Constrained Kinodynamic Planning for a Magnetorquer-Only Actuated Satellite Using a Modified RRT Planner

Mark Stephenson

December 2022

Abstract

While magnetorquers are a reliable attitude actuator for satellites in low-Earth orbit (LEO), they are typically not used alone for attitude control since they are only capable of producing torques in the magnetic field-normal plane. Prior solutions to this underactuated control problem yield orbits-long attitude maneuvers. Using techniques from motion planning, agile maneuvers are computed that are only minutes-long. These maneuvers can also have keepout region constraints for sensitive instruments. This paper introduces novel improvements to the randomly exploring tree (RRT) planner to improve performance on highly underactuated kinodynamic planning problems with time-varying dynamics and narrowly reachable goal states, without additional analysis of the system such as linearization or steering functions.

1 Introduction

Magnetorquers are a commonly used attitude control actuator for satellites in low-Earth orbits. While magentorquers are reliable for long term use due to their lack of expendable resources or moving components, they are typically used in conjunction with other actuators such as reaction wheels because alone magentorquers do not provide full actuation of the system: They can only produce torques perpendicular to the local magnetic field, leaving one inertially-fixed axis uncontrollable. For satellite missions that seek to be cost-effective and risk averse, agile magentorquer-only control would allow satellites to eliminate less-durable, high-cost components such as reaction wheels while maintaining full pointing control of the vehicle.

Because of the challenging underactuated control problem that it poses, magnetorquer-only attitude control of satellites with agile pointing requirements is rarely considered. Prior work on this problem has produced solutions with caveats: [1] requires a passive momentum bias on one axis and results in multi-orbit settling times. [2] and [3] instead use position varying LQR controllers that leverage the changing magnetic field over the course of an orbit to achieve full controllability, but again result in multi-orbit settling times for pointing maneuvers which are insufficient for many applications.

This paper demonstrates the application of kinodynamic motion planning algorithms to the problem to find agile solutions to the magnetorquer-only attitude control problem. A novel variation of the of the rapidly exploring random tree (RRT) algorithm for kinodynamic planning [4] is introduced to solve the problem, called Bonsai-RRT. While previous RRT-based kinodynamic planners focus on exploring state spaces that are dynamically challenging to traverse [5] [6] or require apriori knowledge about the system dynamics to calculate reachable sets [7], Bonsai-RRT focus on converging to a narrow goal state that is reachable only from a narrow region due to underactuation in a high-dimensional state space without relying on additional analysis of the system dynamics. Unlike algorithms that grow trees from the start and the goal like RRT-Connect [8], Bonsai-RRT is applicable for systems in a time-evolving environment (for example, a time-varying magnetic field and time-varying keepout constraints).

Since RRT-based solvers are amenable to state constraints, pointing constraints (i.e. keepout/keep-in cones for sensitive instruments) are added to the problem; the constrained problem has previously been solved using motion planning techniques only for fully actuated satellites in [9]. Various methods of randomizing controls while growing the search tree are compared. Finally, the quality of the solutions are examined and the challenges of applying concepts from the optimal kinodynamic planner Stable Sparse RRT (SST) [10] to Bonsai-RRT are discussed.

2 Problem Formulation

In this section, satellite dynamics for a magnetorquer-actuated satellite are reviewed. Necessary items for kinodynamic planning — such as the state space, control space, state constraints, and distance metric — are defined.

2.1 State Space Representation

The state space of an attitude-maneuvering satellite is expressed as the tuple

$$\vec{x} = (q^{\mathcal{BI}}, \vec{\omega}_{\mathcal{B}}^{\mathcal{BI}}) \in X = SO(3) \times \mathbb{R}^3 \tag{1}$$

where $q^{\mathcal{BI}}$ is the attitude unit quaternion expressing the rotation from the body-fixed $\mathcal{B}$ to inertial $\mathcal{I}$ frame and $\vec{\omega}_{\mathcal{B}}^{\mathcal{BI}}$ is the angular rate of the body-fixed frame with respect to the body frame. Those sub- and superscripts can be assumed throughout this paper unless otherwise noted. The spacecraft's state evolves over time in accordance with Euler's equations of motion

$$\dot{\vec{x}} = \begin{bmatrix} \dot{q} \\ \dot{\vec{\omega}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\Xi(q)\vec{\omega} \\ \boldsymbol{J}^{-1}(\vec{L}_{\mathcal{B}}(t) - \vec{\omega} \times (\boldsymbol{J} \cdot \vec{\omega})) \end{bmatrix} \tag{2}$$

where $\vec{L}_{\mathcal{B}}$ is the body-frame torque [11]. In this problem, torque is only produced by the interaction between the local magnetic field $\vec{B}$ and the magnetic dipole produced by the magnetorquers $\vec{m}$. In the body frame,

$$\vec{L}_{\mathcal{B}} = \vec{m}_{\mathcal{B}} \times \vec{B}_{\mathcal{B}} \tag{3}$$

resulting in under-actuated control since $\vec{L} \perp \vec{B}$.

2.2 Control Space Simplification

Satellites are typically equipped with three orthogonal magentorquers, leading to a cubical domain for the achievable dipole $\vec{m} \in [-m_{max}, m_{max}]^3$. For symmetry under any rotation, this domain is approximated as a sphere with radius r_{max}; this significantly simplifies modeling while only marginally limiting the maximum control authority.

Because of the cross product in Equation 3, any $\vec{L}_{\mathcal{B}}$ can be produced by a $\vec{m}_{\mathcal{B}}$ where $\vec{m}_{\mathcal{B}} \cdot \vec{B}_{\mathcal{B}} = 0$, i.e. only dipoles in the plane normal to $\vec{B}$ need be considered. This reduces the dimensionality of the control space from 3 to 2. A mapping from a normalized, 2-dimensional control input $\vec{u} \in U = \bar{D}_1$ (the unit disk) to a dipole in the $\vec{B}$-normal plane is defined by generating a basis orthogonal to $\vec{B}$

$$\begin{gathered} \hat{C}_x = \text{unit}((\vec{B} \times \hat{x}_{\mathcal{I}}) \times \vec{B}) \\ \hat{C}_y = \text{unit}(\vec{B} \times \hat{C}_x) \\ \vec{m}(\vec{u} = (u_1, u_2)) = m_{max}(u_1\hat{C}_x + u_2\hat{C}_y) \end{gathered} \tag{4}$$

For a fixed $\vec{B}$, Equation 4 yields an inertially constant $\vec{m}$ and thus and inertially constant torque $\vec{L}$; that is, the function is not dependent of satellite state.

2.3 State Constraints

Keepout and keep-in constraints are defined to simulate instruments that have persistent pointing requirements (e.g. a camera that is damaged if pointed at the sun, or a star tracker that must always view a certain region of space). These are defined by tuple

$$K = (\vec{b}_{\mathcal{B}}, \vec{k}_{\mathcal{I}}, \phi) \tag{5}$$

a body-fixed vector $\vec{b}_{\mathcal{B}}$ representing the instrument, an intertially fixed vector $\vec{k}_{\mathcal{I}}$ representing the center of the keepout cone, and cone half-angle ϕ. Keep-in constraints can be formulated as keepout constrains with

$\vec{k}_{keepout} = -\vec{k}_{keep-in}$ and $\phi_{keepout} = \pi - \phi_{keep-in}$. For a set of keepouts $\mathcal{K}$, the attitudes q that the satellite is allowed to take must satisfy

$$\angle(K.\vec{b}_{\mathcal{B}}, q \otimes K.\vec{k}_{\mathcal{I}}) > K.\phi \ \forall \ K \in \mathcal{K} \tag{6}$$

where $q \otimes \vec{v}$ represents the rotation of $\vec{v}$ by q, $q\vec{v}q^{-1}$. That is, the angle between the $\vec{b}$ and $\vec{k}$ must be greater than the cone's half-angle ϕ.

