DEVELOPING AN INTEGRATED GUIDANCE AND CONTROL SYSTEM FOR REACTIVE FREE-FLYER MANEUVERING

A Thesis

by

MICHAEL C. MCCARTHY

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Gregory Chamitoff
Committee Members,	Srinivas Vadali
	Daniel Selva
	Nancy Currie-Gregg
Head of Department,	Ivett Leyva

May 2022

Major Subject: Aerospace Engineering

ABSTRACT

The use of highly autonomous free-flying spacecraft has been investigated for potential utility in future human spaceflight endeavors. In general, 'free-flyer' robots are small, self-sufficient spacecraft that operate near the exterior of larger space structures, such as the International Space Station, and are designed to provide support during various operations. Free-flyer designs and concepts often include a large degree of autonomy to provide mission support with little operational overhead.

One of the building blocks of autonomous free-flyer behavior is safe and reliable point-to-point maneuvering during proximity operations. This thesis explores the development and simulation of an integrated guidance and control (G&C) system to enable safe free-flyer point-to-point maneuvering in proximity to larger space structures, including the avoidance of collision and jet plume impingement. The foundation for this system is an existing trajectory planning method introduced by Roger [1]. This method represents the free-space as a discrete harmonic potential field and uses the resulting field gradient as a condition for generating collision-free trajectories that efficiently account for natural dynamics. A real-time guidance process is built around this method that can manage an internal model of the environment based upon obstacle mapping data and react quickly to dynamic obstacles. A linear-programming jet selection technique is implemented to fulfill six DOF velocity impulse commands using a free-flyer's RCS propulsion system, and additionally is augmented to include jet plume impingement avoidance functionality. Finally, an attitude controller process was developed and implemented to enable the free-flyer to reach and track a desired attitude during translational maneuvers.

To verify the system's capabilities, a test-bed simulation was developed using the SpaceCRAFT platform, specifically utilizing it's modular, asynchronous architecture. In a set of four maneuvering tests set in distinct obstacle environments, the G&C system demonstrated the ability to maneuver the free-flyer to the goal state along collision-free trajectories. In three of the test cases, the plume avoidance strategy results in a large reduction in the accumulated plume cost (36-54%).

Overall, these simulation results demonstrate that the system enabled point-to-point maneuvering for a reference free-flyer design, and support its feasibility and practicality.

This work represents a somewhat unique approach to free-flyer autonomous maneuvering in that it departs from a trajectory planning, or offline-online paradigm. Instead, this approach relies on the refinement of an internal model of the environment and the resulting potential field to perform reactive path-finding. This approach results in better overall flexibility when the free-flyer lacks knowledge of the position, geometry, and motion of nearby obstacles. In the future, the integration of Simultaneous Localization and Mapping (SLAM) and 3D mapping algorithms will help to further verify the feasibility of this approach. Other future improvements include integrating more robust methods of dynamic obstacle avoidance and automated positioning and scaling of the potential field grid.

DEDICATION

To my wonderful parents, sister, and fiancée for their endless love and support.

ACKNOWLEDGMENTS

I would like to start by expressing my deep gratitude to Dr. Gregory Chamitoff for his mentorship and support over the last several years. Working with Dr. Chamitoff in both my graduate and undergraduate studies has exposed me to countless opportunities and broadened my horizons as a student far beyond what I had previously imagined for myself. Additionally, I would like to collectively thank the current and former students of the ASTRO Center, for all of their ambitious work on the SpaceCRAFT platform and related projects as well as for making the lab such a great place to learn and innovate.

The work presented in this thesis was performed during a difficult time for many around the world. The COVID-19 pandemic continues to fundamentally change the structure of everyday life, especially with the rise of remote-work. This thesis work was completed almost entirely remotely, and therefore I would like to thank my committee members and academic advisors for their time and their flexibility in these circumstances.

Finally, I would like to thank my fiancée Nicole and my family for their encouragement and for keeping me grounded in this long, and sometimes difficult process. Only because of this love and support was I able to achieve my goals.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Gregory Chamitoff acting as advisor and Professors Srinivas Vadali and Daniel Selva of the Department of Aerospace Engineering and Professor Nancy Currie-Gregg of the Department of Industrial and Systems Engineering.

The SpaceCRAFT simulation software used to produce the results presented in this thesis was developed by the students of the Texas A&M ASTRO Laboratory, including the author, under the guidance of Professor Gregory Chamitoff. All other work conducted for the thesis was completed by the student independently.

Funding Sources

During the research activities presented in this document, financial support was provided through Teaching Assistantships in the Department of Aerospace Engineering and research grants and contracts from the ASTRO Laboratory.

NOMENCLATURE

ASTRO	Admissible Subspace Trajectory Optimizer
AERCam	Autonomous Extravehicular Robotic Camera
COTS	Commercial Off-the-Shelf
CW	Clohessy-Wiltshire
DOF	Degrees of Freedom
EVA	Extravehicular Activities
FOV	Field of View
GIM	Gradient Impulse Maneuvering
GN&C	Guidance, Navigation, and Control
G&C	Guidance and Control
ISS	International Space Station
JSC	Johnson Space Center
LQR	Linear Quadratic Regulator
LTI	Linear-Time-Invariant
LVLH	Local-Vertical-Local-Horizontal
NASA	National Aeronautics and Space Administration
PFG	Potential Field Guidance
RCS	Reaction Control System
RRT	Rapidly-exploring Random Trees
SLAM	Simultaneous Localization and Mapping
SPHERES	Synchronized Position Hold, Engage, Re-orient Experimental Satellites
VR	Virtual Reality

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Modern human spaceflight is made possible due to the extensive preparation, planning, training, verification, and validation needed to make sure a mission is successful. Each stage of a mission is carefully analyzed and rehearsed beforehand such that any risks, uncertainty, or unexpected failures can be avoided. During the mission, large teams of operators, flight controllers, and support engineers, along with the crew themselves, coordinate to achieve mission objectives safely. This operational overhead is a large barrier to increasingly complex mission operations. As humans seek to expand their presence beyond Earth in ambitious ways, larger space vehicles, stations, and other structures will be built on-orbit, which will require ever-greater operational demands.

For future missions with unprecedented operational challenges, more autonomous technologies will need to be developed and relied upon. Tasks once relegated entirely to the crew will need to be fully or partially automated. This thesis seeks to develop G&C methods to enhance the practicality of one such autonomous technology of significant interest: the robotic free-flyer.

1.1 Robotic Free-Flyers

The free-flyer concept has received attention from both industry and academia for the past couple of decades for its potential to assist with human spaceflight operations. Generally, a free-flyer is an unmanned and untethered robot that can operate in zero gravity through the use of its own on-board propulsion system and is typically intended to operate inside or near the exterior of large space structures like the International Space Station. Free-flyer designs often feature some level of autonomous behavior, through the use of on-board computers, sensors, and actuators. This behavior most often includes the ability to navigate the local environment autonomously such that the robot can reach and hold a desired position without extensive maneuver planning by operators. Free-flyer propulsion systems generally consist of an array of small thrusters or even ducted fans in an internal pressurized environment [3]. A free-flyer's size, shape, and payload depend on the

specific application.

1.1.1 Past Missions and Designs

There have been several past projects developing the concept of free-flying robotic space vehicles. NASA's AERCam Sprint was a small free-flying robotic satellite flight tested on STS-87 in 1997 [4]. Sprint aimed to test the viability of a free-flyer for the role of providing robotic inspection and remote viewing capabilities to assist with extravehicular operations. The AERCam Sprint was flown remotely by an astronaut inside the shuttle, who could view the vehicle from the orbiter window and had access to the robot's camera feed. This test flight was successful and supported the further use of free-flyers for inspection operations. In a later project, the Mini AERCam was developed to expand on the functionality of its precursor and included automated navigation, point-to-point and hold maneuvering, and a considerable reduction in size [5]. With these advanced functionalities, Mini AERCam aimed to provide a wide variety of operational services, including close-up component inspection, anomaly detection, photogrammetry, and sensor positioning [5]. Mini AERCam was integrated and tested, but never flown.

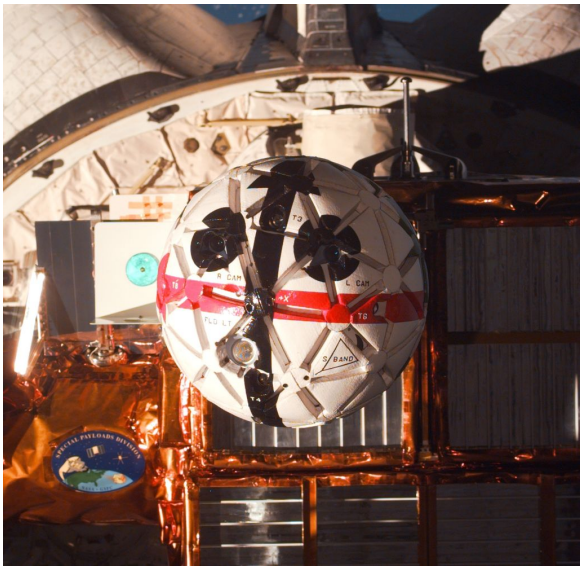

Figure 1.1: AERCam Sprint (Source: NASA) [2]

Figure 1.2: Mini AERCam (Source: NASA) [2]

More recently, the Seeker free-flyer inspector vehicle was developed by JSC as a low-cost effort to continue the development of more advanced autonomous vehicle inspection systems. Seeker's design featured a GN&C system comprised entirely of COTS components, capable of point-to-point maneuvering and relative position holding [6]. Seeker was flown and tested during the Cygnus NG-11 mission, and demonstrated several of it's autonomous inspection capabilities successfully.

While extravehicular operations have been the driving focus of the free-flyer concept, several projects have investigated the use of free-flyers inside large spacecraft, such as the ISS. For example, the SPHEREs project developed a small set of intravehicular free-flyers that were flown to the ISS to serve as a test-bed for formation flying and autonomous maneuvering algorithms [7]. More recently, the Astrobee free-flyers were delivered to the ISS to assist crew with the completion of tasks and procedures and serve as a next-generation platform for free-flying robotics research [8]. Other intravehicular free-flyers have been proposed for use as robotic astronaut assistants, such as the AAR-2 design detailed in [3].

1.1.2 Future Mission Concepts

All of the previous free-flyer concepts and designs were quite small and featured payloads intended to assist crew in operations such as EVA and vehicle inspection. The Mini AERCam, AERCam Sprint, and Seeker vehicle payloads were notably all cameras, so that they could perform inspection and remote viewing tasks. Mini AERCam and Seeker also featured autonomous maneuvering capabilities on some level with the intent of working towards more complex autonomous behavior [5], [6].

There is significant operational potential for autonomous free-flyers, especially for the construction and maintenance of future space stations and other habitable orbital structures. Autonomous free-flyers could be deployed to conduct or assist with procedures and tasks normally performed by the crew during EVAs, such as repair procedures or component installations. For example, a free-flyer outfitted specifically for performing repairs could maneuver autonomously to a specific area, anchor itself to the structure, and perform repair tasks with its payload of tools and manipulators. Another complex task suitable for autonomous free-flyers would be the execution of routine vehicle scanning and anomaly detection, where a free-flyer is deployed to conduct a health survey of the habitable structure using its on-board cameras and sensors. Such a task, if performed by the crew during EVA or using manually controlled robotics, would take significant time to plan and execute. A free-flyer with autonomous maneuvering capabilities could perform these types of scans routinely.

Some of these complex tasks will likely require future free-flyer designs to carry larger, more complex payloads than the cameras and sensors carried by existing designs such as Mini AERCam and Seeker. To support larger payloads, free-flyers will need to have larger structures and propulsion systems as a result. This would increase the risks of operating free-flyers in close proximity to habitable structures, including the risk of vehicle collision or the impingement of free-flyer RCS jets on structural components. The mitigation of these risks is an important aspect of making future free-flyer mission concepts more operationally feasible.

1.2 Major Challenges

The problem of designing, building, and operating free-flyers with the previously discussed autonomous capabilities presents a multitude of engineering challenges to overcome. While different free-flyer applications will require distinct design choices, the common thread among future mission concepts is the need for autonomous point-to-point maneuvering capabilities. Therefore, regardless of the specific application, the development of a G&C system that can meet the unique challenges of free-flyer operations and mitigate key risks is critical.

1.2.1 Collision Avoidance

The flight domain of the free-flyer is characterized by extremely close proximity to larger space structures with potentially complex geometry. The ISS is the current obvious example of such an environment, where the various modules, truss sections, and docked vehicles create a very complex structural environment to navigate. Besides purely structural elements, the ISS also has many external components that are particularly vulnerable, including solar arrays, radiator panels, communication antennas, and scientific instrumentation. Some of these components, such as solar arrays and antennas, may be moving independently from the structure. It is likely that future space stations will have these same vulnerable structural features.

Extensive navigation of this structural environment is an essential, and unique aspect of free-flyer operations. During more typical proximity operations the extent of close-proximity maneuvering is minimized. For example, when a visiting spacecraft docks to the ISS, it first performs maneuvers at a safe distance from the station to attain a position that is lined up for the final approach. This eliminates the risk of collision for large maneuvers, but can be relatively time and fuel inefficient. For most applications, free-flyers are required to fly in close-proximity to the structure. Therefore, safe and robust collision avoidance is an essential aspect of free-flyer autonomous maneuvering.

1.2.2 Jet Plume Impingement

A vehicle operating in close proximity to an orbital structure or large spacecraft usually relies on its RCS thrusters for both translational and attitude control. When a spacecraft fires its RCS jets the resulting rocket exhaust plumes can impinge upon nearby surfaces creating a multitude of problems. Jet plume impingement is an area of significant research both in the modelling of jet plumes and in schemes to reduce and minimize impingement.

One of the most significant consequences of jet plume impingement is the potential to impart momentum onto nearby structures or vehicles. During a variety of proximity operations, jet plume impingement on nearby vehicles can affect their motion such that operations like rendezvous, docking, or payload retrieval become impossible [9]. This was a large consideration for the Space Shuttle orbiter, whose various RCS thrusters posed significant risk of plume impingement during proximity operations with the ISS. Although smaller, modern cargo spacecraft like Northrop Grumman's Cygnus are subject to similar considerations, where risk of plume impingement to nearby vehicles or even components of the same vehicle can cause undue disturbances [10]. Large plume impingement forces can also cause damage to delicate components such as solar panels and radiators, due to structural loads that exceed the design specifications.

Besides disturbance forces, plume impingement can also cause damage to components through heating and surface contamination. For combustion based RCS thrusters, heating and thermal issues due to plume impingement of high temperature rocket exhaust can lead to component or structural failure [11]. For an even broader category of RCS thrusters, expelled gasses and reactants can cause surface contamination components such as solar arrays, cameras, and other instruments.

Existing free-flyer designs and concepts have all been small, and therefore the risk of plume impingement from their miniaturized propulsion systems is limited. Many of the plume impingement effects are negligible or nonexistent for these small free-flyers. For example, the Mini AERCam free-flyer design included a pressurized xenon gas propulsion system, which poses no risk of undue heating to nearby structures [5].

For future free-flyer designs and operations, the increase in size and mass of free-flyer robots

will require larger propulsion systems. Therefore, free-flyer control systems will need to incorporate techniques to actively reduce the occurrence and severity of plume impingement.

1.2.3 Real-Time Operation

During typical proximity operations, vehicle maneuvering is subject to extensive pre-flight design and analysis. During autonomous flight, the desired trajectories are loaded into and executed by the vehicle's GN&C system. Large deviations from the intended trajectory, due to some disturbance or unplanned event, would require a new trajectory to be defined.

For the conceptual free-flyer operations discussed previously, this kind of uncertainty is the rule, rather than the exception. For maximum utility, free-flyers should be able to respond to new commands generated by mission operators or crew members in real time. Additionally, free-flyers need the ability to react to moving structural components, crew members performing EVA, or even other free-flyers. These capabilities require guidance that includes path-finding or trajectory planning algorithms.

Highly automated guidance algorithms can be very computationally expensive. This highlights a major constraint for realizing advanced free-flyer autonomous behavior. Any applied algorithms must be feasibly implementable on a free-flyer's flight computers for real-time operation and, therefore, are limited by the available computational power and memory.

1.3 Research Objectives

The objectives of this thesis are to both design and test a new G&C system suited to the specific challenges of autonomous free-flyer maneuvering. To guide the design of this system, a set of key capabilities have been identified. These capabilities are outlined in Table 1.1. To test the proposed system, a real-time simulation will be developed to simulate a free-flyer vehicle in an orbital obstacle environment.

Capability	**Description**
Point-to-Point Maneuvering	The system must enable the vehicle to maneuver from an initial position and orientation to a desired position and orientation.
Structural Collision Avoidance	To ensure operational safety the system must be able to achieve vehicle trajectories that reliably avoid collision with nearby static obstacles and structures.
Dynamic Object Collision Avoidance	The system must be able to avoid collision with nearby non-structural entities, such as other vehicles or crew undergoing EVA, which generally have unknown motion.
Generalized RCS Control	The system must be capable of vehicle control using on/off jet firing commands for a general vehicle RCS configuration.
Plume Impingement Avoidance	The system needs to incorporate methods to reduce the occurrence of jet plume impingement on nearby structures and components.
Implementation Feasibility	The system processes must be structured and developed to be computationally feasible on flight hardware in real-time.
Real-Time Operation	The system should should provide vehicle guidance commands and perform optimization real-time, with no predictive elements or dependence on trajectory planning.

Table 1.1: Key System Capabilities

1.4 Outline of Thesis

The structure of this thesis is outlined in this section. First, in Chapter 2, the dynamics of proximity operations and path-finding concepts will be reviewed. Also in this chapter, related works that apply path-finding concepts to proximity operations or similar problems will be explored.

The next three chapters will develop the methodology of this thesis. Figure 1.3 visualizes the structure of these chapters in a Venn diagram. First a guidance technique that can enable free-flyer autonomous maneuvering is introduced in Chapter 3. Next, Chapter 4 develops other functional components needed to build an integrated G&C system around this guidance method. This includes developing processes to handle rotational motion, perform jet selection, and optimize for jet plume impingement avoidance. Lastly, Chapter 5 details the design of the test-bed simulation developed to assess the behavior and performance of the G&C system.

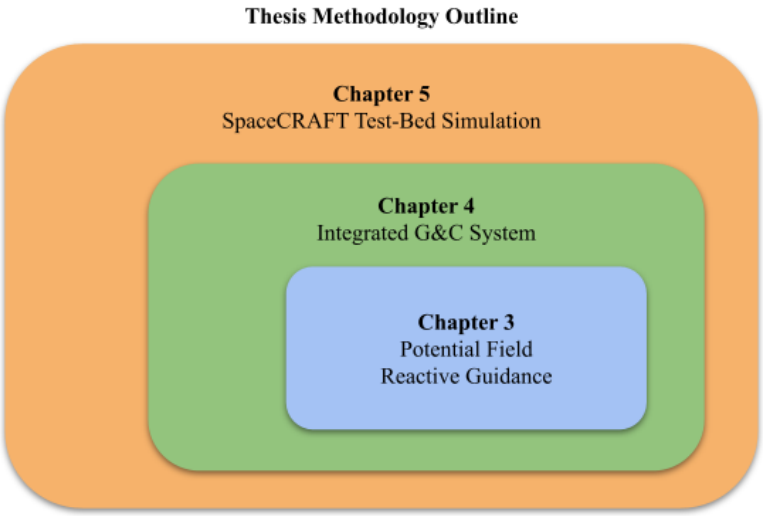

Figure 1.3: Thesis methodology chapter Venn diagram

2. BACKGROUND AND LITERATURE REVIEW

2.1 Free-Flyer Dynamics

The dynamics of free-flyer flight are common to any vehicle involved in proximity operations. These translational and rotational dynamics are nonlinear. For free-flyer operations, however, certain simplifications can be applied to both translational and rotational motion with minimal inaccuracy. These simplified models provide a basis for both analysis and simulation of free-flyer G&C systems.

2.1.1 Relative Orbital Motion

Spacecraft translational motion during proximity operations is governed by the nonlinear relative motion that exists between two orbiting satellites. For satellites in close proximity following circular orbits, this relative motion can be modelled by the linear Clohessy-Wiltshire (CW) differential equations developed in [12]. The CW equations (2.1a), (2.1b), and (2.1c) describe the relative position and velocity between two satellites with respect to a Local-Vertical-Local-Horizontal (LVLH) reference frame centered at one of the satellites, usually designated the *target*.

$$\ddot{x} = 2\omega_0 \dot{z} + u_x \tag{2.1a}$$

$$\ddot{y} = -\omega_0^2 y + u_y \tag{2.1b}$$

$$\ddot{z} = -2\omega_0 \dot{x} + 3\omega_0^2 z + u_z \tag{2.1c}$$

An LVLH reference frame is shown in Figure 2.1. The LVLH axes are defined as follows: the z-axis points directly downwards towards the body, the y-axis is orthogonal to the orbital plane, and the x-axis completes the right-handed triad. The LVLH frame is not an inertial frame and rotates as the target vehicle orbits, the effect of which is accounted for in the equations through the parameter ω_0, or the angular rate of the target vehicle's orbit. Lower, faster orbits will have more

dramatic relative motion dynamics than higher, slower orbits.

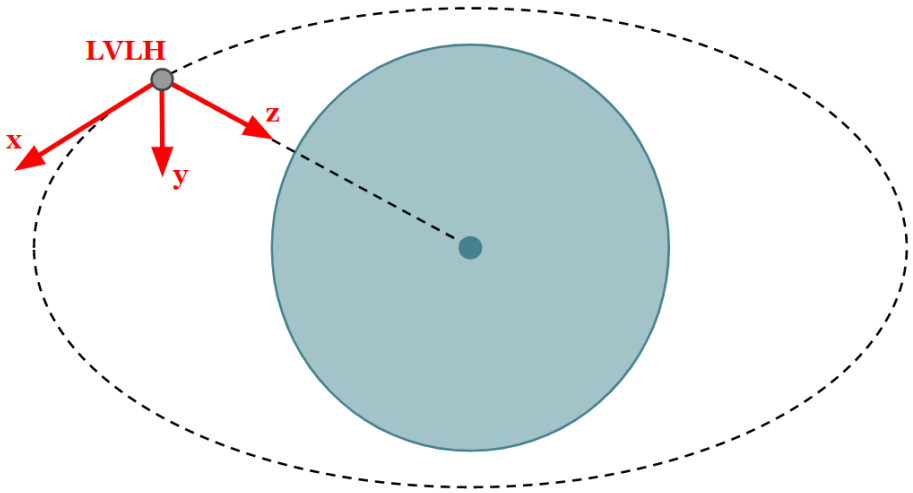

Figure 2.1: The Local-Vertical-Local-Horizontal coordinate frame (LVLH)

The CW equations represent a linear-time-invariant (LTI) system, whose analytical solution can be used to describe the unforced relative motion of one satellite with respect to the target. The translational state vector $\mathbf{x}(t) = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$ can be described at any future time using the state transition matrix $\Phi(t, t_0)$ and an initial state $x(t_0)$ as shown in equation (2.2).

$$\mathbf{x}(t) = \Phi(t, t_0)\mathbf{x}(t_0) \tag{2.2}$$

$$\phi(t,t_0) =$$

$$\begin{bmatrix} 1 & 0 & 6(\omega_0\Delta t - \sin(\omega_0\Delta t)) & \frac{4\sin(\omega_0\Delta t)-3\omega_0\Delta t}{\omega_0} & 0 & \frac{2(1-\cos(\omega_0\Delta t)}{\omega_0} \\ 0 & \cos(\omega_0\Delta t) & 0 & 0 & \frac{\sin(\omega_0\Delta t)}{\omega_0} & 0 \\ 0 & 0 & 4-3\cos(\omega_0\Delta t) & \frac{2(\cos(\omega_0\Delta t)-1)}{\omega_0} & 0 & \frac{\sin(\omega_0\Delta t)}{\omega_0} \\ 0 & 0 & 6\omega_0(1-\cos(\omega_0\Delta t) & 4\cos(\omega_0\Delta t)-3 & 0 & 2\sin(\omega_0\Delta t) \\ 0 & -\omega_0\sin(\omega_0\Delta t) & 0 & 0 & \cos(\omega_0\Delta t) & 0 \\ 0 & 0 & 3\omega_0\sin(\omega_0\Delta t) & -2\sin(\omega_0\Delta t) & 0 & \cos(\omega_0\Delta t) \end{bmatrix}$$

$$where\ \Delta t = t - t_0 \quad (2.3)$$

From the state transition matrix (2.3), the resulting sinusoidal motion is apparent. This matrix leads to a more intuitive understanding of what kind of motion to expect for a given relative position.

2.1.2 Impulsive Maneuvers

Using equations (2.2) and (2.3) together yields the unforced motion of a satellite with respect to the target. Vehicle control during proximity operations is commonly achieved using small impulsive maneuvers to keep the spacecraft moving along a desired trajectory. The short firing intervals of a spacecraft's RCS thrusters can be approximately modeled as an impulse, resulting in an instantaneous change in velocity. Therefore, vehicle trajectories in proximity operations can be accurately modelled as a series of unforced motion curves, punctuated by instantaneous impulsive maneuvers.

The state transition matrix Φ can be partitioned such that equation (2.2) becomes (2.4), where $\mathbf{r}(t) = [x, y, z]^T$ and $\mathbf{v}(t) = [\dot{x}, \dot{y}, \dot{z}]^T$ are the relative position and velocity vectors, respectively.

$$\begin{bmatrix} \mathbf{r}(t) \\ \mathbf{v}(t) \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \begin{bmatrix} \mathbf{r}(t_0) \\ \mathbf{v}(t_0) \end{bmatrix} \quad (2.4)$$

Given a goal time and position, t_1 and $\mathbf{r}(t_1)$, equation (2.4) can be rearranged to solve for the

velocity required at time t_0.

$$\mathbf{v}(t_0) = \Phi_{12}^{-1}(\mathbf{r}(t_1) - \Phi_{11}\mathbf{r}(t_0)) \tag{2.5}$$

The difference between the current velocity, and the velocity given by equation (2.5) gives the impulsive ΔV needed to achieve the goal position $r(t_1)$ at time t_1. This maneuvering scheme is called impulsive maneuvering, and provides a basic method of point-to-point maneuvering during proximity operations.

2.1.3 Rotational Motion

The rotational motion of a free-flyer can be modelled using rigid body rotational dynamics. Euler's equations of rotational motion are shown in (2.6).

$$\boldsymbol{\tau} = I^B\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (I^B\boldsymbol{\omega}) \tag{2.6}$$

Where $\boldsymbol{\tau}$ is the external torque acting on the body and $\boldsymbol{\omega}$ is the body's angular velocity. The components of this equation are expressed with respect to a coordinate frame fixed to the vehicle, and located at its center of mass. I^B is the inertia tensor of the vehicle expressed in this body frame.

$$I^B = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}^B \tag{2.7}$$

If the body-fixed coordinate frame is aligned with the vehicle's principal axes, I^B becomes diagonal and the rotational equations can be expressed as the system (2.9a), (2.9b), and (2.9c).

$$I_{xx}\dot{\omega}_x = (I_{zz} - I_{yy})\omega_z\omega_y + \tau_x \tag{2.8a}$$

$$I_{yy}\dot{\omega}_y = (I_{xx} - I_{zz})\omega_x\omega_z + \tau_y \tag{2.8b}$$

$$I_{zz}\dot{\omega}_z = (I_{yy} - I_{xx})\omega_y\omega_x + \tau_z \tag{2.8c}$$

From this system of equations, the effect of coupling between the principal inertia components

is apparent. For a vehicle with principal inertia components that are equal in value, the system reduces to the following set of linear equations.

$$I_{xx}\dot{\omega}_x = \tau_x \tag{2.9a}$$

$$I_{yy}\dot{\omega}_y = \tau_y \tag{2.9b}$$

$$I_{zz}\dot{\omega}_z = \tau_z \tag{2.9c}$$

Such a simplification is not applicable to a majority of spacecraft designs. However, some previously developed free-flyers had spherical structures that could have roughly equal inertia components. It is likely that this could be the case for futuristic free-flyer designs as well, such that this simplification would become useful.

The integration of any of the above forms of Euler's rotational equations will yield the response of the vehicle's angular velocity $\boldsymbol{\omega}$, due to any external torques. From there, the angular velocity can be used to integrate the rotational state of the vehicle using rotational kinematics.

2.2 Path-Finding Concepts

Path-finding is a fundamental problem of autonomous robotics, and has been solved in innumerable ways for an equally innumerable number of applications. Path-finding problems, at their core, concern the movement of an object from an initial state to a goal state along a collision-free path. There is a wide spectrum of sophistication involved in path-finding algorithms, and the best choice varies for different applications. Even for the specific problem of path-finding during proximity operations, there have been a wide variety of approaches and solutions, each with their own advantages and drawbacks. This section aims to introduce some general concepts related to path-finding that are utilized in this work. A more in-depth review of path-finding methods and concepts is given in [1].

2.2.1 Reactive Path-Finding

Among the many approaches to the problem of robotic path-finding, *reactive* path-finding methods tend to be among the simplest. In reactive path-finding, an object's path or motion responds directly to obstacles in real-time based on some rules or conditions. An example of a simple reactive path-finding method is side-stepping. In side-stepping, an object moves directly towards the goal position. If the object encounters an obstacle, it takes steps in the direction perpendicular to the goal direction until it is clear of obstruction. This behavior is repeated until the object reaches the goal position. While this algorithm is simple and effective in some situations, it is limited by the fact that it can become stuck in concave obstacles. Also, side-stepping brings the object into direct contact with the outer surface of obstacles in its path, which is undesirable for many applications where collision-avoidance is a priority.

