Trajectory Planning for Cubesat Short Time-Scale Proximity Operations

Marin Kobilarov[Note1]
Johns Hopkins University, Baltimore, MD 21218

Sergio Pellegrino[Note2]
California Institute of Technology, Pasadena, CA 91125

This paper considers motion planning for small satellites such as cubesats performing proximity operations in a several meters range 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 obstacle avoidance constraints. The proposed approach is based on constructing a reduced-order parametrization 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.

[Note1] Assistant Professor of Mechanical Engineering, Laboratory for Computational Sensing and Robotics, 117 Hackerman Hall, 3400 N. Charles Street. e-mail: marin@jhu.edu

[Note2] Joyce and Kent Kresa Professor of Aeronautics and Professor of Civil Engineering, Graduate Aerospace Laboratories, 1200 E. California Blvd. MC 301-46. AIAA Fellow. e-mail: sergiop@caltech.edu

NOMENCLATURE

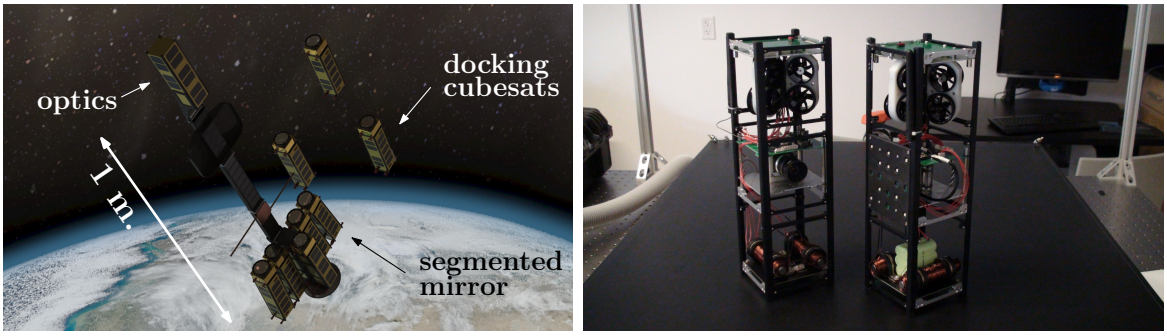
a	= T10 thruster system lateral offsets
$A \subset \mathbb{R}^3$	= region occupied by robot
$\boldsymbol{b} \in \mathbb{R}^3, \beta$	= axis and angle of rotation during trajectory generation
$\boldsymbol{b}, \boldsymbol{b}_4 : [0, \Delta] \to \mathbb{R}^3$	= spline basis functions
B	= control transformation matrix
$\boldsymbol{c} \in \mathbb{R}^3$	= field-of-view cone axis
$\boldsymbol{c}_2, \boldsymbol{c}_3 \in \mathbb{R}^3$	= rotational spline coefficients
$C : S \times U \to \mathbb{R}$	= cost function
C_i	= splines coefficient matrices
d_x, d_y, d_z	= spacecraft dimensions
$\boldsymbol{e}_c \in \mathbb{R}^3$	= sensor axis
$\exp : \mathbb{R}^3 \to SO(3)$	= exponential map
$\boldsymbol{e}_i \in \mathbb{R}^3$	= thruster axis
$\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z \in \mathbb{R}^3$	= standard basis unit vectors
$\boldsymbol{f} : S \times U \to \mathbb{R}^3$, $\boldsymbol{f}_{\text{ext}} : S \to \mathbb{R}^3$	= control and external forces
$F_i : S \times U \to \mathbb{R}$	= constraint functions
h	= T10 thruster system axial offsets
$\mathbb{J} = (J_x, J_y, J_z)$	= moments of inertia
K	= number of mixture components
$\log : SO(3) \to \mathbb{R}^3$	= exponential map
m	= mass, number of waypoints
n_z	= dimensionality of $\mathcal{Z}$
$\mathcal{O}_i \subset \mathbb{R}^3$	= environment obstacles
$p(Z; \boldsymbol{v})$	= parametric density over $\mathcal{Z}$
$\mathbf{prox} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$	= closest distance function
$\boldsymbol{q}_i = (\boldsymbol{x}_i, R_i)$	= waypoint in configuration space
$\boldsymbol{r} : [0, \Delta] \to \mathbb{R}^3$	= interpolating curve in exponential coordinates
$R \in SO(3)$	= rotation matrix in RSW frame
$r_x, r_y, r_z > 0$	= VACCO thrusters offsets
$\boldsymbol{s} \in S = SE(3) \times \mathbb{R}^6$	= complete state
$S_g \subset S$	= goal set
$T > 0$	= planning horizon
$\boldsymbol{u} \in U$	= control inputs
$v_{\max} > 0$	= velocity bound
$\boldsymbol{v} \in \mathcal{V}$	= density parameter
w_k	= mixture weight of each component of p
$\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3$	= position in RSW frame
$\boldsymbol{z} \in \mathcal{Z}$	= trajectory parameter
Z	= random variable over $\mathcal{Z}$
$\mathcal{Z}_{\text{con}}$	= constrained trajectory parameter space

α	= VACCO thrusters angles
$\boldsymbol{\alpha} \in \mathbb{R}^3$	= desired thrust direction
$\boldsymbol{\beta}$	= field-of-view cone angle
$\Delta = T/(m+1)$	= time duration of each segment between waypoints
λ	= cost function weighing factor
$\boldsymbol{\mu}_k$	= mean of each component of p
$\boldsymbol{\rho} \in \mathbb{R}^3$	= exponential coordinates for aligning R and R_d
Σ_k	= variances of components of p
$\boldsymbol{\tau} : S \times U \to \mathbb{R}^3$, $\boldsymbol{\tau}_{\text{ext}} : S \to \mathbb{R}^3$	= control and external torques
$\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$	= angular velocity in body frame
$\omega_c > 0$	= orbit angular rate

I. Introduction

Autonomous navigation and operation in the proximity of multiple objects in space is becoming increasingly important as 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 have 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.

As an example scenario, consider the *Autonomous Assembly of a Reconfigurable Space Telescope* (AAReST) mission concept [1] illustrated in Figure 1. This mission is intended for a near-circular



a) b)

Fig. 1 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.

orbit at 650 km altitude and Table 1 lists the orbit details. The mission relies on the ability of individual cubesats to autonomously undock, orbit around a spacecraft cluster, and redock to a different position. Note that under ideal conditions such reconfigurations are accomplished by exploiting relative orbital dynamics forces and 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 (Table 1). This corresponds to motions no faster than 2 mm/s that can only be achieved with very precise boundary conditions. Such precision is presently difficult to achieve with small spacecraft that perform docking/undocking or manipulating another object in space. For instance, the electromagnetic docking system in development for AAReST is expected to generate initial velocities upon undocking of up to 2 cm/s. More generally, it is realistic to assume that relative velocities between interacting bodies are 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. Furthermore, low-cost spacecraft such as cubesats are currently capable of carrying a limited number of thrusters which imposes further restrictions on trajectory planning.

Motivated by the need for algorithms applicable to shorter time scales and constrained under-

Parameter	Value	Units	Notes
(d_x, d_y, d_z)	$(0.1, 0.1, 0.3)$	m	
$\mathbb{J}$	$(0.03, 0.03, 0.015)$	$\mathrm{kg} \cdot \mathrm{m}^2$	
m	3	kg	
Altitude	650	km	
Inclination	97.985823	deg.	
Orbit period	97.728221	min	
ω_c	.00107154	rad/s	
Velocity	7530.93	m/s	
Relative Perturbations	$\leq 1.9712 \times 10^{-4}$	N	relative $\|v\| \leq 2$cm/s, range ≤ 2 m.
Gravity Gradient	$\leq 2.5830 \times 10^{-8}$	Nm	max at 45°
Solar Torque	$\leq 2.7360 \times 10^{-9}$	Nm	center-of-mass $\pm[0.02, 0.02, 0.02]$ m
Solar Force	$\leq 1.3680 \times 10^{-7}$	N	at max cross area
Drag Torque	$\leq 3.3166 \times 10^{-13}$	Nm	center-of-mass $\pm[0.02, 0.02, 0.02]$ m
Drag Force	$\leq 1.6583 \times 10^{-11}$	N	at max cross area

Table 1 AAReST details.

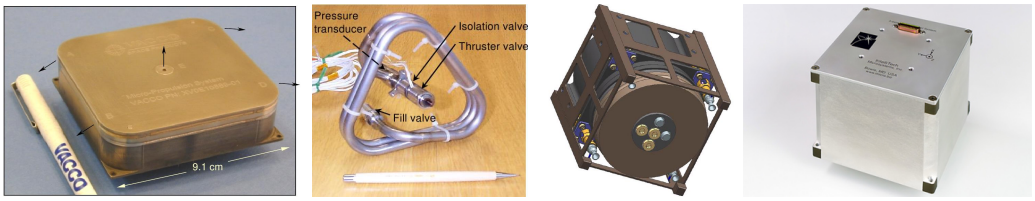
actuated propulsion we develop techniques for designing approximately optimal trajectories that account for rigid body dynamics, orbital dynamics, underactuation, and environment constraints. The optimal reconfiguration problem can be regarded as a constrained nonlinear optimal control problem. The general 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 any-time stochastic global optimization over this reduced space. The case of fully-actuated and under-actuated systems is handled in a unified way through either standard 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.

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 [2, 3]. In this context, convex programming [4, 5] or mixed-integer linear programming [6] have 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 [7, 8]. The final stage of rendezvous has also been studied through a simple linear model [9]. The problem has also been extensively studied in the context of planned future missions such as Terrestrial Planet Finder (TPF) [10] and low cost spacecraft testbeds such as the Spheres [11–13]. The Spheres testbed has also served as a basis for a conceptual mission [14] sharing common aspects with AAReST.

