AUTONOMOUS MOTION PLANNING SPACECRAFT GUIDANCE FOR INSPECTION MISSION

A THESIS
SUBMITTED TO THE DEPARTMENT OF AEROSPACE SCIENCE AND
TECHNOLOGY OF POLITECNICO DI MILANO
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE IN SPACE ENGINEERING

Supervisor: Prof. Mauro Massari

Master thesis of Vishnuvardhan Shakthibala
STUDENT ID: 918843

Academic year 2020-2021

Abstract

Looking at the rapid growth of the commercial space sector, it can be predicted that future spacecraft will require On-orbit servicing (OOS) missions to provide the required fuel, power, and repairs to reduce the cost of the mission. The inspection phase forms the first phase of any OOS mission and is the focus of this thesis work. The thesis aims to develop a framework to find autonomous guidance for low thrust propulsion satellites that intends to map the target object. Due to its ability to solve complex problems with reduced computational cost and its flexibility to handle a wide variety of constraints, Sample based motion planning (SBMP) algorithms are adopted as the solution-producing methodology for the thesis. The novel design framework developed works on the principle of the Fast Marching Tree* (FMT*) algorithm. The main novelty of the work includes the identification of adaptations required to apply SBMP algorithms and modeling of translation and attitude trajectories to produce coupled guidance for inspection. Currently, the framework is primarily applied to the Clohessy Wiltshire dynamics model with the local bounded condition. The validity of the framework is verified by applying it to single and multiple satellite systems to inspect targets of different sizes. It was found that the framework finds the solution which provides good coverage and at the same time is best in terms of fuel and attitude control torque. Overall, the work presents a novel way of using the SBMP algorithm to produce sub-optimal low thrust coupled guidance for multiple satellites that intends to inspect a target object. This thesis work acts as an example to showcase the possibilities of using SBMP algorithms to a wide variety of problems present in spacecraft proximity operations.

To my Ma

Acknowledgments

First and foremost, I would like to thank Professor Mauro Massari for accepting to be my supervisor for this thesis work and providing me the freedom to follow my ideas. His responsiveness to student queries and timely reply showed the importance he gives to students amidst the busy schedules at Polimi.

I am greatly indebted to my parents. Nana and Amma, through so many struggles you made sure I followed my interest and passion. My siblings are always there for me Gaethi, Picchu, and Sarthi they are my GPS. Without the support of my family, I would not have been able to come this far. Thanks to everyone back home.

Throughout my master's studies, I came across many wonderful people who were passionate and competent in the things they did. I would like to thank Pietro Di Marche, Vittorio Sartoretto, Vaishnavi Narayanan, for their companionship and support during my studies at Polimi. My hearty thanks to Manon Mauchaussat for her friendship and support. I extend my thanks to Lorenzo Colannino for his support and for the cogitative discussions we had and still are having over many aspects of life.

Special thanks to my colleagues and friends Rashika Suggenahalli Natesh Babu, Suhruth Mourya, and Kevin Charls. Rashika and Suhruth, thank you for helping me out throughout the way, I will not forget Suhruth's cooking and frequent evening talks with Rashika over a cup of coffee. I thank Kevin Charls for his support. He is someone who would say yes to take a break whenever I called him. I liked the long walks filled with discussions covering the simplest to complex of topics.

I owe my gratitude to everyone who has helped me directly or indirectly to reach the point where I am now. Hope this work will help someone in the slightest way.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

"It is the mark of an educated mind
to be able to entertain
a thought without accepting it."
- Aristotle

Leading up to the first lunar landing, the importance of proximity operation for space missions was demonstrated by Gemini missions. Only because of proven proximity operation capabilities acquired from Gemini missions, the first lunar landing was made possible. From providing capabilities such as rendezvous, docking to the idea of servicing the malfunctioning spacecraft, proximity operation forms an integral part of space missions. To illustrate the importance of proximity operation, two different application themes are discussed. One will focus on On-orbit servicing (OOS) and another focuses on proximity Operations around small bodies. Figure 1.1 shows the different proximity operations used in space missions.

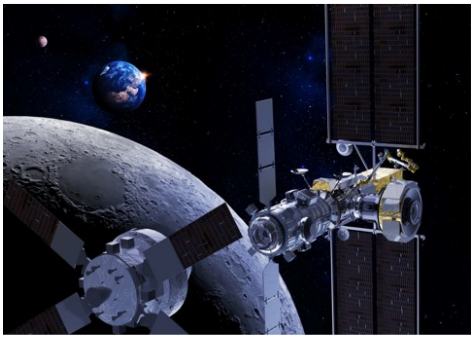
On-orbit Serciving (OOS)

Most of the current spacecrafts are built to stay for fixed amount of time and then it is decommissioned from its operations. The design of the spacecrafts takes into account redundant systems and very long mission life time. This strategies are followed to avoid the risk of losing the spacecraft if it confronts a failure in the components or runs out of fuel. Many satellites perform very well till their operational lifetime but some with small failure in their subsystems enter the inactive phase and deemed inactive[1]. It was estimated that 3 to 4% beginning of life failure occurs which means within 30 days of operation, the satellite fails to operate [1]. The impact of servicing is even more important for satellites which are in GEO-stationary orbit. Because the GEO-stationary space available for satellites to occupy are very limited and are rated high economically. Servicing a satellite in GEO will potentially avoid spending millions of dollars on launching a new satellite. On-orbit servicing aims to provide any means of support to either extend the lifetime of a satellite and relocate it to

provide value added benefit to the mission[31]. Some of the services can be refueling, upgrading, diagnostics, assembling, and repair. The activities involved in OOS broadly categorized into five types [31].

(a) Considered as the first Space rendezvous,this picture shows Gemini VII spacecraft which was taken through the hatch window of the Gemini VI-A spacecraft. Image courtesy NASA, S65-63194

(b) Space Shuttle Atlantis docked with the International Space Station during mission STS-132 in 2010. On the right is ESA's space laboratory Columbus. Image courtesy NASA



(c) Concept for Lunar gateway operation. Image courtesy Thales Alenia Space/Briot

(d) Concept of ESA's De-orbit mission: Aiming at removing large inactive satellites from orbit. Image courtesy ESA

Figure 1.1: Proximity operation mission scenarios

- Inspection

  Inspection activities aims at observing the target object. Inspection activity is accomplished by either satellite module attached to the host or being at a distance (remote surveillance). This activity provides space situational awareness and may be the necessary precursor for other OOS activities. The inspection function includes proximity operation to characterize the phyical state (i.e., position, orientation, and operational status) of an object.

- Relocate

  Modification of the orbit of a space object may be beneficial for constellation ,tactical maneuvering, controlled reentry of Low earth orbit spacecrafts, and rescue of satellites stranded from

a maneuver or due to launch vehicle failure.

- Restore
  Restoring the satellite to its previous operational state will enable satellite to provide greater scientific and economic return. Restore activities include refueling for lifetime extension, providing station keeping support, and providing support to components which failed to operate at the beginning of life.

- Augment
  Augment activity will enhance the capability of the satellite by providing improved hardware that improves the spacecraft performance. A famous example for this activity is the correction of faulty primary mirror of Hubble space telescope.

- Assemble
  Assemble activity aims at building a space system made of small modular systems. Once built the larger space system will have higher capability. ISS is one of the example of this activity. ISS cannot be launched in one launch vehicle, as a result the modular construction was followed to built a largest space structure ever.

With the OOS capabilities, the space missions can become cost effective and it will lead to development of advanced mission architectures. Some of direct advantages of OOS missions [31][15] are (a) Reduce Risk of Mission failure (b) Reduce Mission cost (c) Increased mission performace (d) Improve mission flexibility (e) Enable New missions.
Currently, there is no fully developed OOS spacecrafts available. But the interest towards adapting OOS is becoming real. Many of the Government and Private industries are concentraing on developing the capabilities to OOS. Table 1.1 lists some of the companies[15] who are working on OOS capabilities.

Figure 1.2d and 1.2b shows the NASA's RAVEN and Upcoming OSAM-1 (On-Orbit Servicing, Assembly, and Manufacturing 1) respectively. OSAM-1 will be an important mission which will demonstrate several important capabilities of OOS.

As evident from Table 1.1, companies are trying to develop autonomously OOS capabilities. Humans in the loop servicing missions are already well established dating back to Gemini missions[19]. Servicing of Hubble space telescope using Space shuttle and continuous supply of resources and maintenance of ISS shows that this technology is mature enough. But the drawback of this will be the cost of manned missions. Reasons for stopping the space shuttle program is argued to be the cost of maintenance of the whole program and major risk involved with humans survivability. Another draw back for human in the loop OOS missions is that the communication delays restrict the direct control. For example, spacecraft entering the mars atmosphere cannot wait for signals from earth to control its attitude because the two way communications will requires approxmitaly

Table 1.1: Companies focusing on OOS

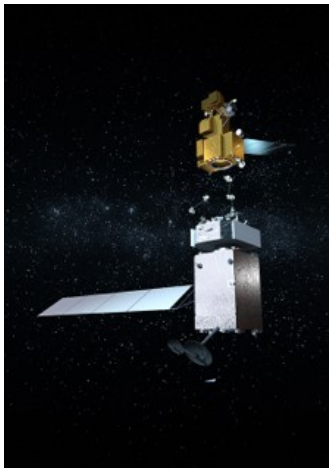
Company/Agency : Mission name	Sector	Aim of the mission
Airbus : O.CUBED	Commercial	LEO and GEO Inspection, life extension, orbit modification, upgrade, and debris removal.
Astroscale : ELSA-d	Commercial	Active debris removal
Busek: SOUL	Commercial	Tethered to the host, performs inspection, repair, debris removal
NASA: RRM and RAVEN	Government	Demontrating on-orbit refueling and robotic vision technology.
DARPA: Orbital Express	Government	Autonomous rendevous,docking,refuel ,and upgrade.
iBOSS: iBOSS	Commercial	Modularity in the design and manufacture reconfigurable spacecraft for on orbit servicing and assembly.
NASA: Lunar Gateway	Government	Next Space station will be assembled in Lunar orbit
NASA: OSAM -1	Government	A robotic spacecraft equipped with the tools, technologies and techniques to rendezvous with, grasp,refuel and relocate a government-owned satellite to extend its life

40 minutes and the whole entry sequence would be done by the time the control seqeunce reaches the spacecraft. These limitations influence the autonomy in the OOS missions.

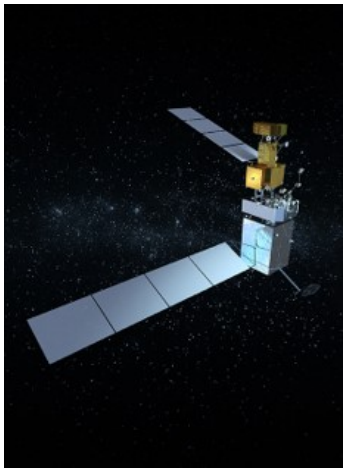
To conduct OOS missions, the problem that needs to be solved may be considered to be composed of two problem categories as identifed in [51]. They are Autonomous Rendezvous and Docking (ARD) and Autonomous Inspection and Servicing (AIS). The solution to these two problems will provide the capability to efficienty conduct OOS operation. Now, next application theme of proximity operation is discussed.

Proximity operations around small bodies

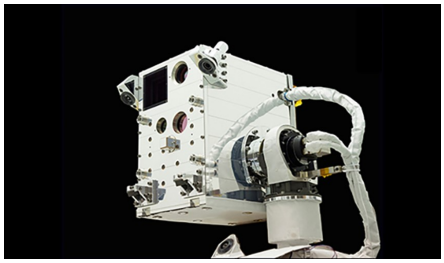
Proximity Operations near small bodies are difficult operations due to large uncertainy involved in the physical parameter[48] and the chaotic dynamics due to the presence of forces such as solar radiation pressure, non-uniform gravitation field, etc. Because of these reasons the spacecraft near the asteroids can become unstable depending on how it is positioned. It is trivial that due to these forces, the dynamics around the small bodies deviate largely from keplerian dynamics. These uncertainties are compensated only if the spacecraft has enough sensitivity and control authority available at disposal. The mission design for small body missions changes drastically from one to another due to small body's varied nature of physical properties such as shape, size, gravitation field, and spin rate, etc. NEAR shoemaker[44][57] missions which was sent to Asteroid Eros followed a orbital trajectory to rendezvous around it. Similarly recent asteroid mission OSIRIS-REX [27] followed orbital trajectory approach to charcaterize the trajectory. Whereas both Hayabusa and Hayabusa 2[61][60] missions followed a hovering type rendezvous around ITOKAWA and RYUGU asteroids respectively. Most of the missions to asteroids aims to land on the surface of the asteroid after rendezvous phase. Rendezvous phase is essentially a Inspection phase where the asteroid is



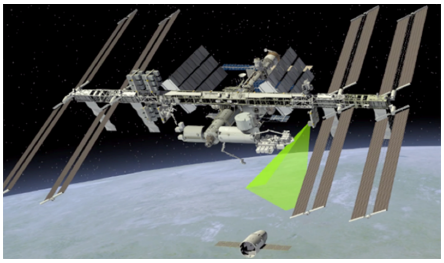
(a) Artist's concept of OSAM-1



(b) OSAM-1 servicing concept



(c) RAVEN MODULE on ISS



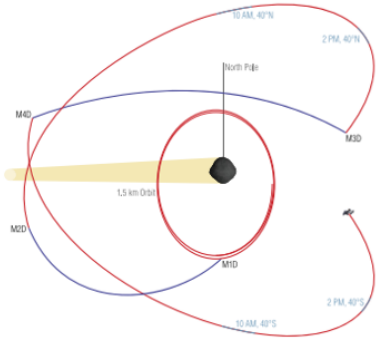
(d) RAVEN Using COMPUTER VISION

Figure 1.2: OOS Demonstration missions

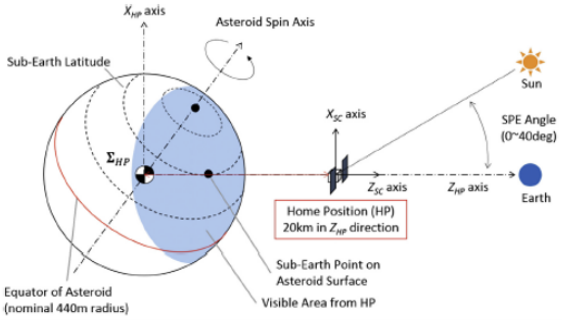
characterized with enough details. Once characterized, the spacecraft moves closer, step by step, and finally attempts to land on the surface. Figure 1.3 shows two different rendezvous approaches followed by HAYABUSA 2 and OSIRIS REX mission. The application theme that is discussed here is categorized as Proximity Operations for primitive bodies (POPD). Now that both application themes are briefly presented, next, a general formulation for guidance problem for proximity operation is described [51].

The General Autonomous Guidance optimal control problem is given by

$$\begin{aligned} \min_{t_f,u} \quad & g_b\left(x\left(t_f\right)\right) + \int_{t_0}^{t_f} g_i(x(t),u(t))dt \\ & \text{subject to the following:} \forall t \in \left[t_0, t_f\right] \\ & \dot{x}(t) = f(t,x(t),u(t)) \\ & u(t) \in \mathcal{U}(t) \\ & x(t) \in \mathcal{X}(t) \end{aligned} \tag{1.1}$$



(a) Osiris Rex proximity operation[27]



(b) Hayabusa 2 proximity operation[65]

Figure 1.3: Small body approach and inspection strategies

where $x \in \mathbb{R}^n$ is the state of the spacecraft, $u \in \mathbb{R}^m$ is the control input, t is time, $f : \mathbb{R}^{n+m+1} \to \mathbb{R}^n$ defines the dynamics, and $\mathcal{U} : \mathbb{R} \to \mathbb{R}^m$ and $\mathcal{X} : \mathbb{R} \to \mathbb{R}^n$ are set-valued maps defining spacecraft control and state constraints. $g_b\left(x\left(t_f\right)\right)$ represents the boundary cost at t_f and $\int_{t_0}^{t_f} g_i(x(t), u(t))dt$ defines the cost along the trajectory $x(t)$. Equation 1.1 can modified to suit for the problem at hand and later in section 1.3, the thesis problem is represented in the similar structure.
A brief description of the state of the art methods used in autonomous spacecraft maneuvering is presented below. These topics are summarized from [51].

1.1 State of the art techniques and current trend

In this section, main motivations which inspired to use some of the ideas in the thesis work is presented. These motivations are inspired by studying the developed state of the art approaches for guidance problem and current trend in space missions.

1.1.1 State of the art approaches

Apollo Guidance

The initial successful autonomous spacecraft guidance was achieved using a program developed by MIT called The COLOSSUS Program [62] for NASA's Apollo program. The COLOSSUS Program included three Digital Auto Pilot systems(DAPs) which were responsible to stabilize and control the Apollo Command Service Module (CSM) as part of its Primary Guidance Navigation and Control system (PGNCS). PGNCS worked on principles that are part of classical control systems. Three DAPs were used for different operations. Orbital Re-entry Digital Autopilot was used to control the entire re-entry sequence. Reaction-Control System Digital Autopilot provided attitude control using Reaction-Control System (RCS) thrusters. Thrust-Vector-Control Digital Autopilot controlled the Command Service Module (CSM) during powered flights.

Model Predicive control (MPC):

Model Predictive control is the feedback law based method to solve an Optimal Conrol Problem (OCP). MPC uses the current state to predict the future states via dynamics for particular amount of time called planning period or time horizon. For an OCP , the cost functional is minimized over this intial states and future predicated states with control inputs for the time horizon. Once the OCP is solved for this time horizon, only the initial control segment is applied to the system and obtained state is considered as the new current step for the next time-horizon iteration of MPC. MPC is also called receding horizon optimal control or moving horizon optimal control because of its repeated application of OCP solutions over finite time-horizon. The receding horizon concept can be suitable for any open-loop optimal control based approach[10] and can be added in the loop as feed back controller. This provides robustness and ability to handle disturbances and mitigate error growth. Also MPC can provide closed loop stability and state convergence in an environment where there is no prior knowledge of disturbance[34]

Artifical Potential Functions

As the name indicates, this methods aims at representing the state space of the vehicle using mathematical potential functions. These potential functions can be of repulsive or attractive type. Goal regions are linked with attractive potentials and obstacles regions are linked with repulsive potentials. A gradient descent/ascent method is often used to trace a feasible path from any initial state with suitable tuning can safely circumnavigate the obstacles and converge to one of the goals. On top of this, an OCP may also be formed to plan a path that minimizes certain cost functional over this gradient force field. The potential functions can be tuned as the surronding environment changes and this is one of the advantages of APF. The main drawback is associated with susceptibility of the method to converge to local minima. Which cannot be avoided without additional hueristic techniques. To get more information on this approach refer to [6][35].

Spacecraft Motion planning

Motion planning(MP) is a class of algorithms used to generate sequence of decisions, called plans,

that safely take a robot or group of robots from a set of initial states to set of desired final states. Here, robot can mean any object for which motion planing algorithms are applied. Applications of Motion planning algorithms range from rovers, complex pharmaceutical molecules to artificial Intelligent agents, etc. Broadly Motion planning algorithms are classified into two broad categories, Exact algorithms and Approximate algorithms. Exact algorithms tries to build the exact configurations space of the robot as well as the environment and has the advantage of completeness to find the solution. but Exact Algorithms cannot be applied for higher dimensions because of computations limitations that comes from construction of configuration space. That is where Approximite algorithm also known as sampled based motion planning (SBMP) algorithms comes to rescue. SBMP algorithms avoid explicit building of the configuration space and considers the collision detection as black box. Augmented with the idea of sampling the configuration space, allows the SBMP algorithms to find solutions faster than Exact Algorithms. This procedure makes the SBMP algorithms independent of the geometric models. The main drawbacks involved in SBMP is the weaker notions of correctness and completeness which says the existence of solutions cannot be guaranteed in finite time without drawing an infinite set of samples. Some of the well known SBMP algorithms include Rapidly-exploring random Tree(RRT)[29], Probabilistic roadmaps(PRM)[23], Fast marching Tree[21] and Ariadne's Clew Algorithm[28]. Many Algorithms are providing a notion of weaker optimality condition called Aymptotic optimality which informs that the algorithm guarantees convergence to optimal soltuion as the number of samples goes to infinity. The applications of the motion planning algorithms was seen in DARPA Urban challenge where the the goal was to autonomously guide a vehicle to the goal point[26][22][56]. Many participating teams used Motion planning Algorithms for autonomous guidance. This shows the application of MPs in real-time cluttered environment. Even though, the SBMP Algorithms are not used in space missions yet, the advantages of using it are studied and advantages of using it are proposed in [52][16]. Since the SBMP algorithms are showing the signs of being used in space missions, the advantages of this algorithms have inspired to use it in this thesis work.

Motivation 1. Adapting the Motion planning Algorithm to solve Optimal Guidance for proximity operation.

1.1.2 Current trend in space missions

Another important technological development that is changing the way the space mission works is low thrust propulsion technology. Low thrust propulsion is a class of propulsion which has higher specific impulse with relatively low thrust, this leads to drastically reduced fuel requirement as compared to chemical propulsion. For the satellites class belonging to Small sat to CubeSat[42]. The low thrust technology provides a tangible way of acquiring capabilities only available to larget satellites. Those capabilities include orbit raising, station keeping, collision avoidance, and deorbiting, etc. Low thrust propulsion is not only dedicated for Near Earth missions, it has been

used in many Interplanetary missions such as Hayabusa and Hayabusa 2[41], and Bepi-colombo missions[11], etc. Use of low thrust propulsion provides huge benefits in terms of weight and leads to increased ability to carry large payloads. When it comes to small to CubeSat category satellites, the mass and volume constraint may not allow it carry enough amount of chemical propellent for high thrust propulsion. This is where low thrust propulsion can be a very suitable candidate. Currently there is over 2500 (October 2020)[42]active small satellites in the orbit and in next five years, it is expected to grow larger as miniaturization is producing benefits at relatively lower cost. Even for Interplanetary missions, space agencies around the world are planning to make use of Cubesats. Marcos was the first ever CubeSat to take part in deep space communication, it relayed the telemetry data when insight was making its entry into the mars atmosphere [40]. LICIACube (Light Italian CubeSat for Imaging of Asteroids)[9] another CubeSat class satellite which is planned to be used for DART mission: a demonstration mission aimed at demonstrating planetary defence strategy[12]. With capabilities of small satellites technologies rapidly growing, if this satellites are augmented with low thrust propulsion system, the combined capabilities will enravel new architecture designs and possibilities for space missions. From above discussions of CubeSat and low thrust propulsion technology following motivations are inspired for the thesis work.

Motivation 2. To include low thrust trajectories inside the guidance for proximity operation.

Motivation 3. Utilize the Smallsat to CubeSat range satellite class as a spacecraft model.

With above motivations, the goal of thesis is described below followed by the various solutions found in literature to solve it.

1.2 Thesis problem definition

In this thesis work, the concentration is on the first phase of the proximity operation called as the Inspection phase. Inspection phase plays an important role in characterizing the target object and becomes a crucial step moving further in the next stages of proximity operation[9][15].

Problem Definition

Inspection problem requires the guidance law to provide maximum coverage of the target with best possible translation and rotational motion. The aim of the thesis is to solve this problem by using the Motion planning algorithms with low thrust propulsion and Small to CubeSat category satellite.

The problem of finding the guidance for the proximity operations are widely studied in the literature. Some of the studies which tried to solve the guidance problem relevant to Inspection mission are described below.

1.2.1 Literature survey

Recently, Starek et al.,[50] showed the efficiency of motion planning (MP) algorithms specifically (Fast marching tree star)FMT* and BFMT* (Bi-directional fast marching tree star) in finding the optimal fuel cost for a spacecraft to go from one point to another desired point around the target object. The framework was applied only to high impulse trajectories. Only translation motion was considered in the framework. Although, the rotational trajectories are not taken into account the framework provides the motivation for using the motion planning algorithms for general guidance problems involving high thrust trajectories. Another important aspect was that MP was able to handle wide variety of constraints which may not be feasible with other solution methods. Bernard et al.,[4] focused on solving the Inspection problem for swarm of satellites. Again here, the high impulse trajectories were considered. And also, the framework made use of passive relative orbits and the trajectories were pre-computed and selection was done based on hueristics. For the rotational motion, dynamic programming is applied to move from one pointing angle to another pointing without considering the explicit attitude dynamics. Teja Nallapu et al., [55] considers swarm of satellites for inspection mission. Specifically authors tried to develop a holistic framework to build translation trajectories and attitude trajectories to map small bodies. Framework only considers high-impulse transfer and attitude motion is not fully utilized(the rotation motion was prescribed in certain way instead of taking into account the full attitude dynamics). Framework assumes circular ring translation trajectory and the satellites are optimally assigned to different position on the ring by using Evolutionary Algorithm. Surovik et al.,[53] proposes a idea where the goals of the mission are converted into reachability sets. By using the hueristics, the optimal delta Vs are found finally leading to optimal solution which satisfies the mission goal. This paper works with complex dynamics of the asteroid and provides a efficient solution to satisfy the mission goals. Even here, high impulse trajectories are considered and author remarks that application of the proposed method will be very costly to apply for low thrust trajectories. Capulop et al.,[10] aims to find solution to Inspection problem by using sample based motion planning algorithm. High impulse trajectories are considered and uses the hueristics idea proposed in [53] to build a reachability set which links the delta Vs and mission goal. The attitude motion is not analyzed explicitly but it is taken into account by using a simple angular rate constraints. This paper tries to include the uncertainties involved in relative position during Inspection mission to analyze the safety of inspecting satellite. The framework uses receding horizon feedback law combined with SBMP to find solutions and has the capability to counter unmodelled perturbation due to the receding horizon method. Maestrini et al.,[33] based their solution on both [53][10]. They utilize the hueristics and receding horizon method. One of the key addition is the low thrust trajectories. The rotational motion is not treated explicitly but the division of operation phases are done to provide flexibility to the framework. The results obtained are compared with a standard relative football orbits and found to have relatively good performance.

Zhou, Ding, et al.,[66] used the principles of PRM*(Probabilistic roadmaps*) and RRT*(Rapidly-exploring random Tree*) sample based motion planning algorithms in a balanced way to build both translation and rotational trajectories for proximity operation. High impulsive trajectories was considered for translation motion and for rotational motion, a piecewise constant control is assumed to take the attitude from one attitude state to another attitude state. Deterministic sampling was used. This paper also included multiple obstacles, control constraints and FOV constraint. In the paper authors succeed at finding coupled guidance for translation motion and rotational motion, coupling occurring via sensor FOV (field of view) constraint and trajectory time stamp constraints. Low thrust trajectories were not considered and assumption of constant piece-wise control input may have limited how the full attitude dynamics might have behaved for vast variety of other controls.

With the knowledge coming from the past papers, the thesis tries to leverage some of the concepts from these papers and tries to develop novel a framework to solve the Inspection problem.

1.3 Thesis problem statement

Directly from the thesis problem definition, the mathematical formulation can be written. Two types of problems are identified from the problem definition, first one is associated with Inspection using single satellite and second one is Inspection using multiple satellite. Depending on the application needs, either of these cases can be applicable.

- Problem A - Single satellite coupled guidance for Inspection Mission.
  This problem formulation uses single satellite and tries to find the best feasible translation and attitude trajectories which gives good coverage.

$$
\begin{aligned}
\underset{\mathcal{U}}{\text{Min}} \quad & \mathbf{J}(x(t), u(t), t) = J_m + J_\tau + J_{coverage} \\
& \text{Subject to} \quad \text{System dynamics} \quad \dot{x} = f(x(t), u(t), t) \\
& \text{Control feasibility} \quad u(t) \in U(t) \quad \forall \quad t \in [t_0, t_f] \\
& \text{Trajectory feasibility} \quad x(t) \in X(t) \quad \forall \quad t \in [t_0, t_f] \\
& \text{Time feasibility} \quad t_f \in T
\end{aligned}
\tag{1.2}
$$

  Where, $\mathbf{J}(x(t), u(t), t)$ is the total objective functional made up of three different objective functionals. J_m corresponds to the fuel cost (equation 4.14), J_τ corresponds to control torque cost coming from attitude actuator (equation 4.25) and finally $J_{coverage}$ is the cost related to compliment of coverage called coverage compliment (equation 4.26). These three objective functionals shall be minimized to obtain best solution for the Inspection problem. Optimization is controlled by the control action u(t) and it is carried out by satisfying the constraints coming from system dynamics, control feasibility, trajectory feasibility and time feasibility (see section 4.2.1).

- Problem B - Multiple satellite coupled guidance for Inspection Missions.

$$\begin{aligned} \underset{N,\mathcal{U}}{\text{Min}} \quad & \mathbf{J}(x(t),u(t),t) = \sum_{i=1}^{N} J_{m_i} + J_{\tau_i} + J_{coverage_i} \\ & \text{Subject to} \quad \text{System dynamics} \quad \dot{x} = f(x(t),u(t),t) \\ & \text{Control feasibility} \quad u(t) \in U(t) \quad \forall \quad t \in [t_0, t_f] \\ & \text{Trajectory feasibility} \quad x(t) \in X(t) \quad \forall \quad t \in [t_0, t_f] \\ & \text{Time feasibility} \quad t_f \in T \end{aligned} \tag{1.3}$$

  This problem is similar to Problem A. But, the difference is the addition of parameter N, which stands for number of satellites. Problem B basically optimizes the Problem A for each satellites $i \in N$. The difficulty involved is that the possibility of collision and pointing interference. The usefulness of solving this problem over Problem A will be analyzed and is one of the goals of the thesis.

1.4 Assumptions and Terminologies

Following are the assumptions considered for this work. These assumptions are used throughout the thesis unless otherwise specified.

- The thesis work concentrates only on the open loop guidance part of the Inspection phase. Relative navigation is assumed to be a black box and not considered in this work.
- Mission scenario is assumed to require a separation of thrusting phase and observation i.e., both thrusting and observing phases will not occur simultaneously. This assumption is valid as, the thesis concentrates on using satellites of Small satellite to Cubesat category. This assumption drives the designing of translation segment (see section 4.1)
- Throughout the thesis, the satellite which will do observation are called as **Servicer** and object which is under observation are called **Target**.
- Throughout the thesis, the term "Inspection problem" is used, this stands for the definition and statement mentioned in section 1.2 and 1.3 respectively. Inspection problem and Inspection missions are assumed to be synonyms.
- Target is assumed to be a general object which may or may not have co-operative characteristics with the Servicer. This assumption informs that the framework developed is independent of the co-cooperativeness of the target.
- Disturbances are not accounted in the work, for both translation and rotation motion.

- Electric propulsion (EP) system design and attitude actuators internal dynamics are assumed to be black box. The outputs: Thrust from EP and Torque from attitude actuators are directly used.
- The mass of the spacecraft is assumed to be constant even though it is losing its fuel via low thrust propulsion system. This is reflected in the fuel cost equation 4.14, where m_0 mass of the satellite is assumed to remain constant.
- In the thesis, the term 'coupled guidance' is used many times, this means that the guidance law is produced considers both translation and rotations motion with their the respective constraints. Coupled guidance does not indicate coupling of translation and rotational dynamics.

