Degree Project in Electrical Engineering, specializing in Systems, Control and Robotics

Second cycle, 30 credits

Hybrid PBVS-IBVS Model Predictive Visual Servoing

For Autonomous Docking of a Free-Flying Robot

TAFARREL FIRHANNOZA PRAMONO

Hybrid PBVS-IBVS Model Predictive Visual Servoing

For Autonomous Docking of a Free-Flying Robot

TAFARREL FIRHANNOZA PRAMONO

Master's Programme, Systems, Control and Robotics, 120 credits
Date: September 28, 2025

Supervisors: Pedro Roque, Pedro Miraldo
Examiner: Dimos V. Dimarogonas
School of Electrical Engineering and Computer Science
Swedish title: Hybrid PBVS–IBVS Modellprediktiv Visuell Servostyrning
Swedish subtitle: För Autonom Dockning av en Fritt Flygande Robot

Abstract

Interest in the development of autonomous rendezvous and docking (ARD) has been growing since the 1960s as it is an important procedure for refueling spacecraft and transferring resources. Achieving centimeter-scale accuracy in ARD commonly relies on Visual Servoing (VS). Two common classical VS methods are Image-Based Visual Servoing (IBVS) and Position-Based Visual Servoing (PBVS) utilize a velocity-based control law that is straightforward, yet suffer from depth or visibility sensitivity and local minima.

This thesis presents a hybrid VS with Model Predictive Control (MPC) docking pipeline for a free-flyer that requires only an RGB-D camera and a Computer Aided Design (CAD) model of a docking station. The pose of the station is initialized by Globally optimal-ICP (GO-ICP) and refined with Generalized Iterative Closest Point (GICP). This estimation seeds a trajectory-tracking MPC that orients the free-flyer to maximize the visibility of geometric point features. A dynamic weighting strategy in the MPC cost is then used to control the soft-switching between PBVS and IBVS. Finally, the feature dynamics is described to the MPC to minimize the image errors while respecting system constraints.

We evaluated the method in Robot Operating System (ROS) with Gazebo on a planar 3-degree of freedom (DOF) free-flyer and further validated it in the KTH Space Robotics Laboratory (SRL), reporting docking success rate, steady-state pose error, and computation time. The results indicated reliable docking without AR tags, using only geometric fiducials, and demonstrated that the PBVS-IBVS transition improves visibility and mitigates local-minimum failures.

Keywords

Computer vision, Free-flyer, Iterative Closest Point, Model Predictive Control, Visual servo

Sammanfattning

Intresset för utvecklingen av autonomous rendezvous and docking (ARD) har ökat sedan 1960-talet, eftersom det är ett viktigt förfarande för att tanka rymdfarkoster och överföra resurser. Att uppnå noggrannhet på centimeternivå i ARD bygger ofta på Visual Servoing (VS). Två vanliga klassiska VS-metoder är Image-Based Visual Servoing (IBVS) och Position-Based Visual Servoing (PBVS), som använder en hastighetsbaserad reglerlag som är enkel men ändå känslig för djup och synlighet samt drabbas av lokala minima.

Denna avhandling presenterar en hybrid VS- och Model Predictive Control (MPC)-baserad dockningspipeline för en fri-flygande plattform som endast kräver en RGB-D-kamera och en Computer Aided Design (CAD)-modell av en dockningsstation. Stationens pose initieras med Globally optimal-ICP (GO-ICP) och förfinas med Generalized Iterative Closest Point (GICP). Denna skattning initierar en banaföljande MPC som orienterar den fri-flygande plattformen för att maximera synligheten av geometriska punktlandmärken. En dynamisk viktning i MPC-kostnadsfunktionen används därefter för att åstadkomma mjuk växling mellan PBVS och IBVS. Detta gör att MPC kan optimera kamerans hastigheter för att minimera bildfelet samtidigt som systembegränsningarna respekteras.

Vi utvärderade metoden i Robot Operating System (ROS) med Gazebo på en plan 3-degree of freedom (DOF) fri-flygande plattform och validerade den vidare i KTH Space Robotics Laboratory (SRL). Vi redovisar andelen lyckade dockningar, stationärt posefel och beräkningstid. Resultaten visar tillförlitlig dockning utan AR-taggar — med enbart geometriska referensmarkörer — och att övergången PBVS–IBVS förbättrar synligheten och minskar fel orsakade av lokala minima.

Nyckelord

Datorseende, Friflygande robot,Iterative Closest Point, Modellprediktiv reglering, Visuell servostyrning

Acknowledgments

First and foremost, I would like to express my gratitude to God Almighty for granting me strength and knowledge, for easing my path through my master's studies, and for connecting me with the wonderful people who supported me. I also thank Professor Dimos V. Dimarogonas for giving me an opportunity to work in the lab as a research assistant and allowing me to do my thesis in this lab.

I especially would like to thank my supervisors, Pedro Roque and Pedro Miraldo. They helped me to be on track with the progress and provided great suggestions and feedback to help me finish this project. I would also thank the other PhD students in the DISCOWER project, Elias Krantz and Joris Verhagen, for giving me insights and helped with the hardware experiments.

I thank my beloved partner, Nabilah Amalina, for her faith in me, for always standing by my side, for keeping my spirits up through my toughest times, and for all her unwavering love and care she provided. I am deeply grateful to my family, especially my Mom and Dad, for their unconditional love and for being there to cheer me up. I also thank my friends in Sweden for making my master's journey more colorful.

Parts of this thesis project have involved the use of Large Language Models (LLMs), such as ChatGPT and Claude, to support formulating ideas, drafting text, grammar checking, and code debugging. However, the scientific contributions, technical results, and proofs presented are entirely from my own work and based on the cited literature and my independent research.

Finally, I would like to express my sincere gratitude to the Indonesia Endowment Fund for Education (LPDP) for awarding me a full scholarship to pursue my master's degree at KTH Royal Institute of Technology.

Stockholm, September 2025
Tafarrel Firhannoza Pramono

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

ARD	autonomous rendezvous and docking
ATMOS	Autonomy Testbed for Multi-purpose Orbiting Systems
BnB	branch-n-bound
CAD	Computer Aided Design
DOF	degree of freedom
EKF	Extended Kalman filter
FOV	field of view
FPFH	Fast Point Feature Histograms
GHT	Generalised Hough transform
GICP	Generalized Iterative Closest Point
GO-ICP	Globally optimal-ICP
GPU	graphics Processing Unit
HITL	hardware in the loop
IBVS	Image-Based Visual Servoing
ICP	Iterative Closest Point
LLM	Large language model
LQR	Linear–Quadratic Regulator
MPC	Model Predictive Control
MPVS	Model Predictive Visual Servoing
MSE	Mean Squared Error
PBVS	Position-Based Visual Servoing
PCA	principal component analysis
PCL	Point Cloud Library
PnP	Perspective-n-Point

RANSAC	RANdom SAmple Consensus
RGB	Red Green Blue
RGB-D	Red Green Blue-Depth
ROS	Robot Operating System
SHOT	Signature of Histograms of OrienTations
SIFT	Scale Invariant Feature Transform
SITL	software in the loop
SRL	Space Robotics Laboratory
SURF	Speeded-Up Robust Features
SVD	Singular value decomposition
TOF	Time-of-Flight
VS	Visual Servoing

Chapter 1

Introduction

The introductory chapter provides an overview of the thesis. It starts with the background of autonomous rendezvous and docking (ARD), highlighting how important visual servoing methods are in space robotics. It is then followed by a problem statement that needs to be tackled in markerless docking and motivates the implementation of hybrid visual servoing. The purpose and objectives of the work are also discussed, clarifying both its academic and practical contributions. The chapter defines the research methodology, examines the ethical and sustainability aspects, and discusses the delimitations of this work. Finally, the thesis's structure is described to guide the reader through the subsequent chapters.

1.1 Background

Since the middle of the 20th century, research on ARD has been active due to its crucial role in space missions such as crew transfers, satellite servicing, orbital assembly, and spacecraft refueling operations [1]. NASA's Technology Roadmap highlighted ARD as a critical technology for developing space exploration capabilities due to its strategic significance [2].

Historically, spacecraft docking procedures predominantly relied on human-in-the-loop teleoperation methods, where human operators remotely control the spacecraft movement [3]. However, teleoperation methods suffer mainly from latency issues, robustness, and resource efficiency. Consequently, research initiatives shifted towards fully autonomous docking methodologies, minimizing or completely eliminating dependency on human operators in spaceship operations.

Extensive research on autonomous docking has incorporated VS method-

ologies, allowing the robot to effectively approach, grasp, and manipulate objects by controlling their relative motion based on visual information [4]. The classical VS methods are classified by the type of feedback information they use into three categories: IBVS, PBVS, and hybrid VS [5, 6].

In IBVS, 2D image features are tracked; the image-plane error between their current and desired feature location is used for the control error feedback, allowing the robot to converge without an explicit 3D model. In PBVS, it will first obtain a full 6D pose estimation of the target and then drive the pose error to zero. This approach often provides smoother trajectories but can be sensitive to camera calibration drift and visibility issues. Hybrid VS blends these 2 error signals by combining Cartesian-space and image-space terms. This method has the advantages of IBVS and PBVS, which can provide a broad convergence region of PBVS while retaining local stability and visibility robustness of IBVS [7, 8, 5]. Therefore, to implement hybrid VS, we need a reliable image-feature detection, followed by a 6-degree of freedom (DOF) pose estimation method.

Current pose estimators are divided into several categories. First, correspondence-based method that utilizes object features. This method has low computational load but is sensitive to texture-less objects [9, 10]. Iterative Closest Point (ICP) and related dense alignment algorithms that refine a coarse pose to sub-millimeter accuracy are also included in this category. Second, a template-based matching, which copes well with low texture yet scales poorly with the number of stored views. Lastly, a regression-based methods that can achieve state-of-the-art robustness at the cost of extensive training data and high computational load [11, 12, 13, 14, 4].

Robust and reliable feature extraction and pose estimation only solve half of the hybrid VS problem. Once the required visual information is available, the controller must respect actuator limits, keep the target in the camera field of view (FOV), and avoid collisions. While classical IBVS or PBVS laws use pure proportional control and cannot handle constraints explicitly, MPC overcomes this limitation. MPC offers a receding-horizon control strategy, predicting the robot's future states over a finite horizon and minimizing a cost function that can follow inequality constraints [15, 16].

1.2 Problem Statement

Despite significant advancements in ARD, current methods still have some limitations. Many autonomous docking systems still depend on special markers (e.g., AprilTags, ARTags, ARToolkit) placed on the target [17], or

on human-assisted teleoperation. Both approaches limit their applicability in semi-structured or uncertain environments. Therefore, relying on artificial markers remains a significant challenge for space applications where robustness and adaptability are essential.

Regarding VS modes, both IBVS and PBVS offer advantages and suffer from inherent drawbacks. Hybrid VS aims to combine the strengths of both approaches, providing improved robustness and convergence. However, designing dynamic switching strategies that enable seamless transitions between IBVS and PBVS remains an active and open research challenge [18, 19].

This thesis develops a combination of visual servo control laws with MPC, utilizing blob-shaped fiducial markers that represent semi-unstructured environment. The work aims to bridge the gap between structured and fully unstructured docking scenarios. The following research questions are prompted for this thesis.

- What is the performance (computational time, accuracy, and robustness) of the proposed hybrid MPVS approach?
- How can dynamic weighting strategies be designed and tuned to achieve high-performance hybrid IBVS–PBVS behavior within an MPC framework?
- How does the design of the switching criteria in a hybrid visual servoing system affect the performance?

1.3 Purpose

The purpose of this master's thesis is to design and validate an autonomous docking system using a hybrid Model Predictive Visual Servoing (MPVS). The system identifies the object (in this case, the docking station) using its Computer Aided Design (CAD) model and an RGB-D camera, then uses MPC to maneuver the free-flyer. This system is tested in the Space Robotics Laboratory (SRL), established under the Wallenberg AI, Autonomous Systems and Software Program (WASP) research program named DISCOWER (Distributed Control in Weightless Environments) at KTH's Integrated Transport Research Lab (ITRL). By showing the centimeter-level accuracy and a high docking success rate, this work provides significant value to one of the lab's objectives—to automate the refueling process.

1.4 Goals

Develop and demonstrate a modular, low-latency docking pipeline that enables a free-flyer to estimate the pose of untagged docking fixtures, and autonomously align and mate with those fixtures through a hybrid VS. This has been divided into the following two sub-goals as referred to in the research questions:

1. Pose Estimation Evaluation: Evaluate the performance of proposed GO-ICP with GICP pipeline.

2. Hybrid VS Efficacy: Design the proposed hybrid VS switching strategies and compare their performance in terms of convergence rate, final pose error, and docking success rate.

1.5 Research Methodology

This thesis uses quantitative and experimental research to answer the research questions: pose error in meters, angular error in degrees, convergence time in seconds, success rate in percent, and computational load in seconds (pipeline update rate).

The overall research approach follows a classic design-build-evaluate cycle. First, a structured literature review compares the strengths and weaknesses of existing pose-estimation approaches and visual-servo controllers. The most suitable combination of a free-flying robot is then selected and implemented as a modular Robot Operating System (ROS) 2 system, with the simulation running in Gazebo Harmonic. The same software is subsequently deployed on the Jetson-powered free-flyer in the DISCOWER SRL for hardware-in-the-loop evaluation. This two-step validation ensures that conclusions about the research questions hold in both virtual and real environments.

1.6 Ethics and Sustainability

This thesis addresses several important ecological, environmental, and social sustainability considerations on the development of autonomous docking process.

From an ecological perspective, the proposed method might contribute to enable key missions such as refueling, crew transfer, assembly, and servicing

in orbit. By improving docking accuracy and robustness, it may help to extend the operational lifetime of spacecraft and minimize material waste.

From an economic perspective, autonomous docking development can reduce mission cost by lowering the damage and risk of failure. Furthermore, the need for manual supervision can be reduced.

From a social perspective, autonomous docking reduce the need for human involvement in a potentially hazardous environment. Furthermore, research on visual servoing often can be adopted in other domains, such as autonomous vehicles or medical robots, bringing the society into more efficient and safer technologies.

1.7 Delimitations

This research is delimited by the following factors:

- 3-DOF microgravity hardware validation: Experiments are restricted to the lab's planar 3-DOF free-flyer. Extending the pipeline to full 6-DOF motion in true microgravity would require a different testbed.
- Learning-based pose estimation: As mentioned in section 1.1, state-of-the-art deep-learning networks for 6-DOF pose estimation demand a high performance of embedded GPU and memory bandwidth that exceeds the capabilities of the onboard computer available on the DISCOWER free-flyer.
- Hybrid weighting strategy: This thesis implemented an approach to dynamically blend PBVS and IBVS in the MPC. Time constraints necessitate further research into alternative switching strategies to develop more efficient and intelligent control schemes.
- Environmental Disturbances: The system does not fully account for extreme object occlusions and extreme lighting conditions. The evaluation assumes static lighting and sensor noise with a small number of occlusions.

1.8 Structure of the thesis

Chapter 2 presents relevant background information and theoretical concepts about visual servo control categories (IBVS, PBVS, hybrid VS), pose estimation methods, quaternions, and MPC fundamentals. Chapter 3 explains

the coordinate system and modeling of the free-flyer. Chapter 4 describes the detailed design of the custom docking station. Chapter 5 details the methods used to solve the problem and how the theories are developed and integrated, starting from processing the point cloud, cascaded ICP, and MPVS controller. Chapter 6 presents the preparation to perform the experiment as well as the experiment results, including the performance of the pose estimation and hybrid VS, and discusses the results of different hybrid VS approaches. Finally, Chapter 7 summarizes the main findings and suggests potential research for future works.

Chapter 2

Background

This chapter addresses the foundations needed for this work. The chapter starts with the overview of Visual Servoing and its classification, highlighting their principles, strengths, and limitations. It also covers several key concepts such as point cloud processing, pose estimation methods, and quaternion representations for 3D rotations. This chapter also introduces Model Predictive Control (MPC) as the control framework that facilitates predictive capabilities with constraint handling during docking procedures.

2.1 Visual Servoing

Visual Servoing (VS) is a common technique that leverages visual information as input of a closed-loop control system for controlling the motion of a dynamical system. The main classification of VS is based on how the control error is defined. In IBVS, the error is expressed directly in the image space using visual features such as points, lines, or contours. In contrast, PBVS defines the error in 3D space, using the estimated pose of the camera or of the object relative to the camera. Hybrid VS combines the 2D features error IBVS with 3D information PBVS, aiming to benefit from each category.

Besides this primary classification, VS systems can also be characterized according to the camera configuration and the number of cameras used. With respect to camera placement, vision data can be obtained from a camera mounted directly on the robot (eye-in-hand configuration) or from a fixed camera, observing the robot in the workspace (eye-to-hand configuration). Based on the number of cameras, there are monocular, binocular (stereo), and multi-camera systems. Monocular systems are easy to use and cost-effective although limited to 2D image measurements without direct depth estimation.

Stereo systems enable depth recovery by using triangulation. However, it requires more complex calibration and Jacobian computation. Multi-camera systems can provide richer information, wider FOV, and better robustness against occlusions, at the expense of increased processing loads [20, 5].

The goal of vision-based control is to minimize an error term that is usually defined by the following equation form:

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^* \tag{2.1}$$

The error is defined from the measured visual features $\mathbf{s}$ from image measurements $\mathbf{m}(t)$, with a set of parameters $\mathbf{a}$ that adds more information about the system, such as 3D models of the object or camera parameters. Vector $\mathbf{s}^*$ represents the desired values of the features.

After $\mathbf{s}$ is chosen, the control scheme can be designed by obtaining the relationship between camera twist (linear and angular velocity of the camera frame) ${}^o\boldsymbol{\xi} = ({}^o\mathbf{v}, {}^o\boldsymbol{\omega})$ and the time evolution of $\mathbf{s}$ through an interaction matrix $\mathbf{L_s}$, which depends on the selection of $\mathbf{s}$. The dynamics of visual features can be described with the following equation:

$$\dot{\mathbf{s}} = \mathrm{L_s}\, {}^o\boldsymbol{\xi} \tag{2.2}$$

2.1.1 IBVS

IBVS uses the image-plane coordinates and pinhole camera model to derive the image Jacobian, known as the interaction matrix, and formulate the control law. This approach directly uses visual features from the image plane to control the robot to the desired configuration. While this method is more robust to calibration and model errors, the main drawback is that the generated 3D trajectory is unpredictable and might be undesirable [21].

2.1.1.1 Camera Perspective Projection

The perspective projection, derived from the pinhole camera model, provides a mathematical model representation to project 3D points in the world into 2D points in the image plane.

Given a 3D point $\mathbf{P} = [X, Y, Z]^\top$ in the camera coordinate frame, its projection in the image plane $p = [x, y]^\top$ is defined by the perspective projection equations below.

$$x = \frac{X}{Z}, \quad y = \frac{Y}{Z} \tag{2.3}$$

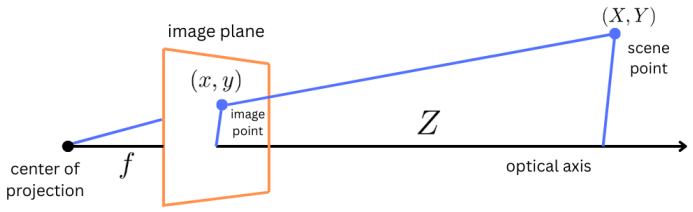

Figure 2.1: Perspective projection [22].

Considering a digital camera that uses pixel-based image sensors, these coordinates then relate to the pixel coordinates (u, v) using the camera's intrinsic parameters, such as focal length, principal point, and pixel size. These parameters then form the camera intrinsic matrix as defined in eq. (2.4), where f_x and f_y are the focal lengths in pixel along x and y axis, and (c_x, c_y) is the pixel coordinate of the principal point.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

2.1.1.2 Interaction Matrix

The interaction matrix linearly relates the spatial velocity of the camera ${}^{o}\boldsymbol{\xi} = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]$ to the time derivative of a visual feature $\dot{\mathbf{s}}$ that can be expressed as stated in eq. (2.2).