A maximum angular velocity constraint ω_{max} is also imposed on the system

$$|\vec{\omega}| < \omega_{max} \tag{7}$$

as is typical for satellites.

2.4 Distance Metric

For kinodynamic planning, a distance metric $d : X^2 \to \mathbb{R}^+$ is defined

$$d(x_1, x_2) = \text{eigenangle}(q_1, q_2) + \alpha|\vec{\omega}_1 - \vec{\omega}_2| \tag{8}$$

noting that for computational efficiency

$$\text{eigenangle}(q_1, q_2) = |2\cos^{-1}(\vec{q}_1 \cdot \vec{q}_2)| \tag{9}$$

where $\cdot$ is the dot product of the quaternions expressed as vectors of their components. α is a tunable parameter with units of seconds that weighs the angular velocity contribution relative to the angle contribution.

2.5 Maneuver Planning Problem

With the kinodynamics of the system defined, the problem objective is to find a maneuver that causes the system to evolve from $\vec{x}_{start}$ to $\vec{x}_{goal}$ while conforming to state constraints. A maneuver $\mathcal{M}$ takes the form of a control schedule that lists the duration τ for which to execute different control functions

$$\mathcal{M} = [(\tau_1, \vec{u}_1(\vec{x})), (\tau_2, \vec{u}_2(\vec{x})), ..., (\tau_n, \vec{u}_n(\vec{x}))] \in (\mathbb{R} \times (X \to U))^n \tag{10}$$

which results in a state trajectory $\vec{x}(t)$. The maneuver is considered successful if the final state is within some convergence radius ϵ of the goal state $\vec{x}_{goal}$

$$d\left(\vec{x}\left(t_f\right), \vec{x}_{goal}\right) < \epsilon \tag{11}$$

and if for all $t \in [0, t_f]$, all state constraints are obeyed: namely, the keepout constraints in Equation 6 and the maximum angular velocity constraint in Equation 7.

3 Methods

Goal-biased RRT is presented, followed by two proposed modifications to improve performance on underactuated systems. Methods for generating control functions are also considered.

3.1 Goal-Biased RRT

The standard goal-biased RRT algorithm (Algorithm 1) [4] is used as a starting point for modified algorithms presented in this paper. Goal-biased RRT grows a search tree (with vertices V, edges E, and corresponding control functions U) in state space through the repetition (N times) of two basic steps: 1) selecting a state space node from V, the tree's vertices, then 2) from that node, propagating a random control function $\vec{u}(\vec{x})$ for a random duration. To select a node $\vec{x}_{near}$ to branch from, a random point $\vec{x}_{rand} \in X$ is sampled from the state space with a uniform distribution, and the node nearest to $\vec{x}_{rand}$ in V is selected as the branching point

$$\vec{x}_{near} \leftarrow \underset{\vec{x}}{\text{argmin}}(\|\vec{x} - \vec{x}_{rand}\| \textbf{ for } \vec{x} \in V) \tag{12}$$

Random control selection is discussed in more detail in subsection 3.4, but it is important that the control duration is also randomized $t \in [0, t_{max}]$ for completeness. With a state to start from and a control function to execute, PROPAGATE runs an ODE solver to evolve the dynamics defined in Equation 2, 3, and 4. After propagating for t, the solution is checked against the dynamic constraints and if it is *valid*, the final state of propagation $\vec{x}_{new}$ is added to the search tree. If $\vec{x}_{new}$ is within ϵ of the goal (Equation 11), the program terminates with success.

To increase the rate at which the tree approaches the goal state $\vec{x}_{goal}$, the tree is goal-biased on some iterations with some probability p_{bias}. On these steps, $\vec{x}_{rand}$ is deterministically selected to be $\vec{x}_{goal}$, meaning that the tree grows from the closest node to the goal state. Then, a control function is selected that best forces the state towards the goal. Since the algorithm does not have prior knowledge of the system's dynamics with respect to the selected control function, k_{bias} random controls are propagated, and the one among them that is produces a dynamically-valid final state closest to the goal state is added to the tree.

Algorithm 1 Goal-Biased RRT solver.

```
1: procedure GOALBIASRRT($N, p_{bias}, k_{bias}, \vec{x}_{start}, \vec{x}_{goal}, \epsilon$)
2:     $V \leftarrow \{\vec{x}_{start}\}$                                   ▷ Collection of nodes
3:     $E \leftarrow \{\}$                                                  ▷ Collection of edges
4:     $U \leftarrow \{\}$                          ▷ Collection of control functions used at each edge
5:     for $1 : N$ do
6:         if RAND() $< p_{bias}$ then                  ▷ Bias toward goal with probability $p_{bias}$
7:             $\vec{x}_{near} \leftarrow \underset{\vec{x}}{\text{argmin}}(\|\vec{x} - \vec{x}_{goal}\|$ for $\vec{x} \in V)$      ▷ Find the closest node to $\vec{x}_{goal}$
8:             $d_{test} \leftarrow \infty$
9:             for $1 : k_{bias}$ do                 ▷ Find the best of $k_{bias}$ random control functions
10:                $t_{test}, \vec{u}_{test}(\vec{x}) \leftarrow$ RANDOMCONTROLFN()
11:                $\vec{x}_{test}, valid \leftarrow$ PROPAGATE($\tau_{test}, \vec{x}_{near}, \vec{u}_{test}(\vec{x})$)
12:                if $valid$ and $d(\vec{x}_{test}, \vec{x}_{goal}) < d_{test}$ then
13:                    $t_{new}, \vec{u}_{new}(\vec{x}), \vec{x}_{new} \leftarrow t_{test}, \vec{u}_{test}(\vec{x}), \vec{x}_{test}$
14:                end if
15:            end for
16:        else                                       ▷ Otherwise explore in a random direction
17:            $\vec{x}_{rand} \leftarrow$ RANDIN($X$)                 ▷ Sample a random point in state space $X$
18:            $\vec{x}_{near} \leftarrow \underset{\vec{x}}{\text{argmin}}(\|\vec{x} - \vec{x}_{rand}\|$ for $\vec{x} \in V)$      ▷ Find the closest node to $\vec{x}_{rand}$
19:            $t, \vec{u}_{new}(\vec{x}) \leftarrow$ RANDOMCONTROLFN()
20:            $\vec{x}_{new}, valid \leftarrow$ PROPAGATE($\tau, \vec{x}_{near}, \vec{u}(\vec{x})$)
21:        end if
22:        if $valid$ or $d_{test} \neq \infty$ then           ▷ Add to tree if propagation obeyed constraints
23:            $V \leftarrow V \cup \{\vec{x}_{new}\}$
24:            $E \leftarrow E \cup \{(\vec{x}_{new}, \vec{x}_{near})\}$
25:            $U \leftarrow U \cup \{(\tau, \vec{u}_{new}(\vec{x}))\}$
26:            if $d(\vec{x}_{new}, \vec{x}_{goal}) < \epsilon$ then                    ▷ Check if within $\epsilon$ of goal
27:                return $\mathcal{M} \leftarrow$ BUILDMANEUVER($V, E, U, \vec{x}_{new}$)        ▷ Return maneuver $\mathcal{M}$
28:            end if
29:        end if
30:    end for
31: end procedure
```