A cubesat with a single thruster and ACS can navigate to a given position by simply orienting and accelerating and decelerating in a straight line. This strategy becomes very inefficient for more complicated trajectory, e.g. one that avoids obstacles and satisfies other constraints. Instead, the differential flatness [15–17] property of the spacecraft should be exploited for following any desired trajectory by smoothly varying the attitude and controlling the thruster along the whole motion. Flatness is an intrinsic dynamical system property which can be used to transform an underactuated system into a system evolving in a fully-controllable flat output space. Flatness has been used for spacecraft control purposes but limited primarily to attitude control problems (see e.g. [18]). An unrelated but interesting application of flatness for handling avoidance constraints appears in [19]. Other methods for handling complex dynamics and underactuation include reducing the dynamics to a driftless system through decoupling vector fields [20] or exploiting symmetries to encode the dynamics through a maneuver automaton [21]. Decoupling vector fields are more closely related to the straight line motion strategy and require the system to stop at various points along the motion which leads to suboptimal performance. Employing a maneuver automaton relies on the invariance of trajectories, e.g. with respect to a constant gravity vector, and does not directly apply to the case with coupled orbital and underactuated rigid body dynamics during proximity operations.

This papers does not assume a particular propulsion model and considers even the case when the spacecraft has only a single thruster aligned with the center of mass. Its main goal is to develop a principled approach for dealing with various actuation capabilities given the current state of nanosatellite propulsion methods (see [22] for a survey). Existing technologies such as cold gas and



a) b) c) d)

Fig. 2 Cold-gas propulsion systems and 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

pulsed plasma thrusters are directly applicable to cubesats today as long as they can be miniaturized and properly packaged. An example in this context is the MEPSI (Micro-Electromechanical-based Picosat Satellite Inspection Experiment)-type spacecraft developed at the Aerospace Corporation in their innovative use of rapid prototyping for compact component integration [23] and of smart materials such as photostructurable glass/ceramic [24]. While there are a number of recent developments in this direction cold-gas systems such as VACCO's five thrusters MEMS system [25], Aerojet's CubeSat Hydrazine Adaptable Monopropellant Propulsion System (CHAMPS)[26], and SSC Nanospace MEMS Propulsion module [27] should be mentioned as examples of promising technology for the near future. In addition, new technologies such as ion thrusters (see e.g. [28]) and electrospray systems (e.g. [29]) result in higher efficiency and ΔV. 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.

The specific novel contributions of this work are:

a. motion planning for a class of underactuated spacecraft among obstacles,

b. combining differential flatness techniques [17] and stochastic optimization [34] to achieve close to real-time performance

c. experimental demonstration of a cubesat mock-up planning scenario on an air table.

The developed system is based on estimating the vehicle state, performing real-time trajectory optimization, and executing the resulting controls directly without the need for closed-loop trajectory

tracking. While this strategy has proven successful in a simplified laboratory setting, the paper does not formally consider the important issue of robustness to sensing and model uncertainty which are critical for realistic applications.

The paper is laid out as follows. Section II formulates the trajectory planning problem in terms of the system dynamics, constraints, and objective function. The process of generating feasible trajectories is outlined in §III, while §IV describes how to parametrize them in numerically convenient form. Section V then describes the stochastic optimization approach for computing an approximately globally optimal solution. The proposed algorithm is illustrated using computed examples in §VI. Finally, an experimental study using two cubesats performing relative navigation and docking in a laboratory setting is described in §VII.

II. 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 frame attached to the central body (target), such as an RSW frame [30] where the convention is that the x-axis is on-track and the z-axis points away from Earth. 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 $\mathbb{J} = \mathrm{diag}(J_x, J_y, J_z)$. 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 $\boldsymbol{\tau}_{\mathrm{ext}} : S \to \mathbb{R}^3$ and $\boldsymbol{f}_{\mathrm{ext}} : S \to \mathbb{R}^3$, respectively. Significant external forces and torques in LEO are due to relative dynamics and gravity gradients (see Table 1). For near-circular orbit the forces are simplified using the Hill-Clohessy-Wiltshire (HCW) linearized model, i.e.

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

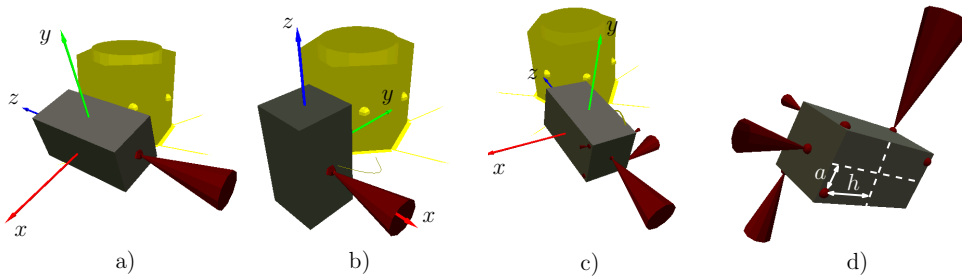


Fig. 3 Four thruster configuration studied (cones represent thruster plumes proportional to the total thrust integrated over a short sampling period): a) single thruster along z-axis (ACS+Z); b) two opposite thrusters along x-axis (ACS±X) with only one thruster firing; c) five-thruster system (VACCO) also shown in Figure 2a; d) conceptual 10-thruster system (T10).

with constant $\omega_c > 0$ denoting the circular orbit angular rate (see Table 1). The external torques due to gravity gradients are defined according to

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

where $\boldsymbol{e}_z = (0, 0, 1)$. Note that in the context of very close proximity operations such as in AAReST (Figure 1) 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

$$\widehat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{4}$$

Four basic thruster models covering a range of configurations of practical interest are detailed next. The first configuration is currently being considered for several mission implementations; the

second represents a minor increase in complexity; the third is an example of a fully-integrated, multi-thruster system that has some flight heritage; and the fourth is the ideal configuration, although currently unavailable.

1. Full attitude control and one thruster along body-fixed 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 3a shows a model of this configuration while actual hardware that could be employed to realize this system is shown in Figure 2. 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 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

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

where $\boldsymbol{e}_z = (0, 0, 1) \in \mathbb{R}^3$. 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 angular velocity achievable by the ACS.

2. Full attitude control and two opposite thrusters along body-fixed x-axis (ACS±X) Consider a cubesat with an ADC and two opposing thrusters along the x-axis aligned with the center of mass (see Figure 3b). 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

$$\boldsymbol{\tau}(\boldsymbol{s}, \boldsymbol{u}) = (u_1, u_2, u_3), \quad \boldsymbol{f}(\boldsymbol{s}, \boldsymbol{u}) = u_4 R \boldsymbol{e}_x, \tag{6}$$

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

3. VACCO system without ACS. Consider the micro-thruster system [25] shown in Figure 2a and the model shown in Figure 3c. Since no ACS is present in this system the control forces can be

expressed as

$$\begin{bmatrix} \boldsymbol{\tau}(\boldsymbol{s},\boldsymbol{u}) \\ \boldsymbol{f}(\boldsymbol{s},\boldsymbol{u}) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix} B\boldsymbol{u}, \tag{7}$$

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

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

using the notation $c\alpha \equiv \cos\alpha,\ s\alpha \equiv \sin\alpha$ and where $r_x, r_y, r_z > 0$ define the offsets of the VACCO thruster valves from the center of mass in the x, y, and z axes respectively, while $\alpha = 15^o$ denotes the angle with which the four lateral thrusters are pointing along 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 rows 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.

4. Fully actuated ten 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 3d and can be defined similarly to (7) but with

$$B = \begin{bmatrix} 0 & -h & 0 & h & h & 0 & -h & 0 & 0 & 0 \\ h & 0 & -h & 0 & 0 & h & 0 & -h & 0 & 0 \\ a & a & a & a & -a & -a & -a & -a & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}, \tag{8}$$

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

The spacecraft 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

$$F_i(\boldsymbol{s}(t), \boldsymbol{u}(t)) \geq 0 \tag{9}$$

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

$$F_1(\boldsymbol{s}, \boldsymbol{u}) = v_{\max} - \|\dot{\boldsymbol{x}}\|. \tag{10}$$

Obstacle constraints require 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 (9) can be written as

$$F_2(\boldsymbol{s}, \boldsymbol{u}) = \min_i \mathbf{prox}(\mathcal{A}(R, \boldsymbol{x}), \mathcal{O}_i), \text{ for all } t \in [0, T]. \tag{11}$$

We use a standard collision checking algorithm implemented by Proximity Query Package (PQP) [31] to compute **prox**. *Field-of-View (FOV) constraints* require the spacecraft to point within a given sensing cone and are expressed as

$$F_3(\boldsymbol{s}, \boldsymbol{u}) = \boldsymbol{c}^T R \boldsymbol{e}_c - \cos\beta, \tag{12}$$

where $\boldsymbol{c} \in \mathbb{R}^3$ is a unit vector parallel to the cone axis, $\beta \in [0, \pi]$ is the cone angle, and $\boldsymbol{e}_c$ is a unit vector defined with respect to the spacecraft body frame. Typically, $\boldsymbol{e}_c$ is aligned with the axis of a sensor such as a camera with field of view β.