In this thesis, we will use the simplest image feature, which is point features. By differentiating eq. (2.3) using the quotient rule, the derivative of the feature in the image plane can be obtained as shown in eq. (2.5).

$$\begin{cases} \dot{x} = \dfrac{\dot{X}Z - X\dot{Z}}{Z^2} = \dfrac{\dot{X} - x\dot{Z}}{Z} \\ \dot{y} = \dfrac{\dot{Y}Z - Y\dot{Z}}{Z^2} = \dfrac{\dot{Y} - y\dot{Z}}{Z} \end{cases} \tag{2.5}$$

Then, assuming the feature is rigidly fixed in the scene, rigid-body kinematics is used to get the relationship with ${}^{o}\boldsymbol{\xi}$ as defined in eq. (2.6) and the cross product is explicitly written in eq. (2.7).

$$\dot{\mathbf{P}} = -\mathbf{v} - \boldsymbol{\omega} \times \mathbf{P} \tag{2.6}$$

$$\begin{cases} \dot{X} = -v_x - \omega_y Z + \omega_z Y \\ \dot{Y} = -v_y - \omega_z X + \omega_x Z \\ \dot{Z} = -v_z - \omega_x Y + \omega_y X \end{cases} \tag{2.7}$$

By combining eqs. (2.3), (2.5) and (2.7), the image feature dynamics can be obtained as follows:

$$\begin{cases} \dot{x} = -\dfrac{v_x}{Z} + \dfrac{x\, v_z}{Z} + x\, y\, \omega_x - (1 + x^2)\, \omega_y + y\, \omega_z, \\ \dot{y} = -\dfrac{v_y}{Z} + \dfrac{y\, v_z}{Z} + (1 + y^2)\, \omega_x - x\, y\, \omega_y - x\, \omega_z. \end{cases} \tag{2.8}$$

Finally, the interaction matrix for point features can be defined in eq. (2.9).

$$\mathbf{L_s} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{2.9}$$

A single point feature gives only 2 equations. To fully observe a 6-DOF camera, at least 3 point features are needed so that the interaction matrix has full rank. We can do this by defining $\mathbf{s}$ into a feature vector $\mathbf{s} = (s_1, s_2, \dots, s_n)$ and stacking the interaction matrix:

$$\mathbf{L_s} = \begin{bmatrix} \mathbf{L_{s_1}} \\ \mathbf{L_{s_2}} \\ \vdots \\ \mathbf{L_{s_n}} \end{bmatrix} \tag{2.10}$$

After obtaining $\mathbf{L}_s$, combining eqs. (2.1) and (2.2), we obtain that the error derivative can be written as in eq. (2.11), where $\mathbf{L}_e = \mathbf{L}_s$.

$$\dot{\mathbf{e}} = \mathbf{L_e}\, {}^o\boldsymbol{\xi} \tag{2.11}$$

Assuming ${}^o\boldsymbol{\xi}$ to be the input to the robot controller and trying to achieve, for example, an exponential decrease of the error, we can derive it in the eq. (2.12).

$${}^o\boldsymbol{\xi} = -\lambda \mathbf{L_e^+ e} \tag{2.12}$$

Where $\mathbf{L_e^+}$ is the Moore-Penrose pseudo-inverse of $\mathbf{L}_s$. However, in real application, it is impossible to get the $\mathbf{L}_e$ or $\mathbf{L}_e^+$ perfectly. Therefore, $\widehat{\mathbf{L}_e^+}$ is used as the symbol for the approximation of the $\mathbf{L}_e$ and $\mathbf{L}_e^+$.

$$^{o}\boldsymbol{\xi} = -\lambda\widehat{\mathbf{L}_e^{+}}\mathbf{e} \tag{2.13}$$

2.1.2 PBVS

PBVS, also known as 3D visual servoing, minimizes the error between the current and desired camera pose for defining $\mathbf{s}$ (eq. (2.14)). The pose of the camera is typically reconstructed using a Perspective-n-Point (PnP) algorithm for 2D-3D correspondence or, in RGB-D systems or 3D-3D correspondence methods, by registering a live point cloud to a CAD model using ICP.

Because the pose is reconstructed in 3D, PBVS requires accurate camera intrinsic parameters so that 2D image features can be correctly projected and a geometric model of the target can establish 2D-3D correspondences. 3D-3D correspondence methods will also have inaccurate 3D points if the camera is not well calibrated [23].

The visual feature is usually represented as a translational and rotational vector in 3D space, making the induced 3D trajectory more predictable, often resembling a 3D straight line. Nevertheless, the object might get out of FOV due to the lack of control over the image space. The visual feature vector is defined as

$$\mathbf{s}(t) = \begin{bmatrix} ^{c*}\mathbf{t}_c \\ \theta\mathbf{u} \end{bmatrix} \in \mathbb{R}^6 \tag{2.14}$$

Where $\mathbf{t}$ is the translation and $\theta\mathbf{u}$ is the angle parameterization for the rotation of the camera frame c with respect to the reference frame $c*$. Using this definition, the desired feature value $\mathbf{s}^*$ is simply the zero vector.

2.1.2.1 Interaction Matrix

For the translational part, the derivative of $^{c*}\mathbf{t}_c$ is obtained by rotating the current linear velocity into the desired frame $^{c*}\mathbf{t}_c = \mathbf{R}\mathbf{v}_c$. For the rotation part, the derivative of $\theta\mathbf{u}$ is $\dot{\theta\mathbf{u}} = \mathbf{L}_{\theta\mathbf{u}}\boldsymbol{\omega}_c$, where $\mathbf{L}_{\theta\mathbf{u}}$ is defined in eq. (2.15) below [24].

$$\mathbf{L}_{\theta\mathbf{u}} = \mathbf{I}_3 - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{\operatorname{sinc}\theta}{\operatorname{sinc}^2\frac{\theta}{2}}\right)[\mathbf{u}]_\times^2 \tag{2.15}$$

Putting the translation and rotation blocks together gives the whole 6x6 interaction matrix:

$$\mathbf{L_e} = \begin{bmatrix} \mathbf{R} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix} \tag{2.16}$$

2.1.3 Hybrid VS

Hybrid visual servoing was introduced to combine the strengths of PBVS and IBVS schemes while overcoming their individual drawbacks. There are several approaches to implementing hybrid VS, including the switching schemes, partitioned approach, and combined Jacobian methods.

2.1.3.1 Switching Schemes

Hybrid switched system treats IBVS and PBVS as two complementary subsystems along with a discrete switching controller that decides when to run each of them [25, 18]. The main goal is to leverage IBVS's pixel-level accuracy and PBVS's fast global convergence while avoiding their failure modes. An intuitive way to define the subsystem function is by monitoring two quadratic Lyapunov functions online, which can be described as follows:

$$V_I(t) = \frac{1}{2}\|\mathbf{e}_I(t)\|^2, \quad V_P(t) = \frac{1}{2}\|\mathbf{e}_P(t)\|^2 \tag{2.17}$$

where $V_I(t)$ and $V_P(t)$ define the IBVS and PBVS Lyapunov function, respectively. While running the PBVS controller, the system will switch to the IBVS controller whenever the IBVS Lyapunov function is higher than a maximum acceptable feature error ($V_I(t) > \gamma_I$). It will switch back to the PBVS controller when the PBVS Lyapunov function is higher than a maximum acceptable pose error ($V_P(t) > \gamma_P$). The parameter γ_I and γ_P are user-defined constants that are chosen conservatively based on task geometry. In practice, these thresholds may be adjusted empirically to balance performance and robustness.

2.1.3.2 Partitioned Approaches

This method uses a decoupled system between the translation and rotation motion. IBVS-style feedback regulates the three translational DOF, while a PBVS-style term drives the three rotational DOF. Hence, this method is usually referred to as 2.5D VS. In [26], the authors use (i) image-plane coordinates of one point and (ii) the log-depth of that point to control translation, and the angle-axis extracted from the homography between current and desired images

to control rotation. Because the translation interaction matrix is triangular, the three translation equations are virtually uncoupled, and the rotation error is expressed in 3D space, this decoupling provides straight-line camera motions and smooth feature trajectories even with large initial displacements.

2.1.3.3 Combined Jacobian Methods

Instead of choosing different **s** for each DOF, a weighted combination of IBVS and PBVS interaction matrices can be constructed as shown in eq. (2.18) below:

$$\mathbf{L}_{\text{hybrid}} = W_{\text{I}}\mathbf{L}_{\text{IBVS}} + W_{\text{P}}\mathbf{L}_{\text{PBVS}} \tag{2.18}$$

Where W_I and W_P are diagonal weight matrices that satisfy $W_I + W_P = 1$ and can be varied smoothly over time, depth, or feature visibility, enabling continuous transition from pure IBVS to pure PBVS and back. Because $\mathbf{L}_{\mathbf{hybrid}}$ remains a full 6x6 full-rank matrix (assuming the original L's are full-rank) and each weight is strictly positive, $\mathbf{L}_{\mathbf{hybrid}}$ remains non-singular, so the classical pseudo-inverse controller in eq. (2.13) can be applied without modification.

2.2 Point Clouds

A 3D Point cloud is a collection of points $\mathbf{P} = \{\mathbf{p}_i\}_{i=1}^{N}$ where each $p_i = [x_i, y_i, z_i]^\top \in \mathbb{R}^3$. Combining these points together describe the visible geometry of a scene. In this project, point clouds generated by an RGB-D camera are used to estimate the pose of the docking station, which is later refined and employed by the MPC. Raw point clouds are usually noisy and therefore preprocessing techniques are required to filter the noise and outliers.

2.2.1 Down-Sampling

RGB-D cameras are capable of producing low-resolution point clouds of 640 x 480, resulting in 307,200 points. Processing every point can significantly increase computational resources since some operations (e.g., thresholding) have O(n) complexity, and more complex algorithms (like K-nearest-neighbor filtering) have O(nk).

One method to reduce the number of points in a point cloud while retaining the information is to use voxel grid downsampling. It is performed by using an octree to segment the point cloud into multiple cube regions (voxels). All the

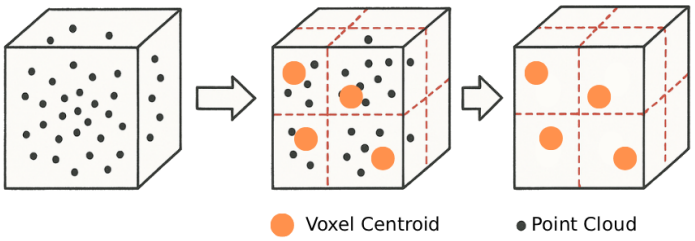

Figure 2.2: 3D voxel grid downsampling process schematic[27].

points in each voxel will be reduced into the one center-of-mass point cloud, resulting in a smaller-sized point cloud but still precise enough to work with.

2.2.2 Plane Segmentation with RANSAC

Plane segmentation is an important part for preprocessing point clouds, especially in indoor environment where planar structures such as walls, floors, and tables are abundant. Planar structures are able to be computed by depth information using depth sensors like Time-of-Flight (TOF) sensors that generate 3D point clouds in real-world coordinate [28].

Plane segmentation methods can be performed with 3 approaches: edge-based, normal-based, and RANSAC segmentation. Edge-based segmentation has the fastest computation time but the worst accuracy. Normal-based segmentation has proven more reliable but slower than edge based method. RANSAC can provide more robust plane segmentation with the highest time consumption [29, 30, 31].

A plane in 3D space can be represented as eq. (2.19) or in normalized form eq. (2.20), where $n(a,b,c)$ is unit normal vector, $p(x,y,z)$ is any point on the plane and d is the distance from origin to plane.

$$ax + by + cz + d = 0 \tag{2.19}$$

$$\mathbf{n}^\top \mathbf{p} + d = 0, \quad \|\mathbf{n}\| = 1 \tag{2.20}$$

$$\mathbf{n} = \frac{(\mathbf{p}_b - \mathbf{p}_a) \times (\mathbf{p}_c - \mathbf{p}_a)}{\|(\mathbf{p}_b - \mathbf{p}_a) \times (\mathbf{p}_c - \mathbf{p}_a)\|}, \quad d = -\mathbf{n}^\top \mathbf{p}_a \tag{2.21}$$

By randomly selecting minimum number of points in the scene (3 non-collinear points to define a plane), the candidate plane parameters can be obtained eq. (2.21). Then, we find the inlier using eq. (2.22) for every other

points in the point cloud. A point is an inlier if the is $\delta_i < \tau$, where τ is the threshold. After N iterations, the plane model will be selected with the highest number of inliers [30].

$$\delta_i = |\mathbf{n}^\top \mathbf{p}_i + d| \tag{2.22}$$

2.3 Pose Estimation

2.3.1 Correspondence-based Methods

2.3.1.1 2D and 3D Descriptors

Correspondence-based pose estimation methods rely on defining geometrical relationships between a known object model and observed data (e.g., image features or 3D points) to recover the rigid transform $\mathbf{T} = [\mathbf{R}|\mathbf{t}] \in SE(3)$ between the sensor (camera) and the object. The main difficulty therefore shifts from solving for $\mathbf{T}$ itself to establishing reliable correspondences. The pose can be recovered analytically or with a few iterations when a set of at least three or four non-coplanar correspondences is available, and additional correspondences could be used to increase robustness towards noise and outliers.

The core approach is divided into three steps. First, extracting local descriptors around keypoints. To find the key descriptors, Scale Invariant Feature Transform (SIFT) [32] and Speeded-Up Robust Features (SURF) [33] are widely used for RGB images. Whereas on raw point clouds, 3D descriptors like Fast Point Feature Histograms (FPFH) or Signature of Histograms of OrienTations (SHOT) are preferred. The second step is matching: each descriptor is paired with its nearest neighbor in the model, making a set of correspondences. The third step is solving the pose to recover a 6-DOF rigid body transform that has the most matched pairs.

When only an image is available, 2D descriptors computed in the image plane will be paired to a known 3D model, and the PnP solvers calculate the estimated camera pose. Because reprojection relies on accurate intrinsics, calibration errors disproportionately affect PnP solutions. That is why PnP solvers are usually used together with RANSAC.

Similarly, with RGB-D image, because both the scene and the reference cloud live in the same Euclidean space, correspondences are matched directly in $\mathbb{R}^3$, removing the perspective coupling inherent in 2D imagery. The estimated pose can then be obtained either with a closed-form absolute

orientation solution (e.g., Horn's method), Singular value decomposition (SVD), or by performing a consensus test such as SAC-IA [34].

2.3.1.2 ICP

Iterative Closest Point (ICP) algorithm has been a common technique for 3D point cloud registration since it was introduced in 1992[35]. As a part of correspondence-based methods, ICP formulates alignment by performing iterative procedure to find the correspondences and compute rigid transformation that minimizes a chosen geometric error between 2 partially overlapping point clouds. The optimization repeats until the point clouds are well-aligned. The objective is monotonically non-increasing with each iteration and will eventually converges to a fixed point (generally a local minimum).

We now explain the basic point-to-point ICP formulation. Let $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{N}$ denote the source cloud and $\mathcal{Q} = \{\mathbf{q}_i\}_{i=1}^{M}$ to be the target cloud. The goal is to find the rigid transformation $\mathbf{T} = (\mathbf{R}, \mathbf{t})$ that minimizes alignment error. At iteration k, every transformed point $\mathbf{R_k p_i} + \mathbf{t_k}$ find its closest point in Q (e.g, via kd-tree/octree), yielding correspondence sets of pair $\mathcal{C}_k = (\mathbf{p}_i, \mathbf{q}_{m(i)})$. Next, the alignment step estimates the transformation by minimizing eq. (2.23).

$$(\mathbf{R}_{k+1}, \mathbf{t}_{k+1}) = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{(\mathbf{p}_i, \mathbf{q}_{m(i)}) \in \mathcal{C}_k} \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_{m(i)} \right\|^2 \tag{2.23}$$

A common method to find the optimal $\mathbf{R}$ and $\mathbf{t}$ is by using the SVD method. First the point sets are centered to the centroid of each source $\bar{\mathbf{p}}$ and target $\bar{\mathbf{q}}$ points sets. Then, a cross-covariance matrix is formed as sum of outer products of the centered corresponding pairs as written in eq. (2.24), where $\tilde{\mathbf{p}}_\mathbf{i}$ and $\tilde{\mathbf{q}}_\mathbf{i}$ denotes centered source and target cloud, respectively.

$$\mathbf{H} = \sum_{i=1}^{N} \tilde{\mathbf{p}}_i \tilde{\mathbf{q}}_i^\top \tag{2.24}$$

Performing SVD to the matrix $\mathbf{H}$ gives 3 matrices (eq. (2.25) that can be used to obtain the $\mathbf{R}$ and $\mathbf{t}$. The optimal rotation $\mathbf{R}$ can be obtained with eq. (2.26) and $\mathbf{t}$ with eq. (2.28). Because $\mathbf{R} = \mathbf{V}\mathbf{U}^\top$ can yield an improper rotation (a reflection) when $det(\mathbf{V}\mathbf{U})^\top = -1$, a diagonal correction is inserted to enforce a proper rotation ($\mathbf{S}$).

$$\mathbf{H} = \mathbf{U}\ \mathbf{V}^\top \tag{2.25}$$

$$\mathbf{S} = \text{diag}(1,\ 1,\ \text{sign}(\det(\mathbf{V}\mathbf{U}^\top))) \tag{2.26}$$

$$\mathbf{R}_{k+1} = \mathbf{V}\mathbf{S}\mathbf{U}^\top \tag{2.27}$$

$$\mathbf{t_{k+1}} = \bar{\mathbf{q}} - \mathbf{R_{k+1}}\bar{\mathbf{p}} \tag{2.28}$$

Finally, checking the convergence is required to know if another iteration is required or not by checking the Mean Squared Error (MSE) of the tranfsormed point cloud.

2.3.1.3 GICP

While standard ICP minimizes a point-to-point Euclidean error with a closed-form SVD update, Generalized Iterative Closest Point (GICP) [36] can be viewed as a continuous generalization of ICP that adapts its weighting per correspondence using local surface covariances and minimizes a Mahalanobis-weighted error. Point-to-point, point-to-plane, and plane-to-plane arise as limiting cases of this single objective; GICP does not explicitly choose a mode, but its local covariances make it behave like the most appropriate variant across different parts of the scene. GICP can improve stability, widens the basin of convergence, and reduces iteration to reach local minimum.

The general working principle is similar with point-to-point, starting with finding correspondences for the transformed source points. Then, GICP estimate a 3×3 covariance that captures local surface shape from a k-NN principal component analysis (PCA). Let $\mathbf{U} = [\mathbf{t}_1\ \mathbf{t}_2\ \mathbf{n}]$ be the orthonormal basis (two tangents $\mathbf{t}_1, \mathbf{t}_2$ and normal $\mathbf{n}$). A common construction is

$$\mathbf{C} = \mathbf{U}\,\text{diag}(\sigma_t^2,\ \sigma_t^2,\ \sigma_n^2)\,\mathbf{U}^\top + \epsilon\mathbf{I}, \tag{2.29}$$

with $\sigma_n^2 \ll \sigma_t^2$ and a small $\epsilon > 0$ for conditioning. Denote by $\mathbf{C}_{p,i}$ and $\mathbf{C}_{q,i}$ the covariances at $\mathbf{p}_i$ and $\mathbf{q}_{m_k(i)}$, respectively.

Instead of minimizing the sum of squared Euclidean distances eq. (2.23), GICP minimizes Mahalanobis-weighted residuals using per-point covariances:

$$
\begin{aligned}
(\mathbf{R}_{k+1}, \mathbf{t}_{k+1}) &= \arg\min_{\mathbf{R},\mathbf{t}} \sum_{(\mathbf{p}_i, \mathbf{q}_{m_k(i)}) \in \mathcal{C}_k} \mathbf{r}_i^\top \, \mathbf{W}_i(\mathbf{R}) \, \mathbf{r}_i, \\
\mathbf{W}_i(\mathbf{R}) &= \left(\mathbf{C}_{p,i} + \mathbf{R}\mathbf{C}_{q,i}\mathbf{R}^\top\right)^{-1}.
\end{aligned} \tag{2.30}
$$

When $\mathbf{C}_{p,i}$ and $\mathbf{C}_{q,i}$ are isotropic ($\sigma_t^2 = \sigma_n^2$), the objective eq. (2.30) reduces to point-to-point ICP objective.