3.2 Random Trees in Underactuated Systems

Highly underactuated systems like the magnetorquer-controlled satellite are challenging for goal-biased RRT. The biasing procedure assumes that goal state $\vec{x}_{goal}$ is probably easily reachable from the closest node $\vec{x}_{near}$ on the search tree; that is, the sequence of controls needed to reach $\vec{x}_{goal}$ is shorter from $\vec{x}_{near}$ than from any other node $\in V$, with the best case being that $\vec{x}_{goal}$ is - in the controls sense - reachable from $\vec{x}_{near}$ with a single control law $\vec{u}(\vec{x})$. Biased steps are only useful if it is reasonably likely that growing the tree from the current best node toward the goal is possible.

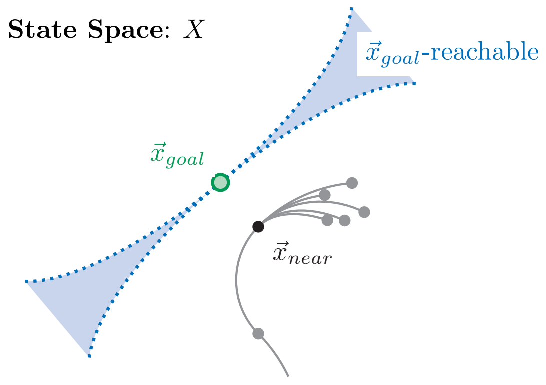

Figure 1: Goal-biased RRT's biased steps from $\vec{x}_{near}$ often get stuck in local minima when $\vec{x}_{goal}$ is reachable only from a small subset of the state space.

In underactuated systems, this is not necessary the case. The biased branch can become caught in a local minimum, at which point there exists no $(\tau, \vec{u}(\vec{x}))$ that when propagated from $\vec{x}_{near}$ produces a point closer to $\vec{x}_{goal}$ than $\vec{x}_{near}$. When this happens, each biased iteration wastes computational effort by grows the tree in a generally useless way, as depicted in Figure 1. Due to probabilistic completeness guarantees of RRT [4], the non-biased, random exploration steps will eventually find a state closer to the goal than a best state stuck in a local minimum; however, given the high dimensionality of the state space and the relatively small $\vec{x}_{goal}$-reachable region, it is likely that the new best point will also get stuck in a local minimum. As such, while finding a solution is probabilistically guaranteed, it can require a prohibitive number of iterations for some underactuated systems.

3.3 Bonsai-RRT Algorithm

A modification to the standard RRT algorithm is presented to overcome the challenges associated with search trees for underactuated systems. The modified algorithm is called Bonsai-RRT, named for its strategy of selecting a relatively good branch that may be stuck in a local minimum near the goal and "bending" it to guide it closer to the goal, as illustrated in Figure 2. Bonsai-RRT selects a branch that is likely stuck at a local minimum, then modifies the control function at a random segment and repropagates the remainder of the branch with the hope of finding an improved local minimum. This procedure leverages information already obtained by the tree of how to traverse the complex, high-dimensional space when improving branches close to the goal, significantly reducing the time spent exploring the space.

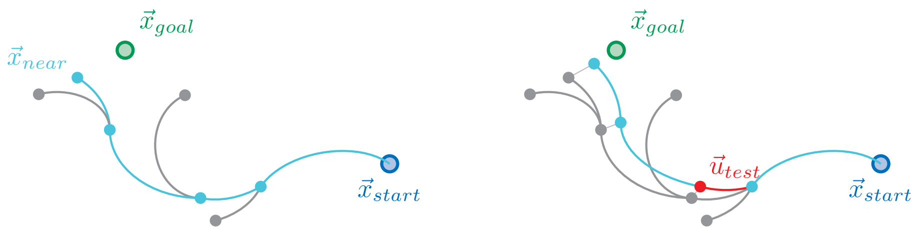

Figure 2: **Left:** A good branch is randomly selected by picking a leaf $\vec{x}_{near}$ close to the goal. **Right:** A new control $\vec{u}_{test}$ is propagated for a randomly selected segment; the same controls are used for the rest of the branch. If the new branch yields improved performance, it is added to the search tree.

Algorithm 2 engages in the branch-bending procedure with probability p_{bend}. The current best branch in the tree is selected to be bent, which is the branch between $\vec{x}_{start}$ and the $\vec{x}_{near}$ closest to $\vec{x}_{goal}$, as described in Equation 12. The maneuver $\mathcal{M}_{branch}$ is the series of control functions and durations that leads from $\vec{x}_{start}$ to $\vec{x}_{near}$; a randomly selected ith element of $\mathcal{M}_{branch}$ is replaced by sampling a new control function segment $(\tau_{test}, \vec{u}_{test}(\vec{x}))$ to yield $\mathcal{M}_{test}$, a new maneuver that may yield a new closest state. It is hypothesized that only moderate changes to the control function are beneficial, since too large of a change will likely modify the end state farther away from the goal. Thus while the new control segment could be a completely random sample of the duration and control function space, it is found that limiting τ_{test} to $\tau_i \pm 10\%$ is beneficial. Additionally, a restriction on modification to the PD control function is discussed in subsubsection 3.4.2.

The test maneuver $\mathcal{M}_{test}$ is propagated to produce new vertices V_{test} and checked against dynamic constraints. If the new maneuver is valid, the node $\vec{x}_{best}$ in V_{test} closest to $\vec{x}_{goal}$ is identified. If $\vec{x}_{best}$ is an improvement, the branch V_{test} is added to the tree; otherwise, a new $\mathcal{M}_{test}$ is generated (up to k_{bend} times). The implementation in Algorithm 2 only adds the bent branch to the tree if improves the global minimum distance to goal (line 16: $d(\vec{x}_{best}, \vec{x}_{goal}) < \min(d(\vec{x}, \vec{x}_{goal}), \vec{x} \in V)$), but alternative formulations that add the branch under less strict conditions could be considered, with the downside of more quickly expanding the tree.

Bonsai-RRT retains the completeness guarantees of RRT because standard RRT exploration steps are always responsible for a nonzero fraction of new nodes added to the tree.

