Rapid Trajectory Generation for Autonomous Spacecraft in Dynamic Environments

Griffin D. Francis
[*] *Florida State University, Tallahassee, Florida, 32310, USA*
Oscar Chuy
[†] *University of West Florida, Pensacola, Florida, 32514, USA*
Emmanuel G. Collins, Jr.
[‡] *University of Louisville, Louisville, Kentucky, 40208, USA*

This paper introduces a methodology for computationally efficient, direct generation of optimal trajectories for autonomous spacecraft rendezvous using sampling with time or distance cost functions. The approach utilizes a randomized A* algorithm called Sampling-Based Model Predictive Optimization (SBMPO) that exclusively samples the input space and propagates the dynamic model of the system. Appropriately selected heuristics enable fast computation of trajectories that end in zero relative velocity. Additionally, this paper presents an implementation of experience-based replanning for the generation of trajectories in dynamic environments. By referencing prior information in replanning, sufficient computational efficiency is achieved for the rapid recalculation of solution trajectories when subjected to observed changes in the planning space. Using a six degree-of-freedom relative motion spacecraft dynamic model, simulation results are illustrated for the generation of rendezvous feasible trajectories in cluttered, dynamic environments that adhere to realistic constraints.

Nomenclature

r	Position vector, m
v	Velocity vector, m/s
a	Acceleration vector, m/s^2
ω	Angular velocity vector, deg/s
q	Rotation quaternion vector
Θ	Rotation matrix
Ω	Kinematic quaternion matrix
m	Mass, kg
J	Inertia matrix, kg-m^2
u	Thrust control input vector, N
τ	Torque control input vector, Nm

I. Introduction

As concluded by recent NASA studies and the United State Space Policy, there is a dire need to develop technologies for the mitigation of orbital space debris.[1] Indicative of the present-day unsustainability of the space junk problem, it was suggested that a minimum of five mitigation missions would be required each year

[*] Graduate Research Assistant, Department of Mechanical Engineering.
[†] Assistant Professor, Department of Electrical and Computer Engineering.
[‡] Dean, J.B. Speed School of Engineering.
This research was sponsored by the Federal Aviation Administration under Cooperative Agreement 10-C-CST-FSU

– *starting in 2015* – to merely maintain the population of orbital debris observed in 2009.[2] The increasingly cluttered state of orbital space poses a threat to the future success of manned and unmanned space flight.

A major hurdle in achieving unmanned technologies for debris mitigation is the development of adequate trajectory planning methods. Intelligent autonomous navigation requires the generation of trajectories and control tracking inputs to traverse a route towards a goal. This intelligence is achieved via a planning algorithm that is, ideally, capable of quickly determining an optimal trajectory and control inputs subject to defined system constraints.

Partly motivated by the necessity to mitigate the threats associated with the growing presence of space debris, various autonomous spacecraft missions have been undertaken, including Orbital Express[3] and Spacecraft for the Universal Modification of Orbits/Front-end Robotics Enabling Near-term Demonstration (SUMO/FREND)[4] by DARPA and Demonstration for Autonomous Rendezvous Technology (DART)[5] by NASA. Future advancement of autonomous spacecraft navigation requires the development of innovative planning algorithms capable of generating trajectories in orbit.

With varying degrees of success, several methods have been attempted for solving the spacecraft trajectory planning problem. These methods are largely representative of the planning techniques already employed for robotic manipulators and wheeled vehicles. Spanning the various algorithms used for ground-based robots, mixed integer linear programming (MILP),[6,7] mixed integer nonlinear programming (MINLP),[8] potential functions,[9] rapidly exploring random trees (RRTs),[10,11] and calculus of variations[12] have been researched as possible solutions.

Critically necessary for aerospace work, the implementation of the system dynamic model allows for explicit consideration of propulsion constraints when planning control inputs. Many of these algorithms are hampered by an inability to deal with the nonlinear dynamics of the spacecraft and/or a lack of combined position and orientation planning capabilities. Furthermore, several are incapable of planning in the presence of obstacles. Only a few report computation times and these times typically prohibit the possibility of using the methods in real-time.

Within the realm of the more general topic of direct trajectory generation, recent research has led to approaches that employ optimization and randomly-generated graphs.[13,15,16] These methodologies seek to combine the computational efficiency obtained via graph-based algorithmic optimization and the global robustness of randomized, sampling-based graph formation. When implemented with A*-type optimization, random graph methods can incorporate a heuristic function that facilitates rapid computation of optimal trajectories. Unlike the randomized A* approach,[15,16] methods such as RRT*[13] do not utilize A* optimization and, as a result, do not benefit from the efficiency achieved by using the predictive behavior associated with an appropriately determined optimistic heuristic.

Sampling Based Model Predictive Optimization (SBMPO)[15,16] incorporates sampling and A* optimization to generate trajectories. SBMPO has been demonstrated as an effective and efficient trajectory planning technique for autonomous underwater vehicles (AUVs),[18] ground-based mobile robots,[19–21] and robotic manipulators.[22] Motivated by the promising results obtained using SBMPO in navigation and planning tasks for ground-based robots, ongoing SBMPO research is focused on the continued refinement of the algorithm and application to additional autonomous systems. In this work, the SBMPO algorithm is studied for application to autonomous spacecraft rendezvous and docking duties with the ultimate goal of real-time trajectory planning for orbital vehicles.

The efficiency of SBMPO is closely linked to the development of an appropriate optimistic A* heuristic. In this paper, two recently developed heuristic functions, that are based on the solutions to the minimum time[22] and minimum distance[23] control problems, are presented. These heuristics facilitate the rapid computation of trajectories that terminate with zero relative velocity, which is a requirement for spacecraft rendezvous trajectory planning. Afforded by the versatility of the planning methodology, a primary contribution of this paper is the efficient computation of rendezvous trajectories that accommodate real world constraints and situational limitations.

II. SBMPO Algorithm and Extensions

This section provides a brief description of SBMPO, associated methodologies and extensions, and identifies the details of the spacecraft rendezvous problem within the context of trajectory generation.

Algorithm 1 SBMPO Algorithm

function KEY(s)
  return $[min(g(s), rhs(s)) + h(s), min(g(s), rhs(s))]$
end function
function HEURISTICCOST(s, v_{goal})
  return minimum time to reach v_{goal}
end function
function INITIALIZE()
  $OPEN \leftarrow \emptyset$ ▷ $OPEN$ is a priority queue
  $g(v_{start}), g(v_{goal}), rhs(v_{goal}) \leftarrow \infty$
  $rhs(v_{start}), h(v_{goal}) \leftarrow 0$
  $h(v_{start}) \leftarrow HeuristicCost(v_{start}, v_{goal})$
  $OPEN \leftarrow OPEN \cup v_{start}$ ▷ prioritize using $key(v_{start})$
end function
function UPDATESTATE(v)
  if $v \neq v_{start}$ **then**
    $rhs(v) \leftarrow min_{v' \in Pred(s)}(g(v') + cost(v', s))$
    if $v \in OPEN$ and $g(v) \neq rhs(v)$ **then**
      $OPEN.update(v, Key(v))$
    else if $v \in OPEN$ and $g(v) = rhs(v)$ **then**
      $OPEN \leftarrow OPEN - v$
    else if $v \notin OPEN$ and $g(v) \neq rhs(v)$ **then**
      $OPEN \leftarrow OPEN \cup v$ ▷ prioritize using $key(v)$
    end if
  end if
end function
function GENERATECHILDREN(u)
  for $controls \in Samples(BRANCHOUT)$ **do**
    $state(u') = RobotModel(state(u), controls)$
    if $IsValid(state(u')$ **then**
      $u' \leftarrow CreateVertex(state(u'))$
      $h(u') \leftarrow HeuristicCost(u', v_{goal})$
      $Succ(u) \leftarrow Succ(u) \cup u'$
    end if
  end for
end function
function MAIN()
  $Initialize()$ ▷ Initializes the algorithm
  while $OPEN.TopKey() < Key(v_{goal})$ or $rhs(v_{goal}) \neq g(v_{goal})$ **do**
    $u \leftarrow OPEN.pop()$ ▷ Top element of the queue
    $GenerateChildren(u)$
    if $g(u) > rhs(u)$ **then**
      $g(u) \leftarrow rhs(u)$
      for $s \in Succ(u)$ **do** ▷ Iterate over all successors of u
        $UpdateState(s)$
      end for
    else
      $g(u) \leftarrow \infty$
      for $s \in Succ(u) \cup \{u\}$ **do** ▷ Iterate over all successors of u including u
        $UpdateState(s)$
      end for
    end if
  end while
  return Trajectory
end function

A. Brief Overview of Sampling-Based Model Predictive Optimization

SBMPO is a sampling-based algorithm that can be used for motion planning with kinematic and dynamic models. It can plan using a variety of cost functions, including the standard sum of the squared error cost function that is commonly used in Model Predictive Control (MPC).[25] At its inception, SBMPO was motivated by a desire to utilize sampling and A*-type optimization in lieu of the nonlinear programming strategies that are commonly employed for optimization in MPC. Use of these techniques provides SBMPO with the ability to avoid local minima, when present,[14] and achieve fast computations when implemented with properly designed A* heuristics.

