Minion: Design and Competition Strategy for the 2024 Maritime RobotX Challenge Minion

Adam Lachguar, Giovanna Ucles, Sagar Sarkar, Dan Lane, Erik Liebergall, Sarthak Aggarwal, Madeline Thomson, Willis Proper, Michael Saravis, Dean Dcruz, Isaac Kay, Matis Abe, Bharat Jagwani, Rohit Vinnakota, Mateus Prudencio, Marshall Yelvington, Jacob Young, Neel Pathi, Brian Park, Aarambh Anand, Missionary Thumati, Andrew Lam, Thomas Jones, Dr. Eric Coyle, Dr. Patrick Currier

***Abstract*—Embry-Riddle Aeronautical University's Team Minion is returning to RobotX with significant improvements to its defending champion fully autonomous surface vessel (ASV), Minion. Team Minion's new design strategy and systems engineering approach, called Minion Process, has allowed a balance of academics, research, and team objectives throughout the team. This design strategy combined with a rigorous multi-step testing process that values safety and innovation has led to an ever-improving toolset for Minion and its Uncrewed Aerial Vehicle (UAV), Kevin. These include software enhancements of a new patent-pending control scheme and better integration of computer vision throughout the system, as well as hardware improvements of azimuthing motor control, new UAV capabilities, and a new ball launcher.**

The team's pre-competition assessment of these tools has led to a robust competition strategy whose basis is on maximizing points while minimizing risks. The team's task tracker, MinionTask, will dynamically select tasks based on the assessed strategic value, known course information, and time remaining to optimize competition performance in qualifying, semi-finals, and finals. Based off results from a combination of simulated and on-water testing, Team Minion is confident in its ability to complete at least six of the nine RobotX 2024 tasks and hopes to repeat its championship performance.

I. Overview

TEAM Minion is an interdisciplinary team of undergraduate and graduate students from Embry-Riddle Aeronautical University (ERAU). In addition to competing in the Maritime RobotX Challenge, the platform is used to provide graduate research and undergraduate experiential learning opportunities. As such, the strategy of Team Minion is to create tools and resources that balance the academic, research, and competition needs of its members. This has lead Team Minion to develop a systems-based design, where mechanical systems and software modules are primarily developed for research or academic purposes, but can be applied to the team's strategy for completing the 2024 competition. This paper will highlight how Team Minion's design strategy determines the software architecture and systems the team develops or improves, how its testing strategy ensures system compliance, and how Minion's systems and software are applied to the task objectives of the 2024 RobotX competition.

A. Minion 2022

Team Minion, now a decade old, has always sought to create an Autonomous Maritime System (AMS), that is rugged, customizable, and easily upgraded to meet the emerging academic, research, and competition needs of its members. The 2022 RobotX boat, which placed 1st overall, utilized a proven differential steering system, introduced a new perception suite, and did not utilize secondary systems of a ball launcher or hydrophone array. Its software code repository was ported to the industry standard Robot Operating System (ROS) for inter-module communication, but its algorithms for vehicle perception and autonomy remained customized solutions.

Since all of Minion's systems are viewed as tools that can be reconfigured to complete research or competition objectives, preparation for the 2024 competition began by determining a Design Strategy that would determine a set of new tools to develop, tools that need improvement, and a process for accomplishing these upgrades. Due to these priorities, the team started with a Design Strategy and developed a Competition Strategy, supported by the Testing Strategy, that fit within the broader design philosophy.

B. Design Strategy

1) Philosophy: Minion is developed on a system of systems design philosophy, where modular hardware and software tools are developed that can be re-configured to achieve complex missions at competition and for research purposes.

When developing a new tool, Team Minion utilizes the following evaluation criteria as the basis of its engineering philosophy:

- **Safety** of personnel, the vehicle, and the environment are a top priority. This is directly reflected in every step of the design process through redundancy in critical systems and improved visual aids for system monitoring.
- **Modularity** makes the system elements interchangeable, allowing for rapid integration and testing. Tools are initially developed at base-level capability and expanded as time and resources allow for improved capability.
- **Performance** focused development is used to evaluate the level of performance each system can achieve. Each subsystem has its own metrics for performance and targets it is seeking to achieve.
- **Novelty** is sought as a means to provide more capability than its competitors, but novelty must be balanced with anticipated performance and safety.

2) Systems Engineering: The team created a unique process and approach called Minion Process, shown in Figure

1, to manage systems engineering. Minion Process utilizes traditional systems engineering sub-processes, such as requirements generation, brainstorming, and design reviews, but continuously re-evaluated deadlines to integrate with its members schedules. This process is based on several well-known systems engineering approaches like Agile [3], Scrum [4], and Jira [5], but is adapted to the balance of academic, research, and team needs. For example, the use of 2 week sprints in Agile and continuous development in Lean have proved to be suboptimal for student volunteers with complex, unpredictable, and non-overlapping schedules.