There are other reactive methods that place more emphasis on avoiding collision with obstacles. In one method, attractive or repulsive forces are assigned to the goal and the obstacles, respectively. Similar to a positively charged particle in an electrostatic field, the object will be attracted to the goal while being repelled by nearby obstacles. This method produces smooth continuous paths from the start to the goal, with good clearance from obstacles, as illustrated in Figure 2.2. To apply this method to actual robotics, these attractive or repulsive forces are virtual, and the resultant force must be applied by the robot's control system. This approach forms the basic principles of path-finding using potential fields, which is covered in more detail in Section 2.2.4.

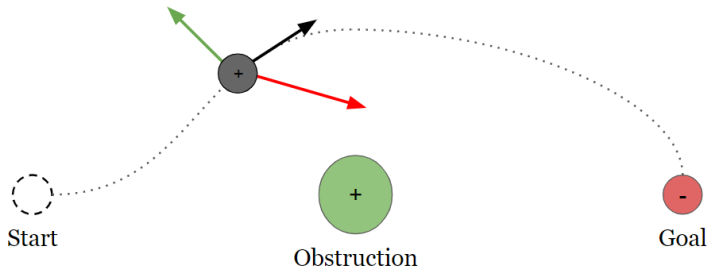

Figure 2.2: Reactive path-finding in the presence of artificial sources and sinks

2.2.2 Configuration Space Representations

Another class of path-finding methods are those that transform the problem into the *configuration space* to generate path-finding solutions, as initially introduced by [13]. The configuration of an object is defined as the set of parameters that accurately describe its state relative to the obstacles in the environment. The configuration space is therefore the set of all possible object configurations. In the configuration space, obstacle boundaries are defined and used to separate free and occupied regions. Solutions to path-finding problems can then be found by searching for a sequence of configurations, i.e. a path, through free-space that connects the initial and goal positions.

The dimensionality of a problem's configuration space depends on the number of object DOF that are relevant to path-finding. For a robotic arm with two joints, the configuration space is 2D. For a free-flyer maneuvering around space structures there are six DOF, but it is useful in most cases to only consider the three translational DOF for path-finding, which results in a 3D configuration space. In general, a configuration space approach to path-finding becomes more difficult for each additional DOF considered, so it is best to use the simplest possible representation.

2.2.3 Free-Space Graphs and Grids

Once a problem has been transformed into the configuration space, there are many different approaches to searching the free-space for path solutions. While continuous path solutions can be found in some cases, most path-finding methods discretize the free-space into a mathematical graph of configuration nodes, connected by free-space links. This approach was taken by Lozano-Pérez in an algorithm that can find a collision-free path for a 2D object among a field of polyhedral obstacles [13]. This problem was transformed into the configuration space by picking a reference point on the object and "growing" the polyhedral obstacles such that the free-space contains all safe positions of the reference point. The vertices of the newly grown obstacles, which are also polyhedrons, were then connected in a structure called a visibility graph. The graph nodes are connected through free-space, and so a path solution can be found by moving the object node to

node until it reaches the goal configuration. Search methods, such as Dijkstra's algorithm [14], can be applied to such graphs to find the shortest distance (or other performance criteria) between any two nodes.

In another early work utilizing a configuration space approach, Brooks [15], [16] generated a similar node graph by mapping the free-space regions between obstacles as generalized cones. The cone center-lines and intersections were used to define a node graph, which was then searched using the heuristic A* search algorithm [17] for increased efficiency.

Another method of discretizing the free-space is to define an evenly spaced grid of nodes, which spans a subset of the configuration space excluding regions occupied by obstacles. This approach avoids the challenge of deriving graph nodes from the free-space through analytical techniques, but creates a computationally difficult search for optimal solutions.

2.2.4 Potential Fields

Another large class of path-finding methods are those that use potential fields to generate solutions. In section 2.2.1, a method for reactive path-finding based on the idea of attractive and repulsive forces was introduced. In that method, the net force acting on the object is a function of its proximity to obstacles and the goal position. If a scalar field could be defined such that its gradient at any position returned the same net force, the result would be a potential field. Potential fields are used in path-planning problems to represent obstacles in the configuration space. Obstacles are defined as areas of high potential, while the goal position is defined as the global minimum. To generate a path from any initial position to the goal position, an object needs only to descend the potential gradient, producing smooth path solutions with wide clearance from obstacles.

A potential function gives a closed-form formula for directly computing the potential values at a particular point in a configuration space. A simple potential function analogous to electrostatic potential is shown by equation 2.10.

$$\phi(\boldsymbol{r}) = \sum_i \frac{\delta}{\delta + |\boldsymbol{r}_i - \boldsymbol{r}|} + \alpha|\boldsymbol{r}_g - \boldsymbol{r}|^2 \tag{2.10}$$

Where $\boldsymbol{r}_i$ is the nearest point on the surface of obstacle i and $\boldsymbol{r}_g$ is the position of the goal point. δ and α are scaling constants. Using this equation, the potential field shown in Figure 2.3 was generated. Potential functions like equation (2.10) are useful because they allow for the direct computation of the potential value and gradient at any location in free-space. For example, Rimon-Kodishek [18] introduced "navigation functions" that include controller constraints, which are used directly to create a bounded-torque feedback controller for robotic path-planning.

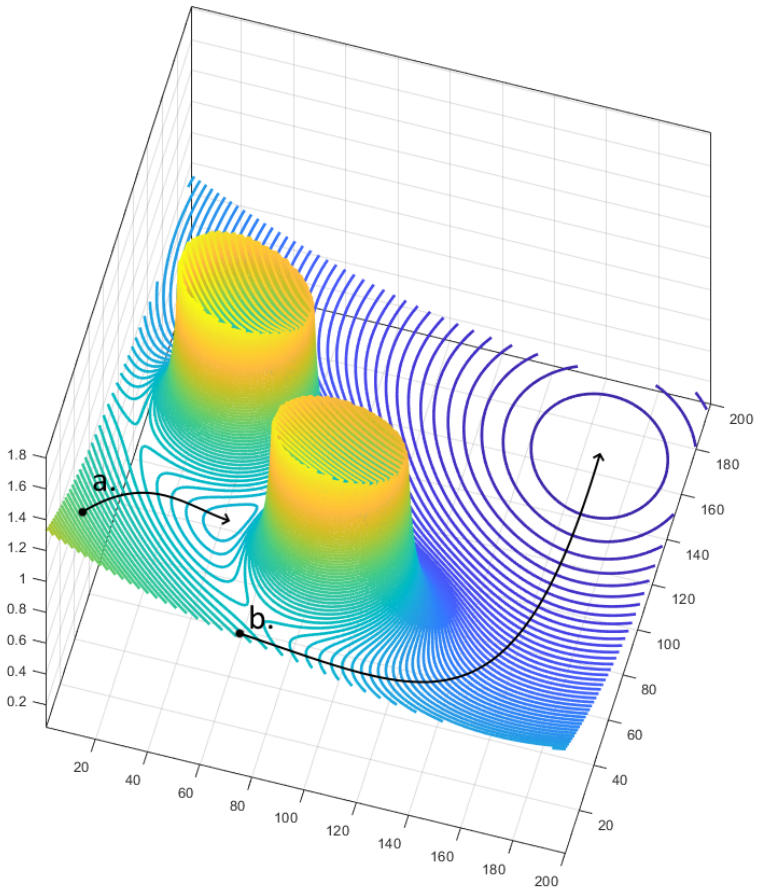

Figure 2.3: Visualization of a 2D potential field generated using a potential function similar to electrostatic potential

Potential functions such as (2.10) are prone to the creation of local minima, causing the object to sometimes converge to the incorrect configuration as shown by path a in Figure 2.3. In [19], an

attempt to eliminate local minima in certain cases through the augmentation of the potential field is outlined. This solution, however, only applies for local minima caused by the goal position being in close proximity to obstacle boundaries.

A better method for dealing with local minima is to use harmonic potential fields. Harmonic potential fields are scalar fields that satisfy Laplace's equation (2.11), a partial differential equation used to model several natural phenomena including incompressible fluid flow and steady-state temperature distributions.

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = 0 \tag{2.11}$$

Harmonic potential fields are useful for robotic path-finding problems, because they guarantee a single, global minimum. In [20], harmonic potential functions derived from a panel representation of obstacles was applied to path-finding for a planar mobile robot and a three DOF manipulator. An example of a harmonic potential field is shown in Figure 2.4 which represents the same configuration space as the field shown in Figure 2.3. In this example, the formation of local minima is avoided in the harmonic field.

Some path-finding methods have used potential fields for representation of obstacles, but without relying on gradient descent to produce solutions. The method introduced in [21] uses potential functions to represent a field of obstacles, and computes minimum-potential valleys that connect larger regions of free-space. These valleys form a node graph, similar to those described in 2.2.3, which is then searched for an optimal path solution. This method differs from a gradient descent method in that no attractive potential is assigned at the goal node.

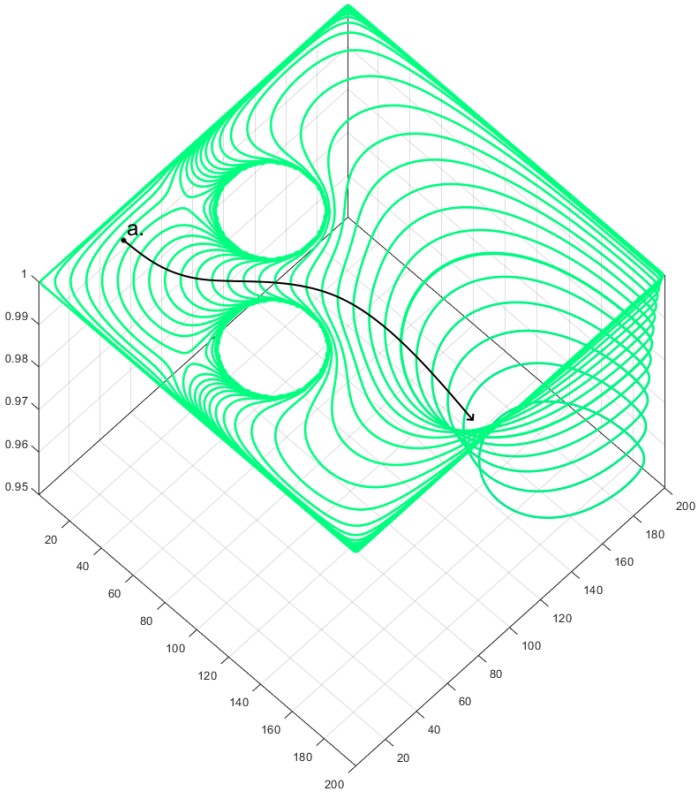

Figure 2.4: Visualization of a 2D harmonic potential field satisfying Laplace's equation

2.3 Applications of Path-Finding to Proximity Operations

Close-proximity maneuvering between two spacecraft is a challenging dynamics and controls problem with 6 DOF. Traditionally, proximity operations have required extensive manual trajectory design and analysis before flight. To aid in trajectory design and enable more autonomous maneuvering, there has been much research into applying path-finding concepts to proximity operations. This section aims to explore some of these works and how they relate to the specific needs of the free-flyer problem.

2.3.1 Trajectory Planning Methods

For most spaceflight operations, trajectory design and execution are distinct and separate processes. Usually, trajectories are designed and analyzed well in advance of a mission and then executed by a vehicle's GN&C system during flight. This separation of trajectory planning and

execution has been preserved in many path-finding algorithms applied to proximity operations. These algorithms, categorized here as trajectory planning methods, are useful because they allow for the integration of complex constraints to generate optimal trajectories. In some cases, these trajectory planner algorithms are fast enough for real-time use.

Two early trajectory planner algorithms developed in [22] and [23] use A* to generate optimal 6 DOF trajectories. In these algorithms, nodes that represent the vehicle state are iteratively expanded by taking small velocity increments, propagating the resulting trajectory, and estimating the costs associated with each node. A* is used to select the nodes with the lowest estimated cost and direct the search to the goal node. In [22], the resulting trajectories are fuel-optimal and account for constraints such as obstacles, jet failures, or keep-out regions. [23] builds off of this approach, and includes jet plume impingement constraints to produce fuel-plume optimal trajectories. The resulting trajectories are followed by a 6 DOF autopilot.

These A*-based algorithms are exploring the admissible space using a discrete set of nodes. Another algorithm called the Admissible Subspace Trajectory Optimizer (ASTRO) instead takes a continuous approach. ASTRO performs polynomial optimization in the presence of a flexible variety of constraints, including static and dynamic obstacles, path constraints, and actuator performance. ASTRO generates an admissible initial trajectory satisfying the boundary conditions and iteratively refines the trajectory towards an optimal or sub-optimal solution with the remaining time. Trajectory following is then delegated to an inner-loop controller. In [24], this algorithm is developed and tested in a true six DOF environment on the ISS using the SPHERES test-bed. [25] applies ASTRO to path planning for space-based robots and quadcopters, integrating additional techniques for localization and mapping of nearby obstacles.

Many more recently developed trajectory planning algorithms explore free-space using random sampling methods. One such algorithm is developed in [26], where a sampling method based on Rapidly-exploring Random Trees (RRT), called RRT*, is used to find optimized trajectories in the presence of an impressive number of constraints. Essentially, the algorithm takes random samples from the configuration space, and evaluates the cost due to various constraints including obstacle

avoidance, camera FOV, plume impingement, and control feasibility. The algorithm directs the sampling towards the more unexplored parts of the configuration space as it progresses. As the extent of sampling increases, the likelihood of finding a solution increases along with its overall quality. A similar algorithm is developed in [27], where random sampling based upon sphere expansion is used to construct a graph of the free-space. This graph is searched for the shortest path, and sequential convex programming is then used to generate a trajectory along this path that is locally optimal. Yet another algorithm is developed in [28] that uses random node sampling paired with machine learning to classify free-space "neighborhoods", from which a trajectory is generated. This particular algorithm is applied to quadcopters, and so an additional trajectory smoothing step is used.

2.3.2 Reactive Proximity Operations

Trajectory planner methods can generate optimal or near-optimal trajectories while considering a large variety of constraints. The main drawback of these algorithms is that re-planning of the trajectory could be required when a vehicle experiences unmodelled disturbances or the environment changes in an unexpected way. This weakness is especially apparent for free-flyer operations, where the ability to maneuver flexibly in the face of uncertainty has much utility. Also, it is desirable for a free-flyer to respond quickly to new position and orientation commands given in real-time, but trajectory planning algorithms require a complete solution to be found before execution is started. Some of the algorithms explored in the previous section, such as ASTRO [24], are likely fast enough for real-time use, but a more direct way to achieve flexible maneuvering is to develop algorithms that utilize reactive path-finding techniques such as potential fields.

One such algorithm is developed in [29] combining the use of potential functions with an LQR controller. This method produces a feedback controller with automatic obstacle avoidance, and is applied to the control of multiple spacecraft simultaneously and tested using SPHERES on the ISS. In [30], convex potential functions based on quaternion components are used to develop a feedback controller for spacecraft reorientation. These potential functions are used to enforce practical attitude constraints such as protecting sensitive instruments from sunlight exposure or

maintaining a communications linkage.

The integration of potential functions into feedback algorithms allows for fast and flexible maneuvering, but there are large drawbacks to this approach. First of all, feedback controllers that use the simpler potential functions can be susceptible to the formation of local minima. Secondly, these controllers do not account for the natural dynamics in the resulting control output, a flaw shared by many of the trajectory planning algorithms as well.

One method that addresses both of these drawbacks is developed in a thesis by Roger [1]. In this method, a trajectory planner is developed that builds a discrete harmonic potential field to represent the free-space in which the free-flyer can maneuver. The potential field gradient is then used as a condition for the generation of new velocity impulses. This method, called gradient impulsive maneuvering (GIM) does not suffer from local minima, and allows the natural dynamics to take over when the vehicle is moving in the direction of the goal. In the original work, this method is applied as a trajectory planner for an inspector free-flyer, but could be adapted for real-time use as a reactive path-finding method.

2.4 Discussion

At the core of the proposed free-flyer G&C system is the need for a robust and flexible path-finding algorithm. The related works covered in this chapter provide several potential approaches, but none that achieve all of the key capabilities on their own. The trajectory planning methods produce optimal trajectories with the inclusion of many constraints, but can lack flexibility in the face of uncertainty or are computationally infeasible for real-time use. Likewise, the reactive techniques introduced allow for flexible control, but struggle to handle more complex constraints and do not produce optimal trajectories. The technique by Roger [1] strikes an interesting balance between trajectory planning and reactive path-finding. The discrete artificial potential field utilized in this technique provides the information needed to navigate from any point in free space, within a subset of the configuration space, to the goal position. The descent of this potential field from any particular position produces a trajectory.

In Roger's work, potential field generation and descent using GIM is integrated into a trajec-

tory planning software tool, taking advantage of simplifications such as perfect, impulsive control and linearized dynamics. Instead, in this thesis the technique developed by Roger is adapted into a real-time free-flyer guidance process. To do this, the potential field is generated live and updated continuously to reflect new information about the environment. GIM is then applied as a control law, allowing the free-flyer to descend the potential field using its RCS propulsion system. This guidance process is then integrated into a broader G&C system that can handle jet selection, perform plume impingement avoidance, and control rotational DOF.

3. REACTIVE GUIDANCE WITH HARMONIC POTENTIAL FIELDS

The key challenge of autonomous free-flyer maneuvering is the need for robust collision avoidance with nearby space structures, spacecraft, or other obstacles. The technique developed by Roger [1] was selected in Chapter 2 as the foundation for a guidance process that can provide this capability. This chapter seeks to review the theory behind this technique and integrate it into a process for real-time free-flyer guidance.

3.1 Artificial Harmonic Potential Field Generation

The guidance technique employed in this thesis is reliant on the ability to generate artificial harmonic potential fields that represent the free-flyer's obstacle environment with respect to the goal position. Environmental obstacles, such as structural components of nearby spacecraft, are represented as volumetric areas of high potential. The goal position is represented as the global minimum of the field. To generate these artificial potential fields, Laplace's equation (2.11) is solved over a subset of the configuration space.

3.1.1 Discretization and Relaxation

Analytical solutions for Laplace's equation, especially in the presence of complex geometry, are infeasible to find. Instead of attempting to solve for the potential field analytically, this approach applies the numerical solutions outlined in [31].

To generate numerical solutions to Laplace's equation, a subspace of the configuration space is discretized into an evenly-spaced grid. This grid defines a control volume that encompasses a free-flyer's operational area. For proximity operations, no natural boundary exists to limit the extent of relative motion between two vehicles. Therefore, this control volume must be sufficiently large enough to not overly constrain potential path-finding solutions. The control volume is fixed with respect to an LVLH reference frame originating at the target spacecraft or space structure. Structural components rigidly attached to the target will therefore become fixed obstacles in the discrete grid. Other structures not fixed to either the target or the free-flyer itself will become

dynamic obstacles from a guidance perspective.

The discrete form of Laplace's equation is the difference equation shown in (3.1). To generate a harmonic potential field, this equation must be satisfied at every point in the discrete grid.

$$\begin{aligned}&\left(\frac{\phi_{i+1,j,k}-\phi_{i,j,k}}{\Delta x^2}-\frac{\phi_{i,j,k}-\phi_{i-1,j,k}}{\Delta x^2}\right)+\\&\left(\frac{\phi_{i,j+1,k}-\phi_{i,j,k}}{\Delta y^2}-\frac{\phi_{i,j,k}-\phi_{i,j-1,k}}{\Delta y^2}\right)+\\&\left(\frac{\phi_{i,j,k+1}-\phi_{i,j,k}}{\Delta z^2}-\frac{\phi_{i,j,k}-\phi_{i,j,k-1}}{\Delta z^2}\right)=0\end{aligned} \tag{3.1}$$

For a fixed distance $\Delta x = \Delta y = \Delta z = h$ between grid nodes in all directions, the discrete form of Laplace's equation can be rearranged into equation (3.2).

$$\phi_{i,j,k} = \frac{1}{6}(\phi_{i+1,j,k} + \phi_{i,j+1,k} + \phi_{i,j,k+1} + \phi_{i-1,j,k} + \phi_{i,j-1,k} + \phi_{i,j,k-1}) \tag{3.2}$$

Therefore, to satisfy Laplace's equation, the potential value at a particular node must be equal to the average value of all the adjacent nodes.

A numerical method called *relaxation* is used to generate discrete solutions to Laplace's equation, and is described in detail in [31] and applied in [1]. In relaxation, each grid node's potential is initialized to a constant high value, often one. The grid node nearest to the goal position, is assigned a low potential value, conventionally zero. Then, equation (3.2) is applied to each free-space node in the grid. This calculation is carried out over the entire grid iteratively so that the low potential value at the goal spreads out over the entire grid. The potential values at each node will converge to satisfy the discrete form of Laplace's equation. Grid nodes that represent geometric boundaries of any obstacles are fixed to the highest potential value. Therefore, these "obstacle nodes" will define areas or volumes of high potential, and will affect the final shape of the potential field. Additionally, the external nodes of the grid are also fixed to the highest potential value such that the walls of the control volume act as high potential boundaries. The result is a discrete

harmonic potential field, where the obstacles and control volume boundaries are represented as areas of high potential and the goal node is the global minimum.

Relaxation can be computationally expensive, especially for large, three-dimensional fields with tightly packed nodes. Therefore, the size of the control volume and the granularity of the grid must be carefully chosen such that the method remains computationally feasible.

Another consideration for this method is the numerical precision of the grid. Even for a fully converged numerical solution, potential values far from the goal node rapidly approach the highest potential value. Therefore, to avoid numerical error, the use of double precision numbers is required to preserve the small differences between adjacent nodes that are far from the goal. Large numbers of iterations using relaxation may be required for the discrete potential field values to converge with low numerical error.

3.1.2 Trilinear Interpolation

The major disadvantage of using discrete potential fields is that potential values and gradients are only defined at grid nodes. For free-flyer G&C, motion through the configuration space is smooth and continuous. To use discrete potential fields in a continuous manner, trilinear interpolation can be used to estimate the potential value at points in-between grid nodes.

Every point within the control volume at coordinates xyz will be contained in a grid cell defined by the lowest indices of the nearby grid nodes, ijk. Equation (3.3) defines the potential value at point xyz in grid cell ijk.

$$
\begin{aligned}
\phi(x,y,z)_{i,j,k} = \phi_{i,j,k} &\cdot (i+1-x)(j+1-y)(k+1-z) \\
&+ \phi_{i+1,j,k} \cdot (x-i)(j+1-y)(k+1-z) \\
&+ \phi_{i,j+1,k} \cdot (i+1-x)(y-j)(k+1-z) \\
&+ \phi_{i,j,k+1} \cdot (i+1-x)(j+1-y)(z-k) \\
&+ \phi_{i+1,j+1,k} \cdot (x-i)(y-j)(k+1-z) \\
&+ \phi_{i+1,j,k+1} \cdot (x-i)(j+1-y)(z-k) \\
&+ \phi_{i,j+1,k+1} \cdot (i+1-x)(y-j)(z-k) \\
&+ \phi_{i+1,j+1,k+1} \cdot (x-i)(y-j)(z-k)
\end{aligned}
\tag{3.3}
$$

The value of the potential will vary linearly between the values of the 8 nodes that define each cell, providing a close approximation of a continuous potential field over the same space. The above equations represent the case where the distance between any two nodes is unity ($h = 1$).

3.1.3 Potential Gradient

Path-finding methods based on potential fields rely on the calculation of the local gradient. Just like the scalar potential, the gradient vector needs to be defined continuously over the configuration space. To achieve this, partial derivatives of the above trilinear interpolation method can be used to estimate the potential gradient at any point *xyz*, within cell *ijk*. These partial derivatives are shown in equations (3.4), (3.5), and (3.6). Equation (3.7) defines the final gradient vector.

$$\begin{aligned}\frac{\partial\phi}{\partial x}(x,y,z)_{i,j,k} = &- \phi_{i,j,k} \cdot (j+1-y)(k+1-z) \\ &+ \phi_{i+1,j,k} \cdot (j+1-y)(k+1-z) \\ &- \phi_{i,j+1,k} \cdot (y-j)(k+1-z) \\ &- \phi_{i,j,k+1} \cdot (j+1-y)(z-k) \\ &+ \phi_{i+1,j+1,k} \cdot (y-j)(k+1-z) \\ &+ \phi_{i+1,j,k+1} \cdot (j+1-y)(z-k) \\ &- \phi_{i,j+1,k+1} \cdot (y-j)(z-k) \\ &+ \phi_{i+1,j+1,k+1} \cdot (y-j)(z-k)\end{aligned} \tag{3.4}$$

$$\begin{aligned}\frac{\partial\phi}{\partial y}(x,y,z)_{i,j,k} = &- \phi_{i,j,k} \cdot (i+1-x)(k+1-z) \\ &- \phi_{i+1,j,k} \cdot (x-i)(k+1-z) \\ &+ \phi_{i,j+1,k} \cdot (i+1-x)(k+1-z) \\ &- \phi_{i,j,k+1} \cdot (i+1-x)(z-k) \\ &+ \phi_{i+1,j+1,k} \cdot (x-i)(k+1-z) \\ &- \phi_{i+1,j,k+1} \cdot (x-i)(z-k) \\ &+ \phi_{i,j+1,k+1} \cdot (i+1-x)(z-k) \\ &+ \phi_{i+1,j+1,k+1} \cdot (x-i)(z-k)\end{aligned} \tag{3.5}$$

$$
\begin{aligned}
\frac{\partial \phi}{\partial z}(x,y,z)_{i,j,k} = &- \phi_{i,j,k} \cdot (i+1-x)(j+1-y) \\
&- \phi_{i+1,j,k} \cdot (x-i)(j+1-y) \\
&- \phi_{i,j+1,k} \cdot (i+1-x)(y-j) \\
&+ \phi_{i,j,k+1} \cdot (i+1-x)(j+1-y) \\
&- \phi_{i+1,j+1,k} \cdot (x-i)(y-j) \\
&+ \phi_{i+1,j,k+1} \cdot (x-i)(j+1-y) \\
&+ \phi_{i,j+1,k+1} \cdot (i+1-x)(y-j) \\
&+ \phi_{i+1,j+1,k+1} \cdot (x-i)(y-j)
\end{aligned}
\tag{3.6}
$$

$$
\nabla \phi(x,y,z) = \begin{bmatrix} \frac{\partial \phi}{\partial x} & \frac{\partial \phi}{\partial y} & \frac{\partial \phi}{\partial z} \end{bmatrix}^T \tag{3.7}
$$

3.2 Field-Based Reactive Guidance

The previous section gives a straightforward method of generating artificial potential fields that relate the obstacle environment to the goal position. A free-flyer could use this information in a variety of ways during maneuvering.

In general, the most direct way to use the potential field is to follow the path of steepest descent. To do this, infinitesimally small steps are taken in the direction of the gradient until the minimum is reached such that the resulting path is a smooth curve that bends around obstacles while approaching the goal point. Analogously, in incompressible fluid flow, steepest descent curves are shown as streamlines, where fluid elements descend the flow potential while bending around obstructions.

In this application, a free-flyer could generate a trajectory connecting the current and goal state in a similar manner. While this would be a valid trajectory, the free-flyer's control system would need to use its RCS jets to cancel the natural relative dynamics and follow the desired smooth, continuous curve. This is generally not an efficient way to maneuver during proximity operations where fuel is a limited resource and excessive jet firings risk undue plume impingement.

3.2.1 Gradient Implusive Maneuvering

Instead of using the path of steepest descent, gradient impulsive maneuvering (GIM), as introduced in [1], is applied here as a real-time control law. In the simplest version of GIM, the vehicle tracks the value of the potential at its current position. When the potential value begins to increase, a velocity impulse $\Delta \boldsymbol{v}$ is generated that redirects the vehicle along the gradient of steepest descent, as given by equation (3.8). The parameter c is chosen to scale the magnitude of the resulting velocity.

$$\Delta \boldsymbol{v} = -c\frac{\nabla \boldsymbol{\phi}}{|\nabla \boldsymbol{\phi}|} - \boldsymbol{v} \tag{3.8}$$

This method provides a way to guarantee the descent of the potential field, while allowing the natural dynamics to take over when the vehicle is approaching the goal. Figure 3.1 compares a path-finding solution generated using GIM with the path of steepest descent. This field was generated and interpolated using the processes described in the previous section.

The translational $\Delta \boldsymbol{v}$ requests that GIM produces are assumed to be applied impulsively. As long as this assumption is true, the method is guaranteed to converge to the goal from any valid starting point.

In a real free-flyer, the RCS jets are non-impulsive and imperfect. The use of a real propulsion system will therefore produce a kinematic response that deviates from the ideal impulse requested by GIM. Since the potential field provides a path to the goal from any starting point in free-space, the method should handle potential deviations and uncertainty robustly.

