Embedded Vision for UUVs: A Robust AprilTag-Based Localization Framework

Aryan Singh, Chrisann Ferrao, Govind Pandey, Jayden Viegas, and Milind Fernandes
Dept. of Electronics and Telecommunication Engineering
Goa College of Engineering, Farmagudi, Ponda, Goa, India
Email: {aryan.singh.etc, chrisann.ferrao.etc, govind.pandey.etc, jayden.viegas.etc}@gec.ac.in

***Abstract*—Accurate localization for unmanned underwater vehicles (UUVs) in GPS-denied environments remains a significant challenge, often requiring expensive and complex hardware. This paper presents a complete, low-cost framework for real-time 6-DOF pose estimation by applying the Perspective-n-Point (PnP) algorithm to detections of AprilTag fiducial markers. The system is implemented on an embedded Raspberry Pi 4B, which processes visual data from an onboard camera, computes the vehicle's global pose, and transmits it directly to a Pixhawk flight controller via the MAVLink protocol for closed-loop control. A multithreaded software architecture was developed to optimize performance. Experimental validation was conducted with a dual-camera module to investigate redundancy; however, the results demonstrate that a single-camera stream provides robust, low-latency localization, as the embedded platform's I/O performance was identified as the primary bottleneck. Ultimately, this work provides a validated, end-to-end architecture that makes vision-based UUV navigation accessible for research and prototyping, establishing a scalable, open-source platform for future developments in autonomous inspection and control.**

***Index Terms*—Underwater robotics, visual localization, AprilTags, MAVLink, Raspberry Pi, GPS-denied navigation, embedded vision**

I. Introduction

Due to the growing demand for underwater exploration and research, there is an increasing need for simpler, less complex, and more cost-effective methods to aid in underwater navigation and localization. While traditional approaches such as Inertial Navigation Systems (INS), Doppler Velocity Logs (DVL), and acoustic positioning methods like Long Baseline (LBL) and Ultra Short Baseline (USBL) offer high accuracy, they are often expensive, technically complex, and resource intensive to deploy. These systems are generally suited for large scale, open sea operations, making them impractical for confined, shallow, or budget restricted applications.

GPS, which is the most common positioning tool on land and in aerial domains, is unusable underwater due to rapid signal attenuation. This further complicates localization, especially in environments where high end navigational aids cannot be justified or sustained.

To address these limitations, this paper explores a simpler, more accessible alternative: vision-based localization using fiducial markers. We specifically employ AprilTags, which leverage inexpensive imaging hardware and robust computer vision algorithms to achieve reliable six-degrees-of-freedom (6-DOF) pose estimation. This approach is particularly valuable not only for operational deployment in confined spaces like flooded mines but also for crucial development phases such as vehicle prototyping and testing. In these scenarios, where deploying complex navigational systems would represent an inefficient use of resources, our method offers a powerful alternative. This positions our work as a niche yet highly effective technique, providing a scalable and resource-efficient pathway to achieving robust localization and, eventually, navigation in GPS-denied settings.

II. Related Work

The challenge of accurate vehicle localization in GPS-denied environments is a well-established problem in robotics, particularly for Unmanned Underwater Vehicles (UUVs) and indoor mobile robots. While methods like Inertial Navigation Systems (INS) and Simultaneous Localization and Mapping (SLAM) are prominent, vision-based fiducial markers have emerged as a cost-effective, robust, and computationally efficient alternative. Our work builds upon a significant body of research focused on the application and optimization of these marker systems.

Research has demonstrated the viability of fiducial markers for precise underwater navigation. A notable example is the work by Xu et al. [1], which presents a visual navigation method for UUVs using multiple ArUco markers arranged in a grid on a towing tank floor. They effectively mitigate noise and drift by fusing pose data from all visible markers using a Mahalanobis-distance-based algorithm, achieving repeatable localization with sub-meter accuracy. Our project aligns with this fundamental approach of using a predefined array of markers within a controlled environment to establish a global reference frame. However, our work diverges in the choice of marker, opting for AprilTags due to their superior robustness to blur, occlusion, and lighting variations [2], which are critical factors in challenging visual conditions.