Algorithm 1 contains pseudocode that describes SBMPO, which has many of the features of the lifelong-planning A* algorithm (LPA*).[26] For each vertex v, the algorithm maintains $h(v)$, $g(v)$, and $rhs(v)$, where $h(v)$ represents the cost-to-goal estimate (known in the artificial intelligence literature as the *heuristic*, $g(v)$ is the *start cost* (i.e., the cost of a lowest cost trajectory from v_{start} to v), and $rhs(v)$ is another estimate of the start cost. For any vertex v, $status_v$, the status of the vertex is given by

$$status_v = \begin{cases} \text{consistent}, & g(v) = rhs(v) \\ \text{inconsistent}, & \text{otherwise} \end{cases}, \tag{1}$$

where *consistent* vertices are vertices that have already been explored, while *inconsistent* vertices are vertices which need to be explored.

All inconsistent vertices are pushed into priority queue Q in the order of their priority, which is a two-component *key* vector {2}. The keys are ordered lexicographically with the smaller key values having a higher priority. An *implicit grid* is used by the algorithm to merge the vertices with similar states, thereby limiting the number of vertices that can exist within any finite region of the state space.

The main function Main() first calls Initialize() to initialize the trajectory-planning problem {2}. Initialize() sets the initial g-values of the vertices, v_{start} and v_{goal}, to infinity and sets their rhs-values to 0 and ∞ respectively. Thus, initially v_{start} is the only locally inconsistent vertex and is inserted into the otherwise empty priority queue (Q) with a key calculated according to {1}. The estimate of the cost-to-goal value of v_{goal} is set to 0, and the estimate of the cost-to-goal value of v_{start} is calculated with the help of $Heuristic\_Cost()$ {4}.

SBMPO expands vertices by picking up the top vertex in Q {4} until v_{goal} is locally consistent or the key of the vertex to expand next is no less than the key of v_{goal} {3}. The top vertex is expanded by sampling the input space {26}, typically using Halton sampling[27] or random sampling. The number of samples is called the *branchout factor*. The input sample (*controls*) and current state ($state(u)$) (i.e., state of the selected vertex u) are passed to the robot model, and the robot model is propagated to determine the new state ($state(u')$) of the system {28}.

The new state $state(u')$, if valid (i.e. it satisfies all constraints), is then added to the graph {4}. The graph is expanded with the help of $Create\_Vertex()$ {30}, which uses an implicit grid[28] to check if the graph already contains a vertex with state close to $state(u')$. If such a vertex exists, then return the vertex and add an edge from the current vertex (i.e., the selected vertex) to the vertex whose state is similar to the new state. Otherwise, return a new vertex u' whose state is the new $state(u')$.

The newly expanded vertex u' is added to the successor list of the selected vertex u {32}. The heuristic value $h(u')$ of the new state is calculated by $Heuristic\_Cost()$ {4}. SBMPO then updates the *rhs*-value and *key*-value of the new vertex as well as their membership in the priority queue if they become locally consistent or inconsistent with $UpdateStates()$ {14}, and finally calculates a trajectory by repeatedly expanding locally inconsistent vertices in the order of their priorities.

When deployed with an appropriate heuristic function, SBMPO can compute near-optimal solutions more rapidly than other graph-based planning algorithms that lack a heuristic for prediction. For example, Figure 1 shows the results produced by SBMPO and RRT* in a typical path planning scenario for a mobile robot with a simple kinematic model given by

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} v_t \cos\theta_{k+1} \\ v_t \sin\theta_{k+1} \\ v_\theta \end{bmatrix} \Delta T, \tag{2}$$

where x and y are the vehicle position components, θ is the vehicle heading, and the control inputs v_t and v_θ are the vehicle's forward velocity and rotational velocity, respectively. In this example, both approaches

Table 1. Performance Comparison of SBMPO and RRT* Path Planning Using a Simple Kinematic Model

	SBMPO	RRT*
Distance (m)	7.39	8.28
Comp. Time (ms)	1.9	50.0

attempt to compute the minimum distance path from (x_{start}, y_{start}) to (x_{goal}, y_{goal}) by exploring a graph that is propagated using the kinematic model in (2). Qualitatively, the resulting trajectories are similar, but, as summarized in the performance comparison in Table 1, SBMPO managed to generate a solution more than one order of magnitude faster than RRT* and, in fact, produced the shorter (i.e., lower cost) path. In complicated planning scenarios, this significant discrepancy in computation time prohibits the use of RRT* and similar approaches. Evident in the simple planning scenario shown in this comparison, the use of a heuristic for prediction facilitates the rapid computation in SBMPO.

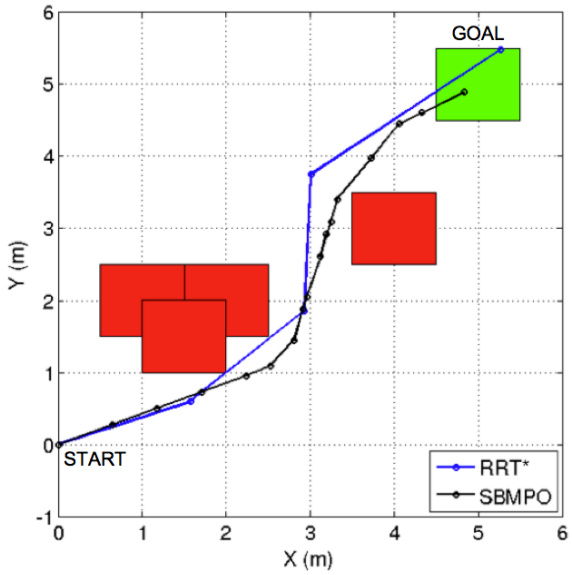


Figure 1. Minimum Distance Paths Computed by SBMPO and RRT*.

B. Cost Functions in SBMPO

In general, SBMPO provides the algorithmic formalism to solve the optimization problem,

$$\min_{u_k,\ldots,u_{k+N-1}} J\,, \tag{3}$$

where J is the cost function, N is the prediction horizon, and optimization is subject to constraints on the control input u_j, and the output from the propagation model y_j. Note that when $N = \infty$, the algorithm runs until the goal configuration is reached. For the examples in this work, the minimum distance and minimum time optimzation problems are solved using the cost functions summarized in Table 2

Table 2. Typical Cost Function Expressions Used in SBMPO

	Cost Function (J)	Variable Definitions
Minimum Distance	$\sum_{i=0}^{N-1} d_{k+i}$	d_j: distance from node j to $j+1$
Minimum Time	$\sum_{i=0}^{N-1} t_{k+i}$	t_j: time from node j to $j+1$

C. Six-DOF Relative Motion Spacecraft Model for Rendezvous

The spacecraft rendezvous and docking work presented in this paper requires the incorporation of the vehicle dynamic model for full six degree of freedom (DOF) trajectory planning. Shown in control affine form, the nonlinear dynamic equation describing the relative motion of the spacecraft with respect to the target is

$$\begin{bmatrix} \dot{v} \\ \dot{r} \\ \dot{\omega} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \tilde{0} \\ \dot{v} \\ -J^{-1}\omega \times J\omega \\ \frac{1}{2}\Omega(\omega)q \end{bmatrix} + \begin{bmatrix} \frac{1}{m}\Theta^T(q)u(t) \\ \tilde{0} \\ J^{-1}\tau(t) \\ \tilde{0} \end{bmatrix}, \tag{4}$$

where r and v represent the position and velocity of the spacecraft with respect to the target, ω is the angular velocity in the spacecraft body frame, q is the rotation quaternion vector, $\Theta^T(q)$ is the transposed rotation

matrix expressed as a function of q, $\Omega(\omega)$ is the kinematic quaternion matrix expressed as a function of ω, J is the inertia matrix, m is the vehicle mass, $u(t)$ is thrust control input vector, $\tau(t)$ is the torque control input vector, and $\tilde{0}$ is the zero vector. As in the research by McCamish,[29] a spacecraft with the characteristics defined in Table 3 was used in simulation.

Essentially, the rendezvous and docking task may be simplified as a trajectory generation problem for an autonomous chase spacecraft navigating relative to a target orbiting body. A successful solution to the problem requires the determination of the trajectory and control inputs necessary to achieve motion from an initial position and orientation to some goal position and orientation. During the planning from initial to goal configuration, the algorithm considers obstacles and other spatial restrictions, known motion and/or acceleration of the target, and, in many cases, nonholonomic vehicle constraints. Also, unlike path generation, the trajectory generation problem involves planning with conditions placed on the goal ending velocity.

Table 3. Spacecraft Characteristics

Parameter	Value
Length (L)	1.0 m
Width (W)	1.0 m
Height (H)	1.0 m
Mass	100.0 kg
Moment of Inertia X	16.67 kg-m^2
Moment of Inertia Y	16.67 kg-m^2
Moment of Inertia Z	16.67 kg-m^2
Number of Thrusters	6
Maximum Thrust Per Axis ($u_{i,max}$)	1.0 N
Number of Reaction Wheels	3
Rotation Wheel Maximum Torque ($\tau_{i,max}$)	0.055 Nm

III. Combined Relative Position and Attitude Trajectory Planning Using a Dynamic Model

Unlike typical mobile robotics applications of A*-type path planning, spacecraft rendezvous trajectory planning requires additional consideration of motion constraints that are fundamentally dictated by the nature of spacecraft docking. Specifically, the docking task requires that the spacecraft approach the goal in a manner that seeks to match the motion of the target and prevents unwanted collision.

SBMPO enables the development of trajectories based on optimizing various physical metrics (e.g., distance, time, or energy). This versatility depends on the development of an appropriate heuristic used in the A*-type planner.

In order to facilitate rapid computation of trajectories, SBMPO relies on a heuristic that predicts the cost-to-goal as nodes are explored by the planner. Enabling the algorithm to efficiently converge to an optimal solution, the A*-type planner requires an optimistic heuristic that is fairly non-conservative. When implemented with a naïve or overly-conservative heuristic, A*-type algorithms are computationally inefficient. Therefore, it is crucial that the heuristic be carefully developed within the context of the planning scenario.