3.2.2 Gradient-Velocity Control Law

Using GIM as described so far, $\Delta \boldsymbol{v}$ impulse requests are generated on an as-needed basis when the potential value begins to increase, as given by the condition (3.9).

$$\phi_k > \phi_{k-1} \tag{3.9}$$

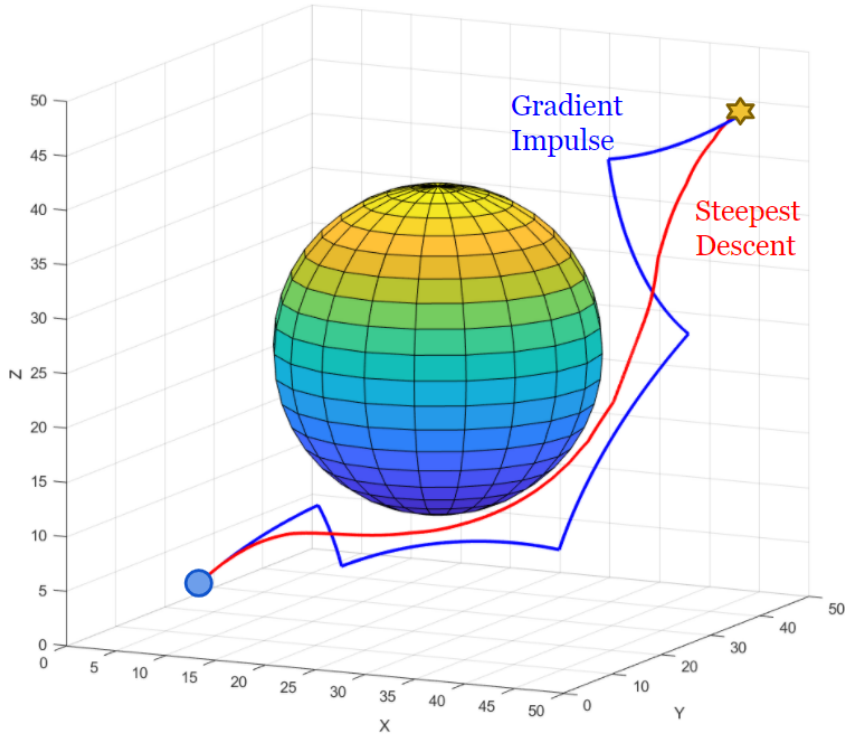

Figure 3.1: Gradient impulse maneuvering vs. path of steepest descent.

By strictly adhering to this condition, the vehicle can drift significantly from the most direct path to the goal given by steepest descent. In some cases, this results in near collisions with the structure, which is undesirable given certain safety requirements or plume impingement considerations.

This condition can be replaced by a stricter one that limits the maximum angle β between the steepest gradient direction and the free-flyer's velocity vector (3.10).

$$\frac{\boldsymbol{v}}{|\boldsymbol{v}|} \cdot \frac{\nabla \phi}{|\nabla \phi|} < \cos\left(\beta_{max}\right) \tag{3.10}$$

β_{max} therefore limits the extent of drift from the path of steepest descent. If $\beta_{max} = 90°$, then this condition becomes equivalent to the original GIM condition (3.9). If $\beta_{max} = 0$, then the condition stipulates that the velocity always be aligned with the gradient direction, which results in the vehicle following the path of steepest descent. In between these extremes, the choice of this parameter involves a trade-off between shorter trajectories with wide clearance of obstacles

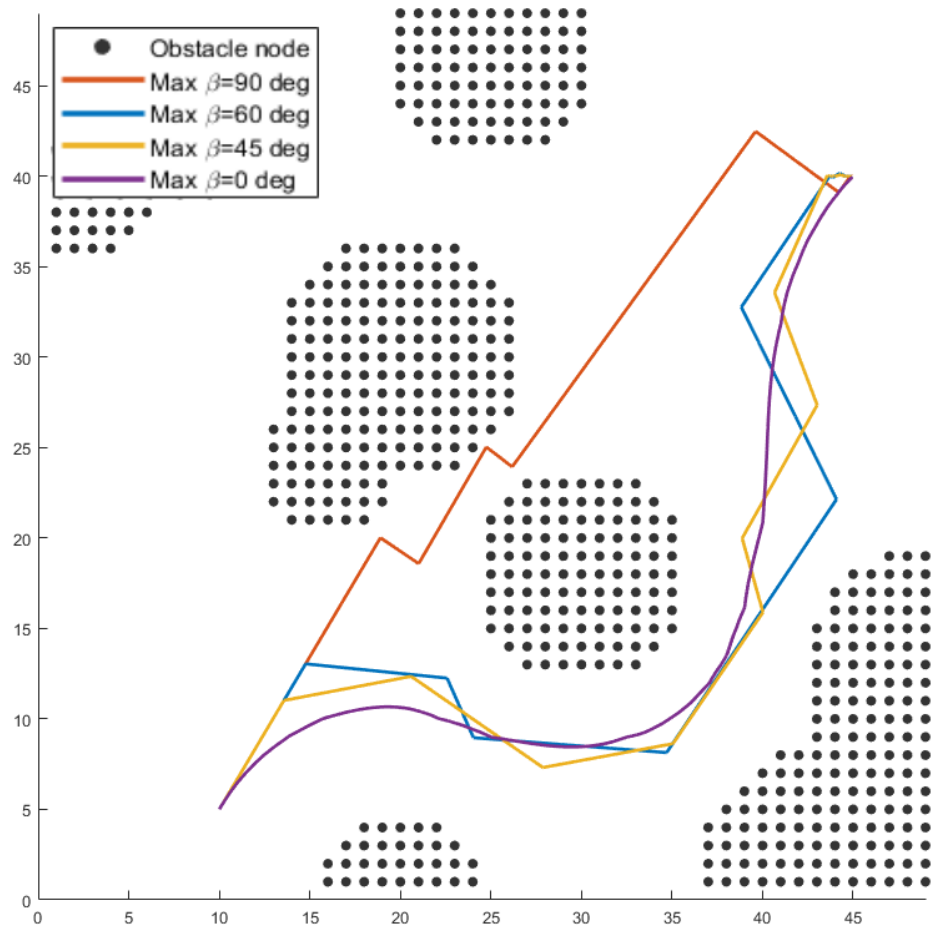

Figure 3.2: 2D example of how the choice of β_{max} affects the resulting path

and longer trajectories that more efficiently take advantage of the natural dynamics. Figure 3.2 shows a 2D example how the choice of β_{max} affects the resulting trajectory. The best choice of this parameter depends upon the specific needs of a particular free-flyer mission concept.

3.3 Potential Field Management

Once the potential field is generated, the information it encodes about the environment and the goal position is static. To allow for guidance that can react to new information about nearby obstacles, the potential field will need to be either augmented or regenerated. Methods for augmenting a potential field exist, but can introduce local minima which defeats the purpose of using harmonic potential fields. In this case, an approach for regenerating the potential field periodically

was developed.

The process to generate the potential field relies on two distinct pieces of information:

- Information about nearby structures and other obstacles, both static and dynamic, which are represented as obstacle nodes in the field
- A commanded goal position, which is used to designate a grid node as the goal node

If either piece of information changes, the resulting potential field becomes outdated, and will require the regeneration of the field values.

3.3.1 Internal Obstacle Model

To facilitate potential field generation, a system for managing an internal model of obstacles was implemented in the guidance process. So far, it has been assumed that the free-flyer has perfect knowledge of its own position and the position and geometry of all surrounding obstacles. In practice, this will not generally be true unless this information is derived externally and explicitly given to the free-flyer before or during flight. When this information is not available, a free-flyer must instead detect the surrounding environment utilizing its on-board sensors for both 3D obstacle mapping and localization. The specifics of these topics are beyond the scope of this thesis, but are explored in the navigation of obstacles for both quad-rotors and space-based robots in [25].

A free-flyer's internal model of the environment, which is used to generate artificial potential fields as previously described, is called the internal obstacle model. The internal obstacle model is an occupancy grid with the same dimensions as the discrete potential grid. In this occupancy grid, each node in the discrete grid is associated with a binary value that denotes if the grid is free or occupied by an obstacle. This internal model is updated as part of the guidance process, responding to new visibility information from the free-flyer's line-of-sight.

As visibility information is received, grid nodes within close proximity to detected obstacle geometry are designated as obstacle nodes. Obstacle nodes that are visible and were once occupied by detected geometry are removed from the model if it is determined that they are no longer occupied. In this manner, obstacle nodes are added and removed from the internal model. This

allows the guidance process to remember previously mapped obstacle data that may be obstructed but also to prune obstacle data that is no longer accurate due to the potential movement of dynamic obstacles. Details on the processes used to simulate this part of the guidance process are given in Section 5.3.

3.3.2 Evasion Mode

The choice of the speed parameter c in equation (3.8) and the maximum angle β_{max} in equation (3.10) impact the free-flyer's ability to avoid dynamic obstacles. For a mostly static environment, it is likely more desirable for the free-flyer to move at lower speeds and have a larger value of β_{max}, since this approach is generally more fuel efficient. This, however, is a bad combination of behavior for avoiding dynamic obstacles, where higher speed and sensitivity to the local gradient is necessary.

It is therefore useful to implement a logic for selecting the appropriate values of these parameters depending on the situation. Such logic was implemented into the guidance process by defining two separate modes, *guidance mode* and *evasion mode*. Guidance mode is simply the default for general-purpose free-flyer maneuvering, and uses a lower speed c and a higher β_{max} angle. When a dynamic obstacle nears the free-flyer, evasion mode is activated increasing the speed c and decreasing β_{max} to bolster collision avoidance behavior.

The specific condition for switching to evasion mode is given in equation (3.11). The free-flyer can compute the relative position $\boldsymbol{r}_{node_i}$ of obstacle nodes in its internal model, and can therefore define a minimum *keep-out-sphere* radius R_{KOS} under which evasion mode is activated.

$$\min |\boldsymbol{r}_{node_i}| < R_{KOS} \tag{3.11}$$

3.4 Final Guidance Process

To summarize the developments of this chapter, the free-flyer reactive guidance solution has three primary functions:

1. Generate $\Delta \boldsymbol{v}$ impulse requests conditionally based upon the instantaneous velocity and the

direction of the local field gradient

2. Continuously receive external visibility information and update the internal obstacle model accordingly

3. Periodically regenerate the potential field based upon the updated obstacle model and position command

An overview of the complete reactive guidance process, as implemented in the broader G&C system, is shown in Figure 3.3

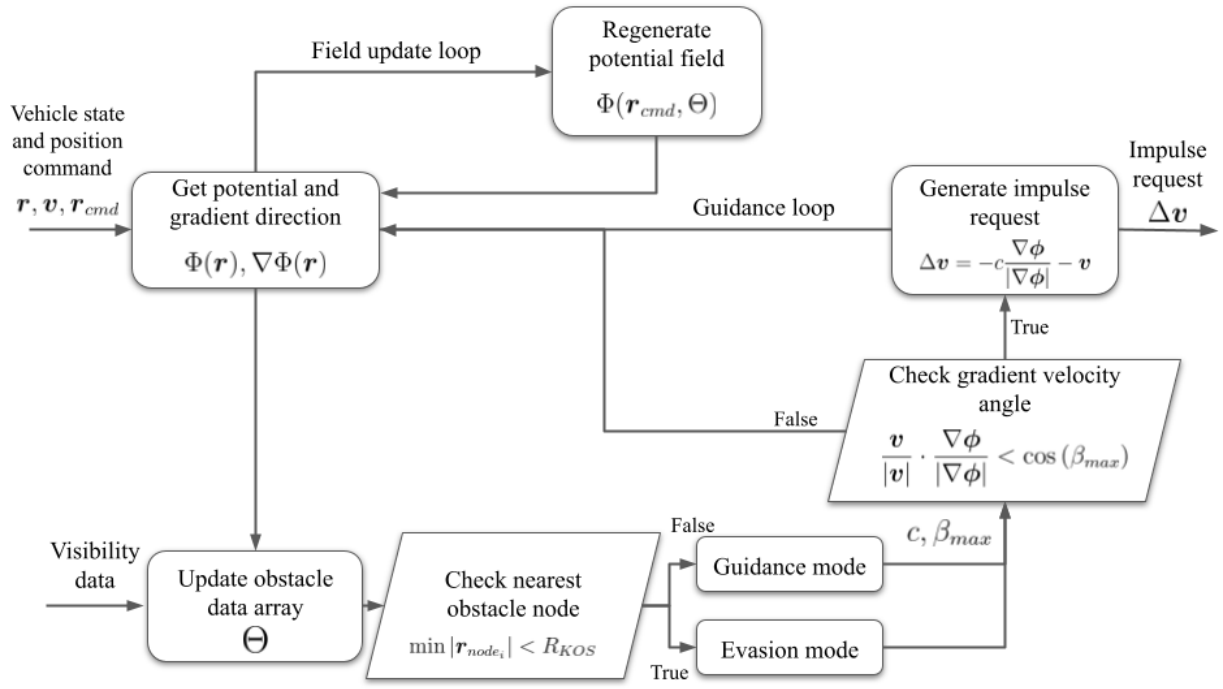

Figure 3.3: Reactive guidance process logical flow

4. INTEGRATED GUIDANCE AND CONTROL SYSTEM

The guidance method described in Chapter 3 enables a vehicle to safely maneuver to a goal position in the presence of nearby obstacles. This method produces ideal $\Delta \boldsymbol{v}$ impulse commands to apply to the vehicle. Of course, any real propulsion system is non-impulsive, and so these $\Delta \boldsymbol{v}$ commands must be converted into finite-time jet firing commands. Also, on its own the guidance method does not account for rotational maneuvering in any way.

To explore the efficacy of the guidance method developed in Chapter 3, it needed to be incorporated into a broader G&C system, which is the topic of this chapter. This system handles the fulfillment of both $\Delta \boldsymbol{v}$ and $\Delta \boldsymbol{\omega}$ requests through jet selection and performs additional functions such as attitude control and plume impingement avoidance.

4.1 Attitude Controller

A free-flyer's rotational degrees of freedom are not addressed by the reactive guidance method alone. To enable rotational maneuvers, an attitude controller based on the general phase-plane controller described in [32] was incorporated into the broader G&C system.

This attitude controller runs as a separate process from translational guidance and generates change in angular velocity, or $\Delta \boldsymbol{\omega}$ requests in response to angular error between the current and desired attitude. The controller has two distinct modes; slew mode and limit-cycle mode. Slew mode is used when the free-flyer is far from the desired attitude, and a high rate of convergence is desired. Limit-cycle mode is applied when the angular error is within the allowable deadband, where much smaller corrections are applied to maintain the desired attitude.

4.1.1 Attitude Error as Axis-Angle Parameters

Given the current and desired vehicle attitude expressed as unit quaternions $\boldsymbol{q}$ and $\boldsymbol{q}_d$, respectively, the error quaternion is given by equation (4.1).

$$\boldsymbol{q}_\epsilon = \boldsymbol{q}^c \circ \boldsymbol{q}_d \tag{4.1}$$

$\boldsymbol{q}_\epsilon$ describes the transformation from the current vehicle body frame, to the desired vehicle body frame orientation. To express the attitude error in a more intuitive form, the components of $\boldsymbol{q}_\epsilon$ can be converted into an axis-angle representation.

$$\boldsymbol{q} = \begin{bmatrix} s \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \boldsymbol{e}\sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{4.2}$$

Using the relation given by equation (4.2), the angular error (4.4) and error axis (4.5) can be derived from the error quaternion (4.3). These error parameters describe the single axis rotation needed to transform the current vehicle body frame to the desired vehicle body frame.

$$\boldsymbol{q}_\epsilon = \left[s_\epsilon, \boldsymbol{v}_\epsilon\right]^T \tag{4.3}$$

$$\theta_\epsilon = 2\arccos\left(s_\epsilon\right) \tag{4.4}$$

$$\boldsymbol{e}_\epsilon = \frac{\boldsymbol{v}_\epsilon}{\sin\frac{\theta_\epsilon}{2}} \tag{4.5}$$

To perform these calculations, care must be taken to handle the singularities that exist when $\theta_\epsilon = 0$ or $\theta_\epsilon = \pi$, resulting in a division by zero. As θ_ϵ tends to zero, $\boldsymbol{e}_\epsilon$ becomes undefined and can be set to any default value. In this particular application, this occurs when the vehicle perfectly achieves the desired attitude. In the control law definition that follows, the error parameters are only used when the vehicle is outside the allowable angular error deadband, and therefore no issues with singularities are encountered.

4.1.2 Axis-Angle Deadband Control Law

The attitude controller developed in this section is based upon the generalized control law described by [32]. Given a general state $\boldsymbol{x}$, a desired state $\boldsymbol{x_d}$, and their rates, the general control law is given by Equation (4.6).

$$\Delta\dot{\boldsymbol{x}} = c\,\mathrm{unit}(\boldsymbol{x_d} - \boldsymbol{x}) + (\dot{\boldsymbol{x_d}} - \dot{\boldsymbol{x}}) \tag{4.6}$$

In this control law, $\boldsymbol{x_d} - \boldsymbol{x}$ is the error between the current and desired state. The output of this control law is the rate change $\Delta\dot{\boldsymbol{x}}$ that will reduce the state and rate error simultaneously. The parameter c is the speed of convergence picked by the control system designer.

For simple attitude control, the control law output is a change in angular velocity, or $\Delta\boldsymbol{\omega}$, expressed in the vehicle body frame. Using the axis-angle error representation, a new form of the control law is described by Equation (4.7).

$$\Delta\boldsymbol{\omega} = c\boldsymbol{e}_\epsilon + (\boldsymbol{\omega}_d - \boldsymbol{\omega}) \tag{4.7}$$

To reach and hold the desired orientation, the desired angular velocity is set to zero: $\boldsymbol{\omega}_d = 0$. The control law then becomes Equation (4.8).

$$\Delta\boldsymbol{\omega} = c\boldsymbol{e}_\epsilon - \boldsymbol{\omega} \tag{4.8}$$

Therefore, the impulse $\Delta\boldsymbol{\omega}$ describes the change in angular velocity needed to cancel out the current angular velocity $\boldsymbol{\omega}$ and direct the new angular velocity along the error axis $\boldsymbol{e}_\epsilon$. This control law is applied when the angular error θ_ϵ exceeds an angular deadband threshold θ_{db}. The resulting impulse command drives the vehicle towards the desired orientation.

The angular speed parameter c is chosen based on the desired speed of convergence. When the vehicle is very far from the desired orientation, a higher speed of convergence is ideal. Alternatively, when the vehicle is very near the desired orientation, it is ideal that the vehicle limit-cycle with minimal speed within the deadband threshold.

To enable the switching between convergence speeds, another deadband threshold was defined as $\theta_b = \theta_{db} - b$, which is another concept derived from the general control law introduced by [32]. The control law (4.8) is applied when θ_ϵ exceeds the inner deadband threshold. The angular speed c is then chosen based on whether θ_ϵ exceeds the outer deadband θ_{db}. The parameter b defines

the distance between the inner and outer deadband. Since attitude control in a real system is non-impulsive, b is chosen to provide an angular buffer for limit-cycle firings to be performed before the outer deadband threshold is reached.

4.1.3 Final Attitude Controller Logic

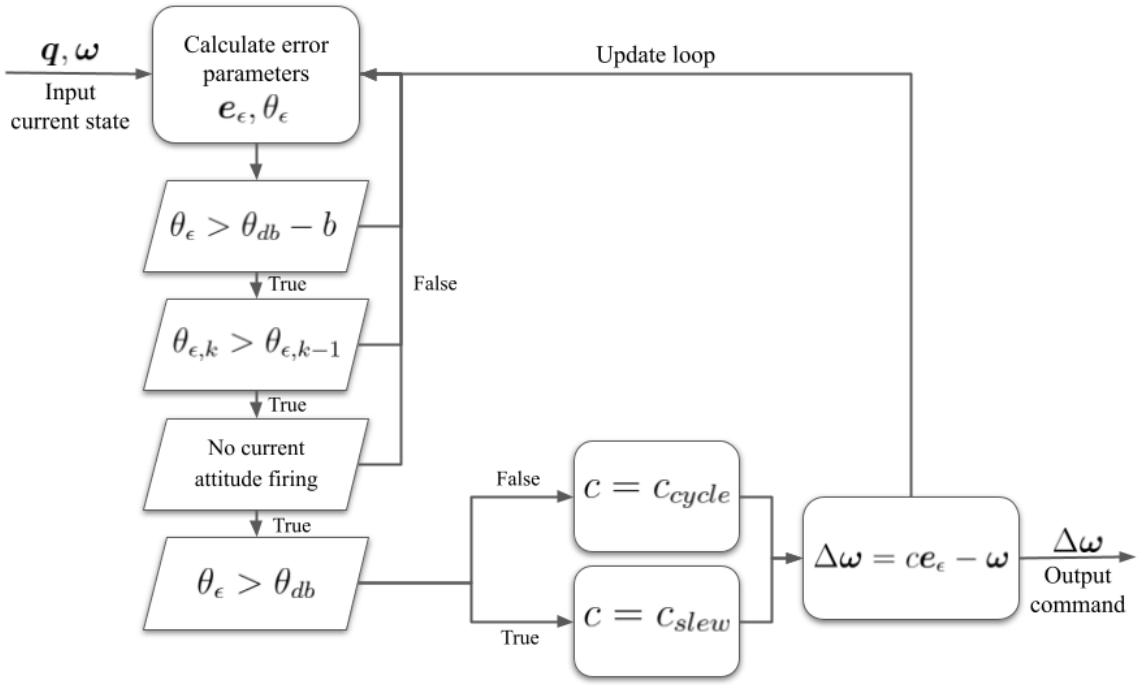

Figure 4.1: Attitude controller logic flow

The complete attitude controller process is shown in the diagram shown in Figure 4.1. Two additional conditions for generating a new $\Delta\boldsymbol{\omega}$ command were included in the final controller logic. After determining that the angular error has exceeded the deadband threshold, the controller also checks that the angular error is increasing in magnitude. The controller then also checks that a previously generated $\Delta\boldsymbol{\omega}$ command is not currently being applied. These two conditions prevent the attitude control from flooding the jet selection process with excessive $\Delta\boldsymbol{\omega}$ requests in the non-zero time it takes to apply control torque.

4.2 Jet Selection and Firing Manager

The reactive guidance method of Chapter 3 and the attitude controller outlined in this chapter produce $\Delta \boldsymbol{v}$ and $\Delta \boldsymbol{\omega}$ requests, respectively, on an as-needed basis. To process these requests, produce jet firing commands, and manage the execution of firing commands, a "jet manager" process was implemented into the free-flyer G&C system.

This jet manager utilizes a specific jet selection algorithm described in [33]. This algorithm takes a six-dimensional, combined $\Delta \boldsymbol{V}$ rate change vector (4.9) and produces an optimal firing command consisting of up to 6 jets.

$$\Delta \boldsymbol{V} = [\Delta \boldsymbol{v}, \Delta \boldsymbol{\omega}]^T \tag{4.9}$$

This algorithm was chosen over simpler jet selection schemes, such as minimum angle or dot product algorithms, because it can produce optimal solutions with a modifiable cost function. As shown later on, this functionality was used to produce plume-fuel optimal jet firings.

4.2.1 Jet Selection as a Linear Programming Problem

To solve for the optimal firing command, jet selection is formulated as an optimization problem solvable by linear programming.

For J jets, minimize

$$z = \sum_{j=1}^{J} c_j t_j \tag{4.10}$$

Subject to

$$\Delta \boldsymbol{V} = \sum_{j=1}^{J} t_j \boldsymbol{a}_j \tag{4.11}$$

$$t_j \geq 0 \ \ \forall j \tag{4.12}$$

The cost function is defined in equation (4.10). For each jet j, t_j is the resulting firing time and c_j is an arbitrary cost weighting. If the cost weightings of all the jets in a given configuration are equal, the minimization of this cost function produces a fuel-optimal solution. As explored in later sections, the jet cost weighting can be utilized to include other constraints, such as plume impingement, into the optimization.

The six-dimensional acceleration vector $\boldsymbol{a}_j$ is the translational and rotational acceleration resulting from the firing of a particular jet. The acceleration vector for jet j is defined in equation (4.13).

$$\boldsymbol{a}_j = \begin{bmatrix} \boldsymbol{F}_j/m \\ I^{-1}(\boldsymbol{r}_j \times \boldsymbol{F}_j) \end{bmatrix} \tag{4.13}$$

In this equation, $\boldsymbol{F}_j$ is the thrust vector associated with jet j in the body frame position defined by $\boldsymbol{r}_j$. The constraints (4.11) and (4.12) enforce that the resulting jet firing commands must perfectly satisfy the $\Delta \boldsymbol{V}$ request with non-negative firing times.

Simplex method, as outlined in [33], can be used to solve this optimization problem. Simplex method is an iterative algorithm that can convert any basic feasible solution of a linear programming problem into another basic feasible solution of a lower cost. This algorithm is applied until the optimal basic feasible solution is found. It can be shown that an optimal basic feasible solution to this problem is indeed an optimal feasible solution [33].

4.2.2 Processing Input ΔV Commands

The reactive guidance and attitude controller processes generate their respective rate change requests on an as-needed basis. Since these processes are independent and run asynchronously, the jet manager must constantly check for new $\Delta \boldsymbol{v}$ or $\Delta \boldsymbol{\omega}$ requests.

When a new request is received, the jet manager immediately incorporates that request into a brand new firing command. If a previous firing command is still being executed, the residual $\Delta \boldsymbol{V}$ vector that has yet to be full-filled is calculated and added to the incoming request.

$$\Delta \boldsymbol{V} = \Delta \boldsymbol{V}_r + \Delta \boldsymbol{V}_{new} \tag{4.14}$$

The residual $\Delta \boldsymbol{V}$ is found by multiplying the acceleration vector of each jet by the positive difference between its firing time and the time elapsed since the command began execution. An outline of the jet manager process is shown in Figure 4.2.

$$\Delta \boldsymbol{V}_r = \sum_{j=1}^{J} \boldsymbol{a}_j \max(t_j - t_{cmd}, 0) \tag{4.15}$$

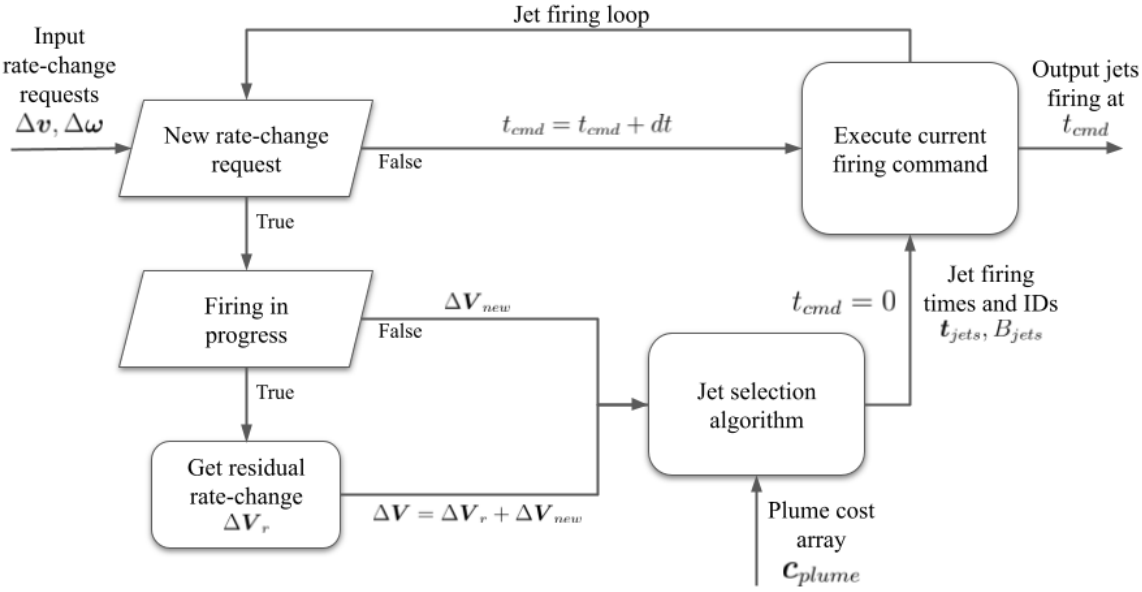

Figure 4.2: Jet selection and manager logic flow

4.2.3 Output Jet Firing Commands and Execution

The jet selection algorithm produces a six-dimensional vector, $\boldsymbol{t}_{jets}$, which represents the firing times for each jet in the basis set B_{jets} of optimal jets. For a particular input, if the optimal solution requires fewer than six jets, the remaining firing times will be equal to zero in $\boldsymbol{t}_{jets}$. The jet manager removes these zero time jets from the final firing command.

Realistic RCS propulsion systems are also limited by their minimum on-time. Therefore, in the formation of output firing commands, any jets with a firing time less than the minimum on-time in the jet selection output are removed.

When an output firing command is generated, the jet manager sends it to the firing execution loop. The command execution time, t_{cmd}, is initialized to zero and increased each loop by the time between updates. The value of t_{cmd} then determines which jets of a particular command are still firing, and the instantaneous list of firing jets is sent to the propulsion system. The update frequency of this loop dictates the temporal resolution of firing command execution.

4.3 Plume Impingement Avoidance

One of the key capabilities of the integrated G&C system is the incorporation of plume impingement avoidance functionality. To do this, plume impingement severity information is integrated directly into the jet selection process. This severity information can be approximated for each of the free-flyer's RCS jets using the position of each jet and available information about surrounding obstacles.

4.3.1 Plume-Fuel Optimal Jet Selection

The cost function given in equation (4.10) can be rewritten in vector form as (4.16).

$$z = \boldsymbol{c}^T \boldsymbol{t} \tag{4.16}$$

$\boldsymbol{c}$ is a vector of jet cost weightings and $\boldsymbol{t}$ is a vector of jet firing times, both of length J, the total number of jets. Given a jet configuration in which all jets have the same fuel consumption rate, a minimum firing time solution corresponds to a minimum fuel solution. The general cost weighting array $\boldsymbol{c}$ can be used to introduce other constraints on the optimization.

To incorporate plume avoidance into the jet selection optimization, the cost array $\boldsymbol{c}$ was used to weight the jets according to their potential for plume impingement with nearby structures and vehicles. The cost weighting is calculated using a plume cost function, which calculates the potential severity of plume impingement for each jet based on known information about nearby obstacles

and the jet's position and direction relative to those obstacles.

The result is an optimization process that produces plume-fuel optimal jet commands. Jets corresponding to relatively large plume costs will have their firing time reduced or eliminated in comparison to the fuel optimal solution. Instead, jet selection relies more heavily on alternative jet combinations to fulfill incoming impulse commands.