The performance of fiducial marker systems, especially on resource-constrained embedded platforms, is another active area of research. The trade-off between the robustness of square markers like AprilTags and the detection speed of circular markers is a key consideration. Ulrich et al. [3] address this by proposing a novel circular marker design that achieves full 6-DOF pose estimation at speeds nearly 30 times faster than conventional square markers, resolving the pose ambiguity that typically limits circular designs. Concurrently,

other research has focused on accelerating the detection of existing markers. Jones and Hauert developed "Frappe," a detection algorithm that offloads the computational workload to the GPU of a low-cost Raspberry Pi [4]. This approach yields a 5x speed improvement and a significant reduction in energy consumption for ArUco detection, making real-time processing more feasible on embedded systems.

Our work is positioned at the intersection of these research thrusts. While we do not propose a new marker or a novel GPU-accelerated algorithm, our contribution lies in the practical implementation and integration of a complete, end-to-end localization system tailored for a real-world control pipeline. By leveraging the robustness of AprilTags on a modern embedded platform (Raspberry Pi 4B with BlueOS [9]) and transmitting the computed pose data directly to a Pixhawk flight controller via MAVLink [10], we demonstrate a practical and accessible framework. This research validates the integration of off-the-shelf components into a cohesive system that bridges the gap between marker detection theory and its application in vehicle navigation and control.

III. System Methodology

The core of our system is a structured pipeline (Fig. 1) designed to convert raw visual information from an onboard camera into a continuous stream of 6-DOF pose data suitable for closed-loop vehicle navigation. This methodology is fundamentally based on solving the Perspective-n-Point (PnP) problem [5] using known visual landmarks. The process is divided into three key stages: robust pose estimation relative to a marker, transformation of this relative pose into a global coordinate frame, and real-time data transmission to the flight controller for state estimation.

A. AprilTag Detection and Pose Estimation

The primary visual localization process begins with the real-time detection of AprilTags in the camera's field of view. This stage leverages the pinhole camera model, which geometrically relates a 3D point in the world to its 2D projection on the image plane. For each captured frame, the algorithm executes the following sequence:

1) **Image Pre-processing:** The input RGB frame from the camera is first converted to a single-channel grayscale image. This step is critical for both efficiency and robustness. It reduces the computational load by simplifying the image data and makes the detection process less susceptible to variations in environmental lighting color, relying instead on stable luminance gradients.
2) **Tag Detection and Corner Localization:** A robust AprilTag detector (`pupil_apriltags`) scans the grayscale image. The library's algorithm performs quad detection and analyzes the binary payload within any identified quadrilaterals to validate them as AprilTags. Upon successful validation, the detector returns the tag's unique ID and, crucially, the precise 2D pixel coordinates (u, v) of its four corners with sub-pixel accuracy.
3) **Pose Calculation via PnP:** The Perspective-n-Point (PnP) problem is then solved to determine the camera's pose relative to the detected tag's local coordinate system. The inputs to this stage are the four 2D pixel coordinates of the corners and their corresponding 3D coordinates in the tag's local frame (e.g., a square of known size centered at the origin on the XY plane). This is an optimization problem that finds the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ that minimize the reprojection error—the squared distance between the observed image projections of the corners and the projected 3D corner points based on the estimated pose. The geometric relationship is described by:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (1)$$

where (X_w, Y_w, Z_w) are the known 3D coordinates of a tag corner in its local frame, (u, v) are its observed image coordinates, $\mathbf{K}$ is the camera's pre-calibrated intrinsic matrix, and s is a scaling factor. The output of OpenCV's `solvePnP` is a rotation vector (`rvec`) and a translation vector (`tvec`). The `rvec` is converted to the full 3x3 rotation matrix $\mathbf{R}$ via Rodrigues' formula. These vectors define the 6-DOF pose (position and orientation) of the tag *in the camera's coordinate frame*.