As defined in the previous section, the spacecraft model describes a fully-actuated vehicle with independent actuators controlling the six degrees-of-freedom along the body axes. Under this presumption, the heuristic functions developed in the following sections may be applied to each of the six independent control inputs (i.e., $u_x, u_y, u_z, \tau_x, \tau_y, \tau_z$).

A. Development of an Appropriate Minimum Time Heuristic

To enable efficient minimum time planning, an appropriate heuristic was formed using the solution to the fundamental time optimal control problem. The minimum time control problem can be solved by forming the Hamiltonian and applying Pontryagin's Maximum Principle.[30] Assuming that the controlled acceleration is bound by

$$\ddot{r} = a, \quad -\underline{a} \leq a \leq \bar{a}, \tag{5}$$

where $\underline{a}$ and $\bar{a}$ are, respectively, the lower and upper limits for acceleration. The solution of the minimum time control problem of Bryson[30] can be generalized to yield[22]

$$\begin{aligned} t^2 - \frac{2v_i}{\underline{a}}t &= \frac{v_i^2 + 2(\underline{a}+\bar{a})r}{\underline{a}\bar{a}}, \text{ if } r + \frac{v_i|v_i|}{2\bar{a}} < 0 \\ t^2 + \frac{2v_i}{\bar{a}}t &= \frac{v_i^2 - 2(\underline{a}+\bar{a})r}{\underline{a}\bar{a}}, \text{ if } r + \frac{v_i|v_i|}{2\underline{a}} > 0. \end{aligned} \tag{6}$$

The minimum time computed using (6) corresponds to the bang-bang optimal controller shown in Figure 2, which illustrates switching curves that take the system to the origin by applying either the minimum or maximum control inputs (i.e., $a = -\underline{a}$ or $a = \bar{a}$). Depending on the initial conditions, the system uses either the minimum or maximum control input to take the system to the appropriate switching curve. For example, if (r_i, v_i) corresponds to point p_1 in Figure 2, then the control input $a = -\underline{a}$ is applied until the system reaches point p_2 on the switching curve. At p_2, the control input is then switched to $a = \bar{a}$, which drives the system to the goal where $r = v = 0$.

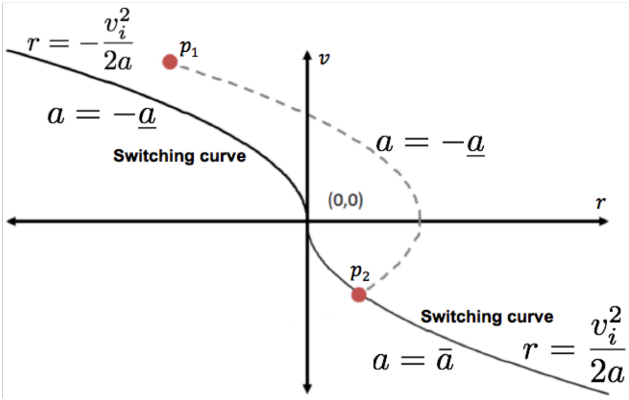


Figure 2. Minimum time control curve.

To compute the minimum time heuristic, it is necessary to determine the upper and lower bounds on the translational and angular acceleration. The dynamic model (4) expresses the translational acceleration as a function of the quaternion vector and the thrust as

$$a = \dot{v} = \frac{1}{m}\Theta^T(q)u(t), \tag{7}$$

where the quaternion vector, q, is the unit vector given by

$$q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T, \tag{8}$$

and the rotation matrix $\Theta(q)$ is given by

$$\Theta(q) = \begin{bmatrix} (1-2q_2^2-2q_3^2) & 2(q_1q_3+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & (1-2q_1^2-2q_3^2) & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & (1-2q_1^2-2q_2^2) \end{bmatrix}. \tag{9}$$

If the pitch, yaw, and roll are all zero angles, then

$$q = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T, \tag{10}$$

causing

$$\Theta(q) = I \tag{11}$$

and, hence, the bounds on translational acceleration become

$$-\frac{1}{m}u_{i,max} \leq a_i \leq \frac{1}{m}u_{i,max},\ i = 1,2,3. \tag{12}$$

The process for finding the bounds on angular acceleration follows a similar process beginning with the relationship from (4),

$$\dot{\omega} = -J^{-1}\omega \times J\omega + J^{-1}\tau(t), \tag{13}$$

where the elements of ω are the first derivatives of roll, pitch, and yaw, i.e.,

$$\omega = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T. \tag{14}$$

If the vehicle model is symmetrical, such that the inertia matrix J is given by

$$J = \sigma I, \quad \sigma > 0, \tag{15}$$

then it can be shown that

$$-J^{-1}\omega \times J\omega = 0, \tag{16}$$

causing (13) to become

$$\dot{\omega} = J^{-1}\tau(t). \tag{17}$$

Hence, the bounds on angular acceleration are

$$-\frac{1}{\sigma}\tau_{i,max} \leq \dot{\omega}_i \leq \frac{1}{\sigma}\tau_{i,max},\ i = 1, 2, 3. \tag{18}$$

In the spacecraft rendezvous problem, the minimum time heuristic $\underline{t}_f$ for a node corresponding to $\langle r(t + \Delta_t),\ v(t + \Delta_t),\ \Theta(q(t + \Delta_t)),\ \omega(t + \Delta_t)\rangle$ is determined by first solving the minimum time problems for $i = 1, 2, 3$ given by

$$\ddot{r}_i = \frac{1}{m}u_i, \quad -u_{i,max} \leq u_i \leq u_{i,max} \tag{19}$$

with the initial conditions

$$r_i(0) = r_i(t), \qquad \dot{r}_i(0) = v_i(t) \tag{20}$$

and the goal conditions

$$r_i(t_{f,i}) = r_{i,goal}, \qquad \dot{r}_i(t_{f,i}) = 0. \tag{21}$$

Then, pitch, yaw, and roll are evaluated in the same manner by solving the minimum time problems

$$\ddot{\phi} = -\frac{1}{\sigma}\tau_1\,, \quad -\tau_{1,max} \leq \tau_1 \leq \tau_{1,max} \tag{22}$$

$$\ddot{\theta} = -\frac{1}{\sigma}\tau_2\,, \quad -\tau_{2,max} \leq \tau_2 \leq \tau_{2,max} \tag{23}$$

$$\ddot{\psi} = -\frac{1}{\sigma}\tau_3\,, \quad -\tau_{3,max} \leq \tau_3 \leq \tau_{3,max} \tag{24}$$

with the initial conditions

$$\phi(0) = \phi(t), \qquad \dot{\phi}(0) = \omega_1(t) \tag{25}$$

$$\theta(0) = \theta(t), \qquad \dot{\theta}(0) = \omega_2(t) \tag{26}$$

$$\psi(0) = \psi(t), \qquad \dot{\psi}(0) = \omega_3(t) \tag{27}$$

and the goal conditions

$$\phi(t_{f,\phi}) = \phi_{goal}, \qquad \dot{\phi}(t_{f,\phi}) = 0 \tag{28}$$

$$\theta(t_{f,\theta}) = \theta_{goal}, \qquad \dot{\theta}(t_{f,\theta}) = 0 \tag{29}$$

$$\psi(t_{f,\psi}) = \psi_{goal}, \qquad \dot{\psi}(t_{f,\psi}) = 0. \tag{30}$$

Finally, the minimum time heuristic $\underline{t}_f$ for the node is computed as

$$\underline{t}_f = \max(\underline{t}_{f,1}, \underline{t}_{f,2}, \underline{t}_{f,3}, \underline{t}_{f,\phi}, \underline{t}_{f,\theta}, \underline{t}_{f,\psi}). \tag{31}$$

B. Development of an Appropriate Minimum Distance Heuristic

For minimum distance *path-only* planning, it is sufficient to use a heuristic that represents the Euclidean distance between the expanded vertex and the goal. However, when applied in a planner that propagates the vehicle's trajectory using the dynamic model, this heuristic leads to a minimum distance trajectory in which the vehicle accelerates, rather than brakes, as it approaches the goal.

In scenarios such as autonomous rendezvous and docking, where path-only planning is inappropriate, it is necessary to develop a heuristic that considers the full state of the vehicle throughout the graph expansion process as the planner develops a trajectory that seeks to match the goal state. For minimum distance trajectory planning, the heuristic that was developed is based on the fact that the distance to goal r, the initial velocity v, the final velocity v_f, and the acceleration a are related by

$$v_f^2 = v^2 + 2ar. \tag{32}$$

In order to stop at the goal, $v_f = 0$, and the relationship becomes

$$\Delta_{stop} = -\frac{v^2}{2\underline{a}}, \tag{33}$$

where Δ_{stop} is the minimum stopping distance, v is the velocity, and the sampled control thrust u is bounded by

$$-m\underline{a} \leq u \leq m\bar{a}. \tag{34}$$

The complete minimum distance rendezvous heuristic[23] has three input parameters: the relative position of the vehicle r, the relative velocity of the vehicle v, and the desired relative position of the vehicle r_G. Shown in Algorithm 2, the return value of the minimum distance rendezvous heuristic, r_{stop}, is the minimum stopping distance required to reach the goal position at a desired velocity subject to the acceleration bounds defined within the sampler.

In the event that the vehicle is approaching the goal too fast and will not be able to stop, lines 2-13 of the algorithm return a minimum stopping distance that requires the vehicle to go past the goal and return. Lines 14-26 deal with the mirror case in which the vehicle has surpassed the goal.