1.5 Thesis contribution

- Develops a novel framework which uses sample based motion planning algorithm to find coupled guidance for Inspection Problem.
- The framework finds holistic guidance for inspection mission by coupling translation and attitude trajectories.
- The developed framework can handle inspection missions involving both single satellite and multiple satellites.
- Various adaptations required to suitably apply SMBP algorithm to the inspection problem is identified.

1.6 Thesis organization

Thesis is organized as follows (figure 1.4):

- Chapter 2
  Introduces the basic concepts of sample based motion planning. Concepts include general properties of SBMP, general building blocks of SBMP. Finally FMT^* is presented followed by the adaptation that needs be done on SBMP algorithms to use it for Inspection problem.
- Chapter 3
  Briefly describes various dynamics involved in the proximity operation. Brief description of reference frames and relative dynamics associated with translation and rotation are presented.
- Chapter 4
  This chapter presents the core part of thesis work. Chapter starts with the overall guidance

framework named as Autonomous Motion planning Spacecraft Guidance for Inspection Mission (AMPSGIM) and goes on to discuss every key block involved. Along with schematic description, the pseudo code of the framework is included.

- Chapter 5
  Chapter mainly concentrates on demonstrating the application of developed frame work to mission scenarios. Target of various size are considered. validity of the framework for inspection mission using single satellite and multiple satellite missions are presented. Results obtained are discussed and analyzed.

- Chapter 6
  Major limitations of the framework is discussed here. Followed by further development section where the improvements and extensions required to enhance the reliability and performance of framework is discussed.

- Chapter 7
  Here, the final conclusion of the thesis is presented which summarizes the overall work.

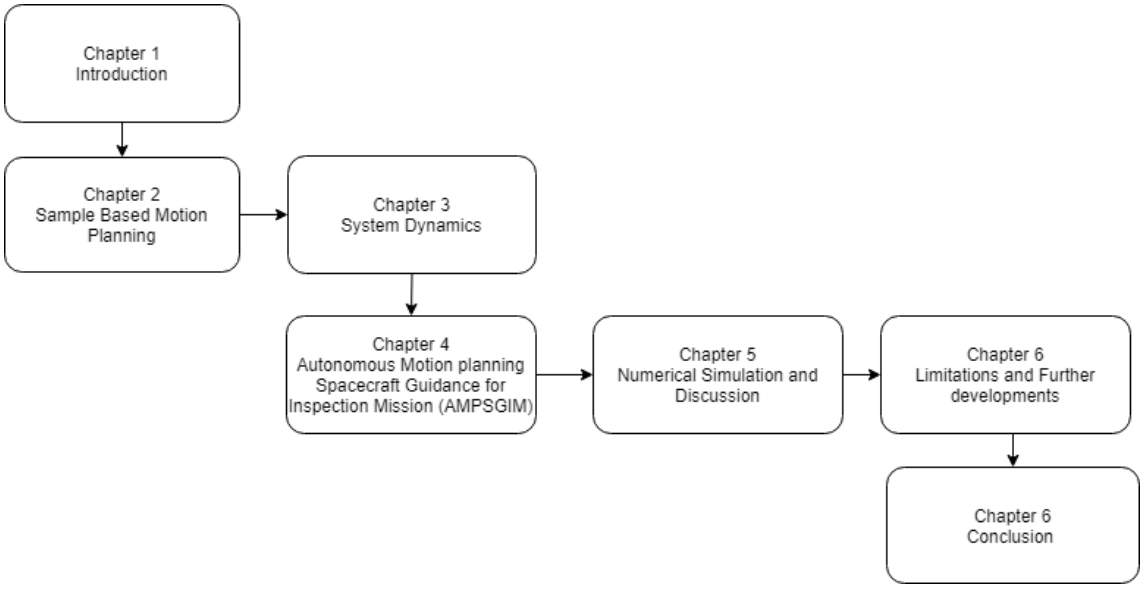


Figure 1.4: Thesis organization

Chapter 2

Sample Based Motion Planning

"Sometimes you climb out of bed in the morning and you think,
I'm not going to make it,
but you laugh inside — remembering
all the times you've felt that way."
- Charles Bukowski

In this chapter, brief background on motion planning (MP) and its characteristics are presented. Starting with basic form of motion planning and building towards sampling based motion planning (SBMP). The chapter ends with description of FMT* the core algorithm selected for the thesis work and adaptations that are needed to use SBMP algorithms to solve inspection problem. Major part of this chapter is based on the book "Planning algorithms" [28].

2.1 Basic form of motion planning

In robotics, the basic form of motion planning problem involves trying to find a path connecting initial position to desired final position without colliding with obstacles. This problem is described as geometric motion planning as it deals only with finding the best geometric path as in figure 2.1. Such problems can be extended to include uncertainties and limitation coming from system dynamics. Which then transforms into a class of problems called Motion planning under differential constraints. Before moving on to fundamental form of motion planning, following terminologies are needed.

- Workspace $\mathcal{W}$: This is the space which includes all the possible configuration of the problem. For example, $\mathcal{W} = \mathbb{R}^2$ represents the 2-D space.

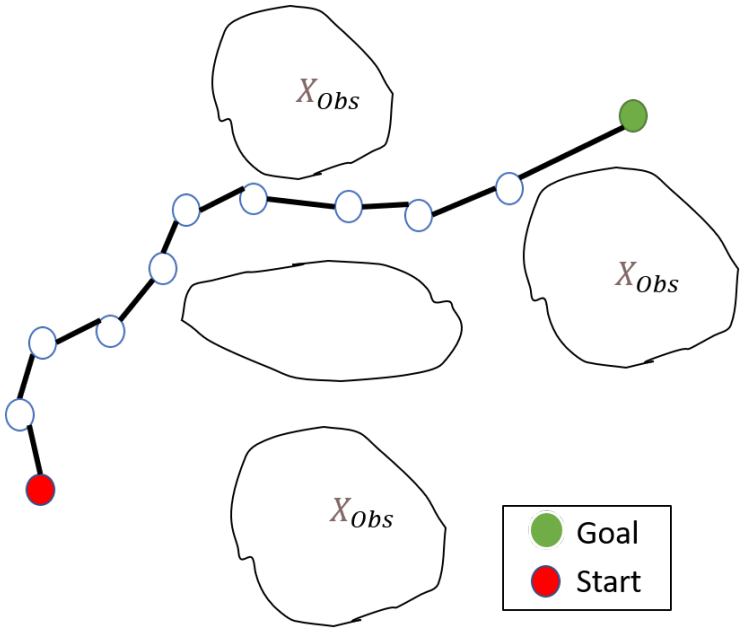


Figure 2.1: Geometric motion planning

- Configuration Space or C-Space: Let A be a vehicle and its configuration at instant t is given by q. C-Space is the space where all the possible configuration of A will be present. The dimension of the C-space is equivalent to the number of degree of freedom of the system.
- Obstacle C-space : Let O represent the obstacle region in W. let $\mathbb{B}(q)$ represents the set of points occupied by configuration q. Then Obstacle C-Space is given by

$$\mathcal{C}_{obs} = \left\{ q \in \mathcal{C} | \ \mathbb{B}(q) \cap \mathcal{O} \neq \emptyset \right\} \tag{2.1}$$

- Free Space
  Free space is the space available after intersection of C-Space C and Obstacle Space C_{obs}.

$$\mathbb{C}_{free} = C / C_{obs}$$

With this notion in mind, Geometric Motion planning problem is given by:
Find a path $\mathbb{X} : [0, 1] \rightarrow C$, such that $\mathbb{X}(0) = q_0$ and $\mathbb{X}(1) = q_G$. Given

1. A Workspace W is defined. It can be either $W = \mathcal{R}^2$ or $W = \mathcal{R}^3$.
2. An Obstacle region $\mathbb{O} \subset W$ is known.
3. A vehicle is represented in W.

4. The Configuration Space C is defined along with C_{free} and C_{Obs}.
5. Initial configuration q_I and goal configuration q_G is defined.

To solve the above problem, we need to construct the configuration space as well as the obstacle region which is considered to be computationally costly process. Even for the smaller dimensions, building the entire configuration space becomes difficult. On top of this issue if the geometric problem involves differential constraints, it opens up a more difficult problem types generally known as motion planning problems under differential constraints. They are the motion planning problems which are consider as the variant of TPBVPs [28] and are extensions of Geometric motion planning problems. In the following, general template for formulation of motion planning problem under differential constraints is presented.

1. Initialize search graph: Use a Initialize procedure to initialize starting nodes. It can be a single node or multiple nodes. These nodes are called as initial state nodes corresponding to the state set X_{init}. The goal nodes X_{goal} are initialized as well.
2. Select Node: Use the given node selection to choose a node $x_{current}$ for further expansion of the search graph.
3. Local Planner method: Use the local planner to sample the control space u_{new} from U and generate its trajectory from $x_{current}$ to x_{new}. This forms the TPBVP solver step and can be solved using different methods based on the need of the problem.
4. Insert an edge in the graph : Use the updating policy to update the search graph with new edge $[x_{current}\ x_{new}]$.
5. Check for solution : Check if the obtained latest node $x_{new} \in \mathrm{X}_{goal}$. If YES stop the iteration, the control path is returned as the solution.
6. Return to Step 2 : If NO, go to step 2 and iterate until the goal region X_{goal} is reached or some termination condition is met.

The description above is similar to Geometric motion planning problem. The goal is to find a path to the desired goal state from initial state. Only because of presence of the differential constraints, step 3 becomes the novelty here. Unlike Geometric Motion planning, now the connection between two nodes becomes a classical TPBVP. Much of the difficulty comes while trying to solve TPBVP. There are two methods that can be used to alleviate this issue. Both the methods act like a black box and provides the search algorithm with solution whenever they are supplied with required inputs.

- Motion Primitive: This method requires solving the TPBVP offline for every possible configurations, each instant of solved TPBVP is called a primitive .Once built, these motion primitives are used during the online searching phase of the algorithm. This technique is utilized in this thesis work for the attitude motion (see section 4.2.2).

- Approximate Method: This method aims to solve the TPBVP online by approximating or by solving few instances. This method is chosen to solve TPBVPs that arise during translation part of the motion planning in this work (see chapter 4).

Solutions to general motion planning(MP) problems requires

- Construction of the configuration space and obstacle space
- Solving TPBVP
- Checking for Collision

which makes the solving process computationally demanding and solving even a small dimensional problems becomes intractable. To gain some insights into this issue, in the next section a brief discussion on computation complexity of the MP problems are presented.

2.1.1 Computational complexity

When it comes to solving any problem, the measure of resources such as time and storage space required are mathematically characterized by Computational complexity. A Problem which can be solved in polynomial time is known as P-Class. The algorithms in P-Class runs at $\mathcal{O}(n^k)$ for some integer k to find the solution or report it if there is none. P-Class problems are the easiest of all the class. If a problem can at-least inform us whether it can be solved in polynomial time by using Non-deterministic algorithm, they are known as NP-Class problems. Non-deterministic algorithms are ideal algorithms which can take right decisions to say whether a problem is solvable in polynomial time or not. PSPACE Class are the problems that can be solved using polynomial amount of storage. Beyond PSPACE we have EXPTIME and EXPSPACE which are problems requiring exponential running time and exponential storage. It is trivial to state that EXPTIME AND EXPSPACE Class problems are intractable as compared to other categories. Another interesting concepts are X-hard and X-complete, where X is a general Class. A Problem X is X-hard, if every instant B in X can be reduced to A in polynomial time. And if a problem is both X-hard and belong to X-Class they are categorized as X-complete. For example, if a problem is NP hard and belong to NP Class then it is called NP-complete. Figure 2.2 shows the difficulties of each class ranging from P-Class problems which are at the inner most to the outer most intractable EXSPACE Class problems. Now that we have some idea about the complexity let us look at the lowest complexity bound that can be found for Motion planning problems. For geometric motion planning problem a simple example would be, Piano Movers problem [46] (a simple path planning problem which is trying find a path from one point to another desired point) the lower bound on complexity is PSPACE-complete. When it comes to Motion planning problem with differential constraint it can be anticipated to have higher difficulty. As can be seen in this paper [8] where the author considered kinodynamic motion planning for a particle with 3-D and 2-D cases, the complexity of the problem found to be NP-hard. This

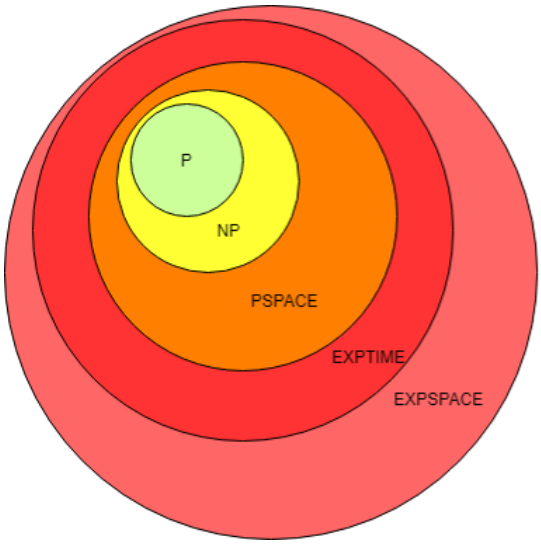


Figure 2.2: Complexity Classes

indicates that even by using efficient algorithms, the problem requires large amount of time and storage to get a solution or report if there is no solution. This is where the advantage of sample based motion planning (SBMP) algorithm comes in. SBMP algorithms samples the configuration space instead of building it entirely. The algorithm does the same thing when it comes to collision checking, it considers the collision checking as a black box and checks only if the sampled point is inside or outside of the obstacle region. In the next sections, SBMP methods are briefly discussed along with their important characteristics.

2.2 Sample based motion planning

2.2.1 Introduction

SBMP algorithms works like a machine which ties together multiple ropes. The rope represents the edge involved going from one point to another point. Usually for dynamic systems, finding solutions to these edge are generally known as two-point boundary value problem(TPBVP). Essentially SBMP algorithms does not solve the TPBVP, instead it ties the already solved TPBVPs together to form a huge line of rope or trajectory which are optimized for an objective at hand. SBMP tries to tie the best possible edges it can find at the sampled nodes. By doing this it tries to find the best possible trajectory as a whole.

2.3 Important concepts for SBMP algorithms

2.3.1 Sampling theory

SBMP considers continuous C-Space as discretized. On top of that, it tries to sample the discretized space. Thus it becomes very important how the sampling is carried out to achieve Resolution completeness or Probabilistic completeness (a notion which informs if a solution can be found by using the SBMP algorithms or not, check 2.3.3). In the following several concepts are presented which characterizes the sampling procedure, followed by brief description of Sampling methods.

Sample set and Sample sequence

It is important to know the differences between sample set and sample sequence. Sample set $\left\{x_i\right\}_{i=1}^{n}$ is the set of samples in C-Space or state space. Sample sequence is the particular ordered samples from sample set. We can construct the sample set from sample sequence but not the otherway.

Denseness

If we can cover every configuration or every state in the state space by sampling then we will find the solution. However, doing this is intractable, and to understand how a sample sequence covers the given space, we have denseness definition. The notion of how a particular sampling sequence can cover the given space is characterized by Denseness. Formally, Let A and B are the subsets in topological Space. The Set A is said to be dense in B if $cl(A) = B$. Basically Denseness informs that it can be possible for a sequence to cover every element of $\mathcal{C} \subseteq \mathbb{R}^m$. If a sequence is called Dense that means it is covering the configuration space efficiently and in that case possibility of finding a solution is higher.

Dispersion

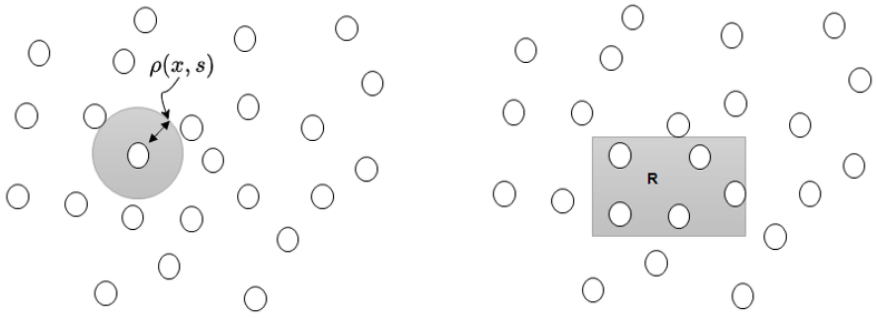
Dispersion measures the ability of the sampling sequence to cover the state space. Basically Dispersion is directly related to resolution completeness. If a sequence is highly disperse then that means sample sequence is not covering the configuration space efficiently leading to lower resolution completeness. Therefore, good sampling sequence will have lower dispersion. Dispersion makes use of metric function, which uses the notion of distance measure for the elements in State Space or higher dimensional configuration space. In the figure 2.3a, we have l_2 dispersion which is the largest radius of the empty normed ball.

The dispersion of a finite set S of samples in a metric space (X, ρ) is

$$\delta(\mathcal{S}) = \sup_{\mathbf{x} \in X} \min_{\mathbf{s} \in \mathcal{S}} \rho(\mathbf{x}, \mathbf{s}) \tag{2.2}$$

Discrepancy

Discrepancy is another important concept which measure the effectiveness of sampling method. The main difference is that unlike dispersion, discrepancy measures how the sample is distributed in a given volume. In mathematics, volume is characterized by measure Space. Let X be a measure



(a) Dispersion (b) Discrepancy

Figure 2.3: Dipsersion and Discrepancy

space. Let R be a collection of subsets of X that are called range space. Discrepancy of a particular sample set S of n points, and range Space, R is defined as

$$\mathcal{D}(\mathcal{S},\mathcal{R}) = \sup_{R\in\mathbb{R}} \left\{ \left| \frac{\mathcal{S}\cup\mathcal{R}}{n} - \frac{\mu(\mathcal{R})}{\mu(\mathcal{X})} \right| \right\} \tag{2.3}$$

where $\mu(.)$ represents the Lebesgue-Measure. Above equation informs us that given samples that fall into range space R, $\frac{\mathcal{S}\cup\mathcal{R}}{n}$, closely covers the actual volume fraction of R, $\frac{\mu(\mathcal{R})}{\mu(\mathcal{X})}$. Therefore, essentially what we want is that the $\mathcal{D}(\mathcal{S},\mathcal{R})$ remains small throughout the state space which gives search algorithm the possibility to find good quality solutions. It has been shown [28] that, for a unit hypercube, $\mathcal{P} \subset [0,1]^d$,for all axis aligned rectangular subsets $R \subset \mathcal{P}$. Discrepancy and Dispersion are related as

$$\delta(S) \leqslant (\mathcal{D}(\mathcal{S},\mathcal{R}))^{\frac{1}{d}} \tag{2.4}$$

Which implies that lower discrepancy leads to lower dispersion but not vice versa.

2.3.2 Sampling method

Sampling forms a main part of sampling based motion planning. There are two broad types of sampling techniques followed depending on the requirement of the application. They are deterministic and random-sampling.

Deterministic sampling approach maintains the reliability; guarantees the coverage of configuration space. The grid resolutions can be modified or changed to achieve good lower dispersion and discrepancy for the sampling. One of the best lower dispersion that is possible is Sukharev grid and forms the lower bound for dispersion and act as a ideal reference to any other sampling methods which tries to attain the lower dispersion. Low discrepancy sampling methods provides lower discrepancy, which, in-turn guarantees lower dispersion equation 2.4. Some of the widely used low

discrepancy methods are Halton and hammersly sampling methods. Halton sequence is a generalized version of van der corput sequence (which uses a binary representation to produce sample to cover the space between [0 1]) to n -dimensional case. Halton sequence is widely used in automation to replace random sampling because of its reliability and ability to produce low-discrepancy sampling. **Random sampling** as the name says samples the space randomly using a probability distribution of a variable that is being sampled. Pseudo random number generation is commonly used sampling method in SBMP problems. The drawback with this method is that it cannot guarantee the reliability of sampling. however, it was shown that the infinite sequence of independently randomly chosen points is only dense with probability one. Which indicates the Random sampling has the ability to efficienty cover the configuration space as number of samples goes to inifinity. Another important thing to note here is that random sampling is faster and easily adapted to variety of sampling including Cartesian and SO(3) groups [[28] Ch 5.2.2]. In this work, both deterministic and random sampling methods are used. However for analysis on the performance of the algorithm, deterministic sampling is primarily used.

2.3.3 Algorithmic properties

- **Completeness**

  Completeness is an important property for an algorithm. It informs whether the algorithm can find a solution in finite time. Exact motion planning algorithms have the ability to find solution in finite time. But for sampling based algorithms, it is very difficult to achieve this, due to its very nature of sampling. Therefore, a weaker notions of completeness are defined both for deterministic sampling and random sampling methods (see section 2.3.2). If a deterministic sampling method with dense sampling can figure out if there is a solution in finite time it is considered Resolution complete and it will not have capability to report failure if it does not find the solution unlike Exact approach. For random sampling method, Probabilistic Completeness is used. Probabilistic completeness informs that as the number of samples goes to infinity, the probability of finding the existence of the solution goes to unity.

- **Asymptotic optimality**

  Sampling based approaches are inherently have limitations on the sampling the configurations space to find the optimal solution. Similar to the weak completeness notion, due to its inability to sample the whole configuration space, a weaker notion of optimality is attributed to sampling based algorithms. If n is sufficiently large, we can expect to find a trajectory x_n which will be closer to the neighborhood of an optimal trajectory x^* but almost often surely offset from it. As a result, the trajectory which is found from sampling will have cost J_n higher than the optimal cost J^*. The weak notion of optimality informs that if the number of sampling goes to infinity then the cost found by the sampling algorithms will approach optimal cost. This

notion is suitable for both deterministic sampling and random sampling methods.

$$\begin{aligned} \lim_{n\to\infty} J_n &= J^* \quad \text{(Deterministic Sampling)} \\ \mathrm{P}\Big[\ \limsup_{n\to\infty}(J_n = J^*)\Big] &= 1 \quad \text{(Random Sampling)} \end{aligned} \tag{2.5}$$

Following two important definitions informs about the difference between obtaining feasible solution and an optimal solution for motion planning. Optimal solutions are strictly feasible solutions but feasible solutions need not be optimal. These two definitions are taken from [17].

Definition 1. Feasible motion planning: Let $X \subseteq \mathbb{R}^n$ be the n-dimensional search space of the planning problem, $X_{\text{invalid}} \subset X$ be the set of invalid states, and $X_{\text{free}} := X \backslash X_{\text{invalid}}$ be the resulting set of permissible states. Let $\mathrm{x}_{\text{start}} \in X_{\text{free}}$ be the initial state and $X_{\text{goal}} \subset X_{\text{free}}$ be the set of desired goal states. Let $\sigma : [0, T] \to X$ be a continuous function of bounded variation (i.e., a sequence of states) and $\Sigma_{feasible}$ be the set of all such paths connecting the start and goal solely through free space that can be executed by the system.

The feasible motion planning problem is then formally defined as finding any feasible path, $\sigma' \in \Sigma_{\text{feasible}}$, in the problem if a solution exists, i.e., $\Sigma_{\text{feasible}} \neq \emptyset$, and otherwise returning failure.

Definition 2. Optimal motion planning: Let $X \subseteq \mathbb{R}^n$ be the n -dimensional search space of the planning problem, $X_{\text{invalid}} \subset X$ be the set of invalid states, and $X_{\text{free}} := X \backslash X_{\text{invalid}}$ be the resulting set of permissible states. Let $\mathrm{x}_{\text{start}} \in X_{\text{free}}$ be the initial state and $X_{\text{goal}} \subset X_{\text{free}}$ be the set of desired goal states. Let $\sigma : [0, T] \to X$ be a continuous function of bounded variation (i.e., a sequence of states) and Σ_{feasible} be the set of all such paths connecting the start and goal solely through free space that can be executed by the system. Let $c : \Sigma \to [0, \infty)$ be a cost function such that all nontrivial, feasible paths have finite and strictly positive costs. Let $\Sigma^* := \left\{\sigma \in \Sigma_{\text{feasible}} \mid c(\sigma) = c^*\right\}$ be the set of all feasible paths with optimal cost, c^*. The optimal motion planning problem is then formally defined as finding any feasible path, $\sigma' \in \Sigma_{\text{feasible}}$, in the problem that has optimal cost, i.e., $\sigma' \in \Sigma^*$, if an optimal solution exists, i.e., $\Sigma^* \neq \emptyset$, and otherwise returning failure.

2.3.4 Exploration technique

Exploration technique dictates how the SBMP algorithms probes the state space and tries to find the solution. There are mainly two types of SBMP exploration techniques. One is Tree based exploration and another is Road-Map based Exploration .

Tree-based exploration: Tree-based exploration also called as single-query planning methods works based on building trees. The method probes the state space and build the trees from initial states till it finds the desired states. Following are the steps involved in Tree-based exploration.

1. Initialization: Initialize the tree by including starting points. One point x_{init} if it a uni-direction tree search, two points x_{init} and x_{goal} if it is bi-direction. For multiple tree search

algorithms, it can be multiple initial points and final points.

2. Vertex Selection Method: Choose a node in the tree for further expansion.

3. Local Planning method: Local Planner is used to connect the tree either to a new sampled node or to group of nodes.

4. Insert an edge in the graph: Once the new sampled node is connected by the Local Planner, it will be added to the existing tree.

5. Check for a solution: Check if the trees encodes a solution path.

6. Return to step 2 : Iterate until the solution path is found or some termination condition is satisfied; the algorithm will report failure.

Some of the examples of tree search exploration techniques are Rapidly-exploring Random Tree(RRT) [30], Expansive Space trees, and Fast Marching Tree (FMT^*)[21]
Figure 2.4 shows tree-based exploration technique. As can be noticed, the tree explores in every possible direction until it finds the desired final state. Advantages of the tree search exploration is that since it is single query technique, it is suitable for dynamic situations. The main complication comes with local planner. If sampling is done in state space, then the local planner is effectively a TPBVP solver. So choosing a efficient, at the same time fast TPBVP solver is essential for the tree-based method. If sampling is done in control space, the new nodes are added to the tree by applying this control sequence and by integrating the dynamics new nodes states can be found. Even though, this method has the ability to satisfy the control constraints, it is not trivial to know which control sequence will give the good coverage in state space. For this thesis work, the sampling is caried out in state space and TPBVP is solved to connect one sampled point state to another.

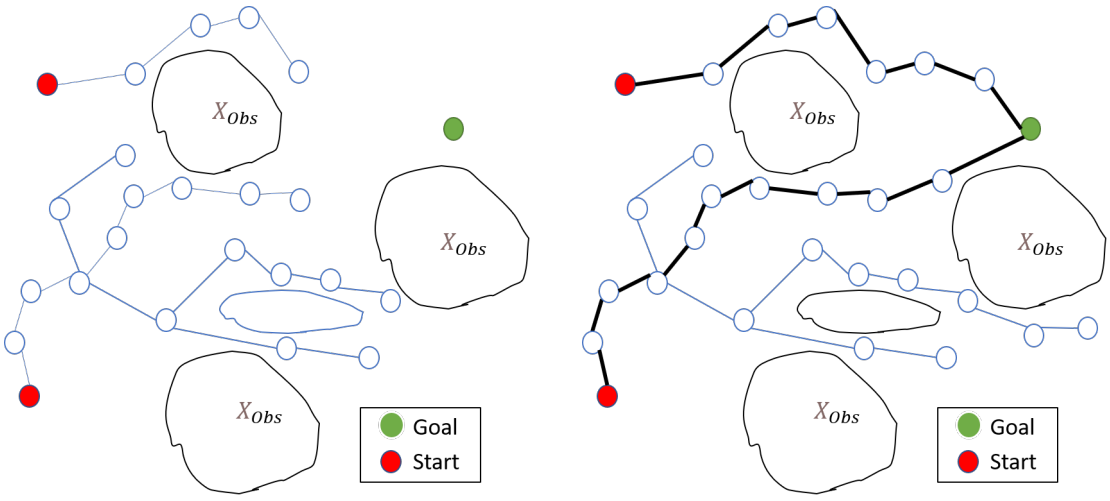
Road-map Exploration

As the name says, Roap-map Exploration method tried to build a map of whole configuration space. Once the map is built the method can find solutions for any initial and final points. Road-Map Exploration methods are generally known as Multi query planners. They are perfectly suitable if the environment remains static[25]. Following are the important steps involved

1. Pre-processing: This step involves building the map. Here, all the available samples are explored and a connectivity graph is built.

2. Local Planner: This step is similar to Tree-based exploration. The difference is that, here once the Initial $x_{init} \in X_{init}$ and final desired states $x_{goal} \in X_{goal}$ are given, a graph search algorithm (depth first search,Dijktra ,A^*,D^*,etc.) is used to determine the edges which connect the X_{init} and X_{goal}.

3. Iterate : Return to step 2 until all path connecting $x_{init} \in X_{init}$ and $x_{goal} in X_{goal}$ are found.

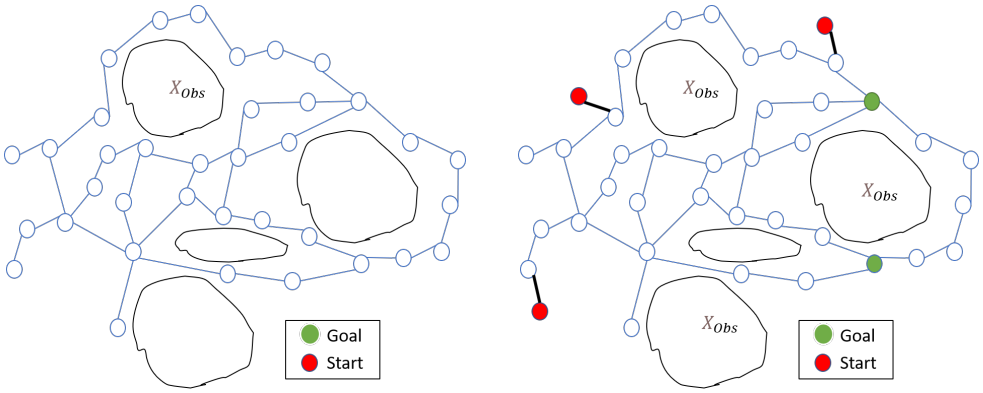


(a) Tree building (b) Reaching the Goal

Figure 2.4: Tree-based Exploration

Figure 2.5a shows the already built road-map for a problem. Now if initial nodes and final nodes are given, the local search algorithm will find a optimal path between them as shown in 2.5b. Some of the Road-map Exploration methods are PRM[25],PRM*[24]

Because of its inability to adapt to the dynamic environment, this method is very efficient only for static environment. If it is applied for dynamic environment, then pre-processing needs to be done again, which is a computationally expensive procedure. For our thesis guidance problem, we are dealing with dynamic environment, so this method will not be suitable.



(a) Generated Road-map (b) Connecting starting and goal nodes

Figure 2.5: Road-map Exploration

2.4 Fast Marching Tree*

In this section, the core search algorithm used in the thesis is presented. For the inspection problem, Tree search exploration method is selected because of its ability to handle dynamic environments. Particularly Fast Marching Tree* (FMT*)[21] is chosen as the core search algorithm. The rational for selecting this particular algorithm is presented in following paragraphs. Detailed discussions of this algorithm can be found in the original paper[21].

