HORNETXI FINAL REPORT

Tuan Pham
Software Individual Contributor
HornetXI
phamanhtuanben@gmail.com

***Abstract*—Team Bumblebee's Hornet Programme is a year-long engineering training initiative designed to cultivate the next generation of capable and committed AUV developers. Participants are tasked with designing and building a fully functional Autonomous Underwater Vehicle from scratch, working across mechanical, electrical, and software disciplines in a structured, competition-oriented environment. This report documents the experience and contributions of a member of the Software subteam, with a specific focus on the Perception and Localisation group's computer vision pipeline. The report outlines the design approach taken, details personal contributions to the system, and reflects on limitations encountered and directions for future improvement.**

I. Introduction and Documentation

A. Programme Overview

The Hornet Programme serves as Team Bumblebee's primary pipeline for developing engineers capable of contributing to a competitive AUV platform. The programme is structured around the demands of the SAUVC competition organised by IEEE OES Singapore Chapter, where AUVs are required to autonomously complete a series of complex underwater tasks without any human intervention during a run. This competition context sets a high technical bar and provides a concrete, external benchmark against which the team's engineering decisions are evaluated.

The programme is therefore designed not merely as an onboarding exercise, but as an engineering experience that mirrors the full development lifecycle of a competition AUV. Participants are expected to engage with real design constraints such as hardware limitations, system integration challenges, and time-bound development cycles, and to take genuine ownership of their respective subsystems. Beyond technical development, the programme is structured to assess attributes that are difficult to evaluate through conventional means: the ability to work collaboratively under pressure, to iterate effectively in the face of setbacks, and to sustain commitment across a year-long project. In this sense, the Hornet Programme functions simultaneously as a training ground and a selection process, ensuring that those who progress to the main team are both technically prepared and culturally aligned with the demands of competitive AUV development.

B. Scope of Problems and Responsibilities

Within the broader AUV development effort, the Software subteam is responsible for the full stack of autonomous behaviour — from raw sensor data processing to vehicle actuation. The subteam is organised into four focus groups, each addressing a distinct layer of the autonomy pipeline.

Perception and Localisation: This group is responsible for equipping the AUV with environmental awareness. This encompasses processing visual and sensor data to detect, classify, and localise objects of interest, as well as determining the vehicle's own position and orientation within its operating environment. The reliability of all downstream systems is directly dependent on the accuracy and robustness of this group's output.

Controls: This group is responsible for translating high-level navigation commands into precise actuator outputs, ensuring the AUV maintains stable motion and accurately executes manoeuvres. This involves the development and tuning of control algorithms that must account for the complex and often unpredictable hydrodynamic forces acting on the vehicle underwater.

Simulation: This group develops and maintains a virtual environment that replicates the AUV's operating conditions, enabling software to be tested and validated without requiring physical deployment. This is critical for accelerating the development cycle and for safely identifying failure modes before live trials.

Communications: This group manages the data interfaces between the AUV's onboard systems and any external monitoring or control infrastructure. This includes ensuring reliable transmission of telemetry data and enabling operator oversight during testing and competition runs. Together, these focus groups form an interdependent software ecosystem, where the output of each group feeds into the next, ultimately enabling fully autonomous operation of the AUV.

II. Implementation

A. Solution and Design Approach

a) *Software Subteam — Overall Design Approach:* The software subteam adopted a modular, distributed architecture built on the Robot Operating System (ROS) as its foundational framework. ROS was selected for its well-established ecosystem of libraries and tools suited to robotic applications, as well as its native support for inter-process communication through a publisher-subscriber messaging model. This architecture allowed each focus group: Perception and Localisation, Controls, Simulation, and Communications to develop and test

their respective subsystems with a degree of independence, while still operating within a shared and well-defined integration layer.

Each subsystem is encapsulated as one or more ROS nodes that communicate via standardised message types over defined topics. This design philosophy enforces clear interface boundaries between subsystems, reducing coupling and making it possible to develop, test, and debug individual components in isolation before integrating them into the full system. The Simulation group further supported this workflow by providing a virtual environment in which software components could be validated without requiring access to the physical vehicle, significantly accelerating the development cycle.

The overall design approach was therefore one of parallel, modular development converging toward a unified autonomous system — with ROS serving as the common backbone that binds all subsystems together.

b) *Perception and Localisation — Role and Design Focus:* Within the software subteam, my work was situated in the Perception and Localisation group, specifically within the computer vision domain. The primary responsibility of this role was to develop the AUV's visual perception pipeline, the system by which the vehicle interprets its camera feed to identify and understand objects in its environment. This forms the first and most critical stage of the autonomy pipeline, as all subsequent decision-making by the controls and task planning systems depends on the quality and reliability of the perceptual output.