Algorithm 2 Velocity-Aware Minimum Distance Heuristic

```
1:  function MinDistHeuristic(r_goal, r, v)
2:      if r < r_goal then
3:          if v < 0 then                                   ▷ headed away from goal
4:              Δ_stop = |−v²/ā|                            ▷ minimum distance required to stop
5:              Δ_r = 2Δ_stop + (r_goal − r)
6:          else if v ≥ 0 then                              ▷ headed toward goal or at rest
7:              Δ_stop = −−v²/a̲                             ▷ minimum distance required to stop
8:              if Δ_stop > (r_goal − r) then
9:                  Δ_r = 2Δ_stop − (r_goal − r)
10:             else
11:                 Δ_r = r_goal − r
12:             end if
13:         end if
14:     else if r > r_goal then
15:         if v ≤ 0 then                                   ▷ headed toward goal or at rest
16:             Δ_stop = |−v²/ā|                            ▷ minimum distance required to stop
17:             if Δ_stop > (r − r_goal) then
18:                 Δ_r = 2Δ_stop − (r − r_goal)
19:             else
20:                 Δ_r = r − r_goal
21:             end if
22:         else if v > 0 then                              ▷ headed away from goal
23:             Δ_stop = −−v²/a̲                             ▷ minimum distance required to stop
24:             Δ_r = 2Δ_stop + (r − r_goal)
25:         end if
26:     end if
27:     return Δ_r
28: end function
```

In the spacecraft rendezvous problem, the minimum distance heuristic Δ_f is determined by first using Algorithm 2 to solve the minimum distance problems for $i = 1, 2, 3$ given by

$$\Delta_{stop,i} = -\frac{\dot{r}_i^2}{2a_i}, \quad -\underline{a}_i \leq a_i \leq \bar{a}_i \tag{35}$$

with the initial conditions

$$r_i(0) = r_i(t), \qquad \dot{r}_i(0) = v_i(t), \tag{36}$$

the goal conditions

$$r_i(t_f) = r_{i,goal}, \qquad \dot{r}_i(t_f) = 0, \tag{37}$$

and the translational acceleration bounds derived in (12). The pitch, yaw, and roll are then evaluated in a similar manner; Algorithm 2 is again used to solve the minimum distance problems given by

$$\Delta_{stop,\phi} = -\frac{L\dot{\phi}^2}{2\dot{\omega}_1}, \qquad -\frac{1}{\sigma}\tau_{1,max} \leq \dot{\omega}_1 \leq \frac{1}{\sigma}\tau_{1,max} \tag{38}$$

$$\Delta_{stop,\theta} = -\frac{W\dot{\theta}^2}{2\dot{\omega}_2}, \qquad -\frac{1}{\sigma}\tau_{2,max} \leq \dot{\omega}_2 \leq \frac{1}{\sigma}\tau_{2,max} \tag{39}$$

$$\Delta_{stop,\psi} = -\frac{H\dot{\psi}^2}{2\dot{\omega}_3}, \qquad -\frac{1}{\sigma}\tau_{3,max} \leq \dot{\omega}_3 \leq \frac{1}{\sigma}\tau_{3,max} \tag{40}$$

with the initial conditions

$$\phi(0) = \phi(t), \qquad \dot{\phi}(0) = \omega_1(t) \tag{41}$$

$$\theta(0) = \theta(t), \qquad \dot{\theta}(0) = \omega_2(t) \tag{42}$$

$$\psi(0) = \psi(t), \qquad \dot{\psi}(0) = \omega_3(t), \tag{43}$$

the goal conditions

$$\phi(t_f) = \phi_{goal}, \qquad \dot{\phi}(t_f) = 0 \tag{44}$$

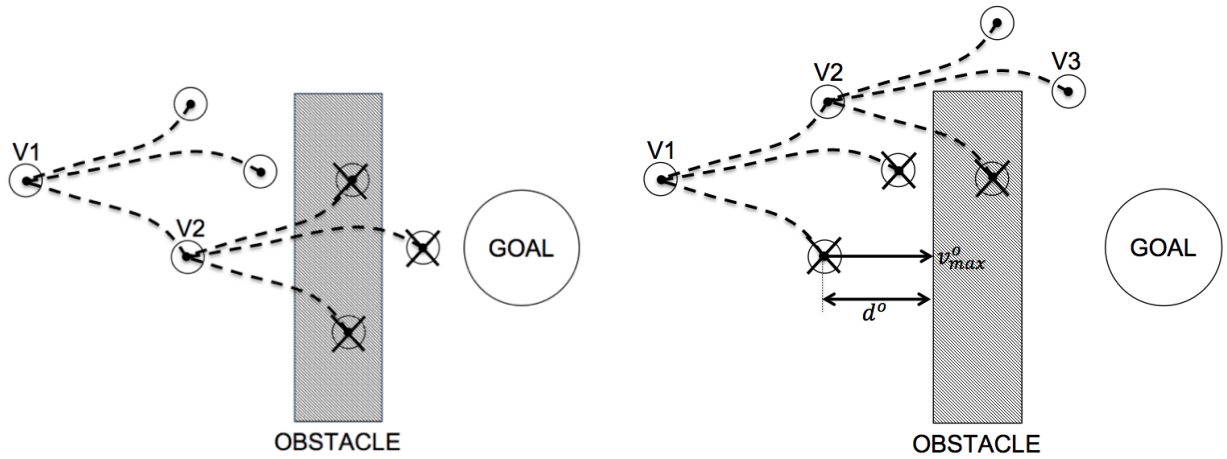
$$\theta(t_f) = \theta_{goal}, \qquad \dot{\theta}(t_f) = 0 \tag{45}$$

$$\psi(t_f) = \psi_{goal}, \qquad \dot{\psi}(t_f) = 0, \tag{46}$$

and – recalling from Table 3 – the vehicle length L, width W, and height H. Finally, the minimum distance heuristic Δ_f is computed as

$$\Delta_f = \max(\Delta_{f,1}, \Delta_{f,2}, \Delta_{f,3}, \Delta_{f,\phi}, \Delta_{f,\theta}, \Delta_{f,\psi}). \tag{47}$$

C. Development of Momentum-Aware, Imminent Collision Detection



(a) Nodes rejected as collision is detected.

(b) Nodes preemptively rejected.

Figure 3. Illustrations of (a) rudimentary collision detection and (b) imminent collision detection.

When planning in cluttered environments, the algorithm requires a collision detection method to reject vertices that spatially violate obstacles. Typically, obstacle avoidance is achieved by simply eliminating vertices that result in direct collision. This method of basic collision detection is illustrated in Figure 3(a).

For trajectory planning, rudimentary collision detection does not consider additional constraints placed on spatially viable vertices in the vicinity of obstacles that are none-the-less invalid for further expansion. For example, suppose that a vertex is generated that, based on its position, does not collide with an obstacle but due to the velocity required to achieve the state at that vertex will, when expanded, inevitably only generate child vertices that are in collision with an obstacle. Eventually, the vertex described would be ignored, but at the cost of computational performance. Ideally, such a vertex, like the one labeled V2 in Figure 3(a), would be preemptively rejected for expansion.

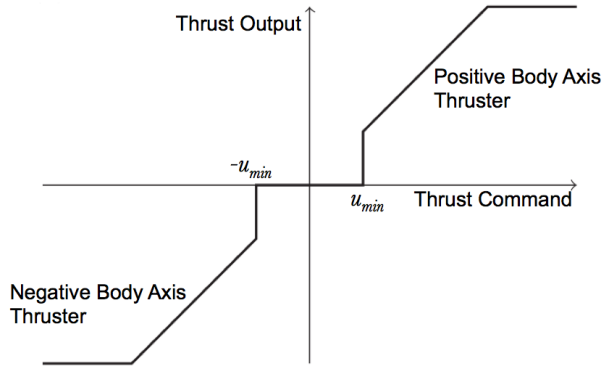
To improve performance, a momentum-based imminent collision detection method was developed for this problem. Utilizing the same relationship necessary for developing the minimum distance heuristic (32), imminent collision detection is based on the vehicle's ability to decelerate prior to collision. When used for collision detection, the maximum viable velocity in the direction of the nearest obstacle, v^o_{max}, and the minimum distance to the nearest obstacle, d^o, are calculated using the relationship

$$v^o_{max} = \sqrt{2\underline{a}d^o}. \tag{48}$$

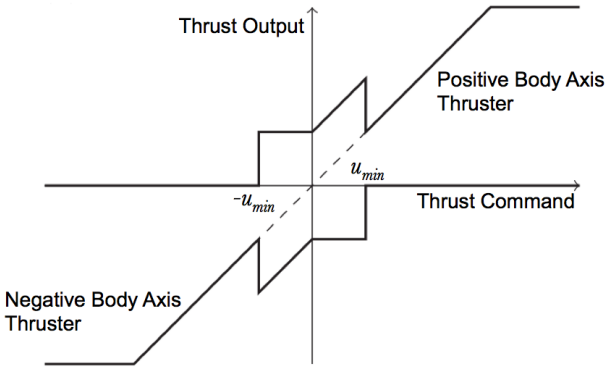
Using imminent collision detection, the vertices are expanded and rejected as shown in Figure 3(b).

D. Minimum Effective Thrust Command Threshold

As discussed, SBMPO is capable of computing optimal trajectories that adhere to the dynamic constraints due to the actuation limits. Previously, this capability was discussed in the context of the usage of a dynamic vehicle model and by enforcing an upper bound on sampled control inputs, but it is also relevant when incorporating additional actuation constraints.