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.

$$\begin{aligned} (\boldsymbol{u}^*, T^*) &= \arg\min_{\boldsymbol{u}, T} \int_0^T C(\boldsymbol{s}(t), \boldsymbol{u}(t))\mathrm{dt}, \\ &\text{s.t. } \boldsymbol{s}(T^*) \in S_g \text{ while satisfying dynamics (1) – (3) and constraints (9),} \end{aligned} \tag{13}$$

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 $\lambda \geq 0$.

Finally, note that the employed continuous thrust model can also be applied to impulsive thrusters through Pulse-Width-Modulation (PWM) at the resolution of their minimum impulse bit and pulse duration.

The nonlinear optimization problem (13) 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 long run-times. This precludes real-time performance and direct implementation on-board the spacecraft. Our approach is to simplify the problem by decomposing it into :

1. local *trajectory generation* between intermediate states called *waypoints* through differential flatness,

2. global *constrained optimization* of sequences of waypoints in order to compute an obstacle-free trajectory for any given desired goal set.

III. Open-loop Trajectory Generation

The task of trajectory generation is to compute a control signal to generate a given feasible trajectory. Any given trajectory of a fully actuated system (e.g. T10 introduced in §II) can be inverted to compute the required control inputs and is feasible as long as the inputs are not saturated. Similarly, underactuated systems (e.g. ACS+Z and ACS±X systems introduced in §II) can be handled by exploiting the *differential flatness* property of the control system.

A. 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. 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. [17] for an in-depth treatment) this work only considers a particular model – a spacecraft modeled as a 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 §II 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 determine 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 (x, y, z, ψ), where ψ is the yaw of the body, are the flat outputs. 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 an attitude given by a rotation matrix R_d. This is also the approach taken in [32] 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

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

The right-trivialized tangent [33] map $\mathrm{dexp}: \mathbb{R}^3 \to L(\mathbb{R}^3; \mathbb{R}^3)$ is such that

$$\partial \exp(\boldsymbol{\omega}) \cdot \delta \cdot \exp(\boldsymbol{\omega}) = \widehat{\mathrm{dexp}(\boldsymbol{\omega})\delta} \tag{15}$$

and is defined by

$$\mathrm{dexp}(\boldsymbol{\omega}) = \begin{cases} \mathbf{I}_3, & \text{if } \boldsymbol{\omega} = 0 \\ \mathbf{I}_3 + \left(\frac{1-\cos\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|}\right)\frac{\widehat{\boldsymbol{\omega}}}{\|\boldsymbol{\omega}\|} + \left(1 - \frac{\sin\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|}\right)\frac{\widehat{\boldsymbol{\omega}}^2}{\|\boldsymbol{\omega}\|^2}, & \text{if } \boldsymbol{\omega} \neq 0 \end{cases} \tag{16}$$

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:*

$$\boldsymbol{u}_{1:3} = \boldsymbol{\omega} \times \mathbb{J}\boldsymbol{\omega} + \mathbb{J}\dot{\boldsymbol{\omega}} - \boldsymbol{\tau}_{ext}, \tag{17}$$

$$u_4 = \boldsymbol{\alpha}^T R\boldsymbol{e}_i, \tag{18}$$

$$\boldsymbol{\omega} = \mathrm{dexp}(-\boldsymbol{\rho})\dot{\boldsymbol{\rho}} + \exp(-\boldsymbol{\rho})\boldsymbol{\omega}_d, \tag{19}$$

$$\boldsymbol{\alpha} = m\ddot{\boldsymbol{x}}_d - \boldsymbol{f}_{\text{ext}}, \quad \boldsymbol{b} = (R_d e_i) \times \boldsymbol{\alpha}/\|\boldsymbol{\alpha}\|, \quad \beta = \cos\left((R_d\boldsymbol{e}_i)^T\boldsymbol{\alpha}/\|\boldsymbol{\alpha}\|\right) \tag{20}$$

$$\boldsymbol{\rho} = \begin{cases} \arg\min_{\left\{\beta\frac{\boldsymbol{b}}{\|\boldsymbol{b}\|},(\beta-\pi)\frac{\boldsymbol{b}}{\|\boldsymbol{b}\|}\right\}} \|\boldsymbol{u}_{1:3}\| & \text{if } \|\boldsymbol{b}\| > 0, \\ \boldsymbol{0}, & \text{if } \|\boldsymbol{b}\| = 0, \end{cases} \tag{21}$$

$$R = R_d \exp(\boldsymbol{\rho}), \tag{22}$$

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 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 straightforward 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. (21)). 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 computed by noting that

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

using the definition (15). Using the relationships $R\widehat{\boldsymbol{\omega}}R^T = \widehat{R\boldsymbol{\omega}}$ and $\exp(-\boldsymbol{\rho})\widehat{\mathrm{dexp}(\boldsymbol{\rho})}\boldsymbol{\delta}\exp(\boldsymbol{\rho}) = \widehat{\mathrm{dexp}(-\boldsymbol{\rho})}\boldsymbol{\delta}$ (see e.g. [39]) this is reduced to

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

Finally, assuming full attitude control the required attitude $R(t)$ is achieved by setting $\boldsymbol{u}_{1:3} = \boldsymbol{\omega}\times\mathbb{J}\boldsymbol{\omega}+\mathbb{J}\frac{d}{dt}\boldsymbol{\omega}-\boldsymbol{\tau}_{\mathrm{ext}}$. In practice, the term $\frac{d}{dt}\boldsymbol{\omega}$ can be implemented using finite difference approximation, i.e. as $(\boldsymbol{\omega}-\boldsymbol{\omega}')/h$ by keeping track of the previously computed $\boldsymbol{\omega}'$ in (19) assuming the control law operates at time-step h. □

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. (17) and (19)). The strategy employed in this work is to rescale the time along a given $\boldsymbol{x}_d(t)$ in order to satisfy the control bounds.

B. Fully Actuated System

Assume that 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 $\mathrm{rank}(B)=6$ and $\{B\boldsymbol{u}\mid\boldsymbol{u}\in U\}\subset\mathbb{R}^6$ 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

$$\boldsymbol{u} = (B^TB)^{-1}B^T\begin{bmatrix}\boldsymbol{\omega}\times\mathbb{J}\boldsymbol{\omega}+\mathbb{J}\dot{\boldsymbol{\omega}}-\boldsymbol{\tau}_{\mathrm{ext}}\\ R^T(\ddot{\boldsymbol{x}}-\boldsymbol{f}_{\mathrm{ext}})\end{bmatrix}. \tag{23}$$

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.

IV. Numerical Parametrization of Trajectories

The finite-dimensional trajectory parametrization employed for numerical global optimization is described next. 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. A waypoint is simply a configuration $\boldsymbol{q}_i = (\boldsymbol{x}_i, R_i)$ defined by its position $\boldsymbol{x}_i$ and orientation R_i. The *parametrized trajectory* $\boldsymbol{z} \in \mathcal{Z}$ is defined by

$$\boldsymbol{z} = (\boldsymbol{q}_1, ..., \boldsymbol{q}_m), \tag{24}$$

where $\mathcal{Z} = (\mathbb{R}^3 \times SO(3))^m$ is the parametrized trajectory space. Thus the infinite dimensional space of continuous trajectories has been encoded through a low-order finite dimensional space $\mathcal{Z}$. Our approach is to connect waypoints with the simplest possible representation ensuring feasible execution. Feasibility imposes smoothness conditions that are accomplished through polynomials of an appropriate order. Interpolation in position and orientation space will be described separately since the latter is a nonlinear space that requires local chart switching. In addition, position interpolation in the fully-actuated and underactuated case are treated differently since the use of differential flatness §III A imposes higher-order smoothness conditions.

A. Fully Actuated Polynomial Curves in Position

Consider the interpolation of a trajectory between two boundary conditions $(\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0)$ and $(\boldsymbol{x}_g, \dot{\boldsymbol{x}}_g)$ that must pass through intermediate waypoints $(\boldsymbol{x}_1, ..., \boldsymbol{x}_m)$. The boundary conditions, the waypoints, and enforcing second order smoothness at the waypoints (i.e. continuity of $\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}$) impose a total of $(4 + 4m)$ conditions (for each of the 3 coordinates in $\boldsymbol{x}$) 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

$$\boldsymbol{x}(t) = C_i \boldsymbol{b}(t - i\Delta), \text{ when } t \in [t_i, t_{i+1}] \tag{25}$$

where C_i is a matrix of constant coefficients and $\boldsymbol{b} : [0, \Delta] \to \mathbb{R}^4$ is defined by $\boldsymbol{b}(h) = (h^3, h^2, h, 1)$ for $h \in [0, \Delta]$.

