GOA UNIVERSITY

CERTIFICATE

This is to certify that the project report entitled

Navigation and Localization of Underwater Vehicle using Fiducial Markers

submitted by

Aryan Singh P.R.No.: 202111969

Chrisann Ferrao P.R.No.: 202111972

Govind Pandey P.R.No.: 202111798

Jayden Viegas P.R.No.: 202111978

This work, conducted under my supervision during the academic year 2024–2025, is a genuine endeavor. It was undertaken as a partial fulfillment of the prerequisites for obtaining the Bachelor of Engineering degree in Electronics & Telecommunications at Goa College of Engineering, Farmagudi, Ponda Goa, affiliated with Goa University.

Date:

Prof. Milind Fernandes
[Project Guide]

Dr. H. G. Virani
[Head of Department]
ETC Engineering Department

Dr. M. S. Krupashankara
[Principal]
Goa College of Engineering

[Internal Examiner]

[External Examiner]

Acknowledgement

We are overwhelmed with gratitude and humility as we acknowledge all those who have helped us successfully complete our Project, turning initial ideas into a working and tangible result.

First and foremost, we extend our heartfelt thanks to our respected Principal, **Dr. M. S. Krupashankara**, and the Head of the Department of Electronics and Telecommunication Engineering, **Dr. H. G. Virani** , for providing us with the essential resources, encouragement, and infrastructure to carry out our project in the best possible manner.

We would like to express our sincere appreciation to our final year project guide, **Prof. Milind Fernandes**, for his continuous guidance, valuable insights, and unwavering support throughout the duration of the project. His inputs were crucial in shaping our technical direction and helping us articulate our work in this report.

We are also grateful to all our faculty members and lab assistants for their cooperation and for granting us access to the required lab equipment, technical support, and project resources. Their encouragement created an environment conducive to innovation and learning.

Lastly, we thank the institution for fostering a research-oriented and hands-on learning atmosphere that empowered us to carry out this project with commitment and confidence.

Aryan Singh
Chrisann Ferrao
Govind Pandey
Jayden Viegas

Abstract

This project presents the development of a low-cost localization and navigation system for Unmanned Underwater Vehicles (UUVs) using vision-based fiducial markers. The system employs AprilTags to estimate the 6-DOF pose of the vehicle in GPS-denied environments, offering an efficient alternative to traditional, expensive localization techniques such as acoustic beacons and inertial navigation systems. The navigation algorithm was implemented by using a USB cameras and a PnP-based method.

The detection pipeline is executed in real-time on a Raspberry Pi 4 Model B running BlueOS, with AprilTag recognition handled by a multithreaded Python script optimized for performance. Pose data is transmitted to a Pixhawk 2.4.8 flight controller via the MAVLink protocol and visualized using QGroundControl. Calibration procedures, distortion correction, and 3D trajectory plotting were incorporated to enhance accuracy and system feedback. Testing in controlled environments demonstrated the system's robustness, real-time responsiveness, and adaptability for confined or structured underwater spaces. This work lays the groundwork for autonomous underwater operations in research, inspection, and educational applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The advancement of unmanned underwater vehicles (UUVs) has brought significant progress in marine exploration, infrastructure inspection, and environmental monitoring. However, achieving accurate localization and navigation underwater remains a critical challenge due to the unavailability of GPS and the high cost and complexity of traditional solutions like Inertial Navigation Systems (INS), Doppler Velocity Logs (DVL), and acoustic-based systems (LBL, USBL). These systems, while effective in open waters, are not practical for confined or budget-sensitive environments such as testing tanks, pipelines, or small-scale deployments. This project proposes a vision-based localization and navigation approach using fiducial markers, specifically AprilTags and ArUco markers, as a low-cost and efficient alternative. Fiducial markers can be placed in the underwater environment and detected using onboard cameras. From the marker detections, the vehicle can estimate its 6DOF pose (position and orientation), enabling reliable localization without the need for expensive acoustic equipment. A detailed comparison between ArUco and AprilTags was conducted. While both are square, binary-coded markers, AprilTags offer better detection robustness, error correction, and performance in low-visibility or partially occluded scenarios—making them more suitable for underwater applications. As a result, AprilTags were chosen for implementation. The project is built on two key algorithmic approaches: Algorithm 1 uses a single camera to detect AprilTags via the apriltag library. It employs the PnP method to estimate the vehicle's pose relative to each tag. This method is efficient, lightweight, and suitable for simple environments.

Algorithm 2 enhances accuracy with a dual camera setup. By solving PnP, P3P, and EPnP with known camera calibration parameters and distortion coefficients, the system produces faster and more precise localization. The vehicle's position is estimated relative to a defined origin tag in the environment.

Initially, the system was developed using Raspberry Pi 3B on Raspberry Pi OS. However, to meet performance demands and enhance compatibility, the platform was migrated to Raspberry Pi 4 running BlueOS, a marine-optimized OS better suited for real-time operations. The system integrates with a Pixhawk flight controller for controlling motors and executing navigation commands. Coordinate data generated by the vision system is sent to the Pixhawk, which uses it to update vehicle movement. Additionally, feedback from Pixhawk's internal estimations is compared with vision-based data to apply error correction and improve stability. This project demonstrates a practical, accessible solution for underwater navigation in niche or confined environments. By reducing dependency on expensive hardware and leveraging computer vision and embedded systems, it offers a scalable platform for testing, academic research, and real-world underwater applications.

1.1 History

Unmanned Underwater Vehicles (UUVs), have evolved significantly over the past century. Their development has been driven by the need to explore, monitor, and operate in underwater environments that are otherwise inaccessible or hazardous to humans.

The origins of UUVs trace back to the 1950s and 60s, primarily for military and research applications. One of the earliest examples was the Special Purpose Underwater Research Vehicle (SPURV), developed in 1957 by the University of Washington with funding from the U.S. Navy. SPURV was used for studying ocean currents and thermal layers, marking the beginning of autonomous underwater exploration.

During the Cold War, UUV development accelerated due to the demand for

underwater surveillance and mine detection. Remotely Operated Vehicles (ROVs) gained popularity in the 1970s and 80s for subsea oil and gas exploration, enabling safe inspection and maintenance of deep-sea equipment.

The 1990s and 2000s saw the emergence of truly autonomous vehicles. Advances in embedded computing, navigation, and battery technology enabled longer and more complex missions. Vehicles like the REMUS (Remote Environmental Monitoring UnitS) and Bluefin AUVs became standards for military, scientific, and commercial tasks.

In recent years, UUVs have found growing applications in environmental monitoring, seabed mapping, marine archaeology, and underwater infrastructure inspection. Modern UUVs are equipped with advanced navigation systems (such as inertial navigation and DVL), high-resolution sonar, and cameras. With the integration of machine learning and vision-based localization techniques, the autonomy and intelligence of these systems continue to improve.

Today, UUVs represent a critical technology for oceanographic research, defense operations, and sustainable marine development, enabling exploration of the vast and largely uncharted underwater world.

1.2 Motivation

Unmanned Underwater Vehicles (UUVs) have become indispensable tools for oceanographic research, subsea inspection, and underwater surveillance. However, one of the most persistent challenges in the operation of these vehicles is accurate localization and reliable navigation in underwater environments. Traditional navigation methods such as Inertial Navigation Systems (INS), Doppler Velocity Logs (DVL), and Long Baseline (LBL) or Ultra Short Baseline (USBL) acoustic positioning systems often suffer from limitations including high cost, bulkiness, complexity, and signal degradation in turbid or cluttered waters. Moreover, these traditional systems are optimized for open-sea navigation and large-scale deployments, which makes them overkill for confined, shallow, or structured underwater environments such as pools, testing tanks, dams, pipelines, or underwater industrial facilities. In such spaces, precise movement and repeatable navigation are more critical than large-area exploration. Additionally, due to acoustic signal in-

terference caused by reflections or surrounding structures, traditional systems may fail or provide poor accuracy. This makes them ill-suited for testing phases, small-scale vehicles, or niche applications that require cost-effectiveness and simplicity without compromising accuracy. This project aims to overcome these challenges by developing a low-cost, vision-based navigation and localization algorithm using fiducial markers. Fiducial markers, such as ArUco, AprilTags, or custom black-and-white patterns, are visual codes that can be uniquely identified by computer vision systems and can provide both position and orientation (pose) information. When placed strategically in the underwater environment, these markers serve as landmarks that enable the UUV to understand its location relative to them using onboard cameras and processing units.

This method offers multiple advantages:

- Cost-effectiveness: Fiducial markers are inexpensive and can be printed on waterproof sheets. The onboard system only requires a camera and a modest processing unit, like a Raspberry Pi or NVIDIA Jetson, which is significantly cheaper than INS-DVL systems.
- Simplicity: The algorithmic pipeline — involving marker detection, pose estimation, and localization — is relatively easy to implement and debug. This simplicity is ideal for educational and prototyping environments.
- Modularity and Scalability: The system can be deployed incrementally, with additional markers improving localization accuracy. It is suitable for structured indoor spaces like research pools or inspection facilities where the layout is known or semi-static.
- Testbed Readiness: Since many research projects and prototype UUVs undergo initial testing in controlled environments, using fiducial markers allows precise ground-truthing and easier debugging of vehicle behavior.

Furthermore, advancements in real-time image processing and robust computer vision libraries such as OpenCV have made it feasible to deploy such systems even on embedded platforms. The project also encourages cross-disciplinary learning, integrating robotics, embedded systems, computer vision, and underwater sensing. This approach fills a specific yet crucial gap: providing a lightweight, portable, and reconfigurable localization method for scenarios where traditional systems are impractical. Applications could include pipeline inspection, marine archaeology

in shallow waters, infrastructure maintenance, and underwater academic research setups. By shifting focus to vision-based localization using fiducial markers, this project not only addresses the limitations of traditional systems but also democratizes underwater robotics by making it more accessible to students, researchers, and small institutions with limited budgets. It paves the way for smart, adaptive, and autonomous underwater systems that can thrive even in confined or challenging environments where acoustic and inertial systems fall short. In conclusion, the motivation behind this project stems from a practical need to simplify underwater localization without sacrificing performance, especially for emerging sectors of robotics where agility, affordability, and adaptability matter more than absolute range or complexity. Fiducial-marker-based systems represent a promising, scalable alternative, one that can accelerate innovation in underwater robotics by making localization intuitive, efficient, and reliable.

1.3 Outline

This report is organized into five chapters, each focusing on a specific aspect of the project:

- **Chapter 1: Introduction**
  Provides an overview of the project, including background, motivation, problem statement, and the overall structure of the report.
- **Chapter 2: Problem Statement**
  Describes the challenges faced in underwater localization, the limitations of traditional navigation methods, and introduces the motivation behind using fiducial markers such as AprilTags for vision-based localization in GPS-denied environments.
- **Chapter 3: Implementation**
  Details the hardware and software design, including system architecture, component integration, calibration, and the vision-based localization pipeline. It also describes MAVLink communication and real-time data visualization.
- **Chapter 4: Results and Discussion**
  Highlights the performance of the implemented system through tests and

visualizations. This includes detection accuracy, localization results, and insights into system behavior under different scenarios.

- **Chapter 5: Conclusion and Future Scope**
  Summarizes the major outcomes of the project and outlines potential improvements and future extensions such as stereo vision support and underwater deployment.

Chapter 2

Project Statement

2.1 Introduction

Localization and its Importance

Localization refers to the process of determining the position and orientation of a robot or vehicle relative to a known reference frame, typically within a defined environment. Accurate localization is essential for autonomous navigation, mapping, and path planning. In underwater environments, traditional GPS is ineffective due to signal attenuation, necessitating alternative localization methods. These include inertial navigation systems (INS), acoustic-based localization (like USBL and beacons), and vision-based methods such as visual SLAM.

