Trajectory Optimization for 3D Scene Reconstruction

Aman Tiwary

A thesis
submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

University of Washington

2024

Committee:

Behçet Açikmeşe

Sawyer Fuller

Santosh Devasia

Program Authorized to Offer Degree:

Mechanical Engineering

University of Washington

Abstract

Trajectory Optimization for 3D Scene Reconstruction

Aman Tiwary

Chair of the Supervisory Committee:
Professor Behçet Açikmeşe
Department of Aeronautics & Astronautics

The work presents an approach to generate a dynamically feasible trajectory for UAVs to perform continuous image acquisition along the trajectory. When given a rough 3D scene proxy, the algorithm generates trajectory and required sensor poses, which yields a high-quality 3D scene reconstruction. We introduce a novel formulation for viewpoint generation, addressing the minimum viewpoint coverage problem in two stages. In the first stage, we solve the linear program with linear constraints that enforce the completeness of 3D reconstruction. In the second stage, we introduce nonlinear integer constraints on the sensor variables, linearize them around the solution of the first stage, and solve a sequence of convex subproblems until convergence. The solution leads to an optimal or near-optimal subset of selected sensors. These viewpoints are then utilized as waypoints to solve a constrained optimal control problem.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

CSTC: Continuous State Triggered Constraint

CTCS: Continuous time constraint satisfaction

DOF: Degree of Freedom

ETE: Explore-then-Exploit

GNC: Guidance, Navigation, and Control

HPR: Hidden Point Removal

LOS: Line of Sight

MVS: Multiview Stereo

PCL: Point Cloud Library

PTR: Penalized Trust Region

SCP: Sequential Convex Programming

SCvx: Successive Convexification

SFM: Structre from Motion

SIP: Sequential Integer Programming

SOCP: Second-Order Cone Programming

STC: State Triggered Constraint

STL: Signal Temporal Logic

TSP: Traveling Salesman Problem

UAV: Unmanned Aerial Vehicle

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Prof. Behçet Açıkmeşe, for his unwavering guidance, support, and encouragement throughout the course of my research. His expertise and insights have been invaluable in shaping my understanding and approach to the complex challenges I encountered.

I am also incredibly thankful to my lab mates—Chris, Skye, Jason, Fabio, Samet, and Ben. Their camaraderie, collaboration, and insightful discussions have made this journey not only intellectually stimulating but also enjoyable. They have played a crucial role in the development of this work, and I am fortunate to have had the opportunity to work with such talented and hard-working individuals.

I want to express my heartfelt thanks to my best friend, Shikha Singh, for being there and supporting me at every step. Your unwavering belief in me has been a constant source of strength, and I couldn't have done this without you.

Lastly, I owe everything to my family for their endless love, encouragement, and sacrifices. Your support has been the foundation of all my accomplishments, and I am deeply grateful for everything you've done to help me reach this point.

Chapter 1

INTRODUCTION

Visual perception, a key element in interpreting the three-dimensional structure of the environment, is instrumental in decision-making and task execution for humans and robots. While humans have a natural ability to process visual information for future action planning, robots rely on sophisticated algorithms to interpret data from range sensors such as stereo cameras, LiDAR, etc., enabling them to make planning decisions to achieve similar goals.

1.1 Motivation and Problem Statement

Over the last few decades, autonomous robots have seen significant advancement. Coverage and surveillance have become critical problem domains where Unmanned Aerial Vehicles (UAVs) are extensively deployed. These UAVs are tasked with covering specific ground areas for crucial applications, including but not limited to wilderness search and rescue operations [7, 36, 39, 60], surveillance [24], delivery [4, 17, 26], site inspection [69], and scene exploration and 3D reconstruction [5,10,21,27–29,35,48,52,63,64], among others. UAVs have become indispensable in the realm of autonomous robotics due to their versatility, efficiency, and, most importantly, their cost-effectiveness. Their ability to perform various tasks and their reliability make them crucial in the above missions.

Despite these benefits, one of the critical challenges in autonomous UAV navigation is effectively handling occlusions. Occlusions can significantly hinder achieving complete coverage of every point in the target environment and create dense environment mapping. Therefore, a significant challenge to facilitate autonomous coverage planning capabilities would be to efficiently cover a given target area.

Most off-shelf planners generate conservative trajectories like an orbit, spiral, or lawn-

mower pattern to cover the scene, irrespective of the geometry of the scene under consideration, and are unable to handle complex structures. Model-based methods for 3D scene reconstruction employ a two-phase mission planning framework, *Explore-then-Exploit* (ETE). In the first phase, a 3D scene proxy is generated by exploratory flight. The primary effort in acquisition planning is then devoted to the next phase, which focuses on a detailed scene reconstruction [15, 28, 48, 64]. Traditionally, acquisition planning further involves two stages. The first involves identifying a "good" set of viewpoints around the object, and the second involves planning a path that goes through all the selected points.

This work adopts a two-phase mission planning framework, *Explore-then-Exploit* (ETE) for 3D scene reconstruction. For the second phase, a two-step optimization approach is proposed to generate a set of optimum viewpoints and a dynamically feasible trajectory for continuous image acquisition along the trajectory.

1.2 Primary Contribution

A systematic optimization-based framework is proposed for model-based trajectory planning for 3D scene reconstruction. This framework aims to identify a "good" set of viewpoints and compute a dynamically feasible trajectory that passes through these viewpoints. Two key contributions of this work include

- A novel formulation is introduced for solving integer linear programs and used to solve the minimum viewpoint problem.

- An optimization-based trajectory planning method is used that encodes the *line-of-sight* constraint in the trajectory planning problem.

1.3 Thesis Outline

This thesis is divided into three branches. The first branch, covered in Chapter 2, defines the pre-processing methods to be applied to the input 3D scene proxy for identifying clusters of interest for data acquisition. The proposed two-step optimization is detailed in Chapters 3

and 4. Chapter 3 introduces the formulation of the minimum viewpoint problem and presents a novel approach for solving integer linear programs. Chapter 4 discusses the formulation of trajectory optimization for continuous data acquisition. Chapter 5 integrates the methods proposed in chapters 2 to 4, demonstrating how they generate a dynamically feasible trajectory with coverage guarantees. Chapter 6 discusses some limitations and concludes with future scope.

Chapter 2

GEOMETRY PROXY AND SEGMENTATION FOR OBJECT IDENTIFICATION

2.1 Introduction

The following chapter outlines the primary steps in processing and analyzing 3D environment data to identify clusters of interest. These clusters play a crucial role in the second phase of mission planning, i.e., viewpoint and trajectory optimization.

The process begins with obtaining a geometric proxy, followed by ground filtering, segmentation, and generating convex hulls. These steps are crucial for isolating relevant objects in the environment. The methods used to represent the data are fundamental as they influence the modeling and analysis of the environment.

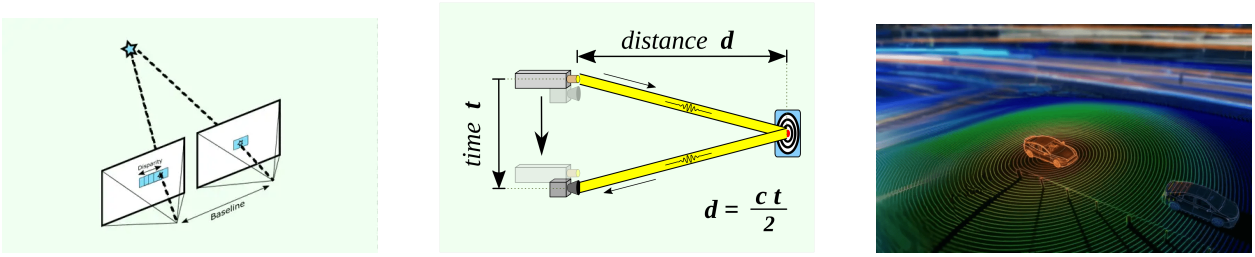
The remainder of this chapter is structured as follows: Section 2.2 discusses different visual sensors used to capture 3D information about the environment and different data structures that are used to represent the captured information. Section 2.3 and Section 2.4 discusses ground filtering and segmentation operation on the input cloud. Section 2.5.

2.2 Geometry proxy axquisition

A geometry proxy is a simplified, coarse representation of the environment that can be efficiently processed and analyzed. Typically, the geometry proxy is generated from the data acquired by range sensors, and it serves as the foundation This data captures the spatial distribution of surfaces in the environment, providing a comprehensive view of the scene.

2.2.1 Depth Sensor Types

The creation of a geometry proxy relies on the type of depth sensors are used. The data collected measures the distance between the sensor and various points in the environment. The type of depth sensor to be used depends on the application as each has its own strengths and limitations. The most common types include stereo cameras, Time-of-Flight(ToF) sensors, and LiDAR.



(a) Stereo Sensor (b) Time of Flight Sensor (c) Light Detection Ranging

Figure 2.1: Types of Depth Sensors(source: [2])

Stereo Cameras

Stereo sensors use two or more cameras placed at a fixed distance apart, called the baseline, to capture images from slightly different viewpoints. The disparity between these images is analyzed, and the depth of each point in the scene is calculated using triangulation.

These sensors are relatively inexpensive compared to other depth sensors, making them accessible for a wide range of applications. They capture both depth and color information simultaneously, which is helpful for tasks that require both 3D structure and visual details, such as object recognition. Unlike active sensors, stereo cameras rely on ambient light. Thus, the accuracy of the depth estimation relies on good lighting conditions. The accuracy of depth measurement decreases with distance, and stereo cameras are generally less effective for long-range applications compared to LiDAR or ToF sensors.

Time-of-Flight (ToF) Sensors

Time-of-flight sensors measure the time taken by the light pulse to travel from the sensor to an object and back. This delay is directly proportional to the distance, allowing the sensor to calculate the depth of each point in the scene.

ToF sensors can capture depth information at high frame rates, making them suitable for real-time applications. These sensors provide accurate depth measurements even in low-light conditions. These sensors are typically small and lightweight, making them easy to integrate into various devices. ToF sensors can be affected by interference, thus reducing accuracy in certain environments. Unlike stereo cameras, ToF sensors typically have lower resolution, which may limit their effectiveness in applications requiring detailed depth maps. Inaccuracies may also by the presence of highly reflective or absorptive surfaces.

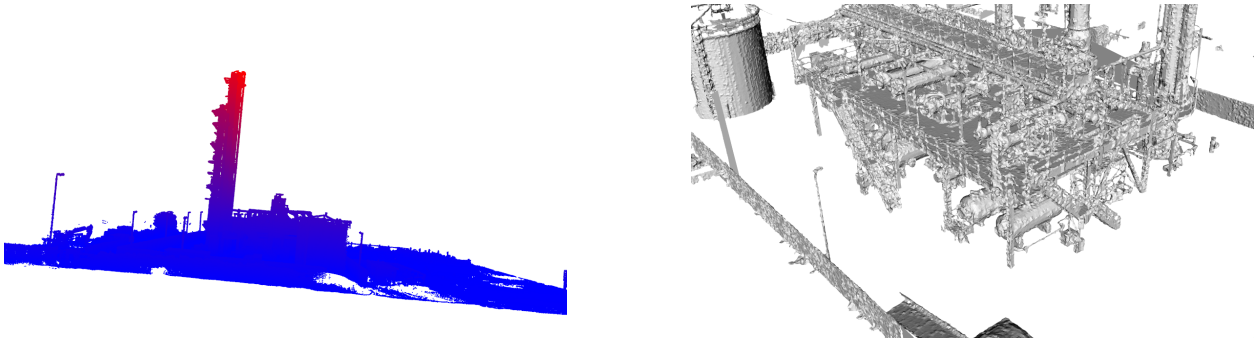
Light Detection and Ranging (LiDAR)

LiDAR sensors emit laser pulses and measure the time it takes for the pulse to bounce off surfaces and return to the sensor. Scanning the environment with LiDAR creates a highly accurate 3D map or point cloud of the surroundings.

LiDAR provides precise distance measurements over long ranges, making it ideal for mapping a large environment. They operate independently of the ambient light, thus allowing them to function in both low and bright-light conditions. Unlike stereo cameras and ToF sensors, LiDAR is expensive and bulkier, thus limiting their use in certain applications. Data captured by LiDAR demands significant computational resources for processing and analysis.

2.2.2 Data Representation Methods

After the 3D data has been captured using the depth sensors mentioned above, it must be represented in a form that is suitable for processing and analysis, as the choice of data structure will strongly affect the strategy for view-path planning [63].



(a) Chemical Plant: Point Cloud (b) Chemical Plant: Mesh

Figure 2.2: Data representations

Point Cloud Representation

A point cloud is one of the most common representations used for data collected by the majority of the depth sensors. It stores spatial information defined by x, y, and z coordinates and can also include additional attributes like reflection intensity and color information within the data structure. This versatility makes point clouds a preferred representation for 3D mapping and inspection. Point clouds are relatively easy to work with, making them ideal for real-time applications. However, the data collected in this format is unstructured, meaning there is no inherent connection between individual points. As a result, further processing, such as generating an octree [42], is often necessary to establish relationships between neighboring points and to facilitate tasks like spatial indexing and efficient search.

Mesh Representation

A mesh is a representation of a 3D object using vertices, edges, and faces, typically in the form of triangles. It offers a continuous and connected representation of surfaces. Some view planning approaches [14, 44] use a triangle mesh to represent the environment in the intermediate step. However, generating and processing mesh is more complicated than handling point clouds. Meshes are usually generated from point clouds, and this conversion can

introduce errors.

2.3 Ground Filtering

Once the geometry proxy has been generated, the next step involves filtering out ground points. This is done to isolate objects above the ground, enabling more effective clustering and analysis. There are many ground filtering algorithms available [62, 66, 67], and an implementation of Progressive Morphological Filter [66] in point cloud library (PCL) [49] is used for removing ground from the input point cloud. Details about the filter can be found in [66].

2.4 Segmentation

With the ground points filtered out, the remaining point cloud consists of objects in the environment. Euclidean clustering groups points based on their spatial proximity, forming clusters that ideally represent distinct objects. The selection of clustering parameters like minimum cluster size, maximum cluster size and distance threshold are critical in ensuring accurate identification of these objects.

Once the Segmentation is done, the goal is to focus on the clusters of interest, depending on the application. Either a user can define the cluster of interest, or criteria such as cluster size, location, and geometric features can be used to prioritize clusters.

2.5 Conclusion

This chapter outlined the process of creating a geometric proxy and using it for filtering and clustering operations in a 3D environment. By understanding the different types of depth sensors, as well as data representation methods like point clouds and meshes, we can effectively capture and analyze the environment. In this work, a point cloud representation of the environment is used. The identified clusters of interest form the basis for further operations, such as viewpoint and trajectory optimization, ensuring that the drone interacts with its environment in an informed and efficient manner.

Chapter 3

INTEGER PROGRAMMING FRAMEWORK FOR VIEW PLANNING

3.1 Introduction

Identifying "good," "ideal," or "reasonable" imaging positions is a crucial research focus across disciplines such as robotics, computer graphics, computer vision, and photogrammetry [28, 35]. In applications that require comprehensive surface modeling, such as object inspection or scene reconstruction, multiple sensor readings from different viewpoints must be combined to obtain a complete description of the object's surface. This challenge is particularly critical when using range sensors, which are limited in capturing an entire scene from a single viewpoint.

Sensor planning, an extensively researched area, seeks to quantify and optimize the relationship between objects being observed and the sensors observing them. In model-based tasks, sensor planning helps determine the viewpoints that maximize visibility while minimizing occlusions, leading to more efficient data collection and processing. This chapter focuses on viewpoint optimization within sensor planning, framing it as an integer linear programming problem.

The combinatorial complexity of selecting optimal viewpoints among a large set of candidates is addressed by leveraging an integer programming framework. The approach models critical factors such as visibility and occlusions to determine the best subset of viewpoints. Despite its effectiveness, the traditional integer programming approach faces scalability challenges when applied to complex environments.

To overcome these challenges, a novel approach is proposed that solves the integer linear program sequentially. This method enhances computational efficiency by narrowing the

search space, allowing the solver to converge to near-optimal solutions quickly.

This chapter is structured as follows: Section 3.2 discusses existing approaches used for view planning. Section 3.3 discusses the formal problem formulation, including objective functions and constraints. Section 3.4 examines traditional methods for solving integer programming problems and highlights their limitations in this context. Section 3.5 presents the hybrid approach integrating machine learning with integer programming techniques. Section 3.6 provides performance evaluations and results, and the concluding insights are in Section 3.7.

3.2 Related Work

Active vision involves determining the pose and configuration of a visual sensor to gather useful information for various tasks [63]. Central to active vision are techniques like view planning, sensor planning, and next-best view (NBV) determination, which are crucial for enabling robot vision systems to cover or detect target objects by processing progressively and analyzing sensory data.

These methods have been widely adopted to significantly enhance the efficiency of robots in perception-aware tasks such as inspection [10, 21, 52, 59], object reconstruction [**?**, 5, 10, 21, 35, 52, 63, 64], and scene exploration [34], where robots rely on a sequence of views from visual sensors. Over the past few decades, view planning has been extensively researched. Scott et al. [51]categorized view planning methods into two main groups: (1) model-based methods [13, 43, 59] and (2) non-model-based methods [5, 29, 40].