Overview

Fast marching Tree* incorporates the features of probabilistic roadmap algorithm(PRM) and sampling-based roadmap of trees (SRT). Also the name fast marching in FMT* comes from the fact that it works like a Fast Marching method which were used to solve wave front propagation problem (stationary Eikonal equation)[49]. Fast marching tree tries to search for solution in the forward direction and systematically grow the tree resembling Fast marching method. Apart from that, many of the important aspects such as direction of search and working principle differ considerably. For example, FMT* is governed by Bellman's optimality condition over randomized grid points whereas in Fast marching method upwind approximation schemes are used to solve Eikonal equation along orthogonal grid points.

Motivation for selection

Following features of the FMT* helped in the decision to use it as the motion planning algorithm for inspection problem.

1. FMT*'s asymptotically optimal property guarantees that solution will be found as number of samples goes to infinity.
2. FMT*'s suitability outperforms many of the state of the art algorithms such as PRM* and RRT* when it comes to higher dimensional motion planning problems.
3. FMT* grows the tree in a systematic and structural way. The Tree is expanded in Cost-to-arrive space.
4. FMT* uses a Lazy dynamic programming approach where once it detects the collision it stops and searches in other direction making the search procedure fast.

2.4.1 High-level description

The FMT* algorithm executes a forward dynamic programming recursions over predetermined sample space and builds a tree steadily outward in a cost-to-go space (cost incurred while moving from current node to another new node). The features of this dynamic programming recursion are,

- The algorithm is tailored to form disk-connected graphs, where two samples are considered as neighbors and hence connectable, if their distance is below a given bound referred to as the connection radius.

- It performs graph construction and graph search concurrently
- For the evaluation of the immediate cost in the dynamic programming recursion, the algorithm "lazily" ignores the presence of obstacles, and whenever a locally-optimal connection to a new sample intersects an obstacle, that sample is simply skipped and left for later as opposed to looking for other connections in the neighborhood.

Evidently the important feature of the FMT* is the third point. The algorithm is considered as "lazy" algorithm because it avoids the number of costly collision checking for the obstacle by entirely avoiding to grow the tree once it finds the new point intersecting with obstacle. This may lead to sub-optimal connections. But in [21] authors proved that central property of FMT* is that the cases where a sub-optimal connection is made become very rare as the number of samples goes to infinity.

2.4.2 Algorithm description

The psuedocode for the implementation of the algorithm is described below. Before looking at the main blocks of the algorithm, main elements involved are described. The FMT* always keeps tracks of sample set $\mathcal{X} = \begin{Bmatrix} x_{init} & X_{free} & X_{goal} \end{Bmatrix}$ denoted as V. V is subdivided into 3 subsets $\mathcal{V}_{closed}$,$\mathcal{V}_{open}$, and $\mathcal{V}_{unvisited}$. As will be seen in the algorithm description, $\mathcal{V}_{unvisited}$ denotes nodes which have not visited. $\mathcal{V}_{open}$ are the nodes which are at the forefront of the tree and considered as frontier nodes. These frontier nodes are the nodes considered for further expansion i.e., connecting frontier nodes to nodes in $\mathcal{V}_{unvisited}$. $\mathcal{V}_{closed}$ are the nodes which are behind the open nodes and are not considered again for the expansion. Algorithms from 2.1 to 2.3 shows the psuedocode of FMT* Algorithm. Psuedocode is divided into small blocks to make it understandable. The main algorithm block is 2.1. In the following some of the building blocks of the Algorithm is described. Figure 2.7 shows the schematics of FMT* working principle.

- Sample(X_{free}): Samples the free space and returns S samples.
- $Near(V, z, r_n)$: This block returns the samples in V which are inside connection radius r_n from z.
- ADDNODE: Takes care of adding nodes to the Tree, Frontier set V_{open}, and removing of nodes from $V_{unvisited}$.
- $Expand(T, z, r_n)$: This is the main block in the FMT* Algorithm. This will expand the tree T by adding best possible nodes present in $V_{unvisited}$.
- Collision Check $(\hat{x}, x)$: This block returns 1 or 0 based on the occurrence of collision while considering adding new edge $(\hat{x}, x)$ to the Tree. 1- Collision occurs and 0 - Collision does not occur.

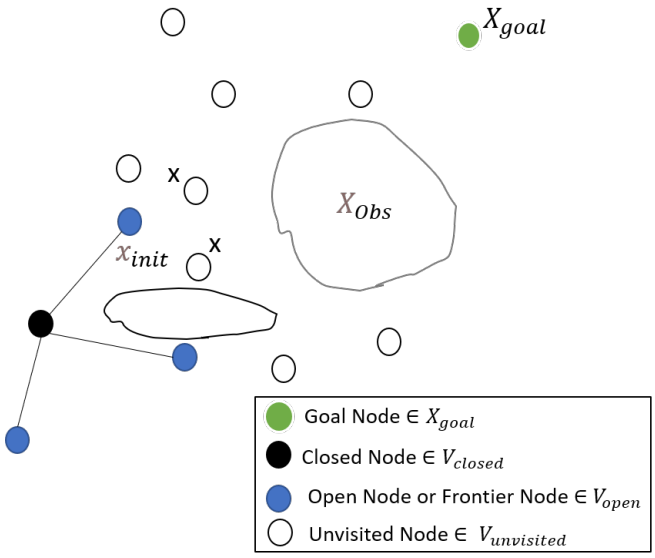


Figure 2.6: Path Planning using FMT^*

- cost(x,T): Calculates cost for node x in tree T.
- cost(x,$\bar{x}$): Calculates the cost to go from the Node $\bar{x}$ to the node x.

At the start of the algorithm, $\mathcal{V}_{closed}$ and tree T is initialized. T will store the information about $\mathcal{V}_{open}$,$V_{unvisited}$ and E. Initial sample x_{init} is added as a root node to the tree T. Using *Sample(n)*, the free space X_{free} sampled to get n samples. These n samples are combined with x_{init} to form a sample set S (line 1). The initial node x_{init} is assigned to z which forms the initial open node and path , σ^* , is empty at this point $\sigma^* = \emptyset$. The main loop starts by calling the *Expand* block which takes inputs as the Tree (T), open node z, and the connecting radius r_n. Expand block as the name indicates expands the tree by finding the best node in sample space. Expand procedure does two rounds of searching, one is from the perspective of open nodes in the tree and another from the un-visited nodes. The first Near block takes inputs as $V_{unvisited}$, current optimal open node z, and r_n. The aim is to find all the unvisited nodes Z_{near} which are inside radius r_n from current optimal open node z. Moving to next step which is a **for loop** in the expand block. Now for each sample x in Z_{near}, second Near block (Expand: line 4) finds the open nodes in the tree (T) within radius r_n and adds them to X_{near}. X_{near} represents all the open nodes which are near unvisited node x. Next find the best open node in X_{near} (Expand: line 5) and store it as x_{min}. Till now collision check was not performed which informs that the connection between the nodes was done lazily (without considering collision checking), At this stage, collision check is done on the obtained best open node x_{min} corresponding to the un-visited node x. If there is no collision, x will be added to the tree to form a new edge $[x_{min}, x]$. If in case the collision occurs, that particular un-visited

node will be dropped from further steps in the Iteration i.e., even if this node is near other open nodes X_{near}, it will not be considered for addition. This is considered as the main characteristics of FMT^* called Lazy expansion process, this procedure reduces the computation cost for collision checking [[21] section 3.1]. This way of doing dynamic programming will not incur cost in terms of

Algorithm 2.1: FMT* Algorithm: Main

Input : Query($x_{init}, \mathcal{X}_{goal}$),Sample radius r_n, *Samplecount* n
1 $S \longleftarrow \{x_{init}\} \cup Sample(n)$
2 $T \longleftarrow Initialize(S, x_{init})$
3 $z \longleftarrow x_{init}, \sigma^* \longleftarrow \emptyset$
4 **while** $\sigma^* = \emptyset$ **do**
5 $T \longleftarrow Expand(T, z, r_n)$
6 **if** *Termination ()* **then**
return failure
7 **else if** $V_{open} = \emptyset$ **then**
8 $T \longleftarrow Insert(T, r_n)$
9 $z \longleftarrow \arg\min_{\hat{x} \in V_{open}} \{Cost(\hat{x}, T)\}$
10 **if** $z \in \mathcal{X}_{goal}$ **then**
11 $\sigma^* \longleftarrow Path(z, T)$
12 **return** σ^*

asymptotic optimality. The algorithm works in such a way that the currently dropped node will be connected to the tree frontier or open nodes via alternative locally-optimal connection during future iterations. This is also supported by the fact that sub-optimal connections made by FMT^* become vanishingly rare as the number of samples goes to infinity [section 3.2 and 4 of [21]]. Now going back to main algorithm (Algorithm 2.1), once expansion block finishes running, original tree T is updated by possible addition of new edges. In the next step (line 6) termination check is performed. If yes the algorithm returns failure. If No, new node will be inserted to the tree T in the form of open node by using Insert block. Insert block gives the algorithm the anytime behavior due to presence of at-least one open node at anytime during the iteration. The Insert block tries to find a sample using Sample block which is near the tree. This gives a possibility to re-open the previously closed nodes for expansion again. Proceeding forward with a tree with at least one node open. Next step in the main algorithm is to find the best open node and assign it to z. In line 10, z is checked if it is in X_{goal} if yes the algorithm has found the path; because of Bellman's principle of optimality the path traced from z back to the root node will be the optimal path. If z $\notin$X$_{goal}$ the algorithm proceeds to line 5 to expand by using the node z until either termination stops it or it finds the node $\in$X$_{goal}$.

Function `INITIALIZE(`S,x_0`):`

$V \longleftarrow \emptyset$
$\epsilon \longleftarrow \emptyset$
$V_{open} \longleftarrow \emptyset$
$V_{unvisited} \longleftarrow \emptyset$
return $T \longleftarrow AddNode(V, E, V_{unvisited}, V_{open}, x_0)$

Function `AddNode(`T, V, E, x`):`

$V \longleftarrow V \cup \{x\}$
$E \longleftarrow V_{open} \cup \{x_{min}, x\}$
$V_{open} \longleftarrow V_{open} \cup \{x\}$
$V_{unvisited} \longleftarrow V_{unvisited}/\{x\}$
return $T \longleftarrow (V, E, V_{unvisited}, V_{open})$

Algorithm 2.2: Expand

Input : Expand(T,z,r_n)
begin
$V_{Newopen} \longleftarrow \emptyset$
$Z_{near} \longleftarrow Near(V_{unvisited}, z, r_n)$
for $x \in Z_{near}$
do
$X_{near} \longleftarrow Near(V_{open}, x, r_n)$
$z \longleftarrow \arg\min_{\hat{x} \in X_{near}} \{Cost(\hat{x}, T) + Cost(\hat{x}, x)\}$
if $CollisionCheck(x_{min}, x)$
then
$(V, E, \mathrm{V}_{unvisited}, V_{Newopen}) \longleftarrow AddNode((V, E, \mathrm{V}_{unvisited}, V_{Newopen}), x)$
$\mathrm{V}_{open} \longleftarrow \mathrm{V}_{open} \cup \mathrm{V}_{Newopen}/\{\mathrm{z}\}$
return $T \Longleftarrow (V, E, V_{unvisited}, V_{open})$

2.5 Adaptation requirements of FMT* to the thesis problem

In the previous sections sample based motion planning algorithms and in particular FMT* algorithm were described briefly. Next step is to identify requirements needed to solve inspection problem 1.3 by using FMT* algorithm. Following are the adaptation requirements

1. Due to the mission flexibility assumptions (see section 1.4 and 4.1), translation and rotational dynamics are decoupled. This requires to adapt the search algorithm in such a way, it can samples both translation and attitude trajectories and finds the best possible solution for the inspection problem.

2. Objective function: Objective function in the above description of FMT^* was simple arc-length. But, Inspection problem poses a multi-objective problem. The selection of nodes for

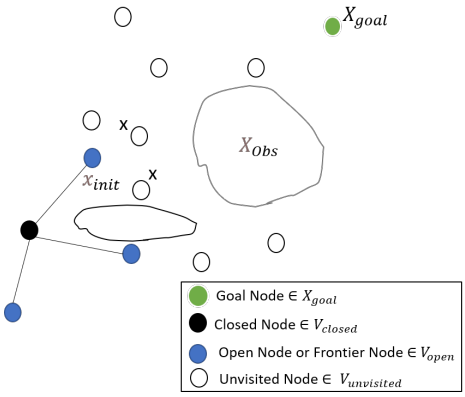
Algorithm 2.3: Insert

Input : Insert(T,r$_n$)
begin
  while $V_{open} = \emptyset \quad and \quad \backsim Terminate()$ **do**
2 $s = Sample(1)$
3 $V_{near} \longleftarrow Near(V, s, r_n)$
4 **while** $V_{near} \neq \emptyset$ **do**
5 $x_{min} \longleftarrow \arg\min_{x \in V_{near}} \{Cost(x, T) + Cost(x, s)\}$
6 **if** $CollisionCheck(x_{min}, s)$ **then**
7 $T \longleftarrow AddNode(T, s)$
8 **break**
9 **else**
  $V_{near} \longleftarrow V_{near} / \{x_{min}\}$
10 **return** $T \Longleftarrow (V, E, V_{unvisited}, V_{open})$

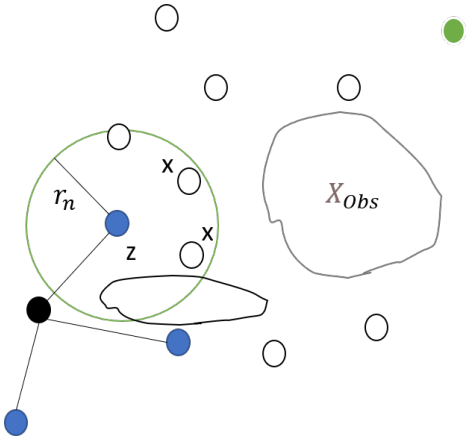
expansions and finding the best minimum candidates points requires consideration of dominated and non-dominated concepts (see section 4.2.4). .

3. Goal conditions are in objective functional space: As seen in previous discussions on motion planning algorithms, the objective of the path planning problem is to find the goal state. But in inspection problem, goal condition lives in multi-objective functional space. This requires FMT* to adapt its searching criteria to look for solution which are good in terms of coverage, fuel cost, and actuator torque cost (see section 1.3). This adaptation influences how the sampling is carried out, as sampling in $\mathcal{R}^3$ will not provide search direction because of absence of goal condition in $\mathcal{R}^3$.

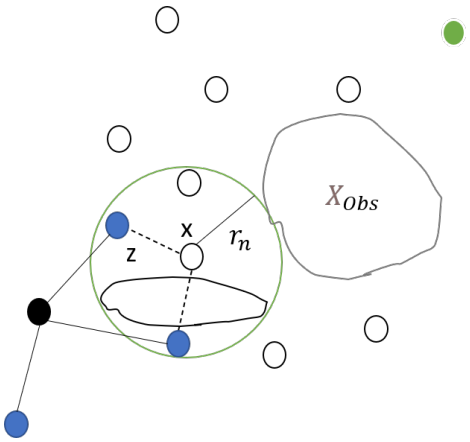
4. Neighborhood Reachability r_n: Neighborhood of a node in path planning problem is usually a distance metric. For Inspection problem, the objective function is different and goal resides in objective functional space. So the Neighborhood nodes are considered near to the frontier node only if they fall below a r_n. Where, r_n now should be defined in objective function space (see section 4.2.6).



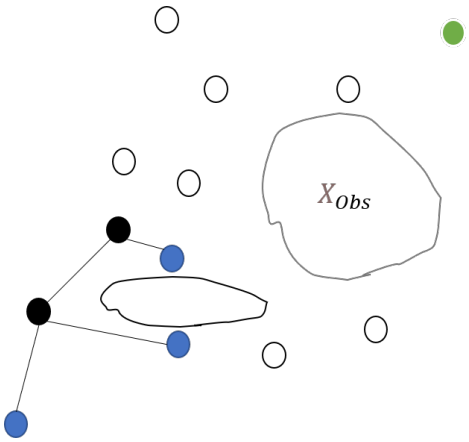
(a) Current optimal frontier node is x_{init}. Next step is to find the neighboring nodes with in raidus r_n



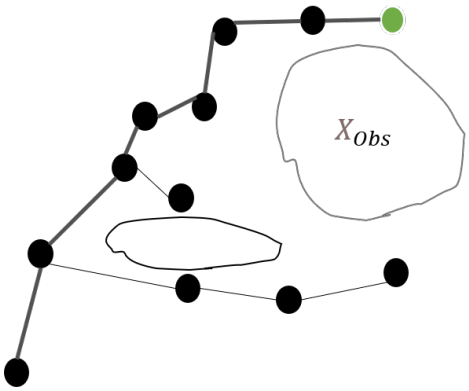
(b) With in the radius r_n, nighboring nodes are found and added as the nodes which are near to x_{init} Z_{near}



(c) From each unvisited node in $x \in Z_{near}$ a neighboring frontier nodes X_{near} are found with in radius r_n.Here, the collision checks are not considered - lazy dynamic expansion



(d) Once the optimal frontier node is found from X_{near}, collision check is performed and if collision is not present the edge is formed between new node and optimal frontier node



(e) The procedure is repeated until the goal region or node is found

Figure 2.7: FMT^* working principle

Chapter 3

System Dynamics

"I... a universe of atoms, an atom in the universe."

- Richard P. Feynman

In this part of the thesis, basic theoretical background related to relative translation and attitude dynamics of spacecraft is described.

3.1 Translation motion

In this section, general translation relative dynamics of the spacecraft is described briefly along with reference frame description. Firstly, reference frames are defined then general dynamics is described. The section ends with linear dynamics model which is used for the thesis work.

3.1.1 Reference frames

Inertial frame

This frame is assumed to be attached to the central body around which the target is orbiting as shown in the figure 3.1. Represented by $\mathcal{I}_{\mathcal{N}}$ with orthonormal basis $\mathcal{I}_{\mathcal{N}} = \begin{Bmatrix} n_x & n_y & n_z \end{Bmatrix}$ and corresponding co-ordinates are represented as $\{\mathcal{X}_I, \mathcal{Y}_I, \mathcal{Z}_I\}$ respectively. Unit vectors n_x and n_y lie in the equatorial plane with n_x coinciding with the line of equinoxes. n_z completes the right hand co-ordinate system. This frame's origin may be at the centre of planet Earth or the Sun depending on the problem that is being solved.

Local vertical local Horizontal (LVLH) frame

This frame is considered as the local frame as it is attached to the body. There are two LVLH frames needed for analysis of proximity operation as shown in figure 3.2. The first frame is attached

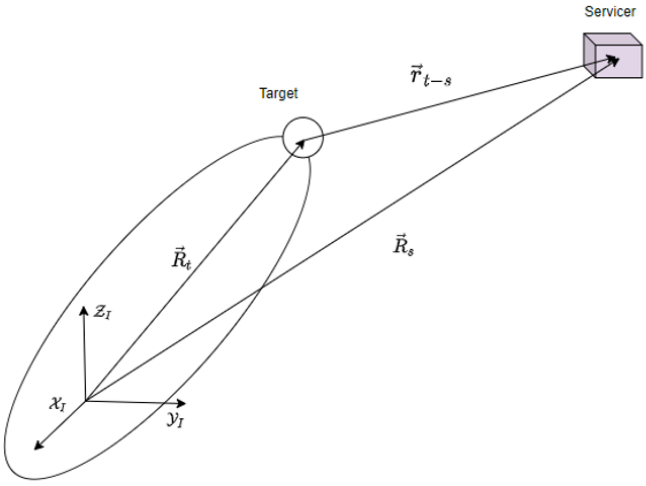


Figure 3.1: Inertial frame with target and servicer

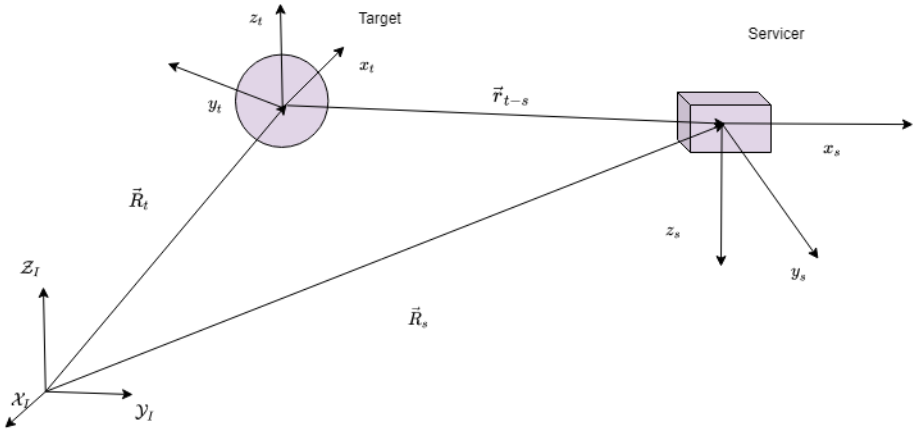


Figure 3.2: LVLH frame of target and servicer

to the target and is denoted by $\mathcal{L}_t$ with orthonormal basis $\mathcal{L}_t = \begin{Bmatrix} l_{xt} & l_{yt} & l_{zt} \end{Bmatrix}$ with co-ordinates represented by $\{x_t, y_t, z_t\}$. l_{xt} is in direction of the position vector from centre of $\mathcal{I}_\mathcal{N}$ frame and pointing outwards. l_{yt} is in direction of orbital tangential velocity vector. Finally l_{zt} completes the right-handed system. In the literature, this directions are commonly referred differently. l_{xt} is called radial direction, l_{yt} as in-track direction, and l_{zt} as out of track direction. These directions put together forms a local- Vertical local Horizontal frame for the target. The relative dynamics of the servicer with respect to the target is described with respect to this frame. Second LVLH frame is attached to the Servicer spacecraft. The frame is similar to the first LVLH frame and denoted by $\mathcal{L}_t$ with orthonormal basis $\mathcal{L}_s = \begin{Bmatrix} l_{xs} & l_{ys} & l_{zs} \end{Bmatrix}$ with co-ordinates represented by $\{x_s, y_s, z_s\}$. l_{x_s} is in direction of the position vector from centre of target frame $\mathcal{L}_t$ and pointing outwards. l_{ys} is in

direction of tangential velocity vector. Finally l_{zs} completes the right-handed system.

3.1.2 Relative dynamics of a spacecraft

General relative dynamics of the servicer with respect to target's LVLH frame is given by 3.11. Interested readers are recommended to look at one of these references for derivation of general relative dynamics[22][2].

$$\begin{aligned}
\ddot{x} - 2\dot{\theta}_t\dot{y} - \ddot{\theta}_t y - \dot{\theta}_t^2 x &= -\frac{\mu\left(r_t + x\right)}{\left[\left(r_t + x\right)^2 + y^2 + z^2\right]^{\frac{3}{2}}} + \frac{\mu}{r_t^2} + d_x + u_x \\
\ddot{y} + 2\dot{\theta}_t\dot{x} + \ddot{\theta}_t x - \dot{\theta}_t^2 y &= -\frac{\mu y}{\left[\left(r_t + x\right)^2 + y^2 + z^2\right]^{\frac{3}{2}}} + d_y + u_y \\
\ddot{z} &= -\frac{\mu z}{\left[\left(r_t + x\right)^2 + y^2 + z^2\right]^{\frac{3}{2}}} + d_z + u_z
\end{aligned} \tag{3.1}$$

where, x, y, z are the co-ordinates in $\mathcal{L}_t$ frame. θ_t and $\dot{\theta}_t$ are the angular velocity and angular acceleration respectively of the target's orbit. $\vec{d} = \{d_x.d_y, d_z\}$ and $\vec{u} = \{u_x.u_y, u_z\}$ are the external disturbance and control action respectively in $\mathcal{L}_t$ frame. r_t is the position vector of the target from Inertial Reference frame $\mathcal{I}_\mathcal{N}$.

3.2 Linearized relative dynamics

Modeling of the relative dynamics as a linear model yield sufficiently accurate results when the spacecraft is at proximity of the target [[2], Ch 5]. With assumptions that servicer's relative position with respect to target is smaller than target's position from the $\mathcal{I}_\mathcal{N}$ frame $r_t >> r_{rel}$. The following equation is obtained.

$$\begin{aligned}
\ddot{x} - 2\dot{\theta}_t\dot{y} - \ddot{\theta}_t y - \dot{\theta}_t^2 x &= \frac{2\mu x}{r^3} \\
\ddot{y} + 2\dot{\theta}_t\dot{x} + \ddot{\theta}_t x - \dot{\theta}_t^2 y &= -\frac{\mu y}{r^3} \\
\ddot{z} &= -\frac{\mu z}{r^3}
\end{aligned} \tag{3.2}$$

Above equation represents the general form of linearized relative dynamics for arbitrary target orbit.

For more restricted case, where the target orbit is considered circular, the linear relative dynamics is governed by Clohessy whiltshire model. In this thesis work, Clohessy whiltshire model is primarily used for the translation relative dyanmics model. But other models which are accurate than this can be included with proper adjustments to the framework (see section 4.2.2).

3.2.1 Description of the C-W dynamics

Clohessy Whiltshire dynamics model [13] is a simple dynamic model which is the linear approximation of equation 3.1. Linearization is made simple by assuming that the target's orbit is circular. This makes the LVLH frame to have fixed angular velocity called mean motion of its orbit. Linearized dynamics is given as

$$\begin{aligned} \ddot{x} - 3n_t^2 x - 2n_t\dot{y} &= u_x \\ \ddot{y} + 2n_t &= u_y \\ \ddot{z} + n_t^2 z &= u_z \end{aligned} \tag{3.3}$$

Where $\begin{Bmatrix} u_x & u_y & u_z \end{Bmatrix}$ are the control inputs which are thrust accelerations. n_t is the mean motion of the target orbit. In State-space form with control inputs is given by

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3.4}$$

where the dynamics matrix A and input matrix B is given by

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n_t^2 & 0 & 0 & 0 & 2n_t & 0 \\ 0 & 0 & 0 & -2n_t & 0 & 0 \\ 0 & 0 & -n_t^2 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

For homogeneous C-W dynamics, the State transitionion matrix $\Phi(t - t_0)$ is given by

$$\Phi(t-t_0) = \begin{bmatrix} 4 - 3\cos n_t(t-t_0) & 0 & 0 & \frac{1}{n_t}\sin n_t(t-t_0) & \frac{2}{n_t}(1-\cos n_t(t-t_0)) & 0 \\ 6(\sin n_t(t-t_0) - n_t(t-t_0)) & 1 & 0 & -\frac{2}{n_t}(1-\cos n_t(t-t_0)) & \frac{1}{n_t}(4\sin n_t(t-t_0) - 3n_t(t-t_0)) & 0 \\ 0 & 0 & \cos n_t(t-t_0) & 0 & 0 & \frac{1}{n_t}\sin n_t(t-t_0) \\ 3n_t \sin n_t(t-t_0) & 0 & 0 & \cos n_t(t-t_0) & 2\sin n_t(t-t_0) & 0 \\ -6(1-\cos n_t(t-t_0)) & 0 & 0 & -2\sin n_t(t-t_0) & 4\cos n_t(t-t_0) - 3 & 0 \\ 0 & 0 & -n_t \sin n_t(t-t_0) & 0 & 0 & \cos n_t(t-t_0) \end{bmatrix} \tag{3.6}$$

and finally to establish a bounded condition [2] for the servicer trajectory around the target. The initial state is required to satisfy following condition.

$$\dot{y}(0) = 2n_t x(0) \tag{3.7}$$

This condition is considered to be locally bounded condition. To get a global bounded condition for 1:1 resonance between servicer and target orbits, the initial state should satisfy the energy constraint condition [18] given by

$$\frac{1}{2}\left\{\left[\dot{x}(0)-\dot{\theta}_t(0)y(0)+\dot{r}_t(0)\right]^2+\left\{\dot{y}(0)+\dot{\theta}_t(0)\left[x(0)+r_t(0)\right]\right\}^2+\dot{z}(0)\right. \\ \left.-\frac{\mu}{\sqrt{\left[r_t(0)+x(0)\right]^2+y^2(0)+z^2(0)}}=-\frac{\mu}{2a_t}\right. \tag{3.8}$$

[64]provides the energy bounded condition for formation flying in presence of J2-effect and[38] gives us the energy bounded conditions for Swarm of satellites in the presence of J2 effect. Equation 3.8 is for the general case which can handle any type of orbit. If we are considering only circular orbits then the equation is reduced to

$$\frac{1}{2}\left\{\left[\dot{x}(0)-n_t y(0)\right]^2+\left\{\dot{y}(0)+n_t\left[x(0)+r_t(0)\right]\right\}^2+\dot{z}(0)\right. \\ \left.-\frac{\mu}{\sqrt{\left[r_t(0)+x(0)\right]^2+y^2(0)+z^2(0)}}=-\frac{\mu}{2a_t}\right. \tag{3.9}$$

3.3 Rotational motion

This section briefly introduces the concepts related to relative attitude dynamics and kinematics.

3.3.1 Relative attitude dynamics

Rigid body dynamics

Rotational motion of an rigid body is governed by Euler rigid body equations[22] given by

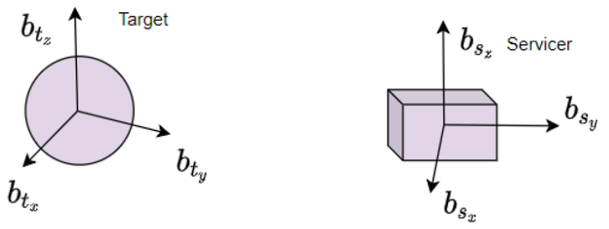
$$J\dot{\vec{\omega}}=J\vec{\omega}\times\vec{\omega}+\vec{u} \tag{3.10}$$

where, J and $\vec{\omega}$ represents the moment of inertia of the rigid body and angular velocity respectively. $\vec{u}$ represents the external torque.

