A Comprehensive Design Report of Our AI-Based LPV

Technical Report Purdue University: AIMM-CC Challenge

AUTHORS:

Purdue University West Lafayette Team.

APRIL 2024

Table of contents

Abstract

This document presents the design and development of an innovative autonomous boat focused on implementing advanced computer vision systems and autonomous navigation strategies. Through a synergy of emerging technologies and ingenious design, our team aims to excel in the competition, addressing unique challenges in marine engineering and robotics. In this work, the team prioritized autonomous navigation and integration of computer vision for path planning. The team uses Ardupilot to set an initial path using GPS odometry and a GoPro wide-angle camera for initial buoy locating. After reaching a distance of less than 7 m, the OAK camera and computer models have high enough accuracy to be used to detect depth and location. Using a YOLO model, the team detects the buoy, and its color and navigation adjusts accordingly. A CUAV V5+ Autopilot is used to control the motor's direction and speed using PWM, steering is done through a linear actuator and Arduino control, and coordination of the dual CANBus GPS modules is crucial for the boat's mobility and maneuverability. In order to achieve the goals of other challenges in this project, a LORA sensor was added, as well as a gripper for the rings and baby-grabbing challenges.

Technical Content

Competition Goals

This section details our specific objectives for the competition, emphasizing the interaction between the boat's design and our approach to the various challenges. We discuss critical considerations between system complexity and reliability, which are fundamental to our strategic approach. The team decided to prioritize autonomous navigation and the Slalom course as the first initial challenge in this work. Therefore, initial development was done in computer vision and controls, which are substantially linked. The team knew the GPS locations of the buoys in the course and therefore selected Ardupilot as a software for navigation and integration using MAVRos. The CUAV V5+ Ardupilot was selected for control. The team utilized two vision systems, one for long-range and one for close-up buoy detection. Switching between these vision systems in Adrupilot is done using a confidence level. The Adrupilot also has an onboard GPS, which provides the system with initial orientation values. The CUAV V5+ also provides a PWM signal to the motor to control speed and direction and to a linear actuator that controls steering. After autonomous movement was achieved, the team built upon this platform to add LORA sensing capabilities and a gripper for grabbing rings and the baby challenge.

Design Strategy

Computer Vision

Data collection and processing

In our role of data preparation as a part of the computer vision team, we were responsible for creating the dataset behind the navigation model. To ensure the effectiveness of our dataset, we employ several methods utilizing

tools such as Python, Roboflow, and Google Collab. To streamline the annotation process for our dataset, the data team leverages Roboflow, ensuring a dynamic and diverse dataset. Additionally, we continuously implement a Python script utilizing YOLOv8 to enhance efficiency. To complete the training pipeline, Google Collab has been used to operate more substantial computing power and speed up the training process.

