

One can now formulate a discrete principle over the whole path $\{q_0, ..., q_N\}$ defined by the successive position at times $t_k = kh$. This discrete principle requires that

$$\delta \sum_{k=0}^{N-1} L_d(q_k, q_{k+1}) = 0,$$

where variations are taken with respect to each position q_k along the path. Thus, if we use D_i to denote the partial derivative w.r.t the i^{th} variable, we must have

$$D_2 L_d(q_{k-1}, q_k) + D_1 L_d(q_k, q_{k+1}) = 0$$

for every three consecutive positions q_{k-1}, q_k, q_{k+1} of the mechanical system. This equation thus *defines* an integration scheme which computes q_{k+1} using the two previous positions q_k and q_{k-1}.

Simple Example: Consider a continuous, typical Lagrangian of the form $L(q,\dot{q}) = \frac{1}{2}\dot{q}^T M\dot{q} - V(q)$ (V being a potential function) and define the discrete Lagrangian $L_d(q_k, q_{k+1}) = hL\big(q_{k+\frac{1}{2}}, (q_{k+1}-q_k)/h\big)$, using the notation $q_{k+\frac{1}{2}} := (q_k + q_{k+1})/2$. The resulting update equation is:

$$M\frac{q_{k+1} - 2q_k + q_{k-1}}{h^2} = -\frac{1}{2}(\nabla V(q_{k-\frac{1}{2}}) + \nabla V(q_{k+\frac{1}{2}})),$$

which is a discrete analog of Newton's law $M\ddot{q} = -\nabla V(q)$. This example can be easily generalized by replacing $q_{k+1/2}$ by $q_{k+\alpha} = (1-\alpha)\,q_k + \alpha\,q_{k+1}$ as the quadrature point used to approximate the discrete Lagrangian, leading to variants of the update equation. For controlled (i.e., non conservative) systems, forces can be added using a discrete version of Lagrange-d'Alembert principle in a similar manner [34].

III. Discrete Mechanics on Lie Groups

Variational integration on Lie groups requires additional development since a Lie group is a nonlinear space with special structure. We first recall the standard variational principle for mechanical systems on Lie groups and state the resulting differential equations of motion. A discrete structure-preserving dynamics update scheme is then constructed. It will serve as a basis for developing the proposed optimal control algorithms.

A. The Continuous Setting

The state trajectory is formally defined as $(g, \xi) : [0, T] \to G \times \mathfrak{g}$ while the control force takes the form $f : [0, T] \to \mathfrak{g}^*$. In most practical cases one can regard g as a matrix, ξ as a column vector, and f as a row vector which "pairs" with velocities through a product $\langle \cdot, \cdot \rangle$ such as the standard dot product. The Lagrange-d'Alembert principle requires that

$$\delta \int_0^T [K(\xi) - V(g)]\, \mathrm{d}t + \int_0^T \langle f, g^{-1}\delta g \rangle = 0, \tag{3}$$

where $\xi = g^{-1}\dot{g}$. The curve $\xi(t)$ describes the body-fixed velocity determined from the dynamics of the system. Variations of the velocity ξ and the configuration g are related through

$$\delta\xi = \dot{\eta} + \mathrm{ad}_\xi\, \eta, \text{ for } \eta = g^{-1}\delta g \in \mathfrak{g},$$

where the Lie bracket operator for matrix groups $\mathrm{ad} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ is defined by

$$\mathrm{ad}_\xi\, \eta = \xi\eta - \eta\xi,$$

for given $\xi, \eta \in \mathfrak{g}$. The continuous equations of motion become (see e.g. [35], [4])

$$\dot{\mu} - \mathrm{ad}^*_\xi\, \mu = -g^* \partial_g V(g) + f, \tag{4a}$$
$$\mu = \partial_\xi K(\xi), \tag{4b}$$
$$\dot{g} = g\xi. \tag{4c}$$

These equations[2] are called the controlled Euler-Poincaré equations and $\mu \in \mathfrak{g}^*$ denotes the system momentum. Given initial conditions, the momentum μ evolves according to (4a). The velocity ξ can be computed in terms of μ using (4b) since $\partial^2_\xi K$ is non-singular allowing the inversion of $\partial_\xi K$. The configuration then evolves according to (4c).

B. Trajectory Discretization

A trajectory is represented numerically using a set of $N+1$ equally spaced in time points denoted $g_{0:N} := \{g_0, ..., g_N\}$, where $g_k \approx g(kh) \in G$ and $h = T/N$ denotes the time-step. The section between each pair of points g_k and g_{k+1} is interpolated by a short curve that must lie on the manifold (Fig. 1). The simplest way to construct such a curve is through a map $\tau : \tau : \mathfrak{g} \to G$ and Lie algebra element $\xi_k \in \mathfrak{g}$ such that $\xi_k = \tau^{-1}(g_k^{-1} g_{k+1})/h$. The map is defined as follows.

Definition III.1. The *retraction map* $\tau : \mathfrak{g} \to G$ is a C^2-diffeomorphism around the origin such that $\tau(0) = e$. It is used to express small discrete changes in the group configuration through unique Lie algebra elements.

[2] $\mathrm{ad}^*_\xi\, \mu$ is defined by $\langle \mathrm{ad}^*_\xi\, \mu, \eta \rangle = \langle \mu, \mathrm{ad}_\xi\, \eta \rangle$, for some $\eta \in \mathfrak{g}$.

Thus, if ξ_k were regarded as an average velocity between g_k and g_{k+1} then τ is an approximation to the integral flow of the dynamics. The important point, from a numerical point of view, is that the difference $g_k^{-1} g_{k+1} \in G$, which is an element of a nonlinear space, can now be represented uniquely by the vector ξ_k in order to enable unconstrained optimization in the linear space $\mathfrak{g}$ for optimal control purposes.

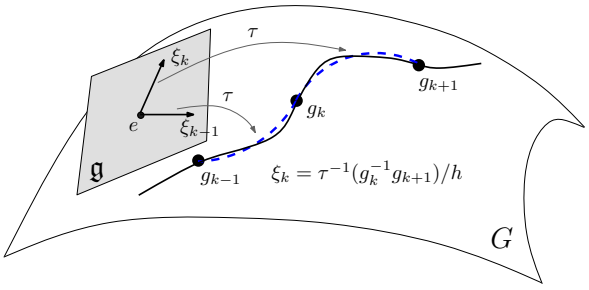


Fig. 1. A trajectory (solid) on the Lie group G discretized using a sequence of arcs (dashed) represented by Lie algebra vectors $\xi_k \in \mathfrak{g}$ through the retraction map τ.

Next, we define the following operators related to τ.

Definition III.2. [11], [14] Given a map $\tau : \mathfrak{g} \to G$, its *right-trivialized tangent* $\mathrm{d}\tau_\xi : \mathfrak{g} \to \mathfrak{g}$ and its *inverse* $\mathrm{d}\tau_\xi^{-1} : \mathfrak{g} \to \mathfrak{g}$ are such that, for a some $g = \tau(\xi) \in G$ and $\eta \in \mathfrak{g}$, the following holds

$$\partial_\xi \tau(\xi) \cdot \eta = \mathrm{d}\tau_\xi \cdot \eta \cdot \tau(\xi), \tag{5}$$
$$\partial_\xi \tau^{-1}(g) \cdot \eta = \mathrm{d}\tau_\xi^{-1} \cdot (\eta \cdot \tau(-\xi)). \tag{6}$$

Note that it can be shown by differentiating the expression $\tau^{-1}(\tau(\xi)) = \xi$ that

$$d\tau_\xi^{-1} \cdot d\tau_\xi \cdot \eta = \eta,$$

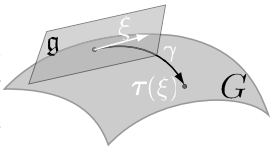
to confirm that these linear maps are indeed the inverse of each other.

Intuitively, after the derivative in the direction η is taken, i.e. $\partial_\xi \tau(\xi) \cdot \eta$, the resulting vector (at point $\tau(\xi)$) is translated back to the origin using right-multiplication by $\tau(-\xi)$ [14]. In practice, as will be shown, these maps are easily derived as $n \times n$ matrices. Finally, we require the tangent maps to be nonsingular over the optimization domain, defined next.

Definition III.3. The *optimization domain* $\mathfrak{D}_\tau \subset \mathfrak{g}$ is a connected open set containing the origin $e \in \mathfrak{g}$ such that $\mathrm{d}\tau_{h\xi}$ (and $\mathrm{d}\tau_{h\xi}^{-1}$) are non-singular for every $\xi \in \mathfrak{D}_\tau$.

The numerical algorithms proposed in the paper are restricted to operate over $\mathfrak{D}_\tau$, i.e. the time-step h and velocities ξ_k are chosen to satisfy $h\xi_k \in \mathfrak{D}_\tau$ for all $k = 0, ..., N-1$.

Retraction Map (τ) Choices: a) The *exponential map* $\exp : \mathfrak{g} \to G$, defined by $\exp(\xi) = \gamma(1)$, with $\gamma : \mathbb{R} \to G$ is the integral curve through the identity of the vector field associated with $\xi \in \mathfrak{g}$ (hence, with $\dot{\gamma}(0) = \xi$). The right-trivialized derivative of the map exp



and its inverse are defined as

$$\operatorname{dexp}_x y = \sum_{j=0}^{\infty} \frac{1}{(j+1)!} \operatorname{ad}_x^j y, \tag{7a}$$

$$\operatorname{dexp}_x^{-1} y = \sum_{j=0}^{\infty} \frac{B_j}{j!} \operatorname{ad}_x^j y, \tag{7b}$$

where B_j are the Bernoulli numbers. Typically, these expressions are truncated in order to achieve a desired order of accuracy. The first few Bernoulli numbers are $B_0 = 1$, $B_1 = -1/2$, $B_2 = 1/6$, $B_3 = 0$ (see [8]).

b) The *Cayley map* $\operatorname{cay} : \mathfrak{g} \to G$ is defined by $\operatorname{cay}(\xi) = (\mathrm{I} - \xi/2)^{-1}(\mathrm{I} + \xi/2)$ and is valid for a general class for quadratic groups that include the groups of interest in the paper. Based on this simple form, the derivative maps become ([8], §IV.8.3)

$$\operatorname{dcay}_x y = \left(e - \frac{x}{2}\right)^{-1} y \left(e + \frac{x}{2}\right)^{-1}, \tag{8a}$$

$$\operatorname{dcay}_x^{-1} y = \left(e - \frac{x}{2}\right) y \left(e + \frac{x}{2}\right). \tag{8b}$$

The third choice is to use canonical coordinates of the second kind (ccsk) [8] which are based on the exponential map and are not considered in this paper.