Model-based methods assume that a priori information about the target object or environment is available, allowing for more precise planning. In contrast, non-model-based methods rely on a sequence of NBVs to gather data incrementally without prior knowledge of the environment.

Modern approaches to view planning can be further classified into two categories: (1) search-based methods and (2) synthesis-based methods [63]. Search-based methods involve sampling candidate views and selecting optimal ones based on task-specific constraints and

goals. For instance, object and scene reconstruction tasks prioritize maximizing information gained from each view, while object detection tasks focus on minimizing uncertainty about the object. On the other hand, synthesis-based methods determine the NBV by establishing a causal relationship between sensor parameters and the task objective rather than relying on an exhaustive search.

An example of a synthesis-based approach is Connolly's work [12], which uses an octree to represent the environment, where the leaf nodes denote empty, occupied, or unseen areas. The viewpoint with the maximum unseen area is selected as the NBV. Similarly, Banta et al. [5] used a voxel representation of the environment and selected the viewpoint that covers the unknown voxels as the NBV. Although synthesis-based methods tend to have lower computational complexity than search-based methods, they are less accurate and may be unreliable for complex structures with self-occlusion.

This chapter details a search-based approach for solving the model-based view planning problem, focusing on its application to 3D scene reconstruction.

3.3 View Planning Problem

The objective is to identify a feasible set of viewpoints that, when combined, have the ability to observe every point on the surface of an object. This is a model-based problem driven by the requirements of a specific task. The models consist of details about the object in the environment and the sensors being used. This prior knowledge, along with the understanding of the task requirements that need to be satisfied, are incorporated into the view planning problem.

3.3.1 Discretization of Viewpoint Space

Before an optimization formulation can be employed to generate a view plan, the viewpoint space must be discretized to generate a finite set of candidate viewpoints that covers the target object or the environment. The viewpoint space defines the positions and orientations a sensor can take. Discretization involves sampling the points on geometric primitives like

spheres, cylinders, or other shapes depending on the characteristics of the target object.

The constraints imposed on the sensing operations are that the camera is directed to the center for a spherical viewpoint space and parallel to the ground for a cylindrical viewpoint space. The viewpoint space completely contains the object. The viewpoints are sampled at a distance that meets the sensor's depth accuracy requirements. Thus, finding a feasible set of viewpoints becomes a problem of selecting a smaller set from the sampled points distributed on the surface of the viewpoint space.

3.3.2 Evaluating Viewability

The effectiveness of any viewpoint in a view planning problem hinges on its ability to capture meaningful information from the surface of the object. This is quantified by evaluating the viewability of the surface points from a given viewpoint. In this subsection, we introduce a method to evaluate viewability using a visibility vector, which indicates which surface points are visible or measurable from the candidate viewpoint. For a surface point to be considered viewable, it must satisfy two essential conditions: occlusion and surface normal alignment.

- *Occlusion Analysis*: Occlusion occurs when part of the object, or another object in the scene, blocks the line of sight between the viewpoint and the target surface point. To determine whether a surface point is viewable, a hidden point removal filter (HPR) [32]. First, frustum culling is done to filter out the points that lie inside the view frustum, and then the HPR filter is used on the frustum culled points to identify points that are not occluded without the need for surface reconstruction. Only unoccluded surface points are marked as visible (see Figure 3.1a).

- *Surface Normal Alignment*: For a surface point to be effectively captured by the sensor, the orientation of the surface relative to the viewpoint must be considered. This is quantified by examining the angle between the surface normal vector at that point and the vector from the viewpoint to the surface point. A point is considered viewable only

if this angle falls within a threshold, ensuring that the sensor captures the surface with sufficient detail (see Figure 3.1b and 3.1c).

3.3.3 Problem Formulation

Let the set of all candidate viewpoints in the viewpoint space be denoted by $\mathcal{V}$. Each viewpoint in the set is denoted by v_i, where $i \in \{1, 2, \cdots, n\}$ and n is the total number of viewpoints sampled. Let $\mathcal{P}$ denote the set of points on the surface of the target object under consideration. Let $\mathcal{L}$ denote the set of leaf nodes in the octree representation of the target object, and each leaf node l_j has a subset of points in $\mathcal{P}$, where $j \in \{1, 2, \cdots, m\}$ and m is the total number of leaf nodes. Furthermore, let $\mathcal{S}$ denote a set which is a subset of the set $\mathcal{L}$, i.e., $s_i \in \mathcal{S}$ consists of leaf nodes $l_j \in \mathcal{L}$ which are visible from the viewpoint v_i.

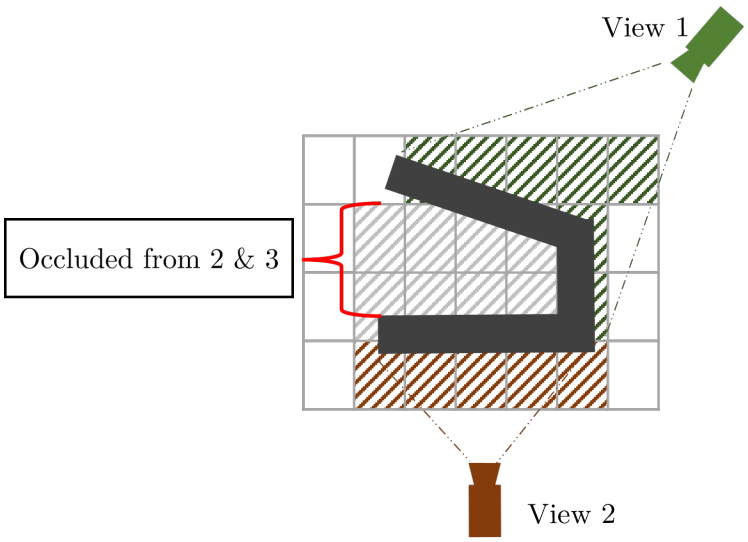
In simple terms, the view planning problem can be viewed as a set-cover-problem, i.e., there is a universal set $\mathcal{L}$ and a set of subsets of the universal set $\mathcal{S}$ such that the union of all the subsets covers the set $\mathcal{L}$. This formulation may make the problem seem simple, but it is an NP-hard problem in combinatorial optimization.

To develop an efficient algorithm, it is necessary to simplify the formulation. Octree leaf nodes are used to represent the target object or environment, enabling the problem to be formulated without accounting for the empty spaces, unlike the 3D voxel representation of the environment, thus reducing the size of the problem.

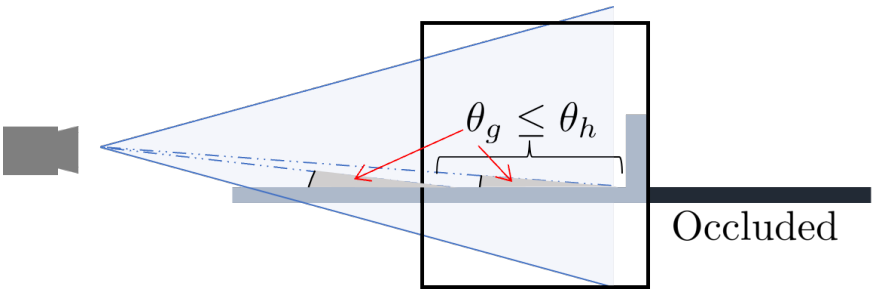
Each viewpoint v_i has a viewability vector a_i associated with it, which is an m dimensional binary vector, entries of which are given by:

$$a_{i,j} = \begin{cases} 1 & \text{if } i^{th} \text{ sensor can see } j^{th} \text{ grid} \\ 0 & \text{otherwise} \end{cases} .$$

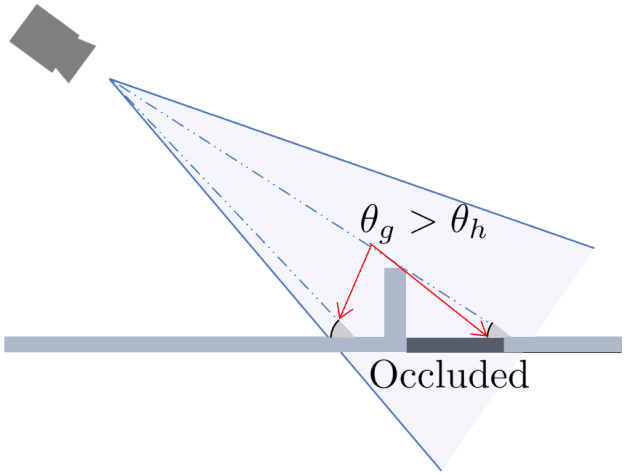
These visibility vectors are stacked to form a binary viewability matrix A of dimension $m \times n$ (see Figure 3.2).



(a) Occlusion



(b) Viewpoint coplanar with the surface to be observed, also results in self-occlusion



(c) Viewpoint with glance angle greater than threshold $\theta_g > \theta_h$

Figure 3.1: Viewablity Criteria

Algorithm 1 Generating Viewability Matrix

Input: $\mathcal{L}, \mathcal{V}$ & A.

Initialize $A \leftarrow \mathbf{0}^{m \times n}$
for $i = 1$ **to** n **do**
    $F_l \leftarrow F_c(\mathcal{L})$ ▷ frustum culling on leaf nodes to filter out nodes outside field of view
    $N_v \leftarrow Q(F_l)$. ▷ Filter out the visible nodes
    for j **in** N_v **do**
        $A(i, \textbf{argwhere}(j \in \mathcal{L})) \leftarrow 1$
    end for
end for

Output: A

3.4 Methods of Solution

In this section, a few of the methods for solving set cover problems are listed, but in this thesis, an optimization-based method is employed.

3.4.1 Integer Linear Programming

Integer Linear Programming (ILP) is an optimization formulation where both the objective and constraints are linear, and the variables are integer. The problem of view planning fits perfectly into this formulation. The viewpoints are either selected or not selected and, thus, are defined as a binary variable

$$v_i = \begin{cases} 1 & \text{if } i^{th} \text{ viewpoint is selected} \\ 0 & \text{otherwise} \end{cases}.$$

The most important constraint is the coverage constraint. Given a set of viewpoints, the coverage constraint ensures that the leaf nodes are visible by at least one of the viewpoints. Given the viewability matrix introduced in Section 3.3.3, the coverage constraint is given by,

$$Av + \delta \geq \mathbf{1},$$

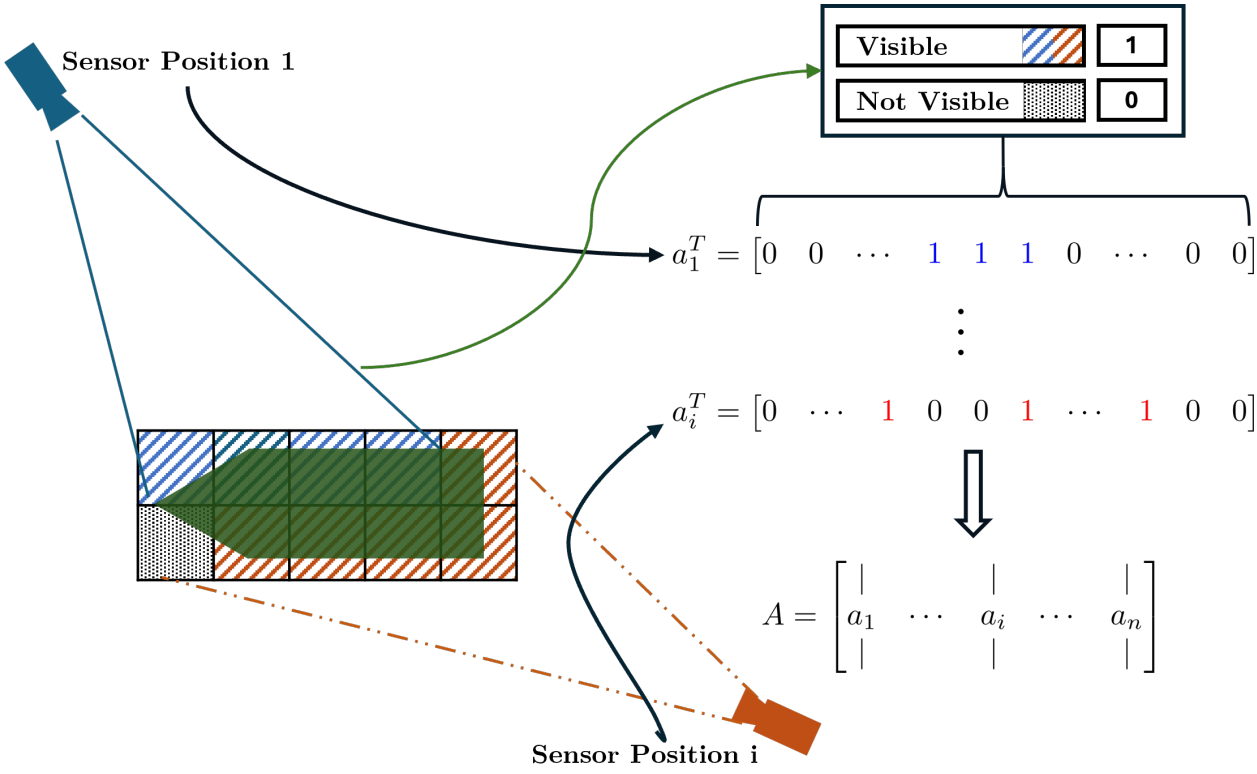


Figure 3.2: Binary formulation of the view planning problem

where $\mathbf{1}$ is an n-dimensional vector of 1's. The resulting ILP is given by,

$$\min_{v,\delta} \quad \mathbf{1_n}^T v + \mathbf{1_m}^T \delta, \tag{3.1a}$$

$$\text{s.t.} \quad Av + \delta \geq \mathbf{1}, \tag{3.1b}$$

$$v_i, \delta_k \in \{0,1\}, \quad i = \{1,2,\cdots,n\}, \quad k \in \{1,2,\cdots,m\}, \tag{3.1c}$$

$$v \in \mathbb{Z}^n_{\geq 0}, \quad \delta \in \mathbb{Z}^m_{\geq 0} \quad A \in \mathbb{Z}^{m \times n}_{\geq 0}. \tag{3.1d}$$

v is the viewpoint vector, and δ is the buffer added to coverage constraint to capture infeasibility, i.e., if k^{th} leaf node is not visible by any of the candidate viewpoints, then $\delta_k = 1$.

As set-cover-problems are NP-hard optimization problems, the computational time to solve them increases exponentially as the problem size increases. Subsequent sections detail approximation algorithms that can solve big problems with optimal or near-optimal solutions.

Also, a new formulation is introduced in Section 3.4.3, which is one of the contributions in this thesis.

3.4.2 Greedy Approximation Algorithm

Greedy algorithms are effective in getting near-optimal solutions to high-dimensional problems. It follows an iterative procedure in which, at each step, the viewpoint that can see the largest number of unseen leaf nodes in the set $\mathcal{L}$ is selected. The algorithm selects $v_i \in \mathcal{V}$, such that v_i maximizes the cardinality of the intersection between the coverage of v_i and the set of uncovered elements in $\mathcal{L}$. Once a viewpoint has been selected, the elements covered by that viewpoint are removed from the set $\mathcal{L}$ and added to an uncovered set $\mathcal{T}$. This process continues until all feasible elements in $\mathcal{L}$ are covered, i.e., all the feasible elements are in set $\mathcal{T}$.

Algorithm 2 Greedy Algorithm

Input: $\mathcal{L}, \mathcal{S}, \mathcal{T}, \&\mathcal{W}$.

1: Initialize $T \in \Phi$.
2: **while** $U \neq \Phi$ **do**
3: select $s_i \in \mathcal{S}$ that covers the maximum elements in $\mathcal{L}$
4: $\mathcal{T} \leftarrow s_i$ ▷ add s_i to $\mathcal{T}$
5: $\mathcal{W} \leftarrow v_i$ ▷ add the corresponding viewpoint to $\mathcal{W}$
6: remove s_i from $\mathcal{L}$
7: **end while**

Output: $\mathcal{W}$

3.4.3 Sequential Integer Programming

A new approach to solving ILP problems is introduced in this section. If we look at the ILP formulation carefully in Equation (3.1), it can be seen that it is the integer constraint on the variables z and δ (Equation (3.1c)) that makes the problem difficult. If we remove that

constraint temporarily, the formulation is a simple linear programming problem for which various methods exist to solve it efficiently.