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

$$C_0 = [\boldsymbol{x}_0 \ \dot{\boldsymbol{x}}_0 \ \ddot{\boldsymbol{x}}_0 \ \boldsymbol{x}_1]A, \tag{26}$$

$$C_i = [C_{i-1}E \ \ \boldsymbol{x}_{i+1}]A, \text{ for } i = 1, ..., m, \tag{27}$$

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

$$C_{m+1}\dot{\boldsymbol{b}}(\Delta) = \dot{\boldsymbol{x}}_g \tag{28}$$

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

B. Underactuated Polynomial Curves in Position

Similarly to the fully-actuated case boundary conditions are given in terms of $\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0$ and $\boldsymbol{x}_g, \dot{\boldsymbol{x}}_g$ and the trajectory must pass through the intermediate spinets $(\boldsymbol{x}_1, ..., \boldsymbol{x}_m)$. 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 $4 + 5m$ conditions (for each of the three coordinates in $\boldsymbol{x}$) 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 position trajectory is expressed as

$$\boldsymbol{x}(t) = \begin{cases} C_0\boldsymbol{b}(t), & \text{when } t \in [0, \Delta], \\ C_i\boldsymbol{b}_4(t - i\Delta), & \text{when } t \in [t_i, t_{i+1}], \text{ for } i = 1, ..., m, \end{cases} \tag{29}$$

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

Similarly to §IV A 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 are found using the following recursion

$$C_0 = [\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0, \ddot{\boldsymbol{x}}_0, \boldsymbol{x}_1]A_3, \tag{30}$$

$$C_1 = [C_0E_3 \ \ \boldsymbol{x}_2]A_4, \tag{31}$$

$$C_i = [C_{i-1}E_4 \ \boldsymbol{x}_{i+1}]A_4, \text{ for } i = 2, ..., m, \tag{32}$$

The unknown initial acceleration $\ddot{\boldsymbol{x}}_0$ is found by solving the linear equation

$$C_{m+1}\dot{\boldsymbol{b}}_4(\Delta) = \dot{\boldsymbol{x}}_g \tag{33}$$

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

C. Polynomial Curves in Rotation

Consider the interpolation of a trajectory between two boundary conditions $(R_0, \boldsymbol{\omega}_0)$ and $(R_g, \boldsymbol{\omega}_g)$ that must pass through intermediate waypoints $(R_1, ..., R_m)$. In order to perform interpolation the curve along each segment is represented locally using exponential coordinates $\boldsymbol{r} : [0, \Delta] \to \mathbb{R}^3$. Using the definitions (14)–(15) this is accomplished by setting

$$(R(t_i + h), \boldsymbol{\omega}(t_i + h)) = (R_i \exp(\boldsymbol{r}(h)),\ \operatorname{dexp}(-\boldsymbol{r}(h))\dot{\boldsymbol{r}}(h))\,, \tag{34}$$

whenever $t \in [t_i, t_{i+1}]$. Interpolation can now be performed by assuming a polynomial form for the curve $\boldsymbol{r}(h)$. The simplest scheme ensuring continuity of $R(t)$ and $\boldsymbol{\omega}(t)$ is to employ quadratic curves in the first m segments and a cubic in the last segment. The quadratic curves joining a given state $(R_i, \boldsymbol{\omega}_i)$ to the next rotation R_{i+1} are given by

$$\boldsymbol{r}(h) = \boldsymbol{\omega}_i h + \frac{\log(R_i^{-1}R_{i+1}) - \Delta\boldsymbol{\omega}_i}{\Delta^2}h^2,$$

where $h \in [0, \Delta]$. The cubic curves joining two given states $(R_i, \boldsymbol{\omega}_i)$ and $(R_{i+1}, \boldsymbol{\omega}_{i+1})$ are given by

$$\boldsymbol{r}(h) = \boldsymbol{\omega}_i h + \boldsymbol{c}_2 h^2 + \boldsymbol{c}_3 h^3,$$

where

$$\boldsymbol{c}_3 = \frac{\operatorname{dexp}^{-1}(-\log(R_i^{-1}R_{i+1}))\boldsymbol{\omega}_{i+1} + \boldsymbol{\omega}_i}{\Delta^2} - 2\frac{\log(R_i^{-1}R_{i+1})}{\Delta}, \quad \boldsymbol{c}_2 = \frac{\log(R_i^{-1}R_{i+1})}{\Delta^2} - \frac{\boldsymbol{\omega}_i}{\Delta} - \Delta\boldsymbol{c}_3.$$

These expressions were obtained using the logarithm map $\log : SO(3) \to \mathbb{R}^3$ which is the inverse of the exponential $\log = \exp^{-1}$ and is defined by

$$\log(R) = \begin{cases} \mathbf{0}, & \text{if } \phi = 0, \\ \frac{\phi}{2\sin\phi}(\widetilde{R - R^T}), & \text{if } \phi \neq 0 \end{cases}, \tag{35}$$

where $\phi = \arccos \frac{\text{trace}(R)-1}{2}$ and the operator $\check{\cdot}$ is the inverse of $\hat{\cdot}$ defined in (4). The map dexp^{-1} is the inverse of the right-trivialized tangent (15) and can be computed in closed form according to

$$\text{dexp}^{-1}(\boldsymbol{r}) = \begin{cases} \mathbf{I}_3, & \text{if } \boldsymbol{r} = 0 \\ \mathbf{I}_3 - \frac{1}{2}\widehat{\boldsymbol{r}} + \frac{2-\|\boldsymbol{r}\|\cot\frac{\|\boldsymbol{r}\|}{2}}{2\|\boldsymbol{r}\|^2}\widehat{\boldsymbol{r}}^2, & \text{if } \boldsymbol{r} \neq 0. \end{cases} \tag{36}$$

V. Motion Planning

Motion planning is formulated as finding the optimal parametrized trajectory $\boldsymbol{z}^* \in \mathcal{Z}$ which does not violate the imposed constraints. Once the optimal parameter (i.e. sequence of waypoints) is computed it is mapped back to a continuous path $\boldsymbol{q}(t)$ for all $t \in [0, T]$ using the polynomial interpolations in position space in the fully- and under-actuated cases using (25) and (29), respectively, and in the space of rotations $R(t)$ using (34). The poses $\boldsymbol{q}(t)$ are then mapped back to the full trajectory and control inputs $(\boldsymbol{s}, \boldsymbol{u}) : [0, T] \to S \times U$ as described in §III A for the underactuated case and in §III B for fully-actuated systems.

Global optimization over the trajectory parameter space $\mathcal{Z}$ is accomplished using the recently proposed cross-entropy (CE) motion planning method [34]. The method was chosen for its simple gradient-free implementation suitable for real-time execution on-board the vehicle. The method was selected over other global tree-based randomized planning methods [7, 8, 40] since it globally explores the space but is also expected to quickly converge near the optimal solution. In addition, it is directly applicable to nonlinear dynamics and arbitrary constraints and does not require linearization or additional smoothing steps. Its main limitation is that it cannot handle environments with narrow passages and currently there is no principled way to determine in advance an optimal number of waypoints m for a given environment. To overcome these limitations the method was combined with recent sampling-based methods, in particular RRT* [41], to construct a family of adaptive graph-based methods with improved efficiency [34]. In this work, though, it is sufficient to employ it as a standalone trajectory optimization method since the environments considered are not expected to have such narrow passages.

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 [35]. Note that many of the sampled trajectories will violate the problem constraints (9) 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 $\boldsymbol{z} \in \mathcal{Z}_{\text{con}}$ will be retained during sampling to update the distribution. In addition define the cost function $J : \mathcal{Z} \to \mathbb{R}$ which computes the cost (37) of a given trajectory parametrized by $\boldsymbol{z}$, i.e.

$$J(\boldsymbol{z}) = \int_0^T C(\boldsymbol{s}(t), \boldsymbol{u}(t))\text{dt}, \tag{37}$$

where $\boldsymbol{s}(t)$ and $\boldsymbol{u}(t)$ are obtained from z using the control computations (§III A,§III B).

Let Z be a random variable over $\mathcal{Z}$ with density $p(Z; \boldsymbol{v})$, where $\boldsymbol{v} \in \mathcal{V}$ is the density parameter. Let $n_z = \dim(\mathcal{Z})$. 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 $\boldsymbol{v} = (\boldsymbol{\mu}_1, \Sigma_1, ..., \boldsymbol{\mu}_K, \Sigma_K, w_1, ..., w_K)$ corresponding to K mixture components with means $\boldsymbol{\mu}_k$, covariance matrices Σ_k (excluding identical elements due to the matrix symmetry) and weights w_k. A mixture of Gaussians is employed defined by

$$p(\boldsymbol{z}; \boldsymbol{v}) = \sum_{k=1}^{K} \frac{w_k}{\sqrt{(2\pi)^{n_z}|\Sigma_k|}} e^{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\boldsymbol{z}-\boldsymbol{\mu}_k)}, \tag{38}$$

where $\sum_{k=1}^{K} w_k = 1$. The number of mixture components K is chosen adaptively (see e.g. [36]). 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 [34] for complete details in a general setting).

Note that step (5) for $K = 1$ reduces to simply computing the mean and covariance of the samples; for $K > 1$ it is performed using an adaptive expectation-minimization procedure [36]. The algorithm performs very efficiently when the environment is not very complex. If the optimal trajectory lies in a narrow passage with small probability 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 [34].