C. Discrete Variational Formulation

With a discrete trajectory in place we follow the approach taken in [10], [9] in order to construct a structure-preserving (i.e. group, momentum, and symplectic) integrator of the dynamics. We make a simple extension to include potential and control forces through a trapezoidal quadrature approximation. In particular, the action in (3) is approximated along each discrete segment between points g_k and g_{k+1} through

$$\int_{kh}^{(k+1)h} [K(\xi) - V(g)] \mathrm{d}t \approx h \left[K(\xi_k) - \frac{V(g_k) + V(g_{k+1})}{2} \right], \tag{9a}$$

$$\int_{kh}^{(k+1)h} \langle f, g^{-1}\delta g \rangle \approx \frac{h}{2} \left[\langle f_k, g_k^{-1}\delta g_k \rangle + \langle f_{k+1}, g_{k+1}^{-1}\delta g_{k+1} \rangle \right]. \tag{9b}$$

Variations of Lie algebra elements are related to variations on the group through the following expression which may be obtained through differentiation and application of (6),

$$\delta\xi_k = \delta\tau^{-1}(g_k^{-1}g_{k+1})/h = \mathrm{d}\tau_{h\xi_k}^{-1}(-\eta_k + \operatorname{Ad}_{\tau(h\xi_k)}\eta_{k+1})/h,$$

where $\eta_k = g_k^{-1}\delta g_k$. The operator $\operatorname{Ad}_g : \mathfrak{g} \to \mathfrak{g}$ can be regarded as a change of basis with respect to the argument $g \in G$ (see [3], [35]) and is defined by

$$\operatorname{Ad}_g \xi = g\xi g^{-1}.$$

The discrete variational principle which will form the basis for our discrete optimal control framework can now be stated. The following result is a straightforward extension from [10], [9]. The only difference is that we consider Lagrangians of the form (1) and employ a trapezoidal discretization:

Proposition 1. *A mechanical system on Lie group G with kinetic energy K, potential energy V, subject to forces f, satisfies the following* equivalent *conditions:*

1. *The discrete reduced Lagrange-d'Alembert principle holds*

$$\delta \sum_{k=0}^{N-1} \left[K(\xi_k) - \frac{V(g_k) + V(g_{k+1})}{2} \right] + \sum_{k=0}^{N-1} \frac{1}{2} \left[\langle f_k, g_k^{-1}\delta g_k \rangle + \langle f_{k+1}, g_{k+1}^{-1}\delta g_{k+1} \rangle \right] = 0, \tag{10}$$

   where $\xi_k = \tau^{-1}(g_k^{-1}g_{k+1})/h$.
2. *The discrete reduced Euler-Poincaré equations of motion hold*

$$\mu_k - \operatorname{Ad}^*_{\tau(h\xi_{k-1})}\mu_{k-1} = h\left(-g_k^* \partial_g V(g_k) + f_k\right), \tag{11a}$$

$$\mu_k = (\mathrm{d}\tau_{h\xi_k}^{-1})^* \partial_\xi K(\xi_k), \tag{11b}$$

$$g_{k+1} = g_k \tau(h\xi_k). \tag{11c}$$

Equations (11) can be considered as a discrete approximation to (4). The *discrete Euler-Poincaré* equation (11a) corresponds to (4a). Eq. (11b) is the *discrete Legendre transform* corresponding to (4b), while (11c) is the *discrete reconstruction* analogous to (4c). These equations can be used to compute the next velocity and group elements ξ_k, and g_{k+1}, respectively, given the previous elements ξ_{k-1} and g_k. Fig. 2 gives a more geometric explanation of the update (11a).

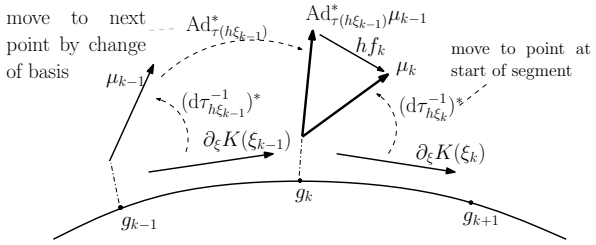


Fig. 2. The discrete covariant version of the Euler-Poincaré equation $\dot{\mu} - \operatorname{ad}^*_\xi \mu = f$, where $\mu = \partial_\xi K(\xi)$. The discrete momentum μ_k at point k is obtained using the right trivialized tangent $(\mathrm{d}\tau_{h\xi_k}^{-1})^*$ which brings the derivative $\partial_\xi K(\xi_k)$ to the body-fixed basis at g_k. The momentum evolution is then expressed through the difference of μ_{k-1} and μ_k, i.e. by transforming μ_{k-1} in that same basis at g_k through the Ad^* map where proper vector subtraction can be applied. The resulting change is caused by forces hf_k (the effects of potential V are omitted for clarity). Note that all vectors shown are elements of $\mathfrak{g}^*$ and are shown above the group configuration only to illustrate the basis with respect to which they are defined.

Boundary Conditions: While the discrete configurations g_k and forces f_k approximate their continuous counterparts at times $t = kh$, we still have not established the exact relationship between the discrete and continuous momenta, μ_k and $\mu(t) = \partial_\xi K(\xi(t))$, respectively. This is particularly important for properly enforcing boundary conditions that are given in terms of continuous quantities. The following equations (12a) and (12b) relate the momenta at the initial and final times $t = 0$ and $t = T$ and are used to transform between the continuous and discrete representation:

$$\mu_0 - \partial_\xi K(\xi(0)) = \frac{h}{2}\left(g_0^* \partial_g V(g_0) + f_0\right), \tag{12a}$$

$$\partial_\xi K(\xi(T)) - \operatorname{Ad}^*_{\tau(h\xi_{N-1})}\mu_{N-1} = \frac{h}{2}\left(g_N^* \partial_g V(g_N) + f_N\right). \tag{12b}$$

These equations can also be regarded as structure-preserving *velocity boundary conditions* for given fixed velocities $\xi(0)$

and $\xi(T)$. They follow from properly enforcing energy balance at the boundaries, achieved by adding the momentum change term $\langle \mu(T), g_N^{-1}\delta g_N\rangle - \langle \mu(0), g_0^{-1}\delta g_0\rangle$ to the discrete action in the principle (10).

The exact form of (11) and (12) depends on the choice of τ. It is important to point out that this choice will influence the computational efficiency of the optimization framework when the equalities above are enforced as constraints. We have specified two basic choices, $\tau = \exp$ (7a) and $\tau = \text{cay}$ (8a) for their ease of implementation and run-time efficiency.

D. Preservation Properties

One of the main benefits of employing the variational numerical framework lies in its preservation properties, summarized by the following theorems.

Theorem III.4. *[9] The discrete flow* (11) *preserves the discrete symplectic form, expressed in coordinates as*

$$\omega_\ell = \frac{\partial^2 \ell(g_k, \tau^{-1}(g_k^{-1}g_{k+1})/h)}{\partial g_k^i \partial g_{k+1}^j}\, \mathrm{d}g_k^i \wedge \mathrm{d}g_{k+1}^j,$$

where $\wedge$ is the standard wedge product between differential forms [35]. The symplectic form can also be written as the differential of the canonical one-form θ_ℓ with $\omega_\ell = \mathrm{d}\theta_\ell$ where

$$\theta_\ell \cdot \delta g_k = \left\langle -\frac{1}{h}\mu_k - g_k^* \partial_g V(g_k), g_k^{-1}\delta g_k \right\rangle.$$

The symplectic form is physically related to the phase space structure. Its preservation during integration, for instance, signifies that a volume of initial conditions would not be spuriously inflated or deflated due to numerical approximations. Volume preservation means that the orbits of the dynamics will have a predictable character and no artificial damping normally employed by Runge-Kutta methods is needed to stabilize the system [7].

Theorem III.5. *[9], [14] The discrete dynamics* (11) *preserves the momentum. In particular, in the absence of potential and non-conservative forces, the update scheme preserves the discrete spatial momentum map $J : G \times \mathfrak{g} \to \mathfrak{g}$,*

$$J(g_k, \xi_k) \cdot v = \mathrm{Ad}^*_{g_k^{-1}}\, \mu_k \cdot v,$$

for any $v \in \mathfrak{g}$; or equivalently $J(g_a, \xi_a) = J(g_b, \xi_b)$, for any time indices a, b.

Practically speaking, whenever the continuous system preserves momentum, so does the discrete. Any change in the momentum then exactly reflects the work done by non-conservative forces. Such a momentum-symplectic scheme also exhibits long-term stable energy behavior close to the true system energy [7]. Another property carried over to continuous case is time-scaling.

Remark 1. *Order of Accuracy.* The order of accuracy of the dynamics update depends on the accuracy of the Lagrangian approximation. Since the trapezoidal approximations (9a) and (9b) are second-order accurate then it can be shown (see [7]) that the discrete equations (11) are also of *second order accuracy*. The trapezoidal rule was chosen since it provides the simplest second-order scheme. Higher-order methods by proper choice of the Lagrangian, termed symplectic Runge-Kutta (see [8], [12], [14]), are possible but not considered in this work.

Remark 2. *Time-scaling preservation.* The trajectory $g_{0:N}$, $\xi_{0:N-1}$ with time-step h satisfies the discrete dynamics (11) subject to forces $f_{0:N}$ if and only if the trajectory $g_{0:N}$, $\{\xi_0/s, ..., \xi_{N-1}/s\}$ with time-step $h' = sh$, subject to forces $\{f_0/s^2, ..., f_N/s^2\}$ satisfies the discrete dynamics, for a given scalar $s > 0$.

Finally, the group structure is exactly preserved since the trajectory $g_{0:N}$ is reconstructed from the discrete velocity $\xi_{0:N-1}$ using the map τ which by definition maps to the group (11c). This avoids issues with dissipation and numerical drift associated with reprojection used in other methods, e.g. in methods based on matrix orthogonality constraints or quaternions.