(a) Thruster deadzone caused by minimum effective thrust constraint.



(b) Restoration of fine thrust control via a paired thruster strategy.

Figure 4. Constrained thruster curves corresponding to (a) unpaired and (b) paired thruster usage.

In the spacecraft rendezvous planning scenario, thruster actuation may be constrained by more than just the maximum directional output (i.e., $\underline{a}$ and $\bar{a}$). Specifically, individual thrusters may have a minimum command threshold that effectively limits the lower bound of the thruster output.[24] When paired with a thruster in the opposite direction, this constraint produces an actuation curve that resembles the nonlinear control deadzone shown in Figure 4(a). As a result, it becomes impossible to deliver small thrust outputs utilizing a single thruster. However, since they are configured as opposing-direction pairs, the thrusters may maintain control linearity in the deadzone region by working in tandem and thus producing a net small thrust output that can be seen in Figure 4(b). Clearly, this compromise comes at a significant cost in terms of fuel consumption but is necessary to preserve fine control. If defining the cost associated with single thruster usage at a vertex k as

$$\text{cost}_u(k) = u(k), \tag{49}$$

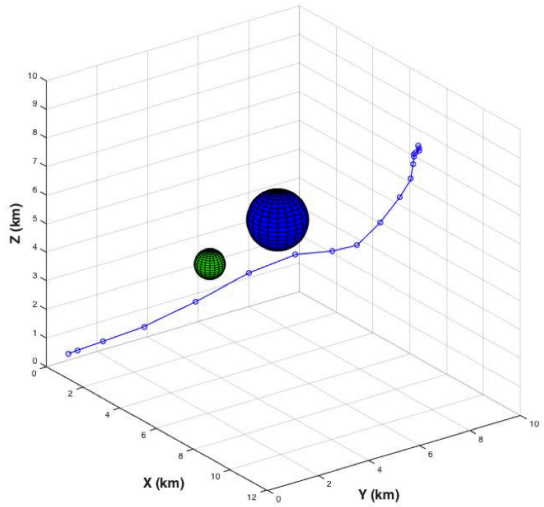
then the cost associated with paired usage of thrusters subject to a minimum effective command constraint u_{min} becomes

$$\text{cost}_u(k) = \begin{cases} 2u_{min} + u(k) & \text{if } |u(k)| < u_{min} \text{ and } u(k) \neq 0, \\ u(k) & \text{if } |u(k)| \geq u_{min}, \\ 0 & \text{if } |u(k)| = 0, \end{cases}$$

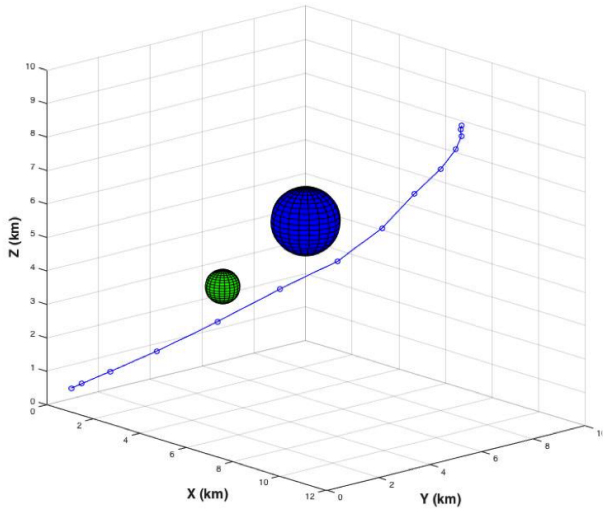
where $u(k)$ is the commanded thrust at vertex k and u_{min} is the threshold for minimum effective commanded thrust. The cost associated with thruster usage, $\text{cost}_u(k)$, is then incorporated as a weighted combinatorial term in the base cost function, J. For example, to include the thruster cost in the minimum distance problem, the cost function is modified to become

$$J = \sum_{i=0}^{N-1} d_{k+i} + \eta \text{cost}_{u,k+i} \tag{50}$$

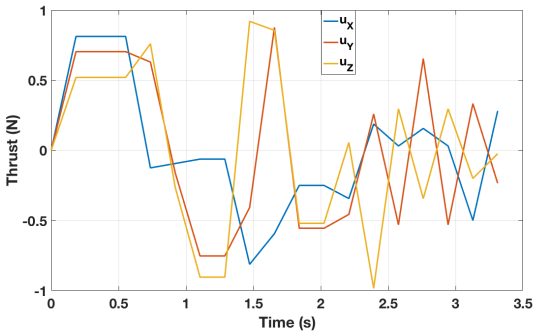
where η is a tunable weighting factor on the interval $[0, 1]$.



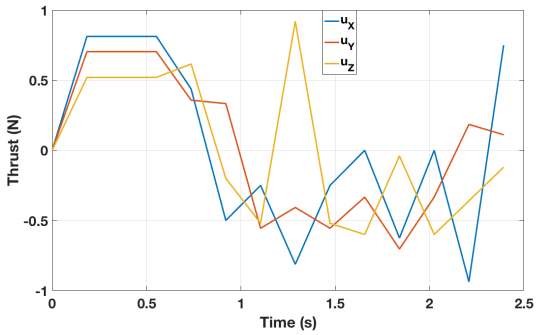
(a) Unoptimized vehicle trajectory subject to minimum effective thrust constraint.



(b) Optimized vehicle trajectory using paired thruster strategy and optimization.



(c) Unoptimized thruster command history.



(d) Optimized command history using paired thruster approach.

Figure 5. Comparison of the minimum distance rendezvous trajectories computed by SBMPO with and without the paired thruster strategy.

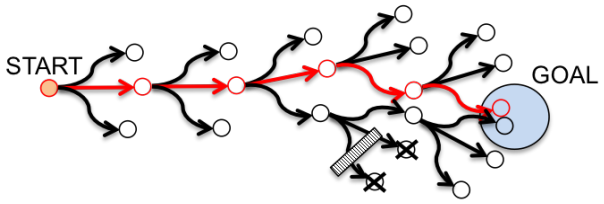
When the minimum effective thrust constraint is applied, a comparison of the unoptimized approach, shown in Figures 5(a) and 5(c), with the paired thruster strategy ($\eta = 0.05$), shown in Figures 5(b) and 5(d), reveals a smoother, qualitatively-similar trajectory that utilizes fewer low thrust commands. In both cases, the minimum distance problem is addressed, but when subject to the minimum effective thrust constraint, $u_{min} = 0.3$ N, the paired thruster cost function described above delivers a 52.6 percent savings in terms of net thruster output and is computed in 27.7 milliseconds on a laptop with a 2.4 GHz Intel Core-2 Duo processor and 16 GB of DDR3 RAM.

E. Efficient Replanning via Lifelong-Planning A* and Receding Horizon

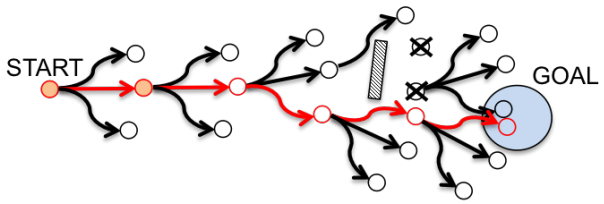
To accommodate non-deterministic constraints in the trajectory generation task, it is necessary to implement an efficient replanning strategy that is capable of delivering optimal solutions *in situ* when encountering changes in the planning environment. Specifically, when the initial solution trajectory becomes invalid due to the movement of obstacles in the planning space, the rapid computation of a new trajectory warrants the use of past planning information, when feasible. Here, this is partially accomplished by utilizing Lifelong-Planning A* (LPA*) for the graph search process in place of the traditional A* algorithm. LPA* provides a framework to use previously computed graph information whenever possible and, if necessary, continue the sampling-based propagation of the graph to produce a new optimal trajectory. A simplified representation of this process is diagrammed in Figure 6.

As anyone forced to encounter a detour when traveling in an unfamiliar region would attest, the use of past information is an intuitively obvious means to speed up replanning. However, in the case of a complex autonomous planning scenario, frequent replanning can become computationally expensive very quickly. In terms of the graph search process, both the number of vertices maintained in the planning space and the size of the priority queue are prone to exponential growth when using past information in replanning. In order to alleviate this problem, the concept of receding horizon planning was implemented alongside LPA* to effectively constrain the planning space by seeking only to compute a trajectory to a subgoal in progress to the final goal state. In other words, the receding horizon replanning approach delivers the optimal trajectory only over a defined number of time steps (i.e., travel time horizon) in pursuit of the goal.

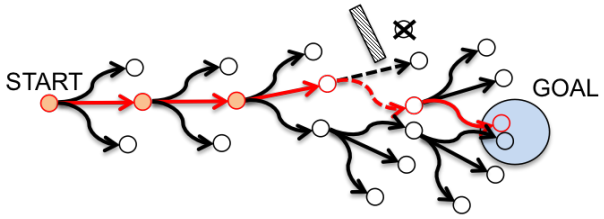
When the horizon is very large, the solution trajectory will reach the goal if possible, but this will occur at the cost of additional memory usage to accommodate the larger graph and priority queue. However, when constrained to an appropriate nearby horizon, the replanning process will deliver an approximately optimal solution trajectory after a number of iterations and, in the process, maintain minimal graph expansion. As identified in Algorithms 3, 4, and 5, the SBMPO psuedocode from Algorithm 1 was modified to accommodate receding horizon planning. Specifically, when generating the graph space, the modified algorithm assigns an additional index to each vertex that identifies



(a) Step A: Initially, an optimal solution trajectory is generated via graph expansion and node connection/removal.