Finally, note that Algorithm 1 assumes that $\mathcal{Z}$ is a vector space. Thus, the identification

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

is employed so that sampling occurs in the product space of unit quaternions S^3 rather than rotation

Algorithm 1: $\boldsymbol{u}(t) \leftarrow$ `Trajectory_Planner`$(\boldsymbol{s}_0, S_g, T, m)$

Initialize CE method with equally spaced waypoints $\boldsymbol{z}_0 \in \mathcal{Z}$ connecting $\boldsymbol{s}_0$ and S_g

Set Σ_0 so that the region $\{\boldsymbol{z} \in \mathcal{Z} \mid (\boldsymbol{z} - \boldsymbol{z}^*)^T \Sigma (\boldsymbol{z} - \boldsymbol{z}^*) < 2,\ 0 \le t \le T\}$ covers the workspace

Choose initial samples $Z_1, ..., Z_N$ from `Normal`$(\boldsymbol{z}_0, \Sigma_0)$; set $j \leftarrow 0$ and $\hat{\gamma}_0 \leftarrow \infty$

for $j = 0 : maxIterations$ **do**

  Update distribution over the elite set $\mathcal{E}_j = \{Z_i \mid J(Z_i) \le \hat{\gamma}_j\}$ through

$$\hat{\boldsymbol{v}}_j \leftarrow \underset{\boldsymbol{v} \in \mathcal{V}}{\operatorname{argmin}} \frac{1}{|\mathcal{E}_j|} \sum_{Z_k \in \mathcal{E}_j} \ln p(Z_k; \boldsymbol{v})$$

  Generate samples $Z_1, ..., Z_N$ from $p(\cdot, \boldsymbol{v}_j)|_{\mathcal{Z}_{\text{con}}}$ and compute the ϱ–th quantile $\hat{\gamma}_{j+1} \leftarrow J_{\lceil \varrho N \rceil}$

  if $KL(p(\cdot, \boldsymbol{v}_{j-1}) || p(\cdot, \boldsymbol{v}_j)) < \epsilon$ **then**
    break

Set the best sample $\boldsymbol{z}^* = \arg\min_{Z_i} J(Z_i)$

Map $\boldsymbol{z}^*$ to trajectory $(\boldsymbol{x}(t), R(t))$ using (25,34) for underactuated, or (29,34) for fully-actuated case

Compute controls $\boldsymbol{u}(t)$ from $(\boldsymbol{x}(t), R(t))$ using (23) in fully-actuated or (1) in under-actuated case

return $\boldsymbol{u}(t)$

matrices $SO(3)$. This enables sampling in the ambient vector space $\mathbb{R}^{7m}$ by initially ignoring the quaternion unit norm constraints and once the waypoints are chosen to project them back to $\mathcal{Z}$.

VI. Applications

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

The spacecraft, obstacles, and asteroid surfaces are modeled as triangulated polyhedra. The obstacle avoidance constraint defined in (12) is then implemented using PQP [31]. All computations are performed using an Intel CORE-i7 2GHz CPU.

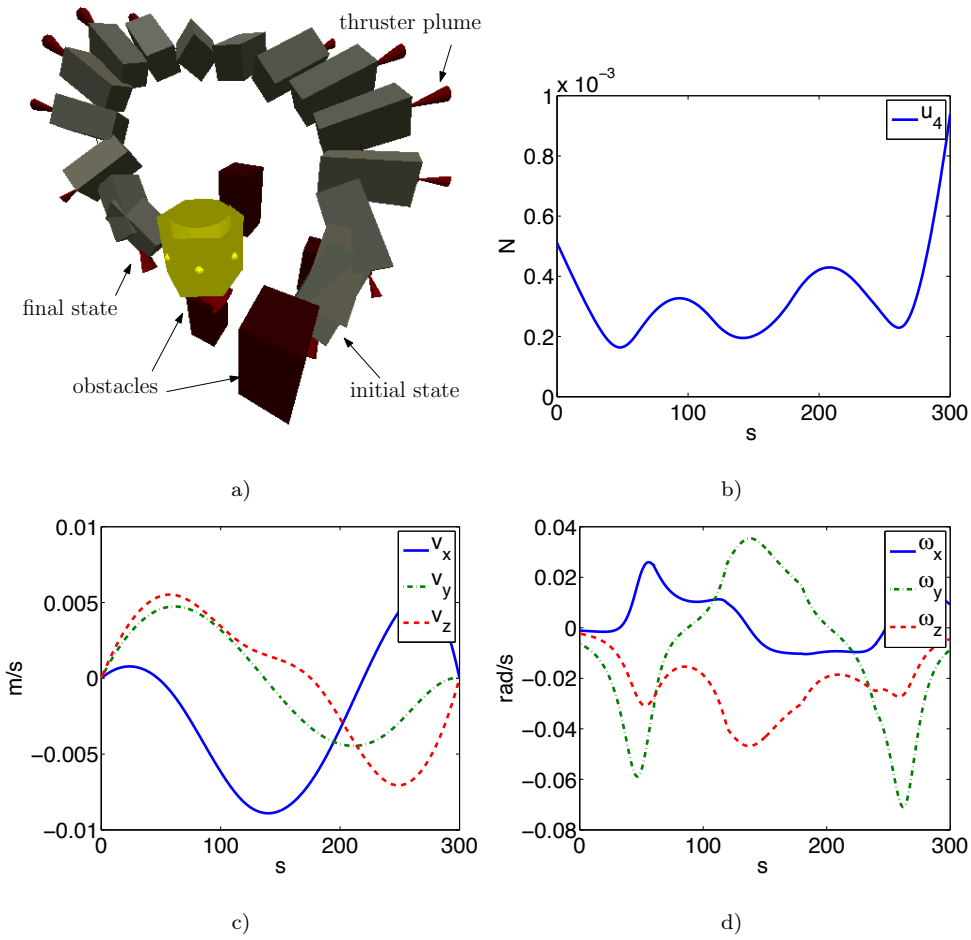


Fig. 4 Reconfiguration of ACS+Z underactuated cubesat: a) resulting path to goal state on opposite side of larger spacecraft; b) force produced by single thruster acting along +Z axis; c) translational velocity components; d) angular velocity components.

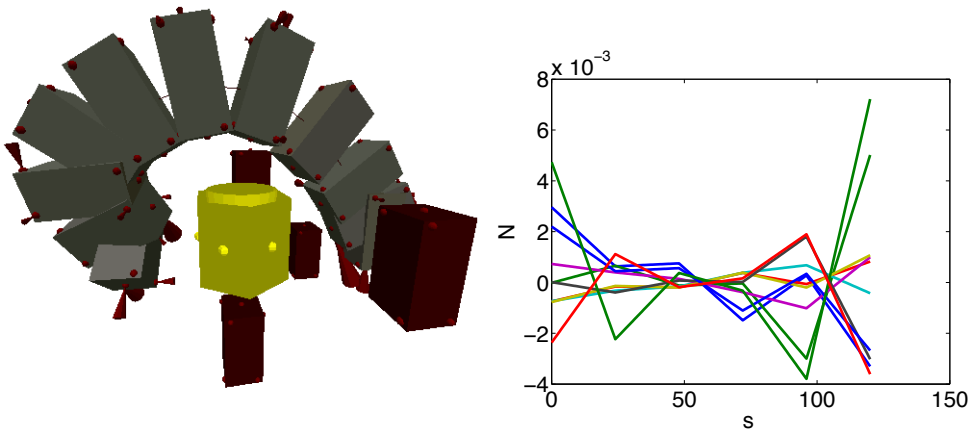
A. Reconfiguration

The reconfiguration task requires a cubesat to navigate around a larger spacecraft and a few other cubesats and arrive optimally to a designated zero-velocity state. The initial and final configurations are one meter apart. The larger obstacle lies directly between these two configurations while the smaller ones are scattered close to the path in order to create a non-trivial planning scenario.

Figure 4 shows the resulting trajectory and controls for the underactuated cubesat ACS+Z (defined in (5)) computed using Proposition 1. The cost function used for the ACS+Z system is

$$C(\boldsymbol{s}, \boldsymbol{u}) = |u_4| + \|\boldsymbol{u}_{1:3}\|, \tag{39}$$

in order to encode the fuel expended by the thruster but also penalize excessive torque required by



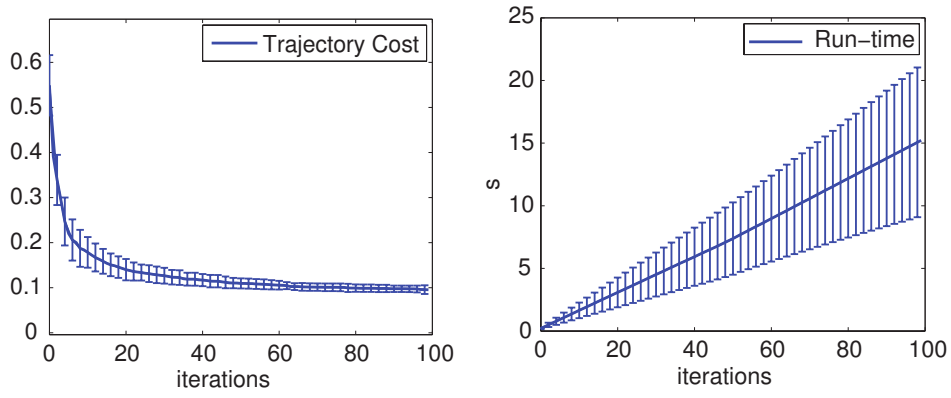
a) b)

Fig. 5 Reconfiguration of T10 fully-actuated cubesat: a) resulting path to goal state on opposite side of larger spacecraft; b) forces produced by thrusters.

the ACS. The trajectory required $\Delta V = 0.089$ m/s and angular momentum change 0.035 Nms. 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 those in Figure 2a-c and ACS in Figure 2d. The computation was based on trajectories with $m = 4$ waypoints and collision checking was performed by tracing the trajectory with a time-step $h = 0.5$ s using the PQP package. The CE optimization method (1) 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 5 shows the resulting trajectory and controls for the fully actuated cubesat T10 (defined in (8)). 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 the bounds on the inputs.

Finally, it is acknowledged that the quality of the computed solution depends on the number of algorithm iterations and number of samples employed. Figure 6 illustrates the dependence between the resulting trajectory cost and required runtime. Real-time performance is achieved by using a smaller number of samples ($N = 250$) and by performing a fixed number of iterations (say 10). In this particular scenario, such strategy results in a control law with performance roughly 20% worse



a) b)

Fig. 6 Trajectory cost and required runtime for CE motion planning method applied to scenario shown in Figure 4 averaged over ten runs with randomly placed obstacles. Number of algorithm iterations shown on x-axis.

than the performance achieved after 100 iterations which has almost converged near an optimum.