IV. Fully Actuated Systems

We first develop the simplest case with a mechanical kinetic energy

$$K(\xi) = \frac{1}{2}\langle \mathbb{I}\xi, \xi\rangle,$$

with full unconstrained actuation, without potential or external forces and without any velocity constraints. The map $\mathbb{I} : \mathfrak{g} \to \mathfrak{g}^*$ is called the *inertia tensor* and is assumed full rank. Since there is full control over f the control effort cost function (2) can be expressed as $J(f) = \int \frac{1}{2}\|f(t)\|^2 \mathrm{d}t$. It is approximated through trapezoidal quadrature, analogously to (10), using the summation

$$J(f) \approx \sum_{k=0}^{N-1} \frac{h}{4}\left(\|f_k\|^2 + \|f_{k+1}\|^2\right). \tag{13}$$

The optimal control problem for the system (11) with given fixed initial and final states $(g(0), \xi(0))$ and $(g(T), \xi(T))$ respectively can be stated as

$$\begin{aligned}
&\textbf{Compute:} \quad \xi_{0:N-1}, f_{0:N} \\
&\textbf{minimizing} \quad \sum_{k=0}^{N-1} \frac{h}{4}\left(\|f_k\|^2 + \|f_{k+1}\|^2\right) \\
&\textbf{subject to:} \\
&\begin{cases}
\mu_0 - \mathbb{I}\,\xi(0) = (h/2) f_0, \\
\mu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})}\, \mu_{k-1} = h f_k, \quad k = 1, ..., N-1, \\
\mathbb{I}\,\xi(T) - \mathrm{Ad}^*_{\tau(h\xi_{N-1})}\, \mu_{N-1} = (h/2) f_N, \\
\mu_k = (\mathrm{d}\tau^{-1}_{h\xi_k})^*\, \mathbb{I}\,\xi_k, \\
g_0 = g(0), \\
g_{k+1} = g_k \tau(h\xi_k), \qquad\qquad k = 0, ..., N-1, \\
\tau^{-1}(g_N^{-1} g(T)) = 0.
\end{cases}
\end{aligned} \tag{14}$$

The constraints follow directly from the discrete mechanics (11), boundary conditions (12), and by noting that $\partial_\xi K = \mathbb{I}\xi$. The last equation ensures that the difference between the given and reconstructed configurations is zero.

A. Optimality Conditions

Trajectories satisfying the constrained nonlinear optimization problem (14) are computed through the derivation of optimality conditions stated in the following proposition.

Proposition 2. *The trajectory of a discrete mechanical system on a Lie group G with algebra $\mathfrak{g}$ and Lagrangian $\ell(\xi) = \frac{1}{2}\langle \mathbb{I}\,\xi, \xi\rangle$ with fixed initial and final configurations and velocities $(g(0), \xi(0)) \in G \times \mathfrak{g}$ and $(g(T), \xi(T)) \in G \times \mathfrak{g}$ minimizes the total control effort only if the discrete body-fixed velocity curve $\xi_{0:N-1}$ satisfies the following conditions:*

Necessary Conditions for Optimality

$$\nu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})}\,\nu_{k-1} = 0, \qquad k = 1, ..., N-1 \tag{15a}$$

$$\tau^{-1}(\tau(h\xi_0)\cdots\tau(h\xi_{N-1})\cdot(g(0)^{-1}g(T))^{-1}) = 0, \tag{15b}$$

where:

$$\nu_k = (\mathrm{d}\tau^{-1}_{h\xi_k})^*\partial_\xi \mathcal{K}_{(\lambda_{0:N},k)}(\xi_k), \tag{15c}$$

$$\mathcal{K}_{(\lambda_{0:N},k)}(\xi_k) = \langle (\mathrm{d}\tau^{-1}_{h\xi_k})^*\mathbb{I}\xi_k, \lambda_k - \mathrm{Ad}_{\tau(h\xi_k)}\lambda_{k+1}\rangle/h, \tag{15d}$$

$$\lambda_0^\flat = 2\left(\mu_0 - \mathbb{I}\,\xi(0)\right)/h, \tag{15e}$$

$$\lambda_k^\flat = \left(\mu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})}\,\mu_{k-1}\right)/h, \quad k = 1, ..., N-1 \tag{15f}$$

$$\lambda_N^\flat = 2\left(\mathbb{I}\,\xi(T) - \mathrm{Ad}^*_{\tau(h\xi_{N-1})}\,\mu_{N-1}\right)/h, \tag{15g}$$

$$\mu_k = (\mathrm{d}\tau^{-1}_{h\xi_k})^*\,\mathbb{I}\xi_k. \tag{15h}$$

Note: The proposition defines Nn equations (15a)-(15b) *in the Nn unknowns $\xi_0, ..., \xi_{N-1}$. A solution can be found using nonlinear root finding.*

Proof: Define the discrete action $\mathcal{S}$ according to

$$\begin{aligned}
&\mathcal{S}(\xi_{0:N-1}, f_{0:N}, \lambda_{0:N}) \\
&= \langle \mu_0 - \mathbb{I}\xi(0) - hf_0/2, \lambda_0\rangle \\
&\quad + \sum_{k=1}^{N-1}\langle \mu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})}\,\mu_{k-1} - hf_k, \lambda_k\rangle \\
&\quad + \langle \mathbb{I}\xi(T) - \mathrm{Ad}^*_{\tau(h\xi_{N-1})}\,\mu_{N-1} - hf_N/2, \lambda_N\rangle \\
&\quad + \sum_{k=0}^{N-1}\frac{h}{4}\left(\|f_k\|^2 + \|f_{k+1}\|^2\right)
\end{aligned}, \tag{16}$$

where $\mu_k = (\mathrm{d}\tau^{-1}_{h\xi_k})^*\,\mathbb{I}\xi_k$ should be regarded as a function of ξ_k. Taking variations $\delta\mathcal{S}$ with respect to f_k and λ_k we obtain[3]

$$\lambda_k^\flat = f_k = (\mu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})}\,\mu_{k-1})/h.$$

Next, freeze the adjoint trajectory $\lambda_{0:N}$ and define the functions $\mathcal{K}_{(\lambda_{0:N},k)} : \mathfrak{g} \to \mathbb{R}$, for $k = 0, ..., N-1$ by

$$\mathcal{K}_{(\lambda_{0:N},k)}(\xi) = \langle (\mathrm{d}\tau^{-1}_{h\xi})^*\,\mathbb{I}\xi, \lambda_k - \mathrm{Ad}_{\tau(h\xi)}\,\lambda_{k+1}\rangle/h. \tag{17}$$

The ξ-dependent discrete action along fixed $\lambda_{0:N}$ can be rewritten as

$$\mathcal{S}_{\lambda_{0:N}}(\xi_{0:N-1}) = h\sum_{k=0}^{N-1}\mathcal{K}_{(\lambda_{0:N},k)}(\xi_k).$$

[3]The superscript operators $\flat : \mathfrak{g} \to \mathfrak{g}^*$ (flat) and $\sharp : \mathfrak{g}^* \to \mathfrak{g}$ (sharp) are used to convert between vector fields and their duals (one-forms). Under identification $\mathfrak{g} \sim \mathbb{R}^n$, $\flat$ can simply be regarded as converting a column vector into a row vector, and $\sharp$ as the opposite operation [35].

For less cluttered notation the shorthand expression

$$\mathcal{K}(\xi_k) := \mathcal{K}_{(\lambda_{0:N},k)}(\xi_k), \tag{18}$$

will also be employed since the index k in $\mathcal{K}_{(\lambda_{0:N},k)}$ becomes clear from the argument ξ_k. The point is that λ in (17) should be regarded as fixed, i.e. not dependent on ξ. The optimality conditions can now be regarded as a set of equations satisfying the dynamics of another *higher order* discrete *Hamiltonian system* with discrete Lagrangian $\mathcal{L} = \mathcal{K}$ through

$$\delta\mathcal{S}_{\lambda_{0:N}}(\xi_{0:N-1}) = 0 \iff \nu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})}\,\nu_{k-1} = 0, \tag{19}$$

where $\nu_k = (\mathrm{d}\tau^{-1}_{h\xi_k})^*\partial_\xi\mathcal{K}(\xi_k) \in \mathfrak{g}^*$ is a momentum-like quantity for the system with Lagrangian $\mathcal{L}$. The relation (19) is nothing but the discrete Euler-Poincaré equation of this new system and was obtained in the same way the standard dynamics update (11a) followed from the principle (10). This key insight leads directly to a convenient numerical scheme for computing the optimal controls.

The final configuration g_N is computed by reconstructing the curve from the velocities $\xi_{0:N-1}$ and the boundary condition $g_N = g(T)$ is enforced through the relation (15b) without the need to optimize over any of the configurations g_k. ■

We point out the resulting formulation does not require optimizing over additional Lagrange multiplier variables. It has the minimum possible problem dimension and avoids convergence and instability issues due to improper multiplier initialization.

B. Implementing the Necessary Conditions.

An optimal trajectory is computed as the root of equations (15a)-(15b). Their exact form depends on the momentum expression (15c) which can be computed numerically using finite differences, e.g. using:

$$\begin{aligned}
&\langle \nu_k, \eta\rangle \\
&\approx \frac{1}{2\epsilon}\left[\mathcal{K}_{(\lambda_{0:N},k)}(\xi_k + \epsilon\,\mathrm{d}\tau^{-1}_{h\xi_k}\eta) - \mathcal{K}_{(\lambda_{0:N},k)}(\xi_k - \epsilon\,\mathrm{d}\tau^{-1}_{h\xi_k}\eta)\right],
\end{aligned} \tag{20}$$

along basis elements $\eta \in \mathfrak{g}$ with a small $\epsilon > 0$. In other words, the components of ν_k with respect to a chosen Lie algebra basis $\{e_i\}$ are computed according to $\nu_k^i = \langle \nu_k, e_i\rangle$ for any Lie group G.

Alternatively, the momentum can be expressed in closed form by differentiating the kinetic energy $\mathcal{K}$ to obtain

$$\begin{aligned}
&\langle \nu_k, \eta\rangle \\
&= \langle (\partial_\xi \mathrm{d}\tau^{-1}_{h\xi_k}\cdot\mathrm{d}\tau^{-1}_{h\xi_k}\eta)^*\,\mathbb{I}\,\xi_k, \Delta\lambda_k\rangle \\
&\quad + \langle (\mathrm{d}\tau^{-1}_{h\xi_k})^*\,\mathbb{I}(\mathrm{d}\tau^{-1}_{h\xi_k})\Delta\lambda_k + h\,\mathrm{ad}^*_{\mathrm{Ad}_{\tau(h\xi_k)}\lambda_{k+1}}\mu_k, \eta\rangle,
\end{aligned} \tag{21}$$

where $\Delta\lambda_k = \lambda_k - \mathrm{Ad}_{\tau(h\xi_k)}\,\lambda_{k+1}$. Expression (21) is derived using straightforward differentiation (one can also consult [22] for more details) and using A.1. One can choose to implement the necessary conditions using either (20) or (21).