Algorithm 2 Bonsai-RRT solver, modified from Algorithm 1, GoalBiasRRT.

```
1: procedure BonsaiRRT($N, p_{bend}, k_{bend}, \mu_{bend}, ...$)
2:     $V \leftarrow \{\vec{x}_{start}\}$
3:     $E \leftarrow \{\}$
4:     $U \leftarrow \{\}$
5:     for $1 : N$ do
6:         if Rand() $< p_{bend}$ then                                  ▷ Perform Bonsai bending operation
7:             $\vec{x}_{near} \leftarrow \underset{\vec{x}}{\mathrm{argmin}}(\|\vec{x} - \vec{x}_{goal}\|$ for $\vec{x} \in V)$
8:             $\mathcal{M}_{branch} \leftarrow$ BuildManeuver($V, E, U, x_{near}$)      ▷ Select the current best branch to modify
9:             for $1 : k_{bend}$ do
10:                $\mathcal{M}_{test} \leftarrow \mathcal{M}_{branch}$
11:                $i \leftarrow$ Rand($1 : \mathrm{len}(\mathcal{M}_{branch})$)                ▷ Select a control segment to change
12:                $(\tau_{test}, \vec{u}_{test}(\vec{x})) \leftarrow$ RandomControlFn()      ▷ Optional: Restrict new $\tau$ to $\tau_i \pm 10\%$
13:                $\mathcal{M}_{test}[i] \leftarrow (\tau_{test}, \vec{u}_{test}(\vec{x}))$
14:                $V_{test}, valid \leftarrow$ PROPAGATE($\vec{x}_{start}, \mathcal{M}_{test}$)
15:                $\vec{x}_{best} \leftarrow \underset{\vec{x}}{\mathrm{argmin}}(\|\vec{x} - \vec{x}_{goal}\|$ for $\vec{x} \in V_{test})$
16:                if $valid$ and $d(\vec{x}_{best}, \vec{x}_{goal}) < \min(d(\vec{x}, \vec{x}_{goal}), \vec{x} \in V)$ then   ▷ Add to tree if improvement
17:                    $V \leftarrow V \cup V_{test}$
18:                    $E \leftarrow E \cup \{(\vec{x}_i, \vec{x}_{i+1}), \vec{x}_i \in V_{test}\}$
19:                    $U \leftarrow U \cup \mathcal{M}_{test}$
20:                    if $d(\vec{x}_{best}, \vec{x}_{goal}) < \epsilon$ then                 ▷ Check for solution
21:                        return $\mathcal{M} \leftarrow$ BuildManeuver($V, E, U, \vec{x}_{best}$)
22:                    end if
23:                    break
24:                end if
25:            end for
26:        else
27:            Perform lines 6-29 of GoalBiasRRT                        ▷ Grow tree as normal
28:        end if
29:    end for
30: end procedure
```

3.4 Control Command Randomization

The function RandomControlFn returns a tuple $(\tau, \vec{u}(\vec{x}))$ that forms a segment of a maneuver. Recall that the duration of the control function τ is randomized between 0 and some τ_{max}, as is required for completeness. Two methods of generating random control functions are considered.

3.4.1 Constant Control

Recall the mapping constructed from the unit disk to the control space in Equation 4. The simplest random control function is a constant control $\vec{u}(\vec{x}) = \vec{u}_{rand}, \vec{u}_{rand} \in \bar{D}_1$. While the simplicity is appealing, this function will struggle to reach goal states: Even if $\vec{x}_{goal}$ is reachable with a constant, nonzero control input, unless τ is selected to be a certain value exactly, the goal state will be undershot or overshot.

3.4.2 Proportional-Derivative Control

A simple Proportional-Derivative (PD) controller is proposed as alternative to using constant control inputs. The controls are designed to drive the system towards $\vec{x}_{rand}$ on non-biased steps and towards $\vec{x}_{goal}$ on biased steps. Primarily, this solves the overshooting problem encountered using constant control functions, eliminating the need for a specific duration to be randomly selected.

The PD controller attempts to drive the attitude and angular velocity to some $\vec{x}_{targ}$. This can be reformulated into a linear control law, with state vector that is a combination of the vector part of the error quaternion and the angular velocity error.

$$\vec{y} = [[i\ j\ k] \cdot q * q_{targ}^{-1}; \vec{\omega} - \vec{\omega}_{targ}] \tag{13}$$

and control matrix with proportional and derivative gains K_P and K_D

$$\mathbf{T} = \begin{bmatrix} I_{3\times3}K_P & 0 \\ 0 & I_{3\times3}K_D \end{bmatrix} \tag{14}$$

where the desired control torque is

$$\vec{L}_{\mathcal{B},des} = -\mathbf{T}\vec{y} \tag{15}$$

Since the achievable torque is limited to the $\vec{B}$-normal plane, $\vec{L}_{\mathcal{B},des}$ is projected onto that plane

$$\vec{L}_{\mathcal{B},proj} = \vec{L}_{\mathcal{B},des} \cdot \hat{C}_x + \vec{L}_{\mathcal{B},des} \cdot \hat{C}_y \tag{16}$$

which is achieved with a magnetic dipole of

$$\vec{m}_{des} = \frac{1}{|\vec{B}|}\left((-\vec{L}_{\mathcal{B},des} \cdot \hat{C}_y)\hat{C}_x + (\vec{L}_{\mathcal{B},des} \cdot \hat{C}_x)\hat{C}_y\right) \tag{17}$$

This is then converted into the control space U and clamped to obey physical limitations

$$\vec{u} = \begin{cases} [\hat{C}_x\ \hat{C}_y] \cdot \vec{m}_{des}/m_{max} & \text{if } |\vec{m}_{des}| \leq m_{max} \\ [\hat{C}_x\ \hat{C}_y] \cdot \text{unit}\,(\vec{m}_{des}) & \text{if } |\vec{m}_{des}| > m_{max} \end{cases} \tag{18}$$

Selection of the gains K_P and K_D can be fairly arbitrary since the primary objective is to craft a control function that tends towards zero control effort when approaching the goal state $\vec{x}_{goal}$, which any positive gains will achieve. During non-biased exploration steps, the combination of random gains and random target states yields controllers that frequently saturate and target non-equilibrium states; this behavior is no worse than using random constant controls for exploration.

When performing the bending operation in Bonsai-RRT, only the gains K_P and K_D and the duration τ were altered; the target state $\vec{x}_{targ}$ for a given function remained the same. This choice was made under the conjecture that introducing too much variation in bending steps would negatively affect performance by spending time propagating maneuvers that have little likelihood of reaching the goal state.

4 Results

The choice of physical and algorithm parameters is discussed. The performance of each algorithm is benchmarked under a fixed computation time limit. Search tree and solution quality and validity are presented for representative runs of each algorithm.

4.1 Simulation Parameters

The problem is considered for a 3U CubeSat in LEO, but the algorithm could be applied to other satellite architectures. Since the orbit of the satellite is a known $\vec{r}(t)$ and the Earth's magnetic field can be modeled as a function of position $\vec{B}(\vec{r})$, the magnetic field can be expressed as a function of time $\vec{B}(t)$. Since the satellite is operating in LEO,

$$|\vec{B}| \in [30, 60] \ \mu\text{T} \tag{19}$$

The resulting maneuvers are short relative to the orbital period so the further approximation that $\vec{B}$ is constant can be made, though a nonconstant $\vec{B}(t)$ would not affect the algorithms presented. The system's physical parameters are listed in Table 1. For constrained simulations presented in this paper, the keepout constraints in Table 2 are used.