We propose a 2-stage optimization framework to solve the ILP in Equation (3.1) with some reformulations. In the first stage, we focus on catching any infeasible nodes, i.e., if there are any nodes not visible by any of the sampled candidate viewpoints. We reformulate the problem as follows:

$$\min_{v,\delta} \quad \mathbf{1}^T\delta, \tag{3.2a}$$

$$\text{s.t.} \quad v \geq 0, \quad v \leq 1, \tag{3.2b}$$

$$Av + \delta \geq \mathbf{1}, \tag{3.2c}$$

$$\delta \geq 0, \tag{3.2d}$$

$$v \in \mathbb{R}^n, \quad \delta \in \mathbb{R}^m, \quad A \in \mathbb{Z}_{\geq 0}^{m \times n}. \tag{3.2e}$$

The solution to Equation (3.2) (z^*, δ^*) is used as a reference for the second stage, where integer constraint is imposed on the optimization variable. The second stage formulation is given by:

$$\min_{v,\delta} \quad \mathbf{1}^T z, \tag{3.3a}$$

$$\text{s.t.} \quad v \geq 0, \quad v \leq 1, \tag{3.3b}$$

$$Av + \delta \geq \mathbf{1}, \tag{3.3c}$$

$$\delta \geq 0, \tag{3.3d}$$

$$\boxed{\textit{Integer Constraint}}, \tag{3.3e}$$

$$\delta \leq \delta^*, \tag{3.3f}$$

$$v \in \mathbb{R}^n, \quad A \in \mathbb{R}^{m \times n}, \quad \delta \in \mathbb{R}^m. \tag{3.3g}$$

Integer Constraint

The integer constraint in Equation (3.3f) can be expressed using the following non-convex functions:(1) $min\{x, (1-x)\}$, (2) $x(1-x) = 0$, and (3) $(x(1-x) + c^2)^{1/2} - c$

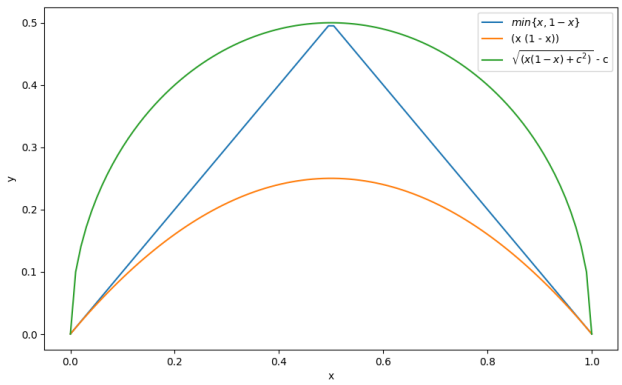


Figure 3.3: Integer constraint functions

The second-stage optimization problem is solved using the sequential convex programming (SCP) framework. SCP is a first-order optimization method that relies on solving a series of convex approximations to the original non-convex problem. One complication here is the non-convex integer constraint. The $minx, 1-x$ is not continuously differentiable, which poses difficulties for first-order methods like SCP. Instead of using $min\{x, 1-x\}$, we use the other two functions as they are differentiable.

The non-convex functions are convexified using first-order Taylor series expansion about ref v^* and δ^*. If $f(x)$ is the non-convex integer function, we define an operator $l(\cdot)$, which is given by

$$l(f(v), v^*) = f(v^*) + \left.\frac{\delta f}{\delta v}\right|_{v^*} (v - v^*). \tag{3.4}$$

3.5 Conclusion

This chapter presents a sequential integer programming framework for solving high-dimensional complex problems. This framework leverages the benefits of sequential convex programming to obtain optimal or near-optimal viewpoints that guarantee coverage, a crucial aspect in applications like inspection, surveillance, and 3D scene reconstruction.

This chapter starts by introducing the view-planning problem and draws similarities

SIP Formulation

First Stage Optimization

$$\begin{aligned}
&\underset{v,\delta}{\text{minimize}} && \mathbf{1}_m^T \delta, \\
&\text{subject to.} \\
&\underline{\text{Box Constraint:}} \\
&&& v \geq 0, \quad v \leq 1, \\
&\underline{\text{Non Negative Constraint:}} \\
&&& \delta \geq 0, \\
&\underline{\text{Coverage Constraint:}} \\
&&& Av + \delta \geq 1
\end{aligned}$$

Second Stage Optimization

$$\begin{aligned}
&\underset{v,\delta}{\text{minimize}} && \mathbf{1}_n^T v + w_{TR}||v - v^*||_2^2 + w_{vb}||v_{vb}||_1 \\
&\text{subject to.} \\
&\underline{\text{Box Constraint:}} \\
&&& v \geq 0, \quad v \leq 1, \\
&&& \delta \geq 0, \quad \delta \leq \delta^*, \\
&\underline{\text{Coverage Constraint:}} \\
&&& Av + \delta \geq 1, \\
&\underline{\text{Integer Constraint:}} \\
&&& f(v_i^*) + \left.\frac{\delta f}{\delta v}\right|_{v_i^*} (v_i - v_i^*) = v_{vb,i}, \quad i \in \{1, 2, \cdots, n\}
\end{aligned}$$

Algorithm 3 SIP Algorithm

Input: A, convergence tolerances $(\epsilon_{tr}, \epsilon_{vb})$

$\bar{v}, \delta^* \leftarrow$ **argmin{Stage-1 Formulation}**
while not converged **do**
    $v^* \leftarrow$ **argmin{Stage-2 Formulation}**
    if $||v - \bar{v}||_2 < \epsilon_{tr}$ **and** $||v_{vb}||_1 < \epsilon_{vb}$ **then**
        converged
    end if
    $\bar{v} \leftarrow v^*$
end while

Output: v^*

with the set cover problem, highlighting the inherent challenges due to its NP-hard nature. Some approximation methods that are used to solve these problems are listed. A novel sequential approach is then introduced. The proposed two-stage optimization framework efficiently handles the computational complexity by first solving the relaxed version, i.e., without integer constraint, and then solving the minimum viewpoint problem. Using the SCP framework allows for handling the non-convex integer constraint by linearizing it about the solution of stage 1 formulation and iteratively refines the solution to ensure integer constraints are satisfied at convergence.

This framework not only provides a systematic approach to solving the view planning problem but also opens up avenues for handling large-scale problems. This chapter lays the foundation for further exploration. While this method requires more thorough study and testing, it has the potential to set the stage for future work, including the development of joint optimization strategies that could be applied to other classes of problems involving integer constraints.

Chapter 4

TRAJECTORY OPTIMIZATION VIA SUCCESSIVE CONVEXIFICATION

4.1 Introduction

Trajectory generation is the computation of a time-varying, multidimensional state and control signal that satisfies specifications while optimizing key mission objectives [37]. This work focuses explicitly on dynamically feasible trajectories that adhere to the equations of motion of the vehicle under consideration. This chapter presents a sequential convex programming (SCP) based trajectory planning algorithm that solves minimum-time, 6-DoF UAV motion with line of sight (LoS) constraints. An essential contribution of this chapter is formulating an LoS constraint that ensures continuous-time satisfaction [19] (between discrete sample points).

Solving autonomous guidance problems quickly and reliably presents several challenges. First, these problems are inherently nonlinear and non-convex; second, the accuracy of the discretization scheme heavily influences the validity of numerical solutions; third, selecting an appropriate reference trajectory to initialize the iterative solution process can be difficult [55].

Convex optimization-based guidance has been considered a prime candidate for on-board autonomous guidance applications due to its deterministic behavior, global optimality guarantees, ability to provide certificates of convergence and infeasibility, and availability of efficient interior point method (IPM) algorithms to solve convex problems in real-time [55]. Successive convexification (SCvx) techniques can handle a general class of non-convex optimal control problems but at the cost of increased computational complexity and weakened optimality and convergence guarantees. These methods transform the non-convex problem into a sequence of convex subproblems that are solved iteratively until convergence [31,37,38,41,55].

This chapter employs SCvx for trajectory optimization. The process starts by using a simple, dynamically inconsistent initial trajectory about which the problem is linearized and discretized to generate a convex Second-Order Cone Programming (SOCP) subproblem. This problem is then solved iteratively, where the solution of the previous iteration is used as a reference to update the convex subproblem until convergence to a local optimal solution. The converged solution will satisfy the original nonlinear dynamics and constraints. The remainder of this chapter is structured as follows: Section 4.2 defines the continuous-time non-convex problem description. Section 4.3 discusses the convex reformulation. Section 4.4 defines the discrete-time convex optimization subproblem.

4.2 Problem Formulation

The following section describes the 6-DoF minimum-time problem for UAV in its original continuous-time non-convex form. We use the following two frames:

- $\mathcal{I}$: Inertially-fixed frame (North-West-Up coordinate frame)
- $\mathcal{B}$: Body-fixed frame, centered at vehicle's center-of-mass (North-East-Down frame)

Variables of the form $\square_{\mathcal{F}}$ are quantities expressed in frame $\mathcal{F}$. The remainder of this section details the dynamics and kinematics, state and control constraints imposed on the trajectory, and the boundary conditions enforced at the beginning and end of the trajectory. The section ends with a brief summary of the problem description.

4.2.1 6-DoF UAV Dynamics

Here, the vehicle is treated as a rigid body subject to constant acceleration due to gravity, $g_{\mathcal{I}} \in \mathbb{R}^3$ and negligible aerodynamic resistance. The 6-DoF vehicle dynamics are based on the equations given in [54], and the notation therein is adopted in this work. The variables $r_{\mathcal{I}}$, $v_{\mathcal{I}}$, $q_{\mathcal{B}/\mathcal{I}}$ and $\omega_{\mathcal{B}}$ model the vehicle states $\zeta \in \mathbb{R}^{13}$, and $T_{\mathcal{B}}$ and $\tau_{\mathcal{B}}$ are the control inputs

$u \in \mathbb{R}^6$. The vehicle's attitude dynamics are given by:

$$\dot{q}_{\mathcal{B}/\mathcal{I}}(t) = \frac{1}{2}\Omega(\omega_{\mathcal{B}}(t))q_{\mathcal{B}/\mathcal{I}}(t), \tag{4.1}$$

$$\dot{\omega}_{\mathcal{B}}(t) = J_{\mathcal{B}}^{-1}\left(\tau_{\mathcal{B}}(t) - [\omega_{\mathcal{B}}(t)\times]\, J_{\mathcal{B}}\omega_{\mathcal{B}}(t)\right), \tag{4.2}$$

where $\Omega(\cdot)$ is a 4×4 skew-symmetric matrix of $\omega_{\mathcal{B}}$, which is the angular velocity vector of $\mathcal{B}$ relative to $\mathcal{I}$ expressed in $\mathcal{B}$ coordinates. $J_{\mathcal{B}} \in \mathbb{S}^3_{++}$ is the vehicle's inertia about $x_{\mathcal{B}}$, $y_{\mathcal{B}}$, and $z_{\mathcal{B}}$ and $m \in \mathbb{R}_{++}$ is vehicle's mass.

The position and velocity of the vehicle are expressed in $\mathcal{I}$ and are denoted by $r_{\mathcal{I}}(t) \in \mathbb{R}^3$ and $v_{\mathcal{I}}(t) \in \mathbb{R}^3$, respectively. The vehicle's translation dynamics are given by:

$$\dot{r}_{\mathcal{I}}(t) = v_{\mathcal{I}}(t), \tag{4.3}$$

$$\dot{v}_{\mathcal{I}}(t) = \frac{1}{m}\left(C_{\mathcal{B}/\mathcal{I}}(t)T_{\mathcal{B}}(t)\right) + g_{\mathcal{I}}, \tag{4.4}$$

where $C_{\mathcal{B}/\mathcal{I}}(\cdot)$ is a direction cosine matrix which is associated with $q_{\mathcal{B}/\mathcal{I}}(t)$, that encodes the attitude transformation from $\mathcal{B}$ to $\mathcal{I}$.

4.2.2 Global State Constraints

1. Box Constraint on position:

$$r_{min}, r_{max} \in \mathbb{R}^3,$$

$$r_{min} \leq r_{\mathcal{I}}(t) \leq r_{max}, \tag{4.5}$$

2. Maximum Speed Constraint:

$$||v_{\mathcal{I}}(t)||_2 \leq v_{max}, \tag{4.6}$$

3. Angular Rate Bounds:

$$\omega_{max} \in \mathbb{R}^3$$

$$||M_i\omega_{\mathcal{B}}, (t)||_\infty \leq \omega_{max,i}, \tag{4.7}$$

where $M_i \in \mathbb{R}^{3\times3}$, $i \in \{1,2,3\}$ and

$$M_i = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}, \quad m_{jk} = \begin{cases} 1 & i = j = k \\ 0 & \text{otherwise} \end{cases},$$

4. Maximum Tilt Constraint:

$$\cos\theta_{max} + \tilde{z}_{\mathcal{I}}^T (q_{\mathcal{B}/\mathcal{I}} \otimes \tilde{z}_{\mathcal{B}} \otimes q_{\mathcal{B}/\mathcal{I}}^*) \leq 0, \tag{4.8}$$

where θ_{max} is the maximum allowable tilt angle and $\tilde{z}_{\mathcal{I}}$ and $\tilde{z}_{\mathcal{B}}$ are pure quaternions defined as $\tilde{z}_{\square} = [0, z_{\square}]^T$ and $z_{\square} = [0,0,1]^T$.

5. Obstacle Avoidance Constraint: The obstacle is modeled as a convex keep-out zone (like a cylinder, Polyhedra etc). For simplicity, cylindrical keep-out zone is used. The keep-out zone constraint is given by:

$$p(r_{\mathcal{I}}(t)) = 1 - (r_{\mathcal{I}}(t) - r_{\mathcal{T}})^T A^{-1} (r_{\mathcal{I}}(t) - r_{\mathcal{T}}) \leq 0, \tag{4.9}$$

where $A \in \mathbb{R}^{3\times3}$ is matrix representation of the cylinder $\mathcal{T}$ as shown in Figure 4.1.

4.2.3 Line of Sight Constraint

For the line of sight constraints, we try to look at the direction perpendicular to the axis of the Target object (see Figure 4.1).

$$d = r_{\mathcal{I}} - r_{\mathcal{T}},$$
$$\lambda = < d, \hat{a} >,$$

the projection of d on $\hat{a}$ is given by :

$$\lambda\hat{a} = < d, \hat{a} > \hat{a} = \hat{a}\hat{a}^T d, \tag{4.10}$$

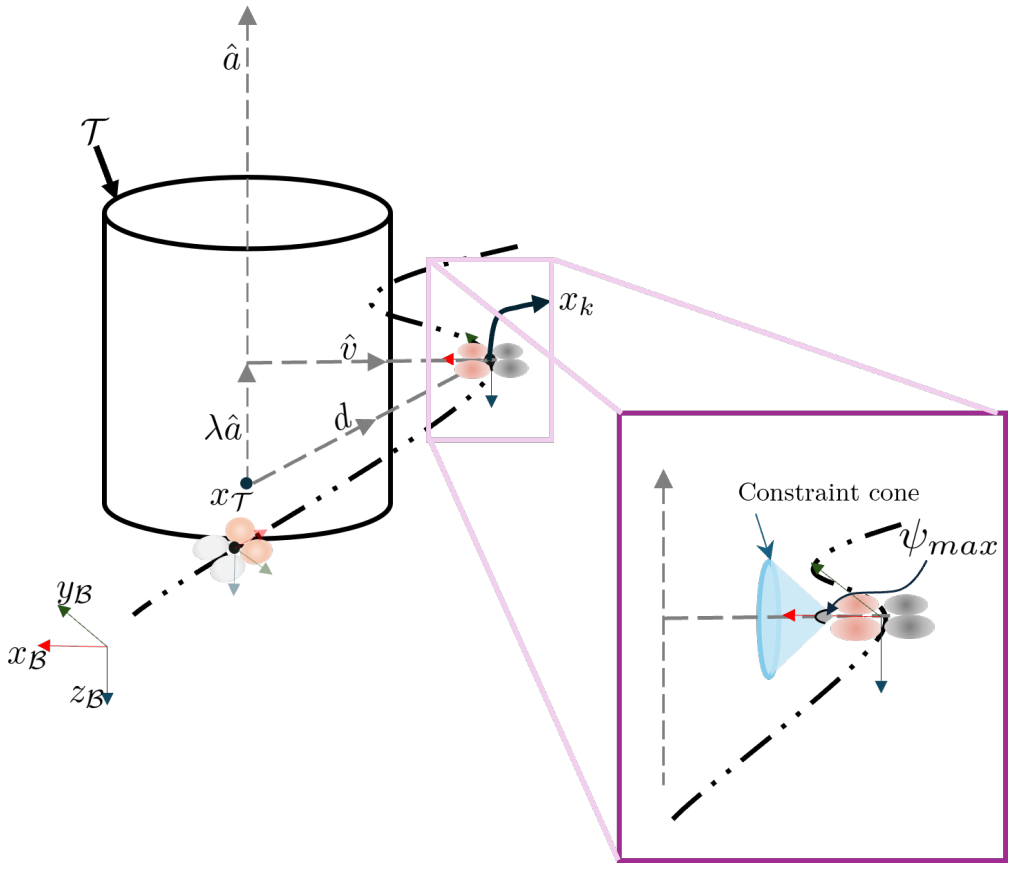


Figure 4.1: Line of Sight Constraint: $\mathcal{T}$ is the target object and ψ_{max} is the maximum allowable relaxation on LoS

such that:

$$\begin{aligned}\hat{v} &= d - \lambda\hat{a} = d - \hat{a}\hat{a}^T d \\ &= (I - \hat{a}\hat{a}^T)d.\end{aligned} \tag{4.11}$$

Let ψ_{max} be the maximum relaxation on the view direction. The View Direction constraint is given by the following equation [33, 58]:

$$c_1(r_{\mathcal{I}}, q_{\mathcal{B}/\mathcal{I}}) = \tilde{x}_{\mathcal{B}/\mathcal{I}}^T \hat{v} + ||C_{\mathcal{B}/\mathcal{I}}^T(t)\hat{v}||_2 \cos\psi_{max} \leq 0,$$

where $\tilde{x}_{\mathcal{B}/\mathcal{I}}$ defines $e_1 = [1, 0, 0]^T$ (or x axis) of Body Frame ($\mathcal{B}$) in Inertial Frame ($\mathcal{I}$).

$$\tilde{x}_{\mathcal{B}/\mathcal{I}} = C_{\mathcal{B}/\mathcal{I}}(t)e_1.$$

Thus the view direction constraint becomes:

$$c_1(r_{\mathcal{I}}, q_{\mathcal{B}/\mathcal{I}}) = (C_{\mathcal{B}/\mathcal{I}}(t)e_1)^T \hat{v} + ||C_{\mathcal{B}/\mathcal{I}}^T(t)\hat{v}||_2 \cos\psi_{max} \leq 0. \tag{4.12}$$

4.2.4 Control Constraints

1. Thrust Vector Bounds:

$$T_{min} \leq ||T_{\mathcal{B}}(t)||_2 \leq T_{max}. \tag{4.13}$$

2. Maximum Jerk Constraint:

$$\frac{d}{dt}(||T_{\mathcal{B}}(t)||_2) \leq j_{max}. \tag{4.14}$$

   This constraint ensures that the commanded thrust does not change too rapidly.