Fiducial Marker-Based Localization

Fiducial markers have emerged as a powerful alternative for real-time pose estimation and localization in robotics. These markers are visually distinctive symbols, often black-and-white patterns arranged in grids, which can be easily identified in images using computer vision algorithms. Among the many marker families developed, ArUco, AprilTag, and ARToolKit are some of the most widely used systems. They enable robots to infer their position and orientation in an environment by detecting the marker's identity and pose from camera images. In underwater and

indoor robotics, where traditional GPS and acoustic methods are ineffective, fiducial markers offer a simple, cost-effective, and visually robust solution. Their low computational cost and lack of dependence on external infrastructure make them ideal for small robots and embedded systems.

Inertial Navigation Systems (INS)

Inertial Navigation Systems estimate the position, velocity, and orientation of a vehicle using data from onboard sensors such as accelerometers and gyroscopes. These systems do not rely on external signals, making them ideal for underwater or GPS-denied environments. However, INS tends to suffer from drift over time due to the integration of sensor noise, requiring periodic correction through external references or fusion with other localization methods.

Ultra Short Baseline (USBL)

USBL (Ultra Short Baseline) systems use acoustic signals to estimate the position of an underwater vehicle. A surface station equipped with a transceiver emits acoustic pulses, and the vehicle responds with its own signal. By measuring the time delay and angle of arrival, the system computes the relative position of the vehicle. USBL is commonly used in deep-sea applications due to its long-range capability, but it can be affected by noise, multipath reflections, and the need for surface-based infrastructure.

Acoustic Beacons

Acoustic beacon-based systems use a network of pre-deployed beacons with known positions to triangulate the location of a vehicle equipped with an acoustic receiver. These systems provide accurate localization over larger underwater areas but require initial deployment and calibration. They are suitable for environments where high precision is required and where visual methods are limited due to poor visibility.

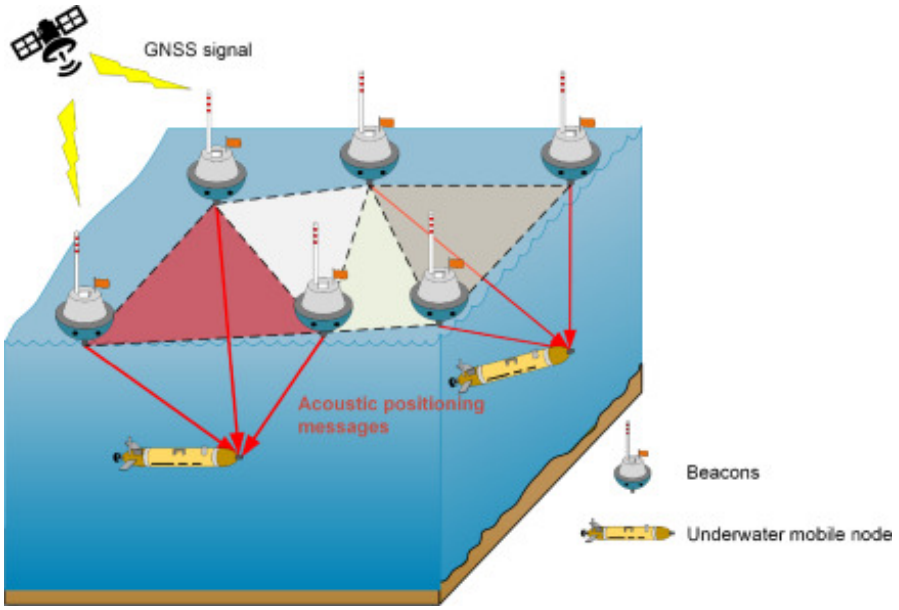

Figure 2.1: Triangulation using acoustic beacons

Visual SLAM

Visual SLAM (Simultaneous Localization and Mapping) is a method where a robot or vehicle constructs a map of an unknown environment while simultaneously estimating its own pose within that map using visual inputs such as camera data. In underwater applications, visual SLAM can be highly effective when combined with fiducial markers or natural scene features. It is computationally intensive but offers real-time performance when optimized, making it suitable for confined environments where acoustic methods are impractical.

2.2 Literature Review

2.2.1 An Underwater Visual Navigation Method Based on Multiple ArUco Markers

A cost-effective visual navigation system for unmanned underwater vehicles (UUVs) using multiple ArUco fiducial markers. Traditional underwater localization approaches, such as ultra-short-baseline (USBL) systems or Doppler velocity logs,

require expensive acoustic infrastructure and suffer from low update rates or noise interference. To address these challenges, the authors deploy a grid of ArUco markers on the floor of a towing tank and calibrate an underwater camera to correct for refractive distortions. By detecting each marker's pose relative to the camera with OpenCV's ArUco module, they invert the transformation to compute the camera's global position. A probabilistic noise model is derived for single-marker estimates, and a Mahalanobis-distance-based fusion algorithm optimally combines measurements from all visible markers, thereby bounding localization error without cumulative drift.

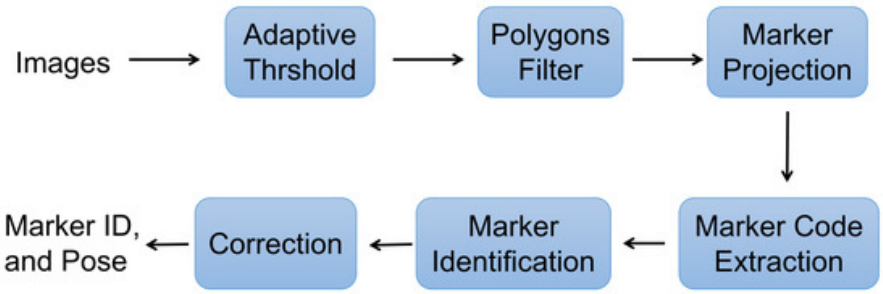

Figure 2.2: The ArUco Algorithm[1].

Extensive experiments with a VideoRay Pro 3 ROV demonstrated the method's effectiveness across various trajectories (lawnmower patterns, closed loops, and arbitrary paths). The fused multi-marker approach achieved position errors under 0.5 m, often around 0.2 m, even when some markers were partially occluded or at the limits of camera range. Unlike odometry, which accumulates error over time, this marker-based system provides consistent, repeatable localization in controlled environments. The authors highlight its applicability for operations near artificial subsea structures, suggesting that permanent ArUco installations could enable precise pipeline inspection or docking tasks without expensive acoustic arrays [1].

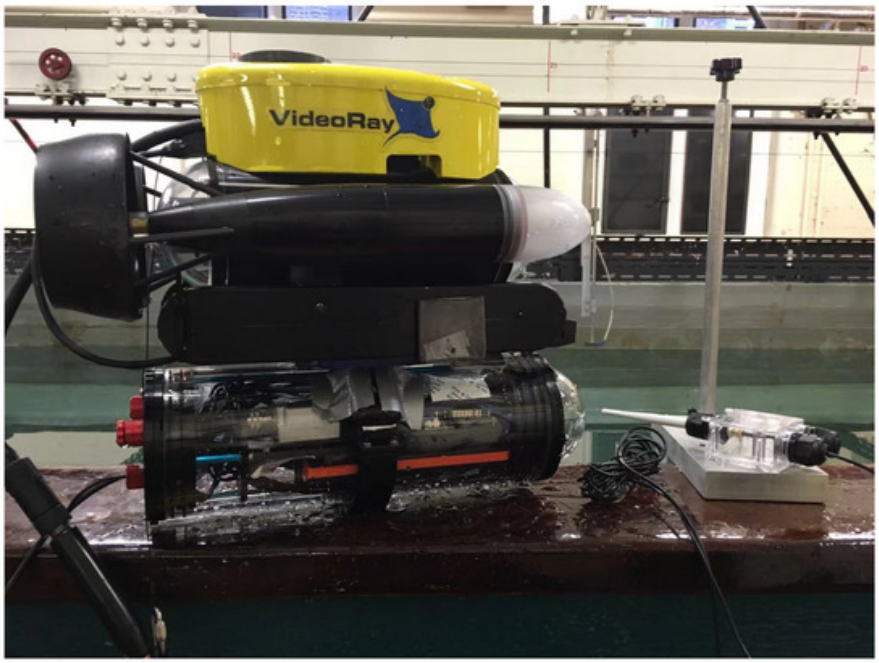

Figure 2.3: VideoRay Pro 3 with tube [1].

Conclusion

In *"An Underwater Visual Navigation Method Based on Multiple ArUco Markers"* by Xu et al., the authors propose a low-cost, vision-based localization system for UUVs by deploying multiple ArUco markers in a structured environment. Using OpenCV's ArUco library for marker detection and pose estimation, the system accounts for noise by modeling single-marker uncertainty and fusing multiple detections using a Mahalanobis-distance-based probabilistic method. This approach eliminates cumulative drift commonly seen in odometry-based systems.

The system was tested using a VideoRay Pro 3 ROV in a towing tank, achieving robust trajectory tracking across various patterns including lawnmower, circular, and random paths. The fused multi-marker strategy significantly improved localization accuracy, with position errors often below 0.5 meters. The study highlights the potential of ArUco-based systems for reliable underwater localization, particularly near man-made structures like pipelines or docks.

2.2.2 Towards Fast Fiducial Marker with Full 6 DOF

This paper introduces an improved fiducial marker system designed for **real-time, full 6 Degrees of Freedom (6 DOF) pose estimation.** Building upon prior designs like **WhyCon** and **WhyCode**, the proposed system maintains high computational efficiency while resolving one of the key challenges in circular marker systems—**pose ambiguity**. The work is highly relevant in robotics, especially for aerial and underwater vehicles where lightweight, inexpensive, and accurate localization is critical [2].

Motivation and Problem Statement

Traditional fiducial markers such as **AprilTag** and **ArUco** are widely used for 6 Degrees of Freedom (6 DOF) pose estimation. They rely on square binary designs and robust decoding, but are computationally intensive, especially when used in real-time embedded systems.

In contrast, circular markers like **WhyCon** offer significant speed advantages and can be processed faster due to their geometric simplicity. However, they suffer from **rotational symmetry**, which introduces **pose ambiguity** and limits their ability to estimate full 6 DOF pose.

WhyCode was introduced to address these issues by combining circular designs with encoded patterns. While it improved ID encoding, it introduced challenges in terms of marker readability and false ID detection in noisy environments.

A recent paper proposes a new circular marker system that aims to overcome these limitations. The proposed marker:

- Allows **unique ID encoding** without increasing computational complexity.
- **Resolves pose ambiguity** caused by symmetrical shapes.
- Maintains **sub-millisecond processing time**, making it suitable for high-speed applications.
- Performs **robustly even under detection noise**, enhancing reliability in real-world conditions.

WhyCon and WhyCode

WhyCon utilizes two concentric circles to achieve extremely fast and reliable detection. However, this design only supports **5 Degrees of Freedom (DOF)**, as it lacks the ability to estimate **yaw** due to its radial symmetry.

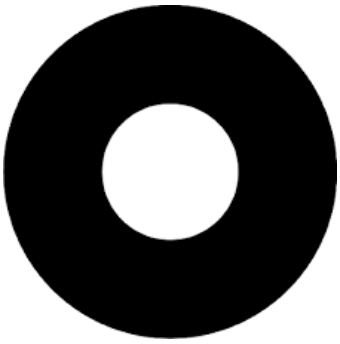

Figure 2.4: Example of a WhyCon marker [2].

To overcome this, **WhyCode** was introduced, which incorporates an angular "necklace" pattern around the circular marker to enable **yaw estimation**, thus achieving full **6 DOF** pose detection. While this approach addresses the symmetry issue, it introduces new challenges. Decoding the angular pattern becomes increasingly **error-prone** at low image resolutions or in environments with variable lighting conditions.