4.3.2 Plume Cost Function

In general, a plume cost function calculates the severity of plume impingement for a potential jet firing at the instantaneous relative position between a jet and nearby obstacles. The specific function applied by [23] was chosen for implementation (4.17).

$$c_j = \sum_{i=1}^{N} w_i F(\theta_i, \boldsymbol{t}_j, \boldsymbol{s}_{ij}) \tag{4.17}$$

In this equation $F(\theta_i, \boldsymbol{t}_j, \boldsymbol{s}_{ij})$ is a function that calculates the impingement of jet j on structural point i. w_i is a cost weighting of structural point i. Structural points are points within a structural component that are assumed to represent the structural geometry well. The total plume cost of jet j is the sum of its impingement cost on every structural point. This function is given by equation (4.18).

$$F = \frac{|\boldsymbol{t}_j|^2 \cos^4 \theta_i}{|\boldsymbol{s}_{ij}|^2} \tag{4.18}$$

In this plume function, $\boldsymbol{t}_j$ is the thrust vector for jet j, $\boldsymbol{s}_{ij}$ is the relative position of structural point i with respect to jet j, and θ_i is the off-axis angle of structural point i with respect to the thrust axis. This relative geometry is visualized in Figure 4.3.

This plume cost function models the approximate shape of jet flow field iso-pressure curves, as detailed in [23]. For a particular structural point and jet combination, the plume firing cost falls off quickly as the structural point moves away from the thrust axis due to the $\cos^4 \theta$ term. The cost also decreases quadratically as the relative distance between the jet nozzle and the structural point increases. Contours of the plume function F are shown in Figure 4.4.

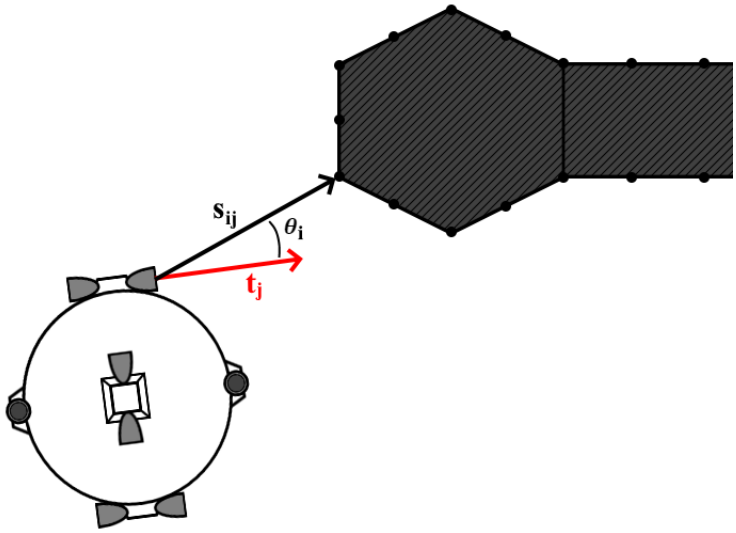

Figure 4.3: Geometry for plume cost function evaluation

4.3.3 Integration of Plume Cost Calculations

The plume cost function (4.17) quantifies the potential severity of plume impingement for each jet given the instantaneous relative position between the vehicle and all other obstacles. The relative position between the vehicle and obstacles will change as the free-flyer maneuvers towards the goal position. Additionally, the orientation of the free-flyer will change the relative direction of each jet with respect to any obstacles. Therefore, the jet cost array $\boldsymbol{c}$ will need to be updated to reflect accurate plume impingement information before each jet firing.

The computation of jet firing costs has the potential to be computationally expensive. For each jet, the plume cost function must be evaluated for every structural point. For integration into the broader G&C algorithm, the computation of jet firing costs can be carried out as an asynchronous process. Therefore, when the guidance logic requests a new ΔV, the jet selection computation can take the most recent jet firing costs to use in the optimization. This avoids the need to wait for a potentially expensive computation during firing command generation.

The structural points used in the plume cost function can come from any source. For free-flyer operations, the position and weightings of these points must be either given to the free-flyer from

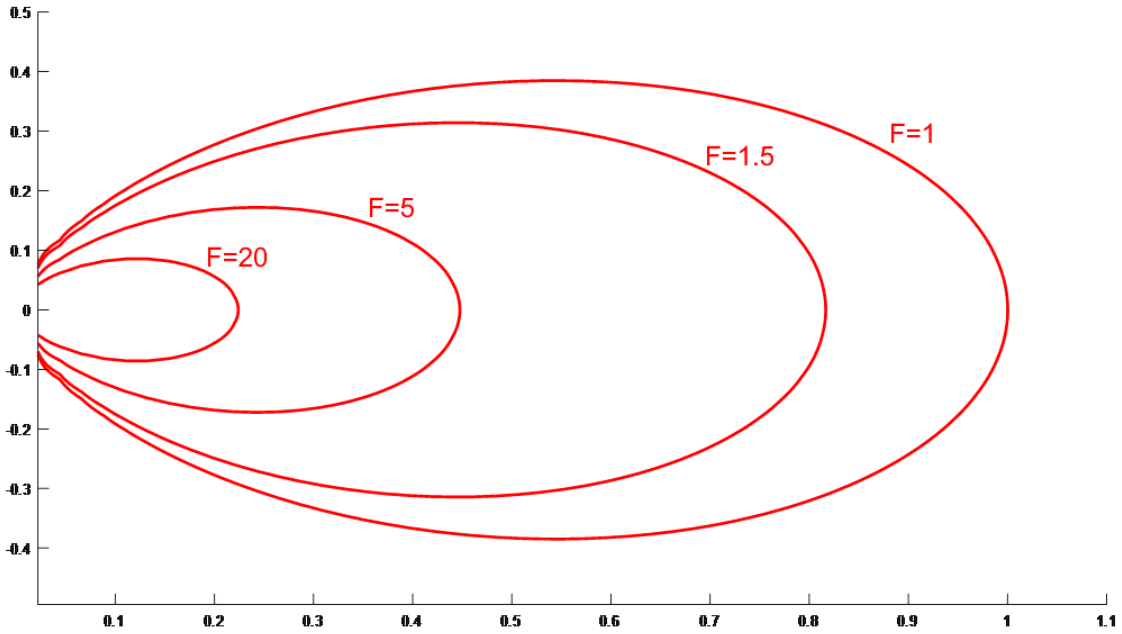

Figure 4.4: Plume cost function contours given a unit thrust vector

stored data, or detected during operation. In this G&C system, the guidance process is already responsible for receiving visibility information and constructing an internal model of nearby obstacles. For simplicity, the discrete plume process uses the available obstacle nodes from guidance to use as structural points. The cost weightings w_i of these structural points, which are used in equation (4.17), are all set to be equal in value. This is done because the guidance process is given no additional information about the sensitivity of structural components to plume impingement.

4.4 Integrated System

Together, this chapter and Chapter 3 develop the theory behind an integrated free-flyer G&C system that fulfills the key capabilities outlined in Chapter 2. The final G&C system consists of four processes running asynchronously. These four processes are referred to simply as Guidance, Attitude Controller, Jet Manager, and Plume Avoidance. The inputs and outputs of these processes are outlined in Table 4.1.

Process	**Input**	**Output**
Guidance	Vehicle translational state, position command, and object mapping information	$\Delta \boldsymbol{v}$ impulse commands and obstacle model
Attitude Controller	Vehicle rotational state and attitude command	$\Delta \boldsymbol{\omega}$ impulse commands
Plume Avoidance	Vehicle state and obstacle model	Plume cost array
Jet Manager	$\Delta \boldsymbol{v}$ and $\Delta \boldsymbol{\omega}$ impulses, plume cost array	Jet firing commands

Table 4.1: Integrated G&C system processes

5. FREE-FLYER TEST-BED SIMULATION

To test and verify the capabilities of the free-flyer G&C system described in Chapters 3 and 4, a real-time simulation of free-flyer proximity operations was designed and developed. This simulation was built on the SpaceCRAFT simulation platform [34], taking advantage of its modular, asynchronous architecture and the powerful visualization tools of the Unreal Engine 4 game engine [35]. Figure 5.1 shows an action screenshot taken during a real-time test-bed simulation run.

The test-bed simulation allowed for broad variability in the design of test cases and was designed to offer completely generalized modelling of the free-flyer problem. This includes configurable obstacle environments, free-flyer jet configurations, vehicle properties, and, of course, G&C system parameters. The test-bed simulation's key capabilities are outlined in Table 5.1.

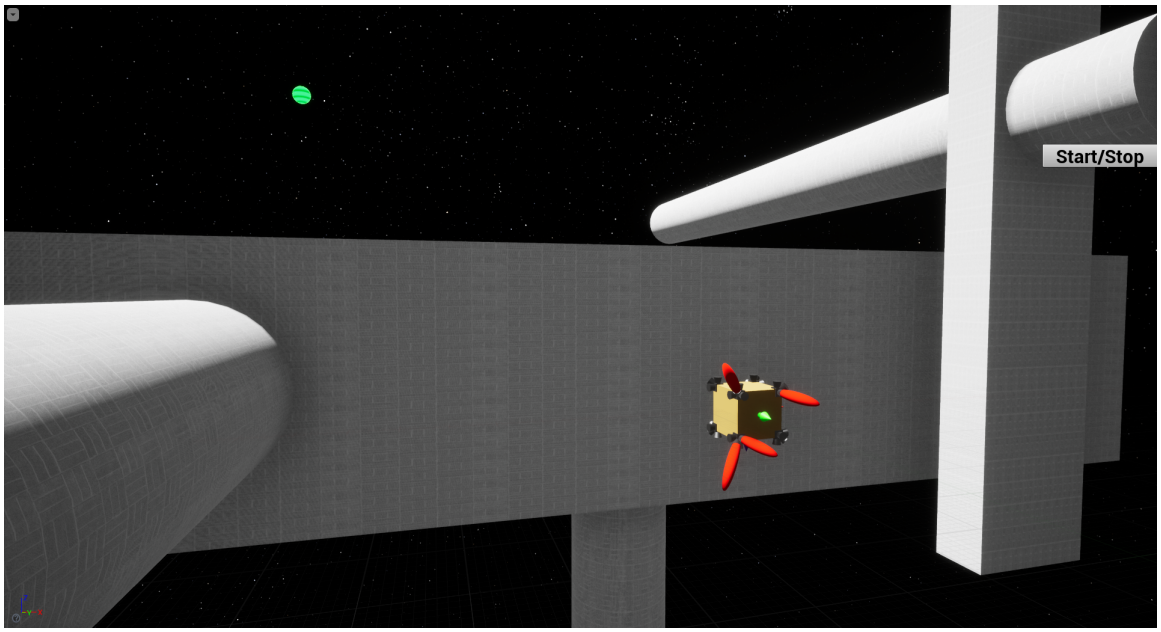

Figure 5.1: Screenshot taken during a free-flyer test-bed simulation run

Capability	Description
Physics Propagation	The relative translational and rotational vehicle dynamics are propagated numerically, with control force and torque inputs.
Asynchronous Processes	Individual processes run in parallel, which avoids undesirable effects of having control systems, physics, and other utility processes running on a single thread.
UE4 Integration	Integration with Unreal Engine 4 provides high-fidelity rendering of the simulation state in real-time, as well as access to other optimized functionality such as collision detection and raycasts.
Configurable Obstacle Environments	Various obstacle configurations and environments can be loaded into the simulation, and can include both static and dynamic obstacles.
Configurable Vehicle and Control Parameters	Vehicle design parameters, such as mass properties, jet configurations, and control system parameters are fully configurable for testing and design iteration.
Data Logging	The simulation performs data logging within each process for post-simulation analysis and plot generation.

Table 5.1: Key test-bed simulation capabilities

5.1 SpaceCRAFT/UE4 Simulation Software

The SpaceCRAFT simulation platform was designed and developed by the students at the ASTRO center at Texas A&M University under the guidance of Professor Gregory Chamitoff. The SpaceCRAFT project began development with the goal of providing a unique platform for real-time space mission simulation with strong VR support.

The SpaceCRAFT platform has matured considerably since its conception in 2016, and has developed an array of useful features and functionality for the development of sophisticated simulations. Simulation development with SpaceCRAFT is primarily done in C++, although alternative language APIs exist. SpaceCRAFT supplies a simulation development environment that is comparable to NASA's TRICK, but with a greater focus on real-time simulations and high-fidelity visualization. SpaceCRAFT's asynchronous architecture, ease of model implementation, and inte-

gration with UE4 motivated its selection for this work.

5.1.1 Components of a SpaceCRAFT Simulation

SpaceCRAFT simulations are built upon the interactions of a few fundamental components. The most basic SpaceCRAFT object is the *entity*, which can represent any object in the simulation. Entities have *parameters* attached to them, which can represent any data owned by an entity. To illustrate, a vehicle sensor can be represented in a simulation as its own entity, with parameters that describe its properties as well as any input or output data it owns.

The last fundamental component is the *system*, which implements any simulation process or model that acts on entities and their parameters. Systems can also have their own *instance parameters* which are available outside of any entity ownership. Systems are user defined C++ code models, which are supplied and built as external packages. Each system has an `init()` and `update()` function that is called by the main SpaceCRAFT executive. For each system, `init()` runs once at simulation start-up and `update()` runs at a user defined frequency. Systems run asynchronously and provide the interaction between entities and their parameters that create the overall simulation state. SpaceCRAFT performs the time-keeping, data-synchronization, and `update()` scheduling for the user.

To define a simulation, `.json` configuration files are used to specify entities and parameters, include system files, and attach systems to specific entities. Therefore, simulation development is reduced to the definition of the simulation configuration file and system programming.

5.1.2 Simulation Non-Determinism

As a result of SpaceCRAFT's asynchronous architecture, simulations are often non-deterministic. This is a common issue among software that utilizes some degree of parallel programming [36]. Each time a SpaceCRAFT simulation is run, the various system `update()` calls and other operations will occur in a different order due to small timing differences. This results in non-determinism when these operations touch data shared by other processes. The test-bed simulation described in the chapter has many different processes operating on shared data, and therefore is highly non-

deterministic. This does not reduce the validity of the individual simulation processes, but did affect the way that the simulation test-bed was used to generate results for this thesis.

5.1.3 UE4 Integration

Another important aspect of SpaceCRAFT is its streamlined integration with UE4. The SpaceCRAFT platform includes functionality to give UE4 objects and classes access to simulation entities and their parameters. This enables high-fidelity simulation rendering, as well as access to functionality that UE4 specializes in, such as user input and collision detection.

The architecture of the SpaceCRAFT platform includes both a client and server. The platform-specific UE4 client connects with the parameter server, which maintains a key-mapped list of all simulation entities and parameters. The server also manages the initialization and update scheduling for any systems during a particular simulation. Notably, the server can run simulations independently of the UE4 client, as long as those simulations do not depend on return data from the client.

5.2 Test-Bed Simulation Architecture

The architecture of the test-bed simulation software is outlined in Figure 5.2. This architecture consists of several discrete processes that read and write data to the SpaceCRAFT parameter server. These processes fall into a several categories, which are designated by color as shown in the diagram's legend.

On the left side of this diagram are all the SpaceCRAFT systems that drive the simulation. A subset of these systems, shown in blue, are the systems belonging to the free-flyer G&C algorithm. These four systems correspond directly to the processes described in Table 4.1.

There are three other systems included in the simulation besides the ones that implement the G&C algorithm. The *Data Monitor* system is a SpaceCRAFT utility, and provides real-time monitoring of server parameters. The other two systems, FF_Physics and FF_Structure, are responsible for modelling the environment of proximity operations for the free-flyer G&C system to interact with.

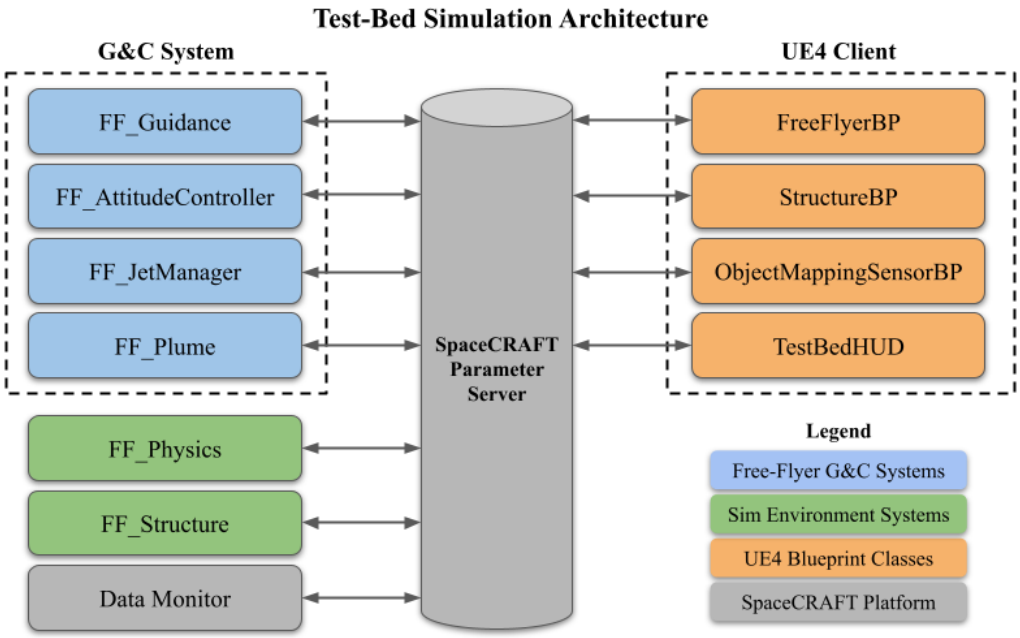

Figure 5.2: Test-bed simulation software architecture

On the right side of Figure 5.2 are all of the UE4 classes used to create a real-time visualization of the simulation. The classes StructureBP and FreeFlyerBP drive the rendering of static meshes used to represent the free-flyer vehicle and structure, respectively. These visual functions do not affect the state of the simulation. The ObjecctMappingSensorBP class drives the creation of simulated visbility data, which is detailed in Section 5.3.

5.2.1 Physics Propagation

One of the primary functions of the test-bed simulation is to model the dynamic response of the free-flyer due to the control output of the G&C algorithm. To accomplish this, the physics propagation system FF_Physics was implemented to handle vehicle state propagation and output.

The vehicle dynamics are modelled by the differential equations introduced in 2.1. For translational motion, the linearized form of the relative orbital motion equations are used. Rotational motion, however, is modelled by nonlinear equations of rigid body dynamics and quaternion kinematics. For simplicity, a basic Runge-Kutta 4th order numerical integration method is applied to

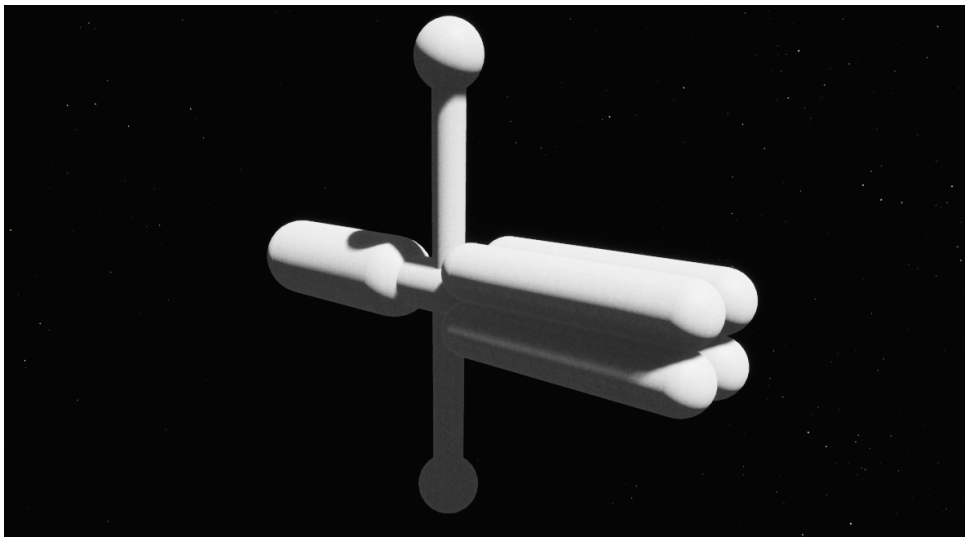

Figure 5.3: An example structure rendered in UE4

both translational and rotational equations [37]. The vehicle control accelerations are determined directly as the sum of the acceleration components of each jet that is currently firing. The current control force and torque vectors are output by FF_JetManager.

During each `update()`, FF_Physics propagates the vehicle state forward by a fixed timestep, computes any derived states and transformations, and outputs the updated state data to the SpaceCRAFT parameter server. This system also logs the state information for post-simulation analysis.

5.2.2 Obstacle Representation

In the test-bed simulation, the structural environment that the free-flyer attempts to navigate is user-defined. These environments are composed of three fundamental 3D shapes; spheres, boxes, and cylinders. By translating, rotating, and scaling these basic shapes, a variety of complex obstacles can be constructed. To define a structure or obstacle field, the user supplies a `.json` file which defines a *structural configuration* in terms of these basic components. An example of a complex structure composed of these simple components in the test-bed simulation is shown in Figure 5.3.

The structural configuration is initialized by the FF_Structure system and output to the parameter server. The components of a structure are static by default, but additional functionality exists

to specify the motion of dynamic obstacles. Each `update()`, FF_Structure propagates the state of any dynamic obstacles according to structure-specific motion models and outputs the new states to the parameter server.

5.3 Simulated Obstacle Mapping

The reactive guidance process handles updates to the free-flyer's inertial obstacle model and potential field data, as discussed in Section 3.3. The inclusion of this functionality was intended to demonstrate that the proposed guidance method is still effective when the free-flyer receives information about only obstacles within its line-of-sight.

The work needed to implement 3D mapping algorithms is extensive, and requires the virtual modelling of sensors such as lidar or Red-Green-Blue-Depth (RGBD) cameras. Such topics are outside the scope of this thesis. To simulate obstacle mapping without actually implementing mapping algorithms and camera models, an analogous method of generating the output of a 3D mapping algorithm was needed. This output is referred to as *visibility data*.

To generate this visibility data, UE4's *raycasts* were used. Raycasts, also referred to as line-traces, detect collisions with rendered objects along a line segment. Each raycast returns a boolean collision flag as well as additional collision data, such as distance from the raycast origin to the point of collision.

To generate visibility data, each node in the potential field grid is tested with a raycast that originates at the instantaneous position of the free-flyer. If a raycast collides with a rendered obstacle, the collision distance is compared to the expected distance between the vehicle and the node. If the collision distance is less than the expected distance, then the node is obstructed from the vehicle's line-of-sight. If the collision distance is roughly similar to the expected distance, then the node is detected. Finally, if the distance exceeds the true distance, or no collision occurs, then the node was undetected. Given this data, the guidance process adds detected nodes to the internal obstacle model. If a previously detected node is suddenly undetected, it is removed from the internal model. Otherwise, undetected or obstructed nodes do not affect the internal model.

By checking all grid points in this manner, the resulting visibility data is essentially analogous

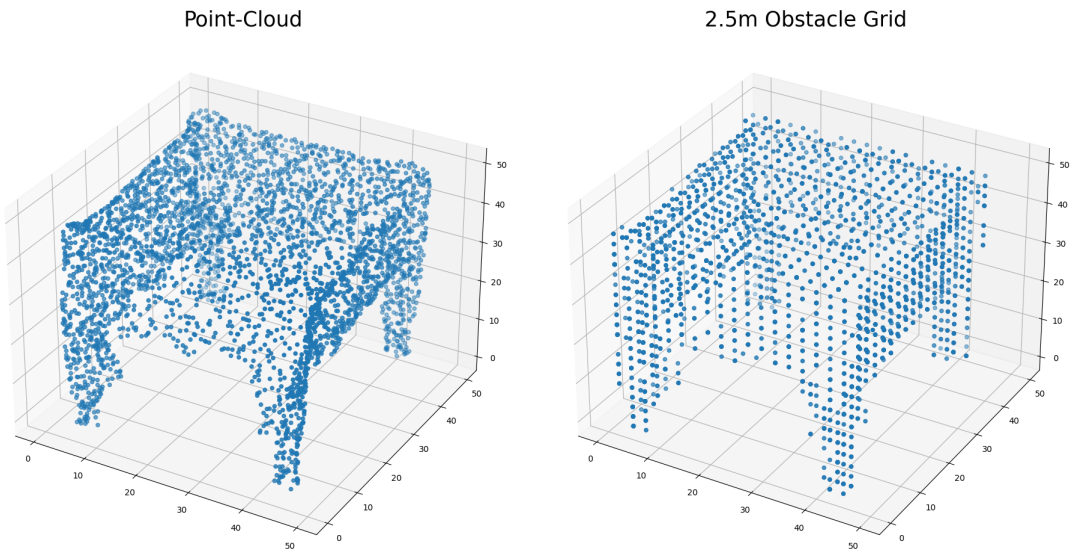

Figure 5.4: Point-cloud to obstacle grid conversion example using data obtained from a lidar sensor

to the output of a typical 3D mapping algorithm. For example, a 3D mapping algorithm could utilize lidar or RGBD camera data to detect points on the surface of nearby obstacles. The collection of these points is a point-cloud, which can be processed further into a triangulated surface mesh representing an obstacle. Although the point-cloud data does not naturally conform to the potential field grid used in the proposed guidance process, it is a simple calculation to determine obstacle nodes from point-cloud data as shown in Figure 5.4.

3D mapping algorithms are often able to refine the surface mesh over time given additional sensor data. If surface geometry is detected behind or inside the existing surface mesh, then mesh vertices can be moved or deleted. The function of such a mapping algorithm in practice is analogous to the simulated mapping that takes place in the test-bed simulation.

5.3.1 Optimizing Raycast Execution

The simulated obstacle mapping approach described so far requires every grid point to be tested with a raycast during a single detection pass. For a 50x50x50 grid resolution, this requires 125,000 raycast computations. This computational expense makes this approach infeasible as described, and resulted initially in very poor performance during real-time simulations. To fix this issue,

several optimizations were implemented to improve the performance for this functionality of the test-bed simulation.

First, a method was determined to reduce the overall number of grid points that need to be tested, without changing the resulting visibility data. From the structure configuration, the true set of existing obstacle nodes can be determined. Then the following is asserted: a raycast test to a grid point that has never been an obstacle node will result in no detection by definition. Therefore, only nodes that are currently or have previously been obstacle nodes are tested by raycast. This drastically cuts down on the total number of line traces that need to be performed for a typical structural configuration.

A second method for improving performance was also implemented. As developed previously, the guidance process updates the potential field values at a relatively low frequency. Due to this low frequency, the computational demand for a large set of raycasts can be spread out over several computational ticks, or updates. Instead, the raycast process can occur in batches of a fixed size. As long as the total number of raycasts can be accomplished between potential field update intervals, the resulting visibility data is not fundamentally affected.

The implementation of these optimizations has made this process of simulating object mapping data feasible for real-time simulation. A screenshot of this process in action is shown in Figure 5.5. It is important to emphasize that these performance challenges and the solutions discussed in this section do not have implications for the G&C system developed in this thesis. Simulation of obstacle mapping data, and the associated computational challenges are aspects of the test-bed simulation alone.

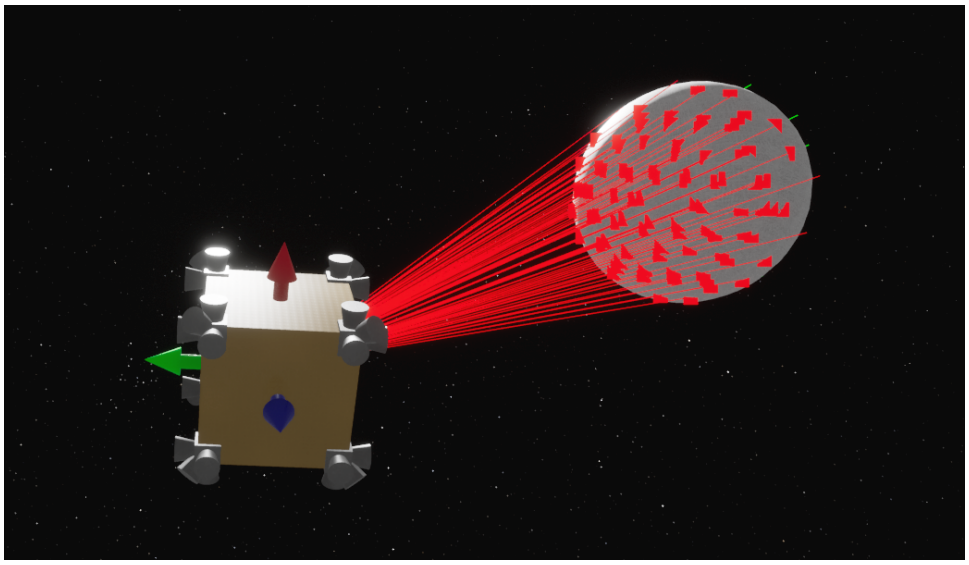

Figure 5.5: Screenshot of the obstacle mapping functionality with visualized raycasts

5.4 Reference Free-Flyer

The G&C system designed in this thesis is intended to provide effective maneuvering regardless of the design of a particular free-flyer. This is not without limitation, as it is assumed that the free-flyer design includes an RCS propulsion system with an array of jets, as opposed to a "turn-and-burn" design. It is also assumed that the free-flyer has relatively balanced dimensions and inertial properties. Within these limitations, a large variety of potential free-flyer designs are compatible. The G&C system is intended to be tuned to a specific free-flyer design by tweaking certain control parameters. Variations in both the vehicle design and control system parameters will affect overall system performance.

To test the G&C system and verify that it meets the desired key capabilities, the simulation results need to reflect the performance of the system implemented for a real free-flyer design that could be viable for the future mission concepts developed in Section 1.1.2. To standardize the approach to testing, a reference free-flyer design was developed for the test-bed simulation. The G&C parameters were chosen through iterative simulation testing to fit the reference free-flyer's design. The following sections outline the reference design and system parameters that are used to

generate the simulation results in the following chapter.