GICP then substitute the SVD blocks (eqs. (2.24) to (2.26) and (2.28) by linearizing the residuals around current pose $\delta\mathbf{x} = [\delta\boldsymbol{\theta}, \delta\mathbf{t}] \in \mathbb{R}^6$ by solving the Gauss–Newton normal equations:

$$
\Big(\sum_i \mathbf{J}_i^\top \mathbf{W}_i^k \mathbf{J}_i\Big)\delta\mathbf{x} = -\sum_i \mathbf{J}_i^\top \mathbf{W}_i^k \mathbf{r}_i^k, \tag{2.31}
$$

.

The pose is then updated on SE(3) using the linearized δx:

$$
\mathbf{R}_{k+1} = \exp([\delta\boldsymbol{\theta}]_\times)\, \mathbf{R}_k, \qquad \mathbf{t}_{k+1} = \mathbf{t}_k + \delta\mathbf{t}. \tag{2.32}
$$

2.3.1.4 Globally Optimal-ICP (GO-ICP)

Globally optimal-ICP (GO-ICP) [37] performs branch-n-bound (BnB) algorithm over the whole SE(3) space to find a global minimum alignment. It eliminates the need for a good initial estimate by systematically partitioning the rotation and translation domains and bounding the lowest possible error in each partition.

GO-ICP does a global search using BnB for each domain. Rotation space SO(3) is parameterized in a radius-π ball, represented using the angle-axis vector $\mathbf{r}$ and Rodrigues' formula eq. (2.33) below to obtain $\mathbf{R}_\mathbf{r}$.

$$
\mathbf{R}_\mathbf{r} = \exp([\mathbf{r}]_\times) = \mathbf{I} + \frac{[\mathbf{r}]_\times \sin\|\mathbf{r}\|}{\|\mathbf{r}\|} + \frac{[\mathbf{r}]_\times^2 \,(1 - \cos\|\mathbf{r}\|)}{\|\mathbf{r}\|^2} \tag{2.33}
$$

For any sub-cube, GO-ICP calculates how much a point can move within that cube. The rotation and translation uncertainty are defined in eqs. (2.34) and (2.36), with σ_r and σ_t as the radius of the rotation ball and half-size of translation cubes.

$$
\|\mathbf{R}_\mathbf{r}\mathbf{x} - \mathbf{R}_{\mathbf{r}_0}\mathbf{x}\| \le g_r(\mathbf{x}) \tag{2.34}
$$

where $g_r(x)$ equals to:

$$g_r(x) = 2\sin\left(\min(\sqrt{3}\sigma_r/2, \pi/2)\right)\|\mathbf{x}\| \tag{2.35}$$

$$\|(\mathbf{x}+\mathbf{t}) - (\mathbf{x}+\mathbf{t}_0)\| \leq \sqrt{3}\sigma_t \tag{2.36}$$

These inequalities are used to filter the transformed points to be inside a ball centered at $\mathbf{R_{r0}x} + \mathbf{t_0}$, with radius $g_r(\mathbf{x}) + g_t$. Therefore, for each data point $\mathbf{x}_i$, one can obtain the upper and lower error bound of its closest-point distance. The upper bound can be calculated by calculating the closest-point distance at the cell center eq. (2.37) and the lower bound tells that the error can't get lower than subtracting the uncertainty radius eq. (2.38).

$$\bar{e}_i = \mathrm{DT}(\mathbf{R_0}\mathbf{x}_i + \mathbf{t}_0) \tag{2.37}$$

$$\underline{e}_i = \max\left\{\bar{e}_i - (g_r(\mathbf{x}_i) + g_t), 0\right\}. \tag{2.38}$$

Summing squares across points gives L2 error bounds for the whole cell:

$$\bar{E} = \sum_i \bar{e}_i^2, \quad \underline{E} = \sum_i \underline{e}_i^2. \tag{2.39}$$

GO-ICP uses nested BnB to search the best pose estimate effectively by running an outer BnB for rotation, and an inner BnB for translation is built inside each of the outer BnB. Then, those L2 error bounds ($\bar{E}$ and $\underline{E}$) are recomputed at every node in both outer (rotation) and inner (translation) Branches. Whenever a new best (lowest) error is obtained, an ICP will be executed to further boost the process. However, if new error is less than the best error, the branch will be pruned. The iteration will stop after the best error is lower than the MSE threshold.

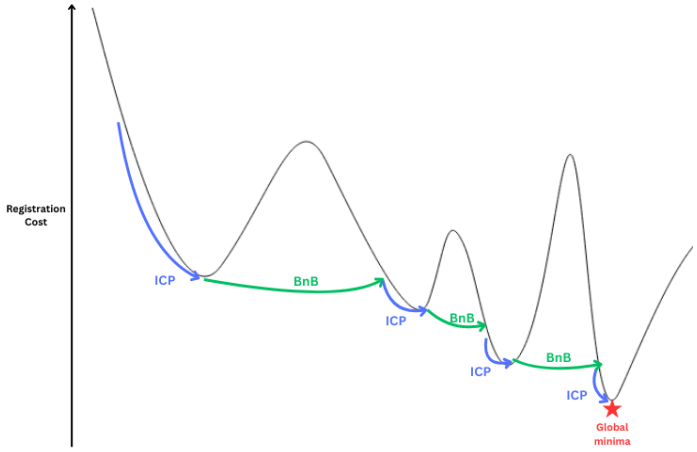

Figure 2.3: ICP used within BnB searches to speed up the process. (source:[37])

Finally, classic ICP minimizes the sum of squared closest-point residuals over all data points, so even a small fraction of outliers can affect the result. GO-ICP provides trimming features by keeping only the best-fitting K points and discarding the rest. The trimming objective is defined in eq. (2.40), where $r \in [0, 1]$ as the trimming percentage (trim factor):

$$E_{\text{trim}}(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{K} e_{(i)}^2, \quad K = (1 - r)N. \tag{2.40}$$

2.3.2 Template-Based Methods

The template-based method follows multiple stage pipeline consist of feature extraction, template matching, and pose inference. In the feature extraction phase, visual descriptors such as edge information, vertices, normal vector, texture patterns, or learned representations from deep neural network from the input image and template database are computed. These templates are pre-rendered or photographed from different viewpoints during an offline phase, using either manual captures or synthetic CAD renders. In the online phase, the template scene are matched with the scene image by maximizing a similarity metric[38].

One of the classical algorithm that utilize this method is LINEMOD [39]. LINEMOD combines silhouette gradient orientations from 2D images and surface normal orientation from depth images to represent object templates. From that, lookup tables are created for similarity indexing. Like other

template-based methods, LINEMOD need quantities of templates to fully cover an object. Nevertheless, object with symmetric structures (e.g, cylinder, cone, etc.) will be seen identical if rotated around the rotational axis. Also, templates captured from nearby viewpoints tend to look almost identical. From these problems, a single object can trigger multiple templates at once. LINEMOD tackles this by choosing every match with a similarity score above a threshold, but it doesn't actually tell how many separate object instances are present [38].

2.3.3 Voting-Based Methods

Voting-based methods are known for their robustness to noisy data, partial occlusion and outliers. The main idea behind this method is to treat pose estimation as a consensus-building process. Features extracted from input images—such as edges, descriptors, keypoints—are used to give votes for candidate pose hypotheses in a parameterized space. The pose with highest accumulation (modes in that space) will be selected as the output. This approach is inspired from Hough transform, originally developed to detect lines and circles in images, which has been adapted to handle more complex transformation for pose estimation.

One well-known algorithm that uses voting-based pose estimation is the extension of classical Hough Transform, that is Generalised Hough transform (GHT) [40]. The GHT enables the object detection by mapping the features from the image space to parameter space representing possible object poses. Each features votes for candidate pose based on a pre-computed reference table, and peaks in the accumulator correspond to the most likely poses.

In learning-based voting methods, a deep neural network is trained to enable each pixel / point predicts the key element of an object should be located in 3D coordinates of canonical keypoints. PVNet [41] takes an image as the input and feed it through a convolutional encoder-decoder network that predicts, for every foreground pixel, a set of unit direction vectors pointing towards six to eight pre-defined keypoints on the object's CAD model. At inference time these vectors are group with a differentiable mean-shift procedure; The resulting density peaks shows the 2D image locations of the keypoints. Finally, a PnP solver construct the full 6-DOF pose. Another learning-based algorithm that uses voting method is PVN3D [42], it extends the same idea as PVNet but only uses RGB-D camera input. Each 3D points predicts 3D vectors pointing toward keypoints in 3D space with a confidence score.

2.4 Quaternion

There are multiple common parameterization of a rigid body's orientation—Euler angles, rotation matrices, and quaternions. In this thesis, we adopt the quaternion representations since the attitude definition of free flyer model in section 3.2 is defined using quaternion operations. Quaternion is a four-parameter unit quantity, we use the scalar-vector (w, x, y, z) convention, defined as follows:

$$\mathbf{q} = \begin{bmatrix} w & x & y & z \end{bmatrix}^\top \tag{2.41}$$

Where $\mathbf{q}_v$ represents the vector part and q_w represents the scalar. Most of the quaternion formulas and definitions are taken from [43].

2.4.1 Main Quaternion Properties

2.4.1.1 Sum

The sum of quaternion has commutative and associative properties. Therefore, it is a straightforward operation. Let $\mathbf{q_1}$, $\mathbf{q_2}$, and $\mathbf{q_3}$ as 2 different quaternions:

$$\begin{aligned} \mathbf{q_1} + \mathbf{q_2} &= \mathbf{q_2} + \mathbf{q_1} \\ \mathbf{q_1} + (\mathbf{q_2} + \mathbf{q_3}) &= (\mathbf{q_1} + \mathbf{q_2}) + \mathbf{q_3} \end{aligned} \tag{2.42}$$

2.4.1.2 Product

Quaternion product is represented with $\otimes$ (p and q are quaternions):

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix}, \tag{2.43}$$

or the operation can also be divided into scalar and vector parts:

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_w q_w - \mathbf{p}_v^\top \mathbf{q}_v \\ p_w \mathbf{q}_v + q_w \mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v \end{bmatrix} \tag{2.44}$$

Since cross-product operations are involved, the quaternion product is not commutative generally. However, if $\mathbf{p}_v \times \mathbf{q}_v = 0$ which indicates if both vector parts are parallel $\mathbf{p}_v \| \mathbf{q}_v$; This includes the special case where one or both vector part is zero (a real quaternion), but it's not limited to it. The

quaternion product is commutative. Furthermore, quaternion product is also associative and distributive over the sum.

2.4.1.3 Conjugate

The quaternion conjugate is defined by

$$\mathbf{q}^* = \begin{bmatrix} q_w \\ -\mathbf{q}_v \end{bmatrix} \tag{2.45}$$

.

2.4.1.4 Norm

The norm of a quaternion is defined by:

$$\|\mathbf{q}\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} \tag{2.46}$$

2.4.1.5 Inverse

The inverse of quaternion can be computed as:

$$\mathbf{q^{-1}} = \mathbf{q}^* / \|\mathbf{q}\|^2 \tag{2.47}$$

However, for unit quaternion, since $\|\mathbf{q}\| = 1$, the inverse is equal to the conjugate itself, $\mathbf{q^{-1}} = \mathbf{q}^*$.

2.4.1.6 Unit Quaternion

A unit quaternion satisfies the normalization constraint and in practice is re-normalized as $\mathbf{q} = \mathbf{q}/\|\mathbf{q}\|$ to control numerical drift. Unit quaternions can be written in the form below [43].

$$\mathbf{q} = \begin{bmatrix} cos\theta \\ \mathbf{u}\, sin\theta \end{bmatrix} \tag{2.48}$$

2.4.2 Rotations in 3D

A (proper) 3D rotation is a linear map that preserves lengths and orientation. The set of all such rotations is the special orthogonal group

$$SO(3) \triangleq \{\, R \in \mathbb{R}^{3\times 3} \mid R^\top R = I,\ \det R = 1 \,\}.$$

Common parameterizations of rotations include the Direction Cosine Matrix (DCM), Euler angles, and unit quaternions.

2.4.3 Direction Cosine Matrix (DCM)

To represents the attitude of a spacecraft, DCM can be used to convert quaternion into rotation matrix since we are dealing with multiple reference frames.

$$\begin{aligned}\mathbf{C}(\mathbf{q}) &= (q_w^2 - \mathbf{q}_v^\top \mathbf{q}_v)\, I_3 + 2\,\mathbf{q}_v \mathbf{q}_v^\top - 2\,q_w\,[\mathbf{q}]^\times \\ &= \begin{bmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y - q_zq_w) & 2(q_xq_z + q_yq_w) \\ 2(q_xq_y + q_zq_w) & 1-2(q_x^2+q_z^2) & 2(q_yq_z - q_xq_w) \\ 2(q_xq_z - q_yq_w) & 2(q_yq_z + q_xq_w) & 1-2(q_x^2+q_y^2) \end{bmatrix}\end{aligned} \tag{2.49}$$

The symbol $[\mathbf{q}]^\times$ on eq. (2.49) means a skew-symmetric operator, defined as follows:

$$[\mathbf{a}]^\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{2.50}$$

This matrix representation allows the cross product of two vectors to be computed as $[\mathbf{a}]^\times \mathbf{b} = \mathbf{a} \times \mathbf{b}$.

2.4.4 Conversions Between Representations

To convert from quaternion to Euler, we can use the following convention.

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}(2(q_wq_x + q_yq_z),\, 1-2(q_x^2+q_y^2)) \\ -\frac{\pi}{2} + 2\operatorname{atan2}\Big(\sqrt{1+2(q_wq_y - q_xq_z)},\, \sqrt{1-2(q_wq_y - q_xq_z)}\Big) \\ \operatorname{atan2}(2(q_wq_z + q_xq_y),\, 1-2(q_y^2+q_z^2)) \end{bmatrix} \tag{2.51}$$

Note that ϕ, θ, ψ are roll, pitch, and yaw angles, respectively.

Conversely, to obtain Quaternion from Euler angles (in Z-Y-X sequence), the following equations can be used:

$$\begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} cos\frac{\phi}{2}\, cos\frac{\theta}{2}\, cos\frac{\psi}{2} + sin\frac{\phi}{2}\, sin\frac{\theta}{2}\, sin\frac{\psi}{2} \\ sin\frac{\phi}{2}\, cos\frac{\theta}{2}\, cos\frac{\psi}{2} - cos\frac{\phi}{2}\, sin\frac{\theta}{2}\, sin\frac{\psi}{2} \\ cos\frac{\phi}{2}\, sin\frac{\theta}{2}\, cos\frac{\psi}{2} + sin\frac{\phi}{2}\, cos\frac{\theta}{2}\, sin\frac{\psi}{2} \\ cos\frac{\phi}{2}\, cos\frac{\theta}{2}\, sin\frac{\psi}{2} - sin\frac{\phi}{2}\, sin\frac{\theta}{2}\, cos\frac{\psi}{2} \end{bmatrix} \tag{2.52}$$

2.4.5 Quaternion Time Derivatives

By representing small perturbations in a vector space, we can derive the time derivative directly from first principles. Let the nominal state to be $\mathbf{q}(t)$ and the perturbed state to be $\tilde{\mathbf{q}} = \mathbf{q}(t + \Delta t)$. By definition of the derivative:

$$\dot{\mathbf{q}}(t) = \lim_{\Delta t \to 0} \frac{\tilde{\mathbf{q}} - \mathbf{q}(t)}{\Delta t} \tag{2.53}$$

Then, with

$$\omega_{\mathcal{L}}(t) \triangleq \frac{d\phi_{\mathcal{L}}(t)}{dt} \triangleq \lim_{\Delta t \to 0} \frac{\Delta\phi_{\mathcal{L}}}{\Delta t}, \tag{2.54}$$

where $\Delta\phi_{\mathcal{L}}$ is a local angular perturbation, corresponds to the angular rates vector $\mathbf{q}$ as follows:

$$\begin{aligned} \dot{\mathbf{q}} &= \lim_{\Delta t \to 0} \frac{\mathbf{q} \otimes \Delta\mathbf{q}_{\mathcal{L}} - \mathbf{q}}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\mathbf{q} \otimes \left(\begin{bmatrix} 1 \\ \Delta\phi_{\mathcal{L}}/2 \end{bmatrix} - \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \right)}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\mathbf{q} \otimes \begin{bmatrix} 0 \\ \Delta\phi_{\mathcal{L}}/2 \end{bmatrix}}{\Delta t} \\ &= \frac{1}{2}\, \mathbf{q} \otimes \begin{bmatrix} 0 \\ \omega_{\mathcal{L}} \end{bmatrix} \end{aligned} \tag{2.55}$$

Then, omega operator $\Omega(\omega)$ can be defined as:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) \triangleq [\boldsymbol{\omega}]_R = \begin{bmatrix} 0 & -\boldsymbol{\omega}^\top \\ \boldsymbol{\omega} & -[\boldsymbol{\omega}]^\times \end{bmatrix} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}, \tag{2.56}$$

making the final form of quaternion derivative to be:

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} \tag{2.57}$$

2.4.6 Distance Between 2 Quaternions

Let $\mathbf{q_1}, \mathbf{q_2} \in \mathbb{H}$ be unit quaternions in scalar–first form that represent 3D orientations. A principled "distance" between orientations is the *geodesic* distance on $SO(3)$, i.e., the minimal rotation angle that maps one attitude to the other. The angle θ of rotation required to get from one orientation to another is given by the formula below:

$$\theta = \arccos\left(2\langle \mathbf{q_1}, \mathbf{q_2}\rangle^2 - 1\right) \tag{2.58}$$

where $\langle \mathbf{q}_1, \mathbf{q}_2 \rangle$ denotes the inner product between two quaternions:

$$\langle \mathbf{q}_1, \mathbf{q}_2 \rangle = q_{1w}q_{2w} + q_{1x}q_{2x} + q_{1y}q_{2y} + q_{1z}q_{2z} \tag{2.59}$$

2.5 Model Predictive Control (MPC)

MPC is a well-known optimization-based control technique that was originally used in petro-chemical process regulation for slow dynamics. However, its defining idea has proved for its accuracy, safety, and robustness in robotics. MPC optimize a finite-horizon model at every sampling instant and apply only the first control input to the system. The process is repeated at the next sampling instant to allow the controller continuously adapt to the true system state until the system has converged to the target state[15]. Unlike classical controllers that treat limits as afterthoughts, MPC incorporates them as algebraic conditions inside the optimization itself. At each sampling instant, the solver searches for a control sequence that minimizes both the cost and satisfies the defined rules/constraints.

A discrete-time nonlinear system can be expressed through equations below:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{2.60}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ and $\mathbf{u}_k \in \mathbb{R}^m$ is the system state and control input at time step k. $f(\mathbf{x}_k, \mathbf{u}_k)$ is the state-transition function. MPC builds upon the principles of linear-quadratic optimal control theory. To understand MPC's foundation, we examine the derivation of optimal control laws using quadratic cost functions. The key distinction between traditional Linear–Quadratic

Regulator (LQR) and MPC lies in the explicit definition of constraints, which were discussed earlier.

The optimization objective is formulated using a cost function that penalizes deviations of the predicted state and control input from their desired references. The optimization problem can be formulated as in eq. (2.61). The terms inside the summation is called stage (running) cost and term at the final horizon called terminal cost. The stage cost shapes behavior along the horizon, while the terminal cost penalizes the end state (often approximating the infinite-horizon cost) and helps with stability.

$$
\begin{aligned}
J_N^* = \min_{x,u} \quad & \sum_{k=0}^{N-1} \left[\hat{\mathbf{x}}_k^T \mathbf{Q} \hat{\mathbf{x}}_k + \hat{\mathbf{u}}_k^T \mathbf{R} \hat{\mathbf{u}}_k \right] + \hat{\mathbf{x}}_N^T \mathbf{P} \hat{\mathbf{x}}_N, \\
\text{s.t.} \\
& \mathbf{x}_{k+1} = f(\mathbf{x_k}, \mathbf{u_k}), \quad \forall k = 0, \dots, N-1, \\
& \mathbf{x}_k \in \mathcal{X}, \quad \forall k = 0, \dots, N-1, \\
& \mathbf{x}_k \in \mathcal{U}, \quad \forall k = 0, \dots, N-1, \\
& \mathbf{x}_0 = \mathbf{x}_{\text{init}}, \\
& \mathbf{x}_N = \mathcal{X}_f
\end{aligned}
\tag{2.61}
$$

Where $\hat{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{x}_r$ is error of current with reference state and $\hat{\mathbf{u}}_k = \mathbf{u}_k - \mathbf{u}_r$ for the control input. $\mathbf{Q}$ and $\mathbf{P}$ are the positive semi-definite weight matrices that penalize the state and input error, and $\mathbf{R}$ is the positive definite weight matrix that penalize the control effort. The defined constraints enforce dynamic feasibility, keep all predicted states and inputs within the constraints $\mathcal{X}$ and $\mathcal{U}$ (defined in eq. (2.62), the initial state will be used as the state on the first time step, and the final step always follows the constraint $\mathcal{X}_f$.