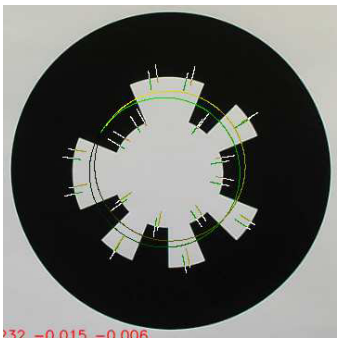

Figure 2.5: Example of a WhyCode marker with angular encoding [2]

The authors conclude that while circular markers are advantageous for fast detection, the problem of **pose ambiguity** remains the core limitation preventing reliable **6 DOF estimation** in real-world applications.

Conclusion

In ***"Towards Fast Fiducial Marker with Full 6 DOF Pose Estimation"***, Ulrich and colleagues present a circular fiducial marker system that balances speed, accuracy, and full pose recovery. By enhancing WhyCode's "necklace" encoding and resolving pose ambiguity through projected-center matching and statistical validation, the system delivers ∼0.9 ms/frame performance—approximately 30× faster than conventional square markers like AprilTag or ArUco—without compromising pose precision.

2.2.3 Frappe: Fast Fiducial Detection on Low-Cost Hardware

The Frappe algorithm is designed to offload as much of the detection process as possible onto the VideoCore IV GPU of the Raspberry Pi Zero. It leverages both the QPU (Quad Processing Units) and VPU (Vector Processing Unit), typically underutilized in most applications, to parallelize image processing tasks. The algorithm achieves over 60 frames per second at 640×480 resolution — a 5× speedup compared to standard ArUco detection — while reducing energy consumption by more than 75Key features of Frappe include: Tile-based image processing: Breaks down the image into 64×32 tiles to parallelize operations across multiple QPUs.

GPU-accelerated edge and corner detection: Uses Sobel filters and Shi-Tomasi methods implemented directly on the GPU.

Perspective correction and decoding: Warps candidate marker regions to a standard form and decodes the binary payload with hardware interpolation and minimal CPU involvement.

Zero-copy memory access: Uses shared memory buffers accessible by both CPU and GPU, eliminating data transfer bottlenecks. [3]

Performance Evaluation and Metrics

The authors evaluated **Frappe**, a GPU-accelerated fiducial detection system, against the standard **OpenCV ArUco** library in both standalone and embedded environments. The performance comparison was carried out using two benchmark video sequences:

- **Office Sequence** — Captured indoors under varying lighting conditions and cluttered backgrounds.
- **Arena Sequence** — Captured using a mobile robot navigating in a controlled testbed with both large and small fiducial markers.

Results Summary

- **Detection Speed:** Frappe achieves an average processing time of approximately **13 ms per frame**, significantly faster than ArUco's **64–169 ms**, depending on detection mode.
- **Corner Accuracy:** Both detectors achieved sub-pixel accuracy, with a **mean absolute error (MAE)** of less than **0.16 pixels**.
- **Energy Efficiency:** Frappe consumes approximately **31 millijoules (mJ) per frame**, in contrast to ArUco's **142 mJ per frame**, demonstrating a significant reduction in power consumption.
- **Navigation Success Rate:** Due to its ability to support higher resolution and faster frame rates, Frappe enabled a **3.5× larger successful operating area** during robot navigation tasks, compared to ArUco.

The **contour tracing step**, typically one of the most computationally expensive parts of marker detection, was retained on the CPU. However, its latency was significantly reduced by implementing a **GPU-assisted mask**, which allowed the system to skip non-edge regions during processing.

Conclusion

The paper *"Frappe: Fast Fiducial Detection on Low-Cost Hardware"* by Jones & Hauert introduces a GPU-accelerated algorithm specifically designed to run on Raspberry Pi Zero platforms. Frappe leverages the onboard VideoCore GPU to accelerate ArUco tag detection and decoding. On 640×480 streams, it achieves frame rates exceeding 60 Hz, about 5× faster than the standard ArUco library, while maintaining comparable accuracy and reducing energy consumption. Real-world testing on a DOTS robot demonstrates improved navigation performance, making Frappe a powerful option for real-time marker tracking in low-cost robotic systems.

2.3 Rationale for Using AprilTags

AprilTags were selected over other fiducial marker systems due to their high robustness, superior error correction capabilities, and reliable pose estimation performance, especially in challenging environments such as underwater. Underwater conditions are typically characterized by low visibility, image blur, variable lighting, and turbidity, all of which can significantly affect the performance of computer vision-based systems. By design, AprilTags are more resilient to such distortions compared to ArUco markers. Given that the focus of this project is on real-time, vision-based localization, it was crucial to utilize a marker system that offers high detection accuracy and robustness. AprilTags were found to fulfill these criteria more effectively than ArUco markers. AprilTags were selected over other fiducial marker systems due to their high robustness, superior error correction capabilities, and reliable pose estimation performance [4].

2.3.1 Advantages of AprilTags in Underwater Environments

AprilTags offer several features that make them more suitable for underwater localization compared to other fiducial markers. The following points highlight key advantages:

1. **High Contrast and Bold Geometry**

AprilTags are composed of thick black borders with high-contrast white regions and a large clear margin (also known as the "white border" around the marker), making them easier to detect in turbid or low-light underwater environments. Their corner features and cell boundaries are well-defined, so even if the image is slightly blurred due to water movement or low focus, AprilTags remain distinguishable.

2. **Error-Tolerant Encoding**
   AprilTags use longer Hamming distances between valid tag IDs. This provides strong error-correction capabilities, meaning that even if parts of the tag are obscured or degraded, the detection algorithm can often still recover the correct ID. This is critical underwater, where image data is frequently affected by reflections, scattering, or murkiness.

3. **Better Performance in Blur**
   Water naturally introduces optical distortion and reduces image sharpness. AprilTags are designed to be resilient under such conditions, with improved performance in the presence of motion blur and perspective distortion due to their precise corner localization and edge refinement algorithms.

4. **Improved Detection Algorithms**
   The AprilTag detection pipeline often uses gradient-based edge detection rather than simple thresholding. This allows for more robust detection in underwater scenes with poor lighting or reduced contrast, where traditional detection methods like those used in ArUco might fail.

2.3.2 How AprilTags Are Better Than ArUco Markers

Criteria	**AprilTags**	**ArUco Markers**
Detection Robustness	High: Better under blur, occlusion, and distortion	Moderate: May fail in noisy or blurred images
Error Correction	Uses Hamming codes with larger distances (e.g., tag36h11) → fewer false positives	Limited error correction → prone to false positives or ID confusion
Marker Design	More complex encoding with well-separated IDs → unique and reliable	Simpler structure; higher chance of ID collision
Detection Accuracy	Higher due to better edge localization and precise corner detection	Lower in high-noise or distorted frames
False Detection Rate	Extremely low due to robust coding and strict decoding criteria	Higher chance of false positives in cluttered scenes
Pose Estimation	More consistent pose output due to sub-pixel accuracy in detection	Works well, but prone to instability under distortion
Library Support	Widely used in robotics (e.g., ROS, SLAM, wrappers for OpenCV)	Fully supported in OpenCV (`cv2.aruco`)

Table 2.1: Comparison of AprilTags and ArUco Markers Based on Detection and Performance Criteria

2.4 Pose Estimation using PnP

2.4.1 What is PnP?

The **Perspective-*n*-Point (PnP)** problem involves estimating the *pose* (rotation and translation) of a calibrated camera from n correspondences between 3D points in the world and their 2D image projections. This is a key technique in computer vision and robotics, especially in applications like localization using fiducial markers (e.g., AprilTags).[13]

2.4.2 Matrix Representation of the Camera Model

The pinhole camera model relates 3D world points to 2D image points using the following equation:

$$s \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \cdot \begin{bmatrix} \mathbf{R} \mid \mathbf{t} \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Where:

- (X, Y, Z): 3D coordinates of a point in object space.
- (u, v): Corresponding 2D pixel coordinates in the image.
- s: Scale factor due to projective geometry.
- $\mathbf{K}$: Intrinsic camera matrix, defined as:

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

- $\mathbf{R} \in SO(3)$: Rotation matrix.
- $\mathbf{t} \in R^3$: Translation vector.

2.4.3 Homogeneous Transformation Matrix

The pose of the camera can also be represented as a homogeneous transformation matrix:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3)$$

2.4.4 Reprojection Error and Optimization

To estimate $\mathbf{R}$ and $\mathbf{t}$, the PnP algorithm minimizes the **reprojection error**:

$$E = \sum_{i=1}^{n} \|\mathbf{x}_i - \pi\left(\mathbf{K}(\mathbf{R}\mathbf{X}_i + \mathbf{t})\right)\|^2$$

Where:

- $\mathbf{x}_i$: Observed 2D point in the image.
- $\mathbf{X}_i$: Corresponding 3D point in object space.
- $\pi(\cdot)$: Projection operation converting 3D to 2D by dividing by the depth (Z coordinate).

This is a nonlinear least squares problem, typically solved using iterative methods such as the Levenberg–Marquardt algorithm.

2.4.5 Distance Estimation

Once the translation vector $\mathbf{t} = [t_x, t_y, t_z]^T$ is obtained, the Euclidean distance between the object and the camera is given by:

$$d = \|\mathbf{t}\| = \sqrt{t_x^2 + t_y^2 + t_z^2}$$

2.4.6 Summary

The PnP algorithm provides a reliable solution for estimating the 6-DoF pose of an object from known 3D-2D correspondences. The algorithm uses the intrinsic camera matrix and iteratively solves for the rotation and translation that minimize the reprojection error, enabling accurate localization and tracking in real-world applications.

2.5 Project Objectives

1. **Develop a robust fiducial marker detection system**
   Detect AprilTags in the environment to extract precise localization parameters.
2. **Implement the detection pipeline on an embedded platform**
   Deploy the system on a Raspberry Pi running BlueOS to achieve efficient real-time processing.
3. **Utilize dual USB cameras with calibration**
   Improve spatial perception by using two cameras and applying lens distortion correction to enhance localization accuracy.
4. **Transmit localization data to the Pixhawk flight controller**
   Establish a stable communication channel to provide real-world position estimates to the Pixhawk.
5. **Visualize the test area and trajectory**
   Create a Matplotlib-based visualization tool to plot the UUV's trajectory, enabling analysis of localization accuracy and overall system performance.

2.6 Project Methodology

To meet the defined objectives of this project, a systematic hardware-software integration strategy was followed. Each objective is addressed through a specific set of implementation steps as outlined below:

2.6.1 Develop a robust fiducial marker detection system

To enable accurate localization of the unmanned underwater vehicle (UUV), a vision-based system was implemented using Python, OpenCV, and the AprilTag detection package. AprilTags are 2D barcoded fiducial markers that are highly robust to variations in scale, perspective distortion, and lighting, making them ideal for structured indoor or controlled underwater environments. Each tag contains a unique binary code embedded in a black-and-white square, allowing the system to distinguish between multiple markers in the scene. The spatial positions of the AprilTags were predefined, with fixed coordinates assigned to each marker in the global reference frame. This allowed the system to accurately estimate the vehicle's position relative to the known tag locations, ensuring reliable and repeatable localization throughout the test environment.

The detection pipeline begins with capturing real-time frames from a USB camera mounted on the UUV. These frames are processed in a multithreaded Python script to ensure minimal latency and maximum throughput. Multithreading allows parallel execution of image capture and processing threads, thereby increasing the frame rate and reducing lag in the pose estimation output — a critical factor in dynamic navigation systems.

Once a frame is received, the AprilTag detector is invoked to scan the image for valid tag patterns. Upon successful detection, the corners of the tag are identified with sub-pixel accuracy, and the unique tag ID is decoded. With known intrinsic camera parameters (focal length, optical center, distortion coefficients), the 3D pose of the marker is calculated using the Perspective-n-Point (PnP) algorithm. This involves solving the geometric relationship between the known 3D coordinates of the tag's corners and their 2D projections in the image plane.