3. Torque Bounds:

$$\begin{aligned}\tau_{max} &\in \mathbb{R}^3, \\ ||M_i\tau_{\mathcal{B}}(t)||_\infty &\leq \tau_{max,i},\end{aligned} \tag{4.15}$$

where $M_i \in \mathbb{R}^{3\times3}$, $i \in \{1,2,3\}$ and

$$M_i = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}, \quad m_{jk} = \begin{cases} 1 & i = j = k \\ 0 & \text{otherwise} \end{cases}.$$

4.2.5 Boundary Conditions

1. Initial and terminal states constraints:

$$r_{\mathcal{I}}(t_0) = r_0, \quad r_{\mathcal{I}}(t_f) = r_f, \tag{4.16}$$

$$v_{\mathcal{I}}(t_0) = v_0, \quad v_{\mathcal{I}}(t_f) = v_f, \tag{4.17}$$

$$q_{\mathcal{B}/\mathcal{I}}(t_0) = q_0, \quad q_{\mathcal{B}/\mathcal{I}}(t_f) = q_f, \tag{4.18}$$

$$\omega_{\mathcal{B}}(t_0) = \omega_0, \quad \omega_{\mathcal{B}}(t_f) = \omega_f. \tag{4.19}$$

2. Initial and terminal control constraints:

$$T_{\mathcal{B}}(t_0) = u_0, \quad T_{\mathcal{B}}(t_f) = u_f, \tag{4.20}$$

$$\tau_{\mathcal{B}}(t_0) = \tau_0, \quad \tau_{\mathcal{B}}(t_f) = \tau_f. \tag{4.21}$$

4.2.6 State Triggered Constraints

State-triggered constraints (STCs) [30,46,47,56–58] are those that get activated when the vehicle is within a prescribed trigger window. A cylindrical trigger window is considered. When the vehicle enters this trigger corridor, the LoS and maximum speed are tightly constrained.

Slow-Zone State Triggered Constraint

Let $A_s \in \mathbb{R}^{3\times3}$ be the matrix that represents the slow zone trigger, then:

$$g(r) = (r - r_T)^T A_s (r - r_T) - 1 \leq 0. \tag{4.22}$$

Let $z \in \mathbb{R}^{n_z}$ represent the states that must remain in the set of continuous STCs (cSTCs), where $n_z = n_r + n_v$. Thus, the updated trigger condition is given by:

$$g_1(z) = (z - z_T)^T M_s (z - z_T) - 1 \leq 0, \quad M_s = \begin{bmatrix} A_s & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix}. \tag{4.23}$$

Inside the slow zone, the maximum velocity is restricted by imposing the following constraint:

$$c_1(z) = z^T M_c z - V_{slow}^2 \leq 0, \quad M_c = \begin{bmatrix} 0_{3\times3} & 0_{3\times3}, \\ 0_{3\times3} & I_{3\times3}, \end{bmatrix}, \tag{4.24}$$

where $V_{slow} < V_{max}$. The continuous-gradient formulation of slow zone constraint is given by:

$$h_1(z) = \min(g_1(z), 0)^2 c_1(z) \leq 0. \tag{4.25}$$

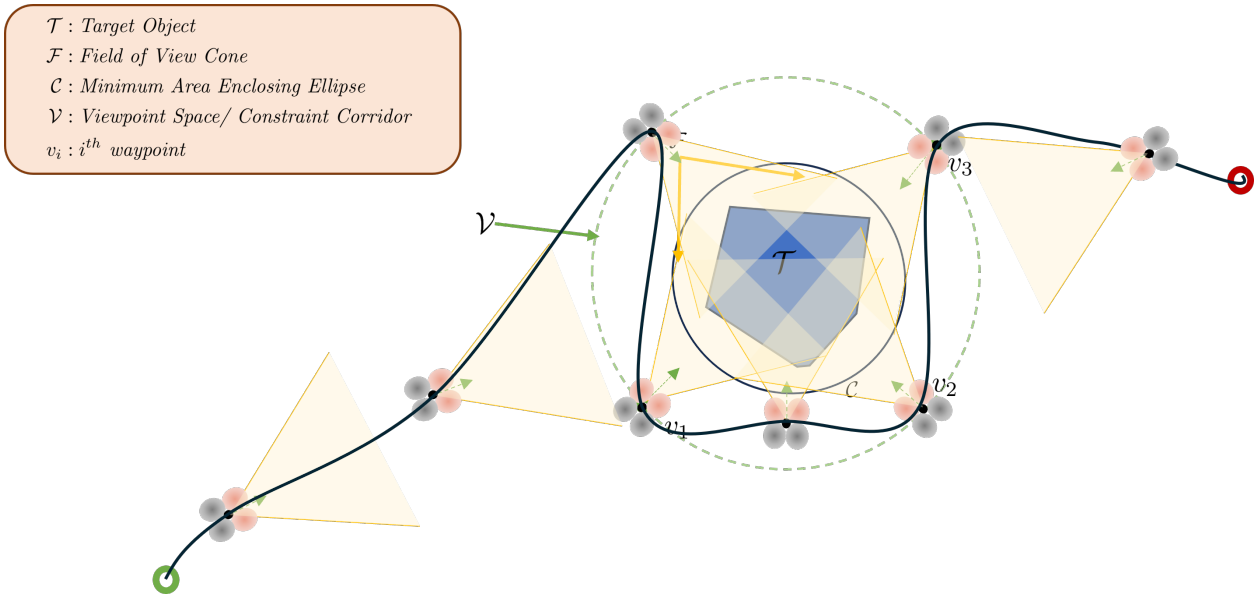
Line-of-Sight State Triggered Constraint

The trigger window given by Equation (4.22) is used to constraint the LoS constraint given by Equation (4.12). The cSTC for the LoS constraint is given by:

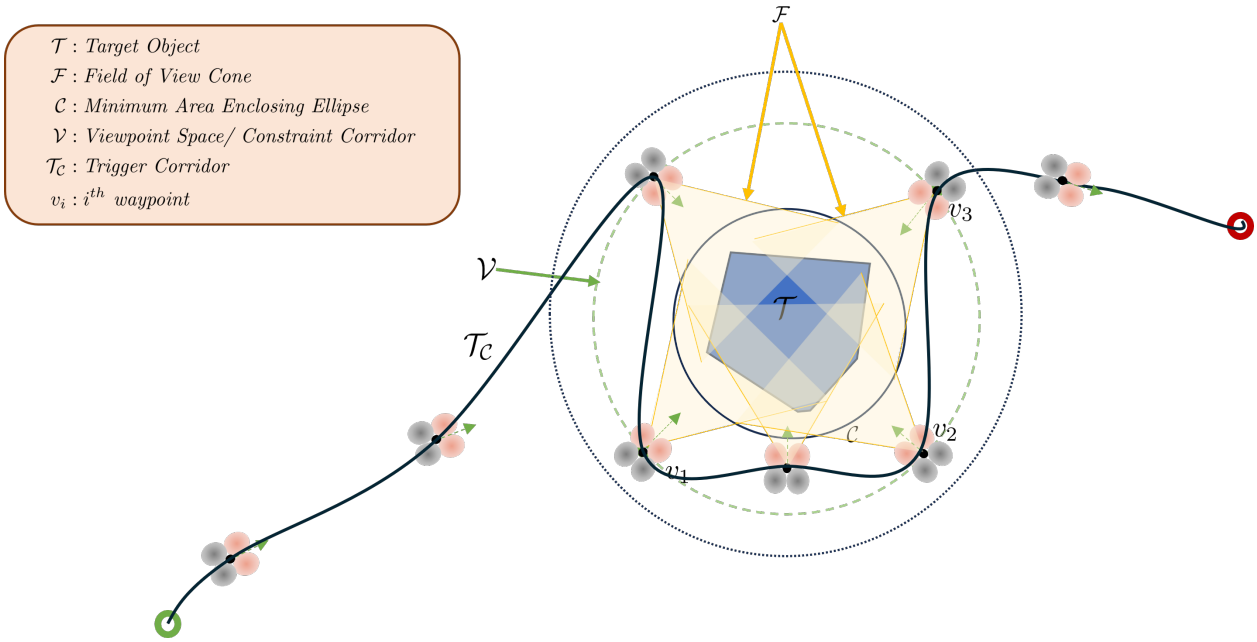
$$h_2(r, q) = \min(g_1(r), 0)^2 c_1(r, q) \leq 0. \tag{4.26}$$

4.2.7 Problem Statement

This section presents two UAV path planning scenarios. The formulation in Problem 1 imposes the LoS constraint (Equation (4.12)) over the entire time horizon along with other constraints described in the Equations (4.1) to (4.8) and Equations (4.13) to (4.21). The formulation in Problem 2 imposes the LoS constraint (Equation (4.12)) only on certain portions of the trajectory, i.e., the constraint is activated when the UAV is within a prescribed trigger window. Here, we consider a cylindrical trigger window, tightly constrained LoS, and maximum speed.



(a) Problem 1: LoS constraint imposed over the entire time horizon



(b) Problem 2: LoS constraint triggered only when UAV is within a prescribed trigger window

Figure 4.2: Problem Formulations with LoS constraint

Problem 1: Non-convex Continuous time Fixed-Final-Time optimal control problem (see Figure 4.2a)

$$\underset{u_{\mathcal{B}}(t),\zeta(t)}{\text{minimize}} \quad \int_0^{t_f} ||\mathbf{u}_{\mathcal{B}}(t)||_2^2 dt$$

subject to.

$$\begin{aligned}
\underline{\text{Boundary Conditions}} \quad & r_{\mathcal{I}}(0) = r_i, \quad r_{\mathcal{I}}(t_f) = r_f, \\
& v_{\mathcal{I}}(0) = v_{\mathcal{I}}(t_f) = 0_{3\times 1}, \\
& q_{\mathcal{B}/\mathcal{I}}(0) = q_i, \quad q_{\mathcal{B}/\mathcal{I}}(t_f) = q_f, \\
& \omega_{\mathcal{B}}(t_0) = \omega_0, \quad \omega_{\mathcal{B}}(t_f) = \omega_f, \\
& T_{\mathcal{B}}(0) = T_{\mathcal{B}}(t_f) = m g_{\mathcal{I}}, \\
& \tau_{\mathcal{B}}(0) = \tau_{\mathcal{B}}(t_f) = 0_{3\times 1}.
\end{aligned}$$

$$\underline{\text{Continuous Time Dynamics}} \quad \dot{\zeta}(t) = F(\zeta(t), u(t)), \zeta \in \mathbb{R}^{13}, u \in \mathbb{R}^6.$$

$$\begin{aligned}
\underline{\text{Non-Convex Control Constraints}} \quad & 0 < T_{min} \leq ||T_{\mathcal{B}}(t)||_2 \leq T_{max}, \\
& \frac{\Delta T_{\mathcal{B}}}{\Delta t} \leq \Delta T_{max}.
\end{aligned}$$

$$\begin{aligned}
\underline{\text{State Constraints}} \quad & ||\omega_{\mathcal{B}}||_\infty \leq \omega_{max}, \\
& ||v_{\mathcal{I}}(t)||_2 \leq v_{max}, \\
& \cos\theta_{max} + z_{\mathcal{I}}^T (q_{\mathcal{B}/\mathcal{I}} \otimes z_{\mathcal{B}} \otimes q_{\mathcal{B}/\mathcal{I}}^*) \leq 0, \\
& 1 - (r_{\mathcal{I}}(t) - r_{\mathcal{T}})^T P_i^{-1} (r_{\mathcal{I}}(t) - r_{\mathcal{T}}) \leq 0, \\
& (C_{\mathcal{B}/\mathcal{I}}(t) e_1)^T \hat{v} + ||C_{\mathcal{B}/\mathcal{I}}^T(t)\hat{v}||_2 \cos\psi_{max} \leq 0.
\end{aligned}$$

$$\underline{\text{Waypoint Constraint}} \quad \exists \quad t \in [0, t_f] \quad : r_I(t) \in \mathcal{W}_{\mathcal{P}}.$$

Problem 2: Non-convex Continuous time Fixed-Final-Time optimal control problem with state triggered line of sight constraint (see Figure 4.2b)

$$\underset{u_{\mathcal{B}}(t),\zeta(t)}{\text{minimize}} \quad \int_0^{t_f} ||\mathbf{u}_{\mathcal{B}}(t)||_2^2 dt$$

subject to.

Boundary Conditions
$$r_{\mathcal{I}}(0) = r_i, \quad r_{\mathcal{I}}(t_f) = r_f,$$
$$v_{\mathcal{I}}(0) = v_{\mathcal{I}}(t_f) = 0_{3\times 1},$$
$$q_{\mathcal{B}/\mathcal{I}}(0) = q_i, \quad q_{\mathcal{B}/\mathcal{I}}(t_f) = q_f,$$
$$\omega_{\mathcal{B}}(t_0) = \omega_0, \quad \omega_{\mathcal{B}}(t_f) = \omega_f,$$
$$T_{\mathcal{B}}(0) = T_{\mathcal{B}}(t_f) = mg_{\mathcal{I}},$$
$$\tau_{\mathcal{B}}(0) = \tau_{\mathcal{B}}(t_f) = 0_{3\times 1}.$$

Continuous Time Dynamics
$$\dot{\zeta}(t) = F(\zeta(t), u(t)), \quad \zeta \in \mathbb{R}^{13}, u \in \mathbb{R}^6.$$

Non-Convex Control Constraints
$$0 < T_{min} \leq ||T_{\mathcal{B}}(t)||_2 \leq T_{max},$$
$$\frac{\Delta T_{\mathcal{B}}}{\Delta t} \leq \Delta T_{max}.$$

State Constraints
$$||\omega_{\mathcal{B}}||_\infty \leq \omega_{max},$$
$$||v_{\mathcal{I}}(t)||_2 \leq v_{max},$$
$$\cos\theta_{max} + z_{\mathcal{I}}^T(q \otimes z_{\mathcal{B}} \otimes q^*) \leq 0,$$
$$1 - (r(t) - r_t)^T P_i^{-1}(r(t) - r_t) \leq 0,$$

Waypoint Constraint
$$\exists \quad t \in [0, t_f] \quad : r_I(t) \in \mathcal{V}.$$

continuous State Triggered Constraints
$$h_1(z) \leq 0, \quad z = [r, v],$$
$$h_2(\tilde{z}) \leq 0, \quad \tilde{z} = [r, q].$$

4.3 Convex Formulation

This section discusses the convex formulation for Problem 1 and Problem 2 by applying modifications to convert a continuous-time non-convex fixed-final-time problem into a convex fixed-final-time SOCP subproblem. The non-linear dynamics over the entire time-horizon, governing the evolution of the state $x(t) \in \mathbb{R}^{13}$ with control input $u(t) \in \mathbb{R}^6$, are given by:

$$\dot{\zeta}(t) = F(t, \zeta(t), u(t)), \quad t \in [0, t_f). \tag{4.29}$$

This section is organized as follows: Section 4.3.1 details the reformulation of the path constraint for continuous-time satisfaction. Section 4.3.2 defines an affine map that normalizes time. Section 4.3.3 details the convexification process of non-linear dynamics and non-convex constraints. Section 4.3.4 details the conversion of an infinite-dimensional optimal control problem to a finite-dimensional problem.

4.3.1 Continuous Time Constraint Satisfaction

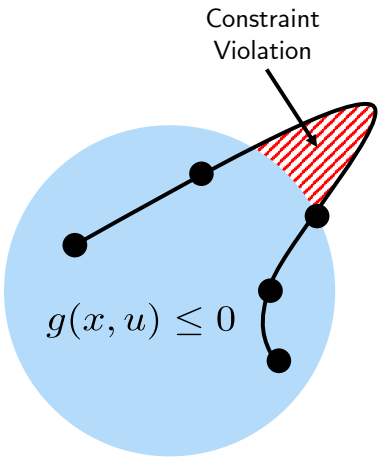


Figure 4.3: Continuous Constraint Satisfaction [19]

Successive convexification takes a *discretize-then-optimize* approach to solve general optimal control problems. As discretization is an important step in this method, the resulting

solutions suffer from inter-sample constraint violations [18] (see Figure 4.3). To address this, the path constraints are reformulated and combined with multiple-shooting discretization [6, 31] by augmenting path constraints to the system's dynamics in order to ensure continuous-time constraint satisfaction.