5.4.1 Design and Properties

The reference free-flyer design includes all of the properties and vehicle configuration information needed to implement the developed G&C system in the simulation environment. For maneuvering, the most important aspect of a free-flyer's design is the RCS propulsion system. This design utilizes an RCS jet array with 32 jets of equal thrust. The jet array given by Table 5.2 shows each jet's position with respect to the body frame, boresight direction, and thrust. The resulting jet configuration is visualized in Figure 5.6. Other important vehicle design properties are included in Table 5.3

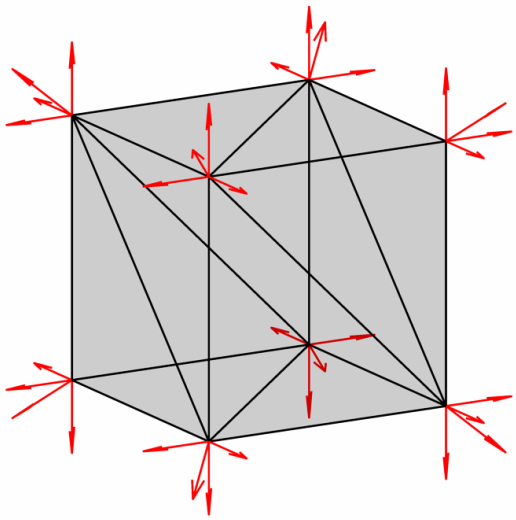

Figure 5.6: Thirty-two jet array used by the reference free-flyer

5.4.2 Default G&C System Parameters

The system parameters include all user-defined parameters in the integrated G&C system described in Chapter 4. The system parameters used to generate the simulation results are shown in Table 5.4.

Jet ID	Position (m)	Direction	Thrust (N)
1	0.5, 0.5, 0.5	1, 0, 0	12.5
2	0.5, 0.5, 0.5	0, 1, 0	12.5
3	0.5, 0.5, 0.5	0, 0, 1	12.5
4	-0.5, 0.5, 0.5	-1, 0, 0	12.5
5	-0.5, 0.5, 0.5	0, 1, 0	12.5
6	-0.5, 0.5, 0.5	0, 0, 1	12.5
7	0.5, -0.5, 0.5	1, 0, 0	12.5
8	0.5, -0.5, 0.5	0, -1, 0	12.5
9	0.5, -0.5, 0.5	0, 0, 1	12.5
10	0.5, 0.5, -0.5	1, 0, 0	12.5
11	0.5, 0.5, -0.5	0, 1, 0	12.5
12	0.5, 0.5, -0.5	0, 0, -1	12.5
13	-0.5, -0.5, 0.5	-1, 0, 0	12.5
14	-0.5, -0.5, 0.5	0, -1, 0	12.5
15	-0.5, -0.5, 0.5	0, 0, 1	12.5
16	-0.5, 0.5, -0.5	-1, 0, 0	12.5
17	-0.5, 0.5, -0.5	0, 1, 0	12.5
18	-0.5, 0.5, -0.5	0, 0, -1	12.5
19	0.5, -0.5, -0.5	1, 0, 0	12.5
20	0.5, -0.5, -0.5	0, -1, 0	12.5
21	0.5, -0.5, -0.5	0, 0, -1	12.5
22	-0.5, -0.5, -0.5	-1, 0, 0	12.5
23	-0.5, -0.5, -0.5	0, -1, 0	12.5
24	-0.5, -0.5, -0.5	0, 0, -1	12.5
25	0.5, 0.5, 0.5	0.577, 0.577, 0.577	12.5
26	-0.5, 0.5, 0.5	-0.577, 0.577, 0.577	12.5
27	0.5, -0.5, 0.5	0.577, -0.577, 0.577	12.5
28	0.5, 0.5, -0.5	0.577, 0.577, -0.577	12.5
29	-0.5, -0.5, 0.5	-0.577, -0.577, 0.577	12.5
30	-0.5, 0.5, -0.5	-0.577, 0.577, -0.577	12.5
31	0.5, -0.5, -0.5	0.577, -0.577, -0.577	12.5
32	-0.5, -0.5, -0.5	-0.577, -0.577, -0.577	12.5

Table 5.2: Position, boresight direction, and thrust for reference free-flyer RCS system

Property	Symbol	Value	Units
Vehicle mass	m	25	kg
Moments of inertia	$\boldsymbol{I}_{diag}$	25, 25, 25	$kg \cdot m^2$
Products of inertia	$\boldsymbol{I}_{cross}$	0, 0, 0	$kg \cdot m^2$
Minimum jet on-time	dt_{min}	10	ms
Vehicle side length	s	1	m
Orbit rate	ω_0	0.001131	rad/s

Table 5.3: Reference free-flyer properties

Property	Symbol	Value	units
Grid resolution parameter	n	50	–
Grid step size	h	1	m
Grid refinement iterations	N	500	–
Guidance speed	c_{guid}	0.5	$\frac{m}{s}$
Evasion speed	c_{evad}	1.5	$\frac{m}{s}$
Gradient-velocity guidance max angle	$\beta_{max_{guid}}$	40	deg
Gradient-velocity evasion max angle	$\beta_{max_{evad}}$	5	deg
Keep-out-sphere radius	R_{KOS}	2.5	m
Limit-cycle angular speed	c_{cycle}	0.5	$\frac{deg}{s}$
Slew-mode angular speed	c_{cycle}	5	$\frac{deg}{s}$
Angular deadband limit	db	5	deg
Angular deadband buffer	b	2	deg
Reactive guidance update freq.	f_{guid}	20	Hz
Potential field update freq.	f_{pf}	2	Hz
Attitude controller update freq.	f_{ac}	60	Hz
Jet Manager update freq.	f_{jm}	60	Hz
Plume avoidance update freq.	f_{pa}	20	Hz

Table 5.4: Integrated G&C system parameters used during simulation testing

The test-bed simulation outlined in Chapter 5 was used to evaluate the performance of the free-flyer G&C system developed in this thesis. The completed system combines several different algorithms and techniques to provide a practical and flexible approach to free-flyer autonomous maneuvering. To assess the performance of the integrated system, simulation runs intended to demonstrate the system behavior were conducted. Additionally, this testing aims to assess whether the system meets the key capabilities outlined in Table 1.1.

6.1 Point-to-Point Maneuvering Tests

The critical capability of the free-flyer G&C system developed in this thesis is to provide safe point-to-point maneuvering in the presence of both static and dynamic obstacles in an orbital environment. To test this capability a series of maneuvering test cases were developed. Each test features a different obstacle environment, which were defined using the structural configuration definition functionality of the test-bed simulation. The obstacle environments used are shown in Figure 6.1.

In each test, the free-flyer is given an initial state and commanded state to achieve. Each test ends when the free-flyer reaches the commanded position. The parameters for each maneuver test

Parameter	Test 1	Test 2	Test 3	Test 4
Environment	Knucklebones	Maze	Ship	Asteroids
$\boldsymbol{r}_0^{LVLH}\ [m]$	20.0, 15.0, 15.0	20.0, 15.0, 15.0	-10.0, -10.0, 0.0	20.0, 15.0, 15.0
$\boldsymbol{r}_{cmd}^{LVLH}\ [m]$	-15.0, -15.0, -15.0	-15.0, -15.0, -15.0	20.0, 10.0, 0.0	-15.0, -15.0, -15.0
$\boldsymbol{q}_{body_0}^{lvlh}$	0.0, 0.0, 1.0, 0.0	0.41, 0.41, 0.81, 0.03	0.0, 0.0, 0.0, 1.0	0.0, 0.0, 0.0, 1.0
$\boldsymbol{q}_{body_{cmd}}^{lvlh}$	0.0, 0.0, 0.0, 1.0	0.0, 0.0, 0.0, 1.0	0.0, 0.0, 1.0, 0.0	0.41, 0.41, 0.81, 0.03
$\boldsymbol{v}_0^{lvlh}\ [m/s]$	0.0, 0.0, 0.0	0.0, 0.0, 0.0	0.0, 0.0, 0.0	0.0, 0.0, 0.0
$\boldsymbol{\omega}_0^{body}\ [rad/s]$	0.0, 0.0, 0.0	0.0, 0.0, 0.0	0.4, -0.5, 0.3	0.0, 0.0, 0.0

Table 6.1: Point-to-point maneuvering test definitions

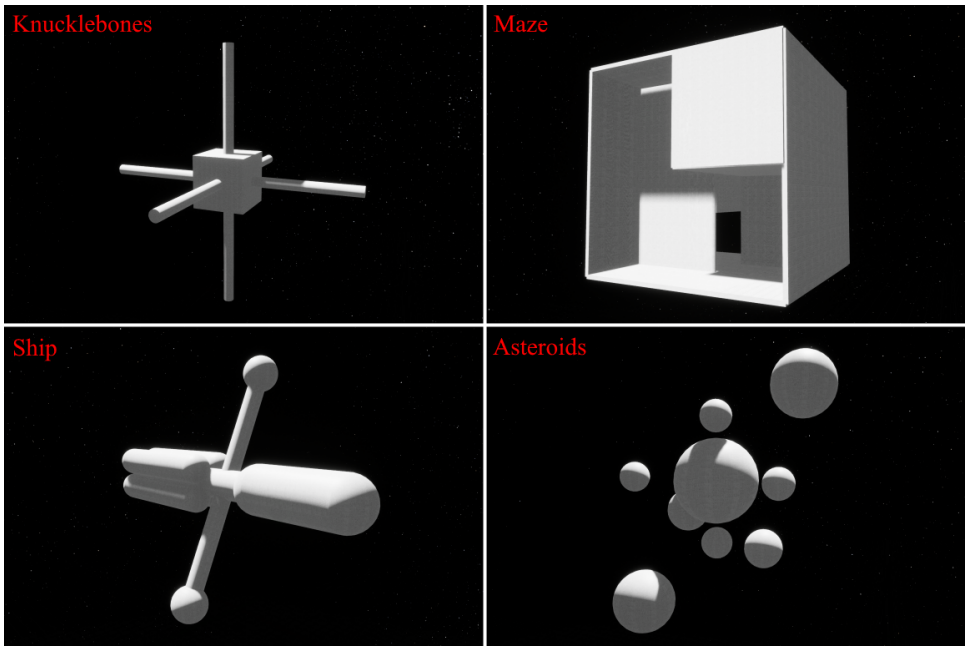

Figure 6.1: The four maneuver testing obstacle environments

are shown in Table 6.1. All tests include simultaneous translational and rotational maneuvers, to test the systems ability to perform both in parallel.

6.2 Maneuvering Test 1

The first maneuvering simulation tested the ability of the free-flyer to operate near a static, symmetrical orbital structure fixed in the local LVLH frame. The specified position command for this test required the free-flyer to move to a point on the opposite side of the structure. Additionally, the free-flyer's attitude command required a 180 degree rotation about its z-axis during this maneuver.

The resulting trajectory of the free-flyer from this test is shown in Figure 6.2. In this figure, the obstacle nodes that were detected by the free-flyer and used in guidance are shown in red. Any obstacle nodes that were undetected appear as white.

Figure 6.3 gives the plots of several guidance parameters. The plot in the upper left of this figure shows the minimum distance between the free-flyer and any obstacle node. The black dashed line shows the distance in which a collision would have occurred based on the vehicle radius, while the

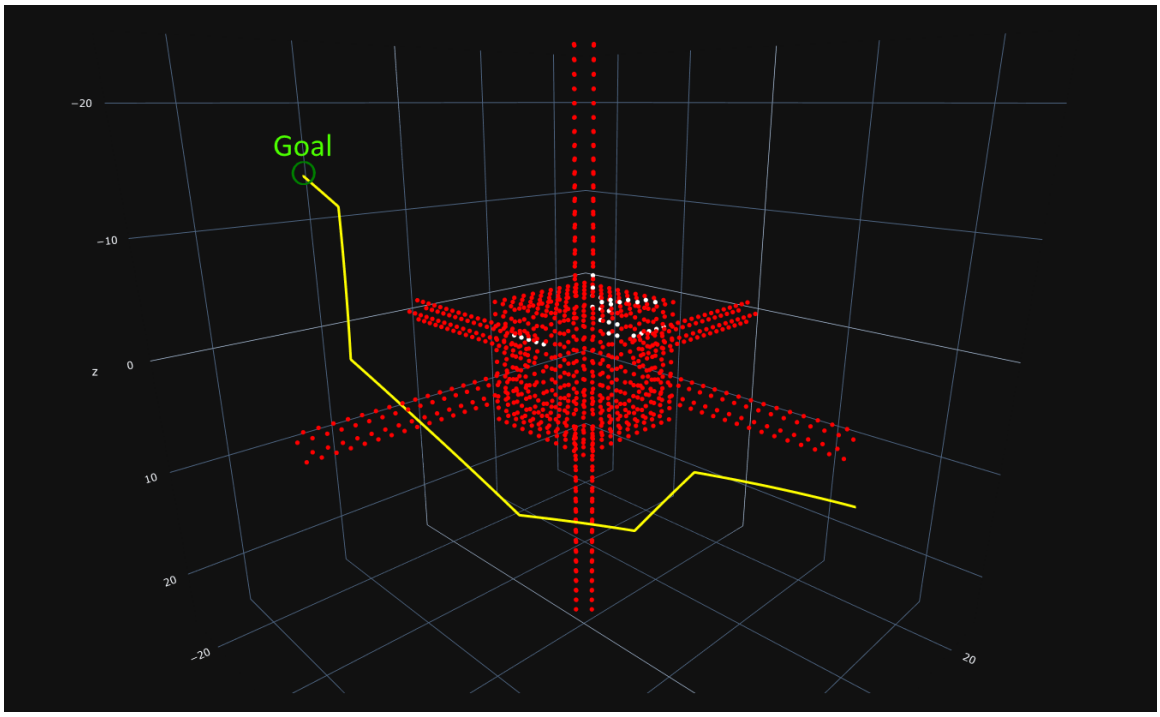

Figure 6.2: Test 1 final trajectory and obstacle model

blue dashed line shows the keep out distance that defines the threshold for switching to evasion mode. From this data, it is clear that the free-flyer avoided collision with the structure. The vehicle maintained its slower speed for normal guidance throughout the maneuver. The β angle and potential value plots show that the free-flyer smoothly descended the gradient with periodic firings.

The rotational response of the free-flyer during the maneuver is shown in Figure 6.4 and 6.5. As shown in the plots, the free-flyer achieved the desired orientation before $t = 20s$ and maintained that orientation by limit cycling for the rest of the maneuver. During limit-cycling the free-flyer's angular error stayed within the allowable deadband.

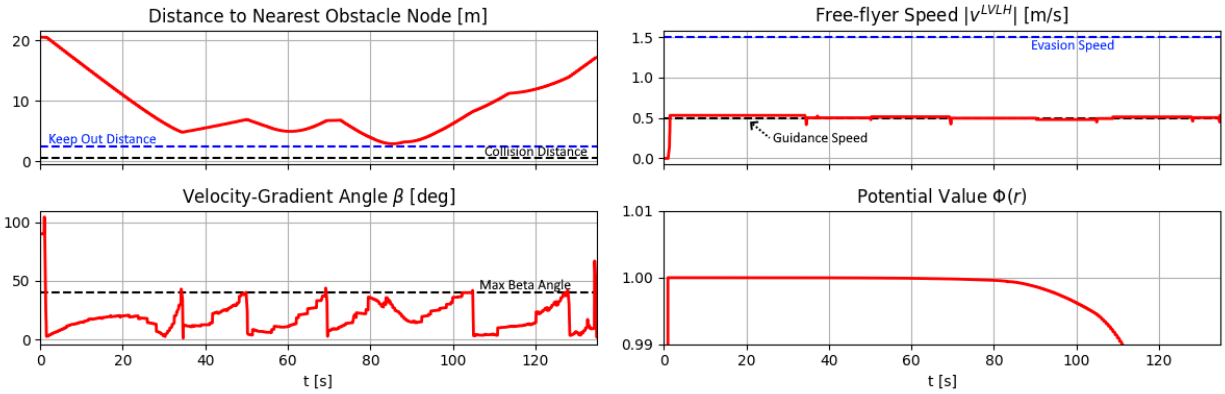

Figure 6.3: Test 1 guidance parameters

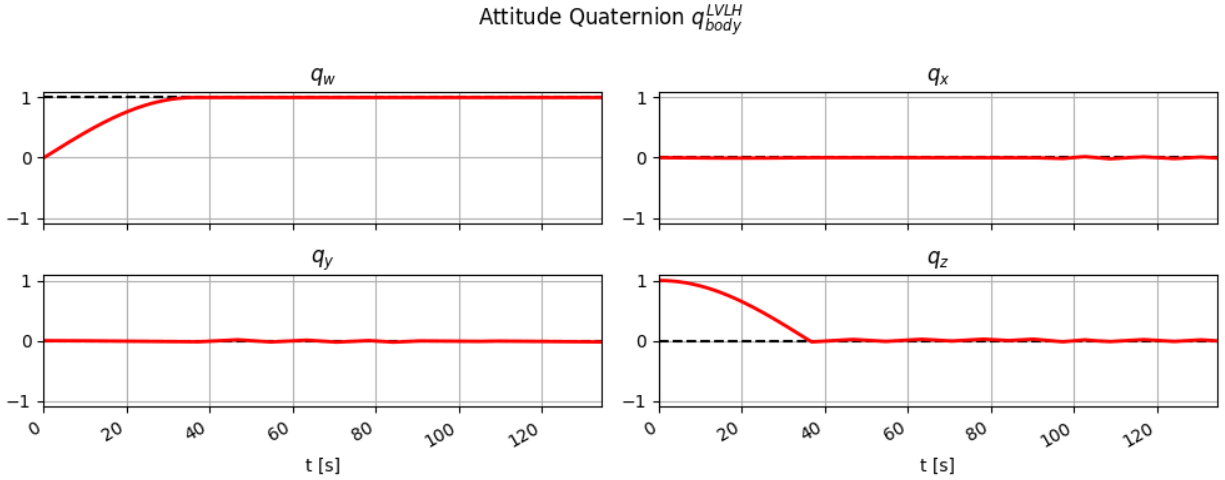

Figure 6.4: Test 1 attitude quaternion components

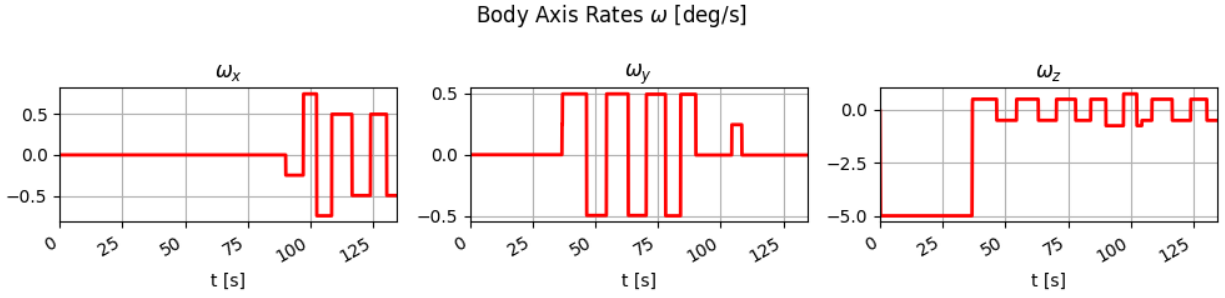

Figure 6.5: Test 1 vehicle body axis rates

6.3 Maneuvering Test 2

The second maneuvering test required the free-flyer to navigate a more constrained environment referred to as Maze. This environment provided the free-flyer with multiple potential paths, but limited line-of-sight. As a result, the capability of the G&C system to adapt to new information as the free-flyer explores the structure was emphasized in this test. During this maneuver, the free-flyer was also commanded to eliminate a large three-axis attitude deviation.

An overview of the final free-flyer trajectory and internal obstacle model is shown in Figure 6.6. There are two main paths in the middle section of this structure through which the free-flyer can pass, labelled A and B in the overview figure. Path A leads to a dead-end, but is closer to the goal position along the z-axis of the LVLH frame. Initially, the free-flyer moved towards path A since it appeared to be a more direct route to the goal based upon what sections of the structure are initially in the vehicle's line-of-sight. Once the free-flyer entered path A, the obstruction was mapped and the potential field was regenerated. The free-flyer reversed direction at this point and headed towards path B, which leads to the commanded goal position. Similarly to test 1, the minimum obstacle node distance plot in Figure 6.7 shows that the vehicle reached the goal position without any collisions or close-encounters with the structure. Also, the reversal in direction is visible as fluctuations in the potential value and β angle plots, as also shown in Figure 6.7.

For the rotational DOF, the free-flyer was commanded to reduce a large angular error along all three rotational axes. The attitude response of the vehicle is shown in Figure 6.8. The vehicle reached the desired orientation in under 30 seconds, and spent the rest of the maneuver limit-cycling. The vehicle body axis rates are shown in Figure 6.9.

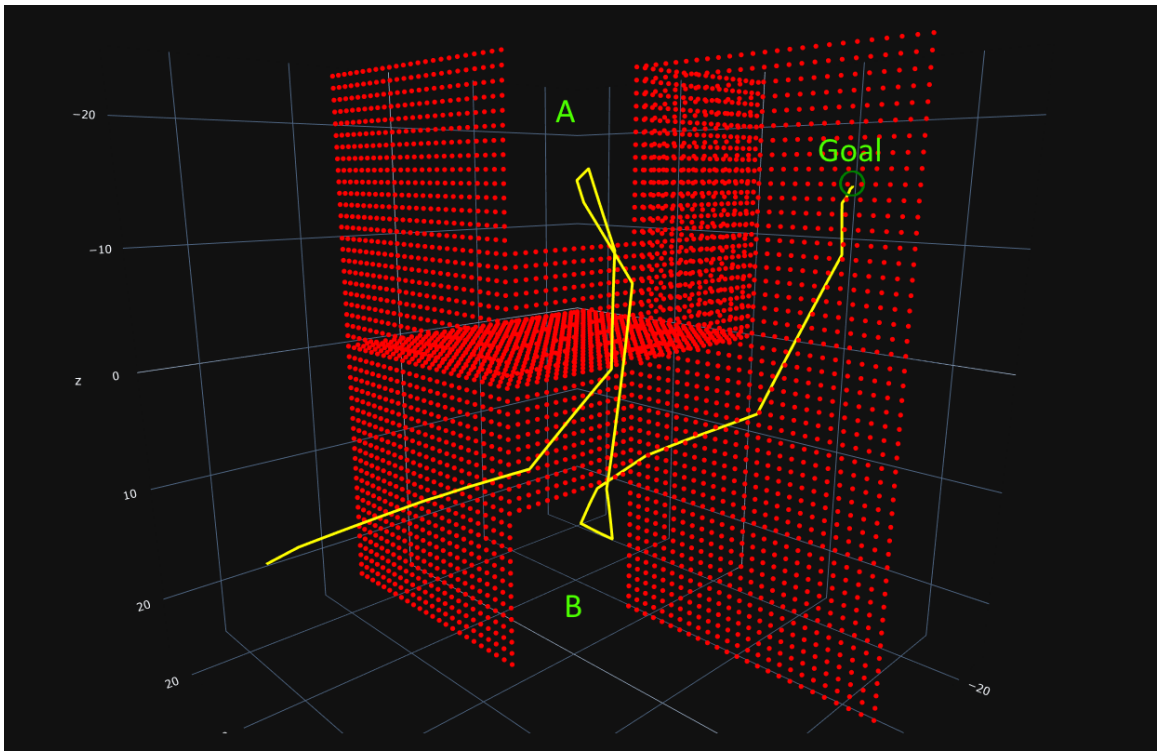

Figure 6.6: Test 2 final trajectory and obstacle model

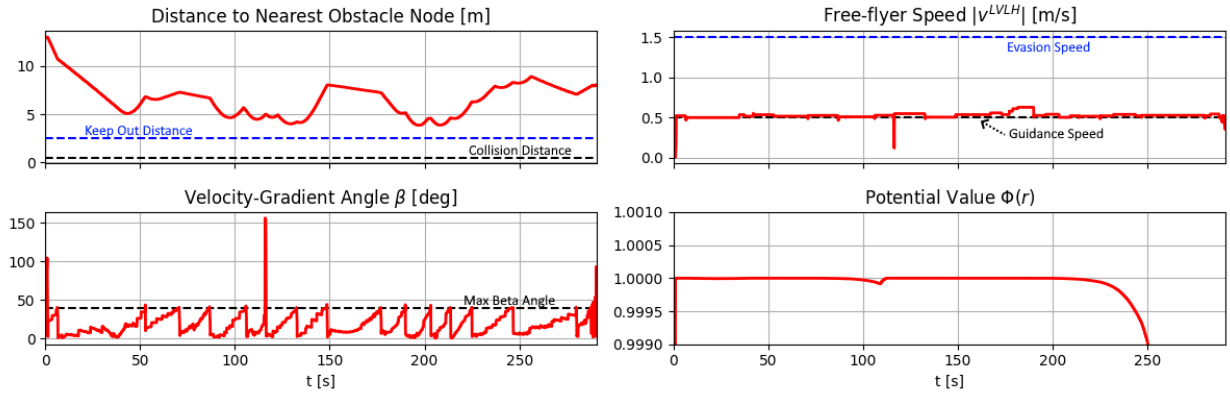

Figure 6.7: Test 2 guidance parameters

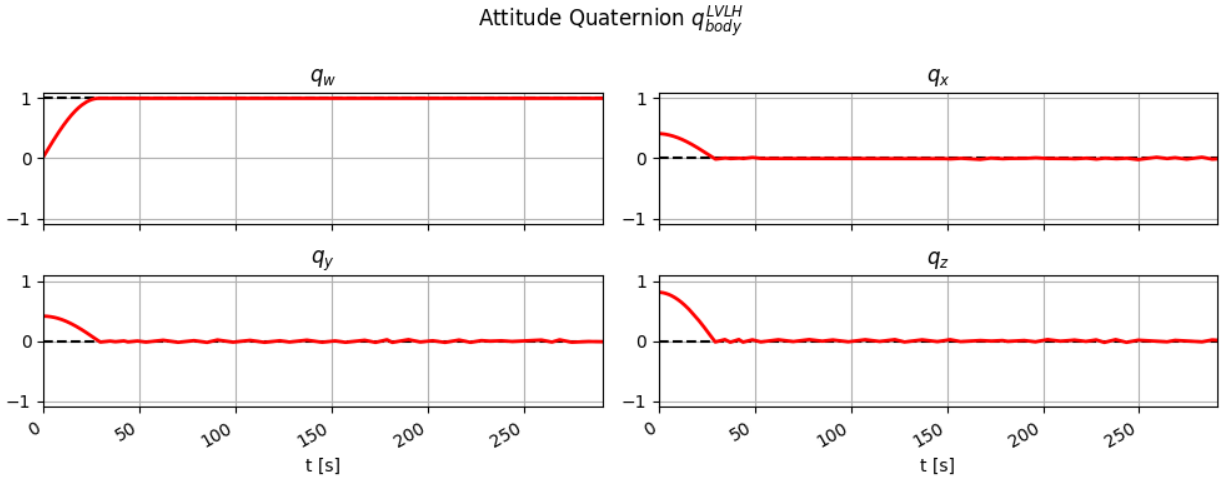

Figure 6.8: Test 2 attitude quaternion components

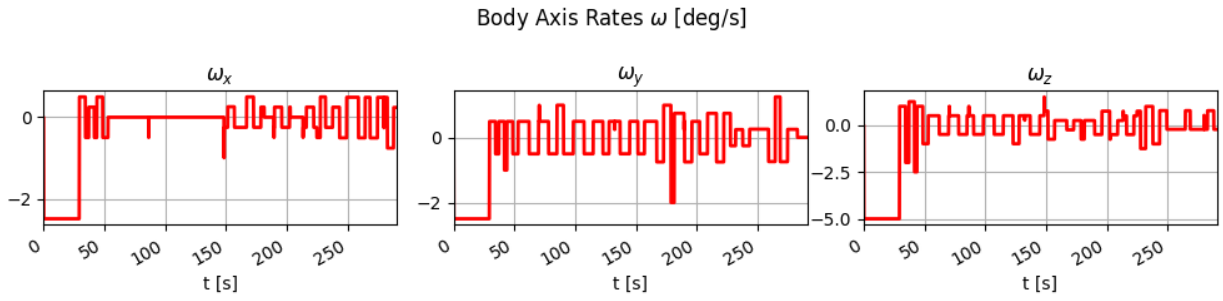

Figure 6.9: Test 2 vehicle body axis rates

6.4 Maneuvering Test 3

The third maneuver test involved a more typical free-flyer operation where the vehicle maneuvered near the exterior of a larger spacecraft. The Ship structure represents the exterior of a fictitious habitable spacecraft. This spacecraft is oriented with it's bow facing along the x-axis of the LVLH frame, which is the orbital direction. The mid-section of the ship features a large set of spinning arms, as shown in Figure 6.10. These arms served as dynamic obstacles for the free-flyer to avoid as it maneuvers from the aft part of the ship to its front, and vice-versa.

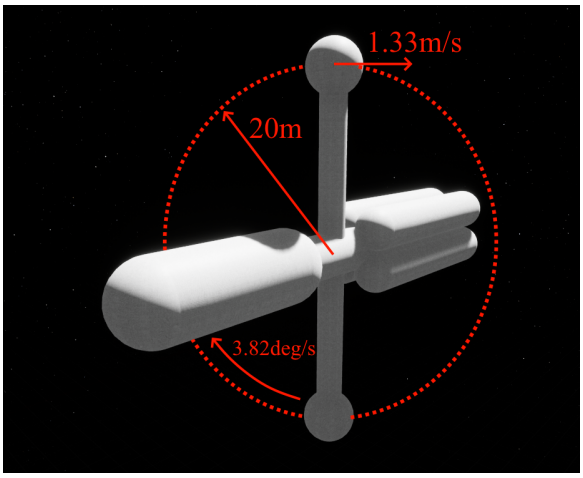

Figure 6.10: The rotating arm obstacles in the Ship obstacle environment

In this test the vehicle was commanded to traverse from the aft side of the ship to the forward side, as well as from the port side of the ship to the starboard. The vehicle was given an orientation command that required a 180 degree rotation about its z-axis. Additionally, the vehicle started with a large three axis angular velocity to control.