B. Asteroid Sampling

The second scenario is a sampling maneuver on the surface of a small asteroid. Motivated by recent missions such MUSES-C [37] and DAWN [38] the proposed methodology can be used to enhance autonomy and sampling capabilities. For instance, a cubesat could be carried on-board a larger spacecraft and dispatched to various sampling points.

Note that only the near-surface navigation problem is considered, 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 of the asteroid, it can perform autonomous motion planning to various points of interest. Unlike the reconfiguration example (§VI A) the external disturbances in this example are dominated by microgravity, i.e. $\boldsymbol{f}_{\text{ext}}$ should additionally encode the asteroid gravitational effects. Gravity is assumed to be constant with a chosen value between e.g. 0.0001 m/s^2 (corresponding to the small Itokawa asteroid) to 0.25 m/s^2 (on Vesta–one of the largest asteroids in the Solar System). The cost function used in the reconfiguration scenario defined in (39) is used in this scenario as well.

Figure 7 shows the optimized obstacle-free trajectory taking the underactuated spacecraft to

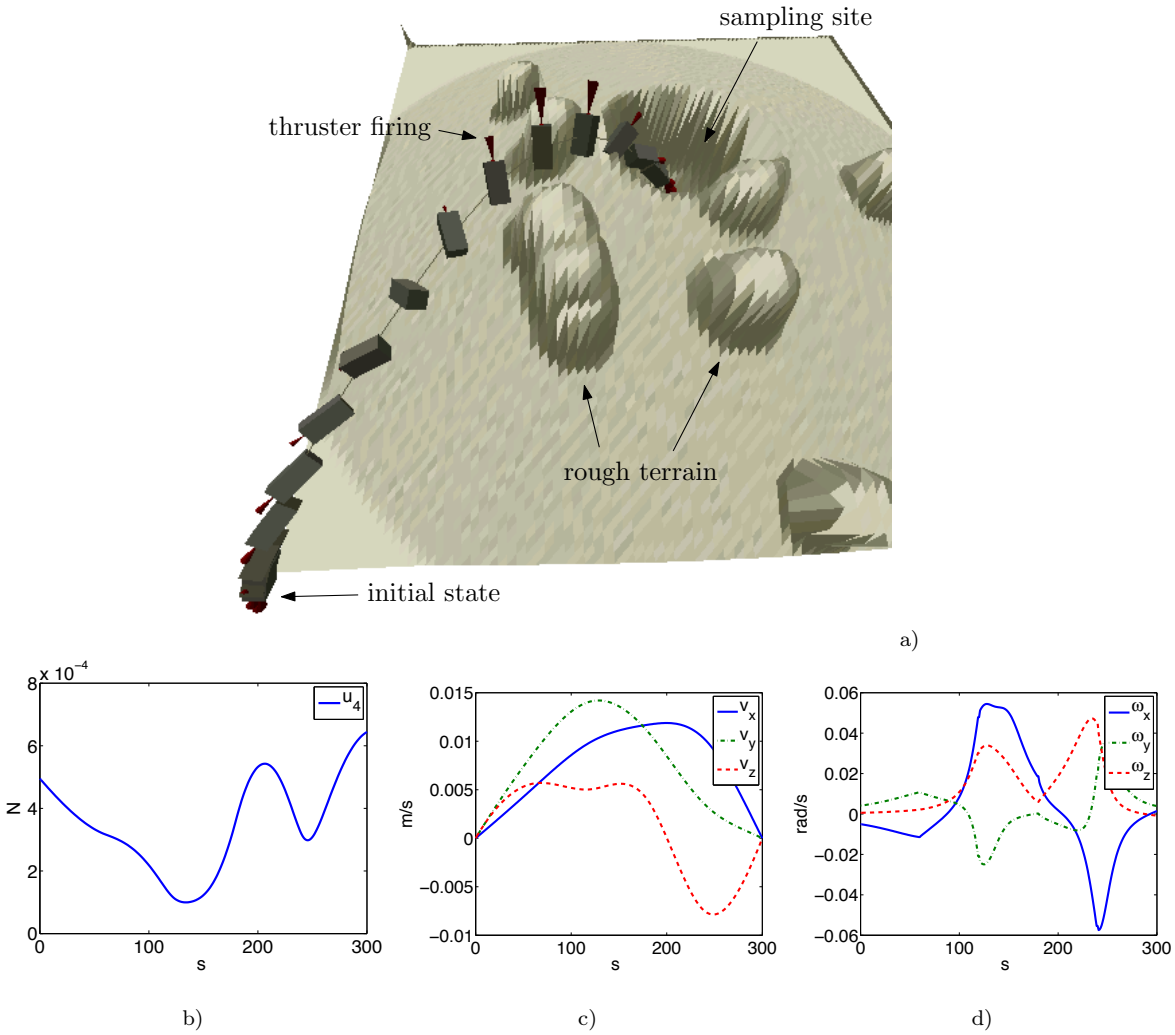


Fig. 7 Asteroid crater sampling using ACS+Z underactuated cubesat: a) optimized path to the bottom of crater; b) resulting force produced by single thruster along z-axis; c) resulting translational velocities; d) angular velocities.

the bottom of a small crater with zero final velocity. The algorithm is setup similarly to §VI A with external acceleration due to gravity set to 0.0001 m/s^2. 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 angular momentum change 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. 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 of reaching the true optimum but at an increased computational effort.

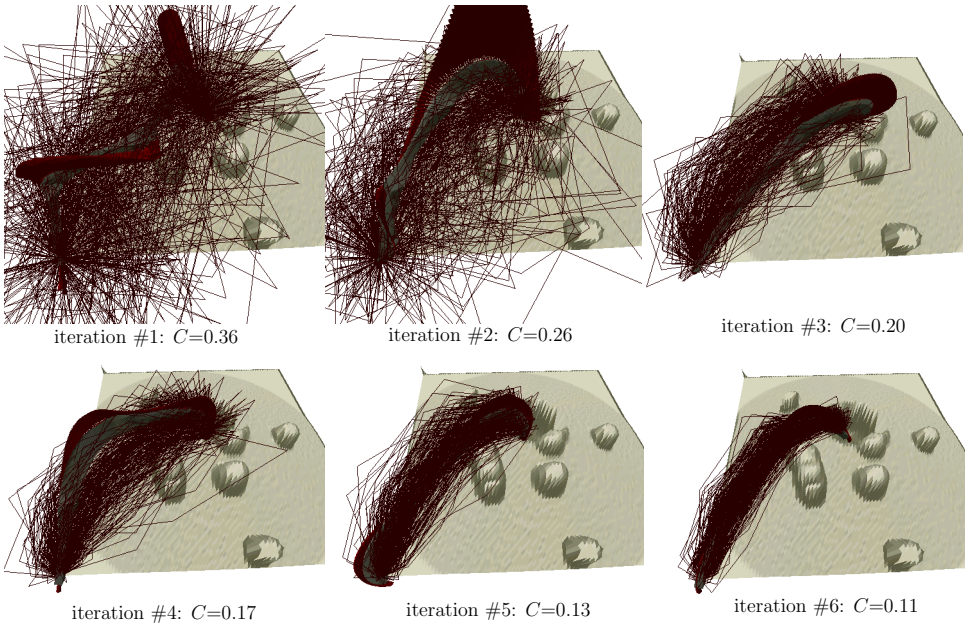


Fig. 8 Several iterations of CE optimization method (§V) applied to sampling scenario. Thin lines connect waypoints corresponding to samples from parametrized trajectory space $\mathcal{Z}$. As the algorithm iterates the distribution and its samples concentrate close to the optimum. Actual trajectory (not shown) passes through the waypoints and is smooth and feasible.

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 system and required 60 seconds for computing the trajectory shown.

VII. Experiments

A simple testbed was developed in order to study the proposed techniques. It consists of an air hockey-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 control board with microprocessors and a single-board-computer with wireless communication. Since at the time of this study no low-cost propulsion system for cubesats was readily available a system consisting of six electric ducted microfans that can generate forces down to 1.0 mN