Body reference frame

Target's body fixed frame is denoted as $\mathcal{B}_t$ with orthonormal basis $\mathcal{B}_t=\left\{b_{t_x}\quad b_{t_y}\quad b_{t_z}\right\}$ and its angular velocity is given by ω_t and shown in figure 3.3. Similarly for Servicer, the body frame is represented by $\mathcal{B}_s=\left\{b_{s_x}\quad b_{s_y}\quad b_{s_z}\right\}$ with angular velocity ω_s. In the figure the frames are attached to the body arbitrarily but in Numerical simulation section (see section 5), the frame will be attached in particular fashion to dictate inspection payload direction. The relative angular velocity with respect to Inertial frame is given by

$$\omega_{rel}^{I}=\omega_t-\omega_s$$



(a) Target body Frame (b) Servicer body Frame

Figure 3.3: Body frame of target and servicer

With respective to target frame, relative angular velocity is given by

$$\omega_{rel}^{t} = \omega_t - \mathcal{R}(q)\omega_s$$

where, R(q) is the rotation matrix between target and servicer's body frame and it is parameterized by quaternions. R can be parametrized by various types of parameters such as Euler angles, Rodrigues parameters and modified Rodrigues parameters. But due to its non singularity property and wider usage quaternions are utilized in this work. Quaternions are formed by 3 vector elements $[q_1\ q_2\ q_3]$ and one scalar elements q_4.

$$q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}$$

Let us assume $\mathcal{I}_t$ and $\mathcal{I}_S$ are the moment of inertia of target and servicer respectively. The relative attitude dynamics equation between the target and servicer in the target's reference frame $\mathcal{B}_t$ is given 3.11 by([2] Ch.9).

$$\mathcal{I}_t\dot{\omega} = \mathcal{I}_t\mathcal{R}{\mathcal{I}_s}^{-1}[\mathcal{D}_s - \mathcal{R}^T(\omega + \omega_t) \times \mathcal{I}_s\mathcal{R}^T(\omega + \omega_t)] - \mathcal{I}_t\omega_t \times \omega - [\mathcal{D}_t - \omega_t \times \mathcal{I}_t\omega_t] \tag{3.11}$$

where $\mathcal{D}_t$ and $\mathcal{D}_s$ are the external disturbances acting on target and servicer respectively. This equation is represented in terms of target's angular velocity and relative angular velocity between target and servicer.

3.3.2 Relative attitude kinematics

Here, all the variables involved in the equations represent in relative sense unless otherwise specified. Relative kinematics are represented by using quaternions as shown below

$$\dot{q} = \mathcal{R}(q)\omega \tag{3.12}$$

For the same reason similar to quaternions where the non-singularity is avoided. The direction cosine matrix (DCM) can be used, but it may incur more computation than quaternions because of parametrizing by 9 elements as compared to 4 parameters of quaternions. The kinematics equation in DCM form is given by

$$\frac{dA}{dt} = \Omega^T A \tag{3.13}$$

where, A is the direction cosine matrix and Ω is a skew symmatrix matrix formed out of angular velocity elements. It is written as

$$\Omega = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{3.14}$$

For detailed derivations and more informations related to attitude dynamics and kinematics refer to [63][2].

Chapter 4

Autonomous Motion Planning Spaceraft Guidance for Inspection Mission

"... he who seeks for methods without
having a definite problem in mind
seeks for the most part in vain"

-David Hilbert

This chapter presents the core idea of the thesis; the guidance framework intended for Inspection mission built by using motion planning algorithm. Chapter starts by presenting the the whole framework while highlighting the high-level features. After this section, various adaptation requirements described in section 2.5 are met by applying different modifications to the motion planning algorithm. Following that, last section is dedicated for the detailed explanation of how the framework works (section 4.3.3). Here, the algorithmic presentation of the developed framework along with its associated functional blocks are explained.

4.1 High-level description

The developed framework aims to solve the problem mentioned in section 1.3 using the motion planning approach. Framework is named as Autonomous Motion Planning Spacecraft Guidance for Inspection Mission(AMPSGIM). As we have seen in the chapter 3, the main elements of the FMT* algorithm needs to be modified if it is to be applied to get guidance law for inspection problem. Figure 4.1 shows the schematics of how the AMPSGIM works. General Inputs and Output of the

framework will be

Input: Number of satellites, Duration of Inspection, and Percentage of coverage required, Target size.

Output: Coverage details, translation and attitude motion of each satellites.

Figure 4.1, shows the core block named as **II**, which includes the FMT^* based algorithm blocks

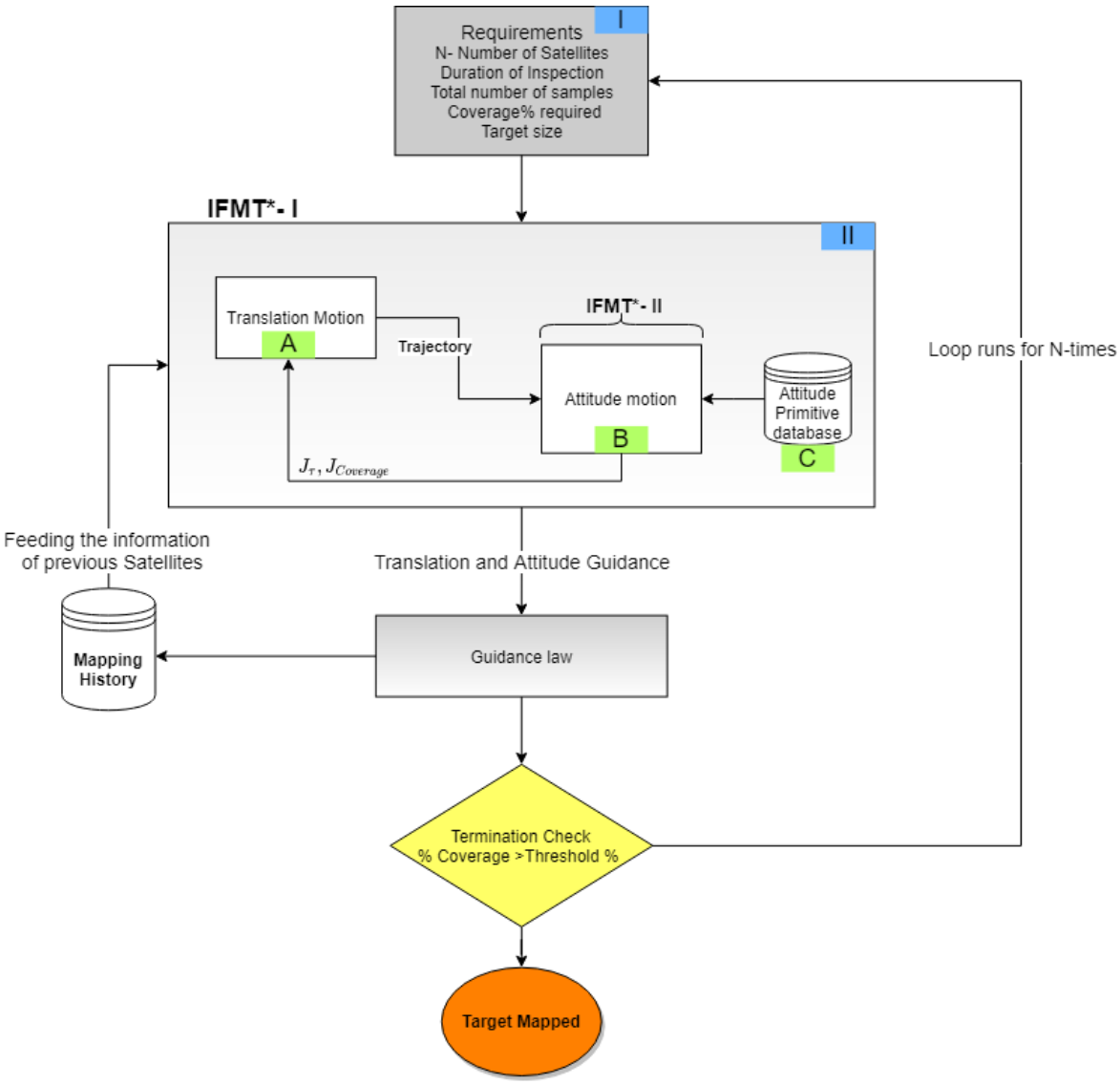
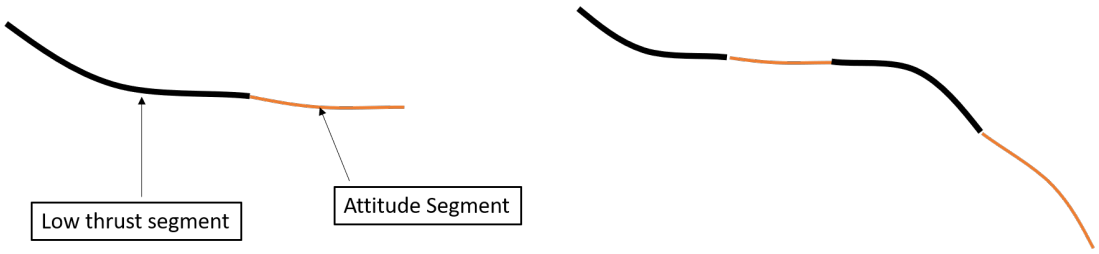


Figure 4.1: Autonomous Motion Planning Spacecraft Guidance for Inspection Mission(AMPSGIM)

grouped into a category called $IFMT^*$ which stands for "Inspection Fast marching tree*". There are two $IFMT^*$s, One is the outer $IFMT^* - I$, which finds the best translation motion. It encapsulates second $IFMT^*$ called $IFMT^* - II$ which takes care of attitude motion. From high level perspective,

$IFMT^* - I$ produces best feasible translation trajectories via block **A** and sends it to $IFMT^* - II$ block **B**, where the suitable attitude trajectories are merged with the given translation trajectories. Once the feasible translation and attitude trajectories are found, the $IFMT^* - I$ will holistically decide which trajectory is good (considers fuel cost, coverage, and control torque). So by doing this, the coupled guidance is produced. The detailed description of how this is accomplished can be found in section 4.3.3. The strategy used here is that the translation trajectories produced by $IFMT^* - I$ are assumed to consist of two types of segments namely low thrust segment and attitude segment figure 4.2. Every translation segment has strictly one low thrust segment and one attitude



(a) Single translation segment (b) Two translation segments

Figure 4.2: Translation segment structure

segment. The translation trajectory linked with attitude segment will be used by $IFMT^* - II$ to find the best feasible attitudes. Inside the attitude segment, the sequence of slew and tracking can be structured according to the needs (see section 4.3.1) of the Inspection scenario. During low thrust segment, the spacecraft is assumed not to do any observation as the attitude of spacecraft is crucial for providing pointing direction for the low thrusters. In the attitude segment, the low thrusters are assumed be off and the spacecraft's attitude is used solely for observation of the target, here, the trajectory is under natural dynamics. This way of dividing the trajectories provides good flexibility for the mission, as supposed to having both low thruster and inspection payloads (for example Visual Camera or Thermal Imager) working together which may incur higher power requirement. These assumptions makes the framework produce solutions decoupled in the translation and attitude dynamics. However, the guidance produced is considered coupled as it tries to find the best feasible solution both in translation and attitude motion for Inspection problem. Once the blocks find the good trajectories for a given satellite, the information about the satellite will be stored in **Mapping History block** and will be used while the block (**II**) runs for next satellite. This step is critical as it provides information for collision avoidance and pointing avoidance between satellites. The loop runs until either the coverage is less than some minimum threshold or all the satellites are considered for the framework.

4.2 Important elements required for the framework

Important elements associated with APMSGIM is presented in this section. These elements form the core part of the framework and tries to adapt Inspection problem suitable for SBMP algorithms. Firstly, we will look at constraints involved in the framework which comes directly from Inspection problem. Secondly, solving TPBVP also known as steering problems by using faster and approximate methods are presented. This part is essential for $IFMT^*$ algorithms which are based on FMT^* workflow. Here, both translation and attitude steering solution methods are presented. This is followed by the description of elements required to run IFMT* algorithms.

4.2.1 Constraints description

Constraints plays a crucial part in deciding whether a solutions that has been obtained is feasible or not. For Inspection problem (section 1.3), the formulation indicates constraints which naturally stems from if dynamics are employed. Along with these constraints, mission requirements enforces constraints which comes in the form of safety, inspection quality, and duration. Constraints are categorized into two types, they are Hard and Soft constraints. Hard constraints are the constraints that ought to be satisfied to consider a solution feasible. Soft-constraints can be compromised during the search for the solution and it will not lead to in-feasibility. In the following paragraphs different constraints involved are described.

- **Hard constraints**

  Control constraint

  To make the translation as well as attitude trajectories feasible. They should satisfy the control constraints. For translation, as low thrust propulsion is used, the maximum thrust becomes critical. Only maximum thrust magnitude condition is considered in this work,

  $$\|T_i\| \leq T_{max} \quad i = 1...3 \tag{4.1}$$

  For attitude motion, the actuators ability to produce the required torque acts as bottle neck for slew and tracking maneuvers during observation.

  $$\|\tau_i\| \leq \tau_{max} \quad i = 1...3 \tag{4.2}$$

  For the thesis, only the max torque constraint is considered and constraints such as maximum momentum storage or saturation condition is assumed to be counteracted by low thrusters during the attitude segment of the trajectory.

  Collision avoidance

  One of the most important of constraints are collision avoidance constraints. It is catastrophic if a spacecrafts that are doing inspection ends up colliding among each other or with the target.

The collision avoidance with target named as C1 can be accomplished by defining a keep out of zone sphere which also takes into account the uncertain environment surrounding the target. This constraint is given by

$$(r_i - r_T) \notin B_{C1} \qquad \forall i \in N_{Satellite} \tag{4.3}$$

where, $B_{C1} = \{z \mid \forall z, \|z\| \leqslant R_c\}$ is collision ball defined with radius R_c. Next collision avoidance constraint named as C2 is between the inspecting spacecrafts. This can be done by checking for collision avoidance at every instant. This check forms a computationally heavy process and there are methods such as convexification of the dynamic obstacle region [37] and making use of splines for shape based method to make this checks faster [36]. However, we are using a simple check at every instant even though it is costly which basically hints that we are assuming collision checking as a black box in the thesis.

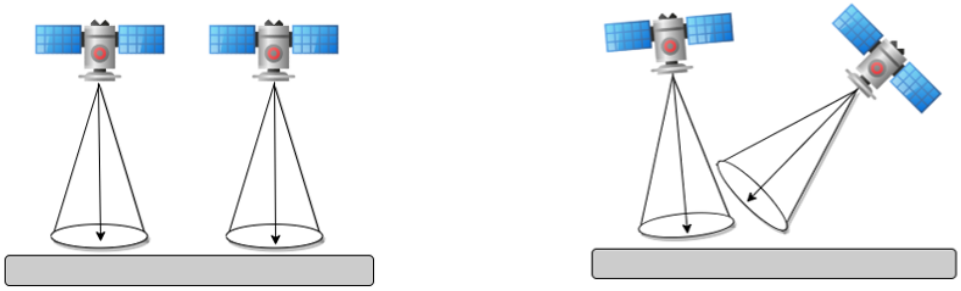
$$(r_i - r_j) \notin B_{C2} \qquad \forall \quad i, j \in N_{Satellite} \quad but \quad i \neq j \tag{4.4}$$

where, $B_{C2} = \{z \mid \forall z, \|z\| \leqslant R_m\}$ for some radius R_m

- **Soft constrains**

  Pointing avoidance

  Pointing avoidance essentially tries to make the spacecraft point their inspection payloads at different desired directions. This will avoid repeatability of covering the same surfaces by two or more satellites at a time. Figure 4.3 shows when this condition will occurs and when it will not occur. This constraint is related to maximizing the coverage, rather than making sure that the trajectories produced is feasible or not. Because of this reason it is considered as one of soft constraints.



(a) No Pointing interference between the satellites,pointing avoidance check will produce false.

(b) Pointing interference between the satellites, pointing avoidance check will produce true.

Figure 4.3: Pointing Avoidance

Inspection duration T

Inspection duration is the maximum time allowed to do inspection. Usually this parameter is handled by the mission requirement. But once it is fixed, it acts as bottle neck on the time available for low thrust translation and observation trajectories (see section 4.3).

- ***Search Space constraint***

  Search Space constraint is used especially for SBMP algorithm. The sampling process is limited inside these region. Search Space constraint takes input from the C1 collision avoidance constraint and Mapping quality requirement i.e., resolution quality. Resolution requirement comes from the mission requirement and payload specifications; which essentially informs the maximum distance inside which the camera can take pictures with required resolution.

4.2.2 Steering problem

One of the main issues related to motion planning algorithms applied for differential systems are the Two point boundary value problems(TPBVP). As pointed out in section 2.1, solving TPBVP is computationally demanding. If TPBVP solver is computationally costly, together with search algorithm, the motion planning algorithm will take lot of time to solve any given problem. So, the aim of this section is to present solutions that are chosen for the framework which are suitable for SBMP algorithms. TPBVP problems are also called Steering problems as the aim is to steer from one point to another point.

Translation motion

As compared to its counter-part "high Impulse trajectories", low thrust transfer trajectory optimization is an inherently difficult problem which does not have analytical solution till date [54]. Direct and Indirect methods[5][14] can be applied to find the best low thrust trajectories, but with higher computation cost which is not a suitable option for SBMP algorithms. In the literature, the research trend moved in a direction where the low thrust trajectory is solved rapidly using different approximate methods[43][54][58][59] with reasonable assumptions and then the obtained solution is used as an initial guess to solve exactly by using standard optimization methods [[14] Ch.5]. For AMPSGIM framework, the first requirement for the method is to find the feasible trajectories and that these feasible trajectories are produced rapidly. Above requirements are met by a class of low-thrust trajectory solvers called shape based methods [43][58][59][54]. Most shape based methods assume a trajectory shape before hand with reasonable assumption so that they obey the thrust constraints. [43] assumes a exponential sinusoidal shape for planer case with only tangential thrust.[58][59] takes new shape function called inverse polynomial to find the in-plane and out of plane trajectory solutions. Both [43][59] proposed methods suitable exclusively for 2-D problem. [54] provides 3-D solution by using Finite Fourier series in addition to optimizing ΔV required. For

this thesis work, polynomial shape functions are used. The selection of polynomial shape function is motivated by the fact that it is simple and produces trajectories rapidly. This method is applied to 3-D case by parametrizing the relative dynamics co-ordinates by polynomials. The assumed shapes satisfy the dynamic, boundary conditions, and control constraint. In the next section, formulation of the method is presented.

Polynomial Shape based method

The problem of finding feasible low trajectories will be formulated below. The end goal of the formulation is to find the parameters involved in the polynomial shape function. Starting out with the C-W dynamics from section 3.2.

$$\begin{aligned} \ddot{x} - 3n_t^2 x - 2n_t\dot{y} &= u_x \\ \ddot{y} + 2n_t &= u_y \\ \ddot{z} + n_t^2 z &= u_z \end{aligned} \tag{4.5}$$

Each of the co-ordinates are represented as polynomial shape function of degree n.

$$x(t) = f(t), y(t) = g(t), z(t) = h(t) \tag{4.6}$$

where $f(t), g(t), h(t)$, are the polynomial function of degree n. Value of n depends on the boundary conditions and problem at hand. For our purpose, n is chosen to be 3, see equation 4.7. This choice is supported by the fact that, the 3 degree of freedom leads to four coefficients which are used to satisfy the given initial and final boundary conditions equation 4.12. Explicitly writing the shape functions in terms of variable τ which is the normalized time variable to avoid the drastic increase of values due to 3rd power involved in time variable t, $\tau = t/t_f$

$$\begin{aligned} f(\tau) &= a_0 + a_1\tau + a_2\tau^2 + a_3\tau 3 \\ g(\tau) &= b_0 + b_1\tau + b_2\tau^2 + b_3\tau 3 \\ h(\tau) &= c_0 + c_1\tau + c_2\tau^2 + c_3\tau 3 \end{aligned} \tag{4.7}$$

Because of the normalization, while taking the derivative, simple conversion needs to be done to convert from t to τ. for example,

$$\begin{aligned} x(\tau) &= x(t/t_f) \\ \frac{d^n x(\tau)}{d(\tau)} &= t_f^n \frac{d^k x(t)}{d(t)} \end{aligned} \tag{4.8}$$

Boundary conditions are given below for each co-ordinates,

$$\begin{aligned} x(t_0) = x_0 \quad y(t_0) = y_0 \quad z(t_0) = z_0 \\ \dot{x}(t_0) = x_0 \quad \dot{y}(t_0) = y_0 \quad \dot{z}(t_0) = z_0 \\ x(t_f) = x_f \quad y(t_f) = y_f \quad z(t_f) = z_f \\ \dot{x}(t_f) = x_f \quad \dot{y}(t_f) = y_f \quad \dot{z}(t_f) = z_f \end{aligned} \tag{4.9}$$

It can be observed that only position and velocity conditions are added. The acceleration conditions are implicitly taken care in control constraints. In terms of normalized time τ

$$\begin{aligned} x(\tau_0) = x_0 \quad y(\tau_0) = y_0 \quad z(\tau_0) = z_0 \\ \dot{x}(\tau_0) = t_0 x_0 \quad \dot{y}(\tau_0) = t_0 y_0 \quad \dot{z}(t_0) = t_0 z_0 \\ x(\tau_f) = x_f \quad y(\tau_f) = y_f \quad z(\tau_f) = z_f \\ \dot{x}(\tau_f) = t_f x_f \quad \dot{y}(\tau_f) = t_f y_f \quad \dot{z}(\tau_f) = t_f z_f \end{aligned} \tag{4.10}$$

Using the BCs, the parameter involved in the shape function can be found (equation 4.8). Basically it is a system of linear equations with parameters gathered into a vector X and boundary conditions as a column vector on the right **b**. Values obtained by evaluating shape functions at initial and final conditions are in the form of Matrix A equation 4.11. Equation 4.12 shows the Linear system for co-ordinate x(t).

$$A_i X_i = b_i \quad \text{i = 1,2,3} \tag{4.11}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \tau_f & \tau_f^2 & \tau_f^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2\tau_f & 3\tau_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} x(\tau_0) \\ t_f \dot{x}(\tau_0) \\ x(\tau_f) \\ t_f \dot{x}(\tau_f) \end{bmatrix} \tag{4.12}$$

After finding the parameters, the dynamics equation (4.5) is inverted to find the thrust acceleration.

$$\begin{aligned} u_x(t) = F(a_0, a_1, a_2, a_3) \\ u_y(t) = G(b_0, b_1, b_2, b_3) \\ u_z(t) = H(c_0, c_1, c_2, c_3) \end{aligned} \tag{4.13}$$

Equations from (4.5) to (4.13) gives the trajectory and corresponding thrust acceleration when supplied with initial and final boundary conditions. It is considered feasible if it satisfies the thrust acceleration constraint $\vec{u} < u_{max}$. Finally the obtained thrust acceleration is utilized to get fuel

consumption by using equation 4.14

$$m(t) = \frac{m_0}{I_{sp} g_0} \int_{t0}^{tf} \|\vec{u}\| \, dt \tag{4.14}$$

and it is considered as the cost-metric $\mathbf{J}_m(x(t), u(t), t)$ for translation motion. In the equation 4.14, m_0 is the initial mass of the spacecraft, g_0 is the acceleration due to gravity at sea level, and I_{sp} is the specific impulse of the low thrust propulsion system. The cost metric $\mathbf{J}_m$ is used as a objective function to find trajectories which consumes less fuel. Therefore, optimization translation steering problem formulation is given as

$$\begin{aligned} \underset{t_f}{\text{Min}} \quad & \mathbf{J}_m(x(t), u(t), t) \\ & \text{Subject to} \\ & \text{System dynamics:} \quad \dot{x} = f(x(t), u(t), t) \\ & \text{Control feasibility:} \quad u(t) \in U(t) \quad \forall t \in \begin{bmatrix} t_0 & t_f \end{bmatrix} \\ & \text{Time feasibility:} \quad t_f \in \begin{bmatrix} t_0 & T_{max} \end{bmatrix} \end{aligned} \tag{4.15}$$

Above problem does not have prescribed final time. Although there is a threshold on the final time T_{max} (coming from inspection duration). t_f is considered as an independent variable to optimize the fuel cost. **Dynamics** $\dot{x} = f(x(t), u(t), t)$ primarily considered to be C-W dynamics (section 3.2). Solutions to the optimization problem (equation 4.15), acts as steering law for the translation part of search algorithm. The accuracy and robustness of the steering law can be increased by using more advanced solution[20][54], but keeping in mind that the TPBVPs solution needs to be produced rapidly is an important aspect. For AMPSGIM framework, as noticed in section 4.1, the translation trajectory is assumed to be made of two segments. One segment called low thrust segment where the low thrust propulsion operates and another called attitude segment is where observation happens. The procedure of generation of translation trajectories required for above mentioned segments is as shown in figure 4.4. Figure 4.5 demonstrates the translation trajectory generation in action. First

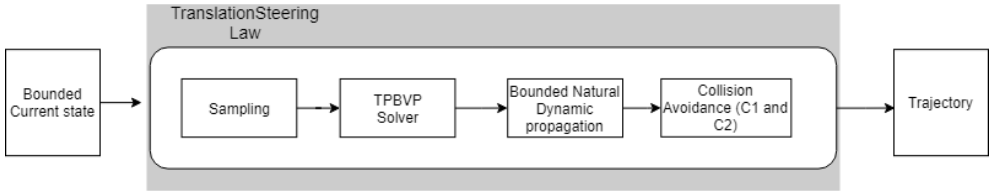


Figure 4.4: Steering law details

sampling procedure finds a point in search space x_{new}. TPBVP produces the solution to connect the bounded current state $x_{current}$ and sampled state x_{new}. This solution is the trajectory under

the control action of low thrust propulsion. After reaching the new point x_{new}, the translation trajectory is propagated under the bounded natural dynamics for the time allocated for attitude segment. This natural trajectory is the segment where the observation occurs. At the end of attitude segment, the whole trajectory is checked for collision avoidance and if there is no collision, the trajectory is considered for the search algorithm. To produce the next translation trajectory the end point of the last observation segment $x_{new_{current}}$ is considered as the initial point and the procedure is repeated. Finally to summarize, figure 4.6 shows the schematics of how the trajectories

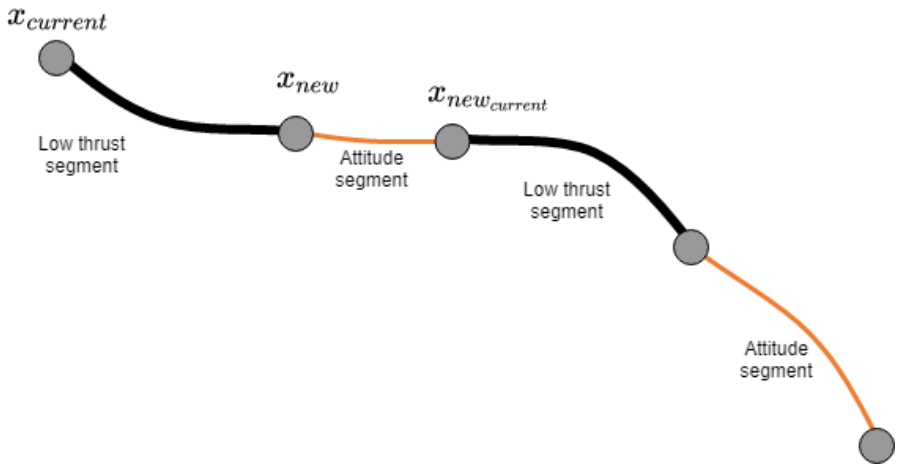


Figure 4.5: Translation trajectory building scheme

are generated for AMPSGIM. In the figure, specifically, for translation steering law block, optimized shape based method is used. The trajectories which are coming out are considered feasible because they satisfy both C1 and C2 collision checks.

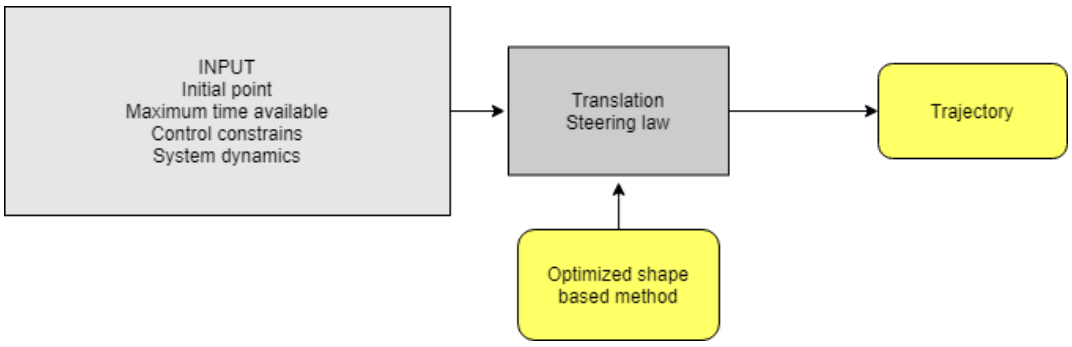


Figure 4.6: Translation Steering law

Attitude motion

For Attitude trajectories, a primitive based motion planning approach was employed taking inspiration from [3][[28]*Ch.14*]. Primitives are the small building block trajectories which are already solved, during the search algorithm several of these primitives are combined together to form large trajectory. Two types of primitives were identified which can provide capability to do essential attitude motions. They are Tracking primitive: spacecraft will be observing the target and Slew primitive: spacecraft performs slew maneuver either to move the camera pointing from one frame to another frame or to shift the control authority from payloads to low thrust propulsion.

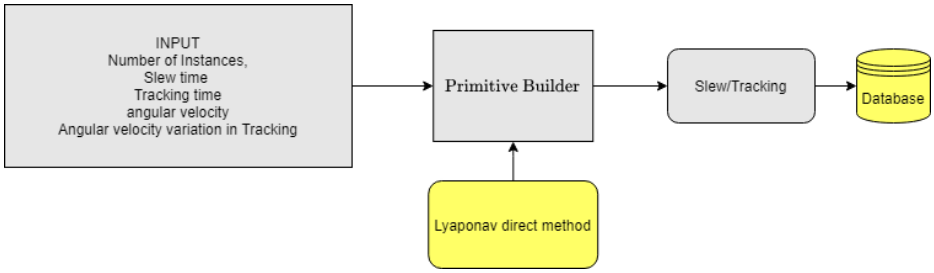
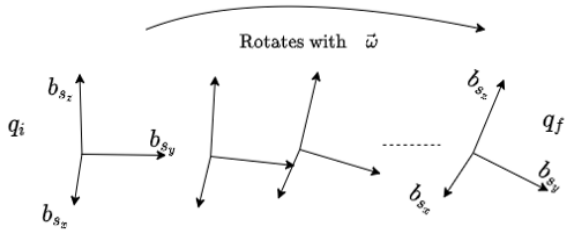
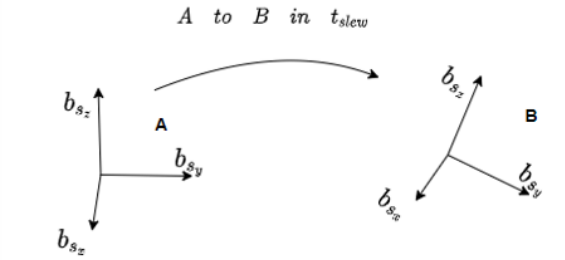


Figure 4.7: Attitude primitive building process

Combining these two primitives, provides the capability to track an object by moving at prescribed angular velocity and to move from one attitude to another. The approach to build the primitives are shown in figure 4.7. The decision to go for building motion primitive is due to the fact that for the case of non-disturbance attitude motion, the attitude trajectories are position invariant. This makes the solution to be obtained before-hand for search algorithm. Because of this reason, high accurate methods can be used to solve attitude TPBVPs. But the limitation comes in in-terms of the storage space required on board. Thus, when the motion primitives are built, it is important to keep in mind the number of instance. This will impact the accuracy and computational effort of the results obtained by search algorithm. Here 'instance' means a slew/tracking primitive applied to move the spacecraft from one attitude configuration to another/moving the current attitude with prescribed angular velocity respectively. Figure 4.8, shows an instance of tracking and slew primitives. Figure 4.8a shows the frame moving from q_i with angular velocity ω ends up at orientations q_f in time $t_{tracking}$. q_f may be equal to q_i. This depends on how the designer wants to built the primitive, the tracking primitive can be built to move from one attitude to another attitude with prescribed angular velocity or it can be made to move from one attitude with angular velocity and return back to the same attitude. The shaping of the angular velocity determines how the tracking primitive moves and it is one of the design parameters. On the other hand, figure 4.8b shows that slew primitive moving the frame A to frame B in given time duration t_{slew}. This forms an instance of the primitives. Like this, N instances of primitives are built with different frames for both tracking and slew and stored to form a attitude primitive database (block **C** in figure 4.1).