V. Underactuated Systems with Control Parameters

We next extend the system dynamics to include non-trivial actuation and position dependent forces. Assume that the control forces are applied along body-fixed directions defined by

the *control covectors* $\{f^1(\phi), ..., f^c(\phi)\}$, $c \leq n$, $f^i : \mathbb{M} \to \mathfrak{g}^*$ which depend on *control parameters* $\phi : [0,T] \to \mathbb{M}$. These extra parameters can be regarded as the shape variables of the control basis, i.e. parameters that do not affect the inertial properties of the systems but which determine the control directions. Assume that the system is controlled using *control input* $u : [0,T] \to \mathbb{U}$ applied with respect to the basis $\{f^i(\phi)\}$. In addition, assume that the system is subject to *configuration-dependent forces* collectively represented by the function $f_{\text{conf}} : G \to \mathfrak{g}^*$ and dissipative *velocity-dependent forces* $f_{\text{vel}} : \mathfrak{g} \to \mathfrak{g}^*$. For instance, forces arising from the potential V take the form $f_{\text{conf}}(g) = -g^* \partial_g V(g)$. while simple viscous resistance or linear drag forces can be expressed as $f_{\text{vel}}(\xi) = -D\xi$, where D is damping positive definite map.

The total force acting in the body frame can then be expressed as the sum of the control and external forces according to

$$f(g,\xi) = \sum_{i=1}^{c} u^i f^i(\phi) + f_{\text{conf}}(g) + f_{\text{vel}}(\xi).$$

In this problem the control effort to be minimized is expressed as $\int_0^T \frac{1}{2}\|u(t)\|^2 \mathrm{d}t$.

Dissipative Force discretization: In our framework velocity-dependent forces $f_{\text{vel}}(\xi_k)$ are defined over the k-th segment, and have no clear meaning over a particular point. The contribution of such forces at a particular point can be specified by assuming the following virtual work approximation

$$\int_{kh}^{(k+1)h} f_{\text{vel}}(\xi) \cdot \eta(t) \approx \frac{h}{2} \langle (\mathrm{d}\tau_{h\xi_k}^{-1})^* f_{\text{vel}}(\xi_k), \eta_k + \mathrm{Ad}_{\tau(h\xi_N)} \eta_{k+1} \rangle,$$

where $\eta = g^{-1}\delta g$ denotes the usual Lie group variations. Such discretization is motivated by the way variations contribute to Hamilton's principle discretization (10)

$$\delta \left(\int_{kh}^{(k+1)h} \ell(\xi) \mathrm{d}t \right) \cdot \eta \approx h \langle (\mathrm{d}\tau_{h\xi_k}^{-1})^* \partial_\xi \ell(\xi_k), -\eta_k + \mathrm{Ad}_{\tau(h\xi_N)} \eta_{k+1} \rangle,$$

where the left and right variations are averaged instead of subtracted. Fig. 2 also helps explain how vectors defined along a segment transform to its start and end points.

The necessary conditions for an optimal trajectory are defined in the following proposition (which extends Prop. 2).

Proposition 3. *A discrete mechanical system with kinetic energy $K(\xi)$ and control input directions $f^i(\phi)$ subject to configuration and dissipative forces $f_{conf}(g)$ and $f_{vel}(\xi)$, respectively, moves with minimum control effort between fixed initial and final states $(g(0), \xi(0)) \in G \times \mathfrak{g}$, $((g(T), \xi(T)) \in G \times \mathfrak{g}$, only if the discrete velocity curve $\xi_{0:N-1}$, control parameters $\phi_{0:N}$, and adjoint variables $\lambda_{0:N}$ satisfy the following conditions:*

Necessary Conditions for Optimality

$$\nu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})} \nu_{k-1} = -h g_k^* \partial_g \langle f_{conf}(g_k), \lambda_k \rangle, \quad k = 1, ..., N-1 \tag{22a}$$

$$\tau^{-1}(g_N^{-1} g(T)) = 0, \tag{22b}$$

$$\mu_0 - \mathbb{I}\,\xi(0) = (h/2) f_0^+, \tag{22c}$$

$$\mu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})} \mu_{k-1} = (h/2)(f_k^- + f_k^+), \ k = 1, ..., N-1 \tag{22d}$$

$$\mathbb{I}\,\xi(T) - \mathrm{Ad}^*_{\tau(h\xi_{N-1})}\, \mu_{N-1} = (h/2) f_N^-, \tag{22e}$$

$$\sum_{i=1}^{c} u_k^i \left(\partial_\phi f^i(\phi_k)^\sharp \right) \lambda_k = 0, \qquad k = 0, ..., N \tag{22f}$$

where $\nu_k \in \mathfrak{g}^*$, $f_k^\pm \in \mathfrak{g}^*$, $u_k \in \mathbb{U}$ are defined by

$$\nu_k = (\mathrm{d}\tau_{h\xi_k}^{-1})^* \partial_\xi \mathcal{K}_{\lambda_{0:N},k}(\xi_k), \tag{22g}$$

$$\mathcal{K}_{\lambda_{0:N},k}(\xi_k) = \langle (\mathrm{d}\tau_{h\xi_k}^{-1})^* \mathbb{I}\xi_k, \lambda_k - \mathrm{Ad}_{\tau(h\xi_k)}\, \lambda_{k+1} \rangle / h - \frac{1}{2} \langle (\mathrm{d}\tau_{h\xi_k}^{-1})^* f_{vel}(\xi_k), \lambda_k + \mathrm{Ad}_{\tau(h\xi_k)} \lambda_{k+1} \rangle,$$

$$f_k^- = \sum_{i=1}^{c} u_k^i f^i(\phi_k) + f_{conf}(g_k) + (\mathrm{d}\tau_{-h\xi_{k-1}}^{-1})^* f_{vel}(\xi_{k-1}),$$

$$f_k^+ = \sum_{i=1}^{c} u_k^i f^i(\phi_k) + f_{conf}(g_k) + (\mathrm{d}\tau_{h\xi_k}^{-1})^* f_{vel}(\xi_k),$$

$$u_k^i = \langle f^i(\phi_k), \lambda_k \rangle, \tag{22h}$$

$$g_0 = g(0), \tag{22i}$$

$$g_{k+1} = g_k \tau(h\xi_k), \tag{22j}$$

Note: The proposition defines $(Nn + (N+1)n + (N+1)m)$ equations (22a)-(22f) *in the $(Nn + (N+1)n + (N+1)m)$ unknowns $(\xi_{0:N-1}, \lambda_{0:N}, \phi_{0:N})$. A solution can be found using standard nonlinear root finding. When the control basis is constant (i.e. $m = 0$) then the optimization is over $(\xi_{0:N-1}, \lambda_{0:N})$ only; $f^i(\phi_k)$ should be replaced with f^i and* (22f) *is omitted.*

Proof: Define the discrete action $\mathcal{S}$ similarly to (16) according to

$$\begin{aligned} \mathcal{S}(\xi_{0:N-1}, u_{0:N}, \phi_{0:N}, \lambda_{0:N}) = \\ &\langle \mu_0 - \mathbb{I}\xi(0) - (h/2) f_0^+, \lambda_0 \rangle \\ &+ \sum_{k=1}^{N-1} \langle \mu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})}\, \mu_{k-1} - (h/2)(f_k^- + f_k^+), \lambda_k \rangle \\ &+ \langle \mathbb{I}\xi(T) - \mathrm{Ad}^*_{\tau(h\xi_{N-1})}\, \mu_{N-1} - (h/2) f_N^-, \lambda_N \rangle \\ &+ \sum_{k=0}^{N-1} \frac{h}{4} \left(\|u_k\|^2 + \|u_{k+1}\|^2 \right), \end{aligned} \tag{23}$$

where we used the shorthand notation

$$f_k^- = \sum_{i=1}^{c} u_k^i f^i(\phi_k) + f_{\text{conf}}(g_k) + (\mathrm{d}\tau_{-h\xi_{k-1}}^{-1})^* f_{\text{vel}}(\xi_{k-1}),$$

$$f_k^+ = \sum_{i=1}^{c} u_k^i f^i(\phi_k) + f_{\text{conf}}(g_k) + (\mathrm{d}\tau_{h\xi_k}^{-1})^* f_{\text{vel}}(\xi_k).$$

Analogously to the fully actuated case (17), keep the multiplier trajectory $\lambda_{0:N}$ frozen and define the function

$\mathcal{K}_{(\lambda_{0:N},k)} : \mathfrak{g} \to \mathbb{R}$ by

$$\begin{aligned}\mathcal{K}_{(\lambda_{0:N},k)}(\xi) =& \langle (\mathrm{d}\tau_{h\xi}^{-1})^* \mathbb{I}\xi, \lambda_k - \mathrm{Ad}_{\tau(h\xi)}\, \lambda_{k+1}\rangle / h \\ &- \frac{1}{2}\langle (\mathrm{d}\tau_{h\xi}^{-1})^* f_{\text{vel}}(\xi), \lambda_k + \mathrm{Ad}_{\tau(h\xi)} \lambda_{k+1}\rangle.\end{aligned} \tag{24}$$

In addition, define the function $\mathcal{V}_{(\lambda_{0:N},k)} : G \to \mathbb{R}$ by

$$\mathcal{V}_{(\lambda_{0:N},k)}(g) = \langle f_{\text{conf}}(g), \lambda_k \rangle.$$

Similarly to (18) assume that following shorthand notation

$$\mathcal{K}(\xi_k) := \mathcal{K}_{(\lambda_{0:N},k)}(\xi_k), \qquad \mathcal{V}(g_k) := \mathcal{V}_{(\lambda_{0:N},k)}(g_k).$$

As the naming suggests, $\mathcal{K}$ and $\mathcal{V}$ play the role of kinetic and potential energies for the higher order system whose dynamics will determine the optimality conditions. The ξ-dependent part of the action (23) can be expressed, along fixed $\lambda_{0:N}$, by

$$\mathcal{S}_{\lambda_{0:N}}(\xi_{0:K}) = h \sum_{k=0}^{N-1} \left(\mathcal{K}(\xi_k) - \frac{1}{2}\left[\mathcal{V}(g_k) + \mathcal{V}(g_{k+1})\right] \right). \tag{25}$$

Note that the action (23) was expressed in terms of each $\mathcal{K}(\xi_k)$ (24) by combining all terms in $\mathcal{S}$ containing ξ_k and using the identity

$$(\mathrm{d}\tau_{-h\xi_k}^{-1})^* f_{\text{vel}}(\xi_k) = \mathrm{Ad}^*_{\tau(h\xi_k)} (\mathrm{d}\tau_{h\xi_k}^{-1})^* f_{\text{vel}}(\xi_k)$$

which follows from A.3.

After extremizing this action, it immediately follows from the general discrete Lagrange-d'Alembert principle (Prop. 1) that

$$\nu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})}\, \nu_{k-1} = -h g_k^* \partial_g \mathcal{V}(g_k), \tag{26}$$

where $\nu_k = (\mathrm{d}\tau_{h\xi_k}^{-1})^* \partial_\xi \mathcal{K}(\xi_k) \in \mathfrak{g}^*$ is a momentum-like quantity for the higher-order system with Lagrangian $\mathcal{L} = \mathcal{K} - \mathcal{V}$. In summary, the relation (22a) follows from applying the variational equations (10) to the action $\mathcal{S}_{\lambda_{0:N}}$.