Let

$$g(t, \zeta(t), u(t)) \leq 0. \tag{4.30}$$

$$h(t, \zeta(t), u(t)) = 0, \tag{4.31}$$

represent inequality and equality path constraints, which are continuously differentiable. We define a new variable as follows:

$$y(t) = \int_0^t \{\max\{0, g(t, \zeta(t), u(t))\}^2 + h(t, \zeta(t), u(t))^2\} dt, \tag{4.32}$$

$$\dot{y}(t) = \max\{0, g(t, \zeta(t), u(t))\}^2 + h(t, \zeta(t), u(t))^2, \tag{4.33}$$

which can be used to augment the non-linear dynamics in Equation (4.29) as follows:

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} \dot{\zeta}(t) \\ \dot{y}(t) \end{bmatrix}, \\ &= \begin{bmatrix} F(t, \zeta(t), u(t)) \\ \max\{0, g(t, \zeta(t), u(t))\}^2 + h(t, \zeta(t), u(t))^2 \end{bmatrix}, \\ &= f(t, x(t), u(t)). \end{aligned} \tag{4.34}$$

4.3.2 Time Interval Dilation

The dynamics are evaluated in the sub-interval $[t_k, t_{k+1})$, where $0 < t_k < t_{k+1} < t_N = t_f^-$ for all $k \subseteq 1 : N - 1$. We define an invertible map $\tau_k(t) : [t_k, t_{k+1}) \to [0, 1)$ given by:

$$\tau_k(t) \coloneqq \frac{t - t_k}{t_{k+1} - t_k}. \tag{4.35}$$

This mapping is known as *time-interval-dilation*, as it normalizes the time interval to $[0, 1)$ by either shrinking or expanding–and hence *dilating*–the original time-interval [31, 41]. Thus

the non-linear dynamics in Equation (4.34) in terms of the dilated time τ_k are given by

$$\begin{aligned} \overset{\circ}{x}(t) = \frac{d}{d\tau_k}x(t) = \frac{dt}{d\tau_k}\frac{d}{dt}x(t) = \underbrace{(t_{k+1}^{-} - t_k)}_{s_k}\dot{x}(t), \\ = s_k f(t, x(t), u(t)) \coloneqq F_s(t, x(t), u(t), s_k), \end{aligned} \tag{4.36}$$

where $s_k \in \mathbb{R}_+$ is the *dilation factor*. At each discrete node on the time grid, the corresponding state and control are denoted as (x_k, u_k), for all $k \subseteq 1 : N-1$.

4.3.3 Linearization

Dynamics

The nonlinear system defined by Equation (4.36) is linearized about an arbitrary reference trajectory $(\bar{x}, \bar{u}, \bar{s})$, which yields a linear time-varying (LTV) system, given by Equation (4.37):

$$\overset{\circ}{x}(\tau_k) \approx A(\tau_k)x(\tau_k) + B(\tau_k)u(\tau_k) + S(\tau_k)s(\tau_k) + d(\tau_k), \tag{4.37}$$

where:

$$A(\tau_k) \coloneqq \nabla_x F_s(\tau_k, \bar{x}(\tau_k), \bar{u}(\tau_k), \bar{s}(\tau_k)), \tag{4.37a}$$

$$B(\tau_k) \coloneqq \nabla_u F_s(\tau_k, \bar{x}(\tau_k), \bar{u}(\tau_k), \bar{s}(\tau_k)), \tag{4.37b}$$

$$S(\tau_k) \coloneqq \nabla_s F_s(\tau_k, \bar{x}(\tau_k), \bar{u}(\tau_k), \bar{s}(\tau_k)), \tag{4.37c}$$

$$d(\tau_k) \coloneqq F_s(\tau_k, \bar{x}(\tau_k), \bar{u}(\tau_k), \bar{s}(\tau_k)) - A(\tau_k)\bar{x}(\tau_k) - B(\tau_k)\bar{u}(\tau_k) - S(\tau_k)\bar{s}(\tau_k). \tag{4.37d}$$

Non-Convex State and Control Constraints

Assuming that the functions defining non-convex state and control constraints are at least once differentiable almost everywhere we approximate every non-convex constraint using the operator $T(\cdot)$, which is the first-order Taylor series approximation given a non-convex function $\xi(\cdot)$ linearized around $\bar{z}$:

$$T(\xi(\cdot), z, \bar{z}) \approx \xi(\bar{z}) + \left.\frac{\delta \xi}{\delta z}\right|_{\bar{z}} (z - \bar{z}). \tag{4.38}$$

4.3.4 Discretization

Discretization converts the linear-time-varying *continuous-time* problem into a linear-time-varying *discrete-time* parameter optimization problem. The entire time horizon is distributed into K evenly spaced nodes with $K-1$ subintervals, each temporal node associated with an index $k \in \{1, 2, \ldots, K\}$ and corresponding normalized time τ_k defined in Equation (4.35).

First-order hold parameterization is done on the control input signal at discrete temporal nodes, and the continuous-time control input signal is obtained by linearly interpolating between the discrete values at successive nodes. Thus over time normalized interval $\tau_k \in [0, 1)$ corresponding to $t \in [t_k, t_{k+1})$, the control profile is parameterized as

$$u(\tau_k) = (1 - \tau_k)u_k + \tau_k u_{k+1}, \quad k \in \{1, \ldots, N-1\}. \tag{4.39}$$

The LTV dynamics in Equation (4.37) is then re-written using the control parameterization in Equation (4.39) and using deviations from the reference, i.e., $(\bar{x}, \bar{u}, \bar{s})$ as shown in Equation (4.40). Henceforth, "=" is used instead of "≈" for simplicity, as Equation (4.40) is a first-order approximation of the original non-linear dynamics:

$$\begin{aligned} \Delta\dot{\overset{\circ}{x}}(\tau_k) = A(\tau_k)\Delta x(\tau_k) + B(\tau_k)(1-\tau_k)\Delta u(\tau_k) \\ + B(\tau_k)\tau_k \Delta u(\tau_{k+1}) + S(\tau_k)\Delta s(\tau_k). \end{aligned} \tag{4.40}$$

Equation (4.40) has a unique solution [37,38], given by Equation (4.41), $\forall t \in [t_k, t_{k+1}), \tau_k \in [0, 1)$:

$$\begin{aligned} \Delta x(\tau_k) = \Phi(\tau_k, 0)\Delta x(0) + \int_0^{\tau_k} \Phi(\tau_k, \xi) \\ \{B(\xi)(1-\tau_k)\Delta u_k + B(\xi)\tau_k \Delta u_{k+1} + S(\xi)\Delta s_k\} d\xi, \end{aligned} \tag{4.41}$$

Equation (4.41) when evaluated at $\tau_k = 1^-$ gives Equation (4.42)

$$\Delta x(1^-) = A_k \Delta x(0) + B_k^- \Delta u_k + B_k^+ \Delta u_{k+1} + S_k \Delta s_k. \tag{4.42}$$

Here,

$$A_k = I_{nx} + \lim_{z \to \tau_{k+1}^-} \int_{\tau_k}^{z} A(\xi)\Psi_A(\xi)d\xi, \tag{4.43a}$$

$$B_k^- = \lim_{z \to \tau_{k+1}^-} \int_{\tau_k}^{z} \{A(\xi)\Psi_{B^-}(\xi) + B(\xi)(1-\xi)\}d\xi, \tag{4.43b}$$

$$B_k^+ = \lim_{z \to \tau_{k+1}^-} \int_{\tau_k}^{z} \{A(\xi)\Psi_{B^+}(\xi) + B(\xi)\xi)\}d\xi, \tag{4.43c}$$

$$S_k = \lim_{z \to \tau_{k+1}^-} \int_{\tau_k}^{z} \{A(\xi)\Psi_S(\xi) + S(\xi)\}d\xi, \tag{4.43d}$$

which are solutions to the initial value problems given by Equation (4.44) :

$$\overset{\circ}{\Psi}_A(\xi) = A(\xi)\Psi_A(\xi), \tag{4.44a}$$

$$\overset{\circ}{\Psi}_{B^-}(\xi) = A(\xi)\Psi_{B^-}(\xi) + B(\xi)(1-\xi), \tag{4.44b}$$

$$\overset{\circ}{\Psi}_{B^+}(\xi) = A(\xi)\Psi_{B^+}(\xi) + B(\xi)\xi, \tag{4.44c}$$

$$\overset{\circ}{\Psi}_S(\xi) = A(\xi)\Psi_S(\xi) + S(\xi). \tag{4.44d}$$

The discretized dynamics in terms of absolute variables are given by Equation (4.45b)

$$x_{k+1} = A_k(x_k - \bar{x}_k) + B_k^-(u_k - \bar{u}_k) + B_k^+(u_{k+1} - \bar{u}_{k+1}) + S_k(s_k - \bar{s}_k) + x_{k+1}^{prop}, \tag{4.45a}$$

$$x_{k+1} = A_k x_k + B_k^- u_k + B_k^+ u_{k+1} + S_k s_k + d_k, \tag{4.45b}$$

where:

$$x_{k+1}^{prop} = \bar{x}_k + \lim_{z \to \tau_{k+1}^-} \int_{\tau_k}^{z} F(\xi, \bar{x}(\xi), \bar{u}(\xi), \bar{s})d\xi,$$

$$d_k = x_{k+1}^{prop} - (A_k\bar{x}_k + B_k^-\bar{u}_k + B_k^+\bar{u}_{k+1} + S_k\bar{s}_k).$$

4.4 Successive Convexification

This section outlines the successive convexification process, which approximately solves a non-convex optimization problem by iteratively solving related convex sub-problems. The i^{th} iteration is denoted by superscript i. Two important modifications to the subproblem are introduced: (1) trust regions and (2) virtual buffer and controls.

4.4.1 Trust Region

The trust region concerns the distance between the solution of the subproblem and the reference about which the problem is linearized to create the subproblem. Linear approximations are accurate only in a neighborhood around the reference; thus, the subproblem solution must be kept close to the linearization point defined by the reference [37]. Another reason for having a trust region is to not deviate too far from the reference, as in certain cases, linearization can render the solution unbounded below (phenomenon also known as *artificial unboundedness*). To avoid this, a soft quadratic trust region is added to the subproblem cost function:

$$J_{TR}(x,u,s) \triangleq \sum_{k=1}^{N} \left[\left(\begin{bmatrix} x_k - \bar{x}_k \\ u_k - \bar{u}_k \end{bmatrix} \right)^T W_{TR} \left(\begin{bmatrix} x_k - \bar{x}_k \\ u_k - \bar{u}_k \end{bmatrix} \right) + s_k^2 * W_s \right], \tag{4.46}$$

where $W_{TR} \in \mathbb{S}_{++}^{m \times m}$ is a positive definite matrix where $m = n_x + n_u$ and $W_s > 0$. This penalizes the optimizer's distance from the reference trajectory, ensuring the solution doesn't venture too far.

4.4.2 Artificial Infeasibility

Linear approximations and a poor initial guess may cause the subproblem to become infeasible even if a feasible solution to the original problem exists exists. One way to mitigate this is to add slack variables to the linearized non-convex constraints but penalize them heavily so that the subproblem is feasible in every iteration. When penalized properly, these slack variables go to zero at convergence for the solution to be feasible to the original problem. The slack variable is usually referred *virtual control* (Equation (4.47)) when added to linearized

dynamics, and *virtual buffer* (Equation (4.49)) when added to linearized constraints [37]:

$$x_{k+1} = A_k x_k + B_k^- u_k + B_k^+ u_{k+1} + S_k s_k + d_k + \upsilon_{c,k}, \tag{4.47}$$

$$J_{vc} = w_{vc} \left[\sum_{k=1}^{N-1} ||\upsilon_{c,k}||_1 \right], \tag{4.48}$$

$$h(\bar{x}(\tau_k)) + \left. \frac{\delta h}{\delta x} \right|_{\bar{x}(\tau_k)} \delta x(\tau_k)) + \upsilon_{b,k} \leq 0, \tag{4.49}$$

$$J_{vb} = w_{vb} \left[\sum_{k=1}^{N} ||\upsilon_{b,k}|| \right]. \tag{4.50}$$

Problem 3: Discrete-time Free-Final-Time optimal control problem

$$\underset{u_{[1:N]}, s_{[1:N]}}{\text{minimize}} \quad w_f \left[\sum_{1}^{N} ||\mathbf{u}_k||_2^2 \right] + J_{TR} + J_{vc},$$

subject to.

Boundary Conditions $r_1 = r_i, \quad r_N = r_f,$

$v_1 = v_N = 0_{3\times 1},$

$q_1 = q_i, \quad q_N = q_f,$

$\omega_1 = \omega_i, \quad \omega_N = \omega_f,$

$T_1 = T_N = m g_{\mathcal{I}},$

$\tau_1 = \tau_N = 0_{3\times 1}.$

Dynamics $x_{k+1} = A_k x_k + B_k^- u_k + B_k^+ u_{k+1} + S_k s_k + d_k + \upsilon_{c,k}.$

Waypoint Constraint $\exists \quad k \in [1, N] \quad : r_k \in \mathcal{W}_{\mathcal{P}}.$

Problem 4: Discrete-time Free-Final-Time optimal control problem with state triggered line of sight constraint

$$\underset{u_{[1:N]},s_{[1:N]}}{\text{minimize}} \quad w_f \left[\sum_{1}^{N} ||\mathbf{u}_k||_2^2\right] + J_{TR} + J_{vc},$$

subject to.

Boundary Conditions $r_1 = r_i, \quad r_N = r_f,$

$$v_1 = v_N = 0_{3\times 1},$$

$$q_1 = q_i, \quad q_N = q_f,$$

$$\omega_1 = \omega_i, \quad \omega_N = \omega_f,$$

$$T_1 = T_N = mg_{\mathcal{I}},$$

$$\tau_1 = \tau_N = 0_{3\times 1}.$$

Dynamics $x_{k+1} = A_k x_k + B_k^- u_k + B_k^+ u_{k+1} + S_k s_k + d_k + v_{c,k}.$

Waypoint Constraint $\exists \quad k \in [1, N] \quad : r_k \in \mathcal{W}_{\mathcal{P}}.$

Chapter 5

MODEL-BASED TRAJECTORY PLANNING FOR 3D SCENE RECONSTRUCTION

5.1 Introduction

In this chapter, we use the methods described in the previous chapters to generate a dynamically feasible trajectory for continuous image acquisition for dense scene reconstruction. This chapter is structured as follows: Section 5.2 covers existing methods being used for 3D scene reconstruction or inspection. Section 5.3 details the approach described in previous chapters in the context of 3D scene reconstruction. Section 5.4 concludes by discussing the results of our approach.

5.2 Related Work

5.2.1 Off-the-Shelf Flight Planners

The most common practice for UAV-based data-capturing tasks is by using off-the-shelf flight planners like Pix4D capture[1], DJI Terra[2], DJI flight planner[3], UgCS[4], DroneDeploy[5], PixHawk Ardu Mission Planner[6], Precisionhawk[7], QGroundControl[8]. The flight planners offer very limited parameter settings and generate conservative trajectories like an orbit, a zigzag,

[1]https://www.pix4d.com/product/pix4dcapture

[2]https://www. dji.com/de/dji-terra

[3]https://www. djiflightplanner.com/

[4]https://www.ugcs.com/

[5]https://www.dronedeploy.com/product/mobile/

[6]https://ardupilot.org/planner/index.html

[7]https://www.precisionhawk.com/

[8]http://qgroundcontrol.com/

or a lawnmower pattern to cover the scene, regardless of the scene geometry and structure. Hence, they typically oversample some regions while undersampling others, especially if there are complex objects with self-occluding structures. Also, most of them use the trajectory without structure from motion (SfM) and Multiview Stereo (MVS) constraint, crucial for 3D scene reconstruction from images acquired from monocular camera [35]. Agisoft recently developed *Metashape*, an optimum path planning tool for image capturing using a rough 3D scene proxy [1, 35]. It generates a set of camera positions with sufficient overlap around the object to cover the entire object. However, it does not guarantee complete coverage in real-world situations [35].

5.2.2 Mode-Based Methods

Model-based methods leverage prior information about the environment, i.e., a 3D scene proxy, to generate a set of optimal viewpoints and trajectories. Agisoft's *Meatshape* uses a rough 3D scene proxy for mission planning [1,35]. Although the camera network is generated with sufficient overlap, it does not guarantee complete coverage.

Recently, *Explore-then-Exploit* (ETE) approach has become popular for planning UAV-based data capturing. This approach consists of two phases. The first is the exploratory phase, which is used to get a geometric proxy by using planning techniques similar to off-the-shelf planners. The second phase is the exploitation phase, where the geometric proxy is leveraged to design the optimum viewpoints and trajectories to acquire images for 3D reconstruction.