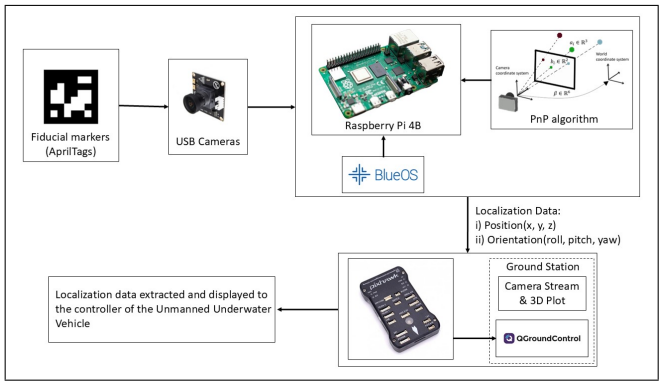

Fig. 1. The localization pipeline, from image capture to MAVLink transmission. The process involves tag detection, PnP pose estimation, transformation to a global frame, and packaging data for the flight controller.

B. Global Coordinate Transformation

A single tag detection only provides the vehicle's position relative to that specific marker. To achieve meaningful localization within the entire operating area, this relative pose must be converted into a consistent, predefined global coordinate frame. This is achieved by using a stored map of all AprilTag locations.

The transformation from the camera's relative view to a global pose is a critical step. From the PnP solution, we have the translation vector $\mathbf{t}$ (the position of the tag's origin in the camera's frame) and the rotation matrix $\mathbf{R}$ (the orientation of

the tag's axes in the camera's frame). The vector from the tag to the camera, expressed in the camera's frame, is $-\mathbf{t}$. To express this vector in the global frame, we must rotate it using the inverse of the camera's rotation relative to the world. Assuming the tag's orientation is aligned with the world frame, this rotation is simply $\mathbf{R}^T$. The camera's global position, $\mathbf{P}_{\text{cam}}$, can then be calculated relative to the tag's known global position $\mathbf{P}_{\text{tag}}$:

$$\mathbf{P}_{\text{cam}} = \mathbf{P}_{\text{tag}} + \mathbf{R}^T(-\mathbf{t}) \tag{2}$$

This calculation is performed for every detected tag, ensuring that each provides an independent, absolute position estimate of the vehicle within the world. This redundancy enhances system robustness and mitigates reliance on any single marker, allowing for smooth tracking even as the vehicle moves and different tags enter and leave the camera's view.

C. Data Transmission for Closed-Loop Control

The final stage of the methodology bridges the perception system with the vehicle's control system by transmitting the computed pose to the flight controller.

1) **MAVLink Message Formulation:** The calculated global position (x, y, z) and orientation, converted to Euler angles (roll, pitch, yaw), are packaged into a `VISION_POSITION_ESTIMATE` MAVLink message. This is a standardized message format in the MAVLink protocol [10], designed specifically for injecting external, vision-based navigation data into autopilots. The message payload includes the 3D position, 3D orientation, and a timestamp.
2) **Sensor Fusion via EKF:** The message is sent from the Raspberry Pi to the Pixhawk over a low-latency serial (UART) link. The flight controller's firmware (ArduSub) is configured to listen for this message. Upon receipt, its onboard Extended Kalman Filter (EKF) fuses this external vision data with its own internal high-frequency sensor readings (e.g., from its accelerometer and gyroscope). The EKF intelligently combines the non-drifting but lower-frequency vision data with the high-frequency but drifting IMU data to produce a smooth, stable, and drift-corrected state estimate that is far more accurate and reliable than what either sensor could provide alone. This enables robust closed-loop position control.

IV. Implementation Details

The methodology was realized using a specific set of hardware and software components, configured into a functional prototype for testing in a controlled indoor environment.

A. Hardware Configuration

The experimental platform was assembled from commercially available components (Fig. 2), emphasizing low cost and modularity to create a testbed for the localization system.