The result of this estimation includes:

- **Translation Vector (t):** Represents the position of the tag in the camera frame as coordinates (x, y, z).
- **Rotation Matrix (R):** Represents the orientation of the tag relative to the camera, which can be converted into Euler angles (roll, pitch, yaw).

By inverting the transformation, the position and orientation of the UUV (cam-

era) relative to the tag are derived, enabling accurate localization within the environment. This 6 Degrees of Freedom (6 DOF) information is fundamental to real-time navigation, control, and mapping.

Several practical considerations were addressed to improve detection robustness. The system was tested under varying conditions, such as changes in lighting intensity, distances ranging from 0.2 to 1.5 meters, and different angles of tag orientation. Tests were also conducted with multiple tags placed in a grid pattern to validate simultaneous multi-tag detection and ID differentiation.

The implementation also accounts for false positives and detection jitter. By introducing temporal filtering (such as exponential smoothing) across frames, the system ensures stable and continuous localization data even when individual frames contain detection noise or minor errors.

Overall, this vision-based approach to localization offers a low-cost, high-precision alternative to conventional underwater positioning systems, especially in confined or GPS-denied environments. AprilTags serve as passive landmarks that require no power or communication, making them ideal for long-term deployment in underwater test tanks or research setups.

2.6.2 Implement the detection pipeline on an embedded platform

To ensure real-time operation in underwater environments, the entire AprilTag detection pipeline was implemented on an embedded computing platform—specifically the Raspberry Pi 4 Model B. This compact, energy-efficient microprocessor was selected for its quad-core ARM Cortex-A72 CPU, 4–8 GB of RAM, and a range of I/O interfaces, making it suitable for robotics applications where onboard processing is critical.

The Raspberry Pi runs BlueOS, a specialized Linux-based operating system developed for marine vehicles. BlueOS is designed to simplify underwater robotics development by offering a browser-accessible dashboard for managing software modules, telemetry, networking, and diagnostics. It supports automatic device recognition, MAVLink routing, and containerized applications via Docker, making

it ideal for seamless integration into UUV systems.

The AprilTag detection pipeline was written in Python using the `apriltag` and `OpenCV` libraries. The detection script is deployed as a standalone process within the BlueOS environment. It continuously captures image frames from one or more USB cameras connected to the Raspberry Pi via the onboard USB 3.0 interface. A multithreaded architecture is employed to handle the separation of concerns: one thread is dedicated to video acquisition, another to AprilTag detection, and a third handles pose estimation and data transmission. This structure reduces bottlenecks and improves the frame processing rate significantly compared to a sequential implementation.

The use of multithreading is particularly important given the constrained computational resources of embedded systems. It allows the system to achieve near real-time performance, with frame rates ranging between 10–20 FPS depending on resolution and the number of visible tags. This is crucial for applications involving moving platforms, where delayed or dropped frames can cause inaccurate localization or control lag.

In addition to core processing, the BlueOS interface provides essential tools for debugging and tuning the detection algorithm. Developers can monitor camera feeds, adjust exposure or resolution settings, and review console output directly through a web browser. The modularity of BlueOS also allows the AprilTag detection service to be managed independently, restarted on-the-fly, or extended with additional features such as data logging or filtering.

Furthermore, the system is designed to be headless after deployment, meaning that once powered on, the Raspberry Pi automatically launches the detection pipeline and begins processing data without requiring manual intervention. This makes the setup highly practical for field testing and deployment in remotely operated underwater vehicles (ROVs) or autonomous underwater vehicles (AUVs).

Overall, implementing the AprilTag detection pipeline on the Raspberry Pi 4 with BlueOS provides an efficient, self-contained, and scalable solution for onboard visual localization. It strikes a balance between performance and portability, and offers an excellent foundation for real-time control and navigation in GPS-denied underwater environments.

2.6.3 Utilize dual USB cameras with calibration

To improve spatial accuracy and depth perception in localization tasks, a stereo vision setup is employed using two identical USB cameras mounted at a fixed baseline distance. This dual-camera configuration allows for the extraction of 3D geometric information from 2D images by analyzing disparity between the left and right camera views. In underwater or confined environments where depth sensors may be unreliable or unavailable, stereo vision offers a robust and passive alternative for enhancing localization accuracy.

Before the stereo system can be used effectively, both cameras must be carefully calibrated. Calibration consists of two main stages: **intrinsic calibration** for each individual camera and **extrinsic calibration** for the stereo pair. Intrinsic calibration accounts for lens distortion, focal length, and optical center, which are necessary to convert raw image coordinates into real-world measurements. This is done using a well-known technique involving the capture of multiple images of a standard checkerboard pattern from different angles and distances. The OpenCV library provides reliable functions such as `cv2.findChessboardCorners` and `cv2.calibrateCamera` for this purpose.

Once intrinsic parameters are estimated, stereo calibration is performed to determine the rotation and translation between the two cameras. This extrinsic calibration enables the construction of rectification transforms, which align the left and right image planes so that corresponding points lie on the same horizontal line. The result is a pair of rectified images that can be used to compute a disparity map using stereo matching algorithms like block matching or semi-global block matching.

The calibration process yields essential outputs such as:

- **Camera matrix:** Encodes focal length and optical center.
- **Distortion coefficients:** Used to correct radial and tangential lens distortion.
- **Rotation and translation vectors:** Define the position and orientation of the cameras relative to each other.

- **Rectification transforms and projection matrices:** Required for image alignment and 3D point triangulation.

All incoming frames from the USB cameras are passed through an undistortion and rectification pipeline before being used for AprilTag detection. This preprocessing step ensures that the tag corners are detected with high geometric fidelity, which in turn improves the accuracy of the Perspective-n-Point (PnP) solution for pose estimation.

The stereo setup also introduces redundancy in detection. In cases where one camera temporarily loses sight of the marker due to occlusion, blur, or lighting effects, the other camera may still retain visual contact, ensuring continued localization without interruption. This fault-tolerant design significantly improves the robustness of the system in real-world underwater conditions where visibility is often inconsistent.

By integrating dual USB cameras with distortion correction and stereo processing, the system achieves higher localization precision and better resilience, both of which are critical for stable underwater navigation and control.

2.6.4 Transmit localization data to the Pixhawk flight controller

To enable effective closed-loop control of the underwater vehicle, the computed localization data, derived from AprilTag detection, is transmitted from the Raspberry Pi to the Pixhawk 2.4.8 flight controller. This data transfer is accomplished via a hardware UART (Universal Asynchronous Receiver/Transmitter) interface using the Raspberry Pi's GPIO pins and the Pixhawk's TELEM2 port. This setup establishes a low-latency serial communication link that supports full-duplex data exchange between the two devices.

The communication is handled using the MAVLink (Micro Air Vehicle Link) protocol, a lightweight and efficient message-passing system widely used in autonomous aerial and marine robotics. MAVLink enables structured transmission of pose data, including 3D coordinates and orientation angles (roll, pitch, yaw), as well as system health status, sensor readings, and command instructions. On

the Raspberry Pi, the `pymavlink` Python library is used to construct and send MAVLink messages in a format compatible with the Pixhawk's firmware.

The Pixhawk runs ArduSub firmware—a specialized version of ArduPilot designed for underwater vehicles. It receives the incoming pose data via the TELEM2 port and treats it as external navigation input. To ensure that the Pixhawk correctly interprets and utilizes this data, a thorough calibration process was performed. This included accelerometer calibration, compass calibration, level horizon setup, and electronic speed controller (ESC) calibration. These steps were essential to align the Pixhawk's internal orientation and control logic with the physical configuration of the UUV. The calibration was carried out through the QGroundControl interface, which guided each step interactively and confirmed proper hardware responses before finalizing the setup.

In addition to receiving pose updates, the Pixhawk continuously transmits internal telemetry, such as IMU data, battery voltage, and pressure sensor-based depth readings, back to the Raspberry Pi or to the ground control station. This two-way communication enables real-time data synchronization and redundancy.

For mission supervision and live monitoring, the system integrates with QGroundControl (QGC), a ground station software that interfaces with the Pixhawk over MAVLink. QGC displays the vehicle's pose in 3D space, including heading, depth, and orientation, and provides mission planning tools, parameter tuning, and live system diagnostics. It also reflects feedback from the external localization system, allowing the operator to observe the UUV's real-time trajectory and intervene if necessary.

This entire setup forms a closed-loop control architecture in which the Raspberry Pi serves as the high-level perception and computation module, and the Pixhawk handles low-level motor actuation and sensor fusion. This continuous exchange of data allows for dynamic trajectory updates, correction of drift, and more stable navigation in GPS-denied environments such as underwater test tanks or confined aquatic structures.

To ensure safety and electrical compatibility, the Raspberry Pi is powered independently via USB or through a dedicated Battery Eliminator Circuit (BEC). All communication lines are appropriately isolated or voltage-shifted to protect both the Raspberry Pi and the Pixhawk hardware. With this robust and calibrated

communication infrastructure in place, the vision-based localization system plays a direct and reliable role in navigating and stabilizing the underwater vehicle.

2.6.5 Visualize the test area and trajectory

In addition to computing localization data, it is essential to represent the movement and behavior of the unmanned underwater vehicle (UUV) visually to understand and validate system performance. To achieve this, a plotting and visualization framework was developed using the Python `Matplotlib` library. This framework takes position estimates generated during navigation, either from real-time AprilTag detection or from stored logs, and plots the UUV's path within a predefined coordinate system representing the test environment, such as a pool or tank.

The primary goal of this module is to aid in the interpretation of system accuracy and to provide a visual understanding of how closely the UUV follows the intended path. By overlaying detected trajectories against known marker positions or expected waypoints, the system allows developers and researchers to assess the consistency and reliability of the localization pipeline. Deviations between the expected and actual trajectory can reveal issues such as incorrect calibration, latency in detection, noisy camera input, or drift in pose estimation.

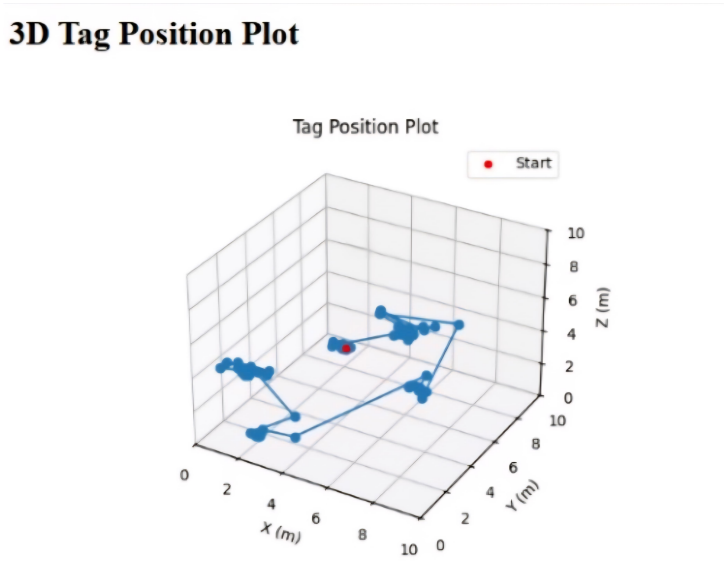

Figure 2.6: Example of a plot using Matplotlib

The framework supports both 2D and 3D plotting modes. In the 2D case, horizontal paths (X vs.Y) are plotted with respect to the test tank layout, while 3D plots incorporate the depth (Z-axis), making them suitable for scenarios involving complex underwater maneuvers. For advanced analysis, multiple trials can be overlaid in different colors to compare the system's response to varying conditions such as lighting changes, marker distance, and tag visibility.

The visualizer also serves as a diagnostic tool. Sudden jumps, inconsistent curvature, or missing data points on the plotted trajectory can indicate potential problems in the marker detection or pose estimation process. In some cases, the visualization output is augmented with marker IDs, orientation arrows, or timestamps to provide additional insights. This feedback loop is invaluable when tuning parameters such as detection thresholds, frame rates, or filtering algorithms.