Minion Process enables the completion of tasks that vary in complexity, scope, and objectives, often following non-linear timelines. For example, the tasks in a course or graduate research project differ significantly from those handled by a volunteer team. Furthermore, if deadlines need to be re-assessed a team lead or advisor must approve the changing deadline. But regardless of the timeline all sub-processes are followed. Once a product prototype is complete, it is tested through software analysis (simulation, ANSYS, etc.) and then put through a rigorous design review with an advisor and team members from the UAV, Hardware, and Software sub-teams. The design can then either be accepted and moved into final development (such as finishing operations and optimization) or rejected for additional improvements. If rejected, a new set of criteria is established for design refinement and Minion Process is re-started.

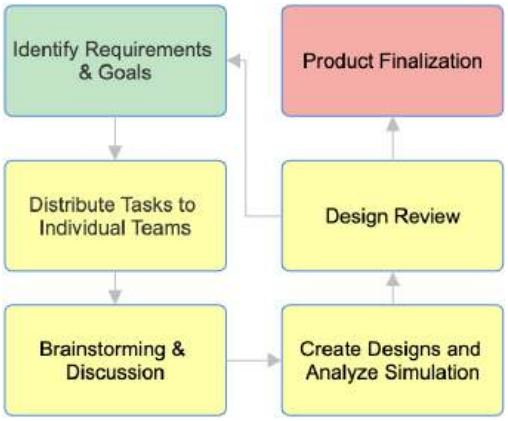


Fig. 1: Systems Engineering Adaptation: Minion Process

3) Design Focus: While Team Minion was highly successful at the 2022 competition, a post-competition analysis of successes and deficiencies identified four key areas where improvement was needed. These are discussed in detail in the published work of [1], and are concisely summarized below:

- **Minion Controls**: Minion's controls scheme was poorly tuned and was limited by the under-actuated nature of differential thrust. This made navigation difficult in high winds and currents, which happened often at competition.
- **Computer Vision**: Vision-based tasks were challenging due to poor camera synchronization, calibration, and training.
- **UAV System**: The only 2022 UAV goal was to get it to take off and fly reliably. This led to the UAV, named Kevin, having limited communications with Minion and poor camera-based object recognition, which are capabilities needed on key tasks.
- **Other Sub-systems**: De-emphasizing Minion's competition-specific sub-systems meant the team never developed a racquetball delivery system or hydrophone array in 2022. This reduced Minion's point earning potential on the entry/exit gate and detect and deliver tasks.

Addressing these four focus areas led to development of hardware tools for directional thrust (Appendix B) and a racquetball launcher (Appendix C), as shown in Figure 2. The UAV, Kevin, has also undergone significant development, with integration of ROS for communication with Minion, and autonomous vision-based landing procedures (Appendix E). Newly developed software tools include a patent-pending control scheme (again Appendix B), YoloV8 vision-based object detection, and Light Detection and Ranging (LiDAR)-based regions of interest (ROI) extracted from imagery for color-detection (Appendix D). For brevity, the details of these tools have been moved to the appendices. As part of Minion's Design Strategy, these appendices were originally written as part of a published document, thesis, or class report.

Fig. 2: Diagram of new hardware components added

Minion's 2024 software architecture is presented in Figure 3. This architecture shows how the new modules for ROI color Extraction, YoloV8, and feedback from Kevin are integrated into the prior software tools. The team's approach to individual tasks in 2022 involved identifying starting criteria and executing a state machine for completing each task, as described in the published work of [1]. With the introduction of new tools for 2024 and new task descriptions [2], each task has undergone a rigorous review by the software team to update the starting criteria and state machine. Furthermore, the task manager, MinionTask, dynamically determines the order in tasks should be completed based on optimization criteria of expected value (discussed in Subsection I-D), the time required to complete a task, and any constraints on task order. MinionTask is key to handling the complexity of the Semi-Finals and Finals Course [6].

C. Testing Strategy

Team Minion places a strong emphasis on testing platform components and software. The testing process used by the