(a) Tracking primitive



(b) Slew primitive

Figure 4.8: Attitude primitives

Primitive builder

In this work, Lyaponav direct method is used to build both types of the primitives. Brief description of the Lyaponav function and Ideal control equations[63][22] obtained from it are shown below for both the maneuvers.

For tracking maneuver, Lyaponav function given by equation 4.16 can be used.

$$V = \frac{1}{2}\vec{\omega}_e^T J \vec{\omega}_e + 2k_2 H\left(q_{4e}\right) \tag{4.16}$$

where, $\vec{\omega}_e$ is angular velocity error and $H\left(q_{4\varepsilon}\right)$ scalar function used to represent the quaternion error. Equation 4.17 shows different forms of scalar function H which will avoid the unwinding problem during control application.

$$\begin{aligned} H\left(q_{4e}\right) &= 1 - q_{4e}^2 \\ H\left(q_{4e}\right) &= 1 - \operatorname{sgn}\left(q_{4e}(0)\right) q_{4e} \\ H\left(q_{4e}\right) &= \arccos^2(q_{4e}) \end{aligned} \tag{4.17}$$

This Lyaponav function satisfies the condition that after reaching the equilibrium point $q_e = [0, 0, 0.1]$ and $\omega_e = 0$, $V = 0$. For tracking case, the reference is made to track another frame which is moving with angular velocity ω_d. ω_d can be designed according to the need of the designer as

explained earlier. The angular error between the frames are given by

$$\vec{\omega}_e = \vec{\omega} - A_c(q)\vec{\omega}_d \tag{4.18}$$

where, $A_c(q)$ is the rotation matrix between reference frame and the moving frame parametrized using quaternions. Now taking the derivative of equation 4.18 and making use of the Euler dynamics equations 3.10. we get

$$\dot{\vec{\omega}}_e = \dot{\vec{\omega}} - \frac{d}{dt}\left(A_e(q)\vec{\omega}_i\right) = J^{-1}(J\vec{\omega}\times\vec{\omega}+\vec{u}) - \frac{d}{dt}\left(A_e(q)\vec{\omega}_d\right) \tag{4.19}$$

Now to prove that the function is asymptotically tracks the moving frame, we need to show that function $\dot{V} \leqslant 0$. Thus, taking the derivative of equation 4.16 and substituting necessary terms, we end up getting,

$$\dot{V} = \vec{\omega}_e^T\left(J\vec{\omega}\times\vec{\omega}+\vec{u} - J\frac{d}{dt}\left(A_e(q)\vec{\omega}_d\right) - k_2\frac{\partial H\left(q_{4_c}\right)}{\partial q_{4e}}\vec{q}_e\right) \tag{4.20}$$

To get $\dot{V} \leqslant 0$ condition and noting that $\vec{\omega}_e^T(J\vec{\omega}\times\vec{\omega}) \neq 0$, the control should take the form given by

$$\vec{u} = -k_1\vec{\omega}_e + k_2\frac{\partial H\left(q_{4e}\right)}{\partial q_{4e}}\vec{q}_e + \vec{\omega}\times J\vec{\omega} + J\frac{d}{dt}\left(A_e(q)\vec{\omega}_d\right) \tag{4.21}$$

With this control action, we get

$$\dot{V} = -k_1\vec{\omega}_e^T\vec{\omega}_e \tag{4.22}$$

which is always negative, given that tuning parameter k is positive (or positive definite if it is considered as matrix). Thus, this control has the capability to guarantee global asymptotic stability for tracking maneuver. For simplification, the tuning parameter is assumed to be a scalar for this work. We can utilize the control (equation 4.21) for tracking maneuvers. For slew maneuvers, similar approach is followed but by using slightly different Lyaponav function given by

$$V = \frac{1}{2}\vec{\omega}^T\vec{\omega} + 2k_2H\left(q_{4e}\right) \tag{4.23}$$

where, $\vec{\omega}$ angular velocity of the reference frame. For the slew motion at the end of final condition the angular velocity of the desired frame will not vary with time, as a result $\vec{\omega}_e = 0$. The control which satisfies the $\dot{V} \leqslant 0$ is obtained as

$$\vec{u} = \vec{\omega}\times J\vec{\omega} + \vec{u} - k_1\vec{\omega} + k_2\frac{\partial H\left(q_{4e}\right)}{\partial q_{4e}} \tag{4.24}$$

Above control equation 4.24 is known as Eigenaxis rotation control. Now that the procedure of how to build both primitives are presented, the overall formulation of the primitive builder with control

torque as an optimization function is given by

$$\begin{aligned} \underset{k}{\text{Min}} \quad & \mathbf{J}_\tau(w(t), q(t), k, t) \\ & \text{Subject to} \\ \text{System dynamics:} \quad & \dot{\omega} = \vec{\omega} \times J\vec{\omega} + u \\ \text{System Kinematics:} \quad & \dot{q} = \mathcal{R}(q)\omega \\ \text{Control feasibility:} \quad & u(t) \in U(t) \quad \forall t \in \begin{bmatrix} t_0 & t_f \end{bmatrix} \\ \text{Time feasibility:} \quad & t_f \in \begin{bmatrix} t_0 & t_f \end{bmatrix} \end{aligned} \tag{4.25}$$

where, $\mathbf{J}_\tau(w(t), q(t), k, t) = \sum(\|\vec{u}\|)^2$. The tuning parameter k which appears in 4.21 and 4.24 as k_1 and k_2 is used as an optimizing variable. The important point to be noted is that both tracking manuever and slew manuevers are built without any knowledge of the target. The frames considered for tracking primitives are imaginary frames, similarly for slew primitives. Once these primitives are built, they form a fundamental blocks which can be combined together to track or slew with respect to the requirement of the Inspection mission.

Some examples of tracking and slew primitives built by these controls are shown below.
Tracking: Table 4.9 shows an instance of an tracking primitive with angular velocity shape

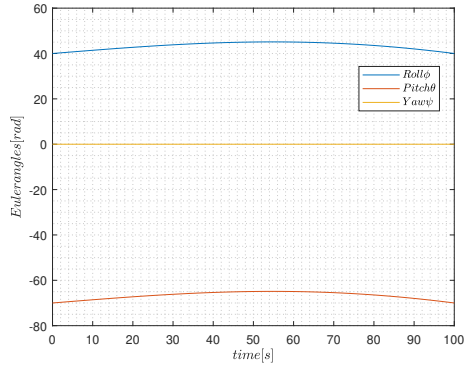
$$\omega_{shape} = w_{frame} \cos\left(\frac{\pi t}{2t_{track}}\right)$$

figure 4.9a shows corresponding pitch, roll, and yaw motion of the frame that needs to be tracked by the reference frame. Once the reference frame is made to follow this imaginary frame. An instance of a tracking primitive is built. This methodology is repeated to form several instances of tracking primitives. The attitude motion associated with tracking maneuver is shown in figure 4.9. As expected the quaternion error is close [0 0 0 1] at every instant of time which verifies that tracking maneuver tracks the prescribed imaginary frame with reasonable accuracy.

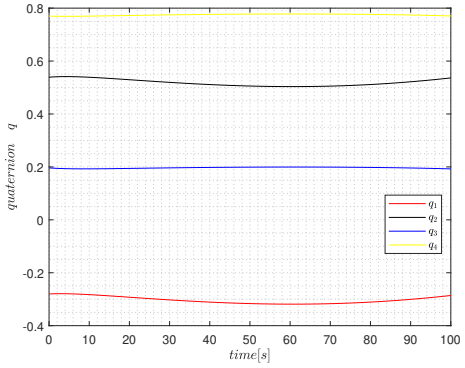
Table 4.1: Tracking primitive example

Tracking Primitive	
q_i	[-0.2802,0.5390,0.1962,0.7698]
$\omega[rad/s]$	[0.0025 0.0025 0]
q_f	[-0.2802,0.5390,0.1962,0.7698]
$t_{tracking}(s)$	100

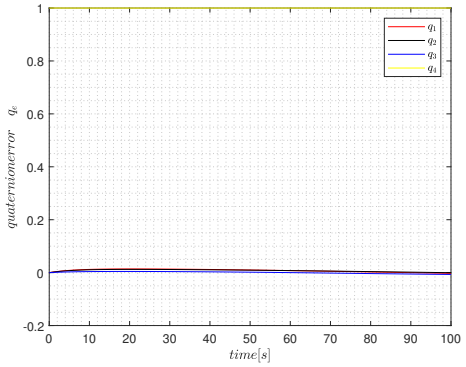
Slew: This maneuver is built to move from one attitude to another attitude frame. Table 4.10 shows parameters for an instant of slew case and results associated with it are shown in figure 4.10. figure 4.10 shows the angular velocity, control torque, quaternion, and quaternion error associated with slew maneuver. As expected the quaternion error goes to [0 0 0 1] verifying that slew manuever is



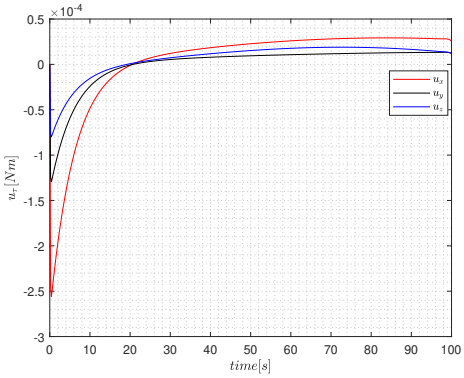
(a) Tracking: Euler angle variation



(b) Tracking: Quaternion variation



(c) Tracking: Quaternion error



(d) Tracking: Control torque

Figure 4.9: Tracking primitive example

successful.

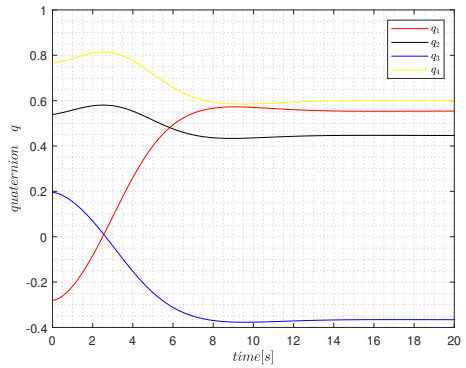
AMPSGIM Framework assumes a sequence of maneuvers to build the attitude trajectories. For example, a simple sequence with tracking and slew primitive with target is shown in figure 4.11.

4.2.3 Coverage objective function and calculation model

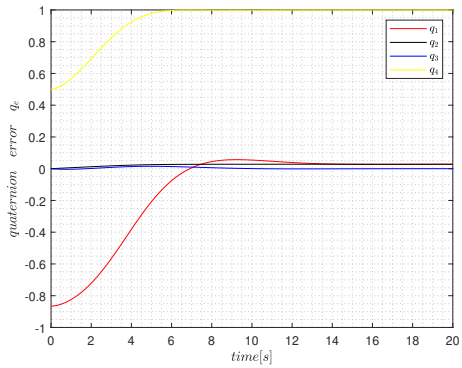
Minimizing the unmapped surfaces of the target forms the most important aspect of the Inspection mission. Coverage is at its simple form keeps track of surfaces that are observed. It is a function of both translation as well as attitude motion, which makes it a complex objective functional. Since the optimization problem is of minimization type, objective function related to complimentary of the coverage called Coverage Compliment C is used.

Table 4.2: Slew Primitive example

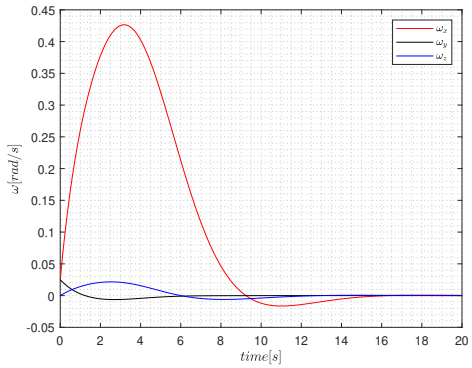
Slew Primitive	
q_i	[-0.2802,0.5390,0.1962,0.7698]
$\omega_i[rad/s]$	[0.0025 0.0025 0]
q_f	[0.5265 0.4394 -0.3687 0.6275]
$t_{slew}(s)$	20



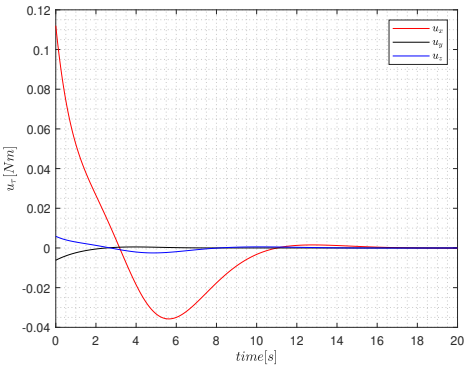
(a) Slew: Quaternion variation



(b) Slew: Quaternion error



(c) Slew: Angular velocity variation



(d) Slew: Control torque

Figure 4.10: Slew primitive example

The formulation of coverage problem is given by

$$
\begin{aligned}
&\underset{x(t),q(t),\omega(t)}{\text{Min}} \quad C(x(t),q(t),\omega(t),u(t),t) \\
&\text{Subject to} \\
&\text{System dynamics:} \quad \dot{x} = f(x(t),u(t),\omega,t) \\
&\text{System Kinematics:} \quad \dot{q} = \mathcal{R}(q)\omega \\
&\text{Control feasibility:} \quad u(t) \in U(t) \quad \forall t \in \begin{bmatrix} t_0 & T_{max} \end{bmatrix} \\
&\text{Time feasibility:} \quad t \in \begin{bmatrix} t_0 & T_{max} \end{bmatrix}
\end{aligned}
\tag{4.26}
$$

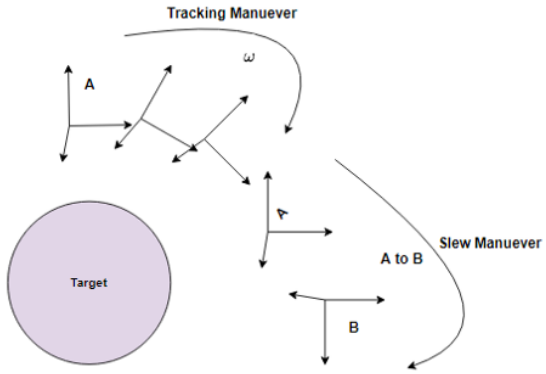


Figure 4.11: Simple Attitude sequence

It can be observed that Coverage compliment C depends on translation trajectory and attitude trajectories. $x(t), q(t), \omega(t), t$ are the optimizing variables. The variable t should lie below a threshold T_{max} which comes from mission design constraint. Solution to equation 4.26 optimal only in terms of Coverage, it may not be optimal for fuel cost and attitude control torque. Whereas the Inspection problem defined in section 1.3, is the combination of all three equation 4.15,equation 4.25, and equation 4.26 to include all the factors to get best possible solution for Inspection mission. Next, a simple method used for coverage calculation is presented

Coverage is estimated by using a simple method. Target is assumed to be of spherical shape and built with n-number of triangulated surfaces with points at the centre as shown in the figure 4.12. The surface is considered covered if the point linked to the surface falls inside Field of the View (FOV) of the spacecraft camera. This method's accuracy depends on the number of points on the target, but large number of points leads to high computation. So trading-off is required between the accuracy of the coverage calculation and computation over-head while modelling the target.

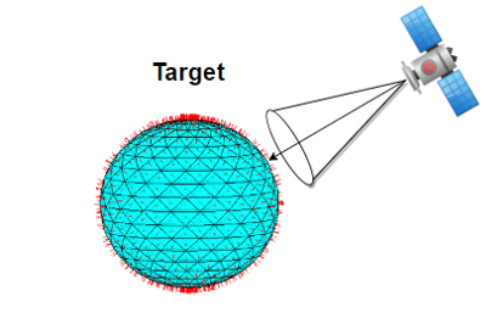


Figure 4.12: Coverage Calulation

4.2.4 Pareto front solution

Path planning described in chapter 3, talks only about optimization of single objective function. But the inspection problem that poses multiple objective functionals (equation4.15,equation 4.25, and equation 4.26), which needs to be optimized to get a best possible solution (see section 1.3). The simplest process of working with multi-objective problem is to scalarize the objectives, where the objectives are multiplied by relative weights decided based on the importance of that particular objective functional and summed together to get a single scalar value. The problem with this method is that deciding the relative weights becomes non-trivial and solutions cannot be differentiated as dominant and non-dominant solutions. A method proposed in [45], provides a way to find dominant and non-dominant solutions by using the Pareto front solution method. Considering dominant solutions leads to finding of best solutions among the set of candidate feasible solutions.

Pareto Front Solution Method

Before looking at the method, Basic definitions relevant to multi-objective optimization are in order.

Definition 3. Non-Dominated and and Dominated Points: A vector of objective functions, $F(x^*) \in Z$, is non dominated iff there does not exist another vector, $F(x) \in Z$, such that $F(x^*) \leqslant F(x)$ with at least one $F_i \leq F_i(x^*)$. Otherwise, $F(x^*)$ is dominated.

Definition 4. Pareto Frontier: The set of all the non-dominated solutions is called as Pareto Frontier.

Consider X_{near}, a set containing 'n' number of candidate points. For each $x_j \in$X$_{near}$, corresponding cost vector is given by $P(x_j)$. Cost vector P is a vector composed of m objective costs. $x_j \in$X$_{near}$ is said be dominated solution $P(x_j) \prec P(x_i)$, if

$$x_i \prec x_j \Leftrightarrow \forall k, p_k\left(x_i\right) \leq p_k\left(x_j\right); \text{ where } 1 \leq k \leq m \quad i \neq j \tag{4.27}$$

where p_k are individual cost vectors present in cost vector P. Therefore the set of dominated nodes D_x is defined as

$$D_x = \left\{x_j \in X_{\text{near}} \mid \exists i \quad x_i \prec x_j\right\} \tag{4.28}$$

To get a good solutions, we only consider candidates which has non-dominant solution.

$$PF_X = X_{near}/D_x. \tag{4.29}$$

Once Pareto Front is obtained, the minimum point among them can be found by following steps,

1. Find the minimum cost tuple L^* - which consists of minimum cost of the each objective function

of the candidate points in PF.

$$L^* = \left(\min_{1 \leq i \leq |PF_x|} l_1 \left(x_i \right), \ldots, \min_{1 \leq i \leq |PF_X|} l_n \left(x_i \right) \right) \tag{4.30}$$

2. Calculate the arc-cost from the candidate solution to the new node x_{new}. Now follow the same procedure as step 1 to find the minimum arc-cost tuple C^* for candidate points.

$$C^* = \left(\min_{1 \leq i \leq |PF_X|} \min_{1} \left(x_i, x_{new} \right), \ldots, \min_{1 \leq i \leq |PF_X|} c_n \left(x_i, x_{new} \right) \right) \tag{4.31}$$

3. Find the best minimum candidate point in PF by dividing each objective function cost of each candidate by the minimum cost of that particular objective function among the candidate points and take a summation as shown in equation 4.32. α_i takes values between $[0, 1]$ and is used to prioritize the objective function, it will not influence the selection of best candidate point.

$$x_{min} \longleftarrow \quad \underset{x_j \in PF_X}{\operatorname{argmin}} \sum_{i=1}^{n} \alpha_i \left[\frac{l_i \left(x_j \right)}{l_i^*} + \frac{c_i \left(x_j, x_{ncw} \right)}{c_i^*} \right] \tag{4.32}$$

This method is named as Min_Pareto in APMGIM framework and employed for two purpose . One is when the parent node is being chosen for next edge addition and another as a general way of finding the best minimum candidate solution among available candidate solutions.

4.2.5 Sampling procedure

Unlike FMT* algorithm for path planning where the sample space is already fixed and FMT* connects to the nearest node as it moves towards the goal direction. The Inspection problem requires sampling in objective functional space to understand the direction of search for the IFMT* algorithms. But sampling in objective space is not possible, as it is not direct to understand the behavior of objective functions. Due to this reason, sampling is carried out in $\mathcal{R}^3$ and $\mathcal{SO}(3)$ and are related to the objective functions which are living in complex objective functional space. Furthermore, connecting of nodes which are near frontier node in the $\mathcal{R}^3$ may not provide a good search direction for inspection problem. Because of this, the sampling procedure is slightly modified. The modified sampling method tries to pick samples from the already defined sample space every iteration of search algorithm. These leads to more samples which in turn increase the possibility of finding a good solutions in objective functional space. Overall following this procedure provides some sense of direction for the search algorithm to follow through to find the solution. Similar idea is followed for SO(3) Sampling.

Characteristics of Samples

$IFMT^* - I$ deals with translation motion which includes a state vector of six dimension. For the

framework the sample points are in $\mathcal{R}^3$. These sample points are restricted to satisfy bounded trajectory conditions equation 3.8 and equation 3.9. Because of this, by using a sample in $\mathcal{R}^3$, velocity can be calculated. Due to this dependency and the requirement to have bounded orbits, the sampling is carried out in $\mathcal{R}^3$. Total sample size for IFMT*-I can be increased based depending on the problem needs.
$IFMT^*-II$, as this block deals with attitude, the sampling can be done using quaterniona[[28] *Ch.*5.2.2]. But primitive database has been built prior to the application of AMPSGIM. Primitive database has finite discrete number of available samples. Thus, each sample is now a Integer number which basically corresponds to a specific state in SO(3). The database acts as mapping function S that converts the SO(3) into Integer value.

$$S\colon \mathcal{SO}(3) \longrightarrow \mathbb{Z}$$

The main issue with this is the accuracy of the solutions. If the discretization is increased the size of the database will increase considerably. Thus, trade-off is required when deciding between the accuracy of the solution and On-board storage.

4.2.6 Neighborhood Reachability and Cost to go

Let x be the node belongs to tree T considered for expansion. Node x can reach only certain number of nodes which are inside its neighborhood. This characteristics of neighborhood is encapsulated by the term called Reachability. Reachability defines the connecting radius r_n which was seen in *NEAR* block of the FMT* algorithm (see section 2.4.2). Connecting radius forms an important part of the FMT* Algorithm as it dictates which nodes to be considered for further addition to the tree. For path planning problems (section 2.1), where the aim is to reach a goal point, the connecting radius can be a metric function such as distance. For our problem, the goal is to find good coverage with best possible solutions for Inspection problem . This entails defining connecting radius for search algorithms in terms of cost to go. Cost to go basically means how much cost it will take to move from current node to another new node.

Definition 5. The reachability set X from state x_0 is the set of all states x_f that are reachable from x_0 such that cost to reach x_f $J(x_0, x_f)$ is less then the threshold

$$R(x_0, \bar{J}) = \{x_f \in X \mid \forall x_f, J(x_0, x_f) \leqslant \bar{J}\} \tag{4.33}$$

As seen in section 1.3, there are three objective functions. Thus three connecting radius are defined for the framework.
Firstly, for the $IFMT^* - I$ connecting radius is defined for translation motion. $\bar{J_m}$ represents the fuel cost as we are dealing with translation trajectory. $\bar{J_m}$ value can be estimated empirically by

knowing the maximum thrust magnitude of the Electric thruster used and by defining a design parameter which can be varied to vary the threshold value of $\bar{J_m}$. From section 4.2.2, Threshold value $\bar{J_m}$ can be estimated as a function of γ, a design parameter and maximum thrust acceleration magnitude u_{max}.

$$\bar{J_m} = m(t) = \gamma \frac{m_0}{I_{sp} g} \int_{t0}^{tf} \|\vec{u}_{max}\| \, dt \tag{4.34}$$

Now for the $IFMT^* - II$ connecting radius J_n is a vector composed of $[\bar{J_\tau} \ \bar{J_c}]$ corresponding to torque cost and coverage compliment respectively. For torque cost J_τ, a parameter β is defined as design parameter. By knowing the maximum torque magnitude of the actuator τ_{max}. The connecting radius $\bar{J_\tau}$ can be found and is given by equation 4.35. $\bar{J_c}$ can be controlled based on the strict requirement of coverage. The lower bound for $\bar{J_c}$ is zero and maximum value can 1 (assuming that coverage is in the normalized form) equation 4.36.

$$\bar{J_\tau} = \beta \int_{t0}^{tf} \|\vec{\tau}_{max}\| \, dt \tag{4.35}$$

$$\bar{J_c} = \epsilon \tag{4.36}$$

where, ϵ is yet a another design parameter which can be varied according to the need of the problem. ϵ lies between 0 and 1. A value of ϵ close to 1 will force the search algorithm to search for good coverage edges for the tree. Moving forward, connecting radius $\bar{J_m}$,$\bar{J_\tau}$,$\bar{J_c}$ are represented as r_{n_m},r_{n_τ},$r_{n_{cov}}$ respectively.

4.3 AMPSGIM algorithm

Section 4.2 has presented important adaptations and modifications that are done on FMT^* algorithm to be suitable for Inspection problem. In this sections, the pseudocode of the framework are shown with description of important blocks involved. The sections ends with detailed explanations on the work flow of the algorithm.

4.3.1 AMPSGIM input parameters

AMPSGIM framework requires numerous parameters to be set up before applying it for any problem. In this section, we will look at various parameters involved and what exactly their impacts are on the framework. Table 4.3 summarizes all the parameters that needs to be given as an input to the framework. Firstly, Let us look at the **General Parameter** category. As first parameters, the minimum information about target i.e., maximum length and keep out zone radius needs to be known. These two parameters influence the search space for the $IFMT^* - I$ translation block. Next

is the duration of the inspection, which acts as a threshold for trajectories which are generated. With Inspection duration, the number of segments for the whole framework can be fixed. Another parameters which will control the framework is the number of satellites. More number of satellites may have desired effects by covering more surface in short span of time. However, more satellites means the algorithm shall solve more TBVPs and checks for collisions and pointing avoidance. Thus number of satellites needs to be chosen sensibly by taking into account the target size, the duration of inspection and the resolution that is needed. If target size is small it will be easy for one satellite to cover the surfaces, on other hand larger targets require more time for a single satellite to cover the surfaces, in that scenario, multiple satellites are preferred. If the duration is long, a single satellite can be sufficient to cover the large number of surfaces. Resolution forms an important factor when the image is taken, inspection payloads have range for its operating conditions where they are designed to work optimally and produce high quality data products (for example visual and thermal images, etc.). This forms a limit on how far away a satellite can go to observe the target. If resolution or quality of image is not a constraint, just by having a single satellite at suitable distance from the spherical target, almost 42.4% of the surfaces can be seen. However, covering 42.4% is not enough for inspection mission to be successful, the quality shall be satisfied as well. Thus resolution acts as a constrain to limit the search space and tries to provide a suitable distance for payloads to work and obtain high quality images. During the application of AMPSGIM, sampling is constrained to be inside the search space defined by target size and resolution requirement. Search space is divided into regions and sampling is done inside it to find the trajectories which satisfy the search space constraint $[r_{min} \quad r_{max}]$. Last parameter in General Parameter category is, the database. As explained in section 4.2.2, database forms a core part of $IFMT^* - II$ and the properties of the database are controlled by 4 parameters. Firstly, total number of instances m_{total}, controls the accuracy and computation aspect of $IFMT^* - II$. The rest of the parameters impact the efficiency of the coverage. Particularly, once the database is fixed, tracking and slew time are fixed as well, which influences the number of observation segment in the framework.

A attitude segment in a single trajectory (figure 4.2) is made of number of observation segments. Each observation segment is combination or sequence of slew and tracking maneuvers as shown in figure 4.13 . In the figure, we can see that each rectangular block has node numbers (from 1 to (n+1)) except for the blocks which are responsible for transition between low thrust segment and attitude segment. Every first observation segment is made of $slew + tracking + slew + tracking$ and last observation segment (nth line in the figure 4.13) is made of $slew + tracking + slew$. Except these two segments, the rest of the observation segments are of the $slew + tracking$ form. First observation segment sequence has slew in the beginning because this is when the low thrust segment ends and attitude segment begins. During this transition, the attitude control authority of the spacecraft shifts from providing pointing for electric thrusters to providing pointing to payloads for observation phase. This transition is accomplished by slew maneuver at the beginning. Similar requirement

Table 4.3: AMPSGIM Input Parameters

Input parameters	
General Parameter	*Paramters description*
Target	Dimensions and keep out zone (KOZ) radius
Number of satellites	N
Inpection duration	Decides the maximum time allocated for Inspection
Search Space $[r_{max}, r_{min}]$	Decides the maximum size of the search space for sampling in translation motion
Number of segments	Decides number of trajectory segments based on the allocated Inspection duration
Number of low thrust segment	T_{size} : Decides how many Low thrust segments are required
Number of observation segment	A_{size} : Decides how many observation segments an attitude segment can have
Database	Attitude database built prior to the search algorithm, below these four parameters decide the property of the Database
Tracking angular velocity	ω
Tracking time	$t_{tracking}$
Slew time	t_{slew}
Total number of instances m_{total}	Maximum number of samples available for IFMT* - II
Translation	*IFMT* -I*
n_{total}	Total number of sample available in the Search Space (for IFMT* -I)
n_{sample}	Number of samples at every step
r_n	Connecting radius
Attitude	*IFMT* -II*
m_{total}	Total number of instances - Decided by th Database
m_{sample}	Number of samples at every step
r_n	Connecting radius
Sampling method	Determistic or Pseudo Random

during second transition: from payload pointing to electric thruster pointing led to inclusion of slew maneuver for the last observation segment. The structuring of observation segments may be designed differently with tracking coming before slewing, but it has to take into account the transition that occur between the low thrust segment and attitude segment. Summation of the time required for both low thrust and attitude segments cannot exceed the inspection duration. Accordingly the various segment numbers can be structured.

Assumption. Attitude motion linked with the low thrust segment is not considered in this work. Also the transition occurring (Slew maneuver) between low thrust segment and attitude segment is assumed as black box and not considered for the calculation. Only the attitude motion involved in the observation are considered.