(b) Step B: Progressing toward the goal, movement of the obstacle is observed to violate the initial solution. A new trajectory is calculated using past graph information and, when necessary, further expansion.



(c) Step C: Obstacle movement is again detected and, utilizing past graph information, the optimal trajectory is restored.

Figure 6. Replanning steps.

Algorithm 3 Modified Initialize Function

function INITIALIZE()
  $OPEN \leftarrow \emptyset$
  $g(v_{start}), g(v_{goal}), rhs(v_{goal}) \leftarrow \infty$
  $rhs(v_{start}), h(v_{goal}), horiz(v_{start}) \leftarrow 0$
  $h(v_{start}) \leftarrow HeuristicCost(v_{start}, v_{goal})$
  $horizon \leftarrow horizon_{lim}$ ▷ Define number of steps to horizon window using $horizon_{lim}$
  $OPEN \leftarrow OPEN \cup v_{start}$
end function

Algorithm 4 Mod. Generate Children Function

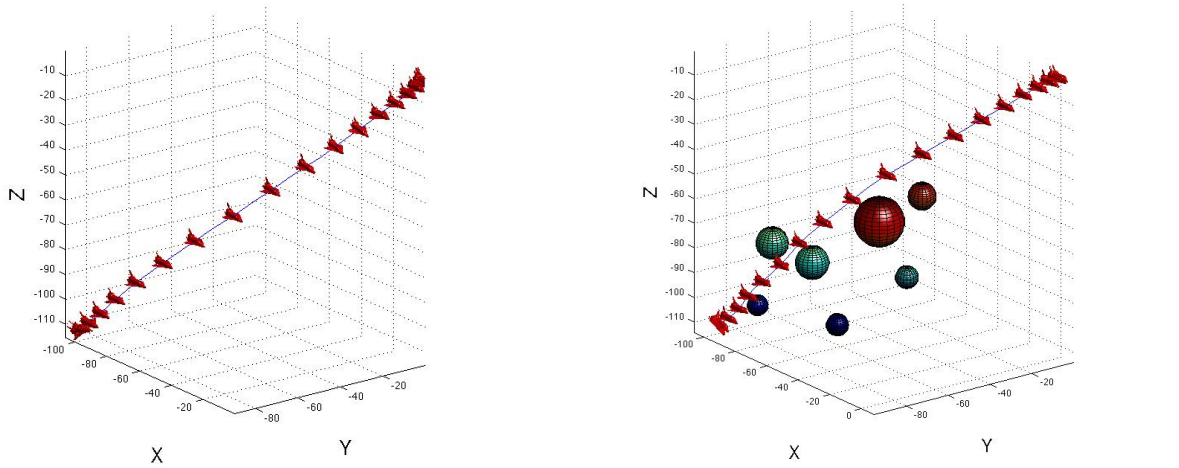
function GENERATECHILDREN(u)
  for $controls \in Samples(BRANCHOUT)$ **do**
    $state(u') = RobotModel(state(u), controls)$
    if $IsValid(state(u')$ **then**
      $u' \leftarrow CreateVertex(state(u'))$
      $h(u') \leftarrow HeuristicCost(u', v_{goal})$
      $Succ(u) \leftarrow Succ(u) \cup u'$
      $horiz(u') \leftarrow horiz(u) + 1$
    end if
  end for
end function

Algorithm 5 Modified SBMPO MAIN Function

```
1: function MAIN()
2:     Initialize()                                          ▷ Initializes the algorithm
3:     while OPEN.TopKey() < Key(v_goal) or rhs(v_goal) ≠ g(v_goal) do
4:         u ← OPEN.pop()                                    ▷ Top element of the queue
5:         if horiz(u) = horizon then                        ▷ Horizon window reached before goal
6:             return Trajectory
7:         end if
8:         GenerateChildren(u)
9:         if g(u) > rhs(u) then
10:            g(u) ← rhs(u)
11:            for s ∈ Succ(u) do                            ▷ Iterate over all successors of u
12:                UpdateState(s)
13:            end for
14:        else
15:            g(u) ← ∞
16:            for s ∈ Succ(u) ∪ {u} do                      ▷ Iterate over all successors of u including u
17:                UpdateState(s)
18:            end for
19:        end if
20:    end while
21:    return Trajectory
22: end function
```

the number of time steps required to reach the associated state (i.e., horizon index). Via the graph expansion process, the trajectory is computed until either the user-defined horizon limit or the goal has been reached. In the event that the horizon limit is reached prior to the goal, the algorithm repeats the planning process using past graph information from the previous iteration until either termination criteria is met.

IV. Simulation Results for Combined Relative Position and Attitude Planning



(a) Minimum distance rendezvous trajectory produced by SBMPO in an uncluttered environment.

(b) Minimum time rendezvous trajectory produced by SBMPO in a cluttered environment.

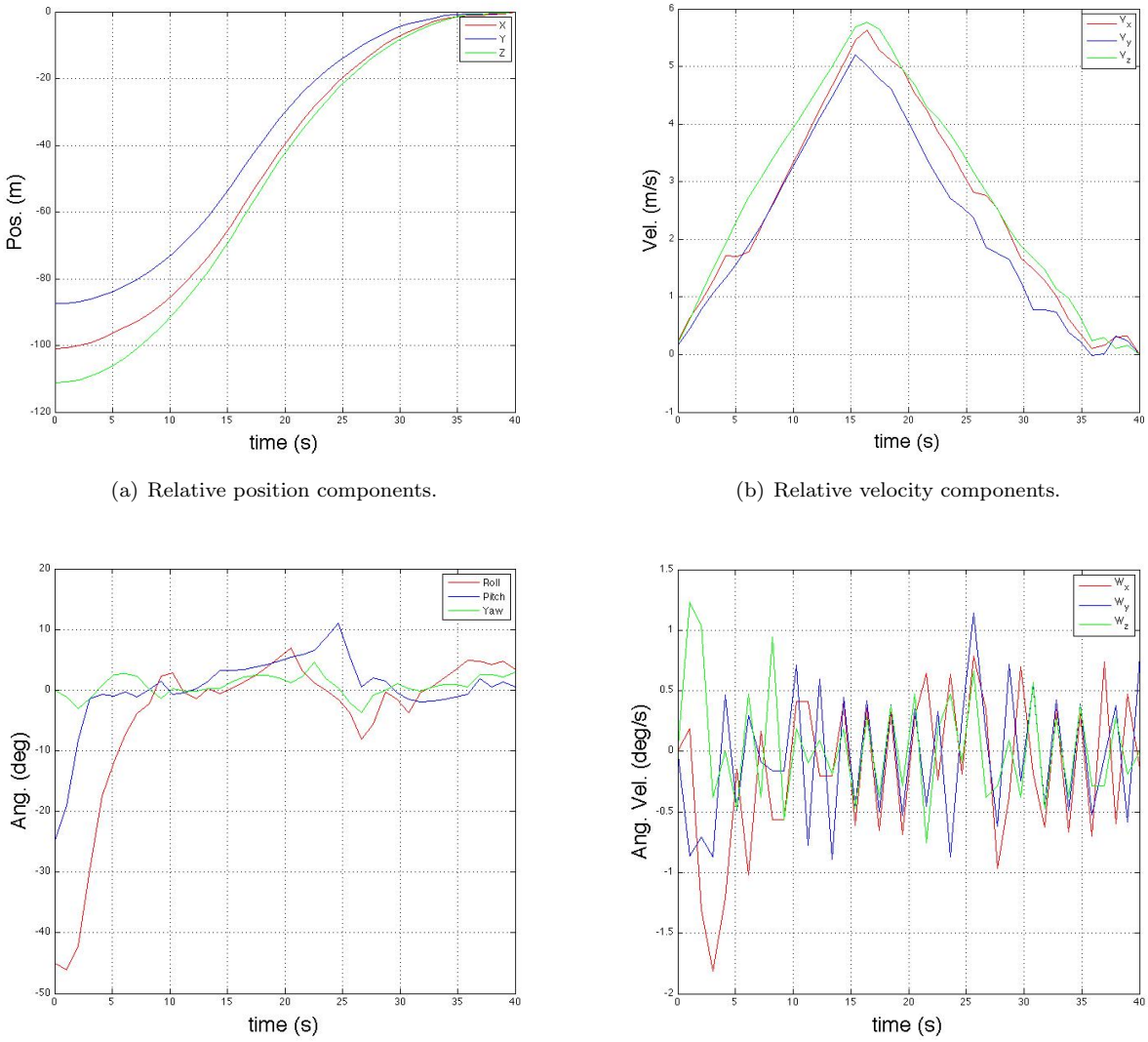
Figure 7. Rendezvous trajectories computed by SBMPO in uncluttered and cluttered environments.

The results presented in this section demonstrate the capability of the planner to rapidly generate six-DOF trajectories that are appropriate for rendezvous in static and dynamic cluttered environments. As in Section D these simulations were completed on a laptop with a 2.4 GHz Intel Core-2 Duo processor and 16

GB of DDR3 RAM.

In each of the simulations, the spacecraft is initially disoriented with respect to the target with Euler angles of $(-45.0°, -25.0°, 0.0°)$ and is positioned at $(-101.0\text{m}, -87.5\text{m}, -111.2\text{m})$. In this configuration, the spacecraft is trailing the target, which, as defined by the system dynamics, is located at the frame origin. Intending to rendezvous with the target, the goal position and orientation is coincident with the frame origin.