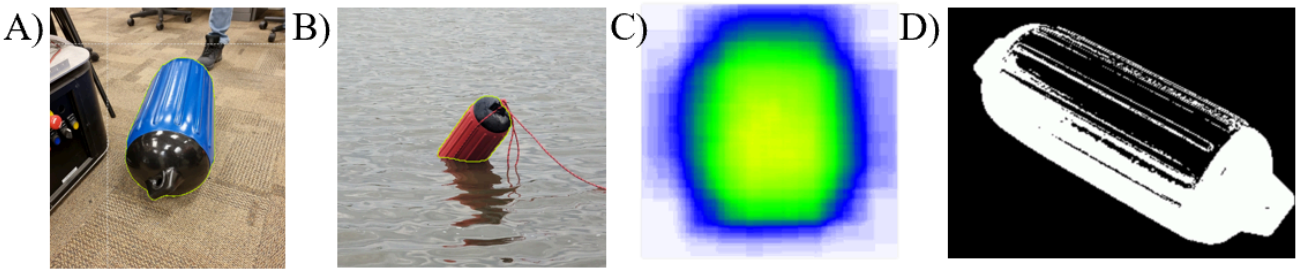

Figure 1. A and B) Buoy images C) Heat map of detection D) Greyscale image

Firstly, we select a representative sample of Google images capturing buoys in various environmental conditions, such as different weather conditions, times of day, and water conditions. Secondly, we ensure a balanced distribution of buoy colors within our dataset, avoiding bias towards any particular color and enabling the model to classify buoys of all hues accurately. Additionally, we include images with different backgrounds, ranging from open water to busy harbors, to train the model to recognize buoys in other contexts. Furthermore, we incorporate images with varied perspectives and orientations, allowing the model to learn to identify buoys from different viewpoints. We also implement data augmentation techniques, such as rotation, translation, and color manipulation, to augment our dataset and expose the model to a broader range of variations. Another augmentation approach we utilized was converting the photos to grayscale. This process helped us focus on detecting the actual buoy and less on the color. We used Roboflow's augmentation feature to convert the dataset to a grayscale image. Our team also utilized the Python libraries OpenCV to augment the color of existing buoy photos. We converted the image to a detectable color scheme, such as HSV color space, and created a mask to isolate the color. Then, we used this mask to convert the mask to the desired buoy color. Lastly, we continuously update and expand our dataset to include new examples and adapt to changing environmental conditions, ensuring our model remains effective and reliable. By utilizing various augmentation methods, cloud computing, and a large, diverse dataset, we hope our efforts will be enough to help our team navigate the course successfully. This process is shown in Figures 1 and 2.

Multiple-Object Detection & Tracking

The YOLOv8 pipeline for buoy detection integrates an initial base detection model to identify bounding boxes around potential buoys in an input frame. Color-specific classification models then ascertain the buoy's color with the highest confidence level, incorporating confidence thresholds for accuracy. Depth information is integrated by calculating the mean depth within the detected bounding box area, utilizing the camera's depth-sensing capabilities. This combined data provides comprehensive output detailing each detected buoy's location and characteristics. The `BuoyDetector` class encapsulates the pipeline's functionality, initialized with paths to the base and color-specific YOLO models, alongside configurable confidence and IoU thresholds for detection precision tuning. The `predict_with_depth` method is the core operation, applying the base model for initial detection, followed by cropping and color classification using color models for each detected region. Depth information is extracted for each bounding box through the `Camera` class's depth sensing, providing mean depth values alongside color and position data in the output. This approach ensures detailed output,

including each buoy's color, central coordinates, and depth information, meeting the specified buoy identification and classification requirements.

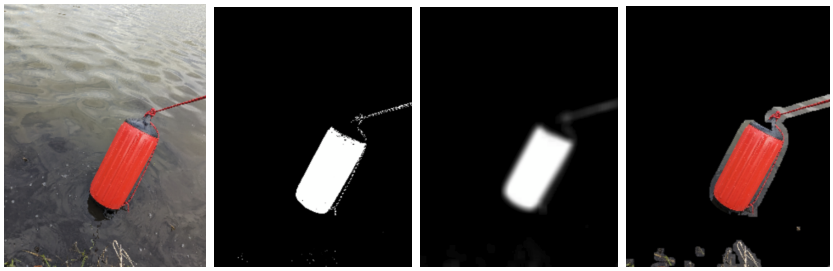

Figure 2. Detection progression

Hardware & CV Pipeline

This computer vision subteam aims to take the software and object detection models we have created and incorporate them into the real-time decisions that the controls team needs to make. To do this, we first had to write a script that reads the data coming from the Oak-D POE camera. There was already a Python package to help with this, so we only had to encapsulate it into a camera class where the YOLO script can instantiate it and continuously get the RGB frames and average depth for each bounding box. Using the camera data, the YOLO script runs all the models it needs and sends the data to the scripts for control and navigation. All this is done on the NVIDIA Jetson Orin Nano. Additional latency testing should be conducted. If latency is too high, there are a few ways to reduce this: use smaller and less intensive models, run models on the camera (a feature of the Oak camera), or use the Oak SDK instead of API. The other hardware component we are working on is incorporating a wide FOV GoPro that is mounted higher on the boat. This will run just buoy detection without color for farther-away detection ranges. The depth detection on the OAK camera is only valid at less than 10 m with an optimal range between 4 - 7m. Therefore, a longer-range direction is needed. The long-range detection is less accurate but allows the control team an initial direction to head until the higher accuracy OAK can pick up the image once buoys reach within 7m (Oak-D preferred range). Then, more accurate data can be computed on for controls.