Overall, plotting and visualizing the test area is a critical component in system validation and debugging. It complements numerical performance metrics by providing an intuitive understanding of how well the localization system functions in practice, enabling iterative improvement and fine-tuning of the underwater navigation pipeline.

2.6.6 Block Diagram Overview

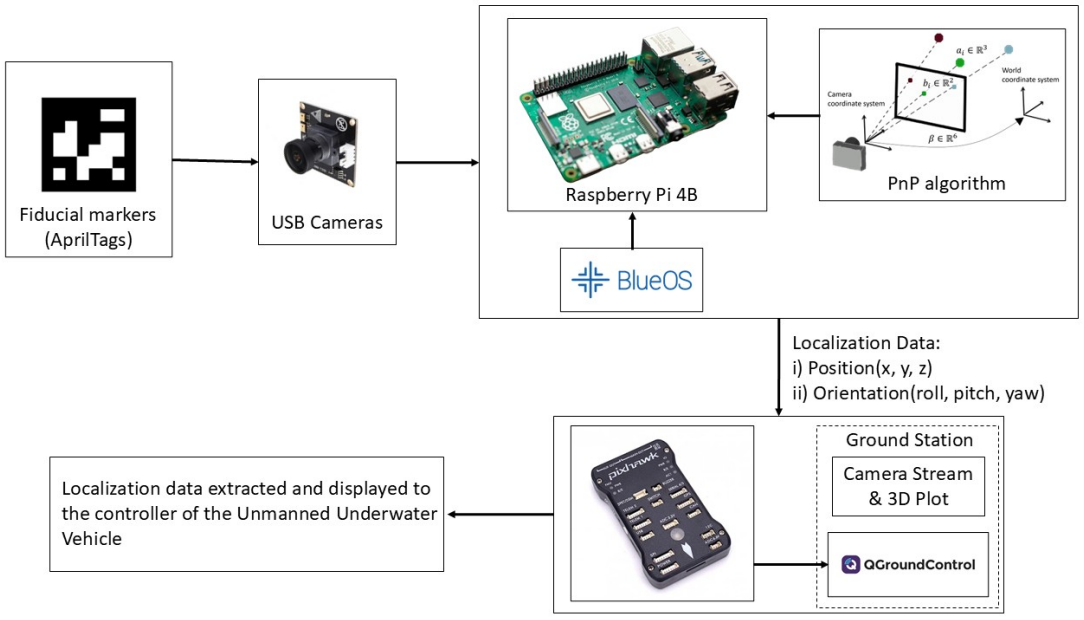

Figure 2.7: Block Diagram for Visual Localization and Control

The diagram above illustrates the end-to-end architecture of the visual localization system implemented for the Unmanned Underwater Vehicle (UUV). It outlines how data flows from the physical environment to the control and visualization interface.

The process begins with the detection of fiducial markers (AprilTags) placed in the environment. These markers act as visual landmarks, each encoded with a unique ID. USB cameras mounted on the UUV capture continuous visual input at a resolution of 2 megapixels and a frame rate of 30 FPS. These camera feeds are directed to the onboard computing unit, a Raspberry Pi 4 Model B.

The Raspberry Pi 4B runs BlueOS, a lightweight, web-accessible operating system tailored for marine robotics. The Pi performs all image processing using Python and OpenCV, and detects AprilTags in real time. Once a tag is detected, the PnP (Perspective-n-Point) algorithm is used to estimate the camera's pose relative to the marker. This results in precise localization data, including position (x, y, z) and orientation angles (roll, pitch, yaw).

The computed pose information is packaged and transmitted using the MAVLink protocol to the Pixhawk 2.4.8 flight controller via the TELEM2 port. The Pixhawk interprets this data and incorporates it into its control logic, enabling the UUV to adjust its trajectory based on external visual feedback.

Simultaneously, the localization data is forwarded to the ground station, where it is visualized using QGroundControl. This includes a real-time 3D plot of the UUV's position, orientation, and live camera feed, offering operators an intuitive interface for mission monitoring and control.

This closed-loop system allows for continuous visual navigation, making it well-suited for GPS-denied environments such as underwater test tanks or confined aquatic areas. The integration of visual perception, onboard computation, flight control, and telemetry visualization ensures accurate and stable movement of the UUV.

Chapter 3

Implementation

3.1 Conceptual Design

The system consists of a Raspberry Pi as the central computer, running BlueOS, and connected to a flight controller (PX4) and cameras as sensors. The cameras detect the Fiducial Markers (AprilTags) placed at predefined locations, and the system calculates the camera's pose (position and orientation) relative to these markers.

This information is then sent to Pixhawk via MAVLink protocol. A ground station PC receives and displays camera streams, 3D plots of the vehicle's path, and coordinate data (XYZ and roll-pitch-yaw), enabling real-time monitoring and control of the system.

3.1.1 Hardware Components

Raspberry Pi 4

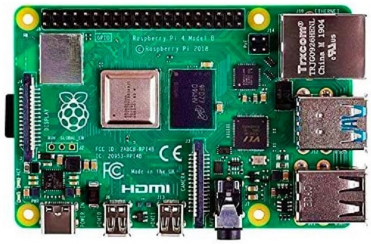

Figure 3.1: Raspberry Pi 4 Model B. [5]

Acts as the core processing unit for the system, the Raspberry Pi captures video input, performs AprilTag detection, and estimates the pose of the vehicle in real time. The Raspberry Pi 3B model was initially used, but it was found to be incompatible with BlueOS. As a result, the system was upgraded to the Raspberry Pi 4B, which also offers significantly better processing performance for vision tasks and multithreading.

Pixhawk 2.4.8 Flight Controller

Figure 3.2: Pixhawk 2.4.8 Flight Controller [6]

Responsible for executing motor commands and collecting sensor data. It provides real-time orientation data, roll, pitch, and yaw, of the UUV, which is displayed on the QGroundControl interface for monitoring and navigation assistance.

USB Camera

Figure 3.3: OV2710 2MP USB Camera used for AprilTag Detection [7]

Mounted on the UUV, the USB camera continuously captures visuals for AprilTag detection. The camera used operates at a resolution of 2 megapixels and streams video at a consistent 30 frames per second (FPS), providing high-quality input suitable for accurate pose estimation. The live feed is processed on the Raspberry Pi using a multithreaded Python script that significantly improves performance, reducing detection time by several milliseconds compared to a previous sequential implementation. This real-time processing capability is crucial for responsive navigation and maintaining localization accuracy in dynamic underwater environments.

Fiducial Markers (AprilTags)

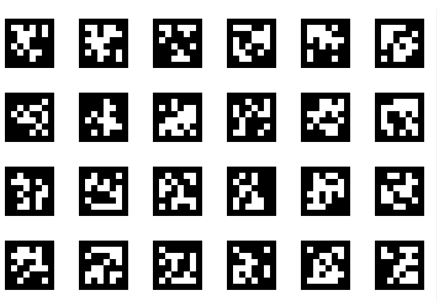

Figure 3.4: AprilTag used for Pose Estimation [8]

Fiducial markers used for localization in this project are based on the AprilTag family. These markers are placed in the environment to provide a known visual reference for the UUV.

Two physical tag sizes were used during testing:

- **Large Tag:** 17.5 cm × 17.5 cm — used for long-range detection or when mounted at fixed points in the testing environment.
- **Small Tag:** 5.5 cm × 5.5 cm — used for close-range navigation or precision testing.

These sizes were selected to balance detection accuracy, visibility underwater, and computational efficiency. AprilTags enable robust 6-DOF pose estimation and are resilient to lighting variation, partial occlusion, and blur.

UUV Frame

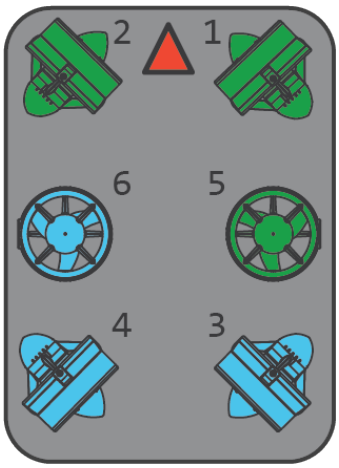

Figure 3.5: UUV Frame with Motor Mounts [9]

A robust and watertight chassis designed to house all onboard components and withstand underwater pressure. The frame is equipped with motor mounts and sensor housings.

6 BLDC Motors

Figure 3.6: Bidirectional Underwater Motor [10]

The UUV is equipped with six bidirectional **Brushless DC (BLDC) motors**, allowing it to maneuver in all directions—including forward/backward (*surge*), left/right (*sway*), and rotation about the vertical axis (*yaw*). These motors are arranged in a configuration that provides precise thrust control, enabling stable and responsive underwater navigation.

BLDC motors are electronically commutated motors that use a three-phase architecture and permanent magnets to generate rotation. Unlike brushed motors, they have no physical brushes, which reduces friction and increases durability—making them ideal for underwater applications where reliability and efficiency are critical. Their high torque-to-weight ratio and precise controllability are essential for maintaining the UUV's stability and executing complex maneuvers in confined aquatic environments.

Electronic Speed Controllers (ESCs)

Figure 3.7: Electronic Speed Controller (ESC) [11]

Translate the control signals received from Pixhawk into actual voltage/current commands for the motors, enabling precise speed and direction control.

3.1.2 Software Components

Python and OpenCV

Python and OpenCV were used for image acquisition and AprilTag detection in the localization system. The implementation leverages multithreading to enhance performance, significantly reducing the detection latency and improving the overall responsiveness during real-time navigation. The system was developed using Python 3.13.5 and OpenCV 4.12.0-Dev, ensuring compatibility with advanced computer vision libraries and efficient handling of high-resolution video streams. [15] [16]

Figure 3.8: Python and OpenCV

AprilTag Detection Package

Figure 3.9: AprilTag Detection Library [12]

A lightweight and robust tool for detecting fiducial markers and extracting their 6-DOF pose relative to the camera. Known for its high reliability in conditions with blur, occlusion, and variable lighting. [17]

NumPy Library

NumPy (v2.3) was essential for performing matrix operations involved in pose estimation, such as computing rotation matrices, Euler angles, and solving the PnP (Perspective-n-Point) problem [18].

Flask Server

Used to create a lightweight web interface for live monitoring and visualization. The Flask server streams video feeds from multiple USB cameras and renders a dynamic 3D plot of the UUV's position using routes. It enables real-time access via a localhost dashboard and supports refreshing plots, streaming MJPEG video,

and integrating multiple visual outputs through custom HTML templates. Flask is fully compatible with the Raspberry Pi and can be easily installed using pip. It runs as a lightweight local server, making it suitable for embedded applications such as onboard telemetry visualization and camera streaming on low-power hardware [19].