This thesis focuses on the second phase of mission planning, i.e., the *exploitation* phase, as mentioned in Chapter 1. Hoppe et al. [28] used a generate-and-test approach, where they first generated a set of candidate camera poses by assigning a fronto-parallel view to each triangle of the proxy-mesh. Then, they create camera clusters relative to their intersection angles given the center of the triangular mesh. They then generate angular bins with a width of 10° and discard cameras with an angle greater than 40°. The camera score is then evaluated by the number of histograms a camera is visible in, i.e., if a camera is in many

bins, that camera is crucial for overall reconstruction as these cameras cover many points on the object's surface. One downside of this method is the camera position initialization. They sample candidate cameras for each triangular mesh in the proxy mesh, which could result in dense search space for bigger-scale structures.

Roberts et al. [48] proposed a unified global optimization problem for viewpoint selection and trajectory planning, modeled as a submodular orienteering problem. This method finds a path that maximizes scene coverage while satisfying SfM and MVS heuristics within the UAV's limited travel budget. Once the coverage model has been created, approximated optimal orientations of all candidate cameras are computed, and the submodular orienteering is then converted to an additive orienteering problem, which is solved as an integer linear problem. However, the method has limitations, particularly the monotonicity of the coverage model, i.e., selecting more cameras never reduces the coverage score. Moreover, they use the travel budget as the stopping criterion, which does not take into account the complexity of the object.

Hepp et al. [27] developed a system for 3D reconstruction of building-scale scenes, which uses a coarse scene proxy to generate a volumetric occupancy map, categorizing the voxels as occupied, free-space, and unobserved. Each voxel is also assigned a measure of observation quality. They used an approximate camera model to calculate the expected information gain from each viewpoint independently without considering MVS heuristics, thus encouraging fronto-parallel views. The submodular optimization with travel budget constraint is solved using a recursive strategy introduced in [9]. This strategy splits the trajectory and travel budget into two parts, selecting the reachable viewpoint with maximum information gain.

All the methods discussed above share a common focus: finding a set of "good" locations for capturing images, which then serve as waypoints for computing the trajectory for the UAV during the path-planning step. As shown in Table 5.1, the path is typically obtained using heuristic solutions, such as formulating it as a traveling salesman problem (TSP) to ensure that all the selected waypoints are visited. The resulting trajectory for image capture is generally piece-wise linear, consisting of straight lines connecting waypoints. However, these

Exploration		Orbit Flight, Nadir Flight, Oblique Images, 2D map and estimated building height
Exploitation	**Position**	Uniform Sampling on Primitive shapes, Random Sampling
	Orientation	Fronto-parallel to normal, Regular pitch & yaw sampling, Random sampling with a bias towards the region of interest (key points)
	Pose Computation	SFM & MVS heuristics, Information Gain Maximization, Genetic Algorithm, Max-Min Optimization, ILP
Path Planning		RRT*, TSP, A*, Dijkstra

Table 5.1: Characteristics of Model-based approach [35]

paths often have sharp turns, leading to increased battery consumption due to the frequent acceleration and deceleration required to track the path. Zhang et al. [64] addressed this issue by formulating viewpoint selection and path planning as a joint optimization problem. Their formulation aims to (1) maximize information gain over the entire trajectory, (2) maximize information gain per unit length of the trajectory, and (3) minimize total turning angles along the trajectory to avoid sharp turns, thereby improving path quality. Although the resulting trajectory is smooth, rewiring the search tree after a fixed number of iterations is required to get the best path in the search tree, which is a computationally expensive process. Furthermore, image acquisition is continuous in the sense that there are viewpoints in between the waypoints.

This thesis presents a new application of SCP-based methods for viewpoint and trajectory

planning for continuous image acquisition. Compared to the above works, the formulation is different. Even though, at the core, both the problems are handled separately as done in [27, 28], the selected viewpoints are treated as waypoints when planning dynamically feasible trajectory, where LoS constraint 4.12 has to be satisfied in continuous time [19], i.e., not only at the waypoints but also between them, thus resulting in a smooth and dynamically feasible trajectory.

5.3 Methodology

In this section, the two-step optimization framework is employed for 3D scene reconstruction via continuous image acquisition. Here, finding a "good" set of viewpoints is treated as one problem (as covered in Chapter 3), and generating a dynamically feasible trajectory through those viewpoints is treated as waypoints, is addressed as a separate problem (as covered in Chapter 4). For both problems, SCP-based methods are used, which provide convergence guarantees [38]. Figure 5.1 gives an overview of the proposed algorithm.

This section is structured as follows: Section 5.3.1 gives a brief overview of processing done on scene proxy. Section 5.3.2 gives a brief on workflow for viewpoint optimization. Section 5.3.3 gives a brief about the trajectory optimization framework.

5.3.1 Point Cloud Processing

Given a 3D scene proxy, point cloud processing operations are done to identify clusters and generate convex hulls, which are used as keep-out zones for the trajectory planning framework.

5.3.2 Viewpoint Optimization

After the cluster of interest has been identified, the first task is to sample candidate viewpoints on some primitive shape (cylinder, cube, etc.). Discretize the environment. Here, octree representation is used to simplify the environment. Then, the viewability vector is

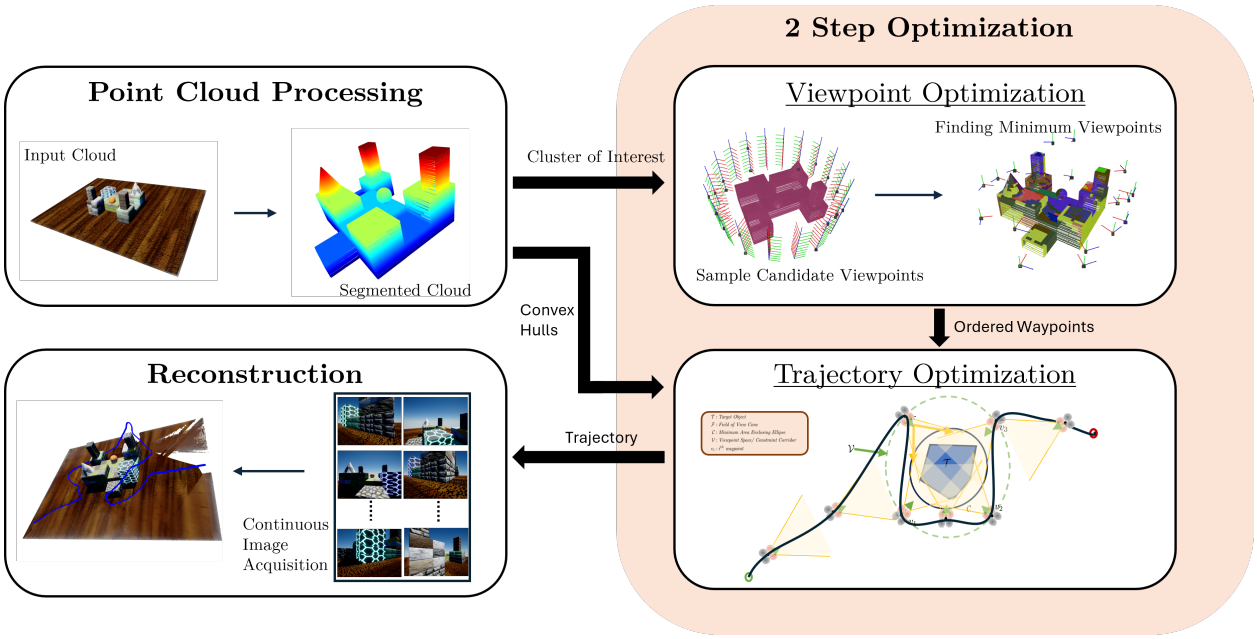


Figure 5.1: Algorithm Overview: The algorithm takes a 3D geometry proxy to generate a set of optimum viewpoints to maximize the visibility of the object. These viewpoints then serve as waypoints for trajectory optimization

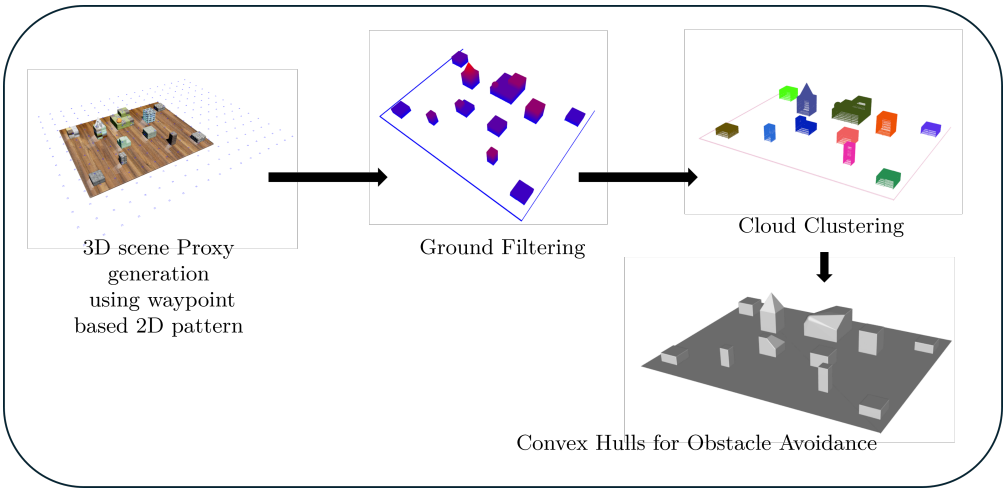


Figure 5.2: Point cloud processing workflow

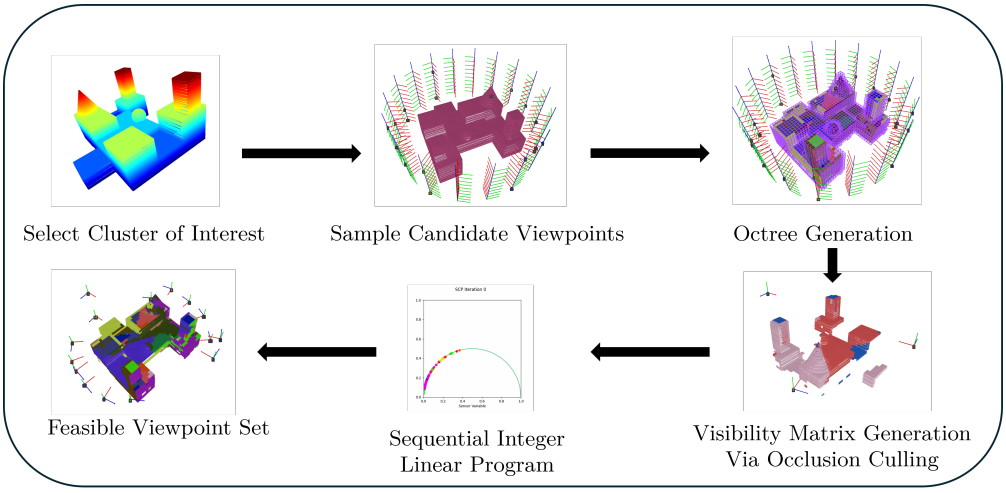


Figure 5.3: Viewpoint optimization workflow

evaluated for all the candidate viewpoints and stacked together to get the viewability matrix. Given matrix A, viewpoint set $\mathcal{V}$, the set-cover problem is formulated and solved via SIP (see **SIP Formulation**).

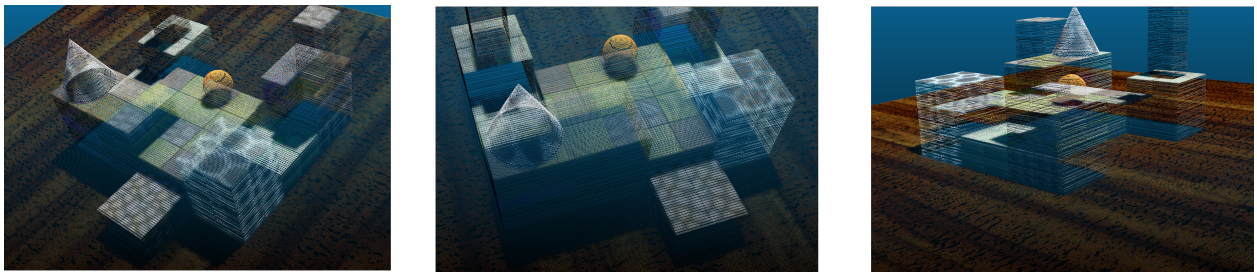
5.3.3 Trajectory Optimization

Once a "good" set of viewpoints is available, these are used as waypoints to be visited by the UAV. TSP formulation is leveraged to order these waypoints with respect to an initial and final position. The ordered waypoint set is then used to generate a dynamically feasible trajectory with LoS constraint (see Problem 3 and Problem 4).

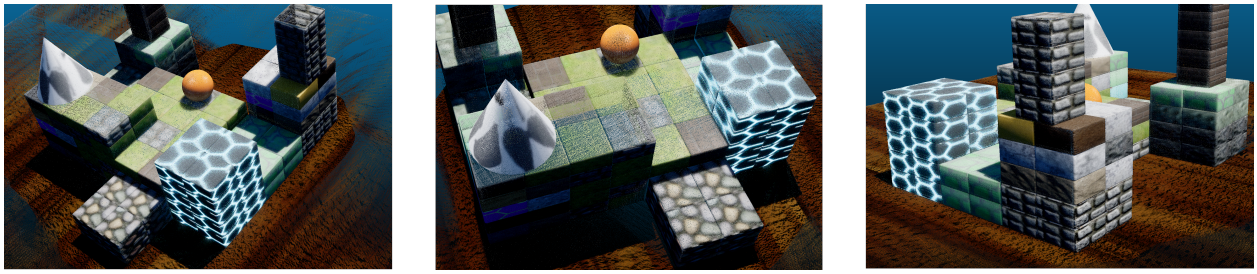
5.4 Results

The proposed framework was tested on Airsim [53]. The data collection and 3D reconstruction were done using [3]. Discrete formulation of Problem 1 and Problem 2 (Problem 3 and Problem 4 respectively) were tested, and a comparison is shown in Table 5.2.

5.4.1 Reconstruction Quality



(a) Point Cloud Generated using Nadir Flight with lawnmower pattern



(b) Point Cloud Generated using Proposed Method with state triggered Line of Sight Constraint

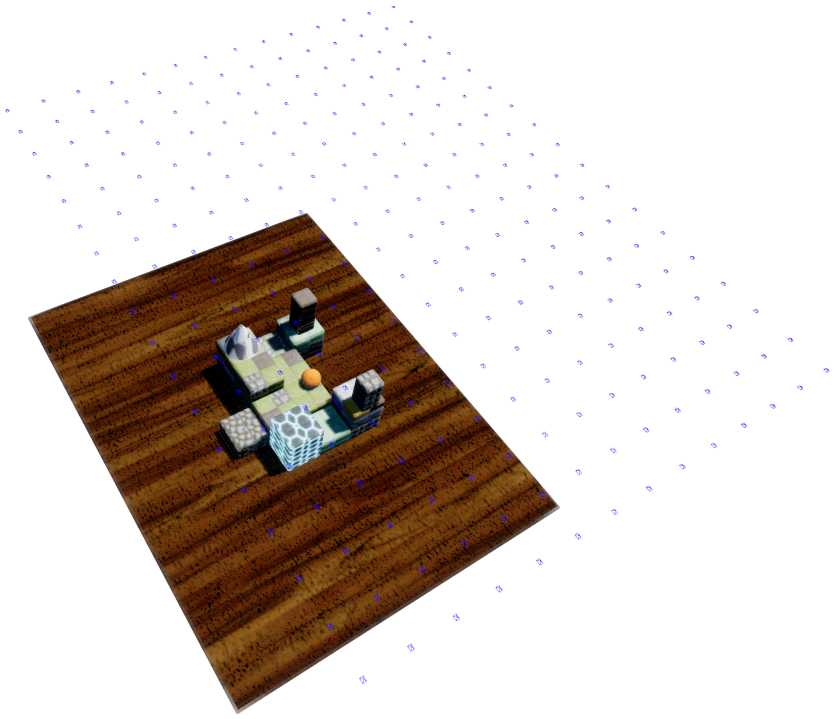
Figure 5.4: 3D Scene Reconstruction (visualization done using [11])

5.4.2 Path Quality

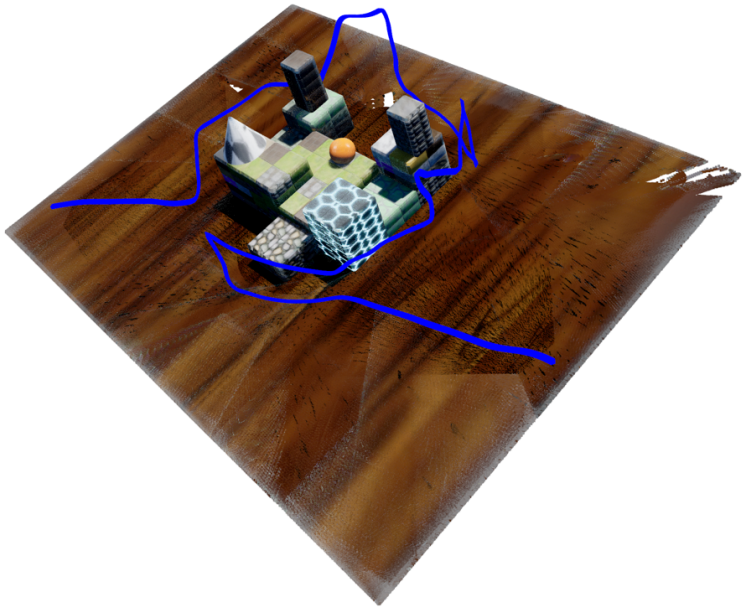
Problem Type	Time (s)	Distance (m)
Formulation 1	478.94	729.25
Formulation 2	523.56	733.195