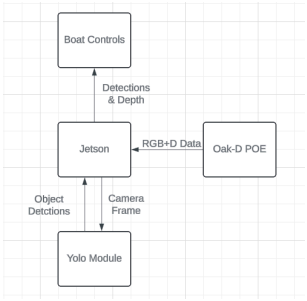

Figure 3.Computer Vision Hardware Pipeline

Control and Navigation

This subsection focuses on the Jetson Board as our primary controller, detailing how it efficiently manages the navigation system. We also explore the role of the CUAV V5+ Autopilot in controlling the motor, linear actuator, and coordination of the dual CANBus GPS modules crucial for the boat's mobility and maneuverability. We use a combination of the CUAV V5+ Autopilot Flight Controller Unit (FCU) flashed with the ArduPilot firmware in the boat configuration and the Jetson Orin Nano as our companion computer running

Mavros2 in a ROS2 Humble Docker Container. We use the Mission Planner Ground Control Station (GCS) running on the Jetson for configuration and waypoint planning. ROS2 service calls through the MAVROS package are used to control the mission and set/alter mission parameters.

System Overview

The autonomous boat is equipped with a CUAV v5+ autopilot interfaced with ArduPilot firmware, designated for marine vessel operation. This setup is complemented by a Jetson Orin Nano companion computer that runs ROS2 Humble in a Docker container. The autopilot's primary function is to manage low-level motor and steering commands, execute the mission plan, and process GPS odometry. Figure 4 A - D shows MAVROS, the Adrupilot software, and execution in a simulated environment.

Sensor Integration

The CUAV v5+ autopilot is integrated with a suite of sensors (Figure 1), including an accelerometer (ICM-20602/ICM-20689/BMI055), gyro (ICM-20602/ICM-20689/BMI055), compass (IST8310), and barometer (MS5611). These sensors are vital for the boat's stability and navigation capabilities. Two external CANBus GPS modules are incorporated into the system for redundancy and enhanced accuracy. The configuration can be seen in Figure 4E.

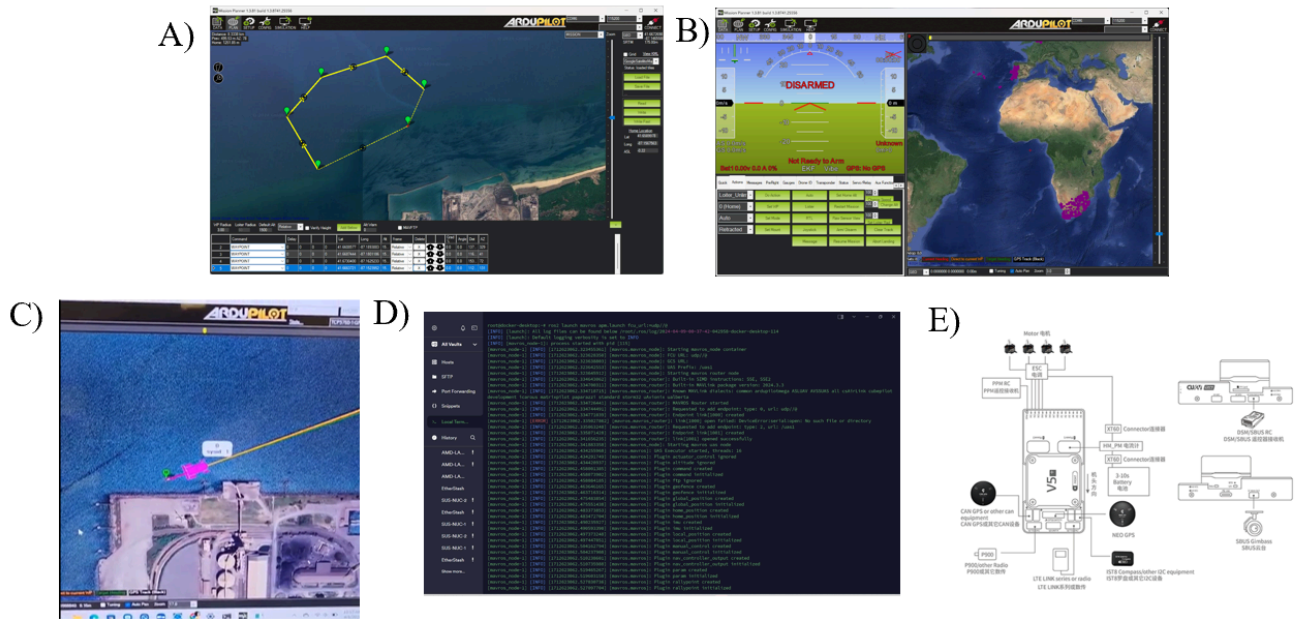

Figure 4. A) Mission plan B) Ardupilot GCS View C) Waypoint Finding D) Docker Container running MAVROS2 E) Configuration for Autonomous Boat Controls