The optimization problem above is solved at every sampling instant. After applying the predicted control input $\mathbf{u_0}^*$ to the system, the horizon recedes and the optimization is repeated with updated state measurements. This is what makes this finite-horizon optimal control into MPC.

$$
\begin{aligned}
\mathcal{X} &= \{\mathbf{x} \in \mathbb{R}^{n_x} : x_{min} \leq \mathbf{x} \leq x_{max}\}, \\
\mathcal{U} &= \{\mathbf{u} \in \mathbb{R}^{n_u} : u_{min} \leq \mathbf{u} \leq u_{max}\}
\end{aligned}
\tag{2.62}
$$

$\mathcal{X}$ and $\mathcal{U}$ are called hard constraints because it must be respected at all times, it is typically used for safety or to to define hardware or critical operating limits. On the contrary, soft constraints can be violated if needed without severe consequences. For example, to change this constraint

$$a^\top z \leq b\,, \tag{2.63}$$

into a soft constraint, we can introduce a non-negative slack variable $\zeta \geq 0$ so the new constraint is modified into

$$a^\top z \leq b + \zeta \tag{2.64}$$

To allow the solver to optimize the slack variable, one can add a positive slack cost $\sigma(s)$ into the cost function. The common cost includes the linear and quadratic cost to give a penalty that increases rapidly with ζ. The cost function for slack variable is defined as follows:

$$\sigma(\zeta) = \frac{1}{2}\zeta^\top Z\zeta + z^\top \zeta. \tag{2.65}$$

2.6 Related work

[44] proposed a two-step vision-based docking system with a hybrid VS to enable an autonomous chaser to mate with a target port. A multi-scale template matcher is used to detect the reflective marker, while a tracker keeps the marker locked in real time. During the coarse-alignment phase, the author applies direct VS: the raw pixel offset of the marker's centroid is mapped through inverse tangent equations to yaw and pitch-rate commands, aligning the camera without estimating depth or interaction matrix. Once the target is aligned, the control switches to IBVS, tracking four retro-reflector points. However, the approach is sensitive to different lighting conditions and requires synthetic templates or markers.

[14] presents *FoundationPose*, a unified framework that uses an RGB-D camera to obtain 6-DOF pose estimation and tracking of novel objects without prior training. It operates in both model-based mode (when a textured CAD model is available) and few-shot model-free mode (when only a small set of reference images is provided). The method bridges these modes using a neural implicit representation for novel-view synthesis, employing a transformer-based architecture with contrastive learning formulation trained on large-scale synthetic data aided by Large language model (LLM).

The pose estimation process first initializes multiple hypotheses uniformly around the object, which are then refined by a learned refinement network. After refinement, a pose selection module predicts their scores and selects the best pose as the output. A key limitation is high computational cost, the

GPU-intensive neural rendering and refinement of hypotheses at every frame make the method unsuitable for development on resource-constrained onboard computers.

Another research by Gans and Hutchinson (2007) [18] proposed a hybrid switched-system VS framework combining PBVS and IBVS to improve stability across a wide range of poses. This method alternates between PBVS and IBVS mode by monitoring two Lyapunov functions—one defined on the image-feature error (IBVS) and one on the pose error (PBVS). The controller switches when either error crosses a user-defined threshold, creating switching surfaces that bring the state to a safe rectangle in a common error space. To mitigate chattering and abrupt switching near the boundaries, they introduced time-varying scalar gains that slow the controller as it approaches a surface. However, the framework inherits common issues associated with Image-Based Visual Servoing (IBVS), such as local minima or singularities in the control law, and it may switch indefinitely, potentially converging to an accumulation point that is not the intended goal.

Chapter 3

System Dynamics

In this chapter, the system dynamics and free-flyer model used in this work are presented. It starts with the definition of the coordinate system and reference frames to describe the robot's motion and interaction with the environment. Then, the mathematical formulation of the 3-DOF free-flyer is described, including its rigid-body dynamics and thruster configuration.

3.1 Coordinate System

This thesis adopts a right-handed ENU convention for most of the reference frames. Reference frames define coordinate representations of vectors, enabling consistent transformations between them.

The global inertial frame, denoted $\mathcal{F}_{map}$, is fixed to the world and serves as the parent of all other frames. The robot base frame $\mathcal{F}_{base}$ is the child frame of map and attached at the centroid of robot's base. The camera link frame $\mathcal{F}_{cam}$ is rigidly mounted at the top of the robot, oriented such that the camera faces backward relative to the base x axis. The transformation in eq. (3.1) describes the pose of the camera frame expressed in the base frame and is used to transform point cloud data to the map frame.

Finally, the camera optical frame $\mathcal{F}_{optical}$ follows the standard machine-vision convention: $+x$ to the right in the image, $+y$ down, and $+z$ along the viewing direction.

$$ {}^{cam}\mathbf{T}_{base} = \begin{bmatrix} -1 & 0 & 0 & -0.1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

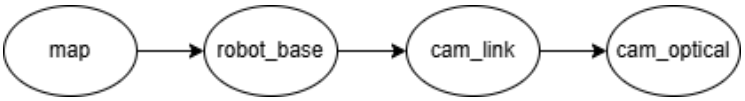

Figure 3.1: simplified TF tree of the system.

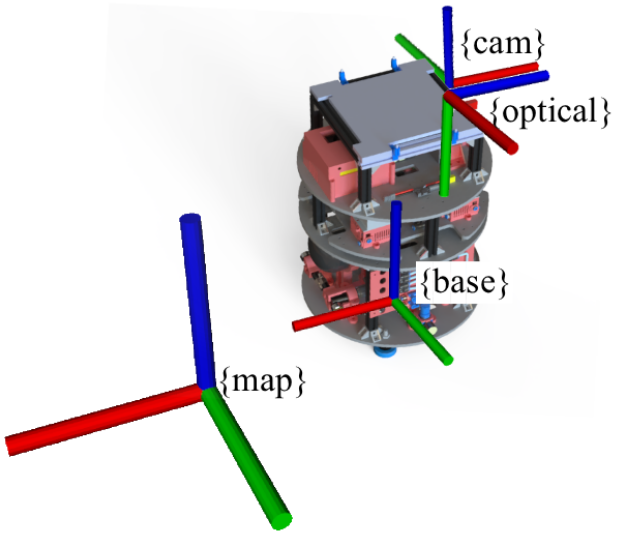

Figure 3.2: Visualization of the coordinate system.

3.2 Free-Flyer Modeling (3-DOF)

The dynamics of the free-flyer robot have to be defined in order to formulate the MPC. The rigid-body dynamics that form the basis for the equations of motion of a thruster-controlled spacecraft are discussed in [45]. In this model, each thruster is assumed to be one-directional, requiring two thrusters in a coaxial configuration to generate bidirectional force along the given axis.

Since this thesis considers a simplified version of the spacecraft constrained to planar motion, we can redefine the dynamics by setting the forces and torques acting on the unused DOF to zero. The planar spacecraft model therefore has 3-DOF—two translational DOF within the plane and one rotational DOF corresponding to yaw about the Z axis. Accordingly, the thruster body force $\mathbf{F}$ and torque $\boldsymbol{\tau}$ simplify into:

$$\begin{aligned} \mathbf{F} &= [F_x \quad F_y \quad 0]^T \\ \boldsymbol{\tau} &= [0 \quad 0 \quad \tau_z]^T \end{aligned} \tag{3.2}$$

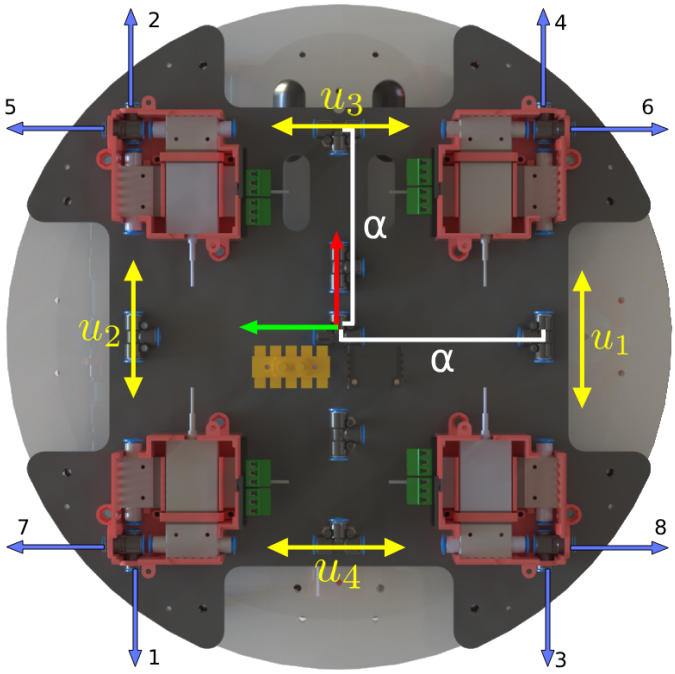

Figure 3.3: Thruster input mapping and configuration of the free-flyer.

Our robot is equipped with eight thrusters, paired into four control inputs such that two opposing thrusters shared a single input ($u_i \in [-1, 1]$), where i denotes the pair index. Defining the four thrust directions to be stacked as columns of the matrix $\mathbf{D} = [\mathbf{d_1} \dots \mathbf{d_4}] \in \mathbb{R}^{2\times4}$, the lever arms are collected as $L = [\mathbf{l_1} \dots \mathbf{l_4}] \in \mathbb{R}^{1\times4}$, α as the distance from robot's COM perpendicular to thruster axis, and the control input as $\mathbf{u} = [\mathbf{u_1} \dots \mathbf{u_4}]^\top$, the net body-frame force is written below:

$$\mathbf{F_{body}} = \mathbf{Du} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{3.3}$$

$$\boldsymbol{\tau} = \mathbf{L}\mathbf{u} = \begin{bmatrix}\tau_z\end{bmatrix} = l_{arm}\begin{bmatrix}1 & -1 & 1 & -1\end{bmatrix}\begin{bmatrix}u_1\\u_2\\u_3\\u_4\end{bmatrix} \tag{3.4}$$

Also, the quaternion now can be represented only with q_w and q_z (refer eq. (3.5)). By that, the DCM ($\mathbf{C}(\psi)$) can also be represented as in eq. (3.6).

$$\mathbf{q} = \begin{bmatrix}w & x & y & z\end{bmatrix}^\top = \begin{bmatrix}cos(\psi/2) & 0 & 0 & sin(\psi/2)\end{bmatrix}^\top \tag{3.5}$$

$$\mathbf{C}(\psi) = \begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix} \tag{3.6}$$

3.2.1 Rigid-Body Dynamics

The dynamics of the spacecraft is derived using the Newton-Euler method, which obtains the rigid-body equations of motion by applying Newton's law to translation and Euler's equation to rotation. The state vector for the free flyer consist of four components eq. (3.7)

$$\mathbf{x} = \begin{bmatrix}\mathbf{p}\\ \mathbf{v}\\ \mathbf{q}\\ \boldsymbol{\omega}\end{bmatrix} \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{S}^3 \times \mathbb{R}^3 \tag{3.7}$$

The translational kinematics of a rigid body must be described. $\mathbf{p}$ represents the position vector expressed in the inertial frame. The kinematics are written as follows:

$$\dot{\mathbf{p}} = \mathbf{v}. \tag{3.8}$$

The kinetic equation is also modeled with respect to the inertial frame. The time derivative of linear momentum equals the net external force according to Newton's Second Law, as shown below:

$$m\dot{\mathbf{v}} = \mathbf{F}. \tag{3.9}$$

To use eq. (3.9), the force at body-frame must be converted into the inertial frame using DCM:

$$\mathbf{F} = \mathbf{C}(\psi)^\top \mathbf{F}_{\text{body}}. \tag{3.10}$$

Combining eqs. (3.3), (3.9) and (3.10) yields the final form of $\dot{\mathbf{v}}$:

$$\dot{\mathbf{v}} = \frac{1}{m}\mathbf{C}(\psi)^\top \mathbf{D}\mathbf{u}. \tag{3.11}$$

To define the attitude kinematics $\dot{\mathbf{q}}$, we can use the quaternion derivation from section 2.4.5. The attitude kinetics can be derived also from Newton's second law of motion for a rigid-body:

$$\boldsymbol{\tau} = \mathbf{J}\dot{\boldsymbol{\omega}} + [\boldsymbol{\omega}]^\times \times (\mathbf{J}\boldsymbol{\omega}) \implies \quad \dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(\boldsymbol{\tau} - [\boldsymbol{\omega}]^\times \mathbf{J}\omega). \tag{3.12}$$

Where $\mathbf{J}$ is the inertia and is the angular velocity. The complete equations of motion for the thruster-controlled spacecraft are presented below.

$$\begin{aligned}
\dot{\mathbf{p}} &= \mathbf{v} \\
\dot{\mathbf{v}} &= \frac{1}{m}\mathbf{C}(\psi)^\top \mathbf{D}\mathbf{u} \\
\dot{\mathbf{q}} &= \frac{1}{2}\boldsymbol{\Omega}(\omega)\mathbf{q} \\
\dot{\boldsymbol{\omega}} &= \mathbf{J}^{-1}(\boldsymbol{\tau} - [\boldsymbol{\omega}]^\times (\mathbf{J}\omega))
\end{aligned} \tag{3.13}$$

Chapter 4

Docking Station Design

A docking station was designed to support the experimental docking scenario with the dimension of 280mmx140mmx650mm (refer fig. 4.2). The structural frame uses V-Slot aluminum extrusions, and the fiducial pattern follows the NASA Circular Navigation Feature (CNF) that is scaled down to 60% of its original dimension to fit the workspace. [46]. To facilitate the final capture, a conical guide is mounted on both the station and free-flyer. Each cone equipped with an embedded neodymium magnet, manufactured using a 3D printer. This provides passive self-alignment and holding force at contact.

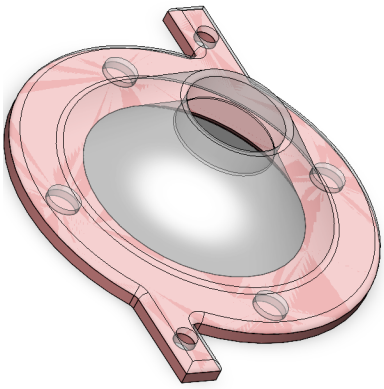

Figure 4.1: Section view of the docking station conical guide.

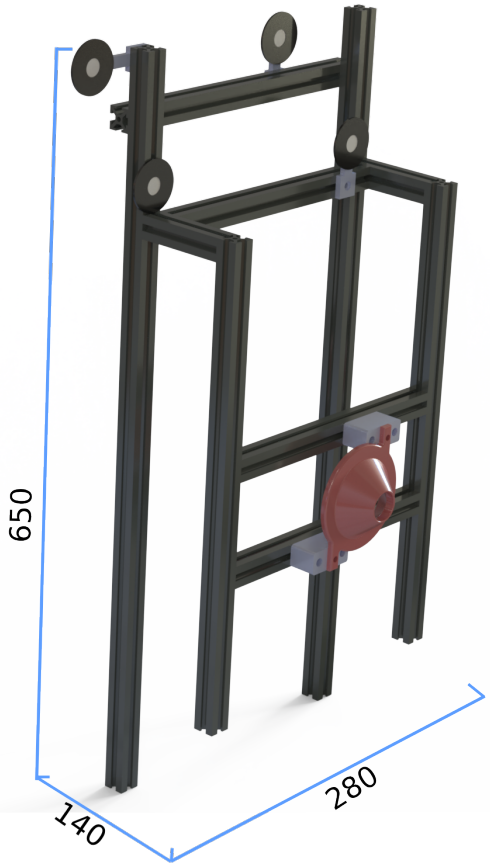

Figure 4.2: Docking station design.

Chapter 5

System Design

This chapter shows the end-to-end pipeline used in both simulation and hardware experiment. The loop combines perception (pose estimation and feature extraction), an MPC with hybrid VS switching (MPVS), and the robot dynamics. Each block will further explained in this chapter. Figure fig. 5.1 represents the control architecture.

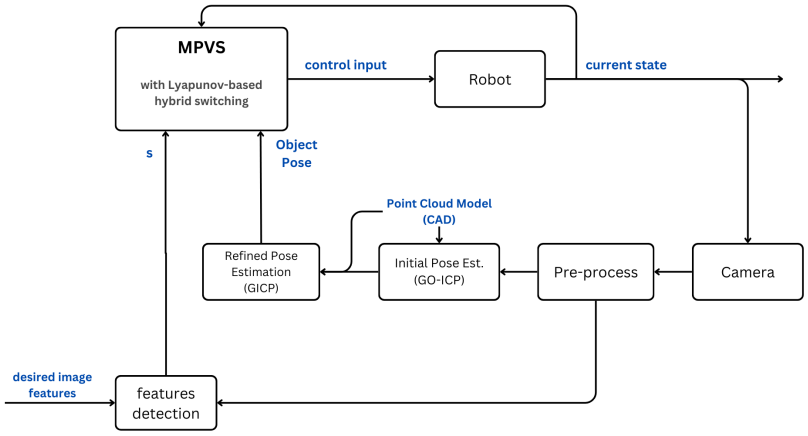

Figure 5.1: Vision-based hybrid control architecture. Blue labels denote signals.

5.1 Point cloud Preprocessing

This thesis relies on the Point Cloud Library (PCL)[47] as the primary toolkit for point-cloud processing, including voxel downsampling, registration, and related utilities.

It is a best practice to always downsample a point cloud before manipulating it to reduce computation while preserving geometric fidelity. The parameter *leaf _size* represents the resolution of the grid—trades accuracy for speed. Larger voxels yield fewer points and faster runtime, while smaller voxels retain finer detail at higher cost. In this thesis, the voxel-grid leaf size is set to 0.012.

After downsampling the point cloud, plane segmentation is performed to detect the floor and walls. Using PCL's RANSAC segmentation with an orientation constraint, we can obtain the plane equation by checking its normal if it is either perpendicular or parallel to an axis. The plane equation is then used to filter the point cloud outside the region of interest.

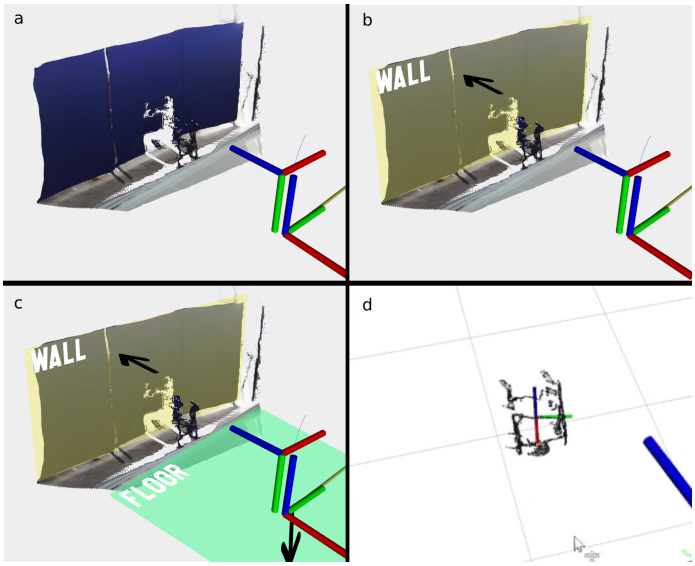

Figure 5.2: (a) Raw point cloud; (b) Detecting wall plane; (c) Detecting floor plane; (d) Filtered point cloud.