- **Processing Unit:** A Raspberry Pi 4B with a quad-core Cortex-A72 CPU and 4GB of RAM served as the onboard computer. This unit was responsible for handling all high-level tasks, including running the vision processing pipeline, hosting the web-based visualization server, and communicating with the flight controller.
- **Control Unit:** A Pixhawk 2.4.8 flight controller was used as the low-level control and MAVLink communication hub. Its primary role in this work was to receive and process the `VISION_POSITION_ESTIMATE` data for state estimation and forward telemetry to the ground station.
- **Perception Unit:** A dual-camera USB module, providing two 2-megapixel video streams, was used for visual perception. The cameras were mounted with a fixed baseline, establishing the hardware foundation for exploring stereo vision techniques for redundancy and improved accuracy.
- **Fiducial Markers:** AprilTags from the 36h11 family (Fig. 3) were used as the essential visual landmarks. They were printed on waterproof media in two sizes—17.5 cm for reliable long-range detection and 5.5 cm for close-range precision—to support varied operational distances.

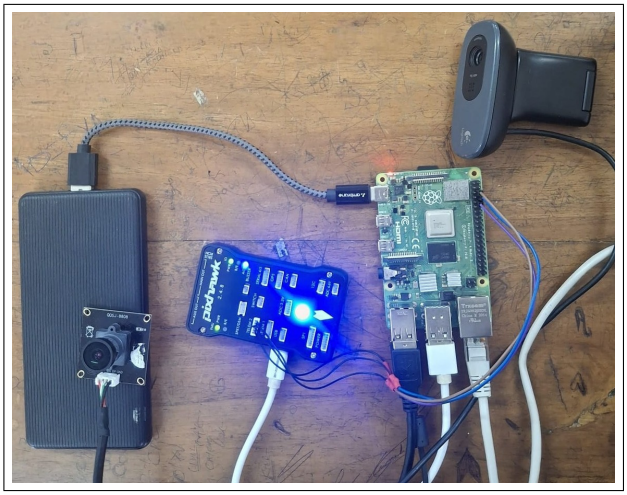

Fig. 2. The complete hardware integration for the localization system. The Raspberry Pi 4B acts as the high-level processing unit, interfacing with the Pixhawk 2.4.8 flight controller and the USB camera module, all powered by a portable supply for testing.

B. Test Environment

All experiments were conducted in a controlled indoor laboratory environment (Fig. 4). To establish a ground-truth reference frame, the AprilTags were placed at carefully measured and surveyed positions on the walls. The area was subject to standard fluorescent office lighting, providing consistent and repeatable conditions for testing the vision system's performance. This setup was critical for validating the localization accuracy by allowing a direct comparison between the system's computed pose and the known marker locations.

C. Software Stack and System Integration

The localization pipeline was developed within a specific open-source software ecosystem designed for robotics applications.

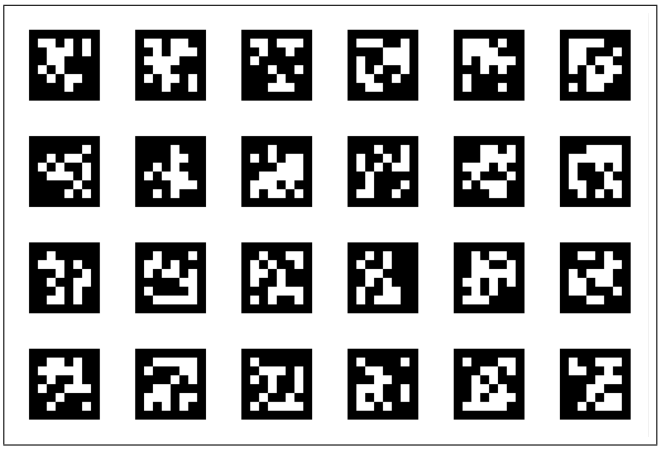

Fig. 3. The AprilTag marker from the 36h11 family implemented in our system. This family was chosen for its strong error-correction capabilities and a large dictionary of unique IDs.