Eqs. (22c)-(22e) enforce the dynamics after taking variations $\delta\lambda_k$, i.e.

$$\begin{aligned}\delta\lambda_0 \Rightarrow \qquad & \mu_0 - \mathbb{I}\,\xi(0) = (h/2) f_0^+, \\ \delta\lambda_k \Rightarrow \qquad & \mu_k - \mathrm{Ad}^*_{\tau(h\xi_{k-1})} \mu_{k-1} = (h/2)(f_k^- + f_k^+), \\ \delta\lambda_N \Rightarrow \qquad & \mathbb{I}\,\xi(T) - \mathrm{Ad}^*_{\tau(h\xi_{N-1})}\, \mu_{N-1} = (h/2) f_N^-.\end{aligned}$$

Variations of the parameters ϕ_k result in

$$\delta\phi_k \Rightarrow \qquad \left\langle \sum_{i=1}^{c} u_k^i \partial_\phi f^i(\phi_k), \lambda_k \right\rangle = 0, \quad \text{for } k = 0, ..., N,$$

which can be rewritten as the relation (22f). In the special case when the control input basis elements f^i are constant, the relation (22f) vanishes. Variations with respect to the controls δu_k result in

$$\delta u_k^i \Rightarrow \qquad -\langle f^i(\phi_k), \lambda_k \rangle + u_k^i = 0, \quad \text{for } k = 0, ..., N,$$

from which the controls u_k can be computed in terms of the multipliers (included as condition (22h)). Since the controls $u_{0:N}$ can be computed internally it is not necessary to include them as part of the optimization variables in Prop. 3.

The remaining equations are identical to the ones derived in Prop. 2. Note that the optimization is not performed over the configurations g_k, since they can be internally reconstructed according to (22i)-(22j). ■

Vector-matrix Form.: The term on the right hand side of (22a) can be better understood under the identification $\mathfrak{g} \sim \mathbb{R}^n$ by treating g as a matrix and all other variables as column vectors. In this case $g^* \partial_g \langle f_{\text{conf}}(g), \lambda \rangle = g^T (\partial_g f_{\text{conf}}(g))^T \lambda$. Similarly, the expression in (22f) should be understood as $\left(\partial_\phi f^i(\phi)^\sharp\right) \lambda = \partial_\phi f^i(\phi)^T \lambda$.

Example: constant force field.: The force (22a) has a closed form whenever the external force is constant in the global frame, i.e. when it can be written as $f_{\text{conf}}(g) = \mathrm{Ad}^*_g f_{const}$. Typical examples of such forces are gravity (on the surface of the Earth) or a simple model of wind blowing in a constant direction. Using A.1 the expression becomes

$$g^* \partial_g \langle f_{\text{conf}}(g), \lambda \rangle = -\,\mathrm{ad}^*_\lambda \mathrm{Ad}^*_g f_{\text{const}} = -\,\mathrm{ad}^*_\lambda f_{\text{conf}}(g).$$

Corollary 1. *The optimality conditions in Prop. 2 and 3 preserve the higher order discrete symplectic form $\omega_{\mathcal{L}} = \mathrm{d}\theta_{\mathcal{L}}$ where the canonical one-form $\theta_{\mathcal{L}}$ is given by*

$$\theta_{\mathcal{L}} \cdot \delta g_k = \left\langle -\frac{1}{h}\nu_k - g_k^* \partial_g \mathcal{V}(g_k), g_k^{-1} \delta g_k \right\rangle.$$

The claim follows directly from Thm. III.4.

Controllability Issues

In the fully actuated case §IV, gradient-based methods are always guaranteed to find a (local) optimum since the constraints are linearly independent. This is not the case with underactuation since controllability is generally not guaranteed. In the discrete setting, lack of controllability appears as a singularity of the optimality conditions which obstructs iterative optimization. This is an issue with any numerical method for solving optimal optimal control problems for constrained systems. In that respect our proposed approach is no better than any other standard nonlinear programming technique. Yet, there appears to be an interesting connection between the standard, i.e. continuous, controllability and its counterpart in our proposed discrete setting. This link is briefly explored next with further development left for future work.

A standard way to define controllability for the type of systems considered in this paper is through the *symmetric product*, denoted $\langle \cdot : \cdot \rangle : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ and defined by

$$\langle \xi : \eta \rangle = -\mathbb{I}^{-1} \left(\mathrm{ad}^*_\xi \mathbb{I}\eta + \mathrm{ad}^*_\eta \mathbb{I}\xi \right).$$

In the continuous setting, iterated symmetric products of the input vector fields $b^i = \mathbb{I}^{-1} f^i$ determine which velocities can be reached while iterated Lie brackets of these reachable velocities determine which configurations are achievable. In particular, exact controllability tests are directly computable assuming the system starts and ends with zero velocity [36]. A similar general claim can be made regarding our discrete setting for $N \to \infty$ since the discrete dynamics approaches the continuous one. However, such a claim is not useful in practice since a realistically implementable algorithm is based on a small N.

In that respect, there is an interesting link between the standard continuous and the required discrete controllability conditions. More specifically, the discrete dynamics (22d) can

be expressed (after setting $\tau = \exp$ and ignoring external forces) as

$$\sum_{i=0}^{\infty} \frac{B_i}{i!2^i} \left(\langle \xi_k :^i \xi_k \rangle - (-1)^i \langle \xi_{k-1} :^i \xi_{k-1} \rangle \right) = h u_i b^i \quad (27)$$

where $\langle \xi :^i \xi \rangle$ denotes taking the product using the first argument recursively i times. In addition, the reconstruction condition (22b) can be expressed through the Baker-Campbell-Hausdorff formula [4] as

$$\tau^{-1}(\tau(h\xi_0)\cdots\tau(h\xi_{N-1})) = h\sum_{k=0}^{N-1}\xi_k + \frac{h^2}{2}\sum_{i,j=0}^{N-1}[\xi_i,\xi_j] + \text{hot}, \quad (28)$$

where "hot" denotes higher-order terms of iterated Lie brackets. Note that if the closure under Lie algebra bracket operation $[\cdot,\cdot]$, denoted $\text{Lie}(\xi_{0:N-1})$, spans all possible directions of motions then (28) ensures that any final configuration $g(T)$ can be reached from any starting configuration $g(0)$. This corresponds exactly to the continuous controllability condition requiring that the Lie algebra closure of achievable velocities has full rank [4]. Note that this similarity applies in the context of *kinematic systems* since the discrete composition of flows in (28) can be regarded as a curve generated by a kinematically reduced continuous system.

It would be interesting to define more precisely the notion of *discrete controllability* through (27) and (28). This will enable the algorithm to determine not only whether a state is reachable but also an appropriate number of discrete segments N required to reach it. As a rule of thumb, any practical implementation should have $N \geq 2 + n$, where $n = \dim(G)$, to account for the two boundary conditions on velocities and to provide at least n discrete flows.

VI. Applications to Matrix Groups

We now specify the operators required to implement Prop. 2 and Prop. 3 for typical rigid body motion groups and general real matrix subgroups. While we have given more than one general choice for τ, for computational efficiency we recommend the Cayley map since it is simple and does not involve trigonometric functions. In addition, it is suitable for iterative integration and optimization problems since its derivatives do not have any singularities that might otherwise cause difficulties for gradient-based methods.

A. $SO(3)$

The group of rigid body rotations is represented by 3-by-3 matrices with orthonormal column vectors corresponding to the axes of a right-handed frame attached at the body. Define the map $\widehat{\cdot}: \mathbb{R}^3 \to \mathfrak{so}(3)$ by

$$\widehat{\omega} = \begin{bmatrix} 0 & -w_3 & w_3 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix}. \quad (29)$$

A Lie algebra basis for $SO(3)$ can be constructed as $\{\widehat{e}_1, \widehat{e}_2, \widehat{e}_3\}$, $\widehat{e}_i \in \mathfrak{so}(3)$ where $\{e_1, e_2, e_3\}$ is the standard basis for $\mathbb{R}^3$. Elements $\xi \in \mathfrak{so}(3)$ can be identified with the vector $\omega \in \mathbb{R}^3$ through $\xi = \omega^\alpha \widehat{e}_\alpha$, or $\xi = \widehat{\omega}$. Under such identification the Lie bracket coincides with the standard cross product, i.e. $\text{ad}_{\widehat{\omega}}\,\widehat{\rho} = \omega \times \rho$, for some $\rho \in \mathbb{R}^3$. Using this identification we have

$$\text{cay}(\widehat{\omega}) = \mathbf{I}_3 + \frac{4}{4 + \|\omega\|^2}\left(\widehat{\omega} + \frac{\widehat{\omega}^2}{2}\right). \quad (30)$$

The linear maps $\text{d}\tau_\xi$ and $\text{d}\tau_\xi^{-1}$ are expressed as the 3×3 matrices

$$\text{dcay}_\omega = \frac{2}{4+\|\omega\|^2}(2\mathbf{I}_3 + \widehat{\omega}), \quad \text{dcay}_\omega^{-1} = \mathbf{I}_3 - \frac{\widehat{\omega}}{2} + \frac{\omega\omega^T}{4}. \quad (31)$$

We point out that with the choice $\tau = \text{cay}$ the optimization domain is not restricted, i.e. $\mathfrak{D}_{\text{cay}} = \mathfrak{g}$ since the maps (31) are non-singular for any $\xi \in \mathfrak{g}$. This is not the case for the exponential map for which $\mathfrak{D}_{\exp} = \{\xi \in \mathfrak{g} \mid \|\xi\| < 2\pi/h\}$ since the exponential map derivative is singular whenever the norm of its argument is a multiple of 2π [8], and the origin requires special handling.

B. $SE(2)$

The coordinates of $SE(2)$ are (θ, x, y) with matrix representation $g \in SE(2)$ given by:

$$g = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix}. \quad (32)$$

Using the isomorphic map $\widehat{\cdot}: \mathbb{R}^3 \to \mathfrak{se}(2)$ given by:

$$\widehat{v} = \begin{bmatrix} 0 & -v^1 & v^2 \\ v^1 & 0 & v^3 \\ 0 & 0 & 0 \end{bmatrix} \text{ for } v = \begin{pmatrix} v^1 \\ v^2 \\ v^3 \end{pmatrix} \in \mathbb{R}^3,$$

$\{\widehat{e}_1, \widehat{e}_2, \widehat{e}_3\}$ can be used as a basis for $\mathfrak{se}(2)$, where $\{e_1, e_2, e_3\}$ is the standard basis of $\mathbb{R}^3$.