BlueOS

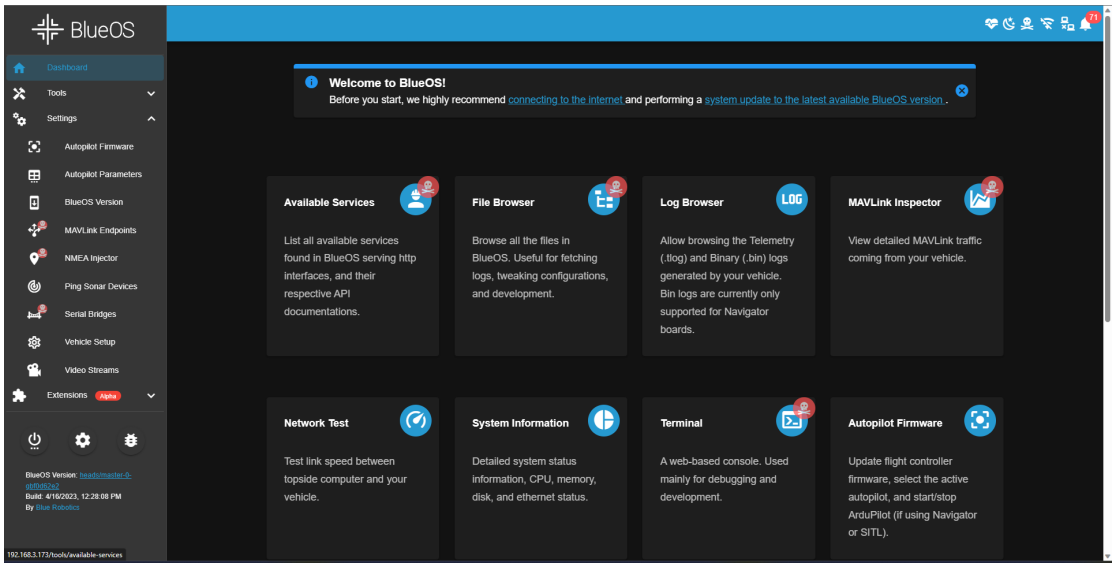

Figure 3.10: BlueOS Web Interface on Raspberry Pi

An operating system tailored for marine robotics, running on the Raspberry Pi. It enables configuration, telemetry, and system diagnostics via a browser-based dashboard. It also supports plug-and-play extensions and integrates well with MAVLink and other vehicle control frameworks. In this project, BlueOS v1.4.0 was used as the onboard operating system for managing communication and system-level tasks[20].

MAVLink Protocol

A lightweight communication protocol that facilitates message exchange between the Raspberry Pi and Pixhawk. It is used to transmit pose estimates and receive telemetry including IMU data and system status [21].

QGroundControl

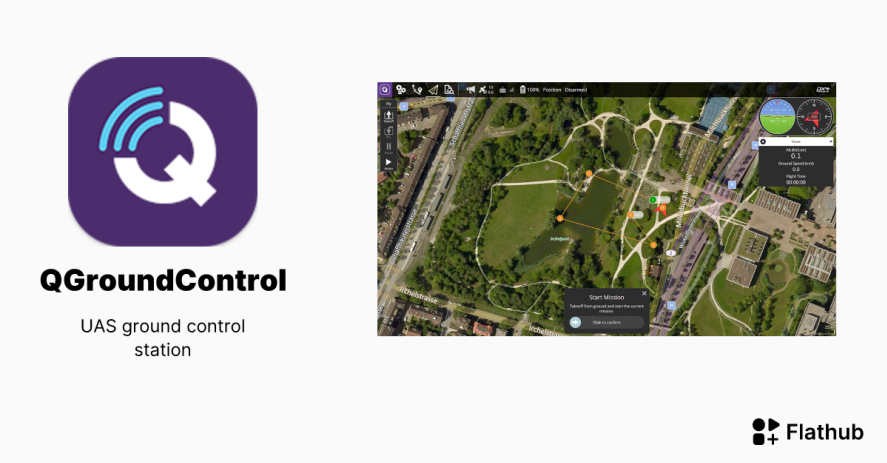

Figure 3.11: QGroundControl: Real-time UUV Monitoring Interface

A ground station software used to visualize telemetry data such as the UUV's roll, pitch, yaw, battery voltage, and position. It provides an intuitive real-time interface for monitoring and guiding the vehicle during missions. In this project, OGroundControl v4.0.0 was used for this purpose [22].

3.1.3 Test Area

Figure 3.12: Indoor test environment with AprilTags placed on the walls, from angle 1

Figure 3.13: Indoor test environment with AprilTags placed on the walls, from angle 2

The experimental setup was carried out in an indoor classroom environment with approximate dimensions of 6m× 10m× 3m(Width × Length × Height).

A total of five AprilTags were strategically mounted on the walls at coordinates that are predefined in the codes. These fiducial markers serve as reference points for visual localization and pose estimation tasks. The placement of tags was carefully chosen to cover different areas of the test space, ensuring consistent visibility for the onboard camera system.

The spacious layout, minimal obstructions, and fixed AprilTag positions provided a controlled and repeatable environment for evaluating the accuracy and responsiveness of the algorithm.

3.1.4 System Flow Description

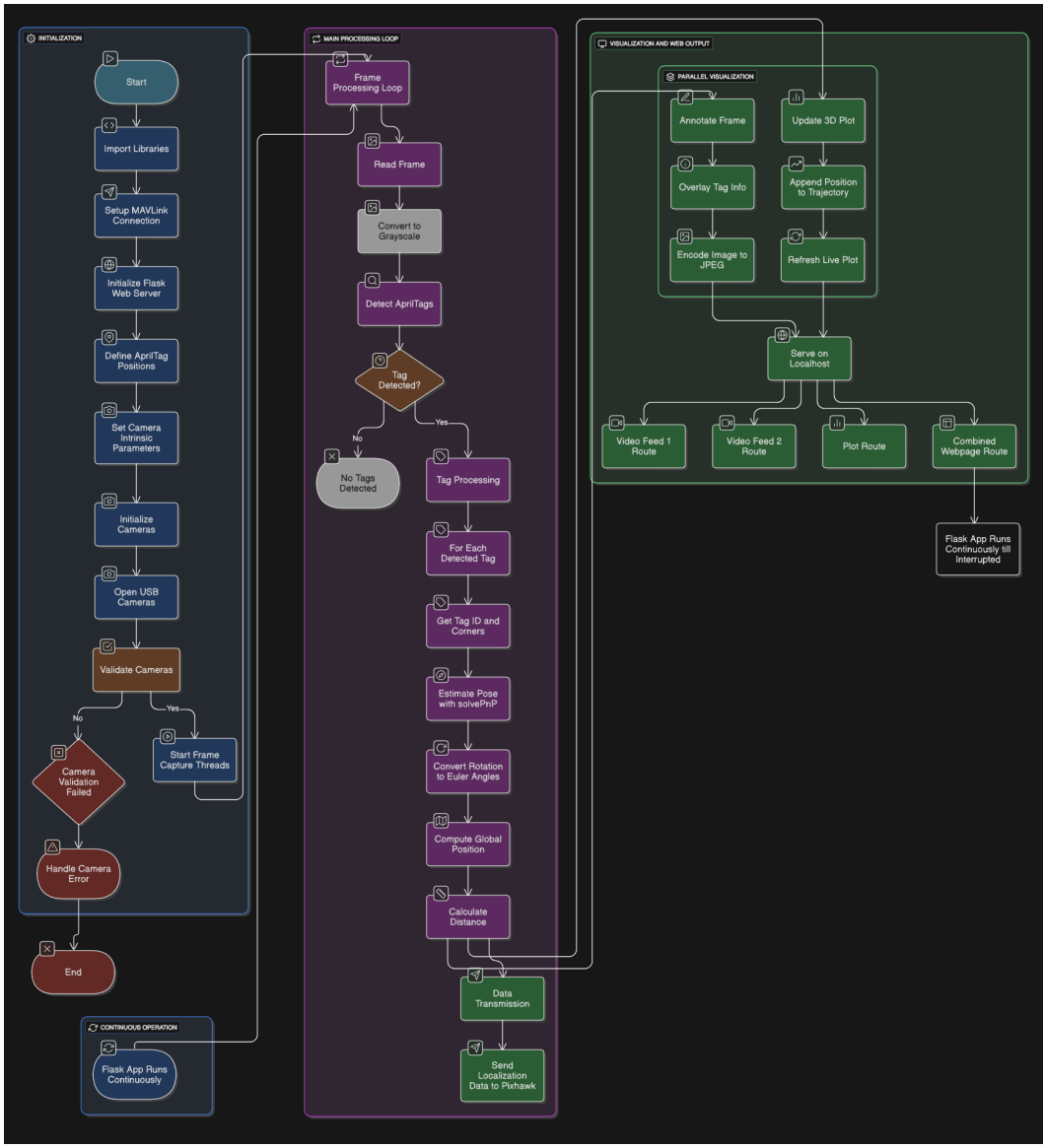

Figure 3.14: Flowchart of the UUV Localization and Visualization Pipeline

Figure 3.12 outlines the complete software pipeline for AprilTag-based localization, visualization, and data transmission in the Unmanned Underwater Vehicle (UUV) system. It is divided into three primary sections: Initialization, Main Processing Loop, and Visualization & Web Output.

1. Initialization: This stage prepares the system for operation. It begins by importing the necessary Python libraries, setting up the MAVLink connection to the Pixhawk, and initializing the Flask web server for live video streaming

and plotting. AprilTag positions are defined, and intrinsic camera parameters are loaded. USB cameras are opened and validated; if any errors occur (e.g., faulty connection or incorrect resolution), the system handles them gracefully and exits if needed. Upon successful validation, the system begins capturing frames in separate threads to allow real-time parallel processing.

2. Main Processing Loop: This core component continuously reads video frames from the camera feed. Each frame is converted to grayscale to optimize AprilTag detection. If a tag is detected, the tag's ID and corner coordinates are extracted. The pose is estimated using the `solvePnP` algorithm, and the resulting rotation matrix is converted into Euler angles (roll, pitch, yaw). The global position is computed relative to a defined world origin, and the distance to each tag is calculated. The localization data is then packaged and sent via MAVLink to the Pixhawk for navigation purposes.

3. Visualization and Web Output: In parallel with the main loop, visualization processes annotate the video frames by overlaying tag information. The UUV's trajectory is updated on a 3D plot, and the output is encoded and streamed via Flask. Multiple web routes serve individual video feeds, the real-time plot, and a combined interface—all accessible through a local browser. This real-time feedback loop enables intuitive monitoring and remote supervision.

Continuous Operation: Once initialized, the entire system runs continuously until interrupted manually. It maintains seamless integration between computer vision, navigation, and user interface components, ensuring efficient underwater localization and mission planning.

3.2 Algorithm: AprilTag-Based Pose Estimation and Visualization System

- **Import all required libraries:** OpenCV, NumPy, pupil_apriltags, pymavlink, Flask, threading, Matplotlib.
- **Initialize MAVLink connection** to Pixhawk via the Raspberry Pi's serial interface.
- **Start the Flask server** to serve video and plot data to a web browser.

- **Load camera calibration parameters:** intrinsic matrices and distortion coefficients.
- **Define 3D world positions** of all fiducial AprilTags in the test environment.
- **Open and validate two USB cameras.**
  - If either camera fails to initialize, terminate the program with an error message.
- **Start two frame capture threads** to allow concurrent image acquisition.
- **For each captured frame (from both cameras):**
  - Convert the image to grayscale.
  - Run AprilTag detection.
  - Filter detected tags based on tag IDs.
- **For each valid detected tag:**
  - Identify tag size (17.5 cm or 5.5 cm).
  - Estimate the 6DOF pose using the `solvePnP` algorithm.
  - Convert the rotation matrix into Euler angles (roll, pitch, yaw).
  - Use the known world coordinates of the tag to calculate the global position of the camera.
  - Send pose data (position and orientation) to Pixhawk via MAVLink.
  - Update the list of positions for trajectory plotting.
  - Annotate the frame with tag ID, position, and orientation data.
- **Plot and update** the 3D trajectory using Matplotlib and serve it via Flask.
- **Stream the annotated video feed and 3D plot** to a web browser on the local network.
- **Repeat the process continuously** until the program is interrupted.