Parameter		Value	Rationale
Inertia Tensor	$\boldsymbol{J}$	$I_{3\times3} \cdot \begin{bmatrix} 41.87 & 41.87 & 6.67 \end{bmatrix}$ g $\cdot$ m^2	Typical 3U CubeSat
Max. Dipole	m_{max}	0.66 A $\cdot$ m^2	Commercially available magnetorquers for CubeSats
Max. Ang. Rate	ω_{max}	$5°/\text{s} = 0.087$ rad/s	Arbitrary
Dist. Metric Param.	α	30 s	Scales angular velocity and angle contribution to same order: $\alpha\omega_{max} \approx \pi$ rad
Mag. Field	$\vec{B}(t)$	$50\hat{z}_{\mathcal{I}}$ µT	Constant magnetic field approximation

Table 1: Physical parameters used in simulations.

Keepout	Body Vector $\vec{b}_{\mathcal{B}}$	Inertial Vector $\vec{k}_{\mathcal{I}}$	Half Angle ϕ [°]
Yellow	$-\hat{z}_{\mathcal{B}}$	$-\hat{x}_{\mathcal{I}}$	50
Orange	$(\hat{x}_{\mathcal{B}} + \hat{y}_{\mathcal{B}})/\sqrt{2}$	$-\hat{y}_{\mathcal{I}}$	20

Table 2: Sensor keepout constraints used in simulations.

Each combination of algorithm $\in$ [GoalBiasRRT, BonsaiRRT] and control function $\in$ [Constant, PD] will be evaluated. Table 3 lists the parameters used for each algorithm. Note the unusually high value selected for p_{bias}, which is typically closer to 0.05; the high value was selected since the system is high dimensional with relatively few obstacles, so branching towards the goal is more beneficial than thoroughly exploring the space. Table 4 lists parameters used in control function randomization. Finally, the optional $\tau_{test} = \tau_i \pm 10\%$ restriction is used in Bonsai-RRT.

	RRT	**Bonsai**
p_{bias}	0.66	0.66
k_{bias}	25	25
p_{bend}	-	0.03
k_{bend}	-	50

Table 3: GoalBiasRRT and BonsaiRRT parameters.

Both	τ_{max}	25 s
PD Control	K_P	[0, 0.005]
	K_D	[0, 0.03]

Table 4: Control function generation parameters.

4.2 Performance Comparisons

The algorithms were implemented in Julia (see Appendix A) and executed in parallel each on a single core of a 2.4 GHz Quad-Core Intel Core i5 processor. For a fair comparison, each algorithm was allowed to iterate for 300 seconds, as opposed executing until a convergence radius ϵ or maximum iteration number N was reached. The solution quality was returned for the node in the search tree closest to $\vec{x}_{goal}$ after the execution timeout was reached. Each trial uses the same constrained attitude maneuver in which the satellite is tasked with a rest-to-rest maneuver 180° about $\vec{B}$ ($d_{start} = \pi$) with keepouts in Table 2. Median results for 100 trials per algorithm are reported in Table 5, with distributions given in Figure 3.

	Control Fn.	d	IQR d	Eig. Ang. [°]	$\Delta\omega$ [°/s]	# Nodes
RRT	Constant	0.61	0.55	15.63	0.50	1.5×10^6
	PD	0.82	0.47	26.48	0.59	0.7×10^6
Bonsai	Constant	0.16	0.26	1.03	0.26	0.8×10^6
	PD	0.01	0.04	0.01	0.02	0.2×10^6

Table 5: Median performance of each search algorithm on a single constrained rest-to-rest attitude maneuver problem. $N_{trials} = 100$; each algorithm is given 300 seconds of execution time. Results are reported for the node closest to $\vec{x}_{goal}$.

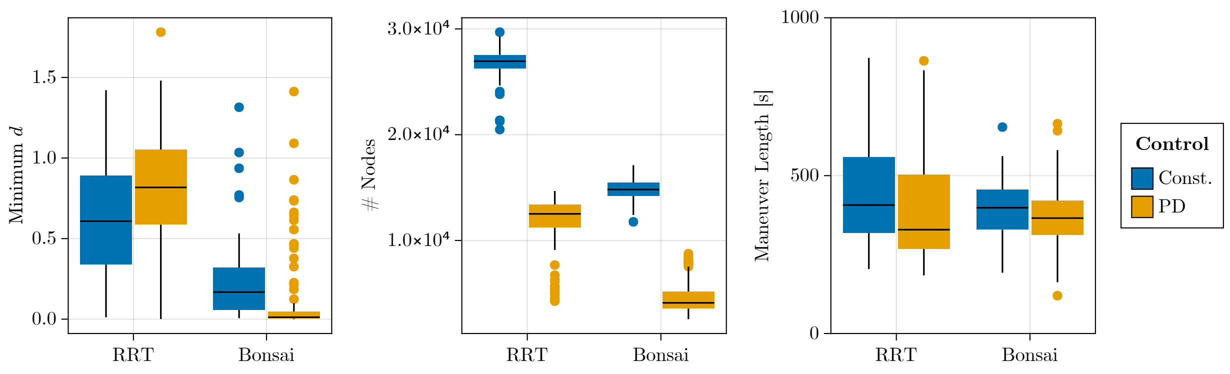

Figure 3: Distributions of minimum d, node count, maneuver duration to minimum d for data tabulated in Table 5. Some RRT maneuver length outliers $\in [1000, 3000]$ s have been omitted.

Figure 3 shows that Bonsai-RRT consistently outperforms standard goal-biased RRT for this problem, even with an uninformed control method, constant control. Bonsai-RRT with PD control is the only combination of tree search algorithm and control law that tends towards zero, while the others produce a quasinormal distribution about a nonzero value. Bonsai-RRT with PD control achieves this superior performance even with the smallest search tree and thus lowest memory requirement. Since this benchmark was time limited and Bonsai-RRT with PD control requires the most computation per iteration due to evaluation of the PD control law and the bending procedure, fewer nodes were added in the allotted time.

Each of the solution methods produce comparably long maneuvers. Since neither algorithm attempts to find optimal solutions, there is no expectation for length. This implies that the algorithms are all finding similar solutions, likely due to the strong goal bias. Since incomplete maneuvers are included in the figure, the minimum cost for a valid maneuver does not necessarily correspond to the lowest value in the boxplot.

For each algorithm, individual runs are analyzed to explain the shortcomings of standard RRT and constant control, and how Bonsai-RRT with PD control overcomes them. subsubsection 4.2.3 explores the performance of successful maneuvers using Bonsai-RRT with PD control. These runs target a convergence radius of $\epsilon = 0.01$ with a maximum of $N = 10000$ iterations.