The map $\tau : \mathfrak{se}(2) \to SE(2)$ is given by

$$\text{cay}(\widehat{v}) = \begin{bmatrix} \frac{1}{4+(v^1)^2}\begin{bmatrix} (v^1)^2-4 & -4v^1 & -2v^1v^3+4v^2 \\ 4v^1 & (v^1)^2-4 & 2v^1v^2+4v^3 \end{bmatrix} \\ 0 \quad\quad 0 \quad\quad 1 \end{bmatrix},$$

while the map $[\text{d}\tau_\xi^{-1}]$ becomes the 3x3 matrix:

$$[\text{dcay}_{\widehat{v}}^{-1}] = \mathbf{I}_3 - \frac{1}{2}[\text{ad}_v] + \frac{1}{4}\begin{bmatrix} v^1 \cdot v & \mathbf{0}_{3\times 2} \end{bmatrix}, \quad (33)$$

where

$$[\text{ad}_v] = \begin{bmatrix} 0 & 0 & 0 \\ v^3 & 0 & -v^1 \\ -v^2 & v^1 & 0 \end{bmatrix}.$$

C. $SE(3)$

We make the identification $SE(3) \approx SO(3) \times \mathbb{R}^3$ using elements $R \in SO(3)$ and $x \in \mathbb{R}^3$ through

$$g = \begin{bmatrix} R & x \\ \mathbf{0} & 1 \end{bmatrix}, \qquad g^{-1} = \begin{bmatrix} R^T & -R^Tx \\ \mathbf{0} & 1 \end{bmatrix}.$$

Elements of the Lie algebra $\xi \in \mathfrak{se}(3)$ are identified with *body-fixed* angular and linear velocities denoted $\omega \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$, respectively, through

$$\xi = \begin{bmatrix} \widehat{\omega} & v \\ \mathbf{0} & 0 \end{bmatrix},$$

where the map $\widehat{\cdot}: \mathbb{R}^3 \to \mathfrak{so}(3)$ is defined in (29).

Using this identification we have

$$\tau(\xi) = \begin{bmatrix} \tau(h\widehat{\omega}_k) & h\, \mathrm{d}\tau_{h\omega_k}\, v_k \\ 0 & 1 \end{bmatrix},$$

where $\tau : \mathfrak{so}(3) \to SO(3)$ is given by (30) and $\mathrm{d}\tau_\omega : \mathbb{R}^3 \to \mathbb{R}^3$ by (31).

The matrix representation of the right-trivialized tangent inverse $\mathrm{d}\tau^{-1}_{(\omega,v)} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3 \times \mathbb{R}^3$ becomes

$$[\mathrm{dcay}^{-1}_{(\omega,v)}] = \begin{bmatrix} \mathbf{I}_3 - \frac{1}{2}\widehat{\omega} + \frac{1}{4}\omega\omega^T & \mathbf{0}_3 \\ -\frac{1}{2}\left(\mathbf{I}_3 - \frac{1}{2}\widehat{\omega}\right)\widehat{v} & \mathbf{I}_3 - \frac{1}{2}\widehat{\omega} \end{bmatrix}. \qquad (34)$$

D. General matrix subgroups

The Lie algebra of a matrix Lie group coincides with the one-parameter subgroup generators of the group. Assume that we are given a k-dimensional Lie subalgebra denoted $\mathfrak{g} \subset \mathfrak{gl}(n,\mathbb{R})$. It is isomorphic to the space of generators of a unique connected k-dimensional matrix subgroup $G \subset GL(n,\mathbb{R})$. Therefore, a subalgebra $\mathfrak{g}$ determines the subgroup G in a one-to-one fashion:

$$\mathfrak{g} \subset \mathfrak{gl}(n,\mathbb{R}) \Longleftrightarrow G \subset GL(n,\mathbb{R}).$$

The two ingredients necessary to convert the necessary conditions in Prop. (2) into algebraic equalities are: a choice of basis for $\mathfrak{g}$; and an appropriate choice of inner product (metric).

Assume that the Lie algebra basis elements are $\{E_\alpha\}_{\alpha=1}^k$, $E_\alpha \in \mathfrak{g}$, i.e. that every element $\xi \in \mathfrak{g}$ can be written as $\xi = \xi^\alpha E_\alpha$. Define the following inner product for any $\xi, \eta \in \mathfrak{g}$

$$\langle\langle \xi, \eta \rangle\rangle = \mathrm{tr}(B\xi^T\eta),$$

where B is an $n \times n$ matrix such that $\langle\langle E_\alpha, E_\beta \rangle\rangle = \delta_\alpha^\beta$ and tr is the matrix trace. Correspondingly, a pairing between any $\mu \in \mathfrak{g}^*$ and $\xi \in \mathfrak{g}$ can be defined by

$$\langle \mu, \xi \rangle = \mathrm{tr}(B\mu\xi),$$

since the dual basis for $\mathfrak{g}^*$ is $\{[E_\alpha]^T\}_{\alpha=1}^k$ in matrix form.

Example: If $\mathfrak{g} = \mathfrak{so}(3)$ then setting $B = \mathrm{diag}(1/2, 1/2, 1/2)$ yields the standard inner product under the identification $\mathfrak{so}(3 \sim \mathbb{R}^3$, i.e. $\langle \mu, \xi \rangle = \mu_\alpha \xi^\alpha$.

Example: If $\mathfrak{g} = \mathfrak{se}(3)$ with basis then setting $B = \mathrm{diag}(1/2, 1/2, 1/2, 1)$ the pairing yields the standard inner product if we identify $\mathfrak{se}(3)$ with $\mathbb{R}^3 \times \mathbb{R}^3$.

Kinetic Energy-Type Metric: After having defined a metric pairing, a kinetic energy operator $\mathbb{I}$ can be be expressed as

$$\langle \mathbb{I}(\xi), \eta \rangle = \mathrm{tr}(BI_d\xi^T\eta),$$

for some symmetric matrix $I_d \in GL(n,\mathbb{R})$.

Example: Consider a rigid body on $SO(3)$ with moments of inertia J_1, J_2, J_3 and Lagrangian $\ell(\xi) = \frac{1}{2}J_i\xi_i^2$ where the ξ_i are the velocity components in the Lie algebra basis defined in §VI-A. The matrix I_d must have the form

$$I_d = \mathrm{diag}(-J_1+J_2+J_3, -J_2+J_1+J_3, -J_3+J_1+J_2)$$

Example: Consider a rigid body on $SE(3)$ with principal moments of inertia J_1, J_2, J_3, mass m, and Lagrangian $\ell(\omega, v) = \frac{1}{2}\left(J_i\omega_i^2 + mv^Tv\right)$, where $(\omega, v) \in (\mathbb{R}^3 \times \mathbb{R}^3) \sim \mathfrak{se}(3)$ are the body-fixed angular and linear velocities using the identification defined in §VI-C. The Lagrangian in this case can be equivalently expressed as $\ell(\xi) = \frac{1}{2}tr(BI_d\xi^T\xi)$, where $\xi \in \mathfrak{se}(3)$ and

$$I_d = \mathrm{diag}(-J_1+J_2+J_3, -J_2+J_1+J_3, -J_3+J_1+J_2, m).$$

With these definitions the optimality conditions in Prop. 2 can be implemented for any given linear group by choosing B, I_d and setting the inner product to the matrix trace. For numerical efficiency though, it is always preferable to employ an identification with a vector space where a standard dot product is used.

VII. Examples

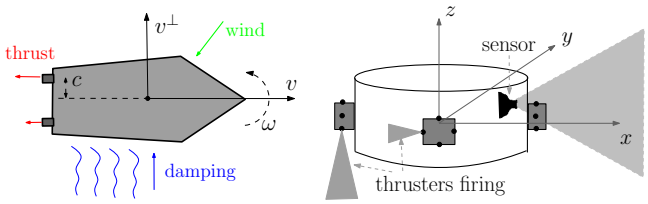


Fig. 3. Planar boat (left) controlled with two thrusters, and subject to hydrodynamic damping and wind forces. Model of a satellite (right) with 16 thrusters and a ranging sensor with limited field-of-view.

A. Planar Boat

Consider a planar boat model (Fig. 3). The configuration space of the system is the group $G = SE(2)$ with coordinates $q = (\theta, x, y)$ denoting orientation and position with respect to a fixed global frame. The body-fixed velocity $\xi \in \mathfrak{se}(2)$ is defined by

$$\xi := (\omega, v, v^\perp),$$

where ω is the angular velocity (*yaw*), v is the forward velocity (*surge*), and $v^\perp$ is the sideways velocity (*sway*). The inertia operator can be written in matrix form as $[\mathbb{I}] = \mathrm{diag}(J, m, m)$, where J is the moment of inertia around the vertical axis and m is the mass. The system is actuated with thrust produced by two fixed propellers placed at the rear of the boat at distance $\pm c$ from the long axis of the boat producing forces u_r and u_l, respectively. The control vector fields corresponding to these inputs are

$$f^1 = (c, 1, 0), \qquad f^2 = (-c, 1, 0).$$

The boat is subject to simple linear damping, commonly employed to model drag at low velocity, encoded as

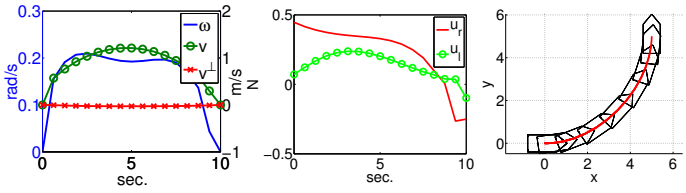
$$f_{vel}(\xi) = -D\xi,$$

where D is a positive definite matrix, and to constant (e.g. from west) wind force $f_{\text{wind}} \in \mathfrak{g}^*$ which results in the body-frame force

$$f_{conf}(g) = \mathrm{Ad}^*_g\, f_{\text{wind}}.$$

The discrete mechanics and necessary conditions for optimality are implemented using Prop. (3) by replacing the retraction map τ and its tangents with the corresponding functions defined in §VI-B. The results of three typical scenarios are given next using the parameters $J = .5$, $m = 1$ kg, $c = .2$ m., $D = \mathrm{diag}(-.5, -.5, -5)$:

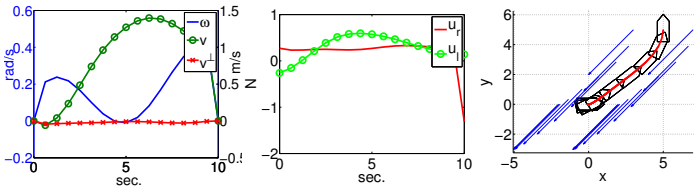
i) A basic case without wind, $f_{\text{wind}} = (0, 0, 0)$. Fig. 4 shows the resulting optimal velocities, controls, and path.