5.2 Cascaded ICP for Pose Tracking

This section presents the pose estimation approach adopted in this work: a two-stage, correspondence-based pipeline that combines GICP for high-accuracy local alignment (refinement) with GO-ICP for global alignment. We choose this combined strategy from all available methods discussed in section 2.3 because it yields a balanced trade-off between computational cost and accuracy, which is suitable for our platform and operating conditions.

5.2.1 Initial Pose Estimation

ICP is a local optimization algorithm, and its performance relies heavily on the initialization. If the initial guess is poor, ICP may converge to a local minimum. To address this, GO-ICP is utilized to provide a robust initial estimate for the subsequent local ICP.

Before applying GO-ICP, both the target and scene point clouds are normalized to fit in $[-1, 1]^3$ space by centering each cloud at the origin and scaling them with the maximum radius among the point clouds. The following pseudo-code summarizes the registration process between the data cloud $\mathcal{D}$ and the model cloud $\mathcal{M}$.

Algorithm 1 Pose estimation with GO-ICP registration.

Require: Point clouds $\mathcal{D} = \{\mathbf{d}_i\} \subset \mathbb{R}^3$, $\mathcal{M} = \{\mathbf{m}_j\} \subset \mathbb{R}^3$
Ensure: $\mathbf{T}_{\text{final}} \in SE(3)$ in original coordinates

$\mathbf{c}_D \leftarrow \text{mean}(\mathcal{D})$; $\quad \mathbf{c}_M \leftarrow \text{mean}(\mathcal{M})$
$\mathcal{D}^c \leftarrow \{\, \mathbf{d}_i - \mathbf{c}_D \,\}$; $\quad \mathcal{M}^c \leftarrow \{\, \mathbf{m}_j - \mathbf{c}_M \,\}$ ▷ center
$r_D \leftarrow \max_{\mathbf{d} \in \mathcal{D}^c} \|\mathbf{d}\|_2$; $\quad r_M \leftarrow \max_{\mathbf{m} \in \mathcal{M}^c} \|\mathbf{m}\|_2$
$s \leftarrow 1/\max(r_D, r_M)$; $\quad \mathcal{D}^n \leftarrow s\,\mathcal{D}^c$; $\quad \mathcal{M}^n \leftarrow s\,\mathcal{M}^c$ ▷ normalize
$(\mathbf{R}^*, \mathbf{t}^*) \leftarrow \text{GO-ICP}(\mathcal{D}^n, \mathcal{M}^n)$
$\mathbf{T}_{\text{data}} \leftarrow \begin{bmatrix} I_3 & -c_D \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$; $\quad \mathbf{T}_{\text{model}} \leftarrow \begin{bmatrix} I_3 & \mathbf{c}_M \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$
$\mathbf{S} \leftarrow \begin{bmatrix} s\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$; $\quad S^{-1} \leftarrow \begin{bmatrix} \frac{1}{s}\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$
$\mathbf{T}_{\text{norm}} \leftarrow \begin{bmatrix} R^* & \mathbf{t}^* \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$
$\mathbf{T}_{\text{final}} \leftarrow \mathbf{T}_{\text{model}}\,\mathbf{S}^{-1}\,\mathbf{T}_{\text{norm}}\,\mathbf{S}\,\mathbf{T}_{\text{data}}$
return $\mathbf{T}_{\text{final}}$

Here, $\mathbf{c}_X$ and $\mathcal{X}^c$ represents the center coordinate and centered point cloud X, r_X represents the maximum distance from the origin, and s the scaling factor. After registration, The estimated transformation is de-normalized to recover the final transformation $\mathbf{T}_{\text{final}}$ in the original coordinate frame.

GO-ICP builds a distance map of the entire scene and searches for the model pose within a bounding box. If the scene contains points that are far outside the object's region of interest, the bounding box becomes large, increasing computational cost, and the closest-point cost may be biased by the clutter, resulting in misalignment. Cropping the scene to the ROI and removing planar structures or outliers mitigate these issues and improve robustness.

The source code of GO-ICP from [37] (C++) was integrated into our system by developing a ROS2 wrapper. As a part of this integration, we implemented the transformation conversion procedure described in section 5.2.1 within the wrapper to process the original point clouds.

5.2.1.1 Pose Transformation Verification

GO-ICP might return an alignment that is orientation-flipped when the data are partially occluded or the object is near-symmetric (refer fig. 5.3). To guard against these cases, we validate the estimated pose by checking axis consistency with respect to the camera frame.

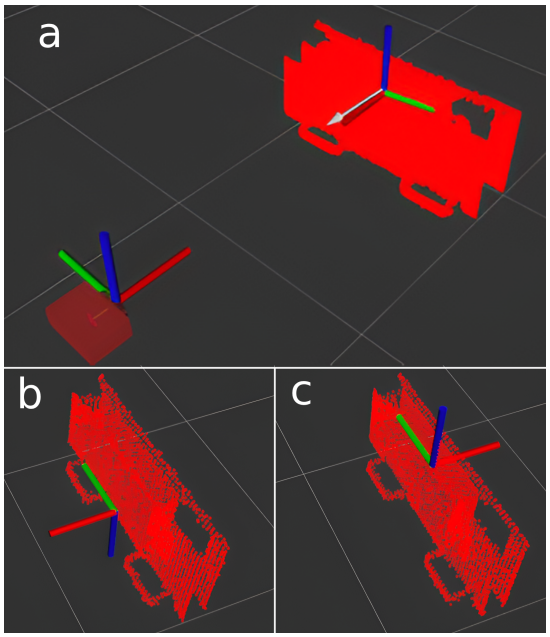

Figure 5.3: (a) Ground truth pose;(b) Flipped orientation (on x axis);(c) Flipped orientation(on y axis).

Let $\mathbf{R}^*$ be the rotation from object to camera. By defining unit axes in the camera frame as $\hat{\mathbf{x}}_{\text{obj}}$ and $\hat{\mathbf{z}}_{\text{obj}}$, we can compute the angles using eq. (5.2):

$$\hat{\mathbf{x}}_{\text{obj}} = \mathbf{R}^* \hat{\mathbf{e}}_x, \quad \hat{\mathbf{z}}_{\text{obj}} = \mathbf{R}^* \hat{\mathbf{e}}_z \tag{5.1}$$

$$\alpha_x = \arccos(\hat{\mathbf{x}}_{\text{cam}} \cdot \hat{\mathbf{x}}_{\text{obj}}), \quad \alpha_z = \arccos(\hat{\mathbf{z}}_{\text{cam}} \cdot \hat{\mathbf{z}}_{\text{obj}}) \tag{5.2}$$

where the dot products are clamped to $[-1,1]$ to avoid numerical errors. We then perform two constraints that encode our expected viewpoint:

(i) the object's x-axis should face the camera, so $\alpha_x \geq 90^\circ + \delta_x$ (with a small margin δ_x, e.g., 5°);

(ii) the object should be upright relative to the camera, so $\alpha_z \leq \theta_z$ (e.g., 20°–30°).

If either test fails, we treat the result as invalid and retry GO-ICP.

5.2.2 GICP Refinement

The transformation result from GO-ICP is then fed into GICP as the initial guess to refine the estimate and update it in a higher frequency. GO-ICP takes around 2000-5000ms while GICP takes 50ms to get the result. The refined transformation is then used as a pose reference in the MPC (will further explained in section 5.3). The full pose tracking pipeline is described below:

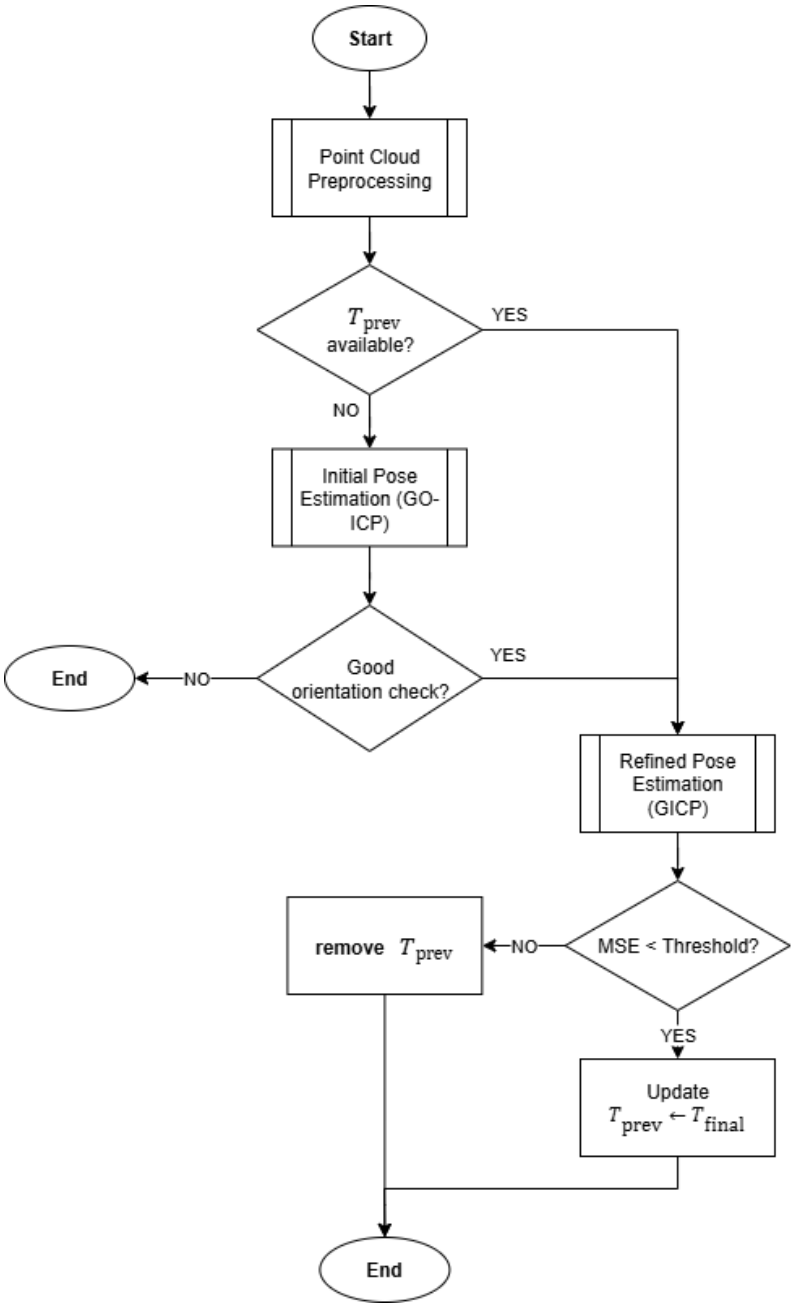

Figure 5.4: Online pose tracking pipeline.

5.3 MPVS Design

This section builds on an existing reference-tracking MPC implementation for the Autonomy Testbed for Multi-purpose Orbiting Systems (ATMOS) free

flyer [48]. This controller reliably follows pose setpoints under actuation and state constraints.

We extend this controller with a hybrid VS scheme for docking with two phases. In the Alignment Phase, the controller uses PBVS to rapidly reduce large pose error and align the robot with the docking station. As robot approaches close range and become approximately perpendicular to the station, the objective dynamically shift to the Docking Phase, prioritizing the image features s (IBVS), yielding high-rate, pixel-level accuracy right up to convergent. Figure 5.5 visualizes the two-step docking phases.

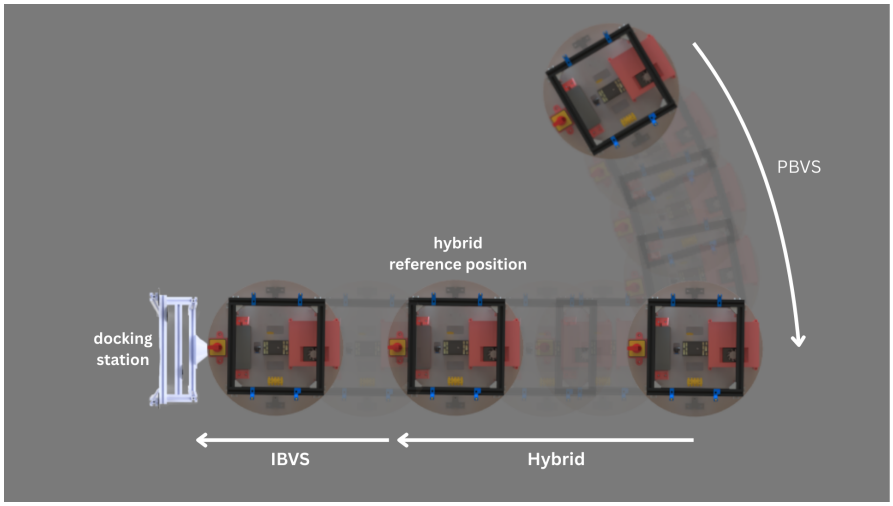

Figure 5.5: Two-step docking phases.

This scheduling exploits the complementary strength of both methods while mitigating their weaknesses. PBVS is efficient when the full pose is observable, but the accuracy tends to degrades near the target when only a partial point cloud is visible. In contrast, IBVS remains responsive and precise in close proximity, yet from farther distance it can suffer from poor interaction-matrix conditioning and convergence to local minima. We will discuss the hybrid VS strategy in this section below.

5.3.1 FOV Constraint Inequality

Since MPC supports inequality constraints, we enforced a FOV bearing constraint that preserves a minimum alignment between camera link axis and the centroid of the object. let $\mathbf{p_k}$ and $\mathbf{q}_k$ to be the robot pose in the inertial frame, $\mathbf{p}_o$ the (estimated) target position, $\mathbf{q}_{k,rot}$ the rotated quaternion,

${}^{cam}\mathbf{q}_{base}$ the $base \rightarrow cam$ rotation (180°yaw on z-axis), and $\mathbf{c}_{cam}$ the camera link axis. The line-of-sight vector $\mathbf{r}_k^{cam}$ is described using quaternion product and DCM in eq. (5.3).

$$\begin{aligned} {}^{cam}\mathbf{q}_{base} &= [cos(\pi/2), 0, 0, sin(\pi/2)] = [0, 0, 0, 1] \\ \mathbf{q}_{k,rot} &= \mathbf{q}_k \otimes {}^{cam}\mathbf{q}_{base} \\ \mathbf{r}_k^{cam} &= \mathbf{C}(\mathbf{q}_{k,rot})^\top (\mathbf{p}_o - \mathbf{p}_k). \end{aligned} \tag{5.3}$$

Using a dot product between two vectors, the instantaneous bearing angle θ_k between $\mathbf{c}_{\mathbf{cam}}$ and $\mathbf{r}_k^{cam}$ obeys

$$\cos\theta_k = \frac{\mathbf{c}_{cam}^\mathsf{T} \mathbf{r}_k^{cam}}{\|\mathbf{r}_k^{cam}\|}. \tag{5.4}$$

With the camera optical axis aligned with the camera-frame x-axis, the camera link axis $\mathbf{c}_{cam}$ is the x axis of $\mathbf{r}_k^{cam}$. Thus, the dot product simplifies into $\mathbf{r}_{x,k}^{cam}$. Requiring $\theta_k \leq \theta_{max}$ (camera half-FOV, or a tighter margin) is equivalently enforced by the inequality:

$$g(\mathbf{x}_k) \;:\; \cos(\theta_{\max}) \|\mathbf{r}_k^{cam}\| - \mathbf{r}_{x,k}^{cam} \;\leq\; 0, \tag{5.5}$$

which geometrically keeps $\mathbf{r}_k^{cam}$ inside a cone of half-angle θ_{max} about $\mathbf{c}_{\mathbf{cam}}$. This form avoids $arccos(.)$ and is numerically well behaved for $\theta_{max} \in [0, 90°]$.

Because temporary loss of visibility can be unavoidable (e.g, obstacles, re-planning, aggressive references), the constraint is soften using slack variable eq. (5.6). fig. 5.6 shows the illustration of of FOV constraint design (top-view).

$$g(\mathbf{x}_k) \;\leq\; \zeta_k, \qquad \zeta_k \geq 0. \tag{5.6}$$

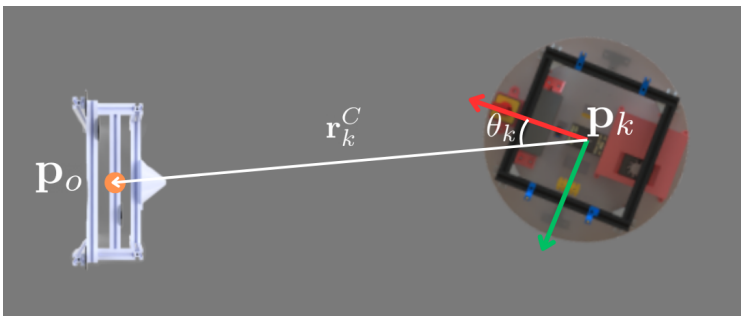

Figure 5.6: Visibility constraint illustration.

5.3.2 Image Dynamics

To simplify notation for this section, the reference frames are abbreviated as follows: map m, base b, camera c, and optical o. From section 2.1, the features' dynamic equation ($\dot{\mathbf{s}} = \mathbf{L}_{\mathbf{s}}^{\mathbf{o}}\boldsymbol{\xi}$) are integrated into the MPC by augmenting the state with the features $\mathbf{s}$. Because $\mathbf{L}_s$ is defined with the camera twist expressed in the optical frame ${}^{o}\boldsymbol{\xi}$, while the free-flyer dynamics are specified in the inertial (map) frame, twists must be converted between frames using the adjoint representation of the homogeneous transform:

$$
\begin{aligned}
{}^{b}\mathbf{T}_{m} &= \begin{bmatrix} {}^{b}\mathbf{R}_{m} & \mathbf{p}_{k} \\ 0 & 1 \end{bmatrix} \text{(map} \rightarrow \text{base)}, \\
{}^{c}\mathbf{T}_{b} &= \begin{bmatrix} {}^{c}\mathbf{R}_{b} & \mathbf{t}_{bc} \\ 0 & 1 \end{bmatrix} \text{(base} \rightarrow \text{cam)}, \\
{}^{o}\mathbf{T}_{c} &= \begin{bmatrix} {}^{o}\mathbf{R}_{c} & 0 \\ 0 & 1 \end{bmatrix} \text{(cam} \rightarrow \text{opt)}.
\end{aligned}
\tag{5.7}
$$

The transformation chain from map to optical is:

$$
{}^{o}\mathbf{T}_{m} = {}^{o}\,\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{b}\,{}^{b}\mathbf{T}_{m}. \tag{5.8}
$$

After obtaining the transformation, the adjoint that maps twists can be calculated using following equation:

$$
\mathrm{Ad}(\mathbf{T}) = \begin{bmatrix} \mathbf{R} & 0 \\ [\mathbf{t}]^{\times}\mathbf{R} & \mathbf{R} \end{bmatrix} \tag{5.9}
$$

Finally, the twist from map to optical can be expressed as in eq. (5.10) and used to update the feature dynamics eq. (5.11)

$$
{}^{o}\boldsymbol{\xi} = \mathrm{Ad}({}^{m}\mathbf{T}_{o})\,{}^{m}\boldsymbol{\xi}. \tag{5.10}
$$

$$
\dot{\mathbf{s}} = h(\mathbf{s}, \mathbf{u}) = \mathbf{L}_{\mathbf{s}}\,(\mathrm{Ad}({}^{m}\mathbf{T}_{o})\,{}^{\mathbf{m}}\boldsymbol{\xi}) \tag{5.11}
$$

5.3.3 Hybrid VS Switching Strategy

Employing a hard switch between PBVS and IBVS modes works in simple scenario. But, in real deployments it could creates discontinuities in both the objective and resulting control law. This can lead to jerky robot motion, control chattering, reduced tracking accuracy, and even temporary feature loss

at the moment of switching. To address these issues, a dynamic switching strategy is adopted, where the controller is scheduled smoothly between PBVS and IBVS based on real-time metrics such as pose confidence (PBVS reliability), interaction-matrix conditioning and feature count or stability (IBVS reliability), or the Lyapunov derivative.

[49] implemented probabilistic framework of each individual control law by designing a risk-based weighting to update the probability to tackle each other's weaknesses. A gaussian distribution is used to construct the weight. Another research by [18] (as mentioned in section 2.6) uses Lyapunov function to toggle the control law between PBVS and IBVS.