4.2.1 Solution Quality: RRT, Constant Control[1]

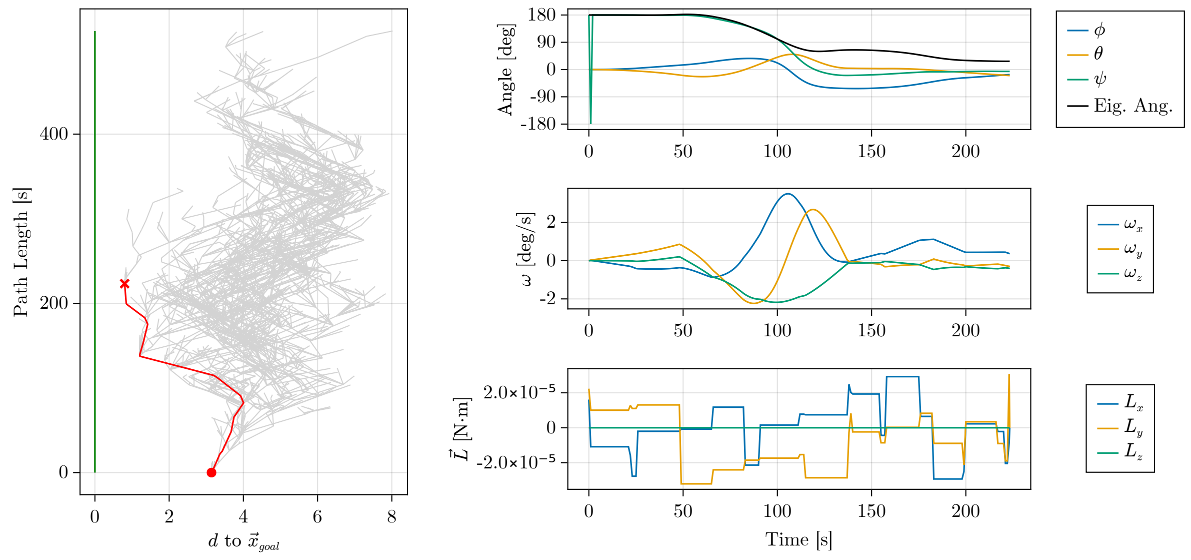

Figure 4: Search tree and solution dynamics for a single run using RRT with constant control, $N = 10000$.

RRT with constant control is examined in Figure 4. The search tree spends the majority of its effort exploring a region of space that is not relevant to the problem. The issue depicted in Figure 1 is apparent here: The algorithm repeatedly attempts to grow the tree from a local minimum, but is never able to improve from that point. The plots of solution dynamics show the system struggling to converge to either the desired angle or angular rate, with a relatively large residual for both at the end of the maneuver.

4.2.2 Solution Quality: Bonsai-RRT, Constant Control

Using the Bonsai-RRT algorithm overcomes the issues observed in the standard RRT runs. The space is more densely explored around the branch that approaches the goal. Nodes near the solution branch where a local minimum was found but exited using the bending procedure are apparent in Figure 5.

The previously discussed shortcoming of constant control is apparent in the plot of the system dynamics. As the goal state is approached, the planner uses many short control segments to make small adjustments. Not only does this make convergence prohibitively slow as seen by the large residuals that remain after 10000 iterations, it produces a control function that would be impractical for physical use. This motivates the implementation of the PD control function.

4.2.3 Solution Quality: Bonsai-RRT, PD Control

Bonsai-RRT with PD control produces satisfactory results, as shown in Figure 6. The tree exhibits focused exploration around the known good path. As the goal state is neared, the utility of the PD controller becomes apparent. Unlike with constant control, the control effort smoothly tapers to zero as the goal state

[1]RRT with PD control is not presented due to its qualitatively and quantitatively similar performance to RRT with constant control.

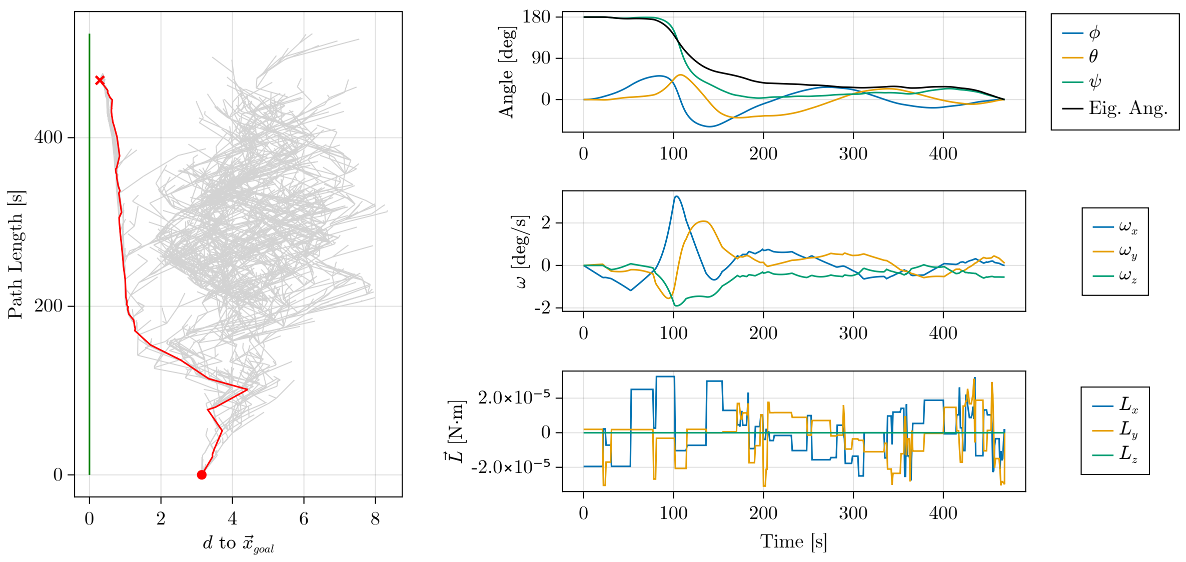

Figure 5: Search tree and solution dynamics for a single run using Bonsai-RRT with constant control, $N = 10000$.

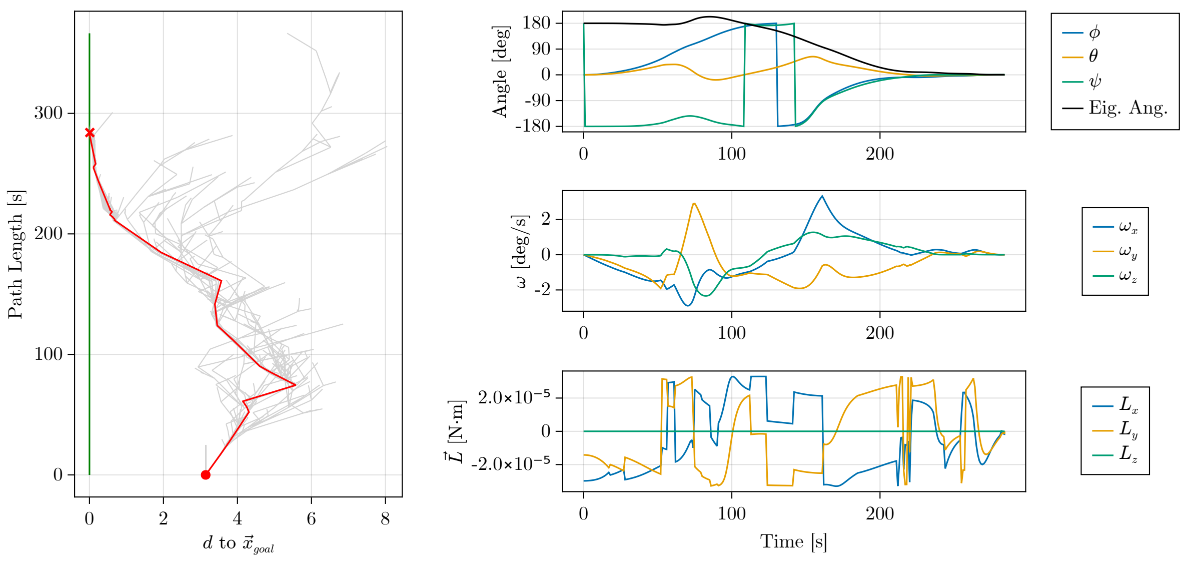

Figure 6: Search tree and solution dynamics for a single run using Bonsai-RRT with PD control, $\epsilon = 0.01$.

is reached. Figure 7 shows convergence to the goal state with greater detail: the final state has a residual of $< 1°$ and $< 0.01°/\text{s}$.