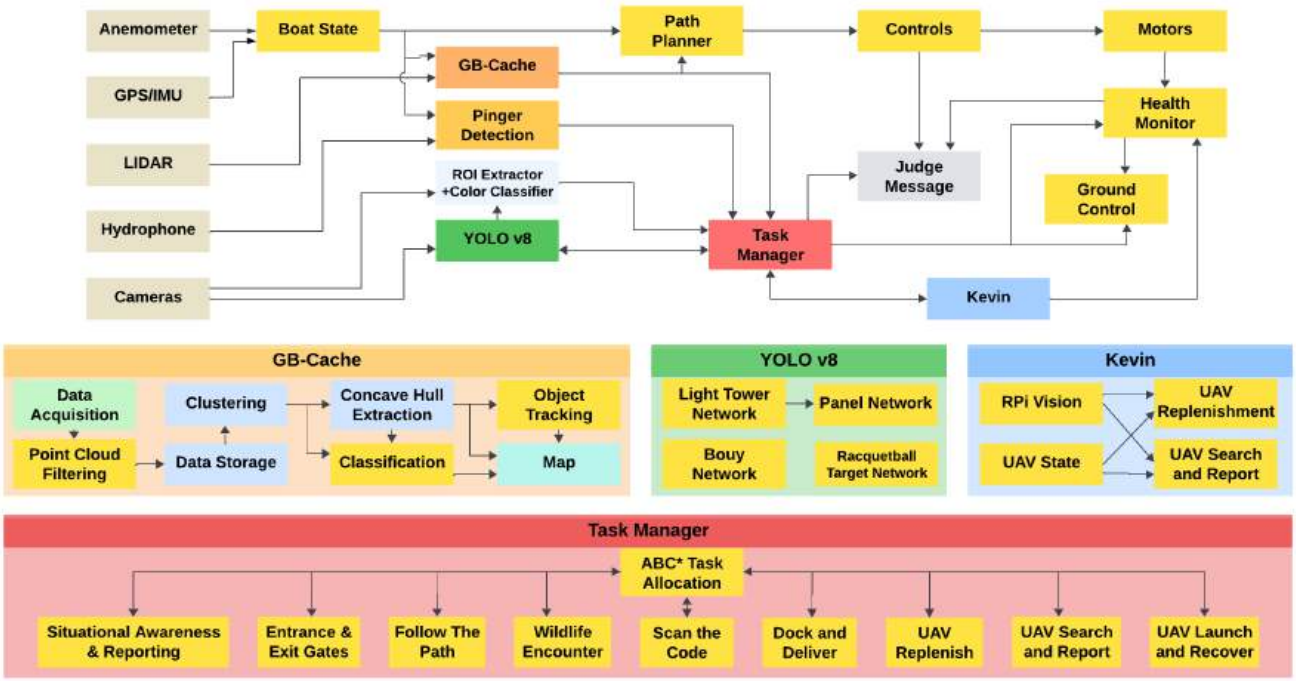


Fig. 3: Software Architecture Flow Chart for Minion

team is broken into a hardware, software, and UAV testing strategy. Each strategy has a tiered structure that emphasizes safety, performance, and effective time allocation of team members.

1) Hardware Testing Strategy: Minion uses the same hardware testing strategy it has used for the last two competition cycles. This process builds incrementally more refined solutions through a series of tests that begin with minimal overhead effort and risk, but end with full-scale testing with the integrated system. The stages of the hardware test strategy are:

1) **Computer-Aided Analysis** consists of using software tools, such as Solidworks, ANSYS, LT-Spice, and Advance Design Systems (ADS) to rapidly test system form, fit, and function of design solutions. This initial stage is used to size components, test loading conditions, and evaluate electrical performance.
2) **Bench Tests** test initial prototypes to ensure it performs the intended operation. In this test the team tests the limits of the design in a safe and controlled setting. Failure points are found and corrected for additional testing or additional design iterations.
3) **Boat Integration**. This is the most intrusive and time consuming testing phase as it causes a pause in all other testing of Minion. As such, boat integration is strategically scheduled at least 2 weeks in advance of on-water testing. Once integrated, in-lab tests are conducted to ensure new installs communicate and operate as expected with Minion's other systems.
4) **On-water testing**. These tests are generally held once a month and consist of multiple test objectives, both hardware and software. On-water tests require at least 4 individuals: an RC driver, manned support vessel (MSV) captain, dock hand, and ground station operator. Test plans are created for on-water testing to identify test conditions and success/failure criteria.

Fig. 4: Image of Minion during On-Water Testing on 9-7-2024, which tested Azimuth Actuation, Livox LiDAR functionality, and time synchronization of LiDAR and vision sensors

2) Software: Since the transition to ROS in 2022, team Minion has used the VRX-supported simulation of the WAM-V to evaluate all perception and decision-making processes before on-water testing. Additionally, for 2024 Team Minion has created a simulated qualifying and finals course based on the 2024 Handbook 2.0 [2], as seen in Figure 5. By first testing all software in the simulated environment, the team can address major issues prior to deployment on the ASV,

allowing on-water tests to focus primarily on fine-tuning real-world performance.

An in-lab vehicle shakedown is conducted the day before on-water testing to verify recent changes and ensure compatibility between the modified code and prior iteration. This process minimizes the risk of wasting valuable on-water time fixing compatibility issues. The shakedown involves several key checks including verifying successful code compilation, ensuring all necessary dependencies are installed on the ASV, confirming the responsiveness of the propulsion system, and ensuring correct networking for the sensors.

During on-water tests, ROS bag files are used to capture all sensor data and inter-algorithm communication. These bags can be used later for performance assessment, and algorithm testing by treating the recorded data as live observations. Camera data however, is stored separately from bag files due to memory considerations. Imagery is logged using H.265 video encoding, which uses significantly less local memory than storing individual images in bag files. For instance, a 2 minute video from Minion's High-Dynamic Range (HDR) camera typically requires approx. 500MB of data storage, but when frames are stored in bag files this balloons to approximately 25GB.