Communication and MissionProtocol

The autopilot communicates with the Jetson Orin Nano via a USB Serial connection. This communication allows for the exchange of high-level mission details and sensor data. GPS waypoint estimation and initial mission planning are conducted through the Mission Planner ground control station. Upon mission initiation, waypoints and task configurations are dynamically updated via ROS2 service calls and parameter settings. The position estimates provided by the Vision team are used to refine the control algorithm, adjusting the mission waypoints in real time. For the slalom task, the initial GPS coordinates of the buoys are used to

calculate a polynomial trajectory. These coordinates are subsequently refined based on the Vision team's updates, and the mission waypoints are adjusted through ROS service calls. A similar strategy is planned for gate navigation tasks. Position holding for specific tasks is achieved using ArduPilot's loiter mode, which utilizes the onboard sensors to maintain a stationary position effectively.

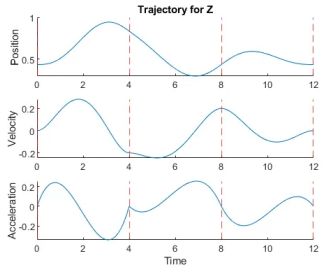

Figure 11. Polynomial Trajectory Planning [1]

Manipulator and Task Interaction

The onboard manipulator is ROS-controlled, allowing for precise movements and interactions with the environment. Image servoing techniques, along with a camera attached to the manipulator, are employed for most tasks, facilitating accurate engagement with task objectives. All sensor data is transmitted over separate ROS topics, ensuring that the system has a consistent and organized stream of information for decision-making processes. Sensor fusion algorithms combine data from the LIDAR, RGBD camera, GPS, and additional onboard sensors to create a comprehensive environmental model crucial for the autonomous navigation of the boat.

Mechanical Design

We decided 2x4 planks would be the best option for mounting most components on the boat due to how easy they were to cut to size and mount. As shown below, a 2x4 frame was made to mount the trolling motor to the boat. This mount was the basis for all other mounting efforts on this boat, which is 2x4s mounted with drywall screws and 3M 4200 marine-grade adhesive. This adhesive was selected as it is the less permanent version of the best-rated marine adhesive, 3M 5200. This adhesive also acts as a sealant, which we use to avoid any potential water ingress points from mounting components to the boat.

Figure 6. Motor Mount

Custom Arduino-based systems were developed for motor control in both direction and speed. These systems are both made using an Arduino Mega and a 10A H-Bridge to supply the regulated power for the motor and linear actuator. The trolling motor stripped the speed switch from it and replaced it with the H-Bridge to allow for PWM (Pulse Width Modulation) based control from the Arduino Mega. The turning component was done using a linear actuator, specifically the PA-14 Mini Linear Actuator, controlled through a separate Arduino Mega. The internal potentiometer was used to track the extension of the linear actuator. This was mounted to a

vertical 2x4 and to the trolling motor with through bolts loose enough inside the linear actuator holes to allow for the bolts to spin freely, allowing for the extension of the linear actuator to cause the motor to rotate along its central shaft.

Linear Actuator

Our linear actuator is controlled by an Arduino Mega 2560 and a 12V power supply. It is a low-current, medium-force actuator usually used for automation and manufacturing purposes. It also includes a fairly simple installation process. On the programming side, we will use two pins to connect to the L298N (motor driver) for extending and retracting and one pin for our position control.

At the initialization of our code, we will turn off the actuator. Then we create a variable in which we will set a desired position (based on an 8-bit value, therefore 512). Once we have this desired value, we will read the analog value of our position pin to determine our current location. If our current location is close to our commanded position, we will leave the linear actuator as it is. If it is too far, we will retract it. Or if it is too close, we will extend it. Finally, we added a small delay to reduce jitter and count for hysteresis.

Motor Control

To control the velocity of the motor, we create a Pulse Width Modulation program through our Arduino microcontroller using an H-Bridge module (MOSFET IRF3205). It is a 10A dual-channel motor drive board that can create a frequency from 400 to 20kHz. We decided to choose this H-bridge because it would provide more current for the motor therefore, we substituted our speed switch with variable velocity. By doing so, we created a variable-speed motor that was controlled similarly to a potentiometer.

In our test program, first, we set our PWM and direction pins as outputs. Then, we set the direction of our motor to move forward and start on a 50% duty cycle. This means it is at only half its top speed. The motor runs for five seconds until it engages a break. To do so, the PWM is set to 0% and turned off for 2 seconds. Once it breaks, it is set in reverse, which means changing the direction and running at 50% duty cycle for five seconds. Finally, we engaged the brake for two seconds to create a 0% duty cycle.