Moving on to next category of parameters. Both **Translation** and **Attitude** category have similar parameters and have similar influence on their respective search algorithms. n_{total} and m_{total} represents total sample size available for search algorithm to search. For attitude, this number m_{total} is fixed as the total number of instances of the database cannot be changed. n_{total} can be changed

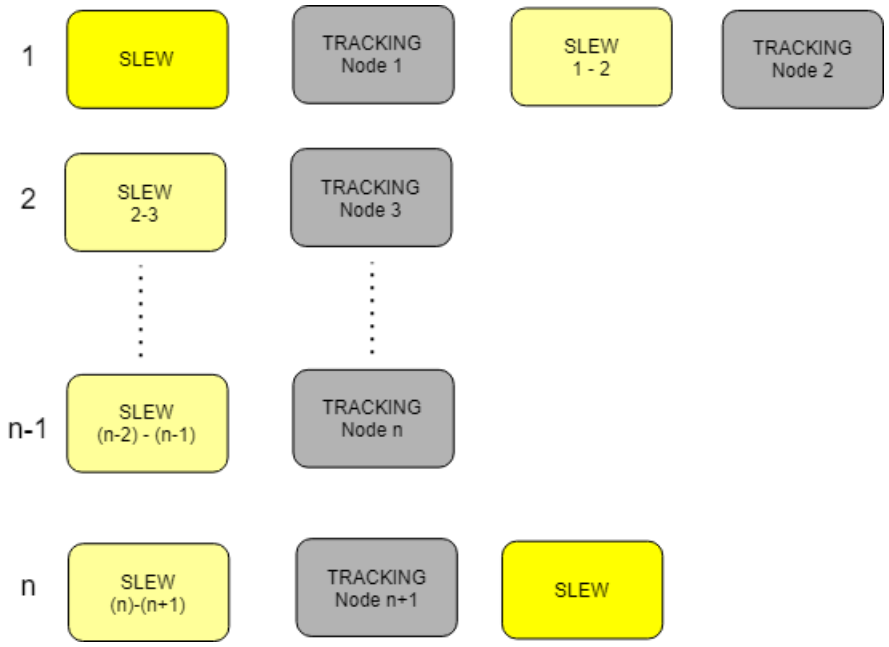


Figure 4.13: Observation segment sequence structure

according to the the problem needs. n_{sample} and m_{sample} are important parameters in $IFMT^*$ family of search algorithms. These two parameters help the search algorithm to sample the algorithm every iteration unlike FMT^*, this is done to facilitate finding best sample points corresponding to objective function space as pointed out in section 4.2.5. Both categories have connecting radius r_n which influences greatly how search is carried out (refer to section 4.2.6). Finally, sampling method type influences how the searching is carried out (see section 2.3.2). Deterministic type is used if the validation and verification is required and for real time implementation they are the go to sampling methods because of their reliability. On the other hand, Random sampling method is fast and has the possibility to produce the better results due to inherent randomness involved .

4.3.2 Building block description

Similar to the building blocks description of FMT* algorithm (section 2.4.2). The IFMT*-I and IFMT*-II have the similar structure but with new modifications and additional blocks as per the adaptation requirements (section 4.2). Following list describes the modified and addition blocks.

- *Initialize_Smart*:
  Determines the initial sample or condition for the satellites. It is called 'smart' because it takes into account the previous satellites coverage maps and tries to find a initial point which does not coincide with previous satellite's initial conditions or current satellite's previous initial conditions and falls near the region where the number of uncovered surfaces are high.

- $Initialize\_Translation(n_{sample}, n_{total})$ and $Initialize\_Attitude(n_{sample}, n_{total})$:
  This functions initializes the first node and finds the best possible adjacent node to the initialized node to form first edge for the tree T.

- $Sample\_translation(n_{sample}, n_{total})$:
  This function samples the search space made of n_{total} and returns $n_{samples}$.

- $Sample\_attitude(m_{sample}, m_{total})$:
  This function samples the sample space made of m_{total} in attitude database and returns $m_{samples}$.

- Min_Pareto:
  Finds the pareto optimal point among the candidates points (section 4.2.4).

- $Collision\_Checking$:
  Returns 1 if the satellite's translation trajectory is coinciding with other satellite's trajectory.

- $Traj\_Generation$:
  Solves translation TPBVP and returns the safe trajectory which lies inside the search space, see section 4.2.2.

- Coverage:
  Calculates the number of covered surfaces by the attitude segment and returns the Coverage compliment.

- Termination:
  Termination function is present in three important blocks. In each block the condition for termination differ. For $IFMT^* - II$ termination condition is usually a limit on searching for attitude trajectories. If the search algorithm does not find any feasible adjacent node with in threshold iteration limit, termination condition returns **1**. Regarding $IFMT^* - I$, the termination condition returns **1** if the $IFMT^* - II$ fails to return the feasible attitude trajectory with in the threshold limit. Only when these two algorithm returns failure, AMPSGIM's termination returns failure to that current iteration and finds new initial point for the satellite using $Initialize\_Smart$ function. If initiation does not improve beyond a threshold iteration limit, the algorithm will report failure and will stop.

4.3.3 Working

In this section, a detailed guide through of how AMPSGIM algorithm works is described. Firstly, prior to running the algorithm all the inputs shall be initialized as explained in section 4.3.1. The Main block of the algorithm is $AMPSGIM\_MAIN$ which encapsulates $IFMT^* - I$ and $IFMT^* - II$. To make it more understandable, these algorithm blocks are explained separately by assuming inputs

and outputs are readily available from each other. There are three datasets used throughout the algorithm: A1 stores the translation related data. A2 stores attitude related data, and C stores target information, maximum coverage, Inspection requirements.

- ***APMSGIM_MAIN***
  The algorithm starts by initializing variables {Sat,Map,Mapdatabase,$N_{satellite}$}. Sat and Map stores the trajectory and coverage details respectively. Mapdatabase stores all the data related to satellites as well as the target surface coverage details. Firstly, the algorithm finds the initial position value for the first satellite and stores it in the translation data set (A1). With this, the algorithm enters the iteration. At line 7, $IFMT^* - I$ receives the inputs and returns the translation and attitude details ($T_{translation}, T_{attitude}$) respectively. Satellite trajectory and mapping details (Sat, Map) are extracted from ($T_{translation}, T_{attitude}$) by using *Extract_MAP* at (line 8) and are stored in Mapdatabase. Next If $IFMT^* - I$ failed to obtain good trajectory, (line 10) Termination condition (see section4.3.2) will determine whether to re-initiate the same satellite (line 12) or (line 11) report failure. If termination returns zero, then the coverage is checked to see if it is higher than threshold value ($Max\_coverage$). If yes, the algorithm successfully completed the Inspection mission. If No, the Algorithm chooses the new satellite and finds new initial condition using *Initialize_smart* and runs the iteration until the $Max\_Coverage$ is reached. At the end, the algorithm returns Mapdatabase which has all the details of every satellite and this solves our Inspection problem (section 1.3).

- **IFMT*-I**
  This is one of the core algorithms of APMSGIM. This block takes care of translation search and in providing translation trajectories to the $IFMT^* - II$. The algorithm receives updated inputs ($A1, A2, C$) from $AMPSGIM\_MAIN$. It starts working by initializing sets such as T,V_{open},$V_{unvisited}$, and a new additional set as compared to FMT^* Algorithm, $NewV_{unvisited}$ is added. The initial condition from A1 is stored to x_{init} ((IFMT*-I: line 4) and to T (IFMT*-I: line 5) . In line 6, *Sample_translation* samples the search space and returns the samples which are stored to $NewV_{unvisited}$. $V_{unvisited}$ is updated by these new unvisited nodes (IFMT*-I: line 7). V_{open} is updated by x_{init} as an initial frontier node. With this information setup, the iteration starts, the condition which controls the iteration is the segment size and it shall be lesser than or equal to T_{size}. As a first step, x_{init} is stored to z. Within this main iteration, z is considered as the frontier node (IFMT*-I: line 11). *Expand* is called, and *Expand* block works similar to Expand block in FMT^* algorithm except for few main differences.

  – The minimum (*Expand*: line 5) is found using Pareto optimal solution by using the MIN_Pareto function (see section 4.2.4).

  – NEAR_Translation has similar structure as FMT* but finding nearest node is totally different as compared to normal path planning FMT^* NEAR block. It receives the current

Algorithm 4.1: *AMPSGIM_MAIN*

Input : Translation{A1} $\longleftarrow \{n_{total}, n_{sample}, T_{size}, \mathrm{r}_{n_m}\}$,
Attitude {A2}$\longleftarrow \{m_{total}, Database, m_{sample}, A_{size}, \mathrm{r}_{n_\tau}, r_{n_{cov}}\}$,{C}

1 $Sat \longleftarrow \emptyset, Map \longleftarrow \emptyset$
2 $Mapdatabase \longleftarrow \emptyset$
3 $N_{satellite} \longleftarrow \emptyset$
4 $y_{init} \longleftarrow Initialize\_smart(N_{satellite}, Mapdatabase)$
5 $A1.yinit \longleftarrow yinit$
6 **while** $N_{satellite} \leqslant Max - N_{satellite}$ **do**
7 $(T, T_{attitude}) \longleftarrow IFMT* - I(A1, A2, C)$
8 $(Sat, Map) \longleftarrow Extract\_MAP(T, T_{attitude})$
9 $Mapdatabase \longleftarrow Mapdatabse \cup (Sat, Map)$
10 **if** $Termination()$ **then**
11 ***return*** *failure*
12 **else if** $Termination()$ **then**
$N_{satellite} = N_{satellite}$
13 $y_{init} \longleftarrow Initialize\_smart(N_{satellite}, Mapdatabase)$
14 $A1.yinit \longleftarrow A1.yinit \cup yinit$
if $Max\_coverage \leqslant Mapdatabase\_Coverage$ **then**
15 $Return \quad Mapdatabse$
16 **else**
$N_{satellite} = N_{satellite} + 1$
17 $y_{init} \longleftarrow Initialize\_smart(N_{satellite}, Mapdatabase)$
18 $A1.yinit \longleftarrow A1.yinit \cup yinit$
19 **return** $Mapdatabase$

node and set V in which neighborhood nodes of current node needs to be found. The iteration starts by taking each x from V and generating a trajectory using *Traj_Generation* function. Generated trajectory T_n is supplied to $IFMT^* - II$ block along with data sets A2 and C. $IFMT^* - II$ returns attitude trajectory (T_attitude) and a variable called *Flag*. Flag =1, indicates feasible attitude trajectory was found for the supplied node x in set V. only if Flag is one and the Trajectory fuel cost is less than connecting radius r_{n_m} (fuel mass) (*NEAR_Translation*: line 4), x will be considered as the nearest node to z. the iteration goes through every node in V and finally returns a set Z_{near}, a collection of neighbouring nodes to z.

After *Expand_Translation* runs, it outputs the updated Tree T. Now at $IFMT^* - I$ line 12 termination condition is checked. If termination condition does not report failure and If $V_{unvisited}$ is empty, *Sample_translation* ($IFMT^* - I$: line 14) samples the search space once again and the obtained nodes are added to $V_{unvisited}$. This one of important difference between $FMT*$ Algorithm in-terms of using the samples. In our case, the $V_{unvisited}$ is never emptied

as the nodes are being added every time the $V_{unvisited}$ is found empty (line 14). Therefore, there is no need to run a separate block to add a new node to V_{open}. V_{open} will always have one open node because of non non-empty $V_{unvisited}$. This may mimick Anytime behavior for the AMPSGIM framework. Now using the Min_Pareto an optimal frontier point is found and stored in z. Iteration runs until it exceeds the segment size and returns, $(T_{translation}, T_{attitude})$ which is the translation and attitude trajectory details.

Algorithm 4.2: IFMT*-I

Input : Translation{A1}
Attitude {A2}
Target {C}

1 $T \longleftarrow \emptyset, V_{open} \longleftarrow \emptyset$
2 $V_{unvisited} \longleftarrow \emptyset, NewV_{univisted} \longleftarrow \emptyset$
3 $T_{initial} \longleftarrow A1.yinit$
4 $x_{init} \longleftarrow$T$_{initial}.xinit$
5 $T \longleftarrow T \cup T_{initial}$
6 $NewV_{univisted} \longleftarrow Sample\_translation(n_{sample}, n_{total})$
7 $V_{unvisited} \longleftarrow V_{unvisited} \cup NewV_{univisted}$
8 $V_{open} \longleftarrow V_{open} \cup \{x_{init}\}$
nl $NewV_{univisted} \longleftarrow \emptyset$
9 **while** $segmentsize \leqslant T_{size}$ **do**
10 $z \longleftarrow x_{init}$
11 $T \longleftarrow Expand\_Translation(T, z, r_{n_m}, A2)$
12 **if** $Termination()$ **then**
13 ***return*** *failure*
14 **else if** $V_{unvisited} = \emptyset$ **then**
15 $NewV_{univisted} \longleftarrow Sample\_translation(n_{sample}, n_{total})$
16 $V_{unvisited} \longleftarrow V_{unvisited} \cup NewV_{univisted}$
17 $\{x_{init}\} \longleftarrow Min\_Pareto(V_{open})$
18 **else**
19 $\{x_{init}\} \longleftarrow Min\_Pareto(V_{open})$
20 **return** $\{T_{translation}, T_{attitude}\}$

- **IFMT*-II**

  IFMT*-II is the second search algorithm which works on the principles of FMT*. This block forms the internal part of IFMT*-I and acts as decider when it comes to trajectories good for the coverage. IFMT*-II works very similar to IFMT*-I. It receives inputs $(A2, C, T_n)$ from IFMT*-I. Where, T_n is the trajectory supplied by the IFMT*-I. The algorithm starts working by initializing sets $(T, V_{open}, V_{unvisited}, NewV_{unvisited})$. The algorithm attaches all the attitudes to the translation trajectory T_n. The function *Initialize_Attitude* tries to find the feasible initial node and finds best possible adjacent node to form a first edge for the

Algorithm 4.3: *Expand_Translation*

Input : $T, z, r_n, A1, A2$
begin
1 $V_{Newopen} \longleftarrow \emptyset$
2 $Z_{near} \longleftarrow Near(V_{unvisited}, z, r_n, A1, A2)$
3 **for** $x \in Z_{near}$ **do**
4 $X_{near} \longleftarrow Near(V_{open}, x, r_n, A1, A2)$
5 $x_{min} \longleftarrow Min\_Pareto(X_{near}, T, x)$
6 **if** $CollisionChecking(x_{min}, x)$ **then**
8 $(V, E, V_{unvisited}, V_{Newopen}) \longleftarrow AddNode((\mathrm{V,E,V}_{unvisited}, V_{Newopen}),x)$
9 $V_{open} \longleftarrow V_{open} \cup V_{Newopen}/\{z\}$
10 **return** $T \Longleftarrow (V, E, V_{unvisited}, V_{open})$

Algorithm 4.4: *Near_Translation*

Input : $V, z, r_n, A1, A2, C$
begin
1 $Z_{near} \longleftarrow \emptyset$
2 **for** $x \in V$ **do**
$T_n - Traj\_Generation(z, x)$
3 $(Tatti, Flag) \longleftarrow \mathrm{IFMT}^* - II(\mathrm{A2,C,T}_n)$
4 **if** $T_n \leqslant r_{n_m}$ $Flag = 1$ **then**
5 $Z_{near} \longleftarrow Z_{near} \cup x$
6 **return** Z_{near}

tree T. *Initialize_Attitude* returns the $Tatti_{initial}$ which includes the two Nodes, an edge and initial frontier node $Tatti.x_{init}$. This node is stored to x_{init} and Tree T is updated by using $Tatti_{initial}$. Now using *Sample_attitude* different quaternion samples are chosen and stored in $NewV_{unvisited}$. This $NewV_{unvisited}$ updates the $V_{unvisited}$ ((IFMT*-II: line 7). V_{open} is updated by x_{init}.

The main iteration starts from storing the x_{init} to z (IFMT*-II: line 10). *Expand_Attitude* is called by sending $(T, z, r_n, A2)$ (IFMT*-II: line 11). Like IFMT*-I *Expand_Attitude* works like Expand block of FMT*, the difference again comes with *Min_Pareto* block which finds the best node by using Pareto optimal solution. Another difference is Pointing Avoidance block which acts like collision check block. Lastly how *NEAR_Attitude* block finds the neighborhood nodes is differently.

Inside *NEAR_Attitude*, the only difference that can be observed as compared to *NEAR_Translation* is the way the cost is calculated. *NEAR_Attitude* calls a function called *Coverage* which finds the coverage for given node x in V and z. If the cost to go is found less than (r_{n_τ},$r_{n_{cov}}$)

Algorithm 4.5: IFMT*-II

Input : Attitude {A2}
Target {C}, T_n

1 $T \longleftarrow \emptyset, V_{open} \longleftarrow \emptyset$
2 $V_{unvisited} \longleftarrow \emptyset, NewV_{univisted} \longleftarrow \emptyset$
3 $Tatti_{initial} \longleftarrow Initialize\_Attitude(m_{sample}, m_{total})$
4 $x_{init} \longleftarrow$Tatti$_{initial}.xinit$
5 $T \longleftarrow T \cup T_{initial}$
6 $NewV_{univisted} \longleftarrow Sample\_attitude(m_{sample}, m_{total})$
7 $V_{unvisited} \longleftarrow V_{unvisited} \cup NewV_{univisted}$
8 $V_{open} \longleftarrow V_{open} \cup \{x_{init}\}$
nl $NewV_{univisted} \longleftarrow \emptyset$
9 **while** *segment size* $\leqslant A_{size}$ **do**
10 $z \longleftarrow x_{init}$
11 $T \longleftarrow Expand\_Attitude(T, z, r_n, Database)$
12 **if** $Termination()$ **then**
13 ***return*** *failure* $\{T_{attitude} = \emptyset, Flag = 0\}$
14 **else if** $V_{unvisited} = \emptyset$ **then**
15 $NewV_{univisted} \longleftarrow Sample\_attitude(n_{sample}, n_{total})$
16 $V_{unvisited} \longleftarrow V_{unvisited} \cup NewV_{univisted}$
17 $\{x_{init}\} \longleftarrow Min\_Pareto(V_{open})$
18 **else**
19 $\{x_{init}\} \longleftarrow Min\_Pareto(V_{open})$
20 **return** $\{T_{attitude}, Flag = 1\}$

Algorithm 4.6: *Expand_Attitude*

Input : $T, z, r_n, A2$
begin
1 $V_{Newopen} \longleftarrow \emptyset$
2 $Z_{near} \longleftarrow Near(V_{unvisited}, z, r_n, A2)$
3 **for** $x \in Z_{near}$ **do**
4 $X_{near} \longleftarrow Near(V_{open}, x, r_n, A2)$
5 $x_{min} \longleftarrow Min\_Pareto(X_{near}, T, x)$
6 **if** $PointingAvoidance(x_{min}, x)$ **then**
7 $(V, E, V_{unvisited}, V_{Newopen}) \longleftarrow AddNode((\mathrm{V},\mathrm{E},\mathrm{V}_{unvisited}, V_{Newopen}),x)$
8 $V_{open} \longleftarrow V_{open} \cup V_{Newopen}/\{z\}$
9 **return** $T \Longleftarrow (V, E, V_{unvisited}, V_{open})$

(torque,coverage compliment) , it will be added to the set Z_{near} which is set of all neighbourhood nodes from node z. Pointing Avoidance (*Expand_Attitude*: line 6), which checks if the nodes are observing the same surface at that particular instant of time as compared previous satellite's pointing. Only if this condition is found to be true, the Tree T is expanded. At the

Algorithm 4.7: *Near_Attitude*

```
Input : V, z, r_n, A2
begin
1   Z_near ⟵ ∅
2   for x ∈ V do
3       cost ⟵ Coverage(z, x, A2)
4       if cost ⩽ (r_{n_τ} r_{n_cov}) then
5           Z_near ⟵ Z_near ∪ x
6 return Z_near
```

end of *Expand_Attitude* block, Expanded tree T is returned. Now at line 12 of IFMT*-II termination condition is checked. If it returns false, the iteration runs with further steps (line 14 to 16) similar to IFMT*-I till attitude segment size exceeds the maximum segment size A_{size}, finally returning the $T_{attitude}$ and Flag =1 . If termination condition is true, the algorithm stops and returns empty Tree $T_{attitude}$ and Flag =0.

Chapter 5

Numerical Simulation and Discussions

"Life is like riding a bicycle.
To keep your balance you must keep moving"
- Albert Einstein

This chapter aims to validate the AMPSGIM framework developed in chapter 4. The validation is carried out via simulation scenarios requiring inspection of targets of varying sizes. Firstly the simulation setup is described which includes the modelling of the spacecraft with necessary real torque and thrust constraints. Next, simulation scenarios are presented along with discussions on the obtained results.

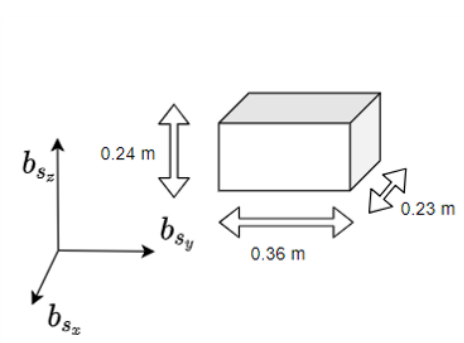
5.1 Spacecraft and Target model

Spacecraft Model

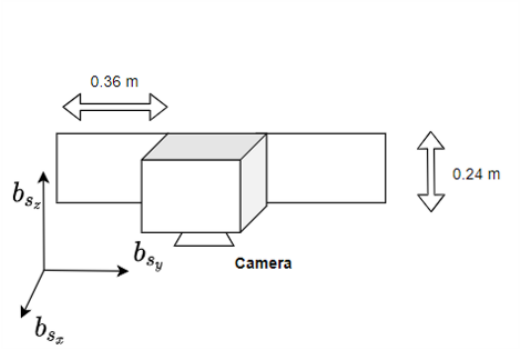
For the purpose of numerical simulation and to respect the motivation mentioned in 1.1, the spacecraft model is assumed to be of cubesat class. In particular a 12U cubesat configuration is assumed with the specifications mentioned in table 5.1. Cubesat is equipped with low thrust propulsion system [7] and attitude control system[47] available in the market. This makes sure that the trajectories which are obtained respects the Cubesat's capability. Figure 5.1 shows the dimensions of the spacecraft and isometric view of an unscaled model of reference Spacecraft. The camera direction is assumed to be aligned in $-b_{s_z}$ direction. While building the attitude primitives 4.2.2, this spacecraft model shall be used. By using this spacecraft model, the results of the attitude primitives are sure to satisfy the control torque constraints.

Table 5.1: Spacecraft Model Specification

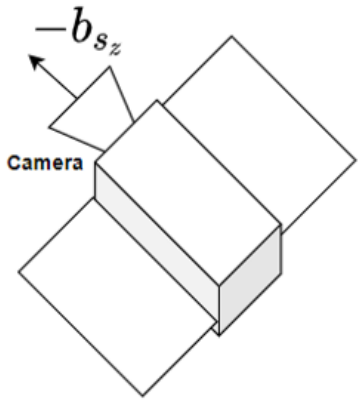
Spacecraft	
Parameters	**Values**
Mass Kg	20
Dimension (l*b*h) m	0.23×0.24 × 0.36
J kgm^2	$\begin{bmatrix} 0.3155 & 0 & 0 \\ 0 & 0.1862 & 0 \\ 0 & 0 & 0.3076 \end{bmatrix}$
J deployed kgm^2	$\begin{bmatrix} 0.3268 & 0 & 0 \\ 0 & 0.1852 & 0 \\ 0 & 0 & 0.3108 \end{bmatrix}$
Thruster thrust T_{max} mN	1.25
Actuator torque τ_{max} Nm	0.112
FOV °	35



(a) Spacecraft Model



(b) Spacecraft solar panel deployed view



(c) Unscaled Isometric view

Figure 5.1: Spacecraft Model

Target Model

Regarding the target, three different sized targets are assumed for the simulation. The keep out zone spheres are defined based on the maximum dimension of target as the diameter plus small arbitrary distance added for safety margin. Three target models considered are

- International Space station [39] is the largest space structure that humans have ever built till date, the dimensions of the ISS are shown in figure 5.2a with keep out zone sphere.
- Envisat is the first largest satellite for ESA dedicated for Earth observation[32]. 5.2b shows the dimensions of the spacecraft with keep out zone sphere.
- Astroid Ryugu[61] is the target asteroid for Hayabusa 2 Mission. This target is considered as its size is larger than artificial satellites. Its maximum dimension and keep out zone sphere is shown in figure 5.3. The chaotic dynamics and other factors involved in complex environment around the asteroid are not treated. Only reason to consider the asteroid is due to its size.

ISS is considered as the primary target for numerical simulation. Other two targets are used to support the simulations to account for applicability of the framework to targets of different sizes.

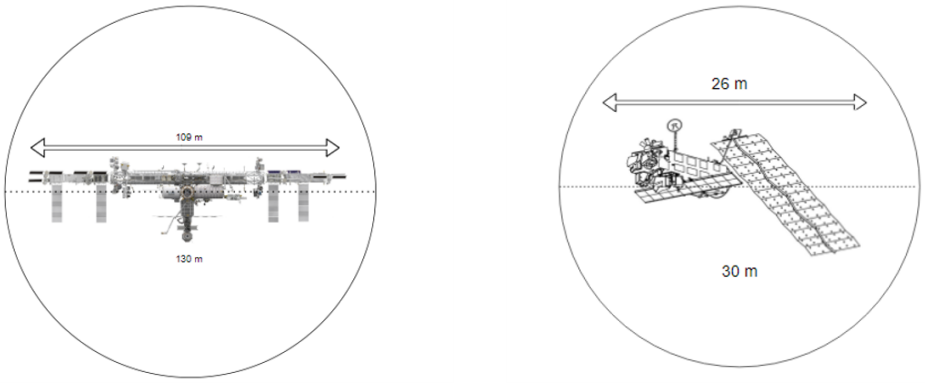
Target orbit parameters are defined as in table 5.2, irrespective of the target model, the orbit is assumbed to remain the same.

Table 5.2: Target Orbit Parameters

Target Orbit Parameter	
a_t **(km)**	6978
e_t	0
${i_t}^\circ$	2
${\omega_t}^\circ$	20
${\Omega_t}^\circ$	10
${\theta_{t_0}}^\circ$	20
$T_{target}(s)$	5.80E+03
h_t (km^2/s)	5.27E+04
n_t	0.0011

5.2 Simulation scenarios

Section 4.3.1, explains what parameters are required for simulation setup. Once the simulation parameters are set, APMSGIM framework can be applied to map the target. Two simulation scenarios are considered for demonstration. Firstly, AMPSGIM is applied to the single satellite inspection problem. Then, multiple satellite inspection problem is treated.



(a) ISS dimension [39] and Keep out Sphere (b) Envisat Image Courtesy: NASA

Figure 5.2: Target category: Satellites

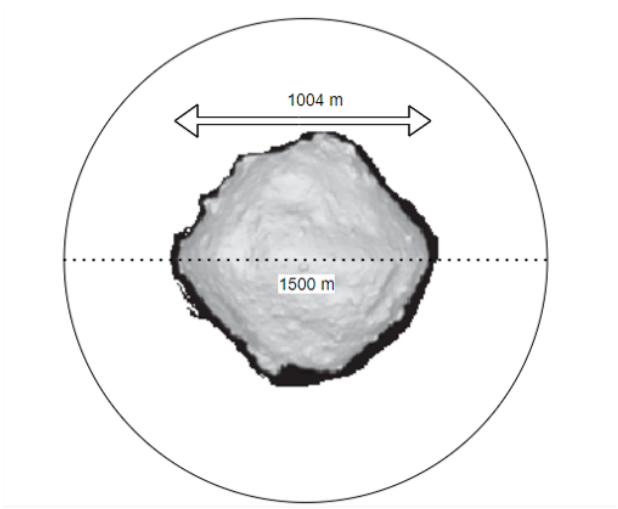


Figure 5.3: Asteroid Ryugu[61] with keep out sphere

5.2.1 Single satellite inspection problem

Single satellite inspection problem case involves using a single satellite for inspection problem. Different cases of single satellite inspection problem are categorized based either on the target size or the type of sampling method employed. The first Case is named as SINGLE CASE-1. Target chosen for this case is ISS model. The setup parameters for this simulation is shown in the table 5.3. Clohessy Whiltshire dynamics is considered as it is be suitable for given target orbit parameters. Attitude database characteristics shows that during tracking maneuver spacecraft attitude angular motion will be higher than the mean motion of the target orbit. Number of samples n_{sample}, m_{sample} of IFMT*-I and IFMT*-I are set to 3 and 8 respectively. Total number of samples in sample space for

translation is set to 1796. Total attitude sample space has 100 tracking samples. For each of these tracking samples or primitives, there are 100 possibilities to move to another configuration (another tracking primitive) via slew maneuver. As a result, in total there are 100 tracking maneuvers + 10000 slew primitives available. Finally, the sampling method is chosen to be Psuedo-random generator. After running the AMPSGIM, following results are obtained. Figure 5.4 shows the Coverage map

Table 5.3: SINGLE CASE-1: Simulation setup

Input parameters	
General Parameter	*Values*
Target	Maximum length =109 m & KOZ =130 m
Number of satellites	1
Translation dynamics	Clohessy -Whiltshire
Inpection duration (ID)	2*T$_{target}$($TargetPeriod$)[s]
Search space	
[r_{max},r_{min}]	[2000 m,130 m]
Number of segments	2
Number of low thrust segment	2
Number of observation segment	7
Database	
Tracking angular velocity	[2.75e-2 2.75e-2 0] [rad/sec]
Tracking time	100 s
Slew time	50 s
Total number of instances m$_{total}$	100 tracking primitives (10000 slew primitives)
Translation	*IFMT* -I*
n$_{total}$	1796
n$_{sample}$	3
r_{n_m}	9.7034e-04 kg
Attitude	*IFMT* -II*
m$_{total}$	100 tracking primitives (10000 slew primitives)
m$_{sample}$	8
[r_{n_τ} $r_{n_{cov}}$]	[0.5630 Nm,0.95]
Sampling method	Pseudo Random

of the satellite for SINGLE-CASE 1. Coverage maps are the 2-D figure which consist of information about all the triangulated surfaces on the target sphere. In the figure, white blocks are uncovered surfaces, rest of the coloured blocks show the covered surfaces which are observed several times, their corresponding numerical value is indicated by the intensity of the colour. Note that number of times a surface has been observed depends on the step size used in the discretization of $t_{tracking}$, thus the values are just an indication and not to be considered as true values. Only main take-away is that the respective surface which has the color code other than white and black are considered observed.

- Red - If the surface has been observed more than 500 times
- Yellow - If the surface has been observed greater then 200 times and less than 500 times.
- Green - If the surface has been observed surfaces at least once and less than 100 times
- White - represents uncovered surfaces
- Black tiles are extra ones added just to make the figure look like a full rectangle.