Table 5.2: Flight time and distance comparison for the two formulations

Figure 5.5 shows the 3D scene reconstructions produced from images captured using Nadir flight with lawnmower pattern 5.5a and the proposed method 5.5b. As seen in Table 5.2, the trajectory generated by Problem 1 takes less time and travels less distance than



(a) Point Cloud Generated using Nadir Flight with lawnmower pattern



(b) Point Cloud Generated using proposed method

Figure 5.5: 3D Scene Reconstruction (visualization done using [68])

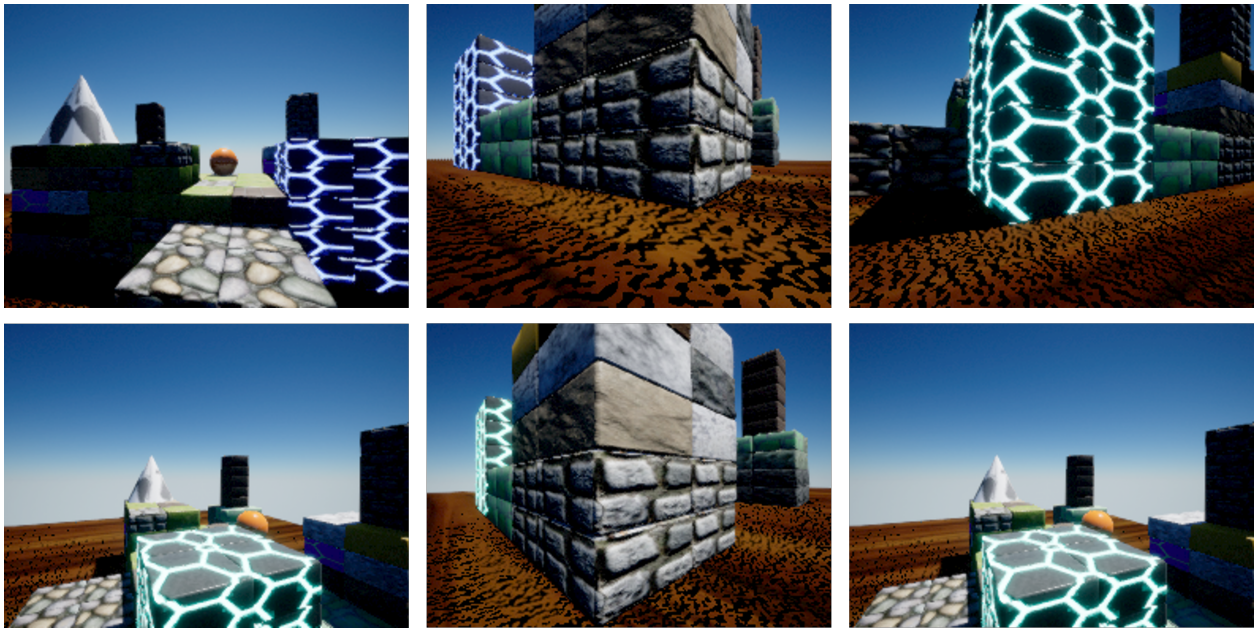
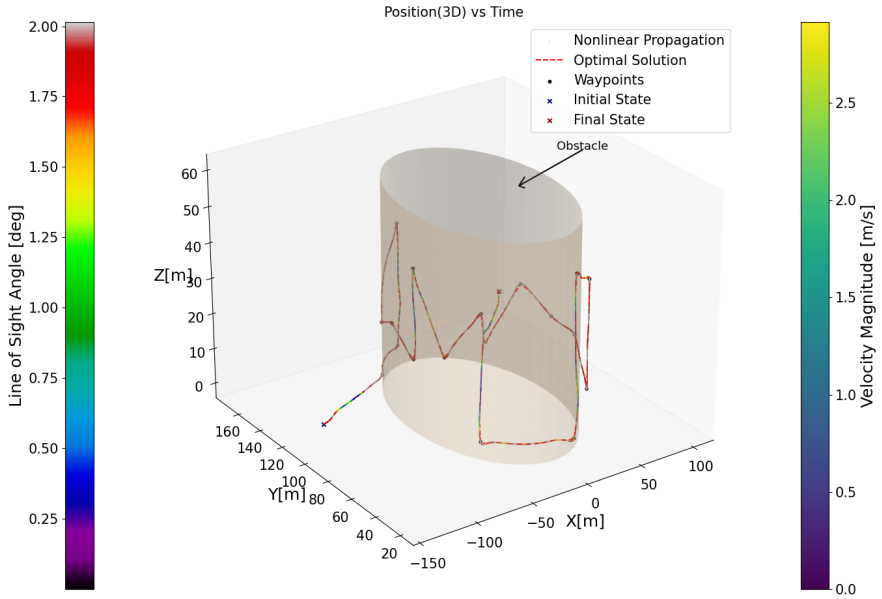
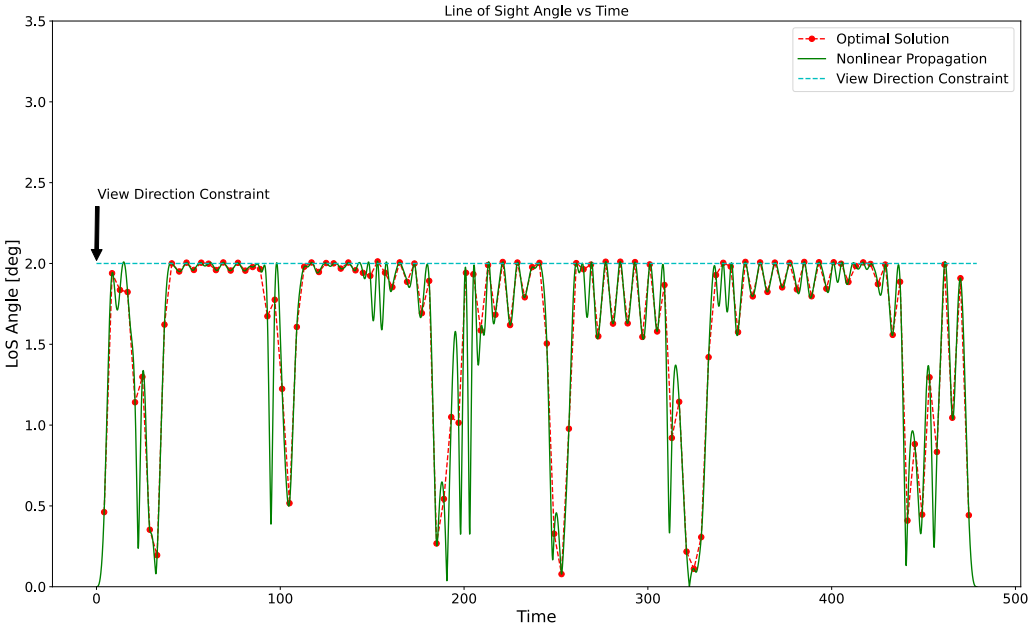


Figure 5.6: Continuous Image Acquisition using the proposed framework

Problem 2. Also, the LoS constraint in Problem 1 is satisfied over the entire horizon (see Figure 5.7). In contrast, Problem 2 exhibits some violations of LoS constraint when the UAV is within the prescribed trigger corridor (see Figure 5.8).

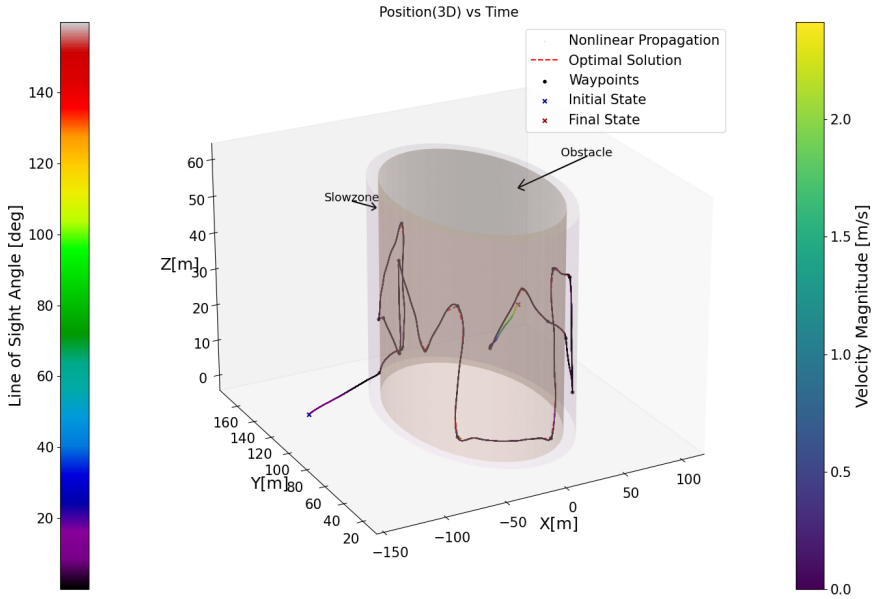


(a) 3D Position Plot

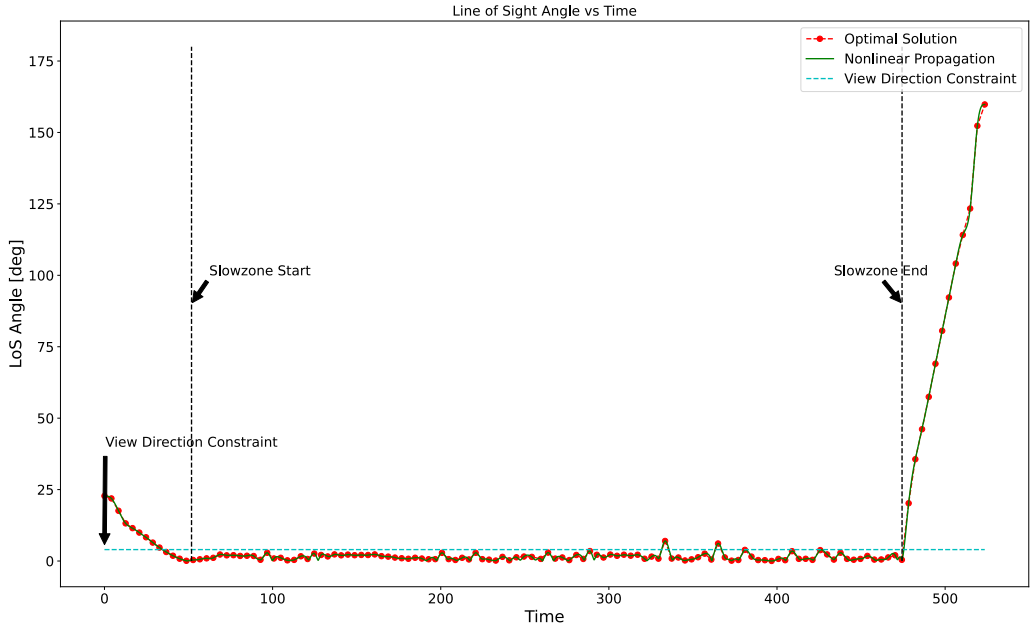


(b) LoS constraint satisfaction plot

Figure 5.7: Problem 3: line of sight constraint over the entire horizon



(a) 3D Position Plot



(b) LoS constraint satisfaction plot

Figure 5.8: Problem 4: State-triggered line of sight constraint

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Summary of Contributions

This thesis presented a comprehensive approach to model-based trajectory planning for 3D scene reconstruction, focusing on optimizing viewpoint selection and trajectory generation separately. The approach integrates key concepts from point cloud processing, view planning, and Sequential Convex Programming (SCP) to solve the non-convex optimization problem.

The thesis starts by exploring the context in which UAVs are employed in applications like search and rescue, surveillance, inspection, and scene reconstruction. The success of these tasks relies on how effectively trajectory planning and viewpoint selection are done. Following this, a framework for generating candidate viewpoints based on geometric primitives such as spheres, cylinders, and other shapes provides flexible coverage around a target object. Techniques like frustum culling and occlusion culling filters are used to evaluate viewpoints.

The first contribution of this work is reformulating the integer linear program problem for view planning to solve it using the SCP framework. The proposed framework captures crucial elements like visibility coverage and can handle bigger problems. Experimental results demonstrated the framework's effectiveness in achieving optimal or locally optimal viewpoint configuration while managing the complexity of large-scale problems.

The second contribution of this work is using an optimization-based trajectory planning method, SCvx, for planning trajectory through candidate viewpoints with continuous data acquisition for dense scene reconstruction. Two scenarios, Problem 1 and Problem 2 were developed. Problem 2 introduces a novel formulation that incorporates STCs [30, 46, 47, 56–58] within the ctcs framework [19], which is crucial for ensuring smooth transitions between multiple target objects while maintaining LoS constraints.

This work aimed to develop a systematic approach using a 3D scene proxy to generate a dynamically feasible and efficient trajectory for inspection.

6.2 Limitations

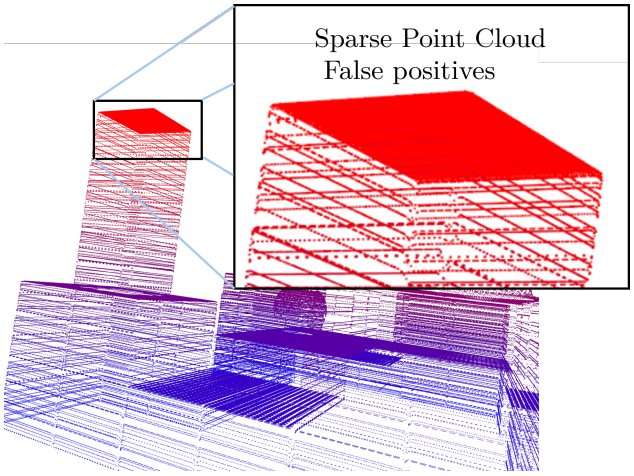
While the proposed framework demonstrates significant advantages, certain limitations must be acknowledged:

- As the viewpoint selection depends on the geometric proxy, sparse point cloud data can result in false positives on cloud visibility when evaluating the viewpoints for generating visibility vector, which results in an inaccurate visibility matrix (see Figure 6.1)

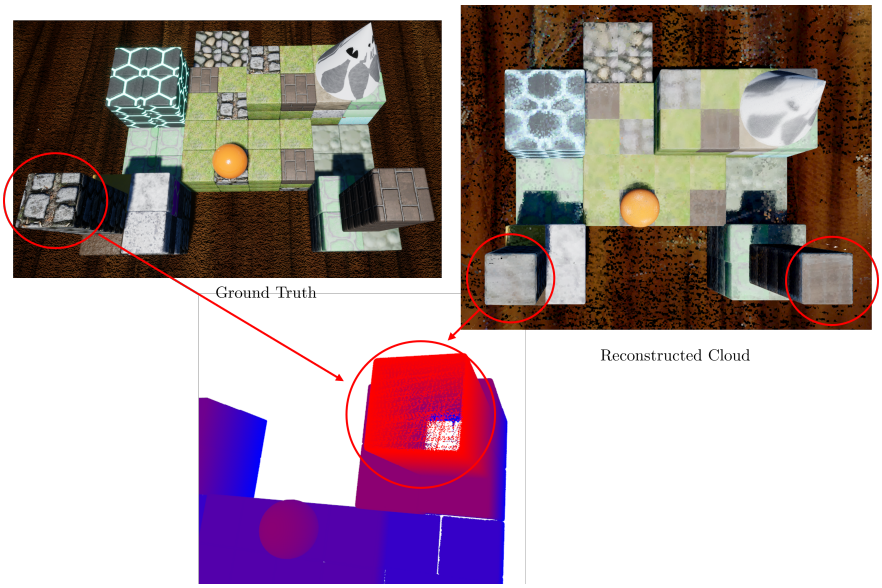
- Despite achieving significantly better cloud reconstruction quality using Problem 2 compared to the off-the-shelf planner (see Figure 5.5), the LoS constraint was violated at some nodes within the prescribed trigger window. The state-triggered LoS constraint is useful when there are multiple target objects, as it facilitates a smooth transition between them. Problem 1 and Problem 2 combine path and control constraints with the system dynamics to ensure continuous-time satisfaction, as outlined in [19]. However, incorporating STCs within the ctcs framework is unprecedented and requires thorough investigation to address the issues highlighted in Figure 5.8b. While Problem 1 implements the LoS constraint in the ctcs [19] framework without trigger region, a comprehensive evaluation is needed to determine the root cause of the observed behavior.

6.3 Future Directions

A key assumption or requirement of this framework is the availability of a 3D scene proxy, either pre-existing or gathered during an initial exploratory flight. An interesting direction for future research would be to extend or adapt the methods discussed in this thesis to handle environments where the 3D scene is unknown.



(a) Sparse input point cloud generated from exploration



(b) Reconstructed point cloud

Figure 6.1: Problem Formulations with LoS constraint

Additionally, another promising avenue would be to explore the integration of view planning and path planning into a unified optimization approach. Instead of the two-step process adopted in this thesis, where viewpoint optimization and trajectory planning are handled separately, a joint optimization framework that simultaneously considers optimal sensor poses and a dynamically feasible trajectory could lead to more efficient and robust solutions. This combined approach could potentially enhance performance in real-world applications.