On-water tests of Minion's hardware and software systems occur simultaneously. Tests occur every 2-8 weeks, depending on down time for installations and the readiness of system software. Because of the logistic burden of on-water testing, there is a general expectation of at least three separate test objectives before the test is scheduled. As the competition approaches, on-water testing may increase in frequency and with fewer, but more time critical, test objectives. Minion may be operated via remote control or autonomous modes during software testing, but safety pilot is always ready to take over autonomous operation.

Following each test, the team conducts a debrief involving all present members. The debrief covers the key accomplishments from the test, as well as any issues or tasks that need to be addressed for the next test. These discussions serve as the starting point for planning the next testing cycle, helping to ensure the team stays focused on improvements and preparations for future tests.

3) UAV: Rigorous testing procedures were implemented for the UAV due to the inherent risks of operating over water and near people. Similar to the ASV platform, the team utilized the VRX simulation environment extensively during the development phase to test and validate the software for autonomous tasks. This approach significantly minimized the risk of accidental crashes during real-world testing. To further ensure the safety of the pilot and crew members, as well as to mitigate risk, multiple checklists were created. These include distinct checklists for the initial build, hardware tests, software tests, routine testing, and autonomous operations. Depending on the type of test being conducted, the relevant checklist is consulted before the test begins and after it is completed, ensuring safety protocols are maintained and all necessary steps are followed.

Tests are conducted in a netted enclosure measuring approximately 50 ft x 50 ft x 30 ft, except when over-water testing is required. This enclosed space allows for safe and controlled testing of the UAV by minimizing risk of environmental interference or accidents. For testing over water, which is a public area, the UAV team needed to obtain permission from a University Safety Review Board (SRB) and the local authority. To gain these permissions Team Minion promised a series of tests that would incrementally build the autonomous capabilities of the UAV. As part of the preparation for this year's competition, the following tests were conducted: 1) On-water testing, 2) Take-off from the ASV, and 3) Landing on a platform. Each test adhered to a structured procedure consisting of four defined steps:

- In cage manual test.
- In cage autonomous test.
- On water manual test.
- On water autonomous test.

In the event of a failure at any stage, the UAV was required to restart from step 1 and successfully complete all prior steps before proceeding.

D. Competition Strategy

New hardware and software tools should allow Minion to perform even better at the 2024 competition. To design a competition strategy around new and existing tools, Team Minion needed to characterize the performance of each tool. This led to assessing the confidence in each tool on a 0 to 100 scale, as shown in Table I. Confidences are based on prior competition results with existing tools and on simulation and on-water test results for new tools. These confidence values were reviewed by hardware members, software members, and team advisors for consensus on generalized performance rather than task-specific performance. However, during this process the team made note that its confidence may change in certain situations, such as low wind conditions and when trying to perceive objects the team tested more extensively.

TABLE I: Confidence Level in Minion's Toolset (0- no confidence, 100 - perfect confidence)

Tool	Confidence (0-100)
Hardware Tools	
-UAV Flight	75
-UAV Pickup/Drop-Off	20
-Racquetball Delivery	50
-Hydrophone System	5
Navigation Tools	
-Path Following	99
-Object Circling	80
-Station-Keeping	75
Perception	
-Mapping	90
-Object Recognition	80
-Color Identification	65
-Fuse Map & Vision	50

The team then used the tool confidences and the task descriptions in the RobotX Handbook v2.0 [2], to determine the relative importance of each task. This procedure started with assessing the complexity of each of the nine on-water tasks as High (H), Medium (M), or Low (L). In this context, complexity generally refers to the number of co-dependent