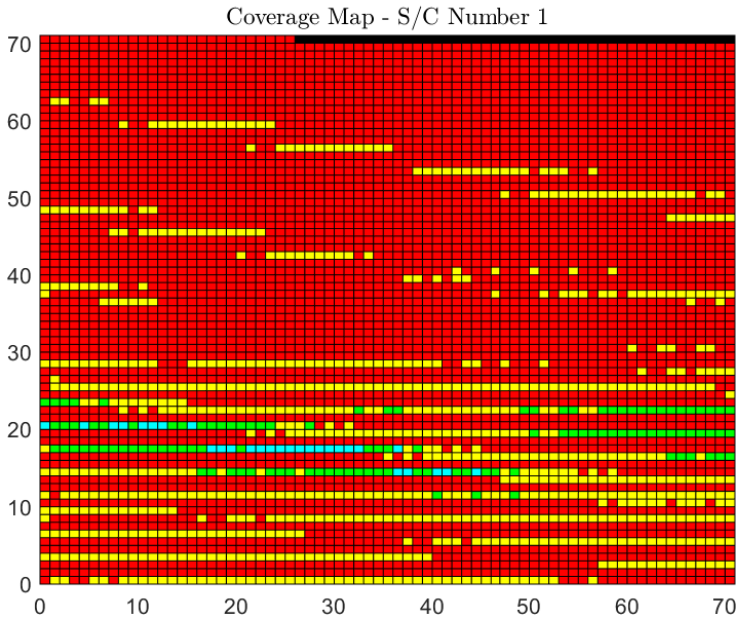


Figure 5.4: SINGLE CASE-1: Coverage Map

For the case considered, the coverage map shows 100% coverage. The translation and rotation trajectories associated with coverage maps are shown in figure 5.5 and figure 5.6 respectively. From Figure 5.5 it is clear that translation trajectories lie inside the threshold (r_{max}) that was mentioned in the parameter setup. Even with this threshold, the search algorithm found a trajectory to observe the large areas of target (leading to 100% coverage). As pointed out earlier, the requirements to come closer to the target could be critical and when this constraints becomes strict, the coverage may not be 100%. Figure 5.6a and 5.6b shows part of rotational motion, particularly a track+slew+track sequence is shown. It can be observed that there is a peak at the centre of control torque (figure 5.6c) which corresponds to slew maneuver. Thrust acceleration applied during low thrust segment of the trajectory is shown in figure 5.7a and figure 5.7b.

Two figures corresponds to first and second low thrust segment respectively. As expected, overall, both translation and rotation trajectories satisfy the hard constraints (see constraints section

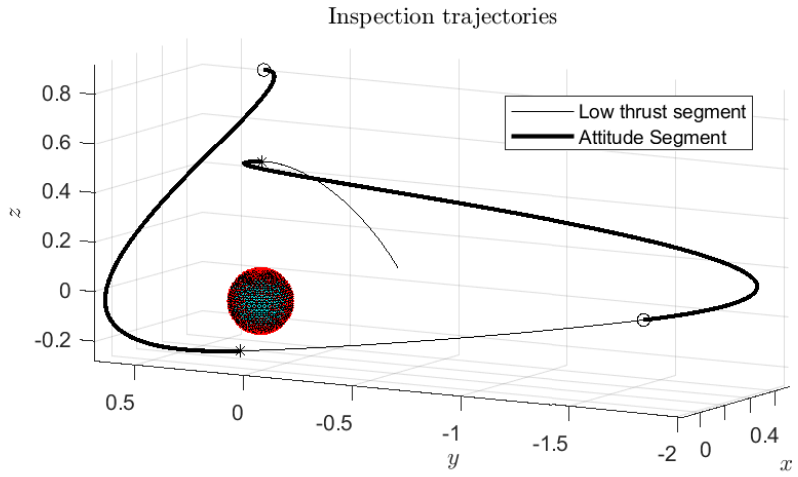


Figure 5.5: SINGLE CASE-1: Inspection translation trajectory

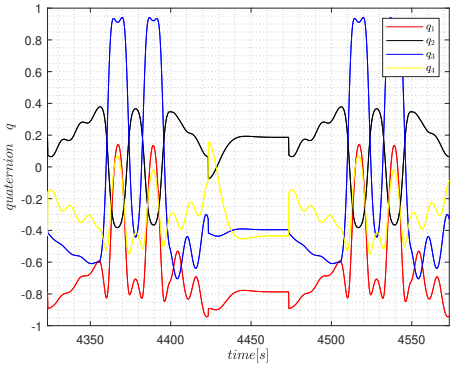
4.2.1). Now, with the same setup, deterministic sampling method is applied. This case is considered as SINGLE-CASE 2 and corresponding results are reported here. The total coverage was found to be 36.58%. Figure 5.8 shows the coverage map. Figure 5.9, shows the translation trajectory corresponding to the Inspection mission.

And the rest of the results are similar to SINGLE CASE-1. The coverage percentage by running deterministic sampling method is always fixed for fixed parameters. On the other hand, psuedo random generator produced results, sometimes are better and sometimes are worse than the SINGLE-CASE 2. Table 5.4 shows coverage for SINGLE CASE-1 for 4 runs, It can be noticed that coverage values are varying indicating the nature of random sampling procedure.

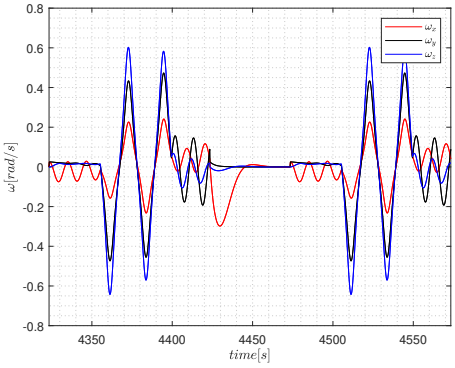
Table 5.4: SINGLE CASE-1: Simulation runs and Coverage

Run Number	**Coverage**
1	68.39
2	64.71
3	17.37
4	64.99

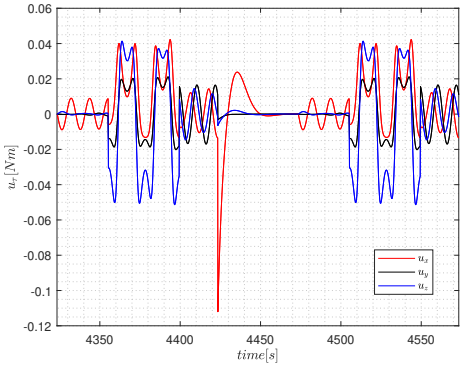
From SINGLE CASE -1 and SINGLE CASE-2, it is evidently visible that psuedo random generator because of randomness sampling procedure finds the results better than deterministic approach. But as pointed out in section 2.3.2 the reliability is not guaranteed. Thus, when a analysis tries to



(a) Quaternions for track+slew+track sequence of Attitude segment



(b) Angular velocity for track+slew+track sequence of Attitude segment



(c) Control Torque for track+slew+track sequence of Attitude segment

Figure 5.6: SINGLE CASE-1: Attitude segment sequence

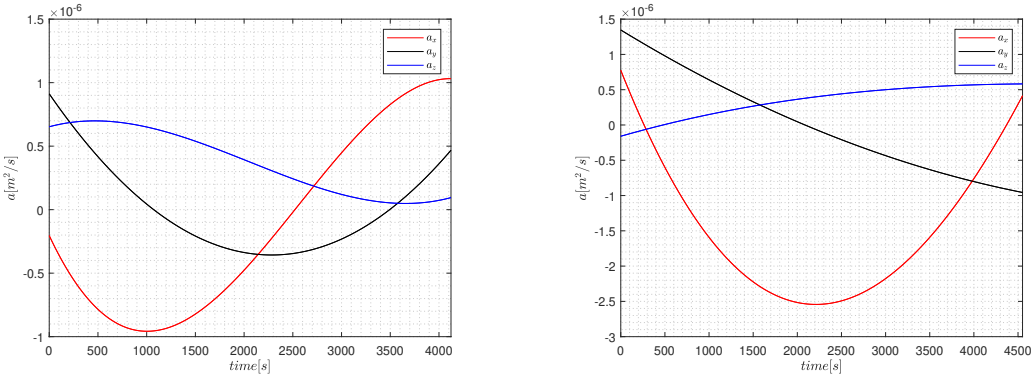
understand the performance behavior of AMPSGIM framework, deterministic sampling method is used unless otherwise specified.

5.2.2 Analysis on variation of input parameters

In this section, various input parameters are varied to understand their impact on the overall performance of the framework.

Connecting radius

Connecting radius forms an important criteria for framework, as it dictates how searching is done by SBMP algorithm. Therefore, the impact of connecting radius r_n on the objective functions are analyzed here. As indicated in section 4.2.6, three connecting radius r_{n_m} r_{n_τ} and $r_{n_{cov}}$ corresponding to fuel cost, torque cost and coveragec compliment are identified for the search algorithm. In



(a) First Thrust acceleration for low thrust segment

(b) Second Thrust acceleration for low thrust segment

Figure 5.7: SINGEL CASE-1: Low thrust segment thrust acceleration

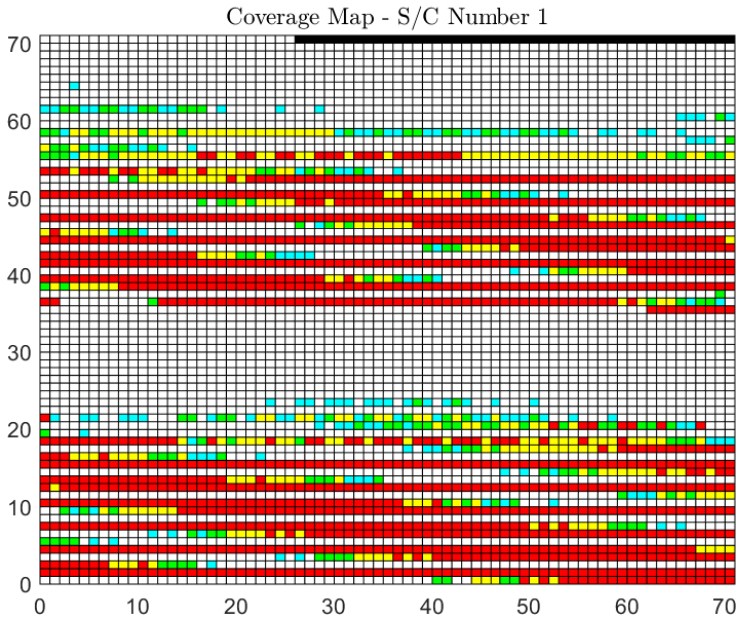


Figure 5.8: SINGLE Case-2: Coverage Map

the following analysis, each of these connecting radius are varied by fixing the other parameters. For this analysis, SINGLE CASE-2, deterministic approach is employed. The setup is modified a bit in parameters and shown in table 5.5. When considering the variation of one connection radius parameter the rest of connecting radius parameter values are assumed to be the ones in SINGLE CASE-1 (table 5.3). Variation of J_{cost} $\{J_{\tau}$,$J_{coverage}$,J_m $\}$ and Coverage with respect to $r_{n_{cov}}$,

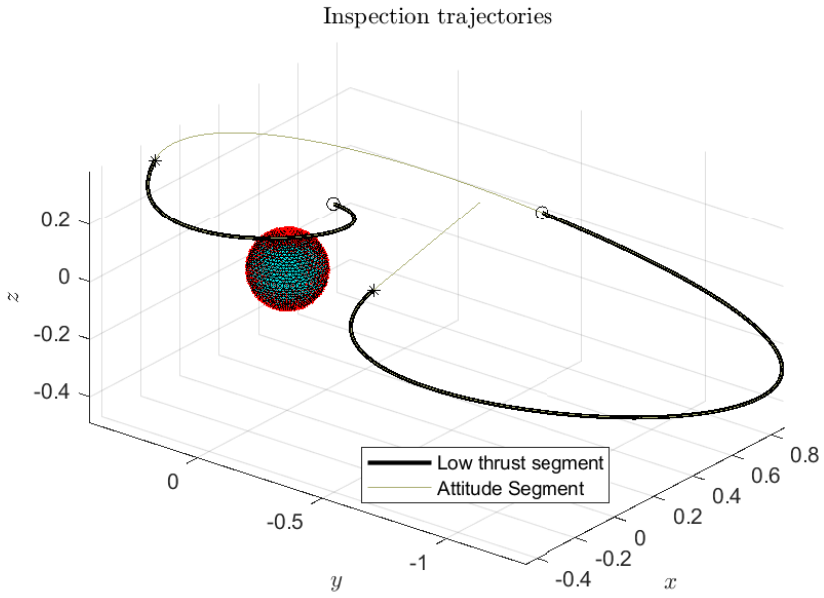


Figure 5.9: SINGLE CASE-2: Inspection translation trajectory

r_{n_τ}, and r_{n_m} are shown in table 5.6, 5.7 , and 5.8 respectively. The common trend that can be observed for all these variables is that, as connecting radius value is decreased, cost corresponding to respective connecting radius decreases. This behavior is important as this will lead to solution which will have lower cost. For example, table 5.7, shows the variation of r_{n_τ}, it can be observed that as the value of r_{n_τ} reduced, the torque cost J_τ improves (cost reduces). But changing one connecting radius value may not improve other cost values. This can be seen by studying the values in the tables, some cost values tend to decrease and some tend to increase showing that there is no direct proof as to whether they are directly influenced by other connecting radius other then their own. In table 5.6, the r_{ncov} (connecting radius corresponding to coverage compliment) is reduced which leads to increase in the coverage and cost linked with it improves. From this analysis, it is to be noted that the performance of the AMPSGIM framework can be controlled by efficiently tuning the connecting radius corresponding to different objective functions involved.

Miscellaneous parameters

Moving on, now search space vector $[r_{min} \quad r_{max}]$ and translation total sample size n_{total} are considered as the variables. Obtained analysis results are reported here. Table 5.9 and 5.10 shows the results for $[r_{min} \quad r_{max}]$ and n_{total} respectively. It is evident from table 5.9, as the search space size is increased, the coverage value is increasing which is expected because spacecraft could see large area of the target.

Table 5.5: SINGLE Case-2 parameters for analysis on variation of input parameters

Input parameters	
Inspection Mission	*Values*
Target	Maximum length =109 m & KOZ =130 m
Number of satellites	1
Transaltion dynamics	Clohessy -Whiltshire
Inpection duration (ID)	2*T$_{target}[s]$
Search Space	
$[r_{max}$,$r_{min}]$	[2000 m,130 m]
Number of segments	2
Number of low thrust segment	2
Number of observation segment	7
Database	
Tracking angular velocity	[2.75e-2 2.75e-2 0] [rad/s]
Tracking time	100 s
Slew time	50 s
Total number of instances m total	100 tracking primitives (10000 slew primitives)
Translation	*IFMT* -I*
n_{total}	1796
n_{sample}	3
r_{n_m}	-
Attitude	*IFMT* -II*
m_{total}	100 tracking primitives (10000 slew primitives)
m_{sample}	5
$[r_{n_\tau}\ r_{n_{cov}}]$	-
Sampling method	Deterministic Sampling

In table 5.10, costs are shown differently. Here, total cost linked with attitude and translation is shown. For attitude, the total cost takes care of J_τ (torque cost) and $J_{coverage}$ (coverage compliment). Translation part is linked to J_m (fuel cost). It can be observed the fact that as the number of samples are increased, the coverage is increasing. Along with coverage, attitude cost is showing a steady decrease as well, with exception of increased fuel cost. But the important factor to note here is that as the sample size is increased the possibility of finding better solution is increasing as well. This is one of the basic properties of FMT* algorithm (see section 2.3.3).

Now that two important cases for AMPSGIM implementation are shown. Some of the extra cases with modified attitude database and target sizes are presented next. First, an extension of SINGLE CASE-2 with new attitude database is shown. The case is named SINGLE CASE-2A. The table shows the characteristics of the new attitude database along with the performance.

The new database tracking maneuver has angular velocity equivalent to the order of magnitude of the mean motion n_t of the target orbit. The impact of having a lower angular velocity in tracking

Table 5.6: Variation of *Coverage* and J_τ $J_{Coverage}$ J_m with respect to $r_{n_{Cov}}$

$r_{n_{cov}}$	Coverage	J_τ [Nm]	$J_{Coverage}$	$J_m[kg]$
0.99	57.4610	3.6429	11.8402	3.22E-06
0.9	70.1002	2.5342	11.2216	2.75E-06
0.88	70.1002	2.6932	11.2338	2.75E-06
0.82	96.5478	2.5342	10.7321	2.64E-06
0.80	96.5478	2.5342	10.7321	2.64E-06
0.78	100	3.6429	9.4576	4.82E-06

Table 5.7: Variation of *Coverage* and J_τ $J_{Coverage}$ J_m with respect to r_{n_τ}

r_{n_τ} [Nm]	Coverage	J_τ[Nm]	$J_{Coverage}$	J_m[kg]
1.8384	99.7772	4.0993	9.8257	1.24E-06
1.4967	99.7772	3.8909	9.9181	1.24E-06
1.4176	99.7772	3.8909	9.9181	1.24E-06
1.3181	70.1002	3.2962	11.4164	2.75E-06

Table 5.8: Variation of *Coverage* and J_τ $J_{Coverage}$ J_m with respect to r_m

r_{n_m} [kg]	Coverage	J_τ [Nm]	$J_{Coverage}$	J_m [kg]
5.00E-05	99.7772	3.6429	9.9148	1.24E-06
1.00E-05	99.7772	3.6429	9.9148	1.24E-06
2.00E-06	99.7772	3.6429	9.9148	1.24E-06
1.50E-06	99.7772	3.6429	9.9148	1.24E-06
7.10E-07	70.1002	3.6429	11.4471	6.74E-07

Table 5.9: Variation of *Coverage* and J_τ $J_{Coverage}$ J_m with respect to Search Space [r_{min}, r_{max}]

r_{min} (km)	r_{max} (km)	Coverage	J_τ [Nm]	$J_{Coverage}$	J_m [kg]
0.5	1	60.4677	3.6429	11.5662	1.49E-06
0.6	1.2	70.7683	3.6429	11.3713	1.68E-06
0.9	1.6	93.4298	3.6429	11.1531	5.93E-07
1.5	3	100	3.6429	10.9448	7.54E-06

Table 5.10: Variation of *Coverage* and $J_{attitude1}$ $J_{translation}$with respect to (n_{total})

n_{total}	Coverage	$J_{attitude}$	$J_{translation}$
1796	17.9844	12.4630	3.81E-06
4996	67.7616	10.4295	1.96E-06
7196	65.7015	10.5994	1.44E-06
12796	66.0356	10.5565	3.36E-06

maneuver indicates that the surfaces covered are lower than SINGLE-CASE 2 scenario. The reason is that with higher angular velocity the spacecraft attitude moves through a large angle giving increased opportunity for the payload to cover more surfaces on the target. As shown earlier in the

Table 5.11: Parameters and performance results for SINGLE CASE-2A

Attitude Primitives	
n_t	0.0011
omega (rad/s)	[0.0025 0.0025 0]
$t_{tracking}(s)$	100
$t_{slew}(s)$	20
Performance	
Coverage	10.528
J_τ [Nm]	5.892
$J_{Coverage}$	30.4412
J_m [Kg]	3.87E-06

section 5.2.2, the performance of this case can be improved by tuning the connecting radius parameters. SINGLE CASE-2A indicates that the development of attitude database plays an important role in getting better solutions.

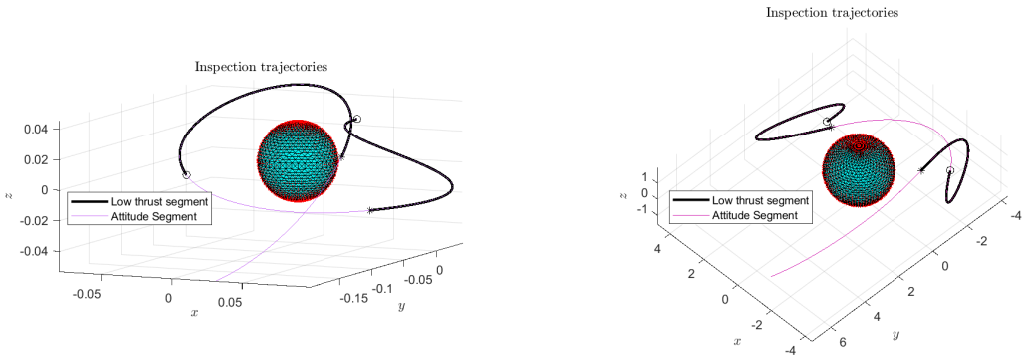
Next two cases deal with two different target sizes. They are named as SINGLE CASE-3 and SINGLE CASE-4. SINGLE CASE-3 corresponds to the target which is of Envisat satellite category. And SINGLE CASE-4 corresponds to the target which is of Asteroid Ryugu category. For both the cases, deterministic sampling method is applied. The input parameters are almost similar to SINGLE CASE-1. The new modified parameters are shown in the table 5.12 along with performance results. Modification in search space is needed as the target size is different. This will ensure the safety and resolution constraints are satisfied. Figure 5.10 shows the translation trajectories

Table 5.12: SINGLE CASE 3 and SINGLE CASE 4 modified inputs and performance

Case Number	$[r_{min}\ r_{max}]$ (km)	$r_{n_{cov}}$	r_{n_τ} [Nm]	r_{n_m} [kg]	Coverage
3	[0.050 0.50]	0.97	0.6937	9.70E-04	64.1713
4	[3.5 9.5]	0.97	0.6937	9.70E-04	25.3803

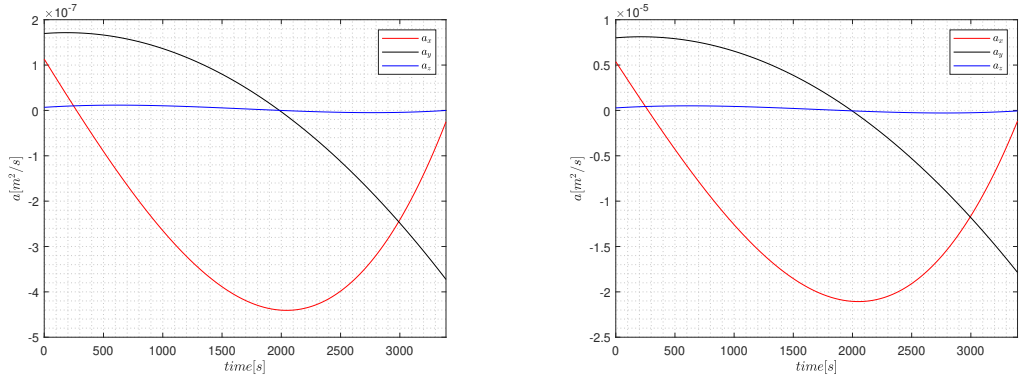
produced for inspection corresponding SINGLE CASE-3 and SINGLE CASE-4 respectively. Figure 5.11a and figure 5.11b shows the thrust acceleration trajectories for SINGLE CASE-3 and SINGLE CASE-4 respectively. The SINGLE CASE-4's thrust acceleration magnitude is higher then SINGLE CASE-3 case. This is because of the size of asteroid, the trajectories are of the order of $O(10^3)$ m.

From table 5.12, Two things can be observed, Envisat which is the smallest out of three targets considered is mapped comfortably by a single satellite with coverage of 64.1713%, whereas for Asteroid Ryugu, the satellite could cover only small portion of the surface. This can be attributed to the fact that the sensor is limited by the FOV and also the search space is constrained by r_{max} which is controlled by the resolution requirement of the mission. During situations where search space is constrained and time available for observation is limited using more than one satellite becomes efficient. In that direction, next section describes the applicability of AMPSGIM for Inspection scenarios using multiple satellites.



(a) SINGLE CASE-3 : Envisat (b) SINGLE CASE-4 : Asteroid Ryugu

Figure 5.10: SINGLE CASE-3 and SINGLE CASE-4 translation trajectory



(a) SINGLE CASE-3 (b) SINGLE CASE-4

Figure 5.11: SINGLE CASE-3 and SINGLE CASE-4 thrust acceleration for first Low thrust segment

5.2.3 Multiple satellite inspection problem

Multiple satellites can be used to perform the inspection when there is limitation on the time available for mapping. In the single satellite cases (SINGLE CASE-1 and SINGLE CASE-2), when the search space size is restricted, it was seen that the coverage was less (see table 5.9). This issue can be overcome by having multiple satellites observing the target. The cumulative coverage of all the satellites will be higher than using a single satellite. The aim of this section is to demonstrate the applicability of AMPSGIM to the multiple satellite case and show the advantages of using it in scenarios where single satellite solution is not efficient.

In that regard, the first case for multiple satellite inspection problem is presented here. This case is called MULTI-CASE 1. MULTI-CASE 1 - is for the ISS target case with two service satellites. The

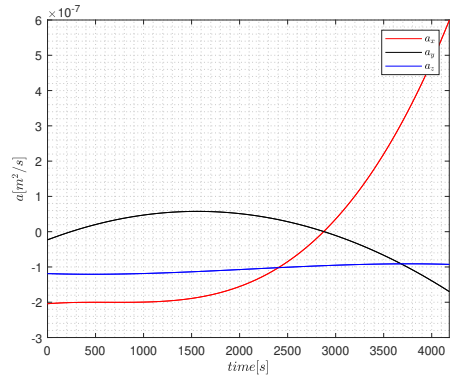
table 5.13 shows the important parameters involved. As compared to single satellite case, number of observation segment is reduced (now it is 5) , separate connecting radius for individual satellites and parameters required for collision and pointing avoidance are added. In table 5.13, the minimum distance between satellite is assumed to be 25 m and as per pointing avoidance, 50% informs that at each tracking sequence of attitude segment, two satellites cover surfaces which are at least 50% different. Pseudo Random generator sampling method is chosen for MULT-CASE 1. For multiple

Table 5.13: MULTI-CASE 1: Target ISS

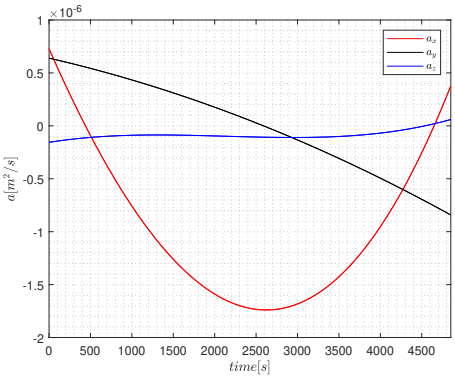
Input parameters	
General Parameters	*Values*
Target	Maximum length =109 m & KOZ =130 m
Number of satellites	1
Transaltion dynamics	Clohessy -Whiltshire
Inpection duration (ID)	2*T$_{target}$
Search space	
[r_{max},r_{min}]	[2000 m,130 m]
Number of segments	2
Number of low thrust segment	2
Number of observation segment	5
Database	
Tracking sngular velocity	[2.75e-2 2.75e-2] [rad/s]
Tracking time	100 s
Slew time	50 s
Total number of instances m total	100 tracking primitives (10000 slew primitives)
Translation	*IFMT* -I*
n$_{total}$	1796
n$_{sample}$	3
$r_{n_{m_1}}$ and $r_{n_{m_2}}$	9.7034e-04,9.7034e-04
Attitude	*IFMT* -II*
m_{total}	100 tracking primitives (10000 slew primitives)
m_{sample}	8
[$r_{n_{\tau 1}}$ $r_{n_{cov1}}$] and [$r_{n_{\tau 2}}$ $r_{n_{cov2}}$]	[0.5630 Nm 0.950],[0.5630 Nm 0.950]
Sampling method	Pseudo Random
Collision avoidance	
Minimum distance	25 m
Pointing avoidance	
Maximum overlap	50%

satellite case (MULTI-CASE 1), the cumulative coverage percentage found to be 95.3162% with 51.6413% coming from satellite 1 and 43.6749% coming from satellite 2. Which is lower than single

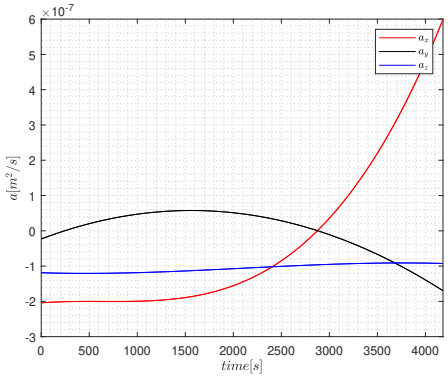
satellite case (SINGLE-CASE 1), but as pointed out earlier, this result is not the exact true solution as the pseudo random generator is used as sampling method. In anyway, possibility of achieving 100% coverage is higher in this case than SINGLE CASE-1. It is important to notice that the observation segment is reduced to 5 as compared to 7 in single satellite case (SINGLE CASE-1). Thus, with less amount of time, two satellites configuration can finish the mapping, this is due to



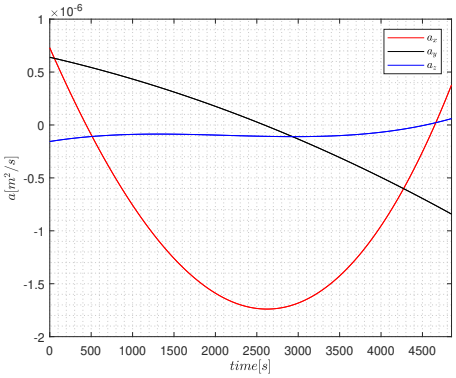
(a) Satellite 1 - Thrust acceleration for first low thrust segment



(b) Satellite 1 - Thrust acceleration for second low thrust segment



(c) Satellite 2 - Thrust acceleration for first low thrust segment



(d) Satellite 2 - Thrust acceleration for second low thrust segment

Figure 5.12: MULTI CASE-1: Low thrust segment: thrust acceleration

trivial reason of using two satellites instead of one. Translation and attitude (track+slew+track sequence) trajectories involved in MULTI-CASE 1 are shown in figure 5.13 and 5.14 respectively. Thrust acceleration for both the satellites are shown in figure 5.12

Now, AMPSGIM showed that it can be applied to find the coupled guidance for multiple satellite inspection case. Next, comparison between single satellite and two satellite inspection are shown to understand if multiple satellites produce efficient solution or not. This two satellite case is called as MULTI CASE 1A. For the comparison case, the search space is strictly restricted, this emulates

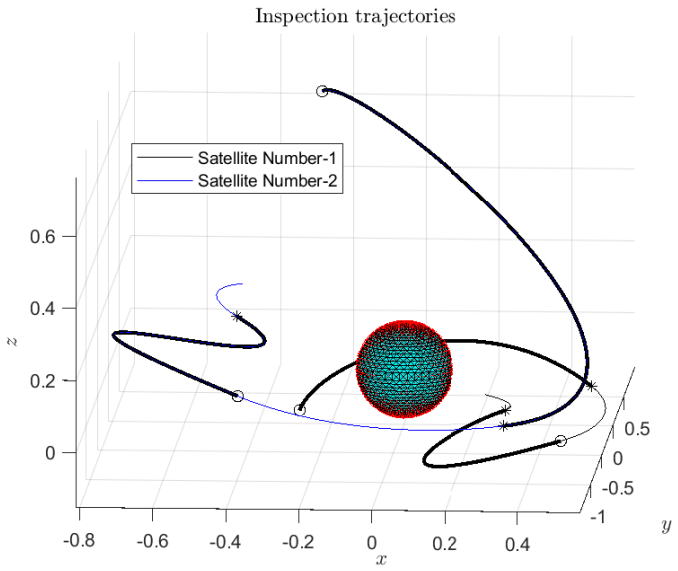
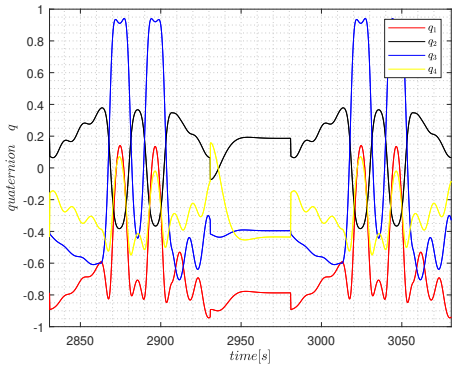


Figure 5.13: MULTI CASE-1: Two satellite Inspection translation trajectories

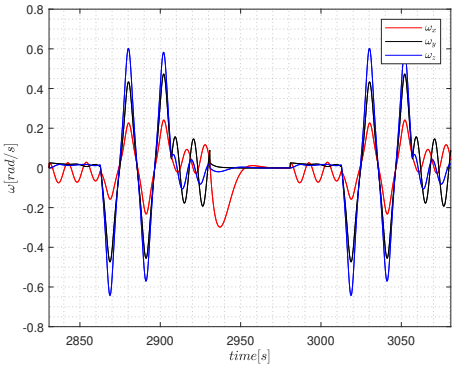
the scenario where the spacecraft is required to be in close proximity of the target and required to take high quality images. Input parameters are similar to MULTI CASE-1 with modification indicated below and in the table 5.14. The target considered is ISS. Deterministic sampling method is used for this case. 70% unique coverage and 25 m minimum threshold distance is considered for pointing avoidance and collision avoidance constraints respectively. The table 5.14 shows the modified parameters and performance of single and two satellite configuration. Single and two satellite systems are given with same inspection duration.