Fig. 4. The controlled indoor test environment with AprilTags placed at known positions on the walls, serving as a global reference frame for localization.

- **Operating System:** The Raspberry Pi ran BlueOS v1.4.0 [9], while the Pixhawk ran the ArduSub firmware, which is tailored for underwater vehicles and includes robust support for external navigation data.
- **Core Libraries:** The main algorithm was implemented in Python 3.13 [12]. Key libraries included OpenCV 4.12-dev [7] for fundamental image processing and PnP solving, the `pupil_apriltags` library (derived from the original AprilTag library [8]) for its highly efficient detection algorithm, and NumPy 2.3 [6] for all numerical matrix and vector operations.
- **Real-time Operation:** The Python script employed a multithreaded architecture. Although a dual-camera module was present, processing was limited to a single stream to ensure a stable frame rate of 10–20 FPS due to hardware constraints on the Raspberry Pi.
- **Communication Link:** The Raspberry Pi and Pixhawk were connected via a direct UART serial link using the Raspberry Pi's GPIO pins. This provided a dedicated, low-latency channel for MAVLink communication that is independent of USB bus traffic, ensuring reliable data transmission to the flight controller.

D. Visualization and Ground Control Station

A dual-purpose visualization system was implemented for diagnostics and real-time monitoring (Fig. 5).

- A **Flask-based web server** running on the Raspberry Pi streamed the annotated video feed to a web browser. This gave operators a direct, low-level view of what the camera was seeing, with overlaid debugging text showing detected tag IDs and computed pose data.
- Simultaneously, **QGroundControl** software [11] served as the high-level ground station. By connecting to the Pixhawk, it received the forwarded `VISION_POSITION_ESTIMATE` data and displayed the vehicle's trajectory and orientation on its 3D map interface.

This combined setup provided a comprehensive view of the system's performance, from raw perception to final state estimation.

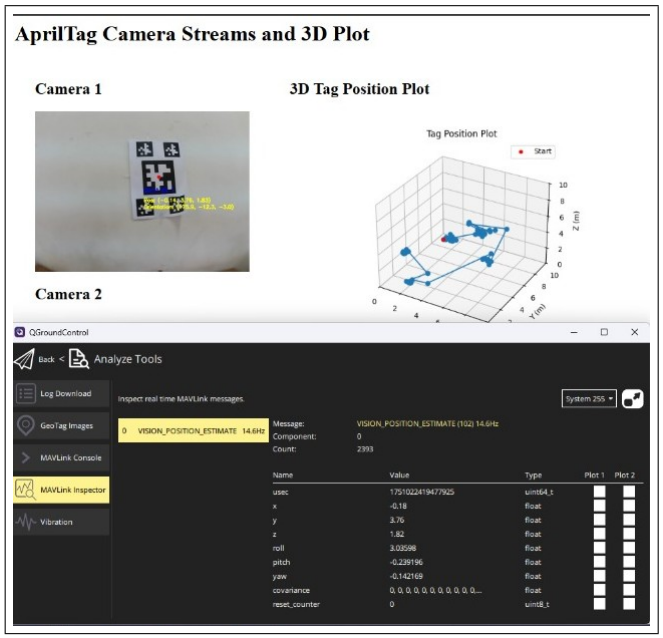

Fig. 5. The integrated ground station interface, showing the Flask web server's real-time annotated video stream alongside the QGroundControl interface displaying the corresponding vehicle telemetry and localization data.

V. Results and Discussion

The performance of the integrated system was evaluated through a series of tests conducted in the controlled indoor environment.

A. Detection Reliability and Pose Estimation Performance

The AprilTag-based localization system, operating with a single camera stream, demonstrated high reliability and robust performance.