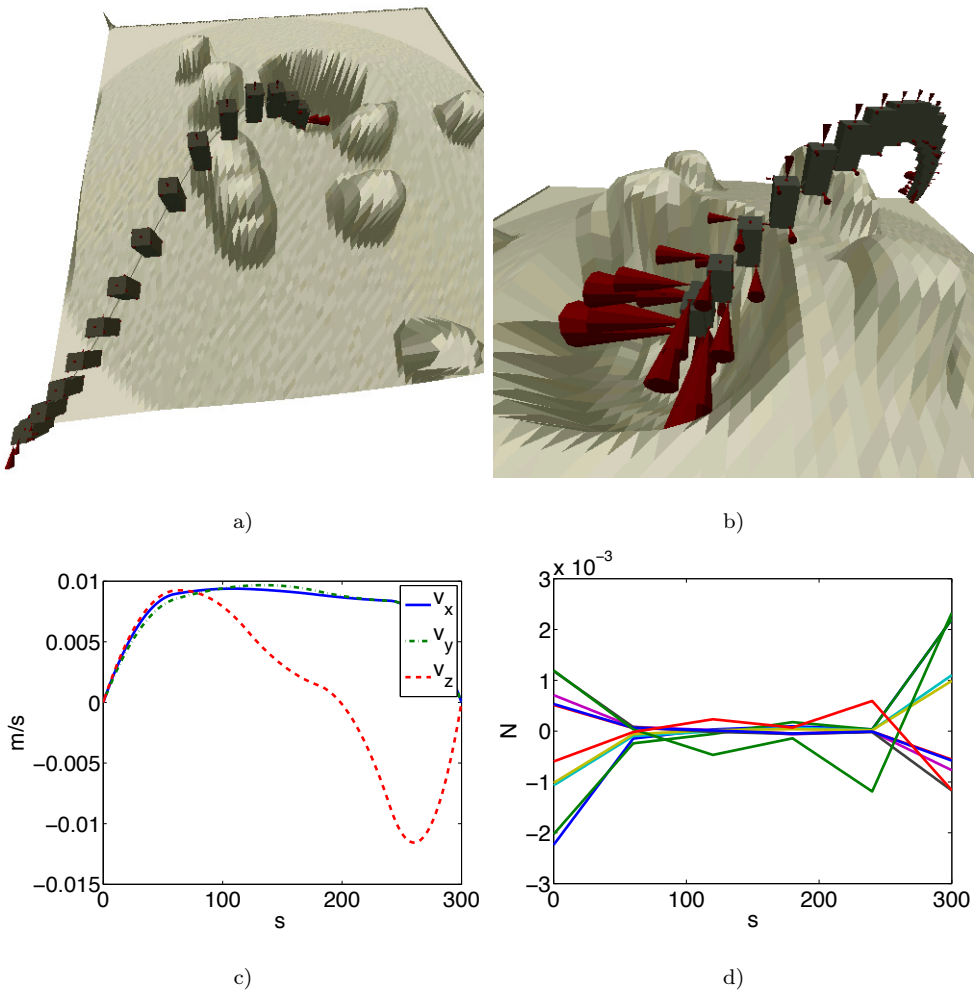


Fig. 9 Asteroid crater sampling using T10 fully actuated cubesat: a) optimized path to bottom of crater; b) close-up view c) resulting translational velocity components; d) resulting forces produced by thrusters.

was built. The fans are used to simulate either underactuated or fully-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 using on-board cameras.

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 2 provides more details of the engineering model.

The proposed motion planning algorithm was implemented and tested on the air table using

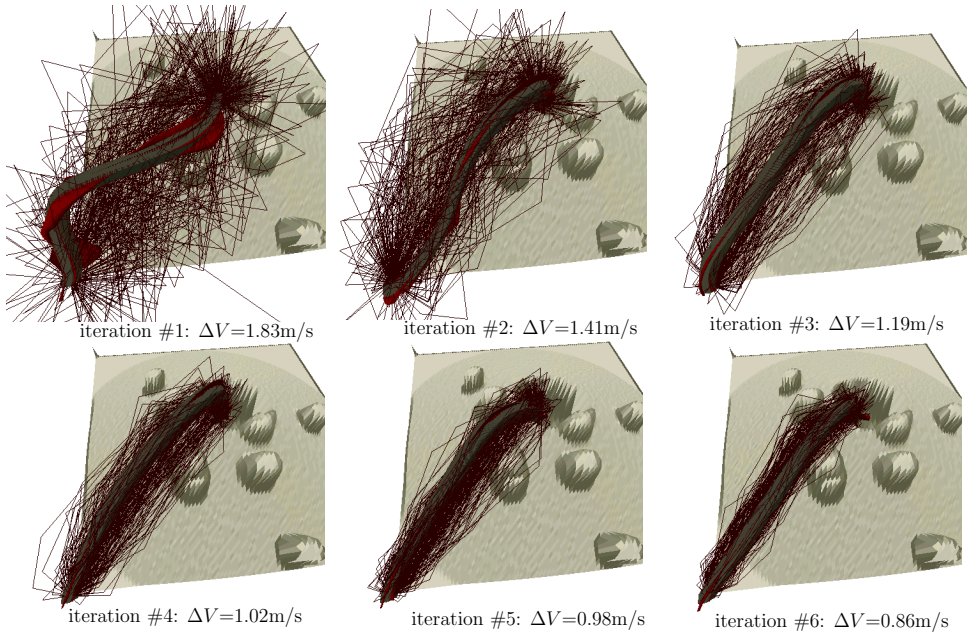


Fig. 10 Iterations of CE method applied to fully-actuated cubesat performing asteroid sampling.

Component	Qty	Description	Mass	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, magnets, LEDs, etc	20g	0.1W
high-level computer	1	Gumstix Overo 600MHz	8g	1W
comms	1	i2c, wifi+bluetooth, antennas	10g	0.9W

Table 2 Details of engineering cubesat model.

a mock-up obstacle environment. Figure 12 shows an example setup and a few frames along the computed motion. Here an earlier prototype of the robotic cubesat is shown. In the context of this simple two-dimensional environment the proposed algorithm performs very efficiently. The CE optimization method converges to a solution every 50 ms. Thus, there was no need for additional trajectory tracking since the path was updated and executed in real-time. Figure 11 shows the assembled robotic surrogate and a few frames along a docking motion using visual feedback from

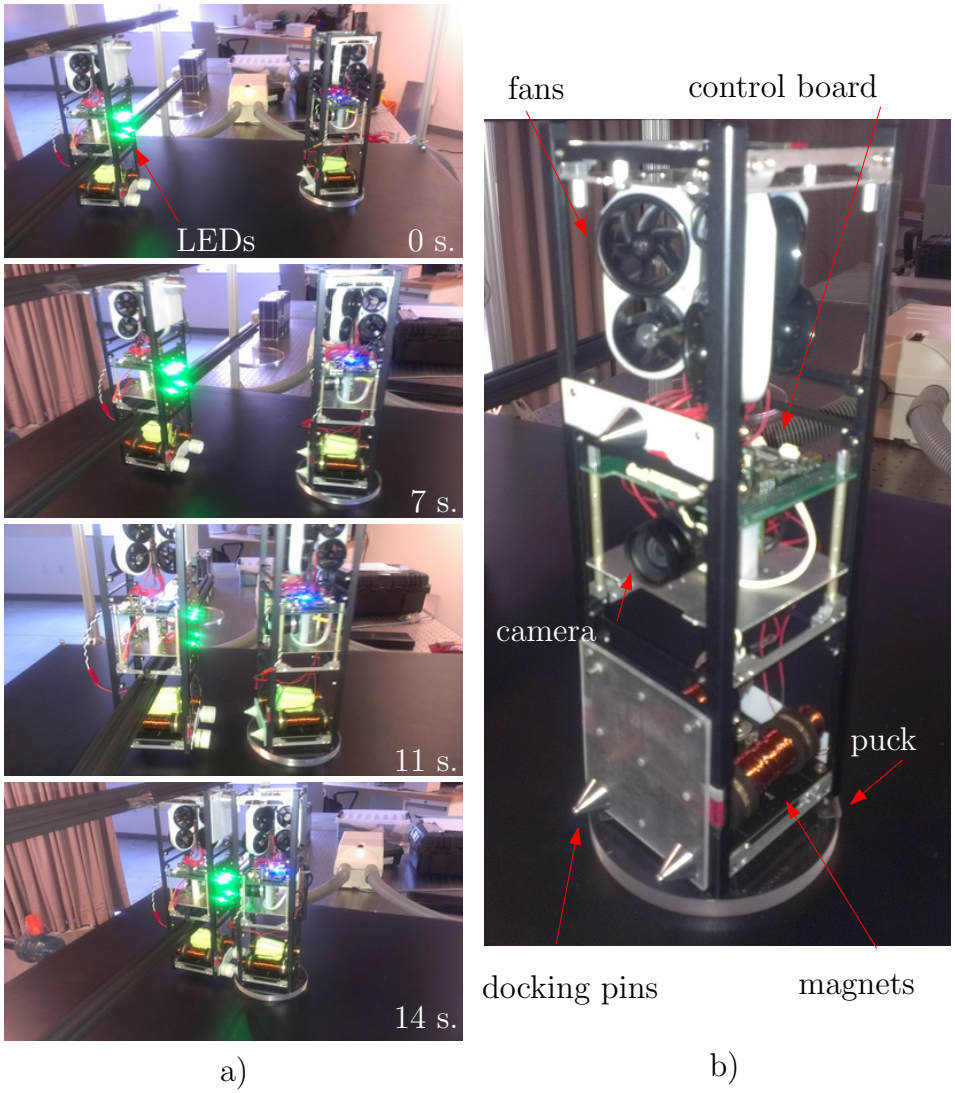


Fig. 11 a) frames along path to docking using visual pose estimation b) close-up view of model

the on-board camera. Extending these results to a realistic 3-D environment imposes additional challenges and requires further study.