A significant portion of the work was carried out independently, particularly in the development and validation of individual vision components. The role also required close integration with other focus groups. The outputs of the computer vision piHere is the Features in the Final Design section:

B. Features in the Final Design

a) *Object Detection and Classification:*
To address the challenge of reliable object detection in a visually degraded underwater environment, the team developed a custom classification model trained specifically on underwater targets relevant to the RoboSub competition, such as poles and flares. Rather than relying on general-purpose pretrained models that are optimised for well-lit, above-water imagery, a custom model was necessary to account for the colour distortion, low contrast, and uneven lighting characteristic of pool environments. The resulting model was integrated into the perception pipeline to provide real-time detection and classification of task-relevant objects within the AUV's camera feed.

b) *3D Depth Estimation via Stereo Vision:*
To determine not just what objects are present but where they are spatially, the team employed a stereo camera setup. By capturing two simultaneous video feeds from cameras separated by a known baseline distance, the system is able to compute disparity maps — comparing corresponding pixels across both feeds to infer depth through triangulation [1]. This yields a 3D pose estimate for detected objects relative to the AUV, providing the downstream control and planning systems with the spatial information necessary to execute precise interactions with targets.

c) *State Estimation via Optical Flow Odometry:*
Accurate self-localisation underwater is a well-known challenge, as GPS is unavailable and acoustic positioning systems are often impractical in competition settings. To estimate the AUV's velocity and track its position over time, the team implemented an optical flow algorithm on a downward-facing camera. Optical flow methods estimate motion by analysing the apparent displacement of pixel patterns across successive frames [2], and have been demonstrated as an effective source of velocity feedback in GPS-denied environments. The resulting velocity estimates are passed through filtering algorithms to perform odometry, building up a continuous estimate of the vehicle's position from integrated motion data.

d) *Motion Control via Thruster Allocation Matrix:*
For stable and precise motion control, the team implemented a thruster allocation framework grounded in the theoretical foundations presented in Fossen's Handbook of Marine Craft Hydrodynamics and Motion Control [3]. Central to this approach is the Thruster Allocation Matrix, which mathematically maps desired generalised forces and torques in each degree of freedom to the corresponding thrust outputs required from each individual thruster. This formulation accounts for the geometric configuration of the thrusters on the vehicle frame, enabling the control system to produce accurate and coordinated motion commands even in the presence of hydrodynamic disturbances.

e) *Mission Planning via Behaviour Trees:*
High-level mission logic was implemented using PyTrees [4], a Python library for constructing and executing behaviour trees. This approach mirrors the architecture of the Nav2 navigation stack widely used in the ROS ecosystem [5], providing a structured and modular framework for encoding complex, conditional mission sequences. Behaviour trees offer clear advantages over monolithic state machines in this context. They are inherently hierarchical, easily extensible, and allow for reactive execution where the vehicle can respond to changing environmental conditions [6].

Fig. 1. Simulation Environment.

f) *Path Planning via RRT:*

For navigating between waypoints and manoeuvring within the competition space, the team employed a Rapidly-exploring Random Tree (RRT) path planner, originally proposed by LaValle [7]. RRT is a sampling-based planning algorithm well-suited to high-dimensional configuration spaces and is capable of generating collision-free paths in environments with obstacles. In the context of the AUV, the RRT planner provides the vehicle with dynamically computed trajectories that guide it toward target positions while respecting the physical constraints of the vehicle and the boundaries of the operating environment.

g) *Software Validation via Underwater Simulation:*

To enable safe and efficient testing without dependence on physical pool access, the team integrated the DAVE simulation environment — an underwater robotics simulator built on Gazebo [8] — into the development workflow. DAVE provides realistic models of underwater dynamics, sensor behaviour, and environmental conditions [9], allowing the team to deploy and validate behaviour trees and other software components in a simulated competition environment. This significantly reduced the risk of untested code being run on the physical vehicle and accelerated iteration across all software focus groups.

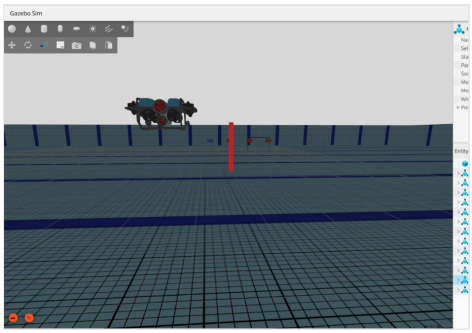

Fig. 2. Simulation Environment.

C. Personal Contribution

My primary contribution within the Perception and Localisation group was in the development and evaluation of the 3D depth estimation pipeline. The core objective was to establish a reliable method for estimating the spatial position of detected objects relative to the AUV using the onboard stereo camera setup. This involved researching, implementing, and benchmarking multiple candidate solutions before arriving at a final approach suitable for the team's hardware and software constraints.

a) *Evaluation of Candidate Approaches:*

stereo_image_proc: The first candidate evaluated was stereo_image_proc, a standard ROS package that performs stereo image rectification and disparity map computation using classical block matching algorithms [10]. The package takes rectified left and right image pairs and produces both a disparity image and a registered point cloud, making it straightforward to integrate within the existing ROS architecture. Its primary advantage lies in its maturity and native ROS compatibility, requiring minimal additional infrastructure to deploy.