We proposed a dynamic switching between PBVS and IBVS by comparing their predicted Lyapunov decay rates and converting them into weights w_p and w_s, which blend the two objectives terms in MPC. The weights are computed externally at each control step and kept fixed over the prediction horizon.

5.3.3.1 Lyapunov Function Candidates

To define quadratic Lyapunov candidate for PBVS, the position error e_p can be used :

$$V_p = \frac{1}{2}\mathbf{e_p^T P e_p}, \quad \mathbf{P} \succ 0 \tag{5.12}$$

differentiating the equation yields the PBVS Lyapunov decay rate:

$$\dot{V}_p = \mathbf{e_p^T P}\dot{\mathbf{e}}_p = \mathbf{e_p^T P v}. \tag{5.13}$$

Similarly, the IBVS Lyapunov decay rate can be constructed from the image features error $\mathbf{e}_s$ and it's dynamics $\dot{\mathbf{s}}$:

$$\begin{aligned} V_s &= \frac{1}{2}\mathbf{e_s^T S e_s}, \quad \mathbf{S} \succ 0 \\ \dot{V}_s &= \mathbf{e}_s^T \mathbf{S}\dot{\mathbf{s}} \\ &= \mathbf{e}_s^T \mathbf{S}\,(\mathbf{L}_s\, {}^o\boldsymbol{\xi}) \end{aligned} \tag{5.14}$$

5.3.3.2 Dynamic Weight Design

Negative Lyapunov values indicate faster decay, while positive values are undesirable as they correspond to energy growth. Therefore, the weight update is restricted to depend only on the negative values. Using the modified softmax function, where $k > 0$ tunes sensitivity and epsilon be a very small value to avoid division by zero $0 < \epsilon \ll 1$, the weights are defined in eq. (5.15) below.

$$w_p = \begin{cases} \dfrac{e^{-k\dot{V}_p}}{e^{-k\dot{V}_p} + e^{-k\dot{V}_s} + \varepsilon}, & \dot{V}_p \leq 0, \\ 0, & \dot{V}_p > 0, \end{cases} \qquad (5.15)$$

$$w_s = 1 - w_p.$$

For comparison, a linear alternative is formulated using a ratio method, as shown in eq. (5.16). Since both $\dot{V}_p$ and $\min\{0, \dot{V}_s\}$ are non-positive, the resulting fraction is always bounded in the range $[0, 1]$.

$$w_p = \begin{cases} \dfrac{\dot{V}_p}{\dot{V}_p + \min\{0, \dot{V}_s\} + \varepsilon}, & \dot{V}_p \leq 0, \\ 0, & \dot{V}_p > 0, \end{cases} \qquad w_s = 1 - w_p \qquad (5.16)$$

To construct dynamic weighting matrices for the states, we define two sets of weights for the PBVS and IBVS modes, since their objectives differ. PBVS priorities pose error and does not account for image features, whereas IBVS emphasizes feature error and robot velocity. The weights are updated using linear scaling scheme, as shown in eq. (5.17). Here, $\mathbf{Q}_p$, $\mathbf{Q}_s$, and $\mathbf{S}_s$ denotes the PBVS weight, IBVS weight, and feature weight, respectively. The terminal costs $\mathbf{P}_{\text{hybrid}}$ and $\mathbf{W}_{\text{hybrid}}$ are defined in the same way, but with an amplified constant.

$$\begin{aligned} \mathbf{Q}_{\text{hybrid}} &= w_p \mathbf{Q}_p + w_s \mathbf{Q}_s \\ \mathbf{S}_{\text{hybrid}} &= w_s \mathbf{S}_s \end{aligned} \qquad (5.17)$$

5.3.4 MPC Formulation

we extended the reference-tracking MPC by adding feature dynamics $s_0 \dots s_7$ into the system state since 4 point features are placed on the docking station. For clarity, the feature dynamics s are described separately from the system state x when expressing the weighting explicitly. Acados [50] is used to solve the nonlinear optimal control problems. The cost function is formulated as a nonlinear least-squares problem, and the system dynamics are discretized using the Explicit Runge–Kutta (ERK) method.

By using the feature dynamics (eq. (5.11)), dynamic weighting (eqs. (5.15) and (5.16), and incorporating FOV constraint (eqs. (5.5) and (5.6), the MPVS optimization problem is formulated as follows:

$$
\begin{aligned}
J^* \;=\; & \min_{\{\mathbf{x}_k,\mathbf{u}_k,\mathbf{s}_k,\zeta_k\}} \sum_{k=0}^{N-1} J_k(\mathbf{x}_k,\mathbf{u}_k,\mathbf{s}_k) + J_N(\mathbf{x}_N,\mathbf{s}_N) \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = f(\mathbf{x_k},\mathbf{u_k}), \quad \forall k = 0,\dots,N-1, \\
& \mathbf{s}_{k+1} = h(\mathbf{s_k},\mathbf{u_k}), \quad \forall k = 0,\dots,N-1, \\
& g(\mathbf{x}_k) \le \zeta_k, \\
& \mathbf{u}_{\min} \le \mathbf{u}_k \le \mathbf{u}_{\max}, \quad \forall k = 0,\dots,N-1, \\
& \mathbf{x}_0 = \mathbf{x}_{\text{init}}, \quad \mathbf{s}_0 = \mathbf{s}_{\text{init}}, \\
& \mathbf{x}_N = \mathbf{x}_G, \quad \mathbf{s}_N = \mathbf{s}_d
\end{aligned}
\tag{5.18}
$$

with:

$$
\begin{aligned}
J_k(\mathbf{x}_k,\mathbf{u}_k,\mathbf{s}_k) &= \|\mathbf{x}_k - \mathbf{r}_k\|^2_{\mathbf{Q}_{\text{hybrid}}} + \|\mathbf{u}_k\|^2_{\mathbf{R}} + \|\mathbf{s}_k - \mathbf{s}_d\|^2_{\mathbf{S}_{\text{hybrid}}} + \sigma(\zeta_k), \\
J_N(\mathbf{x}_N,\mathbf{s}_N) &= \|\mathbf{x}_N - \mathbf{r}_N\|^2_{\mathbf{P}_{\text{hybrid}}} + \|\mathbf{s}_N - \mathbf{s}_d\|^2_{\mathbf{W}_{\text{hybrid}}} + \sigma(\zeta_N).
\end{aligned}
\tag{5.19}
$$

Where $f(.)$ and $h(.)$ is the free-flyer and image feature dynamics, $\|\mathbf{x}\|_{\mathbf{P}}$ equal to the weighted vector norm ($\sqrt{\mathbf{x}^\top \mathbf{P}\mathbf{x}}$), $\mathbf{r}_T$ is the reference state at timestep T, and $\mathbf{s}_d$ is the reference image features.

5.3.5 Penalty Weights Tuning

The performance of the proposed MPC depends on the choice of penalty weights in the stage and terminal costs ($\mathbf{Q}_{hybrid}, \mathbf{P}_{hybrid}, \mathbf{S}_{hybrid}, \mathbf{W}_{hybrid}$). Since these weights directly influence the trade-off between tracking accuracy, control effort, and visual feature alignment, they were tuned using trial-and-error procedure. The tuning process differed depending on the VS mode:

5.3.5.1 Position-Based Mode

The same weights as in the reference-tracking MPC design were used as both modes aim to minimize the free-flyer state error relative to a desired trajectory. Consequently, the penalty weights related to image features ($\mathbf{S}_p$ and $\mathbf{W}_p$) were set to zero.

5.3.5.2 Image-Based Mode

In this mode, the error is defined in the image plane and measured in pixel units. The weights $\mathbf{S}_s$ and $\mathbf{W}_s$ were scaled down to prevent the solver from being too aggressive in minimizing the feature error. This adjustment ensures that the controller balances feature alignment with overall system stability, rather than prioritizing pixel-level accuracy at the expense of smooth control inputs.

5.3.5.3 Hybrid Mode

To tune the hybrid mode, the Lyapunov weights and the softmax temperature parameter k were adjusted to control the smoothness of the blending process. Additionally, the stage and terminal cost penalty weights were gradually shifted from PBVS to IBVS weight. The MPC parameters that are used for the system are described in table 5.1.

Parameter	**Value**
N_t (horizon number)	25
$\mathbf{R}$	diag(40,40,40,40)
PBVS mode	
$\mathbf{Q}_p$	diag(10,10,10,1e2,1e2,1e2,1e4,2e2,2e2,2e2,2e2)
$\mathbf{P}_p$	$30\mathbf{Q_p}$
$\mathbf{S}_p$	0
$\mathbf{W}_p$	0
IBVS mode	
$\mathbf{Q}_s$	diag(0,0,0,3e3,3e3,3e3,0,16e3,16e3,16e3)
$\mathbf{P}_s$	$30\mathbf{Q}_s$
$\mathbf{S}_s$	60e-4
$\mathbf{W}_s$	$60\mathbf{S}_s$

Table 5.1: MPC cost function parameters.

Chapter 6

Experiments and Results

In order to validate the proposed system, we followed a two-stage process. First, simulation in PX4 software in the loop (SITL) with Gazebo to proof the theory and tune the parameters. Second, hardware in the loop (HITL) experiments on the planar free-flyer ATMOS in the ITRL space lab. Both stages use the same pose estimation pipeline and MPC formulation, only differ in the penalty weights, and GO-ICP parameters. The work of this thesis is done in ROS (jazzy) framework.

We evaluated two components: (i) pose estimation, and (ii) hybrid VS (PBVS–IBVS) behavior. For software simulation, the ground truth for the docking station pose comes from Gazebo. For HITL validation, the ground truth is obtained using the motion capture (mocap) system by placing passive markers on the docking stations. To evaluate the pose estimation accuracy, 3 randomized robot poses with the docking station visible in the camera has been selected. For each pose, computed the docking station pose w.r.t the map frame using GO-ICP (initial pose) and GICP (refined pose).

6.1 Software Simulation

6.1.1 Simulation Environment

Gazebo Harmonic is used to set up a simulation environment. A simple scenario requires the COLLADA file (.dae) of the robot, and other objects such as obstacles, walls, and floor). Then, a virtual RGB-D camera needs to be prepared to publish the 2D image and point cloud. ros_gz_bridge package is used to bridge communication between gazebo and ROS2. For briding communication between PX4 SITL and ROS2, uXRCE-DDS is used as a DDS

bridge. The snapshot of the Gazebo world used for the system development is shown in fig. 6.1.

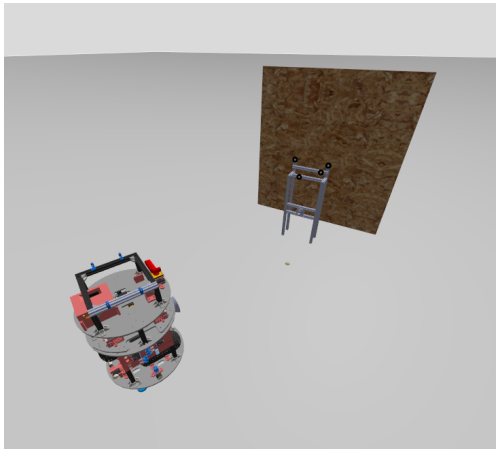

Figure 6.1: Simulation environment used for development.

6.1.2 Simulation Results

The table 6.1 shows the position and orientation error, runtime (for initial alignment), and success count. The position error is calculated using Euclidean norm between estimated and ground-truth translations. The rotational error is calculated using geodesic angle between estimated and ground-truth orientations (section 2.4.6). It can be seen that GICP able to improve the orientation error from the initial estimated pose. The position error doesn't get any lower because the initial estimate was accurate enough. Additionally, the normalized errors which is the relative error with respect to the distance from the robot to the ground truth were lower than 5%.

GO-ICP Mean Position Error [m]	GO-ICP Mean Attitude Error [°]	GICP Mean Position Error [m]	GICP Mean Attitude Error [°]	Success Rate [%]	Distance to Station [m]	Normalized Error [%]	GO-ICP Mean Runtime [s]
0.045	0.534	0.048	0.292	100	2.08	2	3.51
0.051	0.618	0.053	0.890	100	1.58	3	5.13
0.061	0.954	0.050	0.879	80	2.10	2	3.02

Table 6.1: Pose estimation and runtime summary (SITL).

For the hybrid VS results, we ran 5 experiments and highlight the run with best docking time. Multiple metrics are evaluated to analyze the result. First, the individual features' errors and the features' motion are presented below.

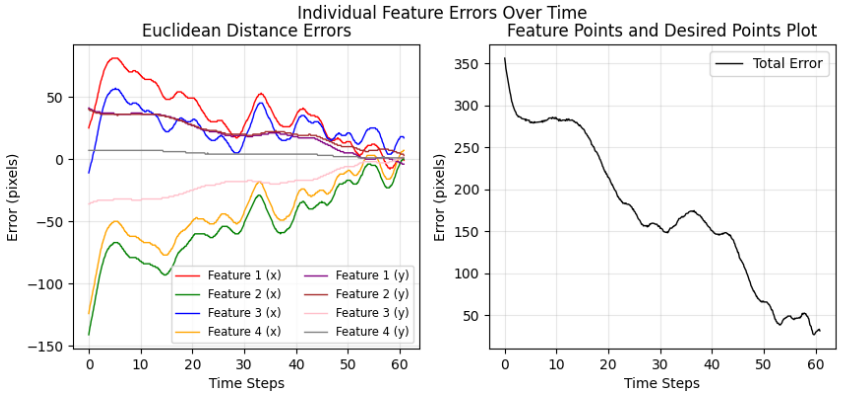

Figure 6.2: Individual feature error (SITL): Discrete mode.

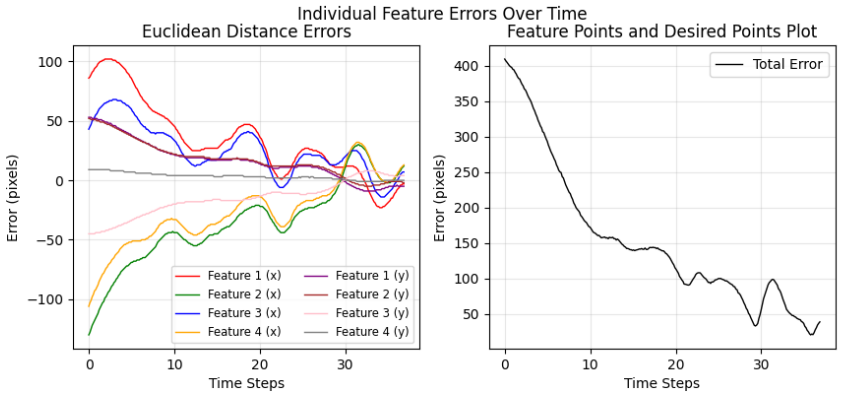

Figure 6.3: Individual feature error (SITL): Ratio mode.

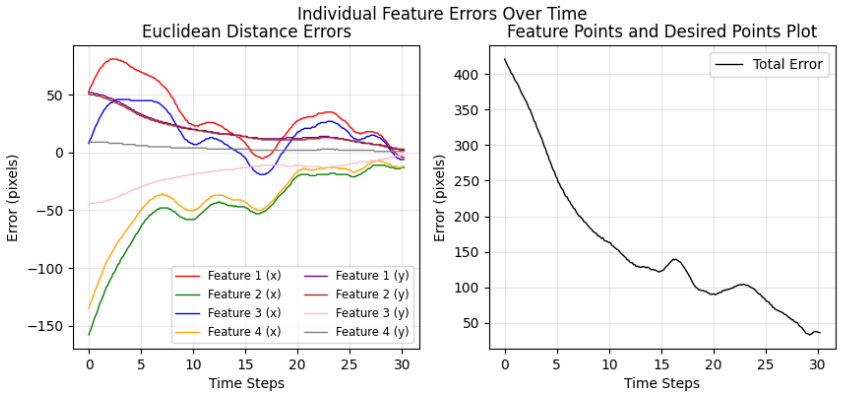

Figure 6.4: Individual feature error (SITL): Softmax mode.

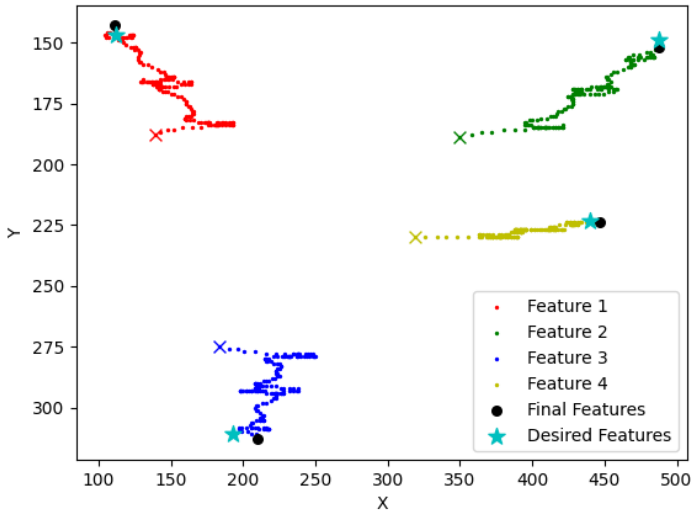

Figure 6.5: Image-feature motion in image plane (SITL): Discrete mode.

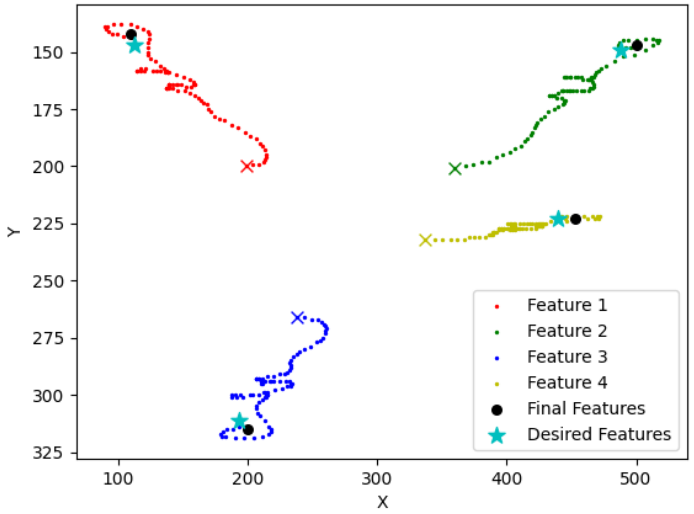

Figure 6.6: Image-feature motion in image plane (SITL): Ratio mode.

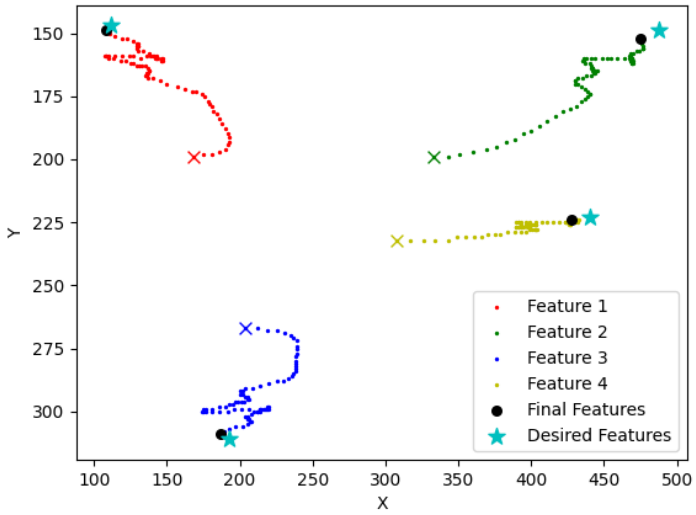

Figure 6.7: Image-feature motion in image plane (SITL): Softmax mode.

Next, the weights and Lyapunov's derivative over time are described in the following.