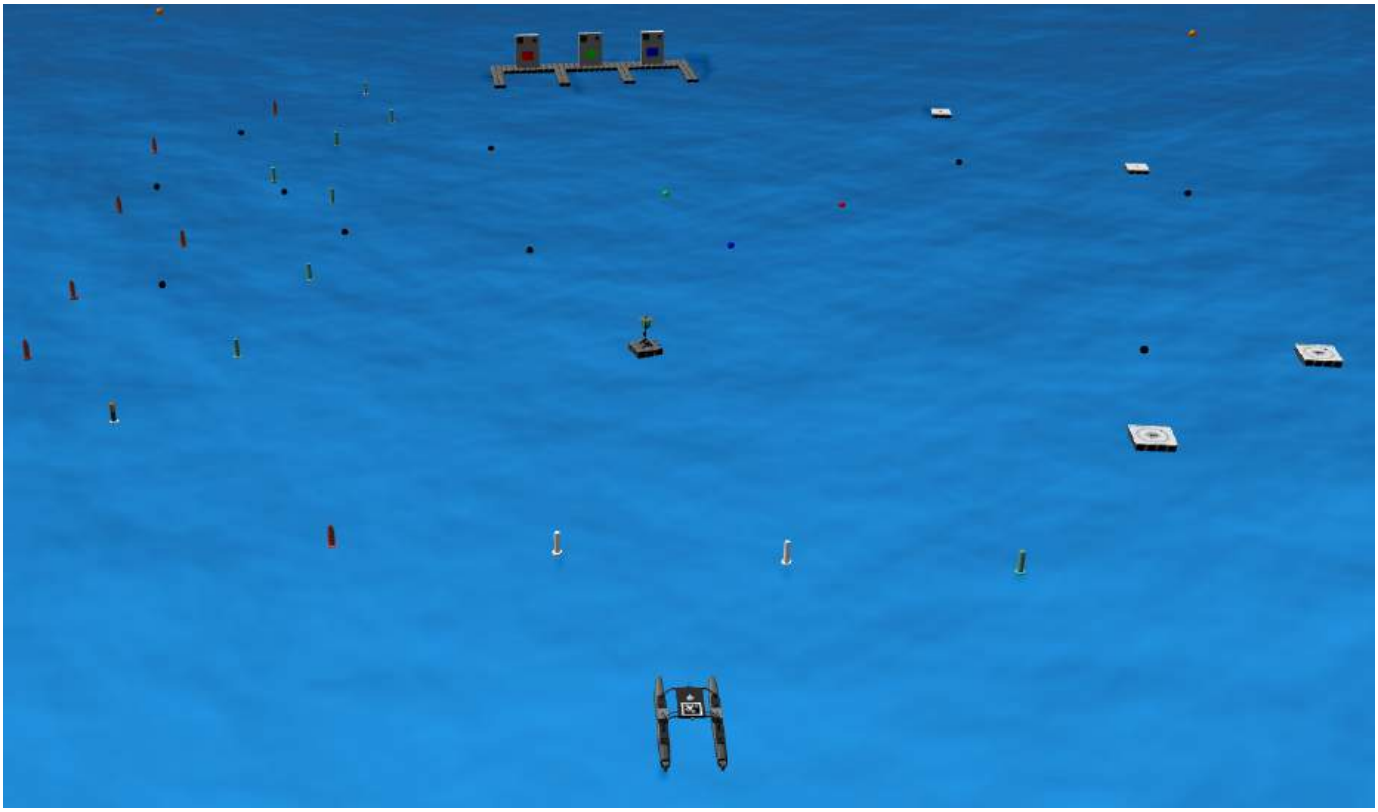


Fig. 5: Finals Course built by Team Minion in Gazebo using Virtual RobotX Tools [7]. The team also built a qualifying course, which changes some course elements and has more spacing between tasks.

Fig. 6: On-water testing of Kevin on 5-12-2024, which tested UAV replenishment landing procedure.

tools and steps required to complete a task. For instance, Follow the Path has a high complexity because it requires Minion's path following, mapping, object recognition and color identification tools to build a successful task solution, while Scan the Code only requires station-keeping and object color identification tools and therefore has a low to medium complexity. These complexities are shown in Figure 7.

In addition to complexity, the team also had to assess the value of each competition task. The competition handbook [2] identifies the maximum points to be earned on each task. However, the value of attempting a task is not solely determined by task points. There are two competition tasks that must be attempted before any other tasks can proceed: Situational Awareness and Entry Gate. As a result, these tasks have the highest priority for development purposes. However, for the remaining tasks, the point value of each task can be decomposed into three separate quantities:

- **Core Points**: The points that can be earned on the task regardless of performance on other tasks.
- **Contingent Points**: The points that can be earned on the task only with knowledge gained from a prior task or after completing a prior task.
- **Future Points**: Points that can be potentially earned on future tasks after completing part or all of the this task.

As shown in Figure 7 the value of some tasks is significantly altered by contingent or future points. While this point decomposition shows the value of a task, it does not yet reflect Team Minion's confidence in being able to complete a task. That is, while a task may be valuable the task complexity and the team's confidence in completing the task should also affect the task value. For this reason the team developed a formula for determining the estimated value of task i, which is described as

Task	Complexity	Maximum Points	Core Points	Contingent Points	Future Points	Confidence	Estimated Value	Priority
Situational Awarness	L	100	100	0	0	~100%	*	1
Entry/Exit	L	1200	1200	0	0	~100%	**	2
Scan the Code	L-M	600	600	0	1900	75%	1500	3
Detect & Dock	M	2000	900	1100	0	70%	850	4
Wildlife Encounter	M-H	1100	600	500	0	70%	750	5
UAV Launch/Recovery	M-H	1500	1500	0	2500	75%	550	6
Follow the Path	M-H	1200	1000	200	0	40%	525	7
UAV Search & Rescue	M	1000	0	1000	0	50%	350	8
UAV Replenishment	H	2300	0	2300	0	20%	325	9

Fig. 7: Estimated value of developing solutions for individual tasks and attempting the task competition qualifying and finals stages. Estimated value is based on Equation 1 which uses the task and contingent task confidence, scoring and future value.
* Task must be completed to get onto semi-finals/finals course
** Part of task must be completed to get onto semi-finals/finals course

$$EV(i) = c_i \left(core_i + \sum_{j=1}^{9} c_j * cg_{i,j} + \sum_{k=1}^{9} c_k * cg_{k,i\prime} \right) \quad (1)$$

where c_i is the confidence in completing task i and $cg_{i,j}$ is the portion of task i's contingent points that depend on task j. For brevity, a list of $cg_{i,j}$ values has not been included in this document, but can be decomposed from the RobotX Handbook v2.0
refref2. This results in a relative estimated value, EV, of each task. The estimated value is then used to rank the team's priority in attempting the task during qualifying, semi-finals, and finals round of competition. The analysis of Figure 7 shows that while the AMS can potentially complete parts of all nine scored tasks, tasks like Scan the Code have much higher value than other tasks.