The final trajectory and internal obstacle model from this test are shown in Figure 6.11. As evident from the final trajectory, the free-flyer successfully navigated to the commanded position. Along the way, the vehicle had a close encounter with one of the swinging arms. This close encounter caused the free-flyer's guidance system to go into evasion mode at approximately $t =$

$22s$, as shown in the guidance plots of Figure 6.12. This section of the trajectory is labelled A in the overview figure. The increased vehicle speed for this collision avoidance maneuver is also visible in the guidance parameter plots.

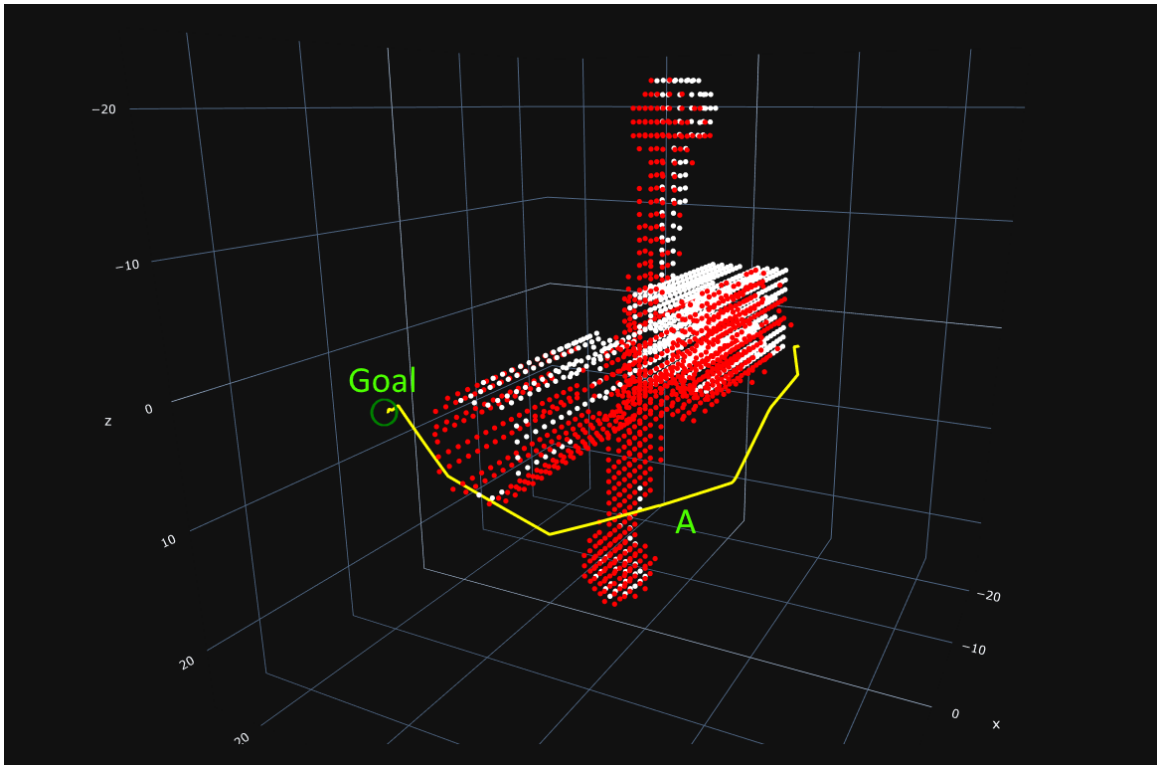

Figure 6.11: Test 3 final trajectory and obstacle model

The attitude response of the vehicle is shown in Figures 6.13 and 6.14. The initial angular velocity was quickly reduced by the attitude controller and the free-flyer reached the commanded orientation with no further deviation from the allowable deadband.

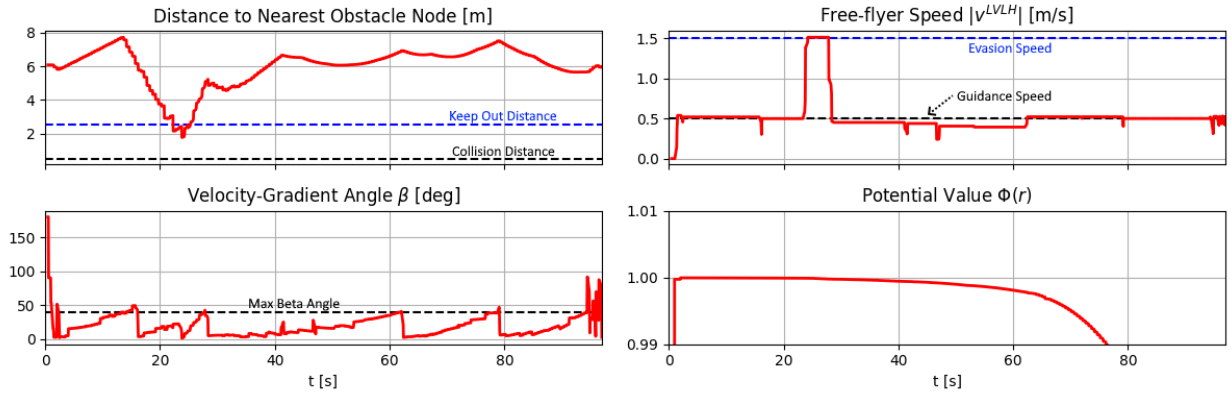

Figure 6.12: Test 3 guidance parameters

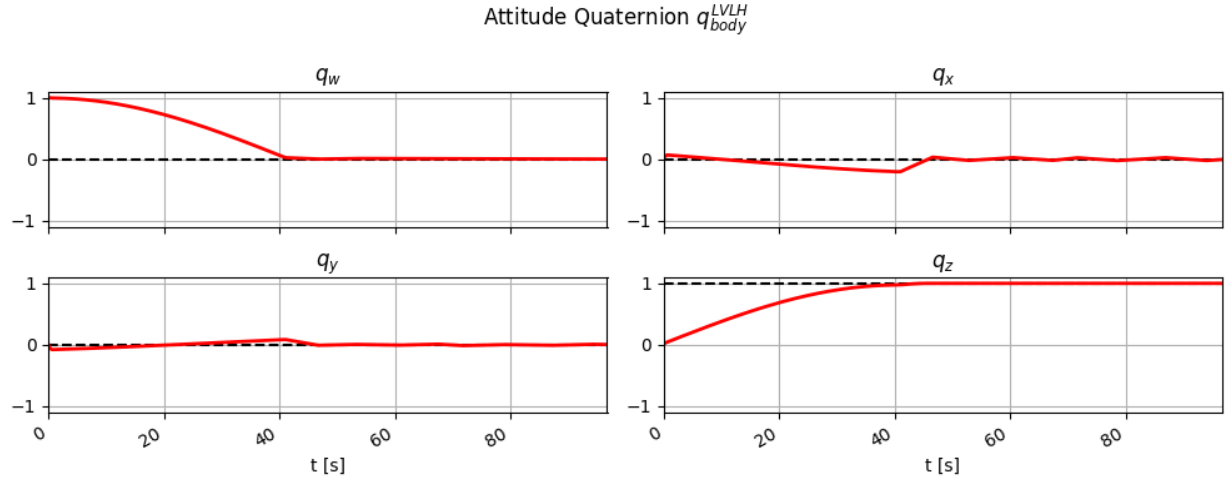

Figure 6.13: Test 3 attitude quaternion components

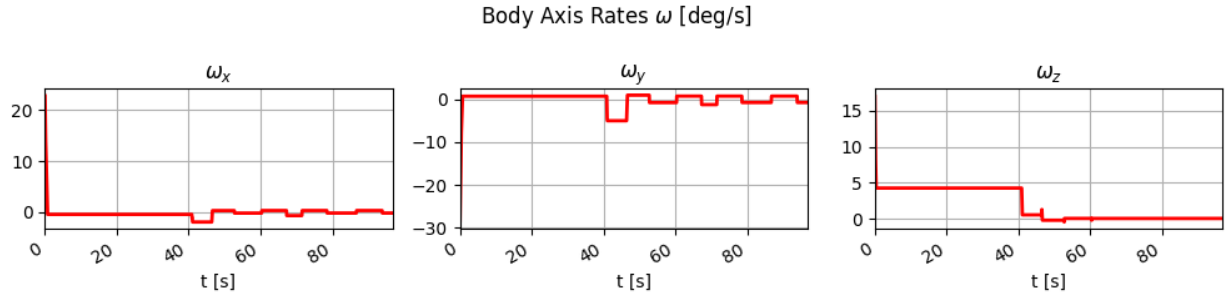

Figure 6.14: Test 3 vehicle body axis rates

6.5 Maneuvering Test 4

For the last maneuver test, the free-flyer was placed in a highly dynamic environment called *Asteroids*. In this environment, there is a mix of static and dynamic spheres. Four of those spheres are set to orbit the central structure, as shown in the environment diagram in Figure 6.15.

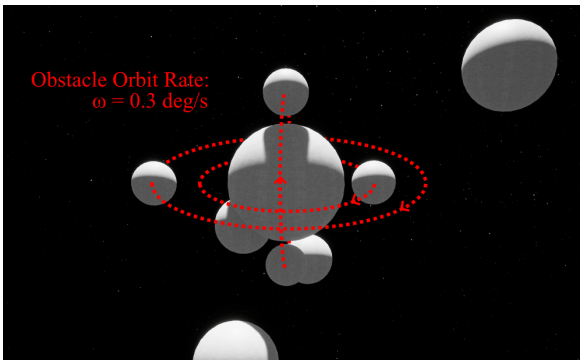

Figure 6.15: The orbiting dynamic obstacles in the Asteroids obstacle environment

The free-flyer was commanded to traverse this obstacle field, placing it in the direct path of the orbiting spheres to test its dynamic obstacle avoidance capabilities. Additionally, the free-flyer was commanded to execute a three-axis rotational maneuver to reach an arbitrary unaligned orientation.

The resulting final trajectory of the free-flyer and the state of its internal model are shown in Figure 6.16. At the section of the trajectory labelled A in this figure, the free-flyer performed evasive maneuvers to avoid one of the orbiting spheres that moved directly into its path. The response of the guidance system to this collision avoidance maneuver is visible in the guidance parameter plots of Figure 6.17. The free-flyer quickly reversed direction and increased its speed to avoid the incoming obstacle.

The attitude response of the free-flyer is shown in Figures 6.18 and 6.19. The free-flyer reached the command orientation without issue, and maintained this orientation despite the collision avoidance event.

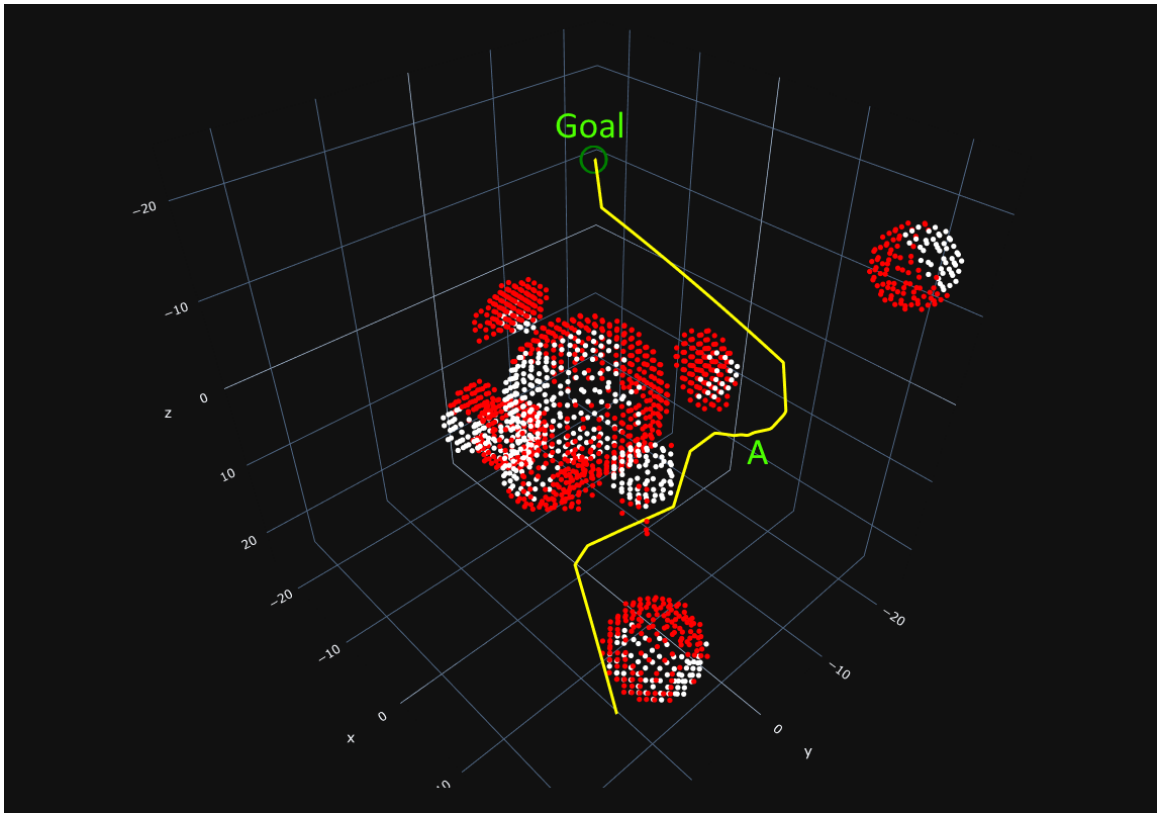

Figure 6.16: Test 4 final trajectory and obstacle model

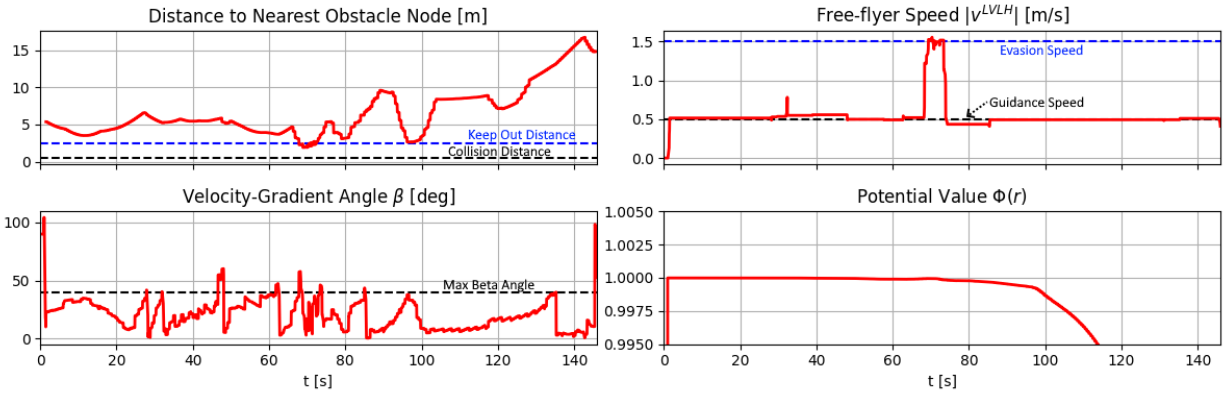

Figure 6.17: Test 4 guidance parameters

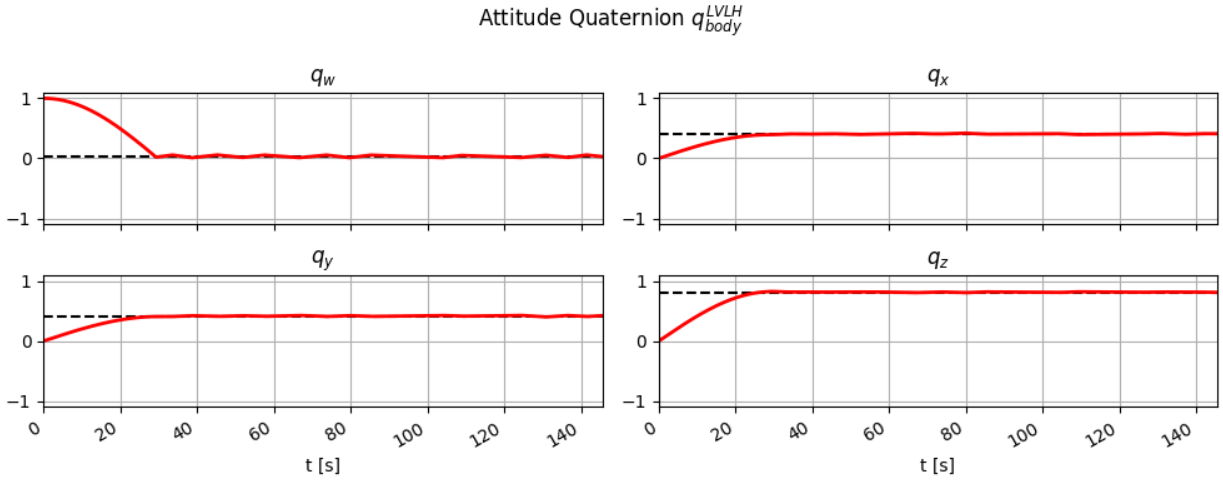

Figure 6.18: Test 4 attitude quaternion components

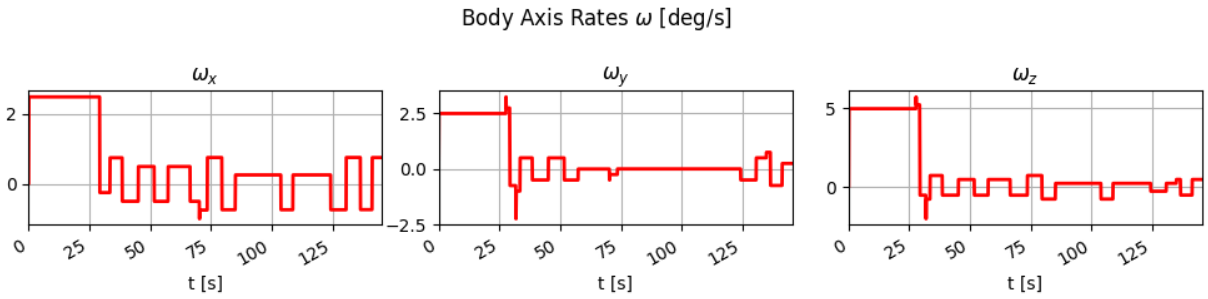

Figure 6.19: Test 4 vehicle body axis rates

6.6 Obstacle Mapping Performance

The previous section demonstrates the G&C system behavior for a variety of environments and initial conditions. In all four tests, the free-flyer was able to maneuver from the initial position to the goal position by constructing an internal model of the surrounding environment based on its line-of-sight. The final trajectory overview of Figures 6.2, 6.6, 6.11, and 6.16 only show the final state of the free-flyer's internal model. For fully static structures, any white nodes in these figures are nodes that were never in the free-flyer's line-of-sight, and therefore were excluded from the internal obstacle model managed by the guidance process. For dynamic obstacles, only the final poses of the obstacles are reflected in these overview figures. The internal model refinement performed by the free-flyer's guidance system is not well shown. To better show the progression of the obstacle mapping process, Figures 6.20, 6.21, 6.22, and 6.23 were created.

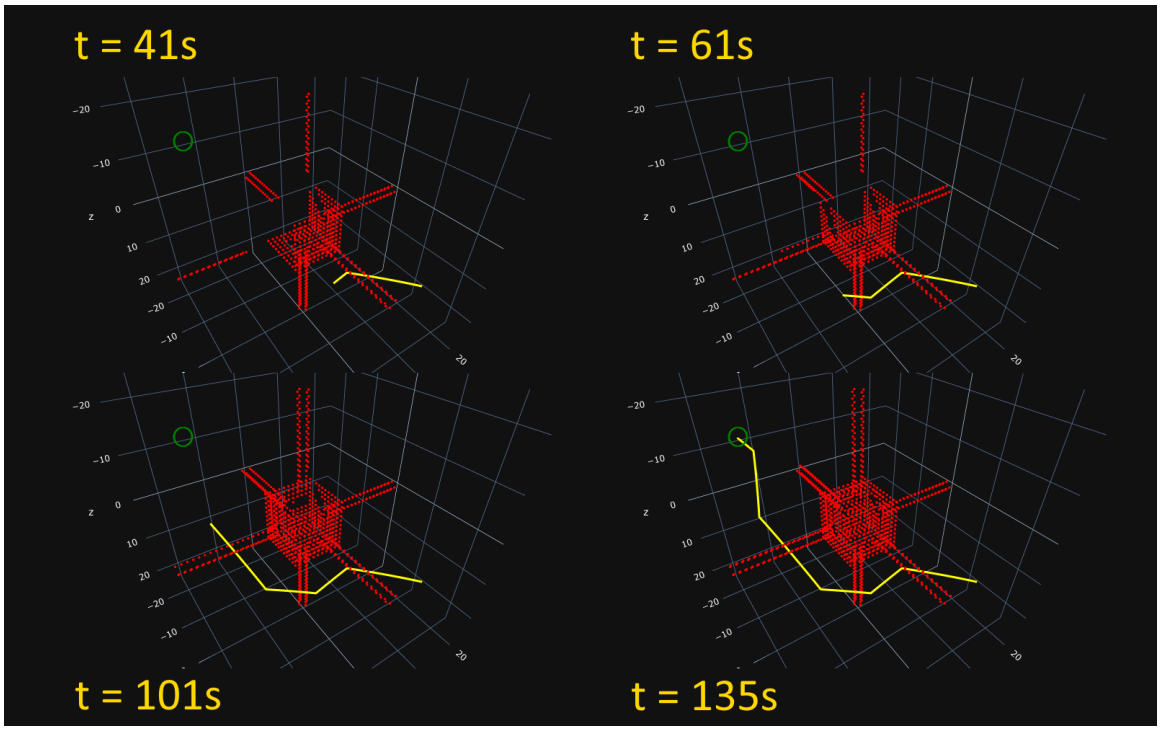

Figure 6.20: Test 1 obstacle mapping progression

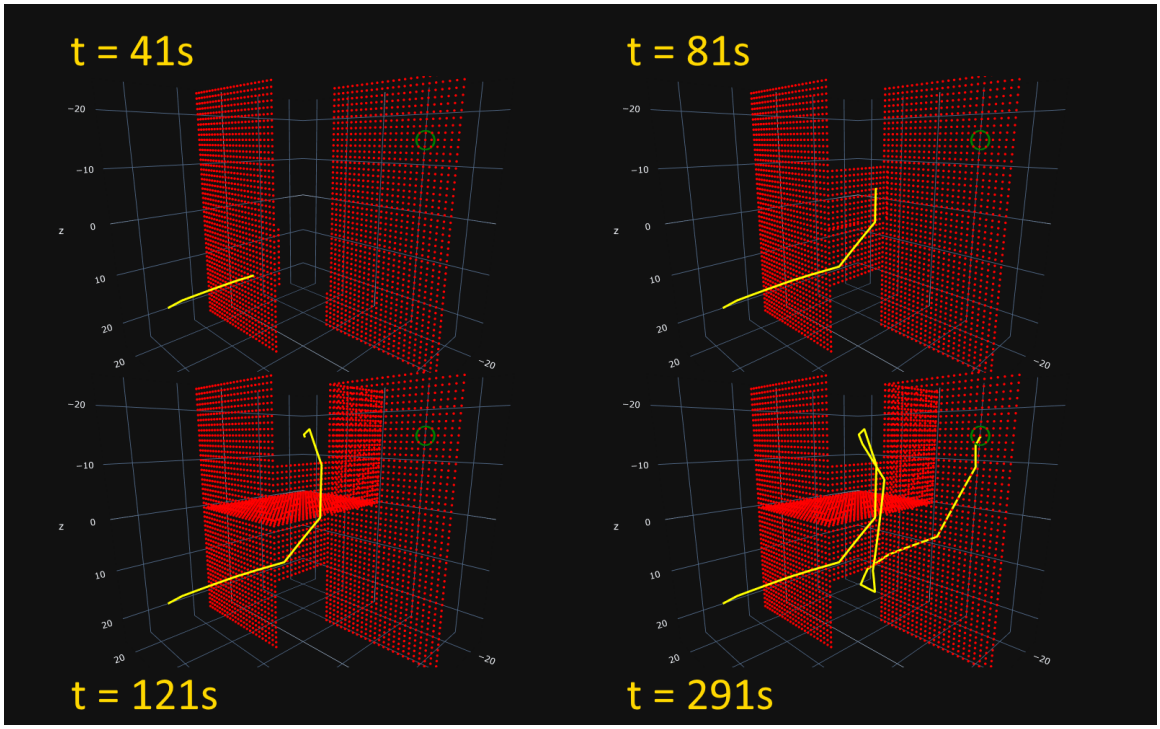

Figure 6.21: Test 2 obstacle mapping progression

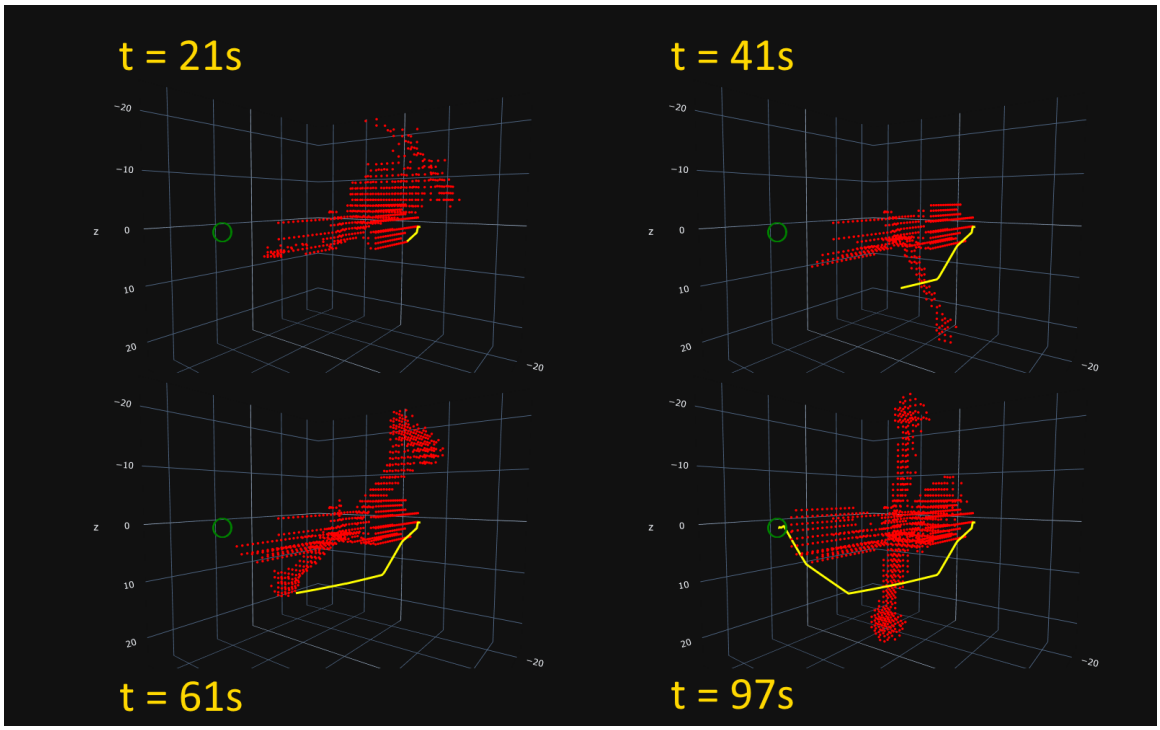

Figure 6.22: Test 3 obstacle mapping progression

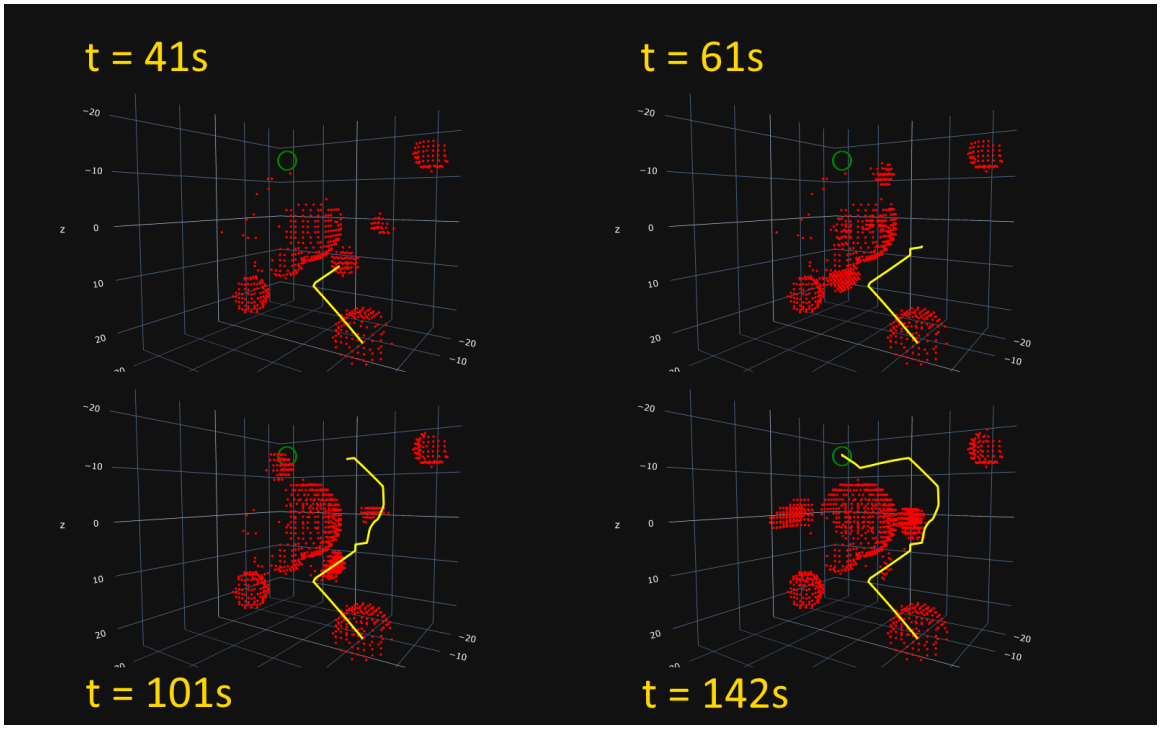

Figure 6.23: Test 4 obstacle mapping progression

6.7 Plume Impingement Avoidance Performance

Another major capability of the G&C system is active plume impingement avoidance. To assess this capability, this thesis attempts to quantify the system's ability to reduce plume impingement and the potential degradation of other performance parameters. For the purpose of generating comparison data, the test-bed simulation includes an option to disable plume avoidance. If this option is enabled, the plume cost for each jet is still calculated, but it is not considered during jet selection. Therefore, the output jet firing commands are time optimal in this mode (or also fuel optimal given that each jet has the same mass flow rate). Using this option, simulation runs with this option enabled allows for the comparison of relative plume impingement costs, as well as other performance parameters.