A. Simulation Results in Uncluttered Environment Using Minimum Distance Heuristic



(a) Relative position components.

(b) Relative velocity components.

(c) Euler angle components.

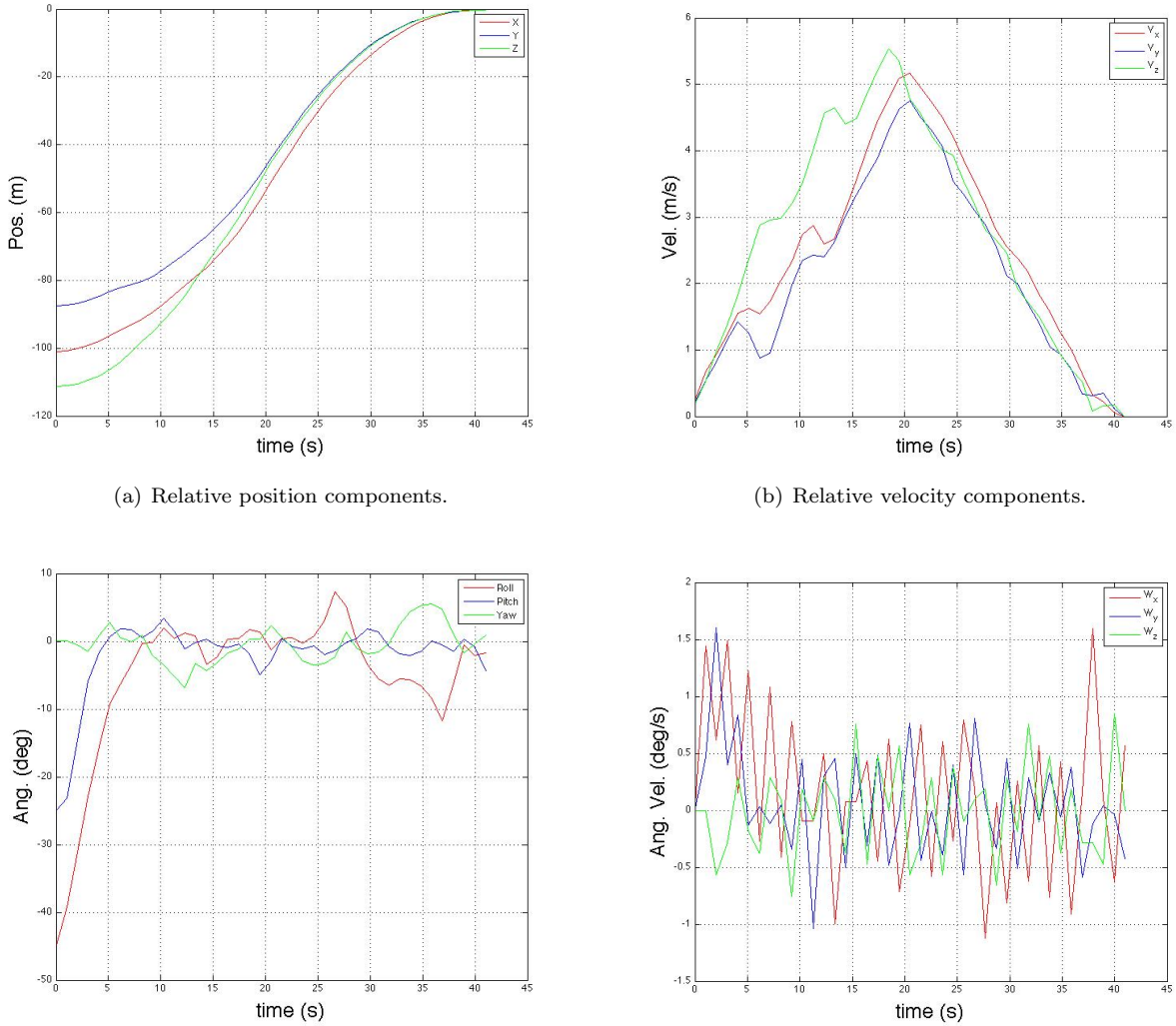
(d) Angular velocity components.

Figure 8. Parameter profiles of the SBMPO solution for a minimum distance rendezvous trajectory in an uncluttered environment.

Most evident when deployed in obstacle-free environments, the planner generates minimum distance trajectories that result in a desired relative position, attitude, and velocity with respect to the target. The results in this section primarily serve to illustrate the effectiveness of the developed heuristic when used in conjunction with the SBMPO algorithm to generate rendezvous feasible trajectories. Shown in Figures 7(a) and 8, simulation results for uncluttered planning correspond to an expected control sequence, where the vehicle decelerates as it closes in on the goal. When planning in uncluttered space, the minimum distance solution approximately matches the minimum time solution driven by the bang-bang control sequence and

is optimal subject to randomized sampling. The rendezvous trajectory, corresponding to a path length of 174.0 meters, was generated in 0.48 seconds.

B. Simulation Results in Cluttered Environment Using Minimum Time Heuristic



(a) Relative position components.

(b) Relative velocity components.

(c) Euler angle components.

(d) Angular velocity components.

Figure 9. Parameter profiles of the SBMPO solution for a minimum time rendezvous trajectory in a cluttered environment.

SBMPO is also capable of generating time-optimal, collision-free trajectories when in cluttered environments, which is desirable for the potential application of the algorithm to orbital debris navigation and removal. Shown in Figures 7(b) and 9, simulation results of minimum time planning in cluttered environments demonstrate the capability of SBMPO to generate feasible rendezvous trajectories in the presence of obstacles. In this simulation, seven randomly-sized and randomly-positioned spherical objects were included as obstacles impeding the path toward the target. As predicted, the resultant minimum time trajectory corresponds to the optimal bang-bang control sequence with some expected divergence due to the presence of impeding obstacles. The collision-free rendezvous trajectory, corresponding to a path length of 179.2 meters, was generated in 2.23 seconds.

C. Simulation Results in Non-deterministic Environment Using Receding Horizon Replanning

In terms of practical application, it is improbable that an autonomous vehicle will maintain absolutely perfect *a priori* information regarding its operating environment. To deal with non-deterministic elements, replanning is a necessity. Here, an uncharacterized moving obstacle is observed as the vehicle approaches the goal and spatially violates the initial solution trajectory. As the vehicle progresses, the obstacle continues to move and, in turn, replanning takes place to efficiently recompute an optimal, dynamically and spatially-viable solution trajectory.

In Figure 10, minimum time solution trajectories are shown in red for the first six time-steps of the simulation scenario. The observed position of the moving obstacle at each time-step is superimposed over its transparent motion profile and the previous solution trajectory is shown in blue. At each step, the moving obstacle is observed to have moved from its previous location relative to the target, and, for $3 \leq k \leq 5$, it is apparent that the obstacle has moved to a position that violates the previously computed solution trajectory. The planner must compute a new solution trajectory to avoid potential collision at these time-steps.

As derived in Section E, efficient replanning is achieved by utilizing prior graph information whenever viable. Here, the total computation time to produce collision-free rendezvous trajectories in the presence of the non-deterministic obstacle is 1.06 seconds when replanning is used in SBMPO. For comparison, it takes a total of 5.60 seconds to compute solution trajectories when replanning is not incorporated and SBMPO plans from scratch at each step.

V. Conclusion

This paper has presented a sampling-based approach to six-DOF spacecraft rendezvous trajectory planning for orbital debris mitigation. The trajectory planning was achieved via Sampling-Based Model Predictive Optimization (SBMPO), a sampling-based algorithm that operates within the system input space.

Facilitating the efficiency of the randomized A* algorithm, this paper presented the construction of two appropriate, optimistic A* heuristics that are based on the solutions to the minimum distance and minimum time control problems. These heuristics enable rapid computation of rendezvous trajectories that end in zero relative velocity.

Considering an obstacle-free planning environment, SBMPO was applied to the spacecraft relative motion model to determine a rendezvous-feasible, minimum distance trajectory. The presentation of the obstacle-free simulation in this paper served to demonstrate the effectiveness of the optimistic A* heuristic and as a comparative baseline.

A necessity when used in orbital debris mitigation missions, SBMPO was demonstrated to be capable of generating rendezvous trajectories in the presence of known obstacles. By implementing a well-developed minimum time heuristic, the algorithm rapidly converged to an optimal solution trajectory that accommodates the rendezvous scenario. Furthermore, the computational performance, despite the obstacles, was comparable to that of obstacle-free planning.

When planning in the presence of obstacles, the efficiency of the algorithm was, in part, due to the use of the imminent collision detection method that is highlighted in Figure 3(b). Although not explicitly presented in the results, it should be noted that this collision detection method may be most effective when deployed in scenarios with a high population and/or high density of obstacles because the number of nodes explored is reduced when imminent collision detection is employed.

Based on the computational speed of the algorithm, this approach is a likely candidate for use in real-time guidance and navigation. Lending to the viability of future autonomous orbital debris removal missions, the computationally rapid results shown in this paper indicate the promising potential for the deployment of SBMPO in autonomous rendezvous and docking problems.

Future work will focus on the implementation of rapid replanning in the algorithm, the identification of planning issues associated with onboard perception of obstacles, and the development of additional optimization metrics, such as minimum fuel consumption. Additionally, future study will investigate unique planning issues associated with spacecraft rendezvous and docking, such as thruster plume impingement and accommodating additional environmental constraints.