Team Minion operates a unique approach to task specification. Minion's task manager, MinionTask, dynamically decides task order by incorporating the estimated value of each task, known course information, and the remaining run time. Team Minion can also turn off use of any task in the event the team's confidence drops to almost zeros, such as in the case of a hardware failure. As such, it is generally expected that the confidences listed in 7 will change in the days leading up to competition and even throughout competition itself. Therefore, the estimated value will be adjusted before competition semi-finals and finals. With confidences above 70% for six of the nine tasks, Minion should be able to qualify for semi-finals quickly and also be highly competitive during later rounds of the competition.

II. Conclusion

Team Minion is confident in its approach to the 2024 competition. This stems from a systems engineering process for developing improved AMS solutions, a testing strategy that builds incrementally more robust solutions, and a competition strategy that balances the strengths of Minion with the challenge and scoring of the competition. Additional resources on the technology developed for Minion can be found in the Appendices attached on later pages of this document. Outside of the component list, which is Appendix A, the remaining appendices highlight new systems developed for the 2024 competitions. Furthermore, in lieu of the optional Appendix B on Test Results, Team Minion has chosen to include test results with the each of the additional Appendices.

Acknowledgments

Team Minion would like to thank our generous external partners for their equipment and financial support over the last three years: Yamaha Motor Company, Volz, Torqeedo, RoboNation, Copenhagen Subsea A/S, Glenair, Mathworks, Northop Grumman, and DS Solidworks. Team Minion would also like to thank their internal partners: the Embry-Riddle Aeronautical University College of Engineering, Mechanical Engineering Department, and Student Government Association.

Many individuals have contributed significantly to the success of team Minion, but the team would like to give a special recognition to our wonderful alumni, who continue to support the team after many years through technical assistance. the team would also like to thank Bill Russo and Mike Potash for their technical assistance, and Christina Groenenboom for logistics support.

References

[1] A. Lachguar, S. Sarkar, E. Coyle, and P. Currier, "Minion: The autonomous surface vessel that won the 2022 Maritime RobotX Challenge," *Naval Engineering Journal*, 2022.

[2] RoboNation, *RobotX Challenge 2024 Team Handbook v2.0*, Robonation, 2024[Online]. Available: $https://robonation.org/app/uploads/sites/2/2024/09/2024-RobotX_Team-Handbook_v2.0.pdf$

[3] Project Management Institute. (2017). Agile practice guide. Project Management Institute.
[4] K. Schwaber and J. Sutherland. (1991) The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game. [Online]. Available: www.scrum.org
[5] Atlassian. (2024) JIRA.[Software]. Available: https://www.atlassian.com/software/jira
[6] J. Hendrickson. "Dynamic Task Allocation in Partially Defined Environments Using A* with Bounded Costs," M.S. thesis, Mechanical Engineering, Embry-Riddle Aeronautical University, Dayatona Beach, 2021. [Online]. Available: https://commons.erau.edu/edt/592/.
[7] B. Bingham, C. Aguero, M. McCarrin, J. Klamo, J. Malia, K. Allen, T. Lum, M. Rawson, and R. Wagar, (2019) "Toward Maritime Robotic Simulation in Gazebo" (Version 1.6.2) [Source code]. Available: https://github.com/osrf/vrx/tree/gazebo_classic.

Appendix A: Component List

Minion

Component	Vendor	Model/Type	Specs	Custom/Purchased	Cost	Year Of Purchase	Reasoning
Waterproof Connectors	Glenair	Series 804 Connectors	https://tinyurl.com/5cr27re2	Purchased	$407.03	2016-Present	Partnership
Propulsion	Copenhagen Subsea	VM Thrusters	https://tinyurl.com/28njj67r	Purchased	$8,000.00	2018	Marine Safe, Efficiency
Power System	Torqedo Power	26-104 batteries	https://tinyurl.com/34asxtrz	Purchased	$2,999.00	2020	Built In Protections
Motor Controls	Piktronic	SAC1-90A AC Drive Controller	https://tinyurl.com/c27zr8bz	Purchased	$2,000	2018	Sold with VM thrusters
CPU	Intel	Xeon E5-2620v3 6-Core 2.4 Ghz	https://tinyurl.com/5yhjr6fx	Purchased	$129.95	2018	High End 2018 Single Thread
Teleoperation	Ubiquiti	RP-5AC-GEN-Rocket Prism	https://tinyurl.com/ym5x7zsc	Purchased	$249	2024	Upgrade of old system
Compass / **Inertial Measurement Unit (IMU)**	Torq Robotics	Pinpoint	Proprietary	Custom	N/A	N/A	Partnership
Doppler Velocity Logger (DVL)	N/A	N/A	N/A	N/A	N/A	N/A	Not Needed
Camera(s)	Leopard Imaging	LI-IMX490 5MP	https://tinyurl.com/53jw5kdd	Purchased	$617	2019	High Dynamic Range
	Flir	Blackfly S USB3	https://tinyurl.com/589mk6fr	Purchased	$361	2019	High Resolution
Hydrophones	Teledyne	TC-4013	https://tinyurl.com/2nyva835	Custom	$250	2018	Partnership