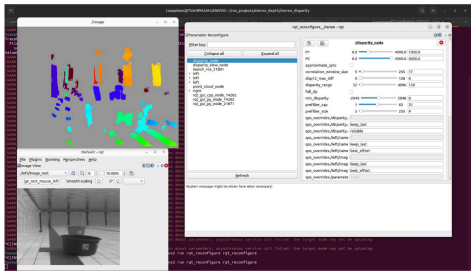

Fig. 3. Testing of stereo_image_proc underwater.

ISAAC ROS Stereo Depth: The second candidate evaluated was NVIDIA's ISAAC ROS Stereo Depth package [11], which leverages GPU-accelerated deep learning models to produce dense depth estimates from stereo image pairs. ISAAC ROS offers significantly higher throughput and improved accuracy in textureless or low-contrast regions compared to classical methods, which is particularly relevant in underwater environments. However, its deployment is tightly coupled to NVIDIA hardware and requires a compatible Jetson or discrete GPU platform, introducing hardware dependency constraints that had to be weighed against its performance benefits.

StereoAdapter: The third candidate evaluated was StereoAdapter [12], a learning-based approach that adapts monocular depth estimation models for stereo input. Rather than relying purely on classical disparity computation or purpose-built stereo networks, StereoAdapter leverages the generalisation capability of pretrained monocular models and fine-tunes them for stereo consistency. This approach was evaluated as a potentially more flexible alternative, particularly in scenarios where stereo-specific training data is limited.

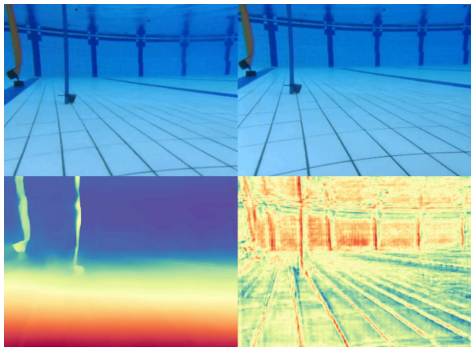

Fig. 4. Testing of StereoAdapter underwater.

b) *Final Design Decision:*

After evaluating all three approaches against criteria including accuracy, computational overhead, hardware compatibility, and ease of integration within the ROS ecosystem, stereo_image_proc was selected as the final solution. Despite being a classical rather than learning-based approach, it offered the most consistent and predictable performance within the team's hardware constraints, and its native ROS integra-

tion significantly reduced the complexity of interfacing with the broader software stack. The package was configured and tuned with appropriate camera calibration parameters to maximise disparity quality, with its output feeding directly into the object localisation pipeline for downstream consumption by the controls and mission planning systems.

D. Limitations

The logistical overhead of preparing an Autonomous Underwater Vehicle (AUV) for a single water test often consumes more time than the actual mission execution. In a competition setting, the bottleneck isn't just the software bugs, but the "pre-dive" friction: the painstaking process of waterproofing hulls, securing tether connections, balancing trim weights, and calibrating sensors in a pressurised environment. When these manual operations are unoptimized, the team loses critical iteration cycles; a thirty-minute code adjustment can easily be buried under three hours of mechanical assembly and teardown, stifling the ability to perform the rapid, real-world testing necessary for a reliable vehicle.

III. Future Work

To resolve this, the focus should shift toward modular hardware and high-fidelity simulation. If time permitted, implementing a "quick-swap" battery and electronics tray system would allow for immediate hardware resets without breaking the primary waterproof seals. Simultaneously, investing in a digital twin environment using Gazebo or Isaac Sim would enable the team to validate control algorithms and sensor fusion logic in a virtual pool before ever touching the water. By automating the deployment checklist through a unified Ground Control Station (GCS) and reducing the mechanical "mean time to repair," the team could transform a day of testing from a single, high-stakes dive into a series of agile, productive sprints.

References

[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2004.

[2] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185–203, 1981.

[3] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, 2nd ed. Chichester, UK: Wiley, 2021.

[4] S. Suomalainen, "PyTrees." [Online]. Available: https://github.com/splintered-reality/py_trees

[5] S. Macenski, F. Martín, R. White, and J. Clavero, "The Marathon 2: A Navigation System," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2718–2725.

[6] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. Boca Raton, FL: CRC Press, 2018.

[7] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," technical report TR98–11, 1998.

[8] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, pp. 2149–2154.

[9] B. Bingham and others, "Toward Maritime Robotic Simulation in Gazebo," in *Proceedings of MTS/IEEE OCEANS*, 2021.

[10] ROS Wiki Contributors, "stereo_image_proc – ROS Wiki." [Online]. Available: http://wiki.ros.org/stereo_image_proc

[11] NVIDIA Corporation, "ISAAC ROS Stereo Depth." [Online]. Available: https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_depth_segmentation

[12] H. Byun and others, "StereoAdapter: Adapting Monocular Depth Models for Stereo Depth Estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.