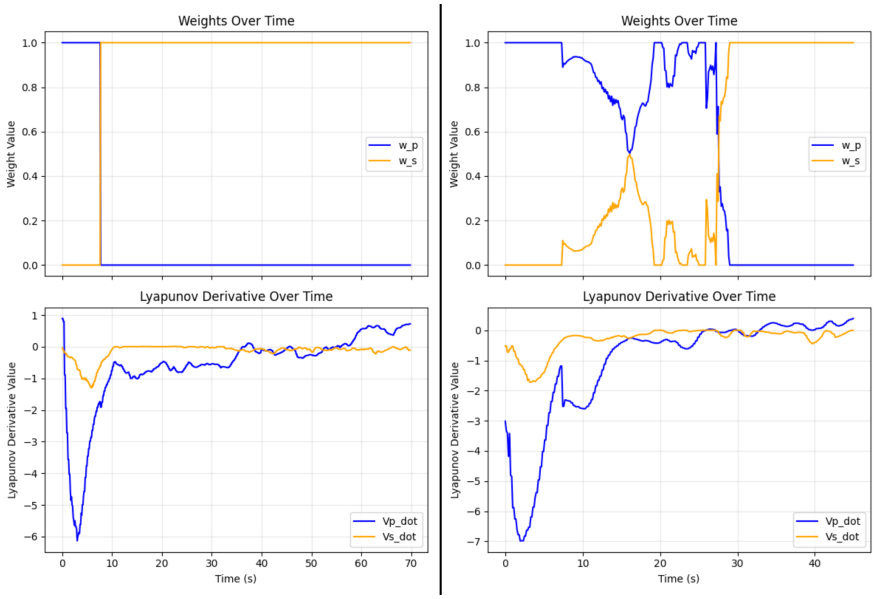

Figure 6.8: VS weights and Lyapunov Derivative (SITL). Left: Discrete mode. Right: Ratio mode.

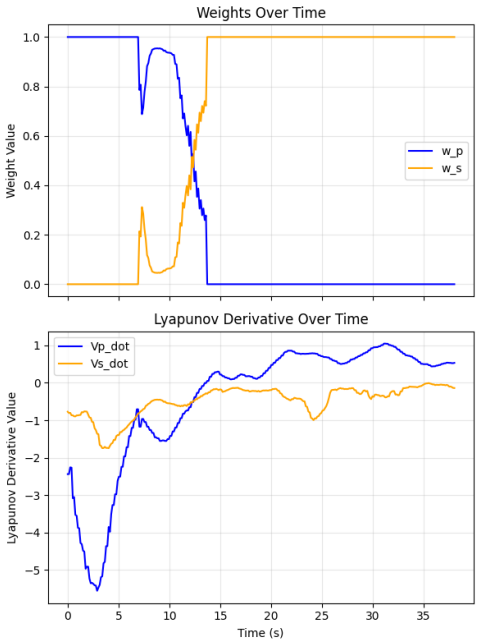

Figure 6.9: VS weights and Lyapunov Derivative (SITL): Softmax mode.

Table 6.2 presents the steady-state error (SSE) of the image features and free-flyer pose in simulation. The SSE values are generally higher than those observed in hardware experiments, as the simulation does not model the passive self-alignment magnet. Without this physical self-alignment effect, the robot must rely on active control to maintain its pose, making it harder to maintain the feature error within a small threshold.

Switching Mode	Image Features SSE [px]	Robot Position SSE [m]	Robot Attitude SSE [°]
Discrete	19.21	0.0307	2.2277
Ratio	20.39	0.0121	0.9332
Softmax	19.57	0.0198	2.2625

Table 6.2: Steady-state error metrics across switching modes (SITL).

Finally, we evaluated the required time from the aligning phase until the free flyer has been docked:

Switching Mode	Average Full Docking Duration [s]	Average Hybrid VS Duration [s]
Discrete	89.86	80.85
Ratio	72.77	63.84
Softmax	45.31	36.67

Table 6.3: Docking duration (SITL).

From the results above, we can see the softmax weighting decreases the feature error more smoothly than the discrete and ratio modes while achieving the shortest docking time. This likely happens because the softmax mode switch smoothly from PBVS to IBVS as soon as it promises a larger negative Lyapunov rate. Furthermore, because the reference hybrid position is chosen to be the intermediate distance between the docking station and the aligned position, it helps bring the robot closer to the docking station, improving feature visibility and the conditioning of the interaction matrix; IBVS then performs the final, precise alignment. Using the discrete mode, however, is more prone for IBVS to converge in the local minima since the initial features error is too big.

6.2 Hardware Validation

ATMOS is a planar free-flyer operating on air-bearings, enabling near-frictionless motion in $SE(2)$. The platform consists of a PX4 flight controller (PX4 6X) for low-level actuation with an onboard computer (Jetson Orin NX). An RGB-D camera (ZED Mini) is mounted on top of the robot. A mocap system provides pose measurements to obtain vehicle's odometry that are fused with the PX4 Extended Kalman filter (EKF) estimator.

6.2.1 Hardware Experiment Results

The same procedure were executed to evaluate the pose estimation error. The normalized pose errors relative to the robot distance to the station were less than 1%. The presented figures below are the experiment result with best docking time.

GO-ICP Mean Position Error [m]	GO-ICP Mean Attitude Error [°]	GICP Mean Position Error [m]	GICP Mean Attitude Error [°]	Success Rate [%]	Distance to Station [m]	Normalized Error [%]	GO-ICP Mean Runtime [s]
0.063	4.373	0.063	4.593	80	1.61	4	7.53
0.078	6.871	0.090	5.785	100	0.91	10	2.72
0.071	6.126	0.079	6.283	80	1.50	5	4.60

Table 6.4: Pose estimation and runtime summary (HITL).

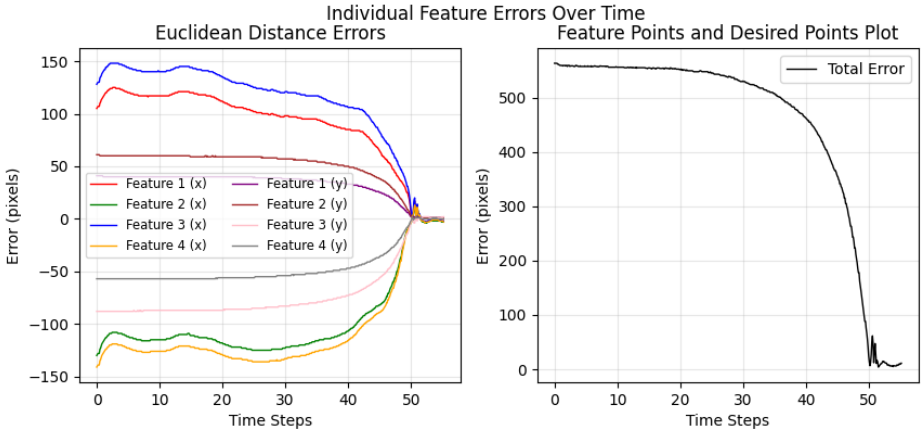

Figure 6.10: Individual features error (HITL): Discrete mode.

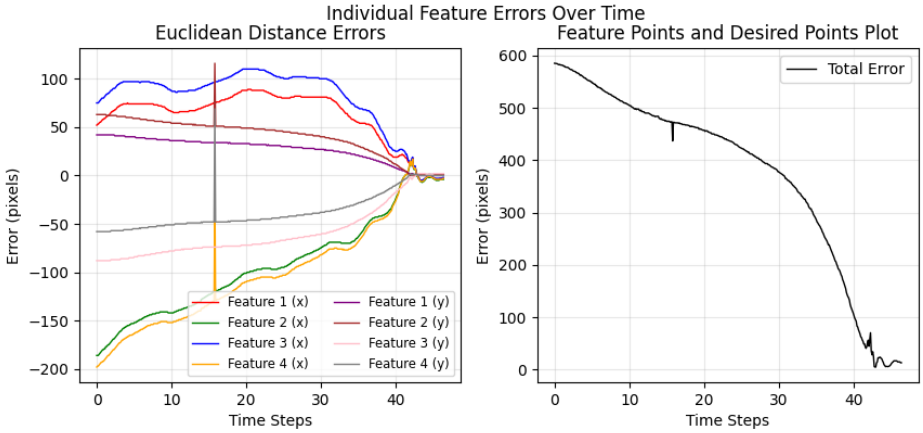

Figure 6.11: Individual features error (HITL): Ratio mode.

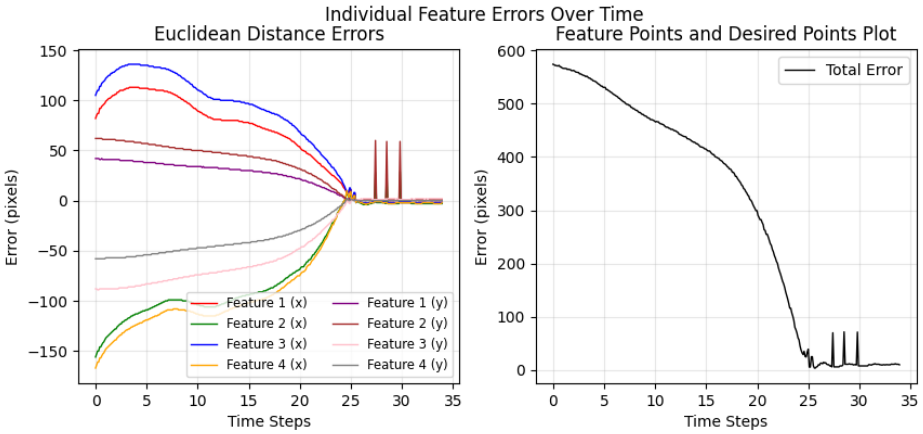

Figure 6.12: Individual features error (HITL): Softmax mode.

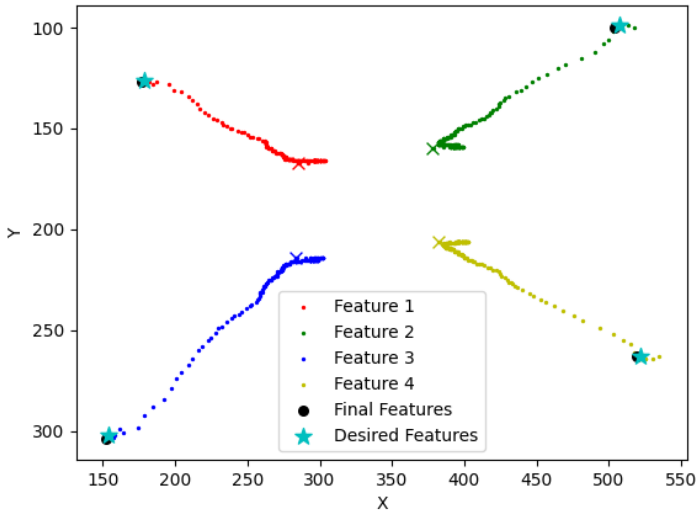

Figure 6.13: Image-feature motion in image plane (HITL): Discrete mode.

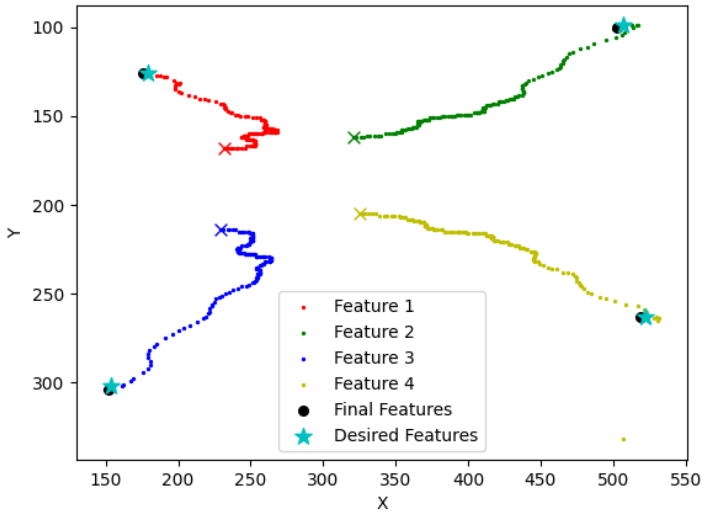

Figure 6.14: Image-feature motion in image plane (HITL): Ratio mode.

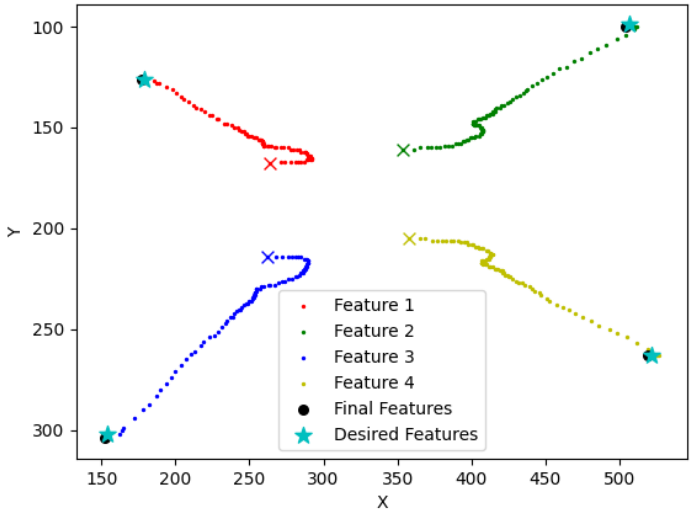

Figure 6.15: Image-feature motion in image plane (HITL): Softmax mode.

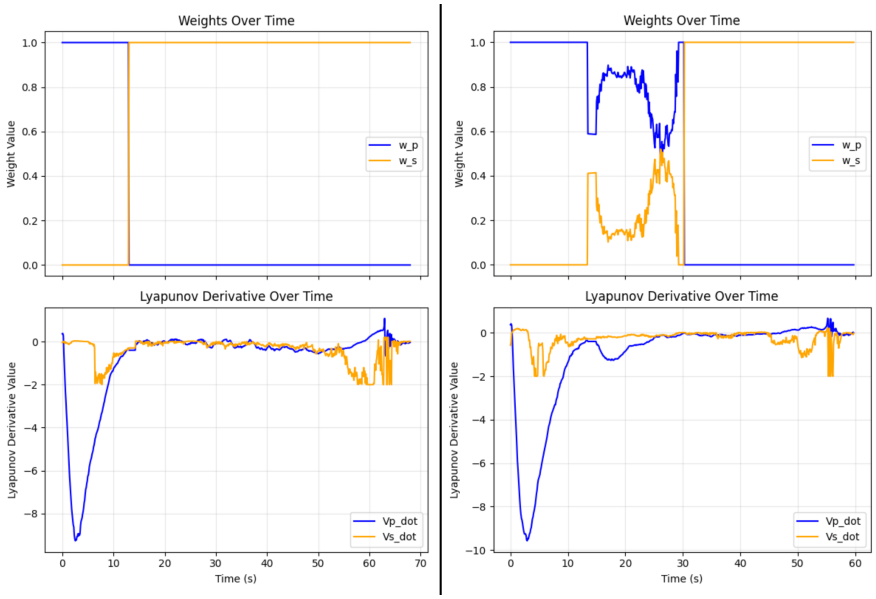

Figure 6.16: VS weights and Lyapunov Derivative (HITL). Left: Discrete mode. Right: Ratio mode.

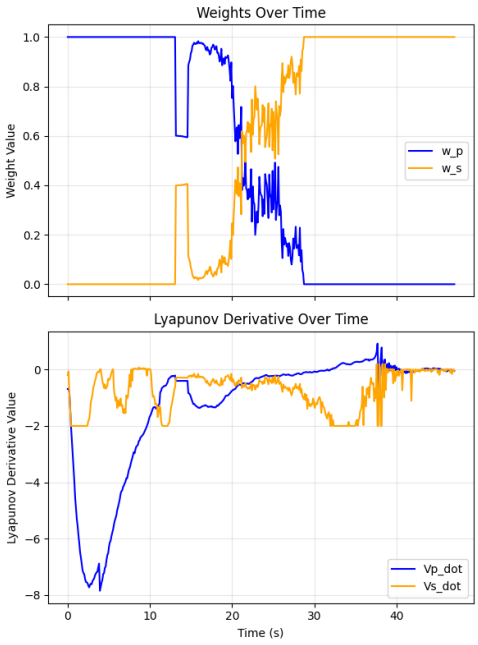

Figure 6.17: VS weights and Lyapunov Derivative (HITL): Softmax mode.

Switching Mode	Image Features SSE [px]	Robot Position SSE [m]	Robot Attitude SSE [°]
Discrete	5.65	0.001	0.143
Ratio	6.63	0.0017	0.18
Softmax	5.09	0.001	0.25

Table 6.5: Steady-state error metrics across switching modes (HITL).

Switching Mode	Average Full Docking Duration [s]	Average Hybrid VS Duration [s]
Discrete	84.65	69.87
Ratio	68.69	51.50
Softmax	55.01	41.66

Table 6.6: Docking duration (HITL).

From the hardware results, we can see that softmax mode gives consistent performance with the simulation results in terms of duration and smoother feature error evolution.

The drawback with Lyapunov and hybrid switching is that they are sensitive to unit mismatches. PBVS Lyapunov function is defined based on position error (in meter), whereas IBVS Lyapunov function is defined from image features error (in pixel). A manual scaling needs to be applied to directly compare the magnitude of both functions. Furthermore, the temprature parameter k in softmax mode also needs to be manually adjusted to control the sharpness of the transition.

6.2.2 System Performance

Three time-critical ROS2 nodes are executed at real-time rates. First, a blob-detection node (OpenCV, HSV thresholding) at 30Hz. Second, the MPVS controller at 15Hz. Third, the pose estimation node at 10Hz. This indicates that the system meets real-time requirements under limited computational resources.

Chapter 7

Conclusions and Future work

7.1 Conclusions

This thesis presented a docking pipeline that combines model-based pose estimation (GO-ICP followed by GICP refinement) by improving the current trajectory-tracking MPC into an MPVS controller through a hybrid VS formulation. The approach requires only an RGB-D camera and a CAD model of the docking station, relying on geometric fiducial points in a semi-structured environment. The cascaded ICP demonstrated robust performance with an acceptable initialization time. Among the tested switching strategies, the dynamic switching approach consistently outperformed discrete switching, with the softmax-based mode achieving the best result. The system was validated in both SITL and HITL environment, showing that it can operate at high frequency within limited computational resources.

7.2 Limitations

The study surfaced several limitations, generalization to full 6-DOF dynamics remains to be tested. The blob detection is further required to be improved. Currently, it assumes adequate texture/contrast and depth quality, and can degrade under small occlusions or extreme lighting change. It also sensitive with image blur and can detect false positive blobs. The dynamic weighting used inside the MPC is heuristic rather than formally optimal, and while effective in practice, it lacks a stability/performance guarantee for all operating conditions. On the pose estimation part, the GO-ICP sometime provides wrong orientation when there are outliers far outside the object's region of interest in the scene point cloud.

Other limitation observed during docking occurs when the robot makes near-contact with the cone guider without being perfectly aligned to the docking station. By that, the cone can unintentionally induce a rotation in the robot's attitude, which may drive the image dynamics into a local minimum and hinder convergence.

7.3 Future work

On the control side, a possibility is to improve the hybrid strategy with visibility/conditioning metrics (e.g., feature count, feature stability, interaction-matrix condition number) rather than fixed heuristics. On the perception side, replacing the point features with lines (e.g., handrail features, docking station structures, etc.) and adopting more robust photometric pipelines would improve resilience to clutter and illumination changes. Furthermore, an outlier removal using image dynamics and adding features consistency could be implemented to make the blob detection more robust.

For pose estimation, an outlier removal algorithm could be performed before running GO-ICP. One approach is to cluster the scene point cloud, compute 3D descriptors (e.g., FPFH or SHOT) for each cluster, and compare them with descriptors derived from the CAD model. The cluster with the highest similarity is then chosen as the region of interest for registration.

For the lab, wrapping all the developed system with a behaviour tree behind a clean user interface (UI) could also be a great addition to the lab facilities, where the operator can look at the UI and with only a single 'Dock' action, the system will run a safety check and perform the full docking process. Finally, integrating mechanical tolerance or compliance into the docking interface design to reduce the sensitivity to misalignment.

References

[1] W. Fehse, *Automated Rendezvous and Docking of Spacecraft*, 1st ed. Cambridge University Press, 2003. [Page 1.]

[2] A. Fear and G. Lightsey, E. "Autonomous rendezvous and docking implementation for small satellites using model predictive control," *Journal of Guidance, Control, and Dynamics*, pp. 1–9, Jan. 2024. doi: 10.2514/1.G007523 [Page 1.]