Minion

Component	Vendor	Model/Type	Specs	Custom/Purchased	Cost	Year Of Purchase	Reasoning
Algorithms	ABC*,Deep Learning, Trilateration	N/A	N/A	Custom	N/A	N/A	Integration and Customization
Vision	YOLO V8	Ultralytics	https://tinyurl.com/yysstms2	Custom	N/A	N/A	Deep Learning Standard
Localization and Mapping	ERAU Team Minion	GB-CACHE	Publication Pending	Custom	N/A	N/A	Fast Update Rates
Autonomy	ERAU Team Minion	MinionTask	https://tinyurl.com/3cy97d8h	Custom	N/A	N/A	Adaptability
Open-Source Software	Open Source Robotics Foundation	ROS Noetic	https://wiki.ros.org/noetic	Purchased	$0	2018	System Integration Benefits
	Open Source Robotics Foundation	Gazebo	https://gazebosim.org/home	Purchased	$0	2018	System Integration Benefits
	Roboflow	Roboflow	https://roboflow.com/	Purchased	$0	2023	Ease Of Labeling/ Training
	Supervisely	Roboflow	https://supervisely.com/	Purchased	$0	2022	Ease Of Labeling/ Training

Additional Components

Additional Components	Vendor	Model/Type	Specs	Custom/Purchased	Cost	Year Of Purchase	Reasoning
LiDAR	Livox	Horizon	https://tinyurl.com/2hd3fzvp	Purchased	$999.00	2020	High End Mid Range Lidar
	Ouster	Velodyne VLP 16	https://tinyurl.com/2sc5vayn	Purchased	$400.00	2016	High End Close Range Lidar
Enclosures	Pelican	1520/1560/1730	https://tinyurl.com/yj9b478b	Sponsored	N/A	2016-Present	Water Proof
Light Panels	Adafruit	8x32 NeoPixel RGB LED Matrix	https://tinyurl.com/b5nn9j2y	Purchased	$59.50	2018	High Visibility
Servos	VOLZ	Volz DA-36 LP servo	https://tinyurl.com/446fds3m	Sponsored	N/A	2021	High Torque
	Linak	LA36	https://tinyurl.com/4y9dsrdd	Sponsored	N/A	2016	Rugid and water resistant
GPU	Nvidia	GTX 1080FE	https://tinyurl.com/mr3746xw	Purchased	$598.00	2014	High End GPU At The Time

kevin

Component	Vendor	Model/Type	Specs	Custom/Purchased	Cost	Year Of Purchase	Reasoning
UAV Frame	Tarot	Iron man 650	https://tinyurl.com/jteu8zp7	Purchased	$119.17	2022	Lightweight, durable frame ideal for UAV
Motors	T-Motors	U5 KV400	https://tinyurl.com/5avm82uz	Purchased	$503.60	2022	Light and High Thrust
ESCs	T-Motors	ALPHA	https://tinyurl.com/42469tdk	Purchased	$279.99	2022	For supplying sufficient amps to the motors.
Propellers	T-Motors	Carbon Fiber	https://tinyurl.com/468kasy6	Purchased	$251.66	2022	Optimized for high efficiency and thrust.
Power Distribution Board	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Autopilot	CUAV-Pixhawk	V6X	https://tinyurl.com/yc5c4bwx	Purchased	$329.00	2024	Controls UAV flight autonomously.
GPS	CUAV	NEO 3	https://tinyurl.com/2k62ws2z	Purchased	$89.00	2024	Accurate Positioning Data
Transmitter	Taranis	QX7	https://tinyurl.com/4cwk7ytw	Purchased	$124.00	2022	Used for manual control.
Receiver	FrSky	X8R	https://tinyurl.com/4dcp56w6	Purchased	$36.99	2024	Long Distance Reciever
Battery - UAV	Yoowo Power, Ovonic	LiPo	https://tinyurl.com/5n7d7bup	Purchased	$163.99	2022	Supplies power to the UAV.
Battery - Transmitter	Lumenier	LiPo	https://tinyurl.com/yub5xkjj	Purchased	$28.99	2022	Powers the transmitter.
Battery Mount	N/A	3D Printed	N/A	Purchased	$0.00	2024	Rigid and replaceable
Onboard Computer	Raspberry Pi	4B	https://tinyurl.com/2s4ynr22	Purchased	$75.00	2022	Fast and easy to use.