(a) (b) (c)

Fig. 4. A computed optimal trajectory between configurations $q(T) = (0, 0, 0)$ and $q(T) = (\pi/2, 5, 5)$ with zero velocities at the boundaries with $T = 10$ sec. Thruster control and simple linear damping model were used. The resulting velocities are plotted in a), controls in b), and path in c). The computation converged after six Newton iterations of the optimality conditions (Prop. (3)).

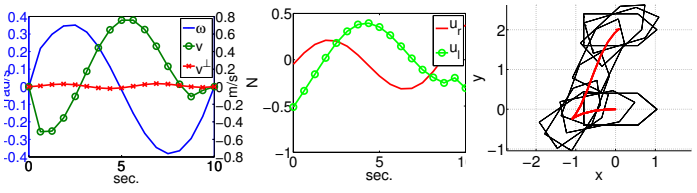
ii) Optimal motion subject to $f_{\text{wind}} = (0, -.1, -.1)$ (Fig. 5). Wind in direction opposite to the motion results in higher cost and straighter trajectory.



(a) (b) (c)

Fig. 5. The same scenario as in Fig. 4 with added configuration-dependent external force. The resulting velocities are plotted in (a), controls in (b), and path in (c). The optimization terminates successfully after nine Newton iterations.

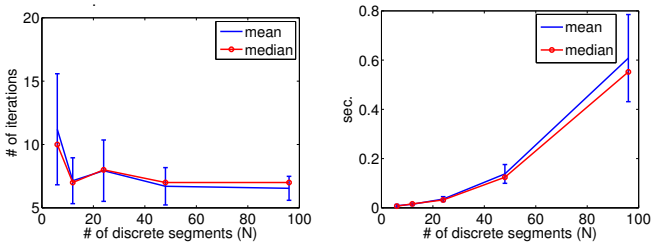
iii) Singular motion (parallel-parking) (Fig. 6). This test illustrates the ability of the algorithm to handle two typical difficulties in optimization. The first is the ability to jump out of singularities and the second is the ability to produce non-smooth optimal trajectories containing cusp points.



(a) (b) (c)

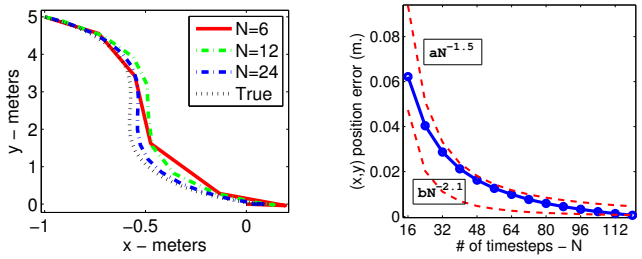
Fig. 6. A more challenging motion between $q(0) = (0, 0, 0)$ and $q(T) = (0, 0, 2)$ with zero velocities mimicking a parallel parking maneuver. The resulting velocities are plotted in (a), controls in (b), and path in (c). The algorithm can naturally compute trajectories with cusps and converges after 7 Newton iterations.

The above tests were repeated for 100 different boundary conditions chosen randomly within a 10x10 m. box and arbitrary orientations. Solutions with resolutions from $N = 6$ to $N = 96$ discrete segments were included in order to study the algorithm efficiency and robustness. Fig. 7 shows the resulting Newton iterations as a function of the time-step resolution. The algorithm is evaluated in terms of the number of iterations required for convergence and the CPU computation times (on a standard PC using C++ code). Note that our implementation is very basic, i.e. it uses finite differences and no information about Jacobian sparsity. The rate at which the discrete solutions approach the true optimum as a function of the resolution is also considered. Fig. 8 shows that the rate is close to quadratic which is consistent with the second-order accuracy of the variational method (see §III-D) used to formulate the optimal control problem. The true optimal trajectory in Fig. 8 is computed using very high resolution (N=256) and with various initial conditions in order to guarantee that it is indeed the global optimum.



(a) (b)

Fig. 7. Number of iterations required to achieve algorithm convergence as a function of the trajectory resolution N shown in (a). The corresponding computation times are shown in (b). The results are averaged over 100 Monte Carlo runs with random boundary conditions.



(a) (b)

Fig. 8. Illustration of the rate at which discrete optimal trajectories approach the true optimal trajectory. Plot a) shows three discrete trajectories of increasing resolution – even at smallest possible resolution N=6 the solution trajectory is qualitatively correct. Plot b) shows the actual convergence rate to the true optimum. The measure is the averaged distance between trajectories in position space as a function of resolution. The rate is close to quadratic, i.e. the graph is bounded by two curves decaying with exponents 1.5 and 2.1.

Optimality: In general it is not possible to claim that any of the solution trajectories are globally optimal. Yet, it is interesting to point out that through the coarse initialization and resolution upsampling (described in §VIII) all computed trajectories were indeed globally optimal (based on comparisons with 100 other randomly chosen initial trajectories $\xi_{0:N-1}$ to seed the iterative solver).

B. Satellite with Thrusters

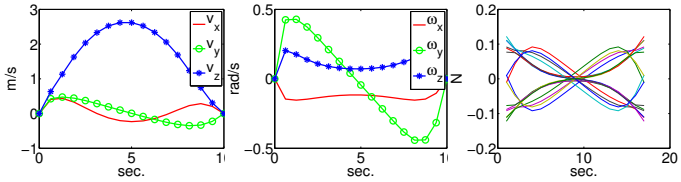
Consider a satellite (Fig. 3) modeled as a rigid body with configuration space $G = SE(3)$ describing its orientation and

position (defined in §VI-C). The system has mass m and principal moments of inertia J_1, J_2, J_3 forming the inertia tensor $\mathbb{I} = \mathrm{diag}(J_1, J_2, J_3, m, m, m)$.

The craft is controlled with forces produced by 16 thrusters placed at distance r from the craft central axis. The total force f can be expressed in terms of the controls $u \in \mathbb{R}^{16}$ in matrix-vector form as $f = Fu$, where the constant matrix F with columns corresponding to the input vector fields f^i has the form

$$F := \begin{bmatrix} 0 & 0 & 0 & 0 & r & 0 & -r & 0 & 0 & 0 & 0 & 0 & -r & 0 & r & 0 \\ r & 0 & -r & 0 & 0 & 0 & 0 & 0 & -r & 0 & r & 0 & 0 & 0 & 0 & 0 \\ 0 & -r & 0 & r & 0 & -r & 0 & r & 0 & -r & 0 & r & 0 & -r & 0 & r \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \end{bmatrix}.$$

The optimal control algorithm is implemented using Prop. 3 based on the Cayley map and its derivatives on SE(3) defined in §VI-C. Fig. 9 shows a typical control scenario. The resulting motion is visualized in Fig. 10.



(a) (b) (c)

Fig. 9. A computed optimal path between the origin and configuration $q(T) = (\pi/2, 5, 5)$ with zero velocities at the boundaries with $T = 10$ sec. The resulting angular velocities, linear velocities, and control curves of the 16 thrusters are plotted in a), b), and c), respectively. The computation converged after nine Newton iterations of the optimality conditions (Prop. (3)).

VIII. Numerical Implementation

Software Package: The presented algorithms along with a library with all Lie group operations used in the paper are implemented and assembled as a Matlab package available at http://www.cds.caltech.edu/~marin/index.php?n=lieopt

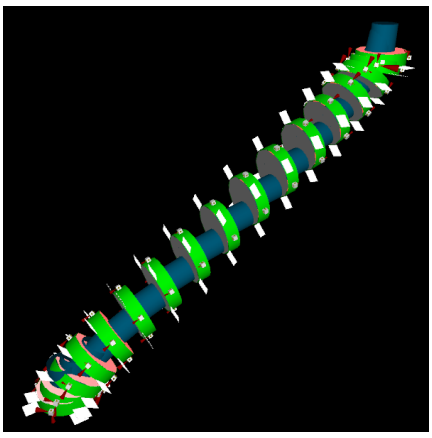


Fig. 10. An optimal trajectory between two given zero-velocity states of the satellite. Thruster outputs are rendered as small red cones emanating from the four boxes around the spacecraft.

It can be applied to new models by specifying their group structure (currently supporting SE(2), SO(3), and SE(3)), inertial properties, control vector fields, and external forces. The example results from §VII are included for easier reference.

Trajectory Initialization and Resolution: Since there is no established strategy for selecting an optimal resolution N, our approach is to start the optimal control computation with some minimum N_0, e.g. enough to satisfy the dynamics. The resolution is then increased by upsampling the trajectory (resulting in $N = 2N_0$ segments) and re-optimizing the new finer trajectory. The process can be repeated as many times as necessary to achieve a desired resolution. Interestingly, Fig. 7 shows that such an approach effectively makes the number of required Newton iterations independent and even decreasing as N increases. In our numerical tests we do not include exact CPU run-times taken which can vary based on implementation but instead analyze the number of iterations required for convergence. In practice, for a reasonable N, the whole process can be implemented in near real-time (e.g. using optimized C-code instead of Matlab).

Singularities: In the underactuated case there are a small set of states which result in singularities of the optimality conditions (Prop. 3). For instance, Fig. 6 illustrates a parallel parking task for which $\Delta g = \tau^{-1}(g(0)^{-1}g(T))$ is perpendicular to the control directions f. A trajectory $\xi_{0:N-1}$ such that ξ_k is parallel to Δg for all k will render the optimality conditions singular. A standard Newton step in this case will fail. The easiest way to overcome this situation, implemented in our system, is to detect the singularity and perturb the trajectory as simply as $\xi_k = \xi_k + \epsilon\,\mathrm{randn}(n)$ for one or more k and a small variation, e.g. $\epsilon = 10^{-3}$. This approach is a simplification of the procedure used by more sophisticated homotopy-continuation methods to detect and handle bifurcations [37] (in our case the split is because the parallel displacement can be achieved equally well by either first moving forward and then backwards, or vice versa).

Real Vehicle Implementation: The run-time efficiency results obtained in §VII suggest that the proposed algorithm is suitable for real-time maneuver control of vehicles such as the boat shown on Figure 3. In particular, Figure 7 shows that a reasonably accurate trajectory (e.g. one with $N = 24$ segments as depicted on Figure 8) can be computed in less than 50 milliseconds with basic unoptimized C++ code. In addition, the expected number of iterations and CPU time are very predictable and the algorithms never failed to converge in the performed 100 random runs. Ultimately, the method can be used to optimally drive a vehicle from its current state to a given state $(g(T), \xi(T))$ in a given time T. Once the algorithm computes the discrete control sequence $u_{0:N}$, the continuous curve u is reconstructed using linear interpolation. The vehicle is then controlled using actuator inputs $u(t)$ at time $t \in [0, T]$. The process can be repeated if the vehicle deviates from its path due to uncertainties.