- **Detection Range:** The system consistently detected the larger 17.5 cm tags at distances up to 1.5 meters and the smaller 5.5 cm tags up to 0.6 meters.
- **Processing Speed:** The single-stream pipeline achieved a stable processing rate between 10–20 FPS on the Raspberry Pi 4B, which is well-suited for the slower dynamics of underwater vehicles.

A primary output of the system is the real-time annotated video stream, which provides immediate visual feedback on detection performance (Fig. 6). The stability of the visual overlay—showing the bounding box, tag ID, and computed global position—confirmed that the pose estimation was consistent, with minimal jitter in the output values. This served as a strong qualitative validation that the PnP algorithm was correctly implemented and that the overall single-camera pipeline was stable.

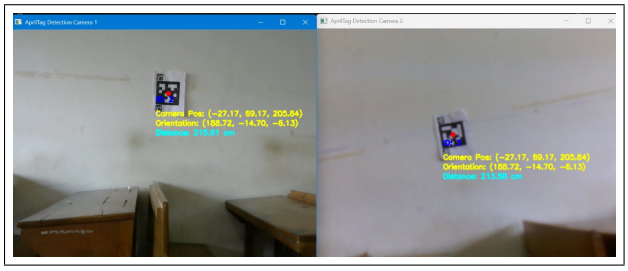

Fig. 6. An annotated frame from the USB camera showing a successful AprilTag detection. The overlay displays the detected tag ID and the computed global position.

B. Dual-Camera System Limitations

While the hardware included a dual-camera module with the goal of exploring stereo vision for redundancy and improved accuracy, we encountered a significant hardware bottleneck. Activating both camera streams simultaneously on the Raspberry Pi 4B led to inconsistent frame rates and severe USB bandwidth limitations, causing one stream to frequently drop frames or freeze. This made a reliable, simultaneous quantitative comparison between monocular and stereo performance infeasible on this platform. This finding is itself a valuable result, underscoring that while a single camera is highly effective for this localization task on the RPi 4B, leveraging true stereo vision effectively would necessitate a more powerful embedded platform with superior I/O capabilities.

C. Real-time Integration and Trajectory Tracking

The successful integration of the entire pipeline, from perception to control, was the ultimate validation of the system's architecture. The MAVLink interface consistently transmitted pose data with low latency. Figure 7 shows a 3D plot of the UUV's computed trajectory as it was manually moved through the test area. The path aligns closely with the known ground-truth marker locations, confirming the end-to-end system is capable of stable, continuous tracking.

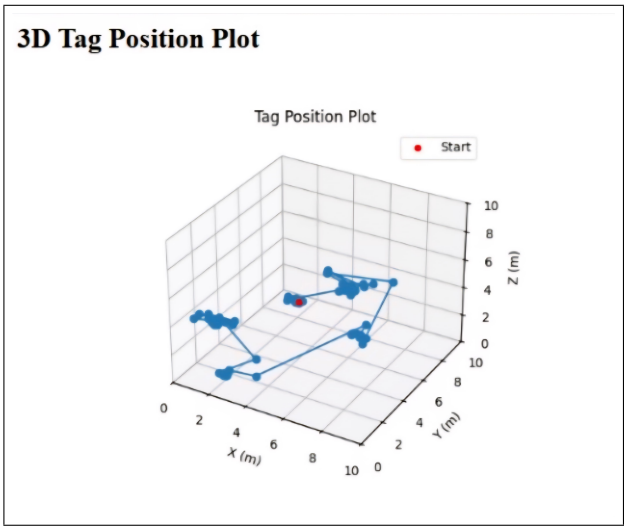

Fig. 7. A 3D trajectory plot generated from the system's output. The blue line represents the UUV's computed path, while the red marker indicates the initial position in the test environment.