Currently, after getting a feasible or good set of viewpoints, these viewpoints, which are waypoints for the UAV, are ordered using heuristics, such as formulating it as TSP to ensure every viewpoint is visited. An interesting idea would be to leverage the Signal Temporal Logic (STL) [61] formulation for waypoint constraint and combine it with trajectory optimization.

BIBLIOGRAPHY

[1] Agisoft. Agisoft:mission planning for complex structures, 2024.

[2] João Alves. Comprehensive overview of depth cameras.

[3] Brendan Alvey, Derek Anderson, Andrew Buck, Matthew Deardorff, and James Keller. Simulated photorealistic deep learning framework and workflows to accelerate computer vision and unmanned aerial vehicle research. *Workshop on Analysis of Aerial Motion Imagery (WAAMI) in conjunction with International Conference on Computer Vision (ICCV 2021)*.

[4] Giulio Attenni, Viviana Arrigoni, Novella Bartolini, and Gaia Maselli. Drone-based delivery systems: A survey on route planning. *IEEE Access*, 11:123476–123504, 2023.

[5] J E Banta, L R Wong, C Dumont, and M A Abidi. A next-best-view system for autonomous 3-D object reconstruction. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(5):589–598, September 2000.

[6] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems*. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984. 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984.

[7] Daniel Broyles, Christopher R. Hayner, and Karen Leung. Wisard: A labeled visual and thermal image dataset for wilderness search and rescue. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9467–9474, 2022.

[8] Evangelos Chatzikalymnios and Konstantinos Moustakas. Landing site detection for autonomous rotor wing uavs using visual and structural information. *Journal of Intelligent Robotic Systems*, 104, 02 2022.

[9] Chandra Chekuri and M. Pal. A recursive greedy algorithm for walks in directed graphs. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 245–253, 2005.

[10] Shengyong Chen, Youfu Li, and Ngai Ming Kwok. Active vision in robotic systems: A survey of recent developments. *Int. J. Rob. Res.*, 30(11):1343–1377, September 2011.

[11] CloudCompare. 3d point cloud and mesh processing software open source project. Available from https://www.cloudcompare.org/.

[12] Christopher I. Connolly. The determination of next best views. *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, 2:432–435, 1985.

[13] C.K. Cowan and P.D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):407–416, 1988.

[14] Jinda Cui, John T. Wen, and Jeff Trinkle. A multi-sensor next-best-view framework for geometric model-based robotics applications. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8769–8775, 2019.

[15] P. Debus and V. Rodehorst. Evaluation of 3d uas flight path planning algorithms. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B1-2021:157–164, 2021.

[16] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 2016. To appear.

[17] Kevin Dorling, Jordan Heinrichs, Geoffrey G. Messier, and Sebastian Magierowski. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2017.

[18] Daniel Dueri, Yuanqi Mao, Zohaib Mian, Jerry Ding, and Behçet Açikmeşe. Trajectory optimization with inter-sample obstacle avoidance via successive convexification. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1150–1156, 2017.

[19] Purnanand Elango, Dayou Luo, Abhinav G. Kamath, Samet Uzun, Taewan Kim, and Behçet Açıkmeşe. Successive convexification for trajectory optimization with continuous-time constraint satisfaction, 2024.

[20] Epic Games. Unreal engine. Available from https://www.unrealengine.com.

[21] Chuanxiang Gao, Xinyi Wang, Ruoyu Wang, Zuoquan Zhao, Yu Zhai, Xi Chen, and Ben M Chen. A UAV-based explore-then-exploit system for autonomous indoor facility inspection and scene reconstruction. *Autom. Constr.*, 148:104753, April 2023.

[22] Paul J. Goulart and Yuwen Chen. Clarabel: An interior-point solver for conic programs with quadratic objectives, 2024.

[23] Markus Grotz. *Active Vision for Scene Understanding*. KIT Scientific Publishing, Karlsruhe, Dec 2021.

[24] Jingjing Gu, Tao Su, Qiuhong Wang, Xiaojiang Du, and Mohsen Guizani. Multiple moving targets surveillance based on a cooperative network for multi-uav. *IEEE Communications Magazine*, 56(4):82–89, 2018.

[25] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[26] Christopher R. Hayner, Samuel C. Buckner, Daniel Broyles, Evelyn Madewell, Karen Leung, and Behçet Açikmeşe. Halo: Hazard-aware landing optimization for autonomous systems. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3261–3267, 2023.

[27] Benjamin Hepp, Matthias Nießner, and Otmar Hilliges. Plan3d: Viewpoint and trajectory optimization for aerial multi-view stereo reconstruction. *ACM Trans. Graph.*, 38(1), dec 2018.

[28] Christof Hoppe, Andreas Wendel, Stefanie Zollmann, Katrin Pirker, Arnold Irschara, and Horst Bischof. Photogrammetric camera network design for micro aerial vehicles. 2012.

[29] Stefan Isler, Reza Sabzevari, Jeffrey Delmerico, and Davide Scaramuzza. An information gain formulation for active volumetric 3d reconstruction. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3477–3484, 2016.

[30] Abhinav G. Kamath, Purnanand Elango, Taewan Kim, Skye Mceowen, Yue Yu, John M. Carson, Mehran Mesbahi, and Behcet Acikmese. *Customized Real-Time First-Order Methods for Onboard Dual Quaternion-based 6-DoF Powered-Descent Guidance.*

[31] Abhinav G Kamath, Purnanand Elango, Yue Yu, Skye Mceowen, Govind M Chari, John M Carson, III, and Behçet Açıkmeşe. Real-time sequential conic optimization for multi-phase rocket landing guidance. December 2022.

[32] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. *ACM Trans. Graph.*, 26(3):24–es, jul 2007.

[33] Unsik Lee and Mehran Mesbahi. Optimal power descent guidance with 6-DoF line of sight constraints via unit dual quaternions. In *AIAA Guidance, Navigation, and Control Conference*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, January 2015.

[34] Ligang Liu, Xi Xia, Han Sun, Qi Shen, Juzhan Xu, Bin Chen, Hui Huang, and Kai Xu. Object-aware guidance for autonomous scene reconstruction. *ACM Transactions on Graphics*, 37(4):1–12, July 2018.

[35] Mehdi Maboudi, Mohammadreza Homaei, Soohwan Song, Shirin Malihi, Mohammad Saadatseresht, and Markus Gerke. A review on viewpoints and path planning for UAV-Based 3-D reconstruction. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 16:5026–5048, 2023.

[36] E. Madewell, E. Pollack, H. Kuni, S. Johri, D. Broyles, J. Vagners, and K. Leung. Beyond visual line-of-sight uncrewed aerial vehicle for search and locate operations. In *AIAA Scitech Forum*, 2024.

[37] Danylo Malyuta, Taylor P. Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behçet Açıkmeşe. Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently. *IEEE Control Systems Magazine*, 42(5):40–113, 2022.

[38] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems and its convergence properties. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, page 3636–3641. IEEE Press, 2016.

[39] Enrico Zenerino Mario Silvagni, Andrea Tonoli and Marcello Chiaberge. Multipurpose uav for search and rescue operations in mountain avalanche events. *Geomatics, Natural Hazards and Risk*, 8(1):18–33, 2017.

[40] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):417–433, 1993.

[41] Skye Mceowen, Abhinav G. Kamath, Purnanand Elango, Taewan Kim, Samuel C. Buckner, and Behcet Acikmese. *High-Accuracy 3-DoF Hypersonic Reentry Guidance via Sequential Convex Programming.*

[42] Donald Meagher. Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer, 10 1980.

[43] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, Inc., USA, 1987.

[44] Richard A Pito. A solution to the next best view problem for automated CAD model acquisition of Free-Form objects using range cameras.

[45] Rerun Development Team. Rerun: A visualization sdk for multimodal data, 2024. Available from https://www.rerun.io/ and https://github.com/rerun-io/rerun.

[46] Taylor Reynolds, Michael Szmuk, Danylo Malyuta, Mehran Mesbahi, Behcet Acikmese, and John M. Carson. *A State-Triggered Line of Sight Constraint for 6-DoF Powered Descent Guidance Problems*.

[47] Taylor P. Reynolds, Michael Szmuk, Danylo Malyuta, Mehran Mesbahi, Behçet Açıkmeşe, and John M. Carson. Dual quaternion-based powered descent guidance with state-triggered constraints. *Journal of Guidance, Control, and Dynamics*, 43(9):1584–1599, 2020.

[48] Mike Roberts, Shital Shah, Debadeepta Dey, Anh Truong, Sudipta Sinha, Ashish Kapoor, Pat Hanrahan, and Neel Joshi. Submodular trajectory optimization for aerial 3d scanning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5334–5343, 2017.

[49] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. IEEE.

[50] Andrey V. Savkin and Hailong Huang. Proactive deployment of aerial drones for coverage over very uneven terrains: A version of the 3d art gallery problem. *Sensors*, 19(6), 2019.

[51] William Scott. Model-based view planning. *Mach. Vis. Appl.*, 20:47–69, 11 2009.

[52] William R Scott, Gerhard Roth, and Jean-François Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys*, 35(1):64–96, March 2003.

[53] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.

[54] Michael Szmuk. *Successive convexification high performance feedback control for agile flight*. PhD thesis, University of Washington, 2019.

[55] Michael Szmuk and Behçet Açikmese. Successive convexification for 6-dof mars rocket powered landing with free-final-time. *arXiv: Optimization and Control*, 2018.

[56] Michael Szmuk, Danylo Malyuta, Taylor P. Reynolds, Margaret Skye Mceowen, and Behçet Açikmeşe. Real-time quad-rotor path planning using convex optimization and compound state-triggered constraints. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7666–7673, 2019.

[57] Michael Szmuk, Taylor P. Reynolds, Behcet Acikmese, Mehran Mesbahi, and John M. Carson III au2. Successive convexification for 6-dof powered descent guidance with compound state-triggered constraints, 2019.

[58] Michael Szmuk, Taylor P. Reynolds, and Behçet Açıkmeşe. Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints. *Journal of Guidance, Control, and Dynamics*, 43(8):1399–1413, August 2020.

[59] G.H. Tarbox and S.N. Gottschlich. Planning for complete sensor coverage in inspection. *Computer Vision and Image Understanding*, 61(1):84–111, 1995.

[60] Teodor Tomic, Korbinian Schmid, Philipp Lutz, Andreas Domel, Michael Kassecker, Elmar Mair, Iris Lynne Grixa, Felix Ruess, Michael Suppa, and Darius Burschka. Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. *IEEE Robotics Automation Magazine*, 19(3):46–56, 2012.

[61] Samet Uzun, Purnanand Elango, Pierre-Loic Garoche, and Behcet Acikmese. Optimization with temporal and logical specifications via generalized mean-based smooth robustness measures, 2024.

[62] Bingxu Wang, Jinhui Lan, and Jiangjiang Gao. Lidar filtering in 3d object detection based on improved ransac. *Remote Sensing*, 14(9), 2022.

[63] Rui Zeng, Yuhui Wen, Wang Zhao, and Yong-Jin Liu. View planning in robot active vision: A survey of systems, algorithms, and applications. *Computational Visual Media*, 6(3):225–245, September 2020.

[64] Han Zhang, Yucong Yao, Ke Xie, Chi-Wing Fu, Hao Zhang, and Hui Huang. Continuous aerial path planning for 3D urban scene reconstruction. *ACM Trans. Graph.*, 40(6):1–15, December 2021.

[65] Jian Zhang and Hailong Huang. Occlusion-aware uav path planning for reconnaissance and surveillance. *Drones*, 5(3), 2021.

[66] Keqi Zhang, Shu-Ching Chen, D Whitman, Mei-Ling Shyu, Jianhua Yan, and Chengcui Zhang. A progressive morphological filter for removing nonground measurements from airborne LIDAR data. *IEEE Trans. Geosci. Remote Sens.*, 41(4):872–882, April 2003.

[67] Wuming Zhang, Jianbo Qi, Peng Wan, Hongtao Wang, Donghui Xie, Xiaoyan Wang, and Guangjian Yan. An easy-to-use airborne lidar data filtering method based on cloth simulation. *Remote Sensing*, 8(6), 2016.

[68] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[69] Zhenyu Zhou, Chuntian Zhang, Chen Xu, Fei Xiong, Yan Zhang, and Tariq Umer. Energy-efficient industrial internet of uavs for power line inspection in smart grid. *IEEE Transactions on Industrial Informatics*, 14(6):2705–2714, 2018.

Appendix A

QUATERNIONS

A.1 Representation

The complex number representation of quaternion is given by:-

$$q = q_0 + q_1\hat{i} + q_2\hat{j} + q_3\hat{k},$$

Where,

$$\begin{aligned}\hat{i}^2 = \hat{j}^2 = \hat{k}^2 = \hat{i}\hat{j}\hat{k} &= -1\\ \hat{i}\hat{j} &= k = -\hat{j}\hat{i}\\ \hat{j}\hat{k} &= i = -\hat{k}\hat{j}\\ \hat{k}\hat{i} &= j = -\hat{i}\hat{k},\end{aligned}$$

The vector representation of quaternion is given by:-

$$q = \left[\underbrace{q_0}_{scalar} \quad \underbrace{\begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}}_{\text{axis of rotation}}\right]^T,$$

The Axis-Angle representation of a quaternion is given by:-

$$q = \begin{bmatrix}\cos\frac{\theta}{2} & \mathbf{a}\sin\frac{\theta}{2}\end{bmatrix}^T,$$

where θ is the angle of rotation and $\mathbf{a}$ is the axis of rotation

A.2 Product

The product of two quaternions p and q is denoted by $p \otimes q$ and is defined as:-

$$\begin{aligned} p \otimes q &= (p_0 + \underbrace{p_1\hat{i} + p_2\hat{j} + p_3\hat{k}}_{p_v})(q_0 + \underbrace{q_1\hat{i} + q_2\hat{j} + q_3\hat{k}}_{q_v}), \\ &= \underbrace{p_0 q_0 - p_v^T q_v}_{scalar} + \underbrace{p_0 q_v + q_0 p_v + p_v \times q_v}_{vector} \\ &= \begin{bmatrix} p_0 & -p_v^T \\ p_v & p_0\mathbf{I} + [p_v\times] \end{bmatrix} \begin{bmatrix} q_0 \\ q_v \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} q_0 & -q_v^T \\ q_v & q_0\mathbf{I} - [q_v\times] \end{bmatrix}}_{\Omega(q)} \begin{bmatrix} p_0 \\ p_v \end{bmatrix}, \end{aligned}$$

where:

$$[q_v\times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix},$$

$$\Omega(q) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix},$$

A.3 3D Rotation

$q_{\mathcal{B}/\mathcal{I}}$[1] is the quaternion of frame $\mathcal{B}$ with respect to frame $\mathcal{I}$. Let $q_{\mathcal{B}/\mathcal{I}} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}$. If $p_{\mathcal{B}} \in \mathbb{R}^4$ is a quaternion representation of a parameter in $\mathcal{B}$ frame ($p_{\mathcal{B}} = \begin{bmatrix} 0 & {}^{\mathcal{B}}p_x & {}^{\mathcal{B}}p_y & {}^{\mathcal{B}}p_z \end{bmatrix}^T$), then we can get the parameter in inertial frame, $p_{\mathcal{I}} =$

[1] B/I $\equiv \mathcal{I} \leftarrow \mathcal{B}$, orientation of $\mathcal{B}$ wrt $\mathcal{I}$

$\begin{bmatrix} 0 & {}^{\mathcal{I}}p_x & {}^{\mathcal{I}}p_y & {}^{\mathcal{I}}p_z \end{bmatrix}$ by the following equation:

$$p_{\mathcal{I}} = q_{\mathcal{B}/\mathcal{I}} \otimes p_{\mathcal{B}} \otimes q^*_{\mathcal{B}/\mathcal{I}},$$

Where $q^*_{\mathcal{B}/\mathcal{I}} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix}^T$ is the conjugate of $q_{\mathcal{B}/\mathcal{I}}$. We can covert the above quaternion product to a series of matrix multiplication as follows:-

$$\begin{bmatrix} 0 \\ {}^{\mathcal{I}}p_x \\ {}^{\mathcal{I}}p_y \\ {}^{\mathcal{I}}p_z \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ {}^{\mathcal{B}}p_x \\ {}^{\mathcal{B}}p_y \\ {}^{\mathcal{B}}p_z \end{bmatrix},$$

$$\begin{bmatrix} {}^{\mathcal{I}}p_x \\ {}^{\mathcal{I}}p_y \\ {}^{\mathcal{I}}p_z \end{bmatrix} = \underbrace{\begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_1q_0) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix}}_{C_{\mathcal{B}/\mathcal{I}}} \begin{bmatrix} {}^{\mathcal{B}}p_x \\ {}^{\mathcal{B}}p_y \\ {}^{\mathcal{B}}p_z \end{bmatrix},$$

Where $C_{\mathcal{B}/\mathcal{I}}$ is the Direction Cosine Matrix for $q_{\mathcal{B}/\mathcal{I}}$.