VIII. Conclusion

The trajectory planning problem for small spacecraft proximity operations has been studied. Motivated by the emergence of cubesat propulsion technologies the proposed method provides 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 has been constructed. It exploits the properties of the dynamics, i.e. differential flatness and

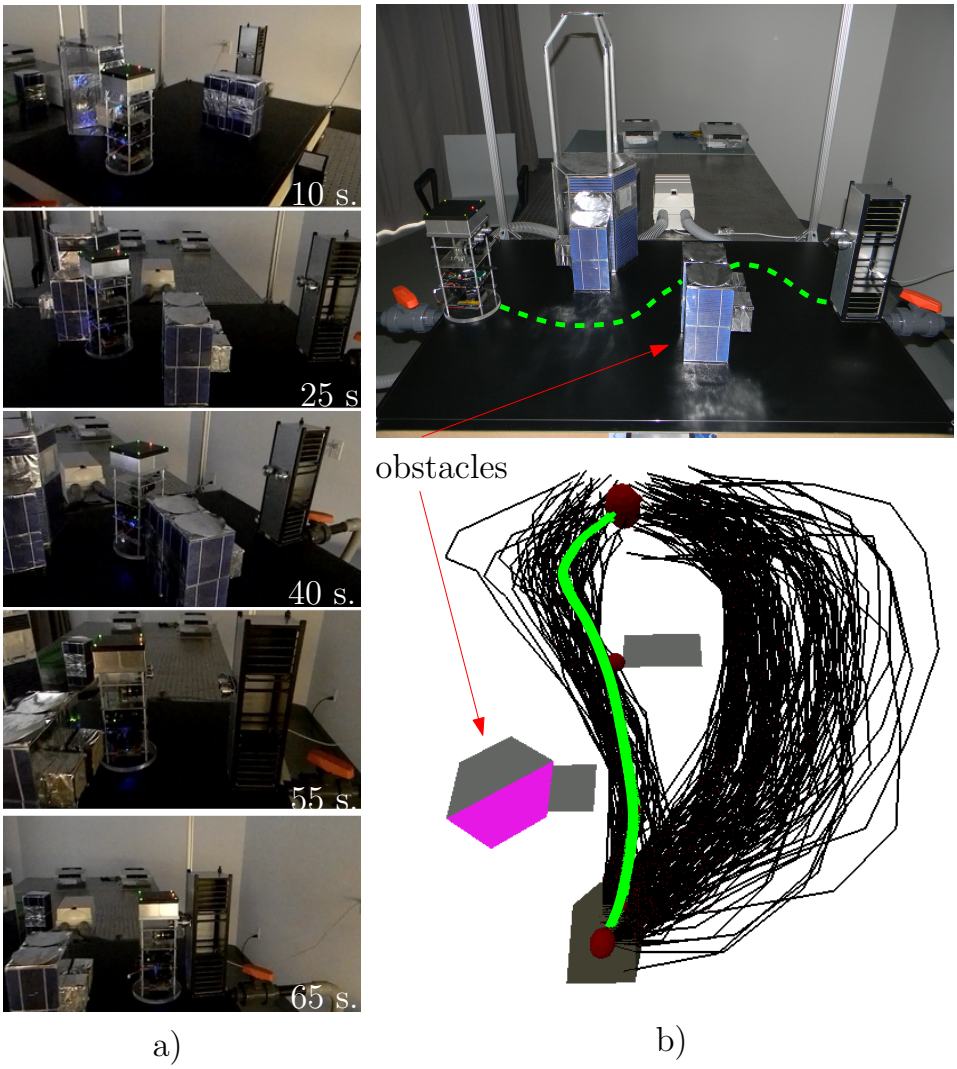


Fig. 12 a) frames along trajectory of cubesat engineering model that avoids obstacles and performs docking b) setup (top) and output of motion planning algorithm showing best path (green lines)

natural external forces for instant trajectory generation based on reduced-order parametrization. Stochastic optimization is then performed to globally search 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 is capable of operating 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 expected that the algorithm complexity can be significantly reduced to enable real-time on-board implementation.

ACKNOWLEDGMENTS

Keith Patterson (Caltech) and Professor Craig Underwood (University of Surrey) are thanked for providing help and advice in setting up the experiment. Financial support from the Keck Institute for Space Studies is gratefully acknowledged.

REFERENCES

[1] Underwood, C., and Pellegrino, S., "Autonomous Assembly of a Reconfigurable Space Telescope (AAReST) for Astronomy and Earth Observation", In *8th IAA Symposium on Small Satellites for Earth Observation*, 2011.

[2] Scharf, D.P., Hadaegh, F.Y. , and Kang, B., "A Survey of Spacecraft Formation Flying Guidance", In *Proceedings of the Intl. Symposium Formation Flying*, 2002.

[3] Scharf, D.P., Hadaegh, F.Y., and Ploen, S.R., "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.

[4] Acikmese, B., Scharf, D.P., Murray, E., and Hadaegh, F.Y., "A Convex Guidance Algorithm for Formation Reconfiguration", In *Proceedings of the 2003 American Control Conference* , 2003.

[5] Sultan, C., Seereram, S., and Mehra, R.K., "Deep Space Formation Flying Spacecraft Path Planning", *Int. J. Rob. Res.*, 26(4):405–430, April 2007.

[6] Richards, A., Schouwenaars, T., How, J.P., and Feron, E., "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.

[7] Frazzoli, E., "Quasi-Random Algorithms for Real-Time Spacecraft Motion Planning and Coordination", *Acta Astronautica*, 53(4âĂŞ10):485 – 495, 2003.

[8] Phillips, J.M., Kavraki, L.E., and Bedrosian, N., "Probabilistic Optimization Applied to Spacecraft Rendezvous and Docking", In *13th American Astronomical Society/AIAA - Space Flight Mechanics Meeting*, Puerto Rico, February 2003.

[9] Breger, L., and How, J.P., "Safe Trajectories for Autonomous Rendezvous of Spacecraft", *AIAA Journal on Guidance, Control, and Dynamics*, 31(5):1478–1489, 2008.

[10] Scharf, D.P., Hadaegh, F.Y., Rahman, Z.H., Shields, J.F., and Singh, G., "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.

[11] Aoude, G.S., How, J.P., and Miller, D.W., "Reconfiguration Maneuver Experiments Using the SPHERES Testbed Onboard the ISS", In *Proceedings of the 3rd International Symposium on Formation Flying, Missions and Technologies*, 2008.

[12] Saenz-Otero, A., and Miller, D.W., "Spheres: a Platform for Formation-Flight Research", volume 5899, page 58990O. SPIE, 2005.

[13] Tweddle, B.E., Saenz-Otero, A., and Miller, D.W., "Design and Development of a Visual Navigation Testbed for Spacecraft Proximity Operations", In *AIAA SPACE 2009 Conference and Exposition*, 2009.

[14] Miller, D.W., Mohan, S., and Budinoff, J., "Assembly of a Large Modular Optical Telescope (ALMOST)", In *SPIE Space Telescopes and Instrumentation*, 2008.

[15] Faiz, N., Agrawal, S., and Murray, R.M., "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.

[16] Murray, R.M., Rathinam, M., and Sluis, W.M., "Differential Flatness of Mechanical Control Systems", In *Proceedings ASME International Congress and Exposition*, 1995.

[17] Van Nieuwstadt, M.J., and Murray, R.M., "Real Time Trajectory Generation for Differentially Flat Systems", *International Journal of Robust and Nonlinear Control*, 8(11):995–1020, 1998.

[18] Tsiotras, P., "Feasible Trajectory Generation for Underactuated Spacecraft Using Differential Flatness", In *AAS/AIAA Astrodynamics Specialist Conference*, 1999.

[19] Louembet, C., "Collision Avoidance in Low Thrust Rendezvous Guidance Using Flatness and Positive b-Splines", In *American Control Conference*, 2011.

[20] Bullo, F., and Lynch, K. M., "Kinematic Controllability for Decoupled Trajectory Planning in Underactuated Mechanical Systems", *IEEE Transactions on Robotics and Automation*, 17(4):402–412, 2001.

[21] Frazzoli, E., Dahleh, M. A., and Feron, E., "Maneuver-Based Motion Planning for Nonlinear Systems with Symmetries", *IEEE Transactions on Robotics*, 21(6):1077–1091, dec 2005.

[22] Mueller, J., Hofer, R., and Ziemer, J., "Survey of Propulsion Technologies Applicable to Cubesats",

Technical report, Jet Propulsion Laboratory, 2010.

[23] Hinkley, D. A., "Picosatellites at The Aerospace Corporation", in: *Small Satellites: Past, Present, and Future*, pages 635–674, Number 20, edited by H. Helvajian and S.W. Janson, 2009.

[24] Janson, S., Huang, A., Hansen, W., and Helvajian, H., "Development of an Inspector Satellite Propulsion Module using Photostructurable Glass/Ceramic Materials", In *AIAA Conference on Micro-/Nanotechnologies*, 2004.

[25] VACCO Industries, ChEMS micro-propulsion system.

[26] Schmuland, D. T., Masse, R. K., and Sota, C. G., " Hydrazine Propulsion Module for Cubesats", In *Small Satellite Conference*, 2011.

[27] SSC Nanospace, Cubesat mems propulsion module.

[28] 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.

[29] BUSEK Space Propulsion Industries, Cubesat electrospray thruster system.

[30] Vallado, D., *Fundamentals of Astrodynamics and Applications*. Primis, 1997.

[31] Gottschalk, S., Lin, M. C., and Manocha, D., " OBBTree: A Hierarchical Structure for Rapid Interference Detection", *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 30:171–180, 1996.

[32] Frazzoli, E., Dahleh, M. A., and Feron, E., "Trajectory Tracking Control Design for Autonomous Helicopters Using a Backstepping Algorithm", In *Proceedings American Control Conference*, pages 4102–4107, Chicago, IL, June 2000.

[33] Kobilarov, M., and Marsden, J., " Discrete Geometric Optimal Control on Lie Groups ", *IEEE Transactions on Robotics*, 27(4):641–655, 2011.

[34] Kobilarov, M., " Cross-Entropy Motion Planning", *International Journal of Robotics Research*, 31(7):855–871, 2012.

[35] Rubenstein, R. Y., and Kroese, D. P., *Simulation and the Monte Carlo Method*. Wiley, 2008.

[36] Figueiredo, M. A. F., and Jain, A. K., " Unsupervised Learning of Finite Mixture Models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[37] Kubota, T., Sawai, S., Hashimoto, T., Kawaguchi, J., and Fujiwara, A., " 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.

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

[39] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration*, ser. Springer Series in Com-

putational Mathematics. Springer-Verlag, 2006, no. 31.

[40] Garcia, I. and How, J. P., "Trajectory optimization for satellite reconfiguration maneuvers with position and attitude constraints", In *American Control Conference (ACC)*, 2005, pp. 889-894.

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