D. Discussion

The collective results validate that a low-cost, single-camera AprilTag system can provide robust and accurate 6-DOF localization for UUVs in structured environments. The real-time performance on the Raspberry Pi 4B demonstrates the feasibility of deploying advanced computer vision systems on accessible, general-purpose embedded hardware. The limitations encountered with the dual-camera setup provide a clear and practical direction for future hardware selection, highlighting the necessity of higher-performance compute platforms for multi-camera or stereo vision applications. The successful MAVLink integration confirms that this system is not merely a standalone perception module but a component ready to be integrated into a larger autonomous navigation stack.

VI. Conclusion and Future Work

A. Conclusion

This paper successfully demonstrated the design, implementation, and validation of a cost-effective, embedded vision-based localization system. By integrating AprilTag markers, a single-camera pipeline on a Raspberry Pi, and a Pixhawk flight controller, we created a complete end-to-end framework capable of providing accurate and reliable 6-DOF pose estimation in real time. The results confirm that this approach is a viable and practical solution for UUV navigation in GPS-denied and structured environments, bridging the gap between theoretical computer vision and applied robotics control. The system's performance validates its suitability for prototyping, research, and operational deployment, while also highlighting key hardware constraints that inform future development and scaling of such systems.

B. Future Work

Based on the insights and limitations identified during this project, several compelling avenues for future work have been identified to enhance the system's capabilities and robustness:

- **Full Stereo Vision Integration on Upgraded Hardware:** The immediate next step is to migrate the existing software pipeline to a more powerful embedded platform, such as an NVIDIA Jetson. This will overcome the processing and I/O bottlenecks encountered on the Raspberry Pi, enabling the full implementation and benchmarking of a stereo-matching algorithm to generate dense depth maps for combined localization and obstacle avoidance.
- **Waterproofing and Underwater Field Trials:** To move from a laboratory prototype to a field-ready system, all electronics must be enclosed in robust, waterproof housings. Subsequent field trials in a controlled test tank and eventually a real-world underwater environment will be crucial for evaluating the system's performance and resilience to practical challenges like turbidity, dynamic lighting, water pressure, and biofouling.
- **BlueOS Interface Enhancement:** To create a more integrated and user-friendly operational experience, future development will focus on embedding the live camera streams and the 3D trajectory plot directly into the BlueOS web dashboard. This would provide operators with a single, unified, browser-based interface for monitoring all critical system outputs in real time.
- **Robust Sensor Fusion with an IMU:** To improve system robustness, especially during periods when no tags are visible or during fast maneuvers, the vision-based pose estimates will be fused with the Pixhawk's onboard IMU data. Implementing this within the flight controller's Extended Kalman Filter (EKF) will create a tightly-coupled visual-inertial odometry (VIO) system, significantly reducing drift and providing a more continuous and reliable state estimate.

ACKNOWLEDGMENT

The authors thank the Department of Electronics and Telecommunication Engineering, Goa College of Engineering, for their support and infrastructure.

REFERENCES

[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, p. 1432, 2021.

[2] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011, pp. 3400–3407.

[3] J. Ulrich, A. Alsayed, F. Arvin, and T. Krajník, "Towards fast fiducial marker with full 6 DOF pose estimation," in *Proc. 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22)*, Virtual Event, 2022, pp. 723–730.

[4] T. Jones and S. Hauert, "Frappe: Fast Fiducial Detection on Low-Cost Hardware," *Sensors*, vol. 23, no. 2, p. 944, 2023.

[5] 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, p. 1111, 2023.

[6] C. R. Harris et al., "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020.

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

[8] AprilRobotics, "AprilTag C Library and Driver," GitHub Repository. [Online]. Available: https://github.com/AprilRobotics/apriltag

[9] Blue Robotics, "BlueOS - An Operating System for Marine Robotics," 2024. [Online]. Available: https://blueos.cloud/docs/stable/usage/overview/

[10] ArduPilot Dev Team, "MAVLink Basics," ArduPilot Developer Documentation, 2024. [Online]. Available: https://ardupilot.org/dev/docs/mavlink-basics.html

[11] QGroundControl Developers, "QGroundControl: Ground Station Software," 2024. [Online]. Available: https://docs.qgroundcontrol.com/

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