[3] M. Wilde, S. C. Hannon, and U. Walter, "Evaluation of head-up displays for teleoperated rendezvous & docking," in *Proceedings of the 2012 IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2012. doi: 10.1109/AERO.2012.6187302 pp. 1–14. [Page 1.]

[4] L. M. Amaya-Mejía, M. Ghita, J. Dentler, M. Olivares-Mendez, and C. Martinez, "Visual servoing for robotic on-orbit servicing: A survey," *arXiv*, vol. arXiv:2409.02324, sep 2024. doi: 10.48550/arXiv.2409.02324. [Online]. Available: https://arxiv.org/abs/2409.02324 [Page 2.]

[5] H. Zhang, M. Li, S. Ma, H. Jiang, and H. Wang, "Recent advances on robot visual servo control methods," *Recent Patents on Mechanical Engineering*, vol. 14, no. 3, pp. 298–312, Aug. 2021. doi: 10.2174/2212797613999201117151801 [Pages 2 and 8.]

[6] Z. Machkour, D. Ortiz-Arroyo, and P. Durdevic, "Classical and deep learning based visual servoing systems: a survey on state of the art," *J. Intell. Robot. Syst.*, vol. 104, no. 1, p. 11, jan 2022. doi: 10.1007/s10846-021-01540-w [Page 2.]

[7] Z. Machkour, D. Ortiz Arroyo, and P. Durdevic, "Classical and deep learning based visual servoing systems: a survey on state of the art," *Journal of Intelligent and Robotic Systems*, vol. 104, no. 1, pp. 1–27, Jan. 2022. doi: 10.1007/s10846-021-01540-w [Page 2.]

[8] F. Chaumette and S. Hutchinson, "Visual servo control. I. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, Dec. 2006. doi: 10.1109/MRA.2006.250573 [Page 2.]

[9] H. Tsubota, S. Kagami, and H. Mizoguchi, "Sift-cloud-model generation method for 6d pose estimation and its evaluation," in *2011 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2011, pp. 3323–3328. [Page 2.]

[10] Z. Ren, X. Tang, G. Ren, and D. Wu, "Research on pose estimation algorithm of non-cooperative target tracked vehicles based on pnp model," *AIP Advances*, vol. 15, no. 3, 2025. [Page 2.]

[11] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, Mar. 2021. doi: 10.1007/s10462-020-09888-5 [Page 2.]

[12] J. Liu, W. Sun, H. Yang, Z. Zeng, C. Liu, J. Zheng, X. Liu, H. Rahmani, N. Sebe, and A. Mian, "Deep learning-based object pose estimation: A comprehensive survey," *arXiv preprint arXiv:2405.07801*, May 2024. doi: 10.48550/arXiv.2405.07801 [Page 2.]

[13] J. Sun, Z. Wang, S. Zhang, X. He, H. Zhao, G. Zhang, and X. Zhou, "Onepose: One-shot object pose estimation without cad models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6825–6834. [Page 2.]

[14] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects," Mar. 2024, arXiv:2312.08344 [cs]. [Online]. Available: http://arxiv.org/abs/2312.08344 [Pages 2 and 28.]

[15] M. Johansson, "Linear Quadratic and Model Predictive Control: Lecture Notes for EL2700," Lecture notes, Stockholm, Sweden, 2024, p. 139. First edition 2018. [Pages 2 and 26.]

[16] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 933–939, Oct. 2010. doi: 10.1109/TRO.2010.2056590 [Page 2.]

[17] X. Bian, W. Chen, D. Ran, Z. Liang, and X. Mei, "Fima-reader: A cost-effective fiducial marker reader system for autonomous mobile robot docking in manufacturing environments," *Applied Sciences*, vol. 13, no. 24, p. 13079, 2023. [Page 2.]

[18] N. R. Gans and S. A. Hutchinson, "Stable visual servoing through hybrid switched-system control," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 530–540, June 2007. doi: 10.1109/TRO.2007.895067 [Pages 3, 12, 29, and 48.]

[19] A. Rastegarpanah, A. Aflakian, and R. Stolkin, "Improving the manipulability of a redundant arm using decoupled hybrid visual servoing," *Applied Sciences*, vol. 11, no. 23, 2021. doi: 10.3390/app112311566. [Online]. Available: https://www.mdpi.com/2076-3417/11/23/11566 [Page 3.]

[20] F. Chaumette, S. Hutchinson, and P. Corke, "Visual servoing," *Springer handbook of robotics*, pp. 841–866, 2016. [Page 8.]

[21] O. Kermorgant and F. Chaumette, "Combining IBVS and PBVS to ensure the visibility constraint," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, CA, USA: IEEE, Sep. 2011. doi: 10.1109/IROS.2011.6094589 pp. 2849–2854. [Online]. Available: https://doi.org/10.1109/IROS.2011.6094589 [Page 8.]

[22] M. Björkman, "Projections and transformations," Lecture slides, 2023, accessed during course lecture at KTH Royal Institute of Technology, October 31, 2023. [Pages xi and 9.]

[23] R. Tedrake, *Robotic Manipulation*, 2024. [Online]. Available: http://manipulation.mit.edu [Page 11.]

[24] E. Malis, F. Chaumette, and S. Boudet, "2–1/2d visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, Apr. 1999. [Page 11.]

[25] F. Chaumette and S. Hutchinson, "Visual servo control, part ii: Advanced approaches," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007. doi: 10.1109/MRA.2007.339609 [Page 12.]

[26] F. Chaumette and E. Malis, "2 1/2 d visual servoing: a possible solution to improve image-based and position-based visual servoings,"

in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. San Francisco, CA, USA: IEEE, 2000. doi: 10.1109/ROBOT.2000.844123 pp. 630–635. [Page 12.]

[27] W. Lyu, W. Ke, H. Sheng, X. Ma, and H. Zhang, "Dynamic downsampling algorithm for 3d point cloud map based on voxel filtering," *Applied Sciences*, vol. 14, no. 8, p. 3160, 2024. doi: 10.3390/app14083160 [Pages xi and 14.]

[28] K. Jin, P. Liu, R. Sun, Z. Wei, and Z. Zhou, "Real-time plane segmentation in a ros-based navigation system for the visually impaired," in *2016 Fourth International Conference on Ubiquitous Positioning, Indoor Navigation and Location Based Services (UPINLBS)*. IEEE, 2016, pp. 170–175. [Page 14.]

[29] S. Oßwald, J.-S. Gutmann, A. Hornung, and M. Bennewitz, "From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2011, pp. 93–98. [Page 14.]

[30] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. [Pages 14 and 15.]

[31] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *2009 IEEE international conference on robotics and automation*. Ieee, 2009, pp. 3206–3211. [Page 14.]

[32] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004. [Page 15.]

[33] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417. [Page 15.]

[34] W. Wei, "Sac-ia algorithm based on parallel kd-tree search and improved feature point selection," in *2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. IEEE, 2023, pp. 348–352. [Page 16.]

[35] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606. [Page 16.]

[36] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435. [Page 17.]

[37] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2241–2254, Nov. 2016. doi: 10.1109/TPAMI.2015.2513405. [Online]. Available: http://ieeexplore.ieee.org/document/7368945/ [Pages xi, 18, 20, and 42.]

[38] R. He, J. Rojas, and Y. Guan, "A 3D Object Detection and Pose Estimation Pipeline Using RGB-D Images," Mar. 2017, arXiv:1703.03940 [cs]. [Online]. Available: http://arxiv.org/abs/1703.03940 [Pages 20 and 21.]

[39] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*. Springer, 2012, pp. 548–562. [Page 20.]

[40] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981. [Page 21.]

[41] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4561–4570. [Page 21.]

[42] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 632–11 641. [Page 21.]

[43] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv preprint arXiv:1711.02508*, 2017. [Pages 22 and 23.]

[44] S. Cho, S. Huh, and D. H. Shim, "Visual Detection and Servoing for Automated Docking of Unmanned Spacecraft," *TRANSACTIONS OF*

THE JAPAN SOCIETY FOR AERONAUTICAL AND SPACE SCIENCES, AEROSPACE TECHNOLOGY JAPAN, vol. 12, no. APISAT-2013, pp. a107–a116, 2014. doi: 10.2322/tastj.12.a107. [Online]. Available: https://www.jstage.jst.go.jp/article/tastj/12/APISAT-2013/12_TJSAS-D-14-00011/_article [Page 28.]

[45] C. M. Pong, A. Saenz-Otero, and D. W. Miller, "Autonomous thruster failure recovery on underactuated spacecraft using model predictive control," Massachusetts Institute of Technology, Cambridge, MA, Technical Report, 2011, also appears as AAS 11-033 in the AAS Guidance and Control 2011 proceedings. [Online]. Available: https://dspace.mit.edu/handle/1721.1/81885 [Page 32.]

[46] S. M. Kelly and S. P. Cryan, "International docking standard (idss) interface definition document (idd): Revision-e," Tech. Rep., 2016. [Page 37.]

[47] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. [Page 39.]

[48] P. Roque, S. Phodapol, E. Krantz, J. Lim, J. Verhagen, F. J. Jiang, D. Dörner, H. Mao, G. Tibert, R. Siegwart *et al.*, "Towards open-source and modular space systems with atmos," *arXiv preprint arXiv:2501.16973*, 2025. [Page 45.]

[49] A. A. Hafez, E. Cervera, C. Jawahar, and I. CVIT, "Stable hybrid visual servo control by a weighted combination of image-based and position-based algorithms," *International Journal of Control and Automation*, vol. 6, no. 3, pp. 149–164, 2013. [Page 48.]

[50] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022. [Page 49.]

TRITA – XXX-EX 2025:0000
Stockholm, Sweden 2025

www.kth.se

€€€€ For DIVA €€€€

{
”Author1”: { ”Last name”: ”Pramono”,
”First name”: ”Tafarrel Firhannoza”,
”E-mail”: ”tafarrel@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
}
},
”Cycle”: ”2”,
”Course code”: ”DA231X”,
”Credits”: ”30.0”,
”Degree1”: {”Educational program”: ”Master’s Programme, Systems, Control and Robotics, 120 credits”
,”programcode”: ”TSCRM”
,”Degree”: ”Master of Science in Engineering”
,”subjectArea”: ”Electrical Engineering, specializing in Systems, Control and Robotics”
},
”Title”: {
”Main title”: ”Hybrid PBVS-IBVS Model Predictive Visual Servoing”,
”Subtitle”: ”For Autonomous Docking of a Free-Flying Robot”,
”Language”: ”eng” },
”Alternative title”: {
”Main title”: ”Hybrid PBVS–IBVS Modellprediktiv Visuell Servostyrning”,
”Subtitle”: ”För Autonom Dockning av en Fritt Flygande Robot”,
”Language”: ”swe”
},
”Supervisor1”: { ”Last name”: ”Roque”,
”First name”: ”Pedro”,
”E-mail”: ”padr@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”DCS” }
},
”Supervisor2”: { ”Last name”: ”Miraldo”,
”First name”: ”Pedro”,
”E-mail”: ”miraldo@merl.com”,
”Other organisation”: ”Mitsubishi Electric Research Laboratory”
},
”Examiner1”: { ”Last name”: ”Dimarogonas”,
”First name”: ”Dimos V.”,
”E-mail”: ”dimos@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”DCS” }
},
"National Subject Categories": "20201, 20208",
"SDGs": "9",
”Other information”: {”Year”: ”2025”, ”Number of pages”: ”xvi,73”},
”Copyrightleft”: ”copyright”,
”Series”: { ”Title of series”: ”TRITA – XXX-EX” , ”No. in series”: ”2025:0000” },
”Opponents”: { ”Name”: ”A. B. Normal & A. X. E. Normalè”},
”Presentation”: { ”Date”: ”2022-03-15 13:00”
,”Language”:”eng”
,”Room”: ”via Zoom https://kth-se.zoom.us/j/ddddddddddd”
,”Address”: ”Isafjordsgatan 22 (Kistagången 16)”
,”City”: ”Stockholm” },
”Number of lang instances”: "2",
”Abstract[eng]”: €€€€


```
\generalExpl{Enter your abstract here!}

Interest in the development of \glsxtrfull{ARD} has been growing since the 1960s as it is an
important procedure for refueling  spacecraft and transferring resources. Achieving centimeter-scale
accuracy in \glsxtrshort{ARD} commonly relies on \glsxtrfull{VS}. Two common classical
\glsxtrshort{VS} methods are \glsxtrfull{IBVS} and \glsxtrfull{PBVS} utilize a velocity-based control
law that is straightforward, yet suffer from depth or visibility sensitivity and local minima.

This thesis presents a hybrid \glsxtrshort{VS} with \glsxtrfull{MPC} docking pipeline for a
free-flyer that requires only an \glsxtrshort{RGBD} camera and a \glsxtrfull{CAD} model of a docking
station. The pose of the station is initialized by \glsxtrfull{GOICP} and refined with
\glsxtrfull{GICP}. This estimation seeds a trajectory-tracking \glsxtrshort{MPC} that orients the
free-flyer to maximize the visibility of geometric point features. A dynamic weighting strategy in
the \glsxtrshort{MPC} cost is then used to control the soft-switching between \glsxtrshort{PBVS} and
\glsxtrshort{IBVS}. Finally, the feature dynamics is described to the \glsxtrshort{MPC} to minimize
the image errors while respecting system constraints.

We evaluated the method in \glsxtrfull{ROS} with Gazebo on a planar 3-\glsxtrfull{DOF} free-flyer and
further validated it in the KTH \glsxtrfull{SRL}, reporting docking success rate, steady-state pose
error, and computation time. The results indicated reliable docking without AR tags, using only
geometric fiducials, and demonstrated that the \glsxtrshort{PBVS}-\glsxtrshort{IBVS} transition
improves visibility and mitigates local-minimum failures.
```

€€€€,
"Keywords[eng]": €€€€
Computer vision, Free-flyer, Iterative Closest Point, Model Predictive Control, Visual servo €€€€,
"Abstract[swe]": €€€€

\generalExpl{Enter your Swedish abstract or summary here!}
Intresset för utvecklingen av \glsxtrfull{ARD} har ökat sedan 1960-talet, eftersom det är ett viktigt förfarande för att tanka rymdfarkoster och överföra resurser. Att uppnå noggrannhet på centimeternivå i \glsxtrshort{ARD} bygger ofta på \glsxtrfull{VS}. Två vanliga klassiska \glsxtrshort{VS}-metoder är \glsxtrfull{IBVS} och \glsxtrfull{PBVS}, som använder en hastighetsbaserad reglerlag som är enkel men ändå känslig för djup och synlighet samt drabbas av lokala minima.

Denna avhandling presenterar en hybrid \glsxtrshort{VS}- och \glsxtrfull{MPC}-baserad dockningspipeline för en fri-flygande plattform som endast kräver en \glsxtrshort{RGBD}-kamera och en \glsxtrfull{CAD}-modell av en dockningsstation. Stationens pose initieras med \glsxtrfull{GOICP} och förfinas med \glsxtrfull{GICP}. Denna skattning initierar en banaföljande \glsxtrshort{MPC} som orienterar den fri-flygande plattformen för att maximera synligheten av geometriska punktlandmärken. En dynamisk viktning i \glsxtrshort{MPC}-kostnadsfunktionen används därefter för att åstadkomma mjuk växling mellan \glsxtrshort{PBVS} och \glsxtrshort{IBVS}. Detta gör att \glsxtrshort{MPC} kan optimera kamerans hastigheter för att minimera bildfelet samtidigt som systembegränsningarna respekteras.

Vi utvärderade metoden i \glsxtrfull{ROS} med Gazebo på en plan 3-\glsxtrfull{DOF} fri-flygande plattform och validerade den vidare i KTH \glsxtrfull{SRL}. Vi redovisar andelen lyckade dockningar, stationärt posefel och beräkningstid. Resultaten visar tillförlitlig dockning utan AR-taggar - med enbart geometriska referensmarkörer - och att övergången \glsxtrshort{PBVS-}\glsxtrshort{IBVS} förbättrar synligheten och minskar fel orsakade av lokala minima.

€€€€,
"Keywords[swe]": €€€€
Datorseende, Friflygande robot,Iterative Closest Point, Modellprediktiv reglering, Visuell servostyrning €€€€,
}

acronyms.tex

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
% The following command is used with glossaries-extra
\setabbreviationstyle[acronym]{long-short}
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
%                                     or \newacronym[options]{label}{acronym}{phrase}
% see "User Manual for glossaries.sty" for the  details about the options, one example is shown below
% note the specification of the long form plural in the line below
\newacronym[longplural={Debugging Information Entities}]{DIE}{DIE}{Debugging Information Entity}
%
% The following example also uses options
\newacronym[shortplural={OSes}, firstplural={operating systems (OSes)}]{OS}{OS}{operating system}


\newacronym{ARD}{ARD}{autonomous rendezvous and
docking}
\newacronym{NASA}{NASA}{National Aeronautics and Space Administration}
\newacronym{MPC}{MPC}{Model Predictive Control}
\newacronym{MPVS}{MPVS}{Model Predictive Visual Servoing}
\newacronym{VS}{VS}{Visual Servoing}
\newacronym{SRL}{SRL}{Space Robotics Laboratory}
\newacronym{DOF}{DOF}{degree of freedom}
\newacronym{FOV}{FOV}{field of view}
\newacronym{LQR}{LQR}{-LinearQuadratic Regulator}
\newacronym{IBVS}{IBVS}{Image-Based Visual Servoing}
\newacronym{PBVS}{PBVS}{Position-Based Visual Servoing}
\newacronym{ICP}{ICP}{Iterative Closest Point}
\newacronym{PnP}{PnP}{Perspective-n-Point}
\newacronym{GPU}{GPU}{graphics Processing Unit}
\newacronym{LLM}{LLM}{Large language model}
\newacronym{ROS}{ROS}{Robot Operating System}
\newacronym{GOICP}{GO-ICP}{Globally optimal-ICP}
\newacronym{GICP}{GICP}{Generalized Iterative Closest Point}
\newacronym{RANSAC}{RANSAC}{RANdom SAmple Consensus}
\newacronym{CAD}{CAD}{Computer Aided Design}
\newacronym{TOF}{TOF}{Time-of-Flight}
\newacronym{ATMOS}{ATMOS}{Autonomy Testbed for Multi-purpose Orbiting Systems}
\newacronym{EKF}{EKF}{Extended Kalman filter}
\newacronym{SITL}{SITL}{software in the loop}
\newacronym{HITL}{HITL}{hardware in the loop}
\newacronym{PCL}{PCL}{Point Cloud Library}
\newacronym{RGBD}{RGB-D}{Red Green Blue-Depth}
\newacronym{RGB}{RGB}{Red Green Blue}
\newacronym{MSE}{MSE}{Mean Squared Error}
\newacronym{BnB}{BnB}{branch-n-bound}
\newacronym{GHT}{GHT}{Generalised Hough transform}
\newacronym{SVD}{SVD}{Singular value decomposition}
\newacronym{PCA}{PCA}{principal component analysis}
\newacronym{SIFT}{SIFT}{Scale Invariant Feature Transform}
\newacronym{SHOT}{SHOT}{Signature of Histograms of OrienTations}
\newacronym{FPFH}{FPFH}{Fast Point Feature Histograms}
\newacronym{SURF}{SURF}{Speeded-Up Robust Features}


% note the use of a non-breaking dash in long text for the following acronym
\newacronym{IQL}{IQL}{Independent -QLearning}

% example of putting in a trademark on first expansion
\newacronym[first={NVIDIA OpenSHMEM Library (NVSHMEM\texttrademark)}]{NVSHMEM}{NVSHMEM}{NVIDIA OpenSHMEM Library}

\newacronym{KTH}{KTH}{KTH Royal Institute of Technology}

\newacronym{LAN}{LAN}{Local Area Network}
\newacronym{VM}{VM}{virtual machine}
% note the use of a non-breaking dash in the following acronym
\newacronym{WiFi}{-WiFi}{Wireless Fidelity}

\newacronym{WLAN}{WLAN}{Wireless Local Area Network}
\newacronym{UN}{UN}{United Nations}
\newacronym{SDG}{SDG}{Sustainable Development Goal}
```