Table 5.14: Comparison of single and two satellite system for Inspection

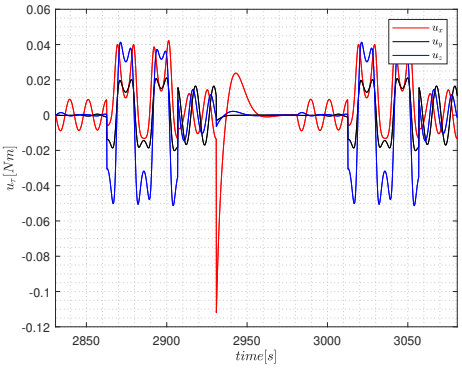
	Single Satellite	**Two satellite**		
Input parameters		Satellite 1	Satellite2	Total
$[r_{min}\ r_{max}]$*$10^3 m$	[0.5,1]	[0.5,1]	[0.5,1]	
r_{n_m} [kg]	9.70E-04	9.70E-04	9.70E-04	
r_{n_τ} [Nm]	0.6937	0.6937	0.6937	
$r_{n_{cov}}$	0.97	0.97	0.92	
n_{total}	1796			
Inspection duration [s]	2*T_{target}			
Performance				
Coverage	48.9391	48.9391	32.3058	81.2449
J_m [kg]	4.29E-06	4.29E-06	1.95E-06	6.24E-06
$J_{coverage}$	23.0757	23.0757	22.7596	
J_τ [Nm]	2.8502	2.8502	5.8836	8.7338



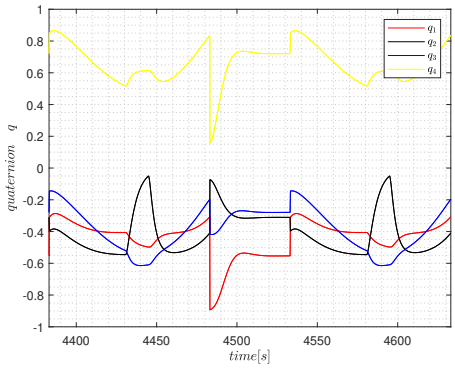
(a) Satellite 1 - quaternions for track+slew+track sequence of Attitude segment



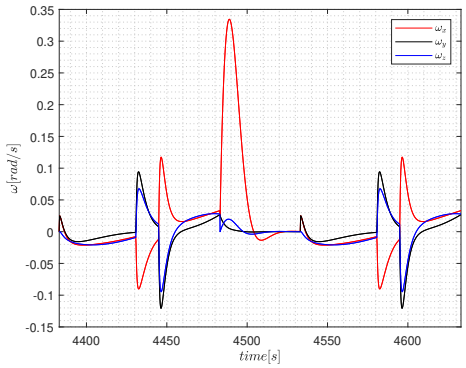
(b) Satellite 1 - angular velocity for track+slew+track sequence of Attitude segment



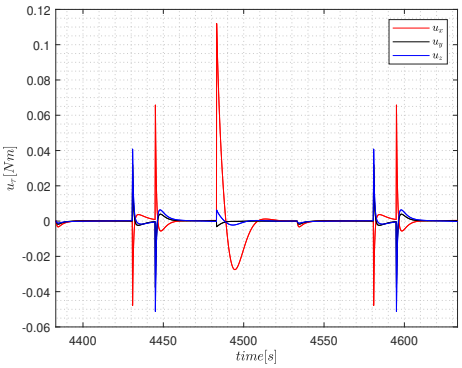
(c) Satellite 1 - Control Torque for track+slew+track sequence of Attitude segment



(d) Satellite 2 - quaternions for track+slew+track sequence of Attitude segment



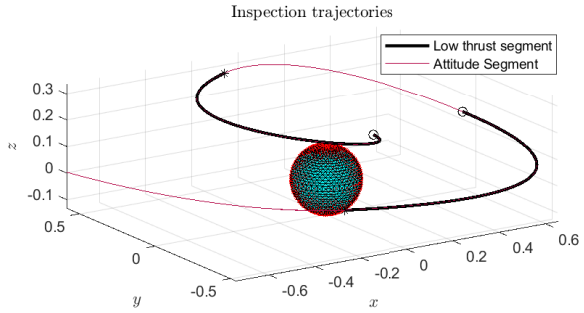
(e) Satellite 2 - angular velocity for track+slew+track sequence of Attitude segment



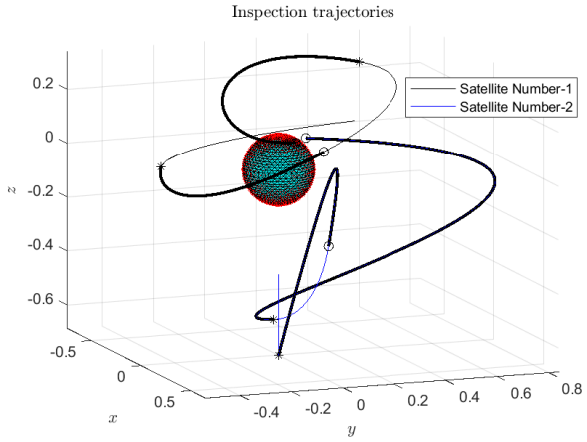
(f) Satellite 2 - Control Torque for track+slew+track sequence of Attitude segment

Figure 5.14: MULTI CASE 1: Attitude motion for two satellites

Input parameters are almost similar to each other for both the cases. The only difference is the connecting radius related to satellite 2 is reduced making it more stricter to search for good coverage solutions. As expected, the performance shows the impact of change of $r_{n_{cov}}$ for satellite 2, the coverage compliment cost is reduced as compared to single satellite case. Important thing to notice here is that as anticipated, for the same amount of time, two satellite case produces higher coverage then single satellite case. The cumulative cost incurred in terms of fuel mass and torque can be seen to have increased, this due to the use of two satellites. As shown in section 5.2.2, various parameters can be adjusted to reduce the respective costs. The inspection translation trajectories for single and two satellite systems are shown in figure 5.15.



(a) Inspection trajectory for single satellite case



(b) Inspection trajectory for two satellite case

Figure 5.15: Single and two satellite configuration translation trajectories

A simple extension of the two satellite case is presented next. This case is called MULTI CASE-2 and consists of 3 satellites. The setup parameters required for MULTI CASE-2 are shown in table 5.15. The main differences are addition of extra connecting radius for third satellite and 50% unique

Table 5.15: MULTI CASE-2 setup parameters

Input parameters	
General Parameters	*Values*
Target	Maximum length =109 m & KOZ =130 m
Number of satellites	1
Transaltion dynamics	Clohessy -Whiltshire
Inpection duration (ID)	2*T$_{target}$
Search Space	
[r_{max},r_{min}]	[1000 m,130 m]
Number of segments	2
Number of Low thrust segment	2
Number of Attitude segment	5
Database	
Tracking Angular velocity	[2.75e-2 2.75e-2 0] [rad/s]
Tracking time	100 s
Slew time	50 s
Total number of instances m total	100 tracking primitives (10000 slew primitives)
Translation	*IFMT* -I*
n_{total}	1796
n_{sample}	3
$r_{n_{m_1}}$ and $r_{n_{m_2}}$, $r_{n_{m_3}}$	9.7034e-04kg,9.7034e-04kg,9.7034e-04kg
Attitude	*IFMT* -II*
m_{total}	100 tracking primitives (10000 slew primitives)
m_{sample}	8
[$r_{n_{\tau 1}}$ $r_{n_{cov1}}$] and [$r_{n_{\tau 2}}$ $r_{n_{cov2}}$] , [$r_{n_{\tau 3}}$ $r_{n_{cov3}}$]	[0.5630Nm 0.97],[0.5630Nm 0.92],[0.5630Nm 0.93]
Sampling Method	Deterministic Sampling
Collision Avoidance	
Minimum distance	25 m
Pointing Avoidance	
Maximum overlap	50%

coverage is considered for pointing avoidance constraints. The performance of MULTI CASE 2 is shown in table 5.16. Comparing the obtained results between two satellite in table 5.14 (MULTI CASE 1A) and in table 5.16, the coverage results are similar, but the differences can be noticed in J_τ and J_m. J_τ has increased in MULTI CASE-2 case, where J_m reduced. The reason for this result is due to the pointing avoidance (PA) constraint. In MULTI CASE 1A 5.14, 70% was the value

Table 5.16: MULTI CASE-2 performance result

Performace	Satellite 1	Satellite2	Satellite3
Coverage	48.9391	32.3058	13.4107
J_m [kg]	3.21E-06	1.20E-06	1.85E-06
$J_{coverage}$	23.0757	22.5088	21.9112
J_τ [Nm]	2.8502	6.2114	3.9035

for PA, whereas in MULTI CASE-2 it was reduced to 50%. This relaxes the pointing avoidance constraint and as a result the trajectories will be considered even if they have less then 70% unique surface coverage with other satellite's coverage. Inference from this variation is that even with good tuning of connection radius, the pointing and collision avoidance plays an important role in deciding which trajectory is the best suitable. Now getting back to MULTI CASE-2, because of three satellite configuration, the total coverage has reached 94.655%. By considering a single satellite, achieving a coverage of this magnitude will require large amount of time. Therefore, this proves that in time limited and close proximity observation situations, multiple satellite configuration produces good coverage than single satellite. The translation trajectories for inspection corresponding to three satellite is shown in figure 5.16. These demonstration proves that the AMPSGIM can be applied

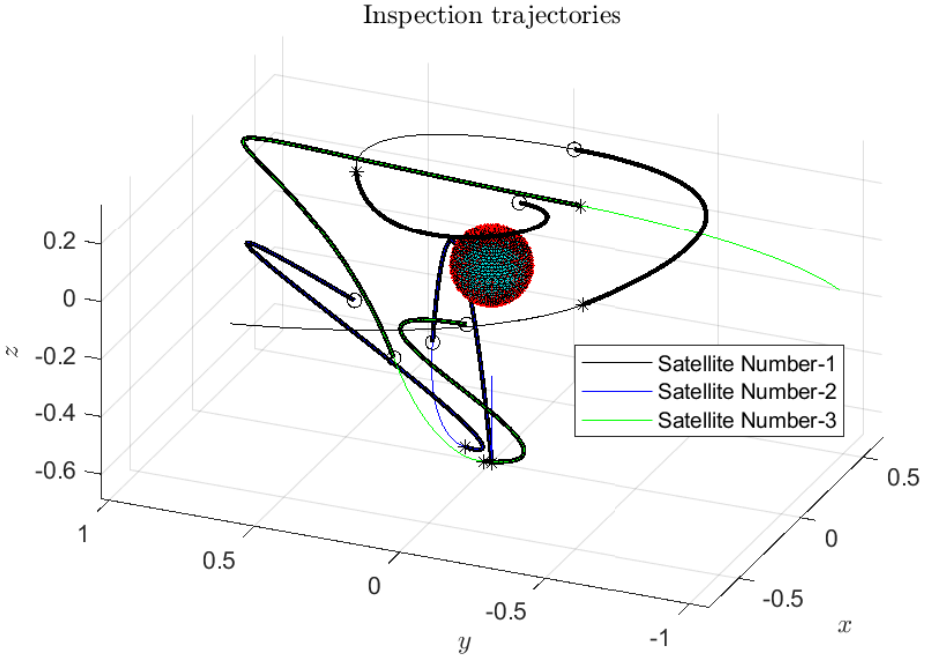


Figure 5.16: MULTI CASE-2: translation trajectories for Inspection

to perform inspection mission with more than one satellites and the extension of this to 4 or 5 satellite system is straightforward and depends on the inspection mission requirement. However, main concern was found to be the computational cost. Computational cost was observed to be high for the cases using more than one satellite. Table 5.17 shows the number of times collision avoidance (CA) and pointing avoidance (PA) has led to in-feasibility of the translation and attitude segment.

Table 5.17: Collision avoidance and pointing avoidance results

Parameters	**SINGEL CASE 2**	**MULTI CASE-1**	**MULTI CASE 2**
PA	0	6	12
CA	0	213	256

These checks lead to increased number of computations required to find the feasible solutions. The issue related to computation cost is briefly discussed in chapter 6 with other limitations and further improvements needed for enhancement of the AMPSGIM framework.

Chapter 6

Limitations and Further developments

"The only true wisdom is in knowing you know nothing."
— Socrates

In this chapter, limitations associated with the developed AMPSGIM framework are discussed first. Then, in the further development section, improvement points are identified to further enhance the reliability and performance.

6.1 Limitations

- Computational Cost: This is one of the main limitations of the framework. Solving TPBVPs, checking for collision avoidance, pointing avoidance, and calculating the coverage using the assumed coverage model requires high computation demand. Because of these reasons, the algorithm takes on the order of $O(10^3)$s for two satellites case to find the coupled guidance while using the pseudo-random generator. Computational cost is also attributed to how the framework architecture is designed. AMPSGIM framework uses two FMT* based algorithms (IFMT*-I and IFMT*-II). To add one edge to IFMT*-I, the IFMT*-II is required to run at least 2 times (refer to *Expand_Translate* of IFMT*-I 4.3.3) assuming that there is a minimum number of nodes involved. This situation is exacerbated by the fact that collision and pointing avoidance needs to be taken care of as well. All these factors make the algorithm computationally costly. If this limitation is not overcome, the framework may not be suitable for real-time applications.

- Accuracy: AMPSGIM uses many simplifications when it comes to finding a coupled guidance. The sampling process assumes that every point sampled satisfies the energy-bound condition. This is done to get the natural trajectories for the attitude segment. Even though it might produce flexibility for observation and low thrust propulsion, the possibility of finding a good solution is reduced. When it comes to attitude steering law, building a database is efficient and good for solving the TPBVPs rapidly. But, the accuracy is reduced as it has a limited number of instances available. Next, dynamic models both for translation and attitude motion needs to be augmented with more realistic models. And finally, the accuracy of coverage calculation also depends on the estimation model assumed so a more efficient model needs to be used.

To mitigate the above limitations and to augment the performance of the AMPSGIM framework, in the next section further developments required are mentioned.

6.2 Further developments

Many improvements and modifications can be done to the framework to improve its performance and increase its reliability. Some of them are identified and listed below

- The framework's performance needs to be validated and verified by comparing with other developed and robust solution methods which solve the inspection mission.
- For the translation relative motion, general linear dynamics for eccentric orbit may replace the more restricted clohessy whiltshire dynamics.
- The framework can be adapted to include disturbances that are present both in translation and attitude motion.
- Efficient collision avoidance and pointing avoidance methods are required to alleviate the computation burden.
- Attitude motion involved in the low thrust segment needs to be accounted for if the framework wants to simulate the true attitude of the satellite system.
- For translation trajectories, higher accurate low thrust trajectory solvers can be used [20][54].
- Possibility of using Online attitude TPBVP solver will enhance the accuracy of the framework by reducing the restrictions it has due to the limited number of instances of the primitives in the database.
- The values of the upper and lower bound of the connecting radius shall be found. In the thesis, these values are computed empirically using simple estimations. By finding this value accurately will help in driving the search algorithm to find good solutions

- Current framework ignores some of the critical constraints coming from the Inspection mission. Some of the constraints related to observation are sun-target-servicer angle constrain; which dictates the quality of the data product, possible occultation constraints, etc. General constraints such as plume impingement which protects the satellites and target from low propulsion plume impingement and payload sensitivity which limits the avoidance of pointing in the direction of the sun.
- A more efficient and accurate coverage estimation model is required to accurately map the target surfaces.
- The framework can be extended to inspect multiple targets.
- More rigorous analysis of sampling procedure is required. Possibility of building a sample space in objective function space will provide good search direction and will increase the probability of finding optimal solution. In [53], author builds a sample space of ΔV which are linked to mission goal. This methodology may be a starting point but it needs to be modified to apply it to low thrust and attitude trajectories which are continuous unlike ΔV.

Along with these, assumptions mentioned in 1.4 can be relaxed to make the framework reliable and robust.

Chapter 7

Conclusion

"If people do not believe that
mathematics is simple, it is only
because they do not realize how
complicated life is"
-John von Neumann

Thesis work set out to develop a framework that can provide the capability to produce coupled guidance for low thrust multiple satellites using motion planning algorithms. In that direction, the FMT* algorithm was selected because of its characteristics and proceeded to adopt the principle of FMT* to solve the Inspection problem. In chapter 4, the core methodology followed to develop the coupled guidance framework for Inspection is presented. The framework is called AMPSGIM and it included all the adaptation needed to use FMT* algorithms for the Inspection guidance. AMPSGIM architecture is designed to take care of both translation and rotational motion. To do that it houses two FMT* based algorithms, one handles the translation part (IFMT*-I) and another handles the attitude part (IFMT*-II).
AMPSGIM adaptation included decoupling of translation and attitude dynamics and finding a way to couple the translation trajectories and attitude trajectories to produce good coverage characteristics. TPBVPs are dealt with and for translation motion, a polynomial shape-based optimization is used. Whereas for attitude motion, a database is built which stored N - number of attitude maneuvers; both slew and tracking maneuvers. This database is used during the search phase of the algorithm, by doing this, the computation required to solve TPBVPs for attitude motion is reduced. Trajectory feasibility constraints, control constraints along mission requirements constraints are effectively added into the framework to work only with feasible solutions. To deal with multi-objective functions involved in the inspection problem, the pareto front solution method is applied inside the search algorithm.

In chapter 5 the validity of the method was verified by applying AMPSGIM to a single satellite case for different target sizes and found that the framework produces solutions that can be used to map the target surface. The impact of various parameters on the performance of the framework is studied and useful observations were reported. The framework's ability to do multi-satellite inspection was demonstrated. The advantages of using multiple satellites in a scenario where time is limited are supported by validation results. Furthermore, limitations and further development required to increase the reliability and performance of the framework are identified and discussed. In conclusion, thesis work developed a framework which can produce low thrust coupled guidance for Inspection mission using motion planning algorithms. From this work, the thesis advocates the possibilities of using sample based motion planning algorithms to solve problem involved in spacecraft proximity operations.

Bibliography

[1] David Akin and Brook Sullivan. "A survey of serviceable spacecraft failures". In: *AIAA Space 2001 Conference and Exposition*. 2001, p. 4540.

[2] Kyle Alfriend et al. *Spacecraft formation flying: Dynamics, control and navigation*. Vol. 2. Elsevier, 2009.

[3] Olov Andersson et al. "Receding-Horizon Lattice-Based Motion Planning with Dynamic Obstacle Avoidance". In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 4467–4474. DOI: 10.1109/CDC.2018.8618964.

[4] Benjamin Bernhard et al. "Coordinated Motion Planning for On-Orbit Satellite Inspection using a Swarm of Small-Spacecraft". In: *2020 IEEE Aerospace Conference*. IEEE. 2020, pp. 1–13.

[5] John T Betts. "Survey of numerical methods for trajectory optimization". In: *Journal of guidance, control, and dynamics* 21.2 (1998), pp. 193–207.

[6] R Bevilacqua, T Lehmann, and Marcello Romano. "Development and experimentation of LQR/APF guidance and control for autonomous proximity maneuvers of multiple spacecraft". In: *Acta Astronautica* 68.7-8 (2011), pp. 1260–1275.

[7] BUSEK. *BUSEK SPACE PROPULSION AND SYSTEMS*. URL: http://www.busek.com/technologies__main.htm. (accessed: 5.07.2021).

[8] John Canny et al. *On the complexity of kinodynamic planning*. Tech. rep. Cornell University, 1988.

[9] Andrea Capannolo et al. "Challenges in LICIA Cubesat trajectory design to support DART mission science". In: *Acta Astronautica* 182 (2021), pp. 208–218.

[10] Francesco Capolupo and Pierre Labourdette. "Receding-horizon trajectory planning algorithm for passively safe on-orbit inspection missions". In: *Journal of Guidance, Control, and Dynamics* 42.5 (2019), pp. 1023–1032.

[11] Jan van Casteren and M Novara. "BepiColombo mission". In: *Memorie della Societa Astronomica Italiana* 82 (2011), p. 394.

[12] AF Cheng et al. "Dart: Double asteroid redirection test". In: *European Planetary Science Congress*. Vol. 7. 2012, pp. 23–28.

[13] WH Clohessy and RS Wiltshire. "Terminal guidance system for satellite rendezvous". In: *Journal of the Aerospace Sciences* 27.9 (1960), pp. 653–658.

[14] Bruce A Conway. *Spacecraft trajectory optimization*. Vol. 29. Cambridge University Press, 2010.

[15] Joshua P Davis, John P Mayberry, and Jay P Penn. "On-orbit servicing: Inspection, repair, refuel, upgrade, and assembly of satellites in space". In: *The Aerospace Corporation, report* (2019).

[16] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. "Robust hybrid control for autonomous vehicle motion planning". In: *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*. Vol. 1. IEEE. 2000, pp. 821–826.

[17] Jonathan D Gammell and Marlin P Strub. "Asymptotically Optimal Sampling-Based Motion Planning Methods". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4 ().

[18] Pini Gurfil. "Relative motion between elliptic orbits: generalized boundedness conditions and optimal formationkeeping". In: *Journal of Guidance, Control, and Dynamics* 28.4 (2005), pp. 761–767.

[19] Barton C Hacker and James M Grimwood. *On the shoulders of Titans: A history of Project Gemini*. Vol. 4203. National Aeronautics and Space Administration, 1977.

[20] Christian Hofmann and Francesco Topputo. "Rapid Low-Thrust Trajectory Optimization in Deep Space Based on Convex Programming". In: *Journal of Guidance, Control, and Dynamics* (2021), pp. 1–10.

[21] Lucas Janson et al. "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions". In: *The International journal of robotics research* 34.7 (2015), pp. 883–921.

[22] John L Junkins and Hanspeter Schaub. *Analytical mechanics of space systems*. American Institute of Aeronautics and Astronautics, 2009.

[23] Sertac Karaman and Emilio Frazzoli. "Optimal kinodynamic motion planning using incremental sampling-based methods". In: *49th IEEE conference on decision and control (CDC)*. IEEE. 2010, pp. 7681–7687.

[24] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The international journal of robotics research* 30.7 (2011), pp. 846–894.

[25] Lydia E Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[26] Yoshiaki Kuwata et al. "Real-time motion planning with applications to autonomous urban driving". In: *IEEE Transactions on control systems technology* 17.5 (2009), pp. 1105–1118.

[27] DS Lauretta et al. "OSIRIS-REx: sample return from asteroid (101955) Bennu". In: *Space Science Reviews* 212.1 (2017), pp. 925–984.

[28] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[29] Steven M LaValle et al. "Rapidly-exploring random trees: A new tool for path planning". In: (1998).

[30] Steven M LaValle and James J Kuffner Jr. "Randomized kinodynamic planning". In: *The international journal of robotics research* 20.5 (2001), pp. 378–400.

[31] Andrew M Long, Matthew G Richards, and Daniel E Hastings. "On-orbit servicing: a new value proposition for satellite design and operation". In: *Journal of Spacecraft and Rockets* 44.4 (2007), pp. 964–976.

[32] Jacques Louet and Stefano Bruzzi. "ENVISAT mission and system". In: *IEEE 1999 International Geoscience and Remote Sensing Symposium. IGARSS'99 (Cat. No. 99CH36293)*. Vol. 3. IEEE. 1999, pp. 1680–1682.

[33] Michele Maestrini and Pierluigi Di Lizia. "Guidance for Autonomous Inspection of Unknown Uncooperative Resident Space Object". In: *2020 AAS/AIAA Astrodynamics Specialist Conference*. 2020, pp. 1–13.

[34] David Q Mayne et al. "Constrained model predictive control: Stability and optimality". In: *Automatica* 36.6 (2000), pp. 789–814.

[35] Colin R McInnes. "Potential function methods for autonomous spacecraft guidance and control". In: *Adv. Astronaut. Sci.* 90 (1996), pp. 2093–2109.

[36] Tim Mercy, Ruben Van Parys, and Goele Pipeleers. "Spline-based motion planning for autonomous guided vehicles in a dynamic environment". In: *IEEE Transactions on Control Systems Technology* 26.6 (2017), pp. 2182–2189.

[37] Daniel Morgan, Soon-Jo Chung, and Fred Y Hadaegh. "Model predictive control of swarms of spacecraft using sequential convex programming". In: *Journal of Guidance, Control, and Dynamics* 37.6 (2014), pp. 1725–1740.

[38] Daniel Morgan et al. "Swarm-keeping strategies for spacecraft under J2 and atmospheric drag perturbations". In: *Journal of Guidance, Control, and Dynamics* 35.5 (2012), pp. 1492–1506.

[39] NASA. *NASA SCIENCE SHARE THE SCIENCE*. URL: `https://science.nasa.gov/toolkits/spacecraft-icons`. (accessed: 21.06.2021).

[40] NASA. *NASA SCIENCE SOLAR SYSTEM EXPLORATION*. URL: `https://solarsystem.nasa.gov/missions/mars-cube-one/in-depth/`. (accessed: 22.06.2021).

[41] Kazutaka Nishiyama et al. "The ion engine system for hayabusa2". In: *Proceedings of the 32nd International Electric Propulsion Conference, Wiesbaden, Germany*. 2011, pp. 11–15.

[42] Dillon O'Reilly, Georg Herdrich, and Darren F Kavanagh. "Electric Propulsion Methods for Small Satellites: A Review". In: *Aerospace* 8.1 (2021), p. 22.

[43] Anastassios E Petropoulos and James M Longuski. "Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories". In: *Journal of Spacecraft and Rockets* 41.5 (2004), pp. 787–796.

[44] L Prockter et al. "The NEAR shoemaker mission to asteroid 433 eros". In: *Acta Astronautica* 51.1-9 (2002), pp. 491–500.

[45] Md Mahbubur Rahman, Leonardo Bobadilla, and Brian Rapp. "Sampling-based planning algorithms for multi-objective missions". In: *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE. 2016, pp. 709–714.

[46] John H Reif. "Complexity of the mover's problem and generalizations". In: *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. IEEE Computer Society. 1979, pp. 421–427.

[47] Honeybee Robotics. *Honeybee Robotics' Microsat CMG modules*. URL: `https://www.honeybeerobotics.com/products/attitude-control-systems/`. (accessed: 5.07.2021).

[48] Daniel J Scheeres. "Close proximity operations at small bodies: orbiting, hovering, and hopping". In: *Mitigation of Hazardous Comets and Asteroids* (2004), pp. 313–337.

[49] James A Sethian. "A fast marching level set method for monotonically advancing fronts". In: *Proceedings of the National Academy of Sciences* 93.4 (1996), pp. 1591–1595.

[50] Joseph A Starek et al. "Fast, safe, propellant-efficient spacecraft motion planning under Clohessy–Wiltshire–Hill dynamics". In: *Journal of Guidance, Control, and Dynamics* 40.2 (2017), pp. 418–438.

[51] Joseph A Starek et al. "Spacecraft autonomy challenges for next-generation space missions". In: *Advances in Control System Technology for Aerospace Applications*. Springer, 2016, pp. 1–48.

[52] Joseph Alexander Starek. *Sampling-based Motion Planning for Safe and Efficient Spacecraft Proximity Operations*. Stanford University, 2016.

[53] David Surovik and Daniel Scheeres. "Heuristic search and receding-horizon planning in complex spacecraft orbit domains". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 25. 1. 2015.

[54] Ehsan Taheri and Ossama Abdelkhalik. "Initial three-dimensional low-thrust trajectory design". In: *Advances in Space Research* 57.3 (2016), pp. 889–903.

[55] Ravi Teja Nallapu and Jekanthan Thangavelautham. "Attitude control of spacecraft swarms for visual mapping of planetary bodies". In: *2019 IEEE Aerospace Conference*. IEEE. 2019, pp. 1–16.

[56] Chris Urmson et al. "Autonomous driving in urban environments: Boss and the urban challenge". In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466.

[57] J Veverka et al. "The landing of the NEAR-Shoemaker spacecraft on asteroid 433 Eros". In: *Nature* 413.6854 (2001), pp. 390–393.

[58] Bradley Wall. "Shape-based approximation method for low-thrust trajectory optimization". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. 2008, p. 6616.

[59] Bradley J Wall and Bruce A Conway. "Shape-based approach to low-thrust rendezvous trajectory design". In: *Journal of Guidance, Control, and Dynamics* 32.1 (2009), pp. 95–101.

[60] Sei-ichiro Watanabe et al. "Hayabusa2 mission overview". In: *Space Science Reviews* 208.1 (2017), pp. 3–16.

[61] Seiichiro Watanabe et al. "Hayabusa2 arrives at the carbonaceous asteroid 162173 Ryugu—A spinning top–shaped rubble pile". In: *Science* 364.6437 (2019), pp. 268–272.

[62] William S Widnall et al. *Apollo Guidance Navigation and Control: Guidance System Operations Plan for Manned CM Earth Orbital and Lunar Missions Using Program COLOSSUS I and Program COLOSSUS IA*. Tech. rep. Technical Report R-577 Section 3, MIT Instrumentation Laboratory, Cambridge, MA, 1968.

[63] Bong Wie. *Space vehicle dynamics and control*. American Institute of Aeronautics and Astronautics, 2008.

[64] Katsuhiko Yamada et al. "New state transition matrix for formation flying in J2-perturbed elliptic orbits". In: *Journal of guidance, control, and dynamics* 35.2 (2012), pp. 536–547.

[65] Tomohiro Yamaguchi et al. "Hayabusa2-Ryugu proximity operation planning and landing site selection". In: *Acta Astronautica* 151 (2018), pp. 217–227.

[66] Ding Zhou et al. "Translational and rotational motion planning for spacecraft close proximity using sampling-based methods". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 233.10 (2019), pp. 3680–3699.