With this program, we can set our direction pin in different ways, allowing us to control both direction and speed. The speed can change to a variable duty cycle, allowing us to make it faster or slower regardless of direction. This can help when moving through buoys or catching a wave.

To power the whole boat, we are using the 12V battery supplied in the kit with the provided wire and bus bars to create a 12V rail any 12V device can tap into. This rail will power the NVidia Jetson, PoE switch, motors, and the step-down circuit to power the lower-power devices.

Communications and Sensors

In this subsection, we address the implementation of Arduino for sensor data collection and transmission, following detailed technical specifications. We also discuss the integration of GPS and LiDAR, which are essential for accuracy in navigation and distance determination.

Lora communication:

We have adopted a solution focused on LoRA (long-range) technology to establish an effective communication and sensing system, particularly for challenges 5 and 8. This strategic choice is based on the unique characteristics of LoRa, which provide us with significant advantages in the maritime context.

LoRa technology stands out for its ability to transmit data over long distances, which is essential in the marine environment where we face long distances and various physical obstacles. Unlike other wireless technologies

such as Wi-Fi or Bluetooth, LoRa offers a much more comprehensive range, crucial for effective communication between our low-power vehicle (LPV) and the deployed sensors.
Another fundamental aspect of LoRA is its low energy consumption. In a project where battery life and device sustainability are critical, the energy efficiency of LoRa ensures that our sensors can operate for extended periods without the need for frequent recharges. Furthermore, the robustness of the LoRa signal against interference and its ability to maintain communication integrity in a noisy and changing marine environment ensure the reliability of the transmitted data.
In terms of implementation, we have integrated LoRa modules into the Arduino MKR WAN 1300 in our LPV and the sensors. This configuration allows for collecting relevant environmental data, such as atmospheric pressure, temperature, humidity, and ambient light, and its efficient transmission to the LPV and, finally, to the base station.

Testing Strategy

This section outlines our testing methodology in the lab and real-world environments. We detail how each component and the system are evaluated to ensure optimal functioning and reliability under competitive conditions.

Laboratory Testing of Individual Components:

At the outset, each component of our LPV underwent individual laboratory testing. This critical step involved evaluating the motor's speed and responsiveness to control signals and confirming the adequacy of torque and current supplied by the controllers to power the motor and linear actuator effectively.
These controlled tests were essential for validating the performance of components like Arduino PWM controllers, ensuring they met our stringent requirements for precision and efficiency.

Initial Water Testing of Individual Components:

As we progressed from the lab to a more challenging environment, we conducted initial water tests in a swimming pool. This phase allowed us to observe the performance of components in an aquatic setting.
During these tests, we focused on confirming the controllers' ability to handle the motor and linear actuator in water. This was not just about mechanical performance; it also included verifying the effectiveness of our written codes and the preliminary integration of all boat components with the central controller.
Identifying the optimal placement of cameras and validating the functionality of our computer vision systems were integral parts of this phase.

Integrated System Testing with Buoys:

Finally, we conducted comprehensive tests of the fully integrated system in an environment with buoys. This testing phase was crucial for evaluating our LPV's ability to navigate and interact with real-world obstacles and targets as it would in competition scenarios.
These tests included trials for tasks like slalom and gate navigation, where coordinating sensors, control systems, and computer vision algorithms were paramount. Our team leveraged the power of the NVIDIA Jetson Orin Nano, the CUAV V5+ Autopilot, and sensor fusion techniques to ensure smooth and precise navigation and task execution.

Acknowledgements

We sincerely thank the AIMM-ICC Challenge organizers at Trine University for their support in our project's journey. Thanks to our mentors, Dr. Richard Voyles, Dr. Samuel Labi, and Dr. Brittany Newell, whose expertise and encouragement have been indispensable. We extend our appreciation to Timothy Murphy from the Naval Surface Warfare Center, Crane Division (NSWC Crane), whose guidance and mentorship were invaluable in advancing our technical and strategic capabilities. Purdue University's supportive environment and resources have been crucial to our development. Finally, we commend our team members for their dedication and teamwork, which were pivotal in turning our ambitious project into a reality.

References

[1] https://medium.com/mathworks/trajectory-planning-for-robot-manipulators-522404efb6f0
[2] V5+ autopilot · cuav-v5