3.3 Hardware Setup and Connectivity

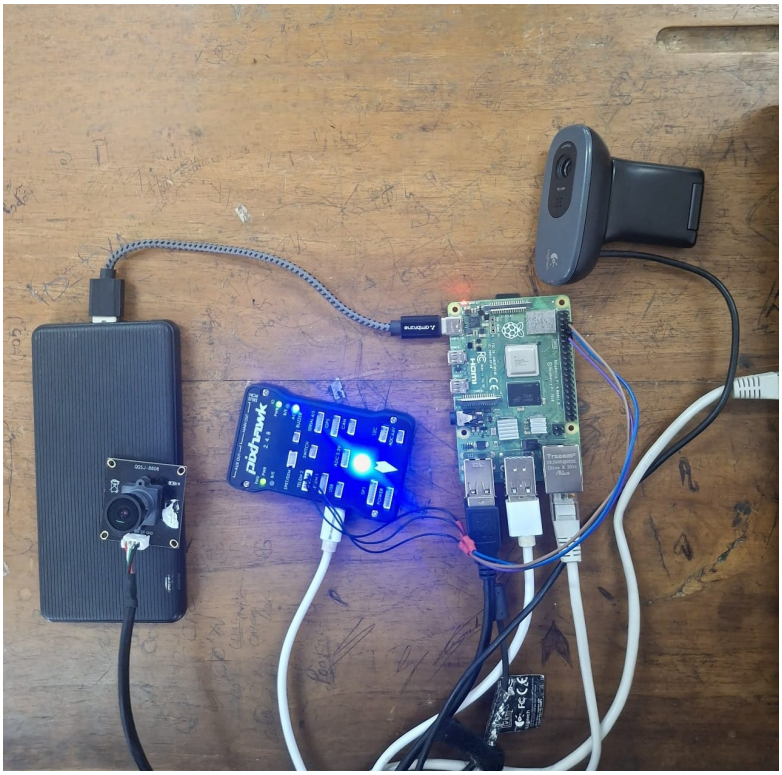

Figure 3.15: Hardware setup showing Raspberry Pi 4, Pixhawk 2.4.8, and USB cameras.

The figure above illustrates the basic bench-top setup used for system testing and development. The components are configured as follows:

- **Power Supply:** A dedicated power bank is used to supply stable power to the Raspberry Pi 4B via its USB-C port.
- **Raspberry Pi 4B:** Acts as the central onboard computer running BlueOS and the AprilTag-based localization algorithm. It also manages communication, camera input, and MAVLink data transmission.
- **Pixhawk 2.4.8:** It is powered via a USB connection from the Raspberry Pi, which also establishes a virtual serial port (e.g., /dev/ttyACM0). For dedicated communication, a UART link is created using direct wiring: Raspberry Pi TX (GPIO14) to Pixhawk RX, RX (GPIO15) to Pixhawk TX, and GND to GND. This setup enables reliable MAVLink-based transmission.

- **USB Cameras:** Two USB cameras were connected to the Raspberry Pi for stereo vision-based pose estimation using a multithreaded Python script. Although the original plan involved using two identical OV2710 2MP cameras, the Raspberry Pi faced device recognition issues, likely due to identical USB IDs. To resolve this, one camera was replaced with a different model, which allowed both streams to be detected. However, bandwidth limitations on the Raspberry Pi led to significant lag and latency in one of the video streams during real-time processing.
- **Ground Station:** The Raspberry Pi is accessed remotely over SSH from a laptop connected via Ethernet or Wi-Fi, enabling code deployment, monitoring, and live debugging.

This modular and compact setup facilitates easy testing of individual components and their integration, serving as the foundation for further development toward field-ready underwater deployment.

Chapter 4

Results and Discussion

4.1 Results

This chapter presents the design, implementation, and evaluation of a real-time localization system for an unmanned underwater vehicle (UUV) using fiducial markers. It details the development of an AprilTag-based detection pipeline, its deployment on an embedded Raspberry Pi platform, and the integration of a dual-camera setup with distortion correction. The chapter further explains how the pose data was transmitted to a Pixhawk flight controller via MAVLink protocol and visualized through a custom trajectory plotting tool. Each section highlights the system's robustness, accuracy, and suitability for GPS-denied indoor environments, validating its performance through field tests and real-time demonstrations.

4.1.1 Fiducial Markers Detection for Localization

A significant outcome of the project was the successful implementation of a fiducial marker detection system using AprilTags, a widely adopted marker system known for its robustness in pose estimation. The detection pipeline was developed using the `apriltag` Python module, and extensive testing was conducted to ensure reliability under varied environmental conditions including changes in lighting, angle, and partial occlusion. The system was capable of detecting multiple tags simultaneously and extracting their unique IDs along with their 6-DoF (Degrees

of Freedom) pose information — position (x, y, z) and orientation (roll, pitch, yaw). This achievement formed the foundation for accurate localization of the unmanned underwater vehicle (UUV) within a bounded test environment.

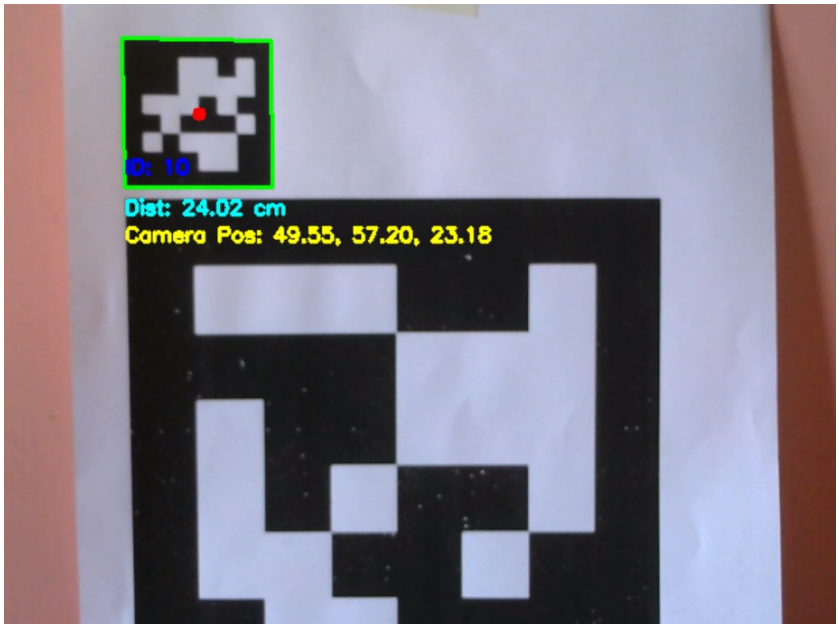

Figure 4.1: AprilTag detection and pose estimation output

4.1.2 Embedded Implemention of the Detection Pipeline

The AprilTag detection algorithm was successfully ported and executed on a Raspberry Pi 4 Model B, running the BlueOS middleware. Given the limited computational resources of embedded systems, the implementation involved optimizations such as downsampling of image frames, region-of-interest (ROI) processing, and multi-threaded execution to maintain efficiency. The final system was capable of processing video feeds from cameras at real-time frame rates (15–20 FPS), which is considered adequate for indoor localization tasks in slowly moving autonomous systems. Additionally, the system booted automatically on startup and remained stable during prolonged use, validating its readiness for field deployment.

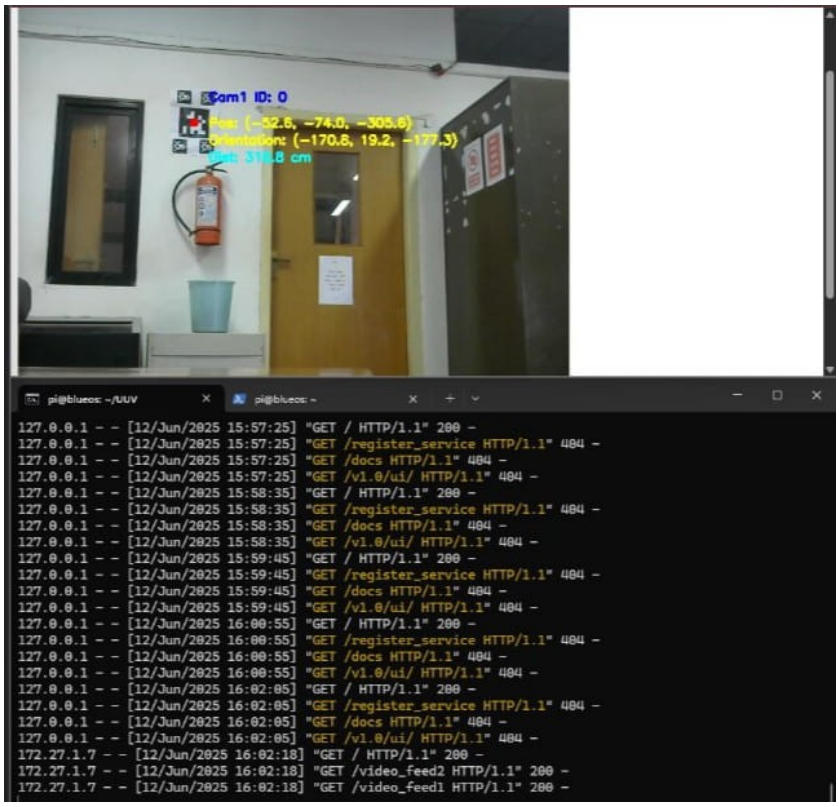

Figure 4.2: Detection pipeline running on Raspberry Pi with BlueOS

4.1.3 Dual-Camera Integration with Real-Time Distortion Correction

A dual-camera setup was configured using two synchronized USB webcams mounted in parallel. To ensure the geometric accuracy of spatial measurements, camera calibration was performed using OpenCV's calibration toolkit, which involved capturing chessboard images and estimating intrinsic parameters and distortion coefficients. These parameters were then used to undistort incoming frames in real-time. This correction significantly reduced radial and tangential distortion effects, which would otherwise introduce localization errors. While stereo depth estimation was not the primary focus, the dual-camera setup contributed to better field-of-view coverage and more stable tag tracking during vehicle motion.

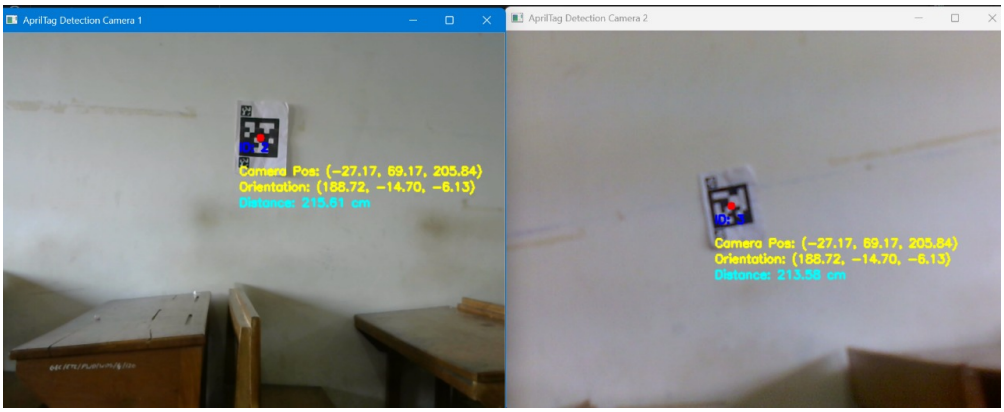

Figure 4.3: Dual camera and distortion correction output

4.1.4 MAVLink-Based Localization Data Transmission to Pixhawk

A critical integration step was the establishment of a communication channel between the Raspberry Pi and the Pixhawk flight controller. This was accomplished using the MAVLink protocol over serial communication (UART/USB). Custom MAVLink messages were constructed to package the tag-derived localization data, including position and orientation, and transmit them at a fixed update rate. The Pixhawk was configured to interpret this incoming data as external positioning input, allowing visualization of the UUV's location in tools like QGroundControl. Field tests confirmed that the pose estimates from the Raspberry Pi were accurately received and processed by the Pixhawk, enabling tight feedback loops for control and navigation algorithms.