Given the non-deterministic nature of the test-bed simulation due to its asynchronous architecture, each simulation run results in a new free-flyer trajectory. Trajectory differences are highly coupled with total accumulated plume impingement cost. Therefore, given the current capabilities of the test-bed simulation, the effectiveness of plume impingement avoidance is evaluated by comparing a set of runs with plume avoidance and a set of runs with avoidance disabled. Figures 6.24, 6.25, 6.26, and 6.27 compare the system performance for each of the four maneuver tests. In each test, five runs are with plume avoidance and five are without.

Table 6.2 includes some statistics about the final cost values and maneuvering time. The plots as well as the average performance data show that the effectiveness of plume avoidance depends on the nature of the obstacle environment. Plume avoidance achieved large reductions in the final plume cost for Maneuvering Tests 1, 3, and 4 as shown. Test 3 however shows a slight plume cost increase with plume avoidance enabled. Notably, this test features the most constrained structural environment, where the free-flyer navigates inside the Maze structure.

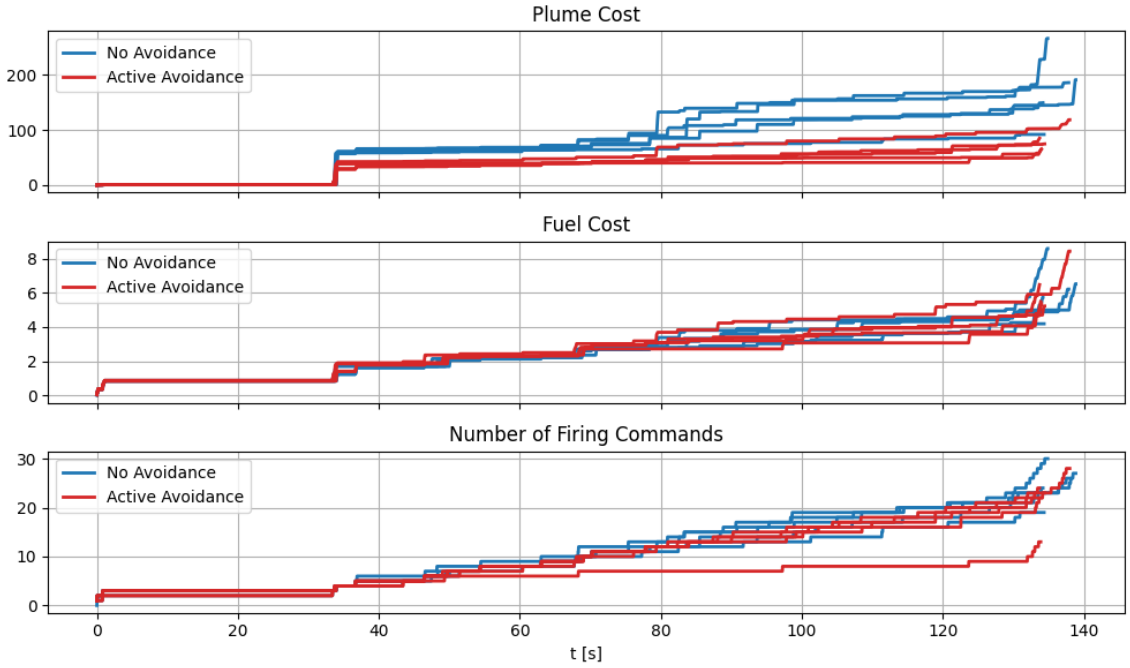

Figure 6.24: Test 1 plume avoidance performance and related parameters

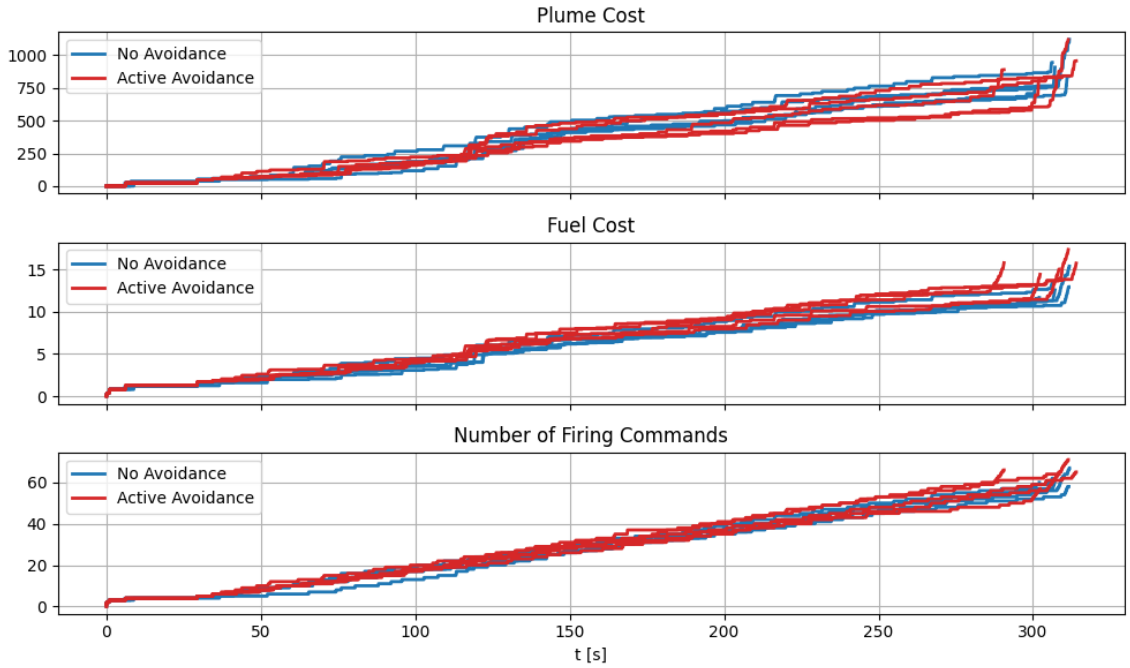

Figure 6.25: Test 2 plume avoidance performance and related parameters

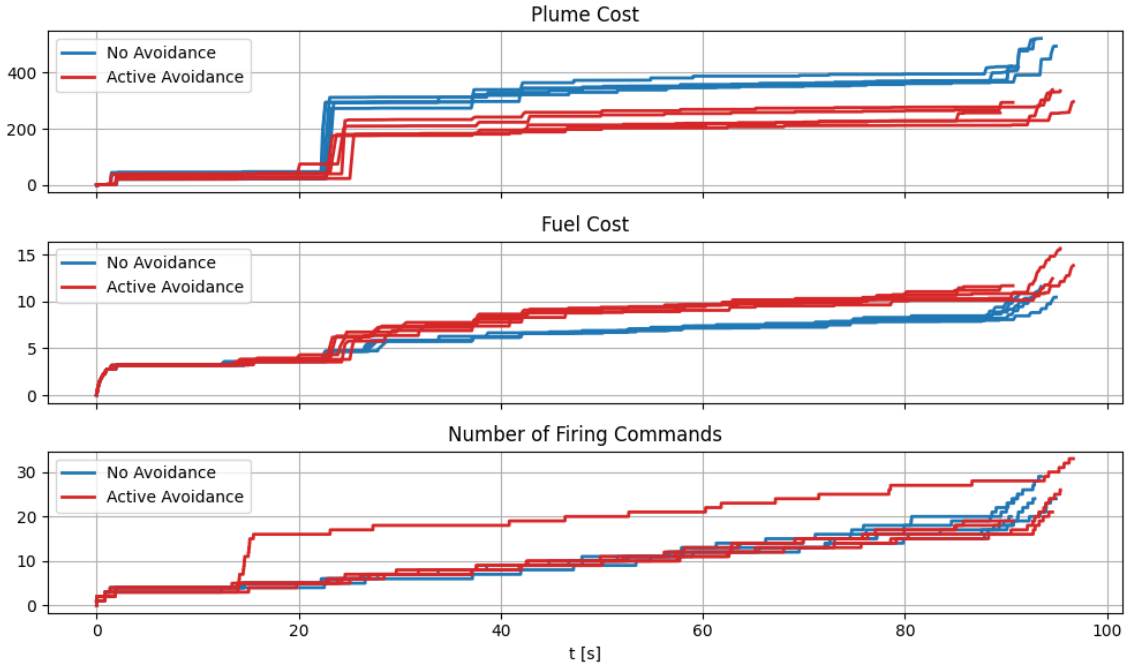

Figure 6.26: Test 3 plume avoidance performance and related parameters

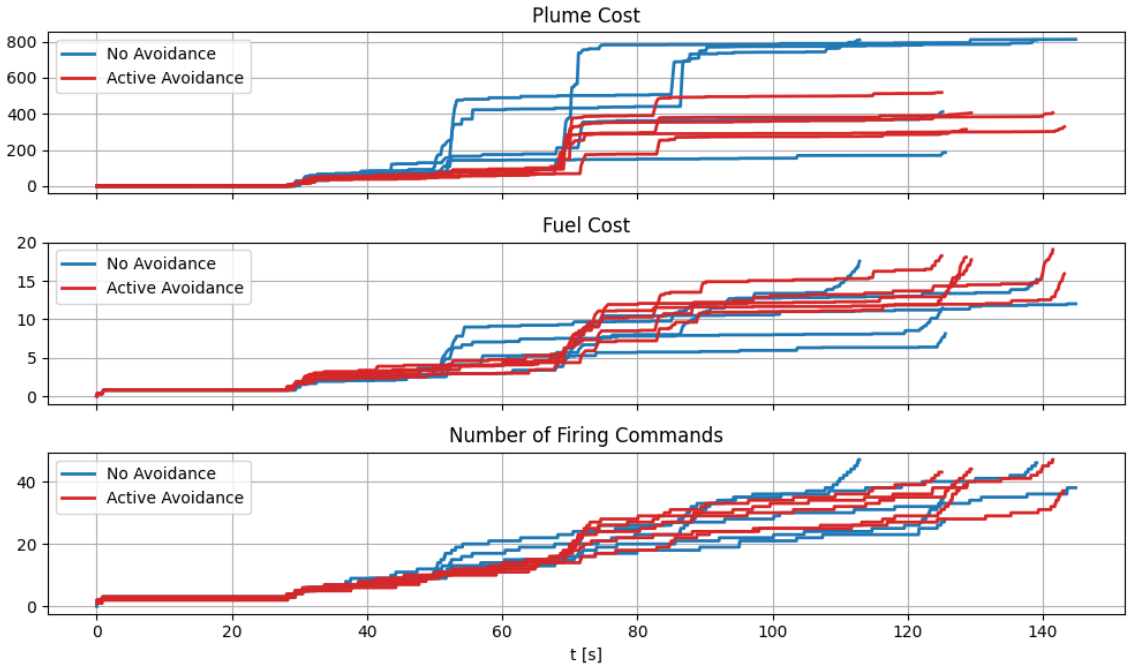

Figure 6.27: Test 4 plume avoidance performance and related parameters

Performance Parameter	Avoidance	Test1	Test2	Test3	Test4
Avg. Plume Cost	Enabled	81.47	924.63	303.82	394.49
	Disabled	176.79	913.73	474.22	605.61
Avg. Fuel Cost [s]	Enabled	6.20	15.66	13.02	17.82
	Disabled	6.25	13.19	10.79	12.87
Avg. Number of Firings	Enabled	22.0	64.6	23.6	42.0
	Disabled	25.2	61.6	24.4	38.6
Avg. Maneuver Time [s]	Enabled	134.78	305.36	93.35	133.54
	Disabled	136.00	307.75	92.61	129.50

Table 6.2: Maneuvering and plume avoidance performance summary

6.8 Discussion

The simulation results presented in this chapter represent the performance of the system for the reference free-flyer and the system parameters chosen for testing. The results are also dependent on the specific software implementation of the G&C system and the architecture of the test-bed simulation with which it interacts. Therefore, with the context of these results being considered, the behavior and capabilities of the system can be assessed.

6.8.1 Obstacle Avoidance Behavior

Among the key capabilities that the G&C system sought to perform, collision avoidance for static obstacles is among the most basic. All four maneuvering test cases demonstrate the ability of the G&C system to maneuver safely in the presence of static obstacles. Even in the Maze obstacle environment, the G&C system enabled the free-flyer to maintain a steady and safe progression towards the goal position, even as its exploration of the structure required it to double back on its path. In this case, lack of complete information about the configuration space did not inhibit the guidance process from producing a safe trajectory. This is also demonstrated in test cases 3 and 4, where the free-flyer reaches the goal position with large portions of those structures remaining unmapped throughout the entire maneuver. This is visible in Figures 6.11 and 6.16 as large regions of white dots representing unmapped obstacle nodes.

The presence of dynamic obstacles complicated the G&C system's ability to maintain safe

and steady progression towards the goal position. A fundamental guarantee of guidance based on the use of potential fields is lost when introducing dynamic obstacles. For impulsive changes in velocity, the GIM approach to reactive guidance covered in Chapter 3 guarantees the descent of the potential field. This is true when the potential field is constant, but not when the field is regenerated periodically. The motion of dynamic obstacles can simply outpace the ability of the G&C system to react accordingly. This is especially true for obstacles that move at a speed greater than the user defined evasion speed. With these limitations in mind, the evasion mode approach to dealing with dynamic obstacles is successful in test cases 3 and 4. The reference free-flyer tested has a fairly large thrust to weight ratio, and was able to quickly move clear of incoming obstacles as shown in the test results. For the application of this G&C system in a real free-flyer operation, the potential nature and speed of any dynamic obstacles would become major factors in the choice of system parameters and even vehicle design.

6.8.2 Attitude Control Behavior

The test results demonstrate that, during the execution of translational path-finding and maneuvering, the attitude control process was effectively uninterrupted. In the four test cases, the free-flyer was required to produce a variety of rotational maneuvers. In all cases, the attitude controller enabled the free-flyer to reach the commanded orientation in an amount of time roughly proportional to the angular error between the initial and commanded attitude. In test case 3, the free-flyer is also given an initial angular velocity to manage. With the high thrust to inertia ratio of the reference free-flyer, the attitude controller was able to quickly eliminate this angular velocity without issue. In all cases, after the commanded attitude was reached, the free-flyer limit cycled within the allowable deadband for the rest of each maneuver.

One potential limitation of the attitude controller is that it will be less effective for free-flyer designs with highly unbalanced moments of inertia. The attitude control law (4.8) directs the new angular velocity along the axis obtained by representing the angular error in an axis-angle representation. The nonlinear terms of equation (2.6) become larger for vehicles with unbalanced inertia tensor components. This non-linearity can render the control law ineffective for large rotational

maneuvers. To adapt the G&C system for unbalanced spacecraft an alternative attitude control law that can handle this nonlinearity should be used, especially for large angle slewing.

6.8.3 Plume Impingement Avoidance Behavior

From the simulation results, the effectiveness of the plume impingement avoidance functionality appears to depend strongly on the free-flyer's environment. With plume avoidance enabled, there is a clear reduction in plume cost in tests 1, 3, and 4. Test 3, however, tells a different story. The plume cost for test 3 is actually slightly higher with plume avoidance enabled than with it disabled.

The reason for this likely has a lot to do with the Maze obstacle environment. The other obstacle environments generally feature convex obstacles with plenty of open space. On the other hand, the Maze environment is essentially concave, with the vehicle operating inside the structure. The plume avoidance functions by allowing the jet selection to choose alternative jet combinations with a lower overall plume cost. In a highly constrained environment there are obstacles in most directions from the vehicle, and therefore fewer opportunities to optimize for plume avoidance.

In these test cases, the structural components were equally weighted in the computation of plume costs. Instead, these results suggest that designating sensitive structural components with a higher structural weighting relative to the rest of the structure would produce more favorable results. Therefore, the parts of the structure that plume impingement can affect the most are prioritized, and jet selection has more favorable jet combinations to choose from.

The results also suggest that this method of plume avoidance can cause degradation of other performance criteria. In test cases 2, 3, and 4 the fuel cost with plume avoidance enabled was slightly higher than with it disabled. This is an expected consequence of optimizing for plume impingement and fuel usage, instead of just optimizing for fuel optimal solutions. The average number of jet firings and total maneuver time were roughly unaffected based on these results, especially given the relatively small run sample size.

One limitation of this plume avoidance approach is that it relies on free-flyer designs with many jets. The simple reason for this is that more jets pointed in different directions create fewer options

for plume avoidance optimization. For a minimal jet configuration with 12 jets there are few, if any alternative jet firings to consider during optimization. Working around this limitation would potentially require a more active approach, changing large aspects of the overall G&C system to be more plume impingement friendly. In particular, with fewer jets the vehicle's attitude becomes a significant factor for plume impingement avoidance and therefore it may be necessary to tie the attitude controller into the optimization process.

7. SUMMARY AND CONCLUSIONS

Free-flyer space robots have huge potential for aiding future human spaceflight and operations. To perform tasks and procedures usually reserved for human crew, both outside and inside a large spacecraft or station, these robots will be required to operate with a high degree of autonomy. One of the fundamental capabilities of autonomous robotics is path-finding, which for free-flyers entails point-to-point maneuvering in the presence of obstacles and other constraints.

7.1 Conclusion

In this thesis, a G&C system that enabled this kind of autonomous maneuvering was developed and tested. This system combined and adapted several existing techniques to control a vehicle in both translational and rotational DOF with a realistic RCS propulsion system. The critical capability of the control system is to enable the free-flyer to maneuver around space structures and obstacles along a collision-free trajectory. This system is distinct from conventional approaches where a complete trajectory is planned at the outset of a maneuver. Instead, this system functions more reactively, and relies on its understanding of the surrounding environment and a set of conditions to guide the free-flyer towards its goal. This approach was intended to give the free-flyer more flexibility in an uncertain environment, while potentially sacrificing some optimality in the final trajectory.

The final G&C system consists of four separate processes that interact to produce free-flyer RCS jet firing commands. Translational manuevering and path-finding is performed through the use of artificial potential fields that are generated using an internal model of the free-flyer's environment. This internal model is constructed based on the assumption that the free-flyer has some ability to detect obstacle geometry within its line of sight, through the use of 3D camera or lidar instrumentation. The other processes handle attitude control, jet selection, and a process to optimize jet firings for plume impingement avoidance. Separating these functions into distinct processes allows for extensive parallelization of the computations, which increases its feasibility for real-time

operation.

These four processes, as detailed in this thesis, were then implemented into a test-bed simulation. The test-bed simulation handles physics propagation, the motion of static and dynamic obstacles, and generates simulated obstacle visibility data for the G&C system. The resulting behavior of the G&C system as observed in the test-bed simulation verified its intended capabilities. In particular, the G&C system enabled the simulated free-flyer to effectively traverse obstacle environments without collision, including both static and dynamic obstacles. The test-bed simulation results also show that the free-flyer was able to effectively control its rotational DOF and achieve a reduction in plume impingement.

The test-bed simulation also demonstrated potential limitations of the system, including the need for more robust obstacle avoidance strategies for high-speed dynamic obstacles and the limited effectiveness of plume avoidance in highly constrained environments. Additionally, the system assumes that the free-flyer has relatively balanced moments of inertia and is not capable of throttling jet thrust. These limitations impose constraints on compatible free-flyer designs. Perhaps the most severe requirement is the need for the free-flyer to have a highly redundant RCS jet array, which creates the alternative jet firings needed by the plume avoidance process. For free-flyer designs with minimal RCS systems, this limitation may degrade plume avoidance performance. This limitation, however, does not affect the path-finding algorithm used in the guidance process.

Overall, the development and testing of this system provided a chance to extensively explore this alternative paradigm for six DOF maneuvering, and what an integrated G&C might look like. Many of the limitations encountered in the generation of simulation results can be addressed through changes to individual processes. Reactive guidance using potential fields provides a powerful basis for free-flyer autonomous maneuvering, and is amenable to being extended and incorporated into a broader system as demonstrated. Additionally, the test-bed simulation provided a useful sandbox for system development and experimentation,

7.2 Future Work

There are many potential directions for future development and expansion of this work. The free-flyer G&C system proposed and tested in this thesis has much room for refinement, optimization, and adaptation. Additionally, the test-bed simulation itself has much room for expanded functionality, including visualization tools, better simulated obstacle mapping, and more complex obstacle geometry.

While there are many potential directions to continue this work, a few major areas stand out as the most significant in terms of validating the feasibility of the proposed approach. The first major area for future development would be to address free-flyer navigation through the use of Simultaneous Localization and Mapping (SLAM) algorithm. These algorithms estimate the state of the vehicle or sensor by building a map of the local environment [38]. The form of this map varies from algorithm to algorithm, but generally contains a set of environment *features* which are tracked to estimate the motion of the sensor. The map produced by SLAM algorithms is used for localization, and generally is not useful for trajectory planning or in this case reactive guidance [25]. Still, the integration of SLAM in this work would relax the simplifying assumption that the free-flyer has perfect state information.

One particular type of SLAM algorithm that would be interesting to integrate is Point-Cloud SLAM. Point-Cloud SLAM uses a sensor such as an RGBD camera to build a point-cloud representation of the environment. To implement Point-Cloud SLAM into this work would require the development of a virtual sensor that models the output of an RGBD camera or similar sensor. The algorithm uses the point-cloud data as features to perform localization. A 3D obstacle mapping algorithm that uses point-cloud data could also be implemented, which would replace the current simulated obstacle mapping process. These expansions would demonstrate the ability of the proposed G&C system to operate using an estimated state and while running actual mapping algorithms, enhancing the feasibility of the system.

Another area for future work is the improvement of dynamic obstacle avoidance. The use of SLAM could enable more robust logic for the avoidance of fast moving obstacles. Specifically, if

dynamic obstacles could be detected as features in a mapping process, the G&C system could react to their estimated future motion instead of just their instantaneous positions. Given the estimated instantaneous position and velocity of a dynamic feature during obstacle mapping, a cone that covers the potential linear motion of the feature can be defined. This cone could be used to generate obstacle nodes in the guidance process, and enable the free-flyer to move out of the obstacle's path much sooner.

The last major area for future work is to incorporate techniques for changing the size, shape, and resolution of the potential field grid automatically to adapt to different situations. In order to not limit potential free-flyer trajectories, the potential field grid needs to properly encompass the obstacle environment. In some cases, especially for very large space structures, the required grid size will become computationally infeasible. To handle maneuvers that cover such large distances, a strategy for moving and resizing the grid could be implemented to allow the free-flyer to reach distance goal positions. To do this would require a method for placing intermediate goal positions for when the true goal position lies outside of the current grid boundaries. As the free-flyer nears both the intermediate and true goal positions, the grid would be moved appropriately and a new intermediate position would be chosen in the direction of the true goal.

REFERENCES

[1] A. B. Roger, *Free-flyer path planning in the proximity to large space structures*. PhD, University of Glasgow, 2003.

[2] S. Loff, "NASA Image Library," Jan. 2022.

[3] Q. Gao, J. Liu, T. Tian, and Y. Li, "Free-flying dynamics and control of an astronaut assistant robot based on fuzzy sliding mode algorithm," *Acta Astronautica*, vol. 138, pp. 462–474, Sept. 2017.

[4] J. Wagenknecht, S. Fredrickson, T. Manning, and B. Jones, "Design, Development and Testing of the Miniature Autonomous Extravehicular Robotic Camera (Mini AERCam) Guidance, Navigation, and Control System," in *26th Annual AAS Guidance and Control Conference*, (Breckenridge, CO), p. 21, American Astronautical Society, Feb. 2003.

[5] D. S. E. Fredrickson, "Mini AERCam: A Free-flying Robot for Space Inspection," tech. rep., NASA Johnson Space Center, 2001.

[6] S. Pedrotty, J. Sullivan, and E. Gambone, "Seeker Free-Flying Inspector GNC System Overview," tech. rep., NASA Johnson Space Center.

[7] D. Miller, A. Saenz-Otero, J. Wertz, A. Chen, G. Berkowski, C. Brodel, S. Carlson, D. Carpenter, S. Chen, S. Cheng, D. Feller, S. Jackson, B. Pitts, F. Perez, J. Szuminski, and S. Sell, "SPHERES: A Testbed For Long Duration Satellite Formation Flying In Micro-Gravity Conditions," tech. rep., MIT Space Systems Lab.

[8] T. Smith, J. Barlow, M. Bualat, T. Fong, C. Provencher, H. Sanchez, and E. Smith, "Astrobee: A New Platform for Free-Flying Robotics Research on the International Space Station," tech. rep., NASA Ames Research Center.

[9] M. Lazaron and J. Alred, "Results of the SPAS-01 RCS plume impingement test," in *23rd Aerospace Sciences Meeting*, Aerospace Sciences Meetings, American Institute of Aeronau-

tics and Astronautics, Jan. 1985.

[10] L. M. Bermúdez, K. J. Barnhart, and C. W. Brunner, "Modeling, Simulation, and Validation of Plume Impingement Effects on the Cygnus Spacecraft," *Journal of Spacecraft and Rockets*, vol. 55, no. 2, pp. 427–436, 2018.

[11] W. Rochelle, J. Hughes, J. D, S. Bouslog, K. Leahy, and S. Fitzgerald, "Plume impingement heating to International Space Station (ISS)," in *30th Thermophysics Conference*, Fluid Dynamics and Co-located Conferences, American Institute of Aeronautics and Astronautics, June 1995.

[12] W. H. Clohessy and R. S. Wiltshire, "Terminal Guidance System for Satellite Rendezvous," *Journal of the Aerospace Sciences*, vol. 27, no. 9, pp. 653–658, 1960.

[13] T. Lozano-Pérez, M. A. Wesley, and F. N. Fritsch, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications of the ACM*, vol. 22, pp. 560–570, Oct. 1979. Publisher: Association for Computing Machinery.

[14] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, Dec. 1959.

[15] R. A. Brooks, "Solving the find-path problem by good representation of free space," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, pp. 190–197, Mar. 1983. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[16] R. Brooks, "Solving the Find-Path Problem by Representing Free Space as Generalized Cones," Memo 674, Massachusetts Institute of Technology Artificial Intelligence Laboratory, July 2002.

[17] D. Ferguson, M. Likhachev, and A. Stentz, "A Guide to Heuristic-based Path Planning," tech. rep., Carnegie Mellon University, 2005.

[18] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 501–518, Oct. 1992. Conference Name: IEEE Transactions on Robotics and Automation.

[19] S. Ge and Y. Cui, "New potential functions for mobile robot path planning," *IEEE Trans. Robotics Autom.*, 2000.

[20] J.-O. Kim and P. Khosla, "Real-Time Obstacle Avoidance Using Harmonic Potential Functions," *IEEE Trans. Robotics Autom.*, Jan. 1992. Publisher: Carnegie Mellon University.

[21] Y. K. Hwang, N. Ahuja, and S. Member, "A potential field approach to path planning," *T-Ra*, 1992.

[22] J. F. Raquet, "Six degree of freedom trajectory planner for spacecraft proximity operations using an A* node search," Master's thesis, Massachusetts Institute of Technology, 1991.

[23] M. C. Jackson, "A six degree of freedom, plume-fuel optimal trajectory planner for spacecraft proximity operations using an A* node search. MS Thesis-MIT," Master's thesis, Massachusetts Institute of Technology, 1994.

[24] G. E. Chamitoff, A. Saenz-Otero, J. G. Katz, S. Ulrich, B. J. Morrell, and P. W. Gibbens, "Real-time maneuver optimization of space-based robots in a dynamic environment: Theory and on-orbit experiments," *Acta Astronautica*, vol. 142, pp. 170–183, Jan. 2018.

[25] B. Morrell, *Enhancing 3D Autonomous Navigation Through Obstacle Fields: Homogeneous Localisation and Mapping, with Obstacle-Aware Trajectory Optimisation*. Thesis, University of Sydney, June 2018. Accepted: 2019-02-14.

[26] Y. Chen, Z. He, D. Zhou, Z. Yu, and S. Li, "Integrated guidance and control for microsatellite real-time automated proximity operations," *Acta Astronautica*, vol. 148, pp. 175–185, July 2018.