Compliance with keepout constraints is graphically validated in Figure 7. Neither satellite-fixed instrument points at its respective inertially-fixed keepout over the course of the maneuver.

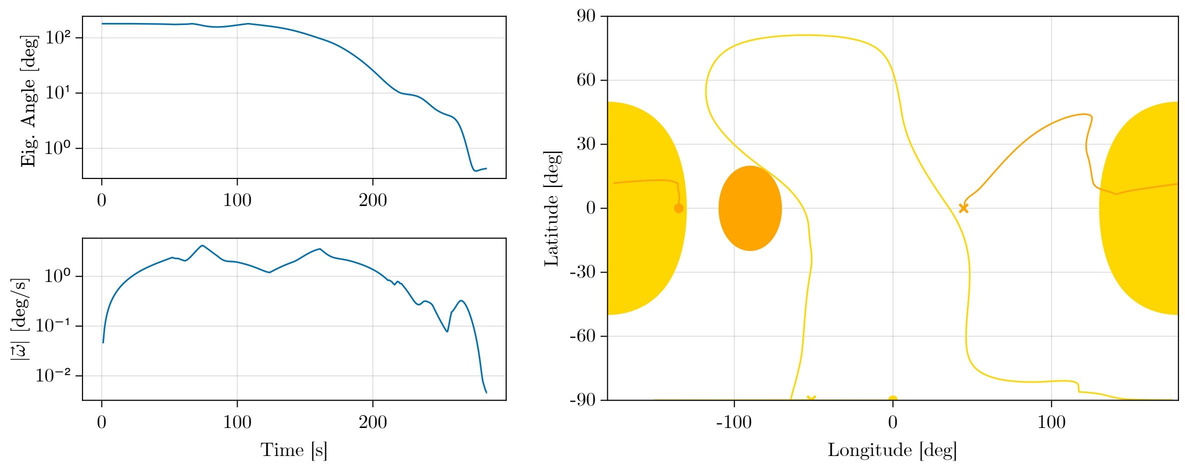

Figure 7: **Left:** Log-scaled dynamics showing quality of convergence. **Right:** Paths of instruments relative to respective keepout regions.

As the one algorithm that consistently converges to a high-quality, further performance benchmarks are evaluated for Bonsai-RRT with PD control. As with the individual runs presented above, the algorithm is given $N = 10000$ iterations to converge to a solution within $\epsilon = 0.01$. Performance is show in Figure 8 for $N_{trials} = 250$.

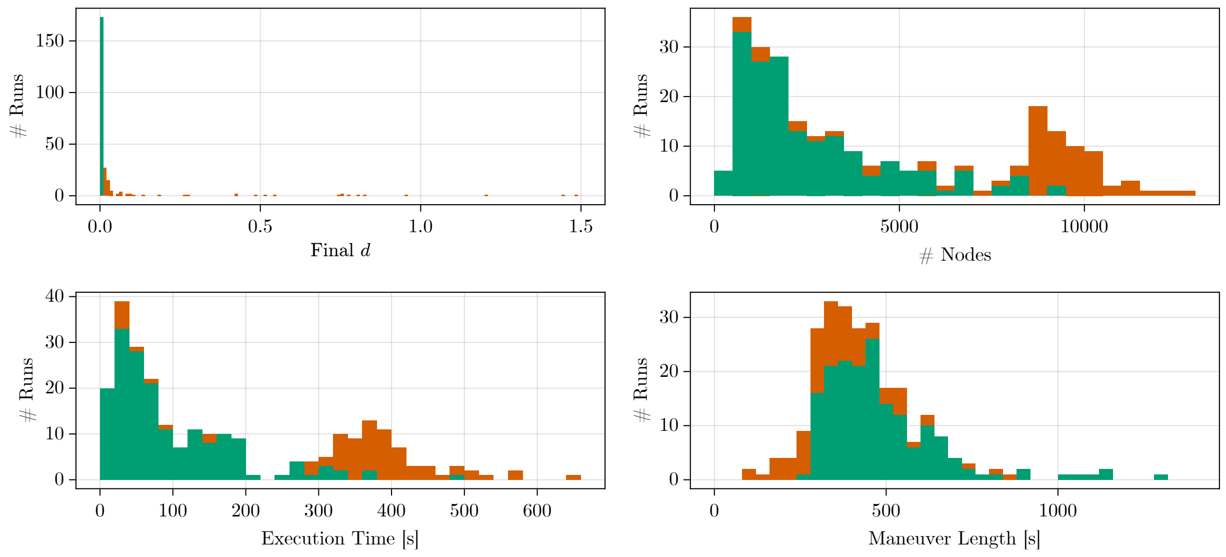

Figure 8: $N_{trials} = 250$ benchmark of Bonsai-RRT with PD control; $\epsilon = 0.01$, $N = 10000$. Runs that successfully converge are in green, others are in red.

With a relatively low iteration cap, 69% of runs reach the convergence radius of $\epsilon = 0.01$. These successful runs tend to find a solution in under 200 seconds and return maneuvers of a relatively consistent length, with occasional outliers. 92% of runs terminated within 0.1 of the goal, implying that the algorithm is tending towards a solution in most cases; however, the bimodal distribution[2] for execution time implies that if a solution is not found within the first 5000 iterations, a solution becomes harder to find due to the increased tree size. While some tree pruning method could be implemented to limit this effect, it would be more pragmatic to simply terminate after 5000 iterations and restart the algorithm, hoping that a solution will be found more rapidly..

5 Discussion

The solutions to the magnetorquer-only attitude control problem found using Bonsai-RRT with PD control mark an extreme improvement over the state of the art [1][2][3], decreasing maneuver times from a few orbits to a few minutes. Additionally, this algorithms successfully finds solutions that obey keepout constraints for sensitive instruments.

In solving this problem, a novel RRT variation is introduced that is capable of quickly finding solutions to highly underactuated kinodynamic planning problems with time-varying dynamics and narrowly reachable goal states without additional analysis of the system.

5.1 Challenges of Optimal Path Planning

Optimal path planning problems aim to find the path between the start and goal that minimizes some cost function. Common cost functions include maneuver duration or control effort expended. A common tree-based algorithm for optimal kinodynamic planning is SST [10]. Briefly, SST works by growing the search tree from the best node among those in a δ_s radius; that is, if two nodes $\vec{x}_n$ and $\vec{x}_m$ are δ_s-close $d(\vec{x}_n, \vec{x}_m) < \delta$ and the cost of the trajectory to one node is less than that to another $\text{costto}(\vec{x}_n) < \text{costto}(\vec{x}_m)$, the tree will prune the node with the higher cost $\vec{x}_m$.