IX. Conclusion

This paper shows how recent developments in the theory of discrete mechanics and Lie group methods can be used to

construct numerical optimal control algorithms with certain desirable features. Preservation of key motion properties leads to robust dynamics approximation. In addition, a singularity-free structure-respecting choice of trajectory representation avoids numerical instability during iterative optimization.

Practically speaking, the message of our approach is that a reliable numerical optimization of vehicle motions on Lie groups (such as a robot modeled as a rigid body) can be accomplished by selecting a coordinate-free and singularity-free trajectory parametrization providing high accuracy and stability at low resolution and complexity. There are existing standard methods which address some of the raised issues. Our approach is to circumvent any numerical problems through a proper design of a general discrete variational framework.

It is necessary to study the precise effect of the discretization resolution N on the optimality of the algorithm and to explore the notion of *discrete controllability*. Future work will address such issues and attempt to apply tools from standard nonlinear controllability to provide formal numerical convergence guarantees in the underactuated case.

APPENDIX A
TANGENT MAP IDENTITIES

The following identities supplement the derivations in the paper.

Lemma A.1 (see [35])**.** *Let $g \in G$, $\lambda \in \mathfrak{g}$, and δf denote the variation of a function f with respect to its parameters. Assuming λ is constant, the following identity holds*

$$\delta\left(\mathrm{Ad}_g\,\lambda\right) = -\,\mathrm{Ad}_g[\lambda, g^{-1}\delta g],$$

where $[\cdot,\cdot]:\mathfrak{g}\times\mathfrak{g}\to\mathfrak{g}$ denotes the Lie bracket operation or equivalently $[\xi,\eta]\equiv \mathrm{ad}_\xi\,\eta$, for given $\eta,\xi\in\mathfrak{g}$.

Lemma A.2. *The following identity holds*

$$\partial_\xi\left(\mathrm{Ad}_{\tau(\xi)}\,\lambda\right) = -[\mathrm{Ad}_{\tau(\xi)}\,\lambda, \mathrm{d}\tau_\xi]$$

Proof: By Lemma A.1

$$\begin{aligned}\partial_\xi\left(\mathrm{Ad}_{\tau(\xi)}\,\eta\right) &= -\,\mathrm{Ad}_{\tau(\xi)}[\lambda, \tau(-\xi)\delta\tau(\xi)]\\ &= -[\mathrm{Ad}_{\tau(\xi)}\,\lambda, \delta\tau(\xi)\cdot\tau(-\xi)]\\ &= -[\mathrm{Ad}_{\tau(\xi)}\,\eta, \mathrm{d}\tau_\xi],\end{aligned}$$

obtained from the tangent definition (5) *and using the fact that* $\mathrm{Ad}_g[\lambda,\eta] = [\mathrm{Ad}_g\,\lambda, \mathrm{Ad}_g\,\eta]$ *(see [35]).* ■

Lemma A.3 (see [14])**.** *The following identities holds*

$$\mathrm{d}\tau_\xi\,\eta = \mathrm{Ad}_{\tau(\xi)}\,\mathrm{d}\tau_{-\xi}\,\eta, \qquad (\mathrm{d}\tau_\xi^{-1})\eta = \mathrm{d}\tau_{-\xi}^{-1}\left(\mathrm{Ad}_{\tau(-\xi)}\,\eta\right).$$

ACKNOWLEDGEMENT

We thank L. Noakes and D. M. de Diego for interesting discussions on closely related topics, and M. Desbrun, G. Johnson, and the paper reviewers for their useful feedback regarding this paper.

REFERENCES

[1] J. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.
[2] J. M. Selig, *Geometrical Foundations of Robotics*. World Scientific Pub Co Inc, 2000.
[3] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC, 1994.
[4] F. Bullo and A. Lewis, *Geometric Control of Mechanical Systems*. Springer, 2004.
[5] O. Junge, J. Marsden, and S. Ober-Blöbaum, "Discrete mechanics and optimal control," in *Proccedings of the 16th IFAC World Congress*, 2005.
[6] J. Moser and A. P. Veselov, "Discrete versions of some classical integrable systems and factorization of matrix polynomials," *Comm. Math. Phys.*, vol. 139, no. 2, pp. 217–243, 1991.
[7] J. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, vol. 10, pp. 357–514, 2001.
[8] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration*, ser. Springer Series in Computational Mathematics. Springer-Verlag, 2006, no. 31.
[9] J. E. Marsden, S. Pekarsky, and S. Shkoller, "Discrete euler-poincare and Lie-poisson equations," *Nonlinearity*, vol. 12, p. 16471662, 1999.
[10] A. I. Bobenko and Y. B. Suris, "Discrete lagrangian reduction, discrete euler-poincare equations, and semidirect products," *Letters in Mathematical Physics*, vol. 49, p. 79, 1999.
[11] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna, "Lie group methods," *Acta Numerica*, vol. 9, pp. 215–365, 2000.
[12] M. Leok, "Foundations of computational geometric mechanics," Ph.D. dissertation, California Institute of Technology, 2004.
[13] P. Krysl and L. Endres, "Explicit newmark/verlet algorithm for time integration of the rotational dynamics of rigid bodies," *International Journal for Numerical Methods in Engineering*, 2005.
[14] N. Bou-Rabee and J. Marsden, "Hamilton-pontryagin integrators on Lie groups," *Foundations of Computational Mathematics*, vol. 9, pp. 197–219, 2009.
[15] L. Noakes, "Null cubics and Lie quadratics," *Journal of Mathematical Physics*, vol. 44, no. 3, pp. 1436–1448, 2003.
[16] M. Camarinha, F. S. Leite, and P. Crouch, "On the geometry of Riemannian cubic polynomials," *Differential Geometry and its Applications*, no. 15, pp. 107–135, 2001.
[17] R. Giambo, F. Giannoni, and P. Piccionez, "Optimal control on Riemannian manifolds by interpolation," *Math. Control Signals System*, vol. 16, pp. 278–296, 2003.
[18] M. Zefran, V. Kumar, and C. B. Croke, "On the generation of smooth three-dimensional rigid body motions," *IEEE Transactions On Robotics And Automation*, vol. 14, no. 4, pp. 576–589, 1998.
[19] C. Altafini, "Reduction by group symmetry of second order variational problems on a semidirect product of Lie groups with positive definite Riemannian metric," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 10, pp. 526–548, 2004.
[20] R. V. Iyer, R. Holsapple, and D. Doman, "Optimal control problems on parallelizable riemannian manifolds: Theory and applications," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 12, pp. 1–11, 2006.
[21] A. Bloch, *Nonholonomic Mechanics and Control*. Springer, 2003.
[22] M. Kobilarov, *Discrete Geometric Motion Control of Autonomous Vehicles*. PhD thesis, University of Southern California, 2008.
[23] T. Lee, N. McClamroch, and M. Leok, "Optimal control of a rigid body using geometrically exact computations on SE(3)," in *Proc. IEEE Conf. on Decision and Control*, 2006.
[24] A. M. Bloch, I. I. Hussein, M. Leok, and A. K. Sanyal, "Geometric structure-preserving optimal control of a rigid body," *Journal of Dynamical and Control Systems*, vol. 15, no. 3, pp. 307–330, 2009.
[25] M. de Leon, D. M. de Diego, and A. Santamaria Merino, "Geometric numerical integration of nonholonomic systems and optimal control problems," *European Journal of Control*, vol. 10, pp. 520–526, 2004.
[26] J. Ostrowski, "Computing reduced equations for robotic systems with constraints and symmetries," *IEEE Transactions on Robotics and Automation*, pp. 111–123, 1999.
[27] E. Johnson and T. Murphey, "Scalable variational integrators for constrained mechanical systems in generalized coordinates," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1249 – 1261, 2009.
[28] J. P. Ostrowski, J. P. Desai, and V. Kumar, "Optimal gait selection for nonholonomic locomotion systems," *The International Journal of Robotics Research*, vol. 19, no. 3, pp. 225–237, 2000.

[29] J. Cortés, S. Martinez, J. P. Ostrowski, and K. A. McIsaac, "Optimal gaits for dynamic robotic locomotion," *The International Journal of Robotics Research*, vol. 20, no. 9, pp. 707–728, 2001.
[30] M. Kobilarov, J. E. Marsden, and G. S. Sukhatme, "Geometric discretization of nonholonomic systems with symmetries," *Discrete and Continuous Dynamical Systems - Series S (DCDS-S)*, vol. 3, no. 1, pp. 61 – 84, 2010.
[31] L. Kharevych, Weiwei, Y. Tong, E. Kanso, J. Marsden, P. Schroder, and M. Desbrun, "Geometric, variational integrators for computer animation," in *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2006, pp. 1–9.
[32] M. Kobilarov, K. Crane, and M. Desbrun, "Lie group integrators for animation and control of vehicles," *ACM Trans. Graph.*, vol. 28, no. 2, pp. 1–14, 2009.
[33] C. Lanczos, *Variational Principles of Mechanics*. University of Toronto Press, 1949.
[34] A. Stern and M. Desbrun, "Discrete geometric mechanics for variational time integrators," in *ACM SIGGRAPH Course Notes: Discrete Differential Geometry*, 2006, pp. 75–80.
[35] J. E. Marsden and T. S. Ratiu, *Introduction to Mechanics and Symmetry*. Springer, 1999.
[36] F. Bullo, N. Leonard, and A. Lewis, "Controllability and motion algorithms for underactuated lagrangian systems on Lie groups," *IEEE Transactions on Automatic Control*, vol. 45, no. 8, pp. 1437 – 1454, 2000.
[37] E. Allgower and K. Georg, *Introduction to Numerical Continuation Methods*. SIAM Wiley and Sons, 2003.

Marin B. Kobilarov is a post-doctoral fellow in Control and Dynamical Systems at Caltech and is affiliated with the Keck Institute for Space Studies. His research focuses on computational control methods that exploit the geometric structure of nonlinear dynamics. He develops autonomous vehicles with applications in robotics and aerospace.

Jerrold E. Marsden is a professor of Control and Dynamical Systems at Caltech. He has done extensive research in the area of geometric mechanics, with applications to rigid body systems, fluid mechanics, elasticity theory, plasma physics, as well as to general field theory. His work in dynamical systems and control theory emphasizes how it relates to mechanical systems and systems with symmetry, along with concrete application areas of dynamical systems and optimal control, including Lagrangian Coherent Structures (LCS), space systems, and structured integration methods. He is one of the original founders in the early 1970's of reduction theory for mechanical systems with symmetry, which remains an active and much studied area of research today.