[27] F. Baldini, S. Bandyopadhyay, R. Foust, S.-J. Chung, A. Rahmani, J.-P. de la Croix, A. Bacula, C. M. Chilan, and F. Hadaegh, "Fast Motion Planning for Agile Space Systems with Multiple Obstacles," in *AIAA/AAS Astrodynamics Specialist Conference*, AIAA SPACE Forum, American Institute of Aeronautics and Astronautics, Sept. 2016.

[28] R. E. Allen and M. Pavone, "A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance," *Robotics and Autonomous Systems*, vol. 115, pp. 174–193, May 2019.

[29] Shawn B. McCamish and Marcello Romano, "Flight Testing of Multiple-Spacecraft Control on SPHERES During Close-Proximity Operations | Journal of Spacecraft and Rockets," *Journal of Spacecraft and Rockets*, vol. 46, pp. 1202–1213, 2009.

[30] U. Lee and M. Mesbahi, "Feedback control for spacecraft reorientation under attitude constraints via convex potentials," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, pp. 2578–2592, Oct. 2014. Conference Name: IEEE Transactions on Aerospace and Electronic Systems.

[31] P. B. Hansen, "Numerical Solution of Laplace's Equation," technical Report, Syracuse University, 1992.

[32] E. V. Bergmann, S. Croopnick, J. Turkovich, and C. Work, "An Advanced Spacecraft Autopilot Concept," *Journal of Guidance and Control*, vol. 2, no. 3, pp. 161–168, 1979.

[33] B. S. Crawford, *Operation and design of multi-jet space-craft control systems.* Thesis, Massachusetts Institute of Technology, 1969. Accepted: 2005-08-10T21:54:12Z.

[34] C. R. Jakubik, A. Johnston, P. Zhong, N. McHenry, and G. Chamitoff, "SpaceCRAFT VR: an Event-Driven, Modular Simulation Platform with Fully-Asynchronous Physics," in *AIAA Scitech 2021 Forum*, American Institute of Aeronautics and Astronautics, Jan. 2021.

[35] "Unreal Engine | The most powerful real-time 3D creation platform."

[36] M. McCool, *Structured Parallel Programming*. Morgan Kaufmann, July 2012.

[37] Richard H. Battin, Ph.D., "Numerical Integration of Differential Equations," in *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA Education Series, 1999.

[38] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, p. 16, June 2017.

APPENDIX A

TEST-BED SIMULATION SCREENSHOTS

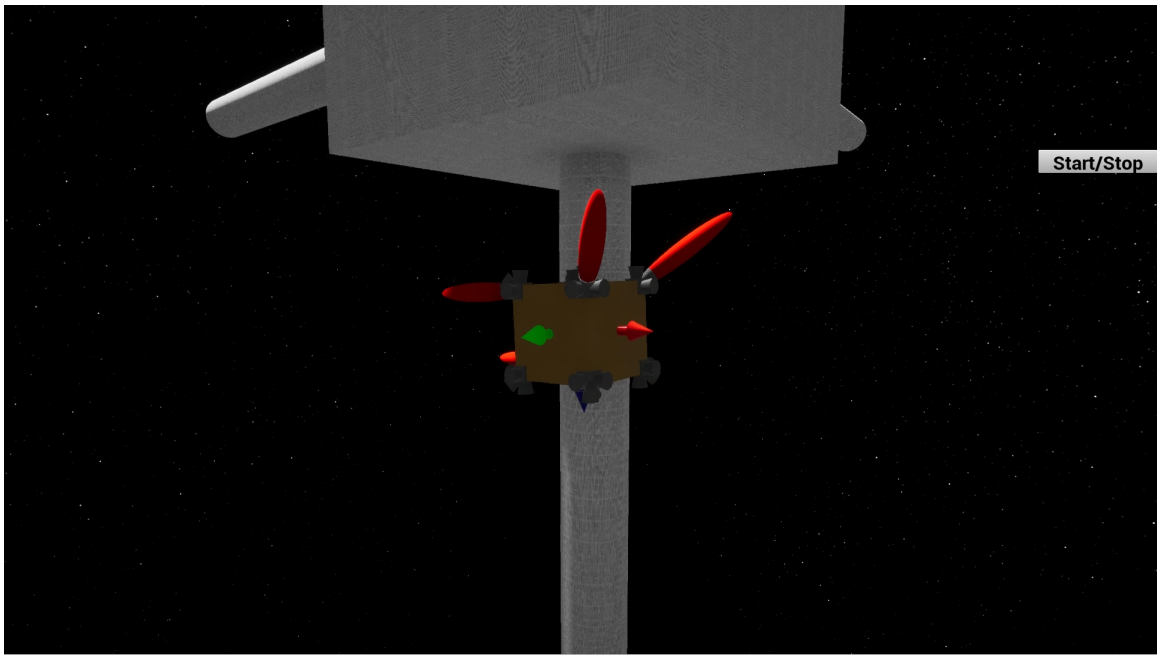

Figure A.1: A screenshot of the test-bed simulation visualization during Test 1

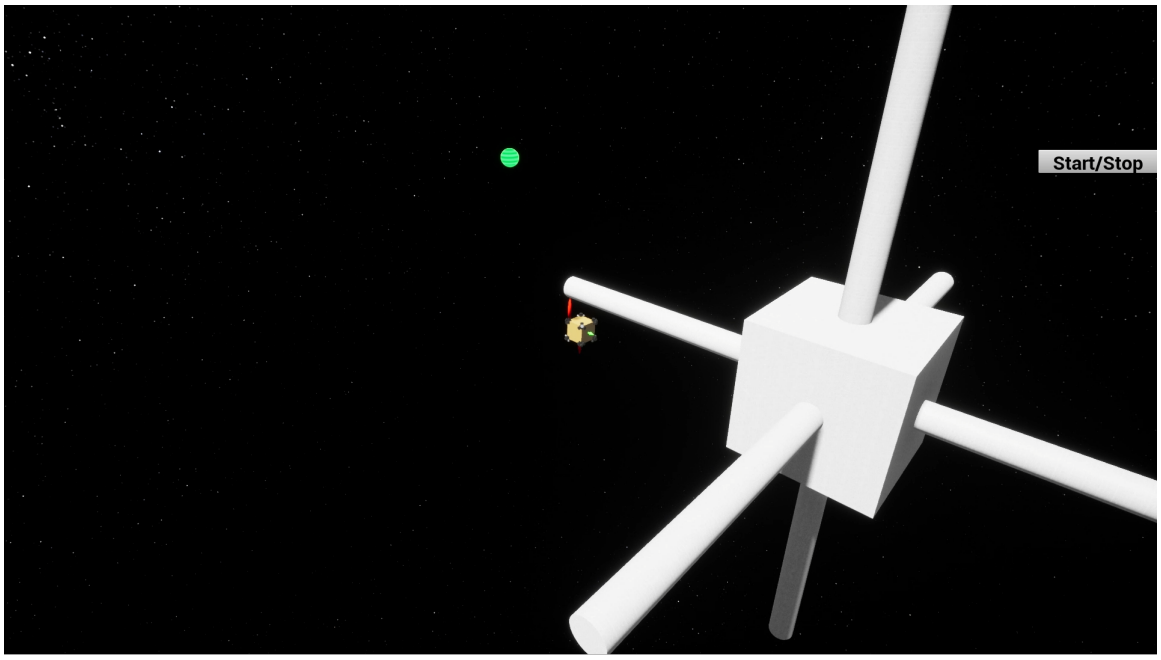

Figure A.2: A screenshot of the test-bed simulation visualization during Test 1

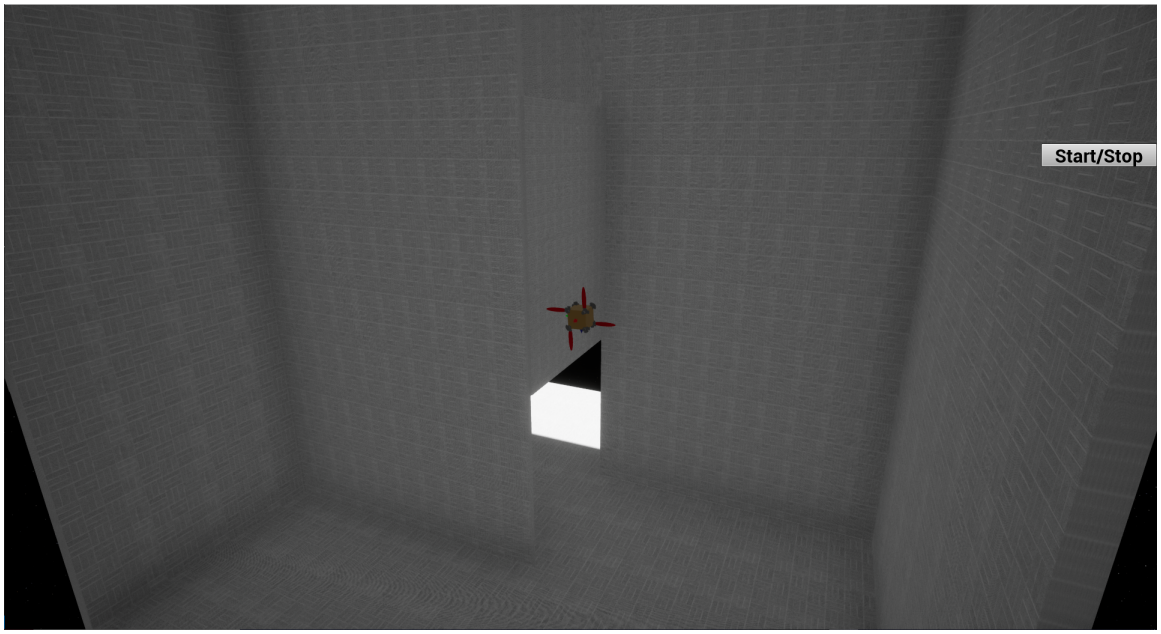

Figure A.3: A screenshot of the test-bed simulation visualization during Test 2

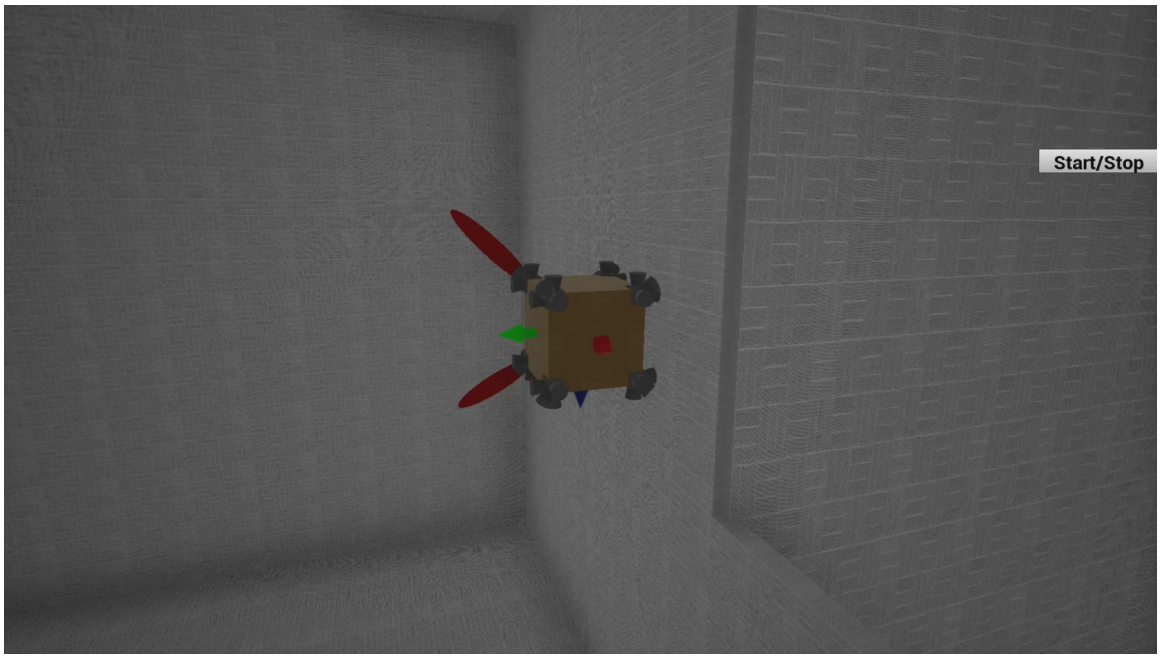

Figure A.4: A screenshot of the test-bed simulation visualization during Test 2

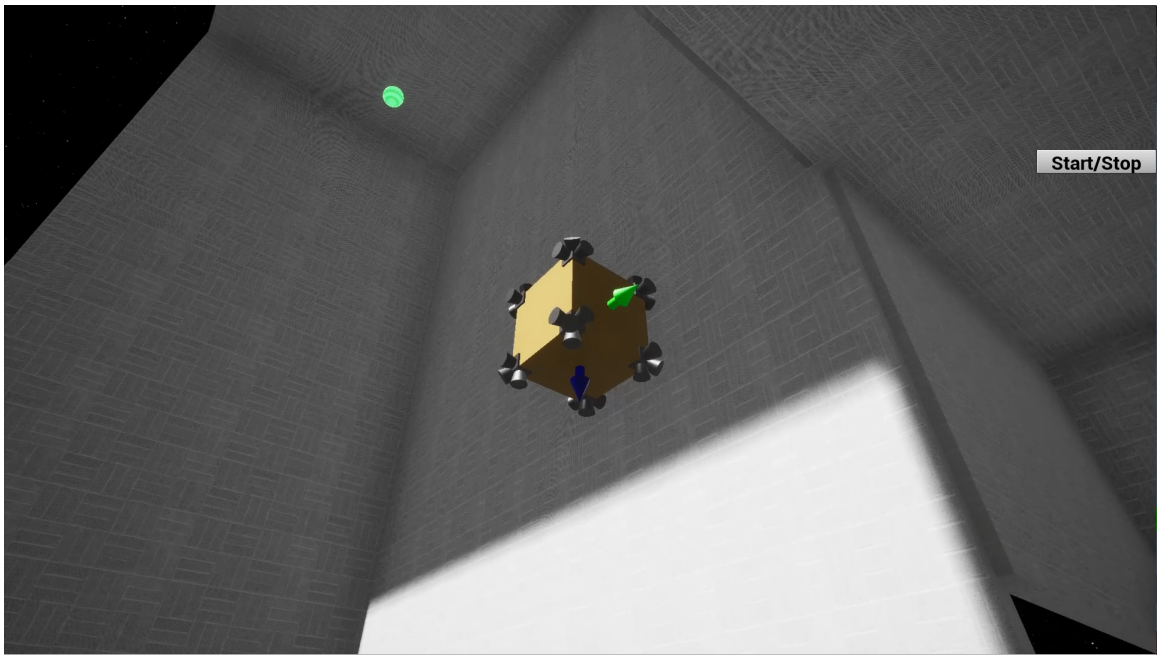

Figure A.5: A screenshot of the test-bed simulation visualization during Test 2

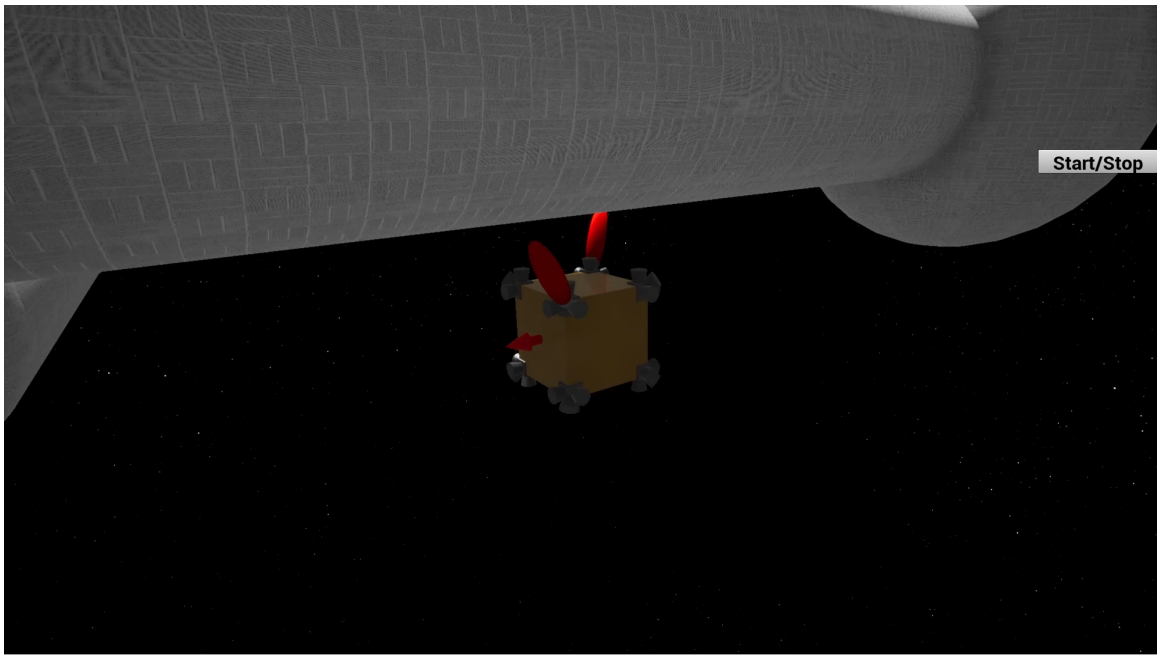

Figure A.6: A screenshot of the test-bed simulation visualization during Test 3

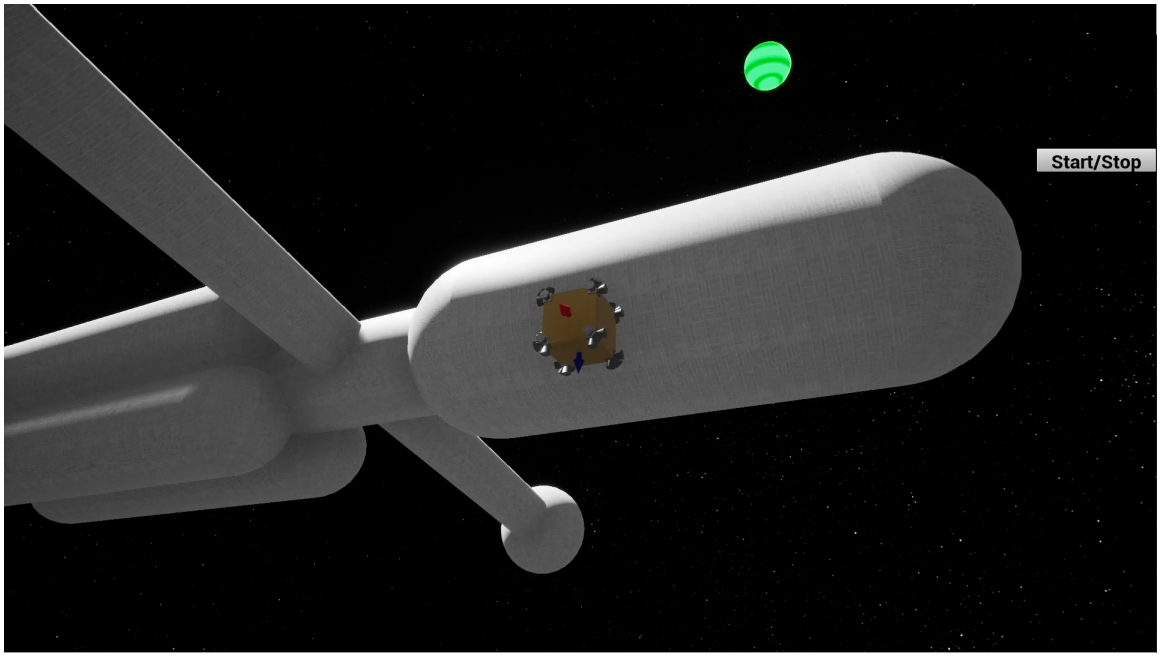

Figure A.7: A screenshot of the test-bed simulation visualization during Test 3

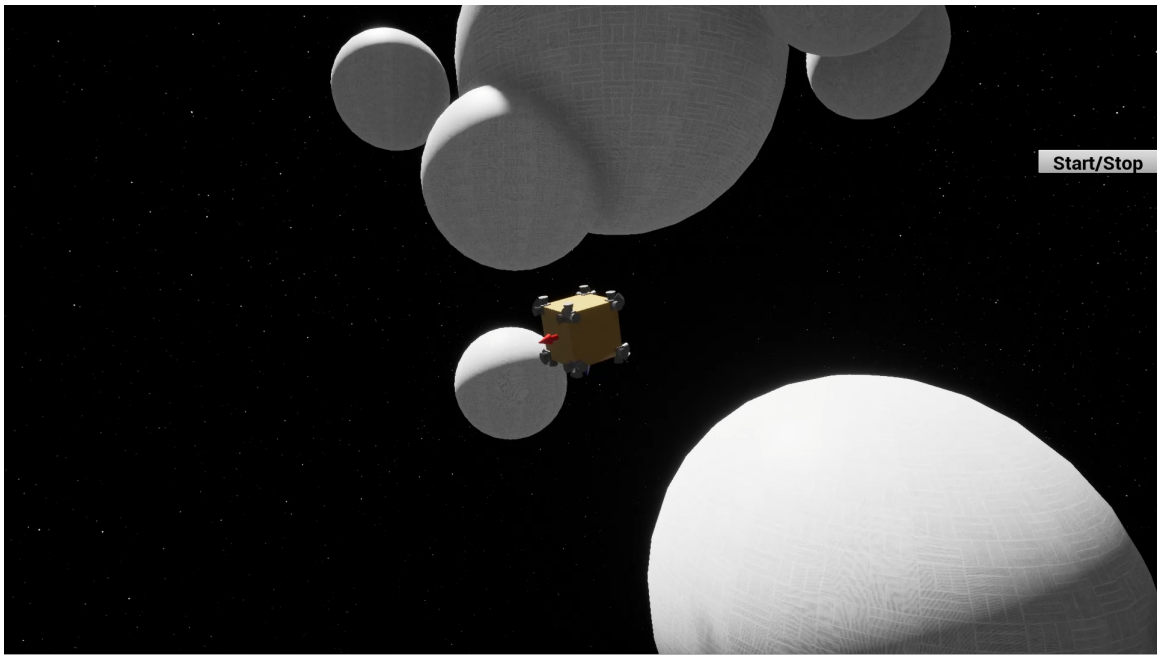

Figure A.8: A screenshot of the test-bed simulation visualization during Test 4

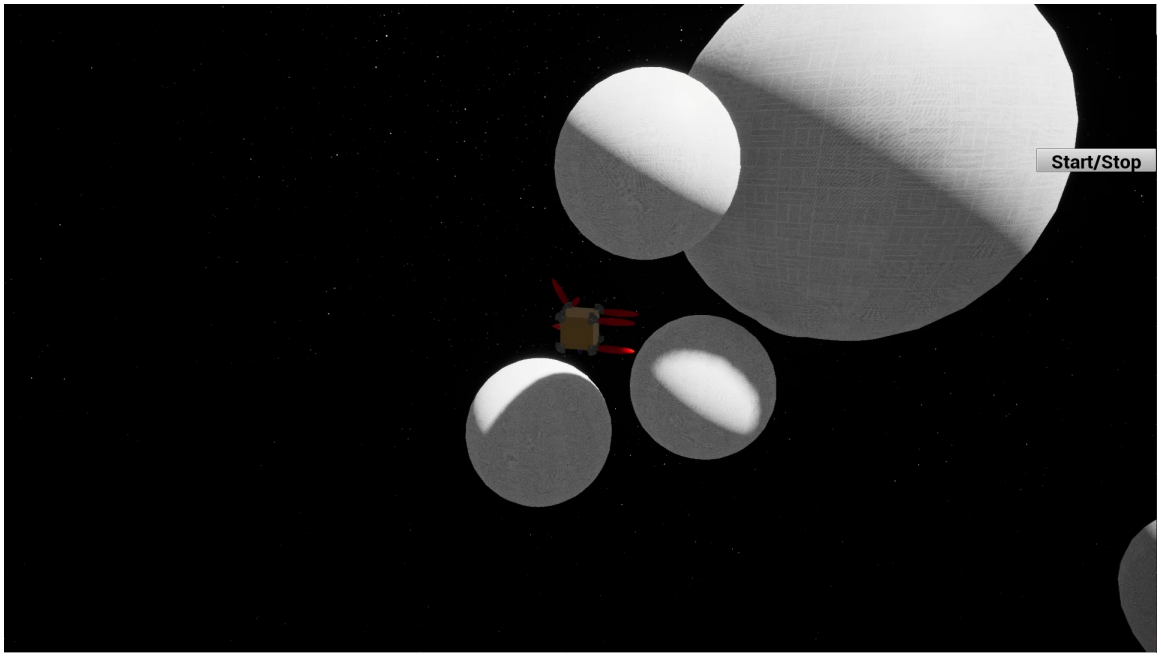

Figure A.9: A screenshot of the test-bed simulation visualization during Test 4

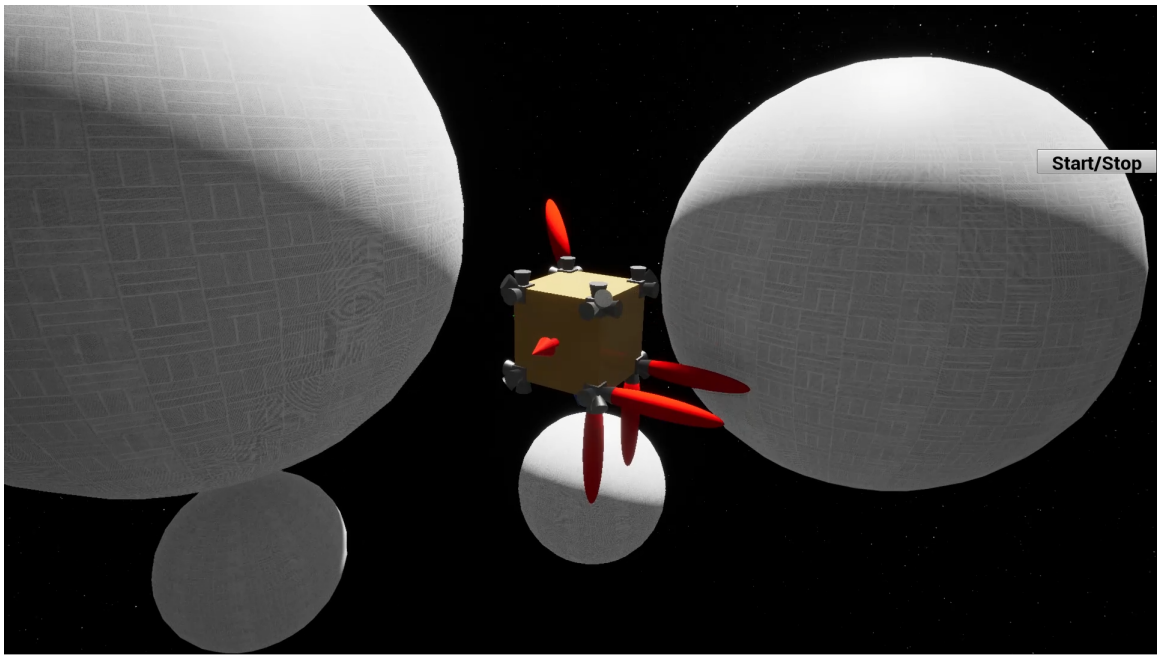

Figure A.10: A screenshot of the test-bed simulation visualization during Test 4

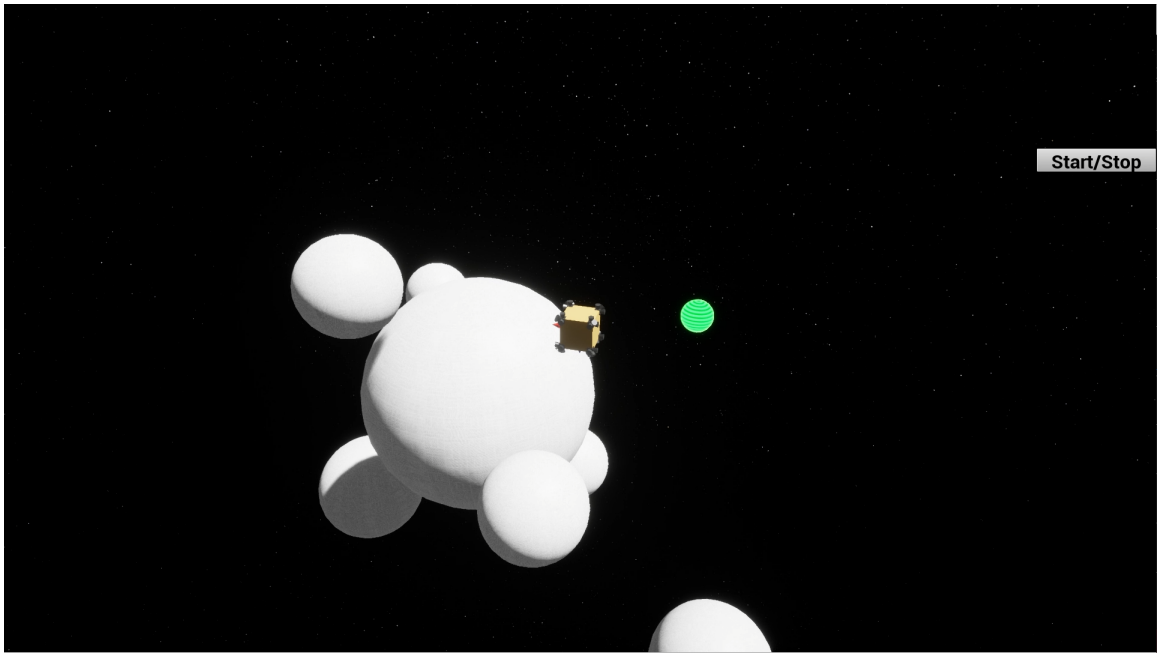

Figure A.11: A screenshot of the test-bed simulation visualization during Test 4