Inherently, SST is challenged by this problem. SST finds an optimal trajectory up to some δ-similarity, requiring that some function of the pruning radius parameters δ_s and δ_{BN} be smaller than the δ-robustness of the problem. The narrow reachability region of $\vec{x}_{goal}$ leads to a low dynamic clearance value for the problem, implying that the problem has a small δ-robustness value. Because of this only a small pruning radius is allowable. This reduces the performance of SST considerably.

δ-robustness aside, adapting Bonsai-RRT to use methods employed by SST presents additional challenges. Naively pruning close nodes that have higher cost does not make use of the difficult-to-obtain information about reaching the goal state. One may try to leverage already-known information on how to reach the goal when pruning, using a similar strategy to the branch bending procedure: For any $\vec{x}_n$ that causes $\vec{x}_m$ to be pruned where $\vec{x}_m$ has a "good" branch as a child, repropagate the controls of $\vec{x}_m$'s branch starting from $\vec{x}_n$. With a small enough pruning radius, the final state of the repropagation would be similar to the branch's original final state; however, it may be unlikely that in a reasonable time the tree will randomly add a node close enough to another node for this procedure to occur. This is supported by the distance vs. path length plot in Figure 6: There are already very few nodes with a similar distance to goal and shorter path length than those in the final path; it is unlikely that any candidate nodes that satisfy this necessary condition are actually close to the path in state space.

Finally, it is reasonable to consider that the solutions produced by Bonsai-RRT are not considerably sub-optimal. Maneuver length across trials is found in a relatively tight distribution, without outliers on the lower end (Figure 8). Since control effort is usually saturated, these maneuvers probably do not represent a class of especially slow maneuvers. As noted prior, the lack of nodes in the region below the solution implies that exploration is rarely finding nodes that potentially could produce a faster solution, which could indicate that they do not exist.

5.2 Future Work

Aside from designing an optimal variant of Bonsai-RRT, further work is necessary to characterize, formalize, and improve Bonsai-RRT. Bonsai-RRT excels at solving problems with a narrowly reachable goal state, high dimensionality, poor controlability, and time varying systems without a linearization of the system dynamics.

[2]Really, the distribution is very right-skewed but the iteration limit lumps the long tail into a second mode.

The algorithm should be evaluated for systems with more complex obstacle topology, as the keepout regions in this problem are fairly simple. Algorithms like [6] tend to focus on problems with more complicated state-space obstacles; it is unknown how Bonsai-RRT would respond to this common challenge.

A possible improvement to the algorithm would be implementing an intelligent method of triggering the bending maneuver, as it currently happens stochastically. A heuristic could be designed to determine if a node is likely a local minimum, and if so, the branch bending procedure could be triggered accordingly.

For the the magnetorquer actuated satellite problem, the solutions in the paper need to be considered for a more physically realistic system with disturbance torques and uncertainty. These semi-open-loop control solutions may be too sensitive to uncertainty to be useful without an closed-loop correction procedure. The computation effort is also excessive for use on current satellites. Since the algorithm often takes minutes to find a satisfactory solution which is tied to a specific maneuver start epoch, planning for the maneuver must begin well before the spacecraft intends to execute the maneuver. Finally, algorithm hyperparameters need to be thoroughly searched to yield the best performance for this problem.

6 Conclusion

Using an RRT-based approach for the mangetorquer-only attitude control problem yields solutions that are two orders of magnitude faster than previous solutions to the problem [1][2][3], yielding truly agile maneuvers with minimally complex hardware with additional constraints. While further progress is required to make this method fit for flight, this paper demonstrates a significant improvement to a dormant problem.

The modifications to RRT that are introduced in this paper offer a powerful tool to solve highly under-actuated kinodynamic planning problems with time-varying dynamics and narrowly reachable goal states without additional analysis of the system. With further study, this algorithm could be applied to a variety of difficult motion planning problems with these properties.

References

[1] A. C. Stickler and K. Alfriend, "Elementary magnetic attitude control system," *Journal of Spacecraft and Rockets*, vol. 13, no. 5, pp. 282–287, May 1976, ISSN: 0022-4650, 1533-6794. DOI: `10.2514/3.57089`. [Online]. Available: `https://arc.aiaa.org/doi/10.2514/3.57089` (visited on 10/28/2022).

[2] L. Musser and W. L. Ebert, "Autonomous spacecraft attitude control using magnetic torquing only," presented at the Estimation Theory Symposium, 1989, p. 16.

[3] M. F. Erturk and C. Hajiyev, "Magnetic torquers only attitude control of a 3u cube satellite," *WSEAS Transactions On Signal Processing*, vol. 18, pp. 128–133, Jun. 24, 2022, ISSN: 2224-3488, 1790-5052. DOI: `10.37394/232014.2022.18.18`. [Online]. Available: `https://wseas.com/journals/sp/2022/a365114-015(2022).pdf` (visited on 10/28/2022).

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

[5] A. Yershova, L. Jaillet, T. Simeon, and S. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain: IEEE, 2005, pp. 3856–3861, ISBN: 978-0-7803-8914-4. DOI: `10.1109/ROBOT.2005.1570709`. [Online]. Available: `http://ieeexplore.ieee.org/document/1570709/` (visited on 12/01/2022).

[6] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *2009 IEEE International Conference on Robotics and Automation*, Kobe: IEEE, May 2009, pp. 2859–2865, ISBN: 978-1-4244-2788-8. DOI: `10.1109/ROBOT.2009.5152874`. [Online]. Available: `http://ieeexplore.ieee.org/document/5152874/` (visited on 11/29/2022).

[7] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany: IEEE, May 2013, pp. 5054–5061, ISBN: 978-1-4673-5643-5 978-1-4673-5641-1.

DOI: 10.1109/ICRA.2013.6631299. [Online]. Available: http://ieeexplore.ieee.org/document/6631299/ (visited on 12/01/2022).

[8] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, San Francisco, CA, USA: IEEE, 2000, pp. 995–1001, ISBN: 978-0-7803-5886-7. DOI: 10.1109/ROBOT.2000.844730. [Online]. Available: http://ieeexplore.ieee.org/document/844730/ (visited on 12/01/2022).

[9] R. Calaon and H. Schaub, "Constrained attitude maneuvering via modified-rodrigues-parameter-based motion planning algorithms," *Journal of Spacecraft and Rockets*, vol. 59, no. 4, pp. 1342–1356, Jul. 2022, ISSN: 0022-4650, 1533-6794. DOI: 10.2514/1.A35294. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.A35294 (visited on 10/31/2022).

[10] Y. Li, Z. Littlefield, and K. E. Bekris, *Asymptotically optimal sampling-based kinodynamic planning*, Feb. 5, 2016. arXiv: 1407.2896[cs]. [Online]. Available: http://arxiv.org/abs/1407.2896 (visited on 10/31/2022).

[11] F. L. Markley and J. L. Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control.* New York, NY: Springer New York, 2014, ISBN: 978-1-4939-0802-8. DOI: 10.1007/978-1-4939-0802-8. [Online]. Available: http://link.springer.com/10.1007/978-1-4939-0802-8 (visited on 10/21/2022).

A Code

Code used in the creation of this paper can be found at https://github.com/Mark2000/ende.