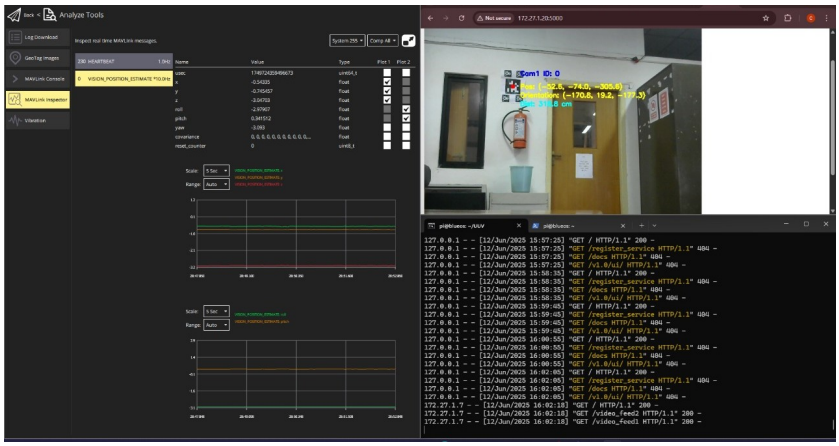

Figure 4.4: Localization data received by Pixhawk using MAVLink

4.1.5 Real-Time Plotting of Localization within Test Environment

To support both qualitative and quantitative evaluation of the system's performance, a custom visualization tool was developed using Python's Matplotlib library. This tool was capable of plotting the UUV's trajectory over time in a 2D coordinate frame, overlaid with fiducial marker locations and boundary outlines of the test environment. The system supported both live data streaming from the localization module and log file playback for post-test analysis. Additional features included path smoothing, error visualization, and performance metrics such as position drift and tag detection rate. These visual outputs proved invaluable in diagnosing system behavior and verifying localization precision across different trial runs.

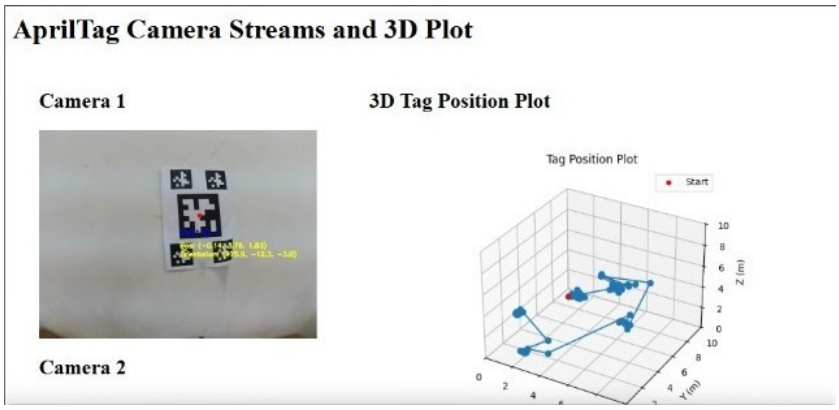

Figure 4.5: Trajectory plot showing localization accuracy within test area

4.2 Discussion

The experimental results obtained clearly demonstrate the feasibility of integrating a real-time AprilTag-based localization system onto an embedded platform for autonomous underwater navigation. The multi-objective implementation—spanning hardware calibration, embedded optimization, and inter-system communication—resulted in a well-synchronized pipeline suitable for UUV deployment in controlled environments.

Notably, the choice of AprilTags offered robust detection even in moderately

noisy conditions, and the use of dual cameras significantly enhanced tracking stability and reduced localization jitter. The successful transmission of external localization data to the Pixhawk flight controller via MAVLink allowed for external pose fusion, which is critical in scenarios where GPS is not available.

Finally, the trajectory visualization tool enabled effective analysis of system accuracy and error trends, helping identify drift and calibrate system parameters iteratively.

Chapter 5

Conclusion

5.1 General Conclusion

- **Effective Vision-Based Localization:** The project successfully demonstrated the use of fiducial markers, specifically AprilTags, for accurately determining the position and orientation of a camera in a predefined underwater-like environment. This highlights the feasibility of vision-based navigation in GPS-denied or confined spaces.
- **Accurate Pose Estimation:** By leveraging pose estimation algorithms such as `solvePnP`, the system was able to compute the camera's global position and orientation (roll, pitch, yaw) relative to known tag positions. The results confirm that the marker-based approach provides high-precision 6DOF localization when camera calibration and marker placement are properly configured.
- **Integration of Multi-Component System:** The project achieved seamless integration of hardware and software components including Raspberry Pi 4, dual USB cameras, Pixhawk 2.4.8 flight controller, and MAVLink communication protocol. The overall system architecture supports modularity, real-time performance, and robust data exchange.
- **Real-Time Monitoring and Visualization:** The implementation of a Flask-based interface allowed for live streaming of camera feeds and 3D visualization of computed trajectories. This feature enabled continuous mon-

itoring, validation of system performance, and improved interpretability of navigation data.

The complete implementation, including source code, configuration files, and documentation, is available on the project's GitHub repository [23].

5.2 Challenges

1. Accurate Real-Time Localization of Cameras Relative to Fiducial Markers

- **Challenge:** Ensuring precise localization of the camera with respect to fiducial markers in a dynamic underwater environment.
- **Solution:** Use high-resolution cameras and robust fiducial marker detection libraries (e.g., AprilTag). Apply real-time filtering techniques such as Kalman filters to reduce noise and improve localization consistency.

2. Dynamic Selection and Prioritization of Multiple Tag Sizes

- **Challenge:** Different tag sizes affect detection distance and reliability, requiring dynamic prioritization in detection.
- **Solution:** Implement an adaptive algorithm to select appropriate tag sizes based on the distance and field of view. Predefine tag sets based on use case (e.g., long-range vs close-range tags).

3. Finding and Updating Camera Parameters

- **Challenge:** Accurate pose estimation depends on precise intrinsic and extrinsic camera calibration.

- **Solution:** Use checkerboard or known fiducial grid patterns for rigorous camera calibration using OpenCV. Update calibration periodically to account for environmental effects such as temperature changes or physical shifts.

4. Using Translation Vectors for Absolute Position Calculation

- **Challenge:** Translating relative tag-to-camera positions into global coordinates can introduce cumulative errors.
- **Solution:** Use known world coordinates of AprilTags to transform relative pose into global position. Establish a consistent global reference frame tied to fixed tag positions.

5. Writing Code for Accurate Positioning in Global Coordinates

- **Challenge:** Ensuring the code properly integrates tag detection, calibration data, and coordinate transformations.
- **Solution:** Use reliable computer vision libraries like OpenCV and test each component rigorously. Validate the global position outputs through controlled trials and visual verification.

6. Dual Camera Interference and Synchronization Issues

- **Challenge:** Simultaneous usage of two USB cameras often leads to frame drop, inconsistent capture rates, or detection interference, especially when both cameras detect overlapping tags.
- **Solution:** Use multithreading to isolate camera streams and optimize processing rates. Carefully align fields of view to avoid redundancy, and selectively prioritize detections based on tag sets per camera.

7. Limitations of BlueOS for Real-Time Vision Applications

- **Challenge:** BlueOS, although designed for underwater platforms, was found to be underperforming for real-time image processing tasks and multi-camera applications.
- **Solution:** Consider hybrid setups where BlueOS handles telemetry and diagnostics, while a parallel vision node (e.g., running on bare Raspberry Pi OS or Docker container) handles camera input and processing. Optimize system resources to balance performance.

5.3 Future Work

- **Integration of Dual-Camera System:** The implementation will be extended to support simultaneous processing from two USB cameras. Leveraging stereo vision will improve depth perception, accuracy of localization, and robustness in challenging underwater environments.
- **BlueOS Interface Enhancement:** Future work includes embedding live camera streams, 3D plots of trajectory, and pose (position and orientation) data directly into the BlueOS web dashboard. This will allow operators to monitor all system outputs in real-time from a single, user-friendly interface.
- **Waterproofing and Environmental Adaptations:** All electronics including the Raspberry Pi, cameras, and wiring will be enclosed in waterproof housings to enable underwater deployment. Additional considerations such as pressure resistance, cable sealing, and thermal management will be addressed.
- **Assembly and Preliminary Testing:** The complete UUV system will be assembled and tested in controlled test tanks or shallow water environments. These tests will help validate software-hardware communication, motor response, and localization stability.
- **Underwater Field Trials:** Following successful lab testing, the UUV will be deployed in actual underwater environments. These trials will evaluate localization accuracy, real-time feedback stability, and system resilience under realistic operational conditions.

- **System Optimization and Feedback Loop:** Insights gained during field trials will be used to optimize both the hardware and software components. This may involve algorithm refinement, improvements in frame rate handling, better synchronization of dual-camera input, and calibration enhancements.

Bibliography

[1] Z. Xu, M. Haroutunian, A. J. Murphy, J. Neasham, and R. Norman, "An Underwater Visual Navigation Method Based on Multiple ArUco Markers," *Journal of Marine Science and Engineering*, vol. 9, no. 12, Art. no. 1432, 2021. [Online]. Available: https://www.mdpi.com/2077-1312/9/12/1432

[2] J. Ulrich, A. Alsayed, F. Arvin, and T. Krajník, "Towards fast fiducial marker with full 6 DOF pose estimation," in *Proc. 37th ACM/SIGAPP Symp. on Applied Computing (SAC '22)*, Virtual Event, 2022, pp. 723–730. doi: https://dl.acm.org/doi/abs/10.1145/3477314.3507043

[3] T. Jones and S. Hauert, "Frappe: Fast Fiducial Detection on Low-Cost Hardware," *Sensors*, vol. 23, no. 2, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s11554-023-01373-w

[4] E. Olson, "AprilTag: A robust and flexible visual fiducial system," *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3400-3407

[5] Available: https://www.adafruit.com/product/4295

[6] Available: https://www.tradeindia.com/products/pixhawk-32-bit-flight-controller-2-4-8-for-drone-quadcopter-c6097559.html

[7] Available: https://www.waveshare.com/ov2710-2mp-usb-camera-a.htm

[8] Available: https://docs.cbteeple.com/robot/april-tags

[9] Available: https://www.ardusub.com/quick-start/vehicle-frame.html

[10] Available: https://vayuyaan.com/shop/quadcopter/1800kv-bldc-motor-brushless-dc-motor-for-drone/

[11] Available: https://www.tytorobotics.com/blogs/articles/what-is-an-esc-how-does-an-esc-work

[12] Available: https://pyimagesearch.com/2020/11/02/apriltag-with-python/

[13] R. Qiao, G. Xu, P. Wang, Y. Cheng, and W. Dong, "An accurate, efficient, and stable Perspective-n-Point algorithm in 3D space," *Applied Sciences*, vol. 13, no. 2, Art. no. 1111, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/2/1111

[14] C. Zhao, C. Li, and J. Li, "An Underwater Visual Navigation Method Based on Multiple ArUco Markers," *Journal of Marine Science and Engineering*, vol. 9, no. 12, 2021. https://doi.org/10.3390/jmse9121432

[15] Python Software Foundation, "Python 3 Documentation," 2024. [Online]. Available: https://docs.python.org/3/

[16] OpenCV Team, "OpenCV: Open Source Computer Vision Library," 2024. [Online]. Available: https://docs.opencv.org/4.x/index.html

[17] The AprilTag 3 visual fiducial system is available at https://github.com/AprilRobotics/apriltag

[18] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. Smith *et al.*, "NumPy Documentation," *NumPy Developers*, 2024. [Online]. Available: https://numpy.org/doc/

[19] Pallets Projects, "Installation — Flask Documentation (Stable)," Flask, 2025. [Online]. Available: https://flask.palletsprojects.com/en/stable/

[20] Blue Robotics, "BlueOS - An Operating System for Marine Robotics." [Online]. Available: https://blueos.cloud/docs/stable/usage/overview/

[21] ArduPilot Dev Team, "MAVLink Basics," *ArduPilot Developer Documentation*, 2025. [Online]. Available: https://ardupilot.org/dev/docs/mavlink-basics.html

[22] QGroundControl Developers, "QGroundControl: Ground Station Software." [Online]. Available: https://docs.qgroundcontrol.com/

[23] Available: https://github.com/sudoaryansin/Navigation-and-Localization-of-UUV-using-Fiducial-Markers.git