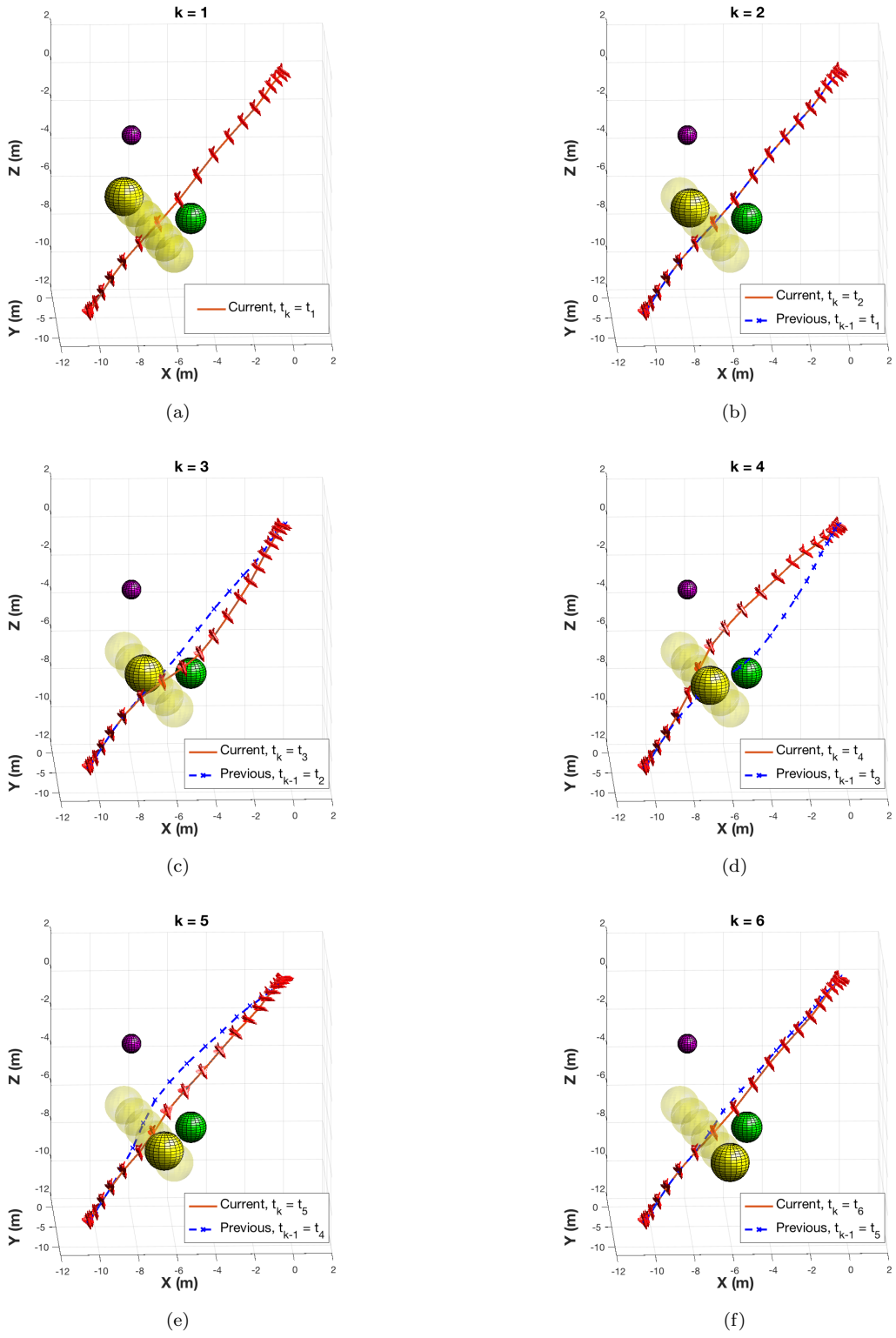


Figure 10. Minimum time rendezvous trajectories computed using SBMPO with iterative replanning to accommodate a moving obstacle.

References

[1]Obama, B., *National space policy of the United States of America*, Executive Office of the President, 2010. [Online] `http://books.google.com/books?id=lNTDYgEACAAJ`

[2]Liou, J., and Johnson, N.,J. Liou and N. Johnson, "A sensitivity study of the effectiveness of active debris removal in LEO," *Acta Astronautica*, vol. 64, no. 2-3, pp. 236–243, 2009.

[3]Shoemaker, J., and Wright, M., "Orbital Express space operation architecture program," in *Proceedings of SPIE*, vol. 5419, 2004.

[4]Creamer, G., "The SUMO/FREND project: Technology development for autonomous grapple of geosynchronous satellites," in *30th Annual AAS Guidance and Control Conference*, 2007.

[5]Rumford, T., "Demonstration of autonomous rendezvous technology (DART) project summary," in *Proceedings of SPIE*, vol. 5088, 2003.

[6]Breger, L., and How, J., "Safe trajectories for autonomous rendezvous of spacecraft," *Journal of Guidance, Control and Dynamics*, vol. 31, no. 5, pp. 1478–1489, 2008.

[7]Richards, A., Schouwenaars, T., How, J., and Feron, E., "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[8]Zhang, J., Tang, G., Luo, Y., and Li, H., "Orbital rendezvous mission planning using mixed integer nonlinear programming," *Acta Astronautica*, 2010.

[9]Zhang, D., Song, S., and Pei, R., "Dynamic obstacle avoidance of autonomous rendezvous and docking using potential function guidance based-fuzzy logic system," in *IEEE International Symposium on Systems and Control in Aeronautics and Astronautics (ISSCAA)*, pp. 34–39, 2010.

[10]Wang, P., Chen, C., Guo, J., and Cui, N., "Randomization-based algorithms for real-time servicing spacecraft motion planning on elliptical orbit," in *IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 4924–4929, 2009.

[11]Aoude, G., How, J., and Garcia, I., "Two-stage path planning approach for designing multiple spacecraft reconfiguration maneuvers," in *International Symposium on Space Flight Dynamics*, 2007.

[12]Henshaw, C., and Sanner, R., "Variational technique for spacecraft trajectory planning," *Journal of Aerospace Engineering*, vol. 23, p. 147, 2010.

[13]Karaman, S., and Frazzoli, E., "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[14]Reese, B., and Collins, E., "A Graph Search and Neural Network Approach to Adaptive Nonlinear Model Predictive Control", *Engineering Applications of Artificial Intelligence*, vol. 54, no. 10, pp. 3002–3014, 2016.

[15]Dunlap, D. D., Caldwell, C. V., Collins, E. G., and Chuy, O., "Motion planning for mobile robots via sampling-based model predictive optimization," in *Mobile Robots*, InTech, 2011.

[16]Dunlap, D. D., Caldwell, C. V., and Collins, E. G., "Nonlinear model predictive control using sampling and goal-directed optimization," in *IEEE International Conference on Control Applications (CCA)*, pp. 1349 –1356, 2010.

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

[18]Caldwell, C. V., Dunlap, D. D., and Collins, E. G., "Motion planning for an autonomous underwater vehicle via sampling based model predictive control," in *OCEANS 2010*, pp. 1 –6, 2010.

[19]Ordonez, C., Gupta, N., Collins, E. G., Clark, J. E., and Johnson, A. M., "Power modeling of the XRL hexapedal robot and its application to energy efficient motion planning," in *Proceedings of the 15th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, Baltimore, MD, July 23-26, 2012.

[20]Gupta, N., Ordonez, C., and Collins, E. G., "Dynamically Feasible, Energy Efficient Motion Planning for Skid-Steered Vehicles," *Autonomous Robots*, Vol. 41, Issue 2, Feb. 2017, pp. 453-471. [Online] `http://link.springer.com/article/10.1007/s10514-016-9550-8`.

[21]Ordonez, C., Gupta, N., Chuy, O., and Collins, E. G., "Momentum Based Traversal of Mobility Challenges for Autonomous Ground Vehicles," in *IEEE Conference on Robotics and Automation*, May 2013, Munich, Germany.

[22]Chuy, O., Collins, E. G., Sharma, A., and Kopinsky, R., "Using Dynamic to Consider Torque Constraints in Manipulators Planning with Heavy Loads," *ASME Journal of Dynamic Systems, Measurement, and Control*, Nov. 2016, DOI: 10.11151.4035168.

[23]Mann, T., "Application of Sampling-Based Model Predictive Control to Motion Planning for Robotic Manipulators," M. S. Thesis, Florida State University, December 2010.

[24]Hartley, E., Gallieri, M., and Maciejowski, J. M., "Terminal spacecraft rendezvous and capture with LASSO model predictive control," *International Journal of Control*, vol. 86, no. 11, pp. 2104–2113, 2013.

[25]Maciejowski, J., *Predictive Control with Constraints*, Pearson Education, 2002.

[26]Koenig, S., Likhachev, M., and Furcy, D., "Lifelong Planning A*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[27]Halton, J. H., "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numerishe Mathematik*, 1960.

[28]Ericson, C., *Real-Time Collision Detection*, Elsevier, New York, 2005.

[29]McCamish, S., Romano, M., and Yun, X., "Autonomous distributed control of simultaneous multiple spacecraft proximity maneuvers," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 630–644, 2010.

[30]Bryson, A. E., and Ho, Y., *Applied Optimal Control: Optimization, Estimation, and Control*, Hemisphere Publishing Corporation, 1975.

[31]Fehse, W., *Automated rendezvous and docking of spacecraft*, Cambridge University Press, 2003.

[32]LaValle, S., *Planning Algorithms*, Cambridge University Press, 2006.

[33]Sidi, M., *Spacecraft Dynamics and Control: A Practical Engineering Approach*, Cambridge University Press, 2000.