Kevin

Component	Vendor	Model/Type	Specs	Custom/Purchased	Cost	Year Of Purchase	Reasoning
Camera	Raspberry Pi	HQ 12MP	https://tinyurl.com/nhzp37tk	Purchased	$50.00	2022	Captures high-resolution imagery
Pool Noodles	N/A	N/A	https://tinyurl.com/3sru3yr5	Purchased	$25.99	2024	Shock Absorption And can float
Telemetry Module	Holybro	V3	https://tinyurl.com/mh2e8jar	Purchased	$61.99	2024	Fast Transmit Rate
Remote ID Module	Holybro	N/A	https://tinyurl.com/35nvb5bb	Purchased	$29.59	2024	FAA Compliance
Propellers Guards	N/A	N/A	https://tinyurl.com/3sac2jkp	Custom	$17.99	2024	Easily Fabricated Cost
Ferromagnetic Plate	CMS MAGNETICS	Stainless Steel	https://tinyurl.com/3c7pusze	Purchased	$12.99	2024	Ferromagnetic Properties
Magnets	TRYMAG	Neodymium	https://tinyurl.com/mr42shd4	Purchased	$79.99	2024	To capture Tin

Appendix B: Control System

Sarthak Aggarwal, Dr. Eric Coyle, Dr. Darris White

I. Introduction

THIS appendix details the development of Team Minion's control strategy for the 2024 Maritime RobotX competition. Building upon previous experiences with the autonomous surface vessel (ASV) Minion, the team has focused on refining the robustness and precision of the vessel's control systems. Past challenges, such as significant control errors and limitations due to an under-actuated system, highlighted the need for improved maneuverability and reliability. By addressing these issues through the implementation of full azimuth control and independent motor steering, Team Minion has transitioned to a new control strategy with an over-actuated control system. This advancement enhances the maneuverability and path following. The subsequent sections provide an in-depth examination of the methodologies, hardware implementations, and results that show Team Minion's latest control strategy enhancements.

II. Literature Review

The field of marine vessel control has seen significant advancements, with companies like Brunswick Corporation and Yamaha developing sophisticated systems to enhance the maneuverability and efficiency of vessels. Numerous patents filed by these industry leaders reflect their investment in technologies for steering, thrust management, and autonomous control.

Brunswick Corporation has pioneered a wide array of control systems for marine vessels, focusing on methods for precise docking, positioning, and maneuvering. Patents such as [1] and [2] outline systems designed to control vessel movement laterally using differential thrust and advanced control mechanisms. Brunswick's joystick-based systems (e.g., [3]) allow for seamless control of multi-engine setups, greatly improving vessel maneuverability during complex operations like docking and close-quarters navigation. Additionally, Brunswick's station-keeping systems [4] enable marine vessels to hold position relative to fixed locations, employing global positioning systems and other sensor-based technologies.

Yamaha has also developed control systems that leverage advanced steering and display technologies for marine vessels. For instance, their marine vessel running control system [5] integrates steering with engine control to provide automated adjustments, enhancing overall vessel handling. Yamaha's focus on integrating vessel display systems with control strategies [6] provides operators with improved situational awareness, allowing for more efficient and safer operation.

While Brunswick and Yamaha's contributions focus on enhancing operator control through joystick methods, differential thrust, and integrated display systems, Minion's approach to propulsion control differs significantly. The primary motivation behind our strategy is the implementation of an autonomous, over-actuated control system that provides independent azimuth control of rim-driven propellers. Unlike joystick-based systems or manually operated differential thrust, our system autonomously determines optimal steering and thrust settings based on real-time feedback. This allows Minion to dynamically balance surge, sway, and yaw movements without human intervention, an approach fundamentally distinct from the manual or semi-autonomous systems developed.

Our goal in implementing this strategy on Minion is to push beyond operator-assist systems and towards fully autonomous maritime navigation. By applying advanced control algorithms that prioritize avoiding thrust reversals and minimizing azimuth changes, our system maximizes efficiency while reducing mechanical wear. These innovations demonstrate our commitment to developing control architectures that are specifically tailored for autonomous surface vessels like Minion.

III. Hardware Setup

Minion utilizes two VM thrusters from Copenhagen Subsea with asymmetric nozzles for main propulsion, which give thrust in both forward and reverse directions. The propulsion system is configured to allow Minion to azimuth and beach both thrusters.

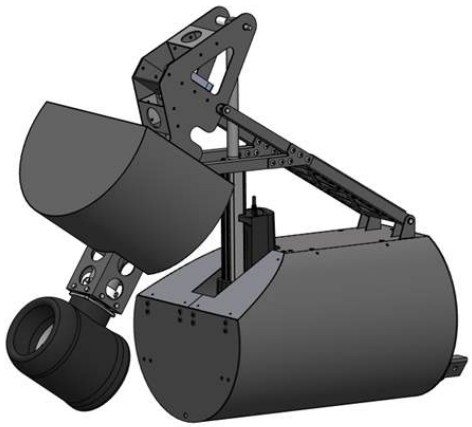


Fig. 1. Minion's Propulsion System [8]

Azimuth control is achieved with Volz DA-36 Low Profile servos[10], which produce 11 N-m of torque and allow the thrusters to rotate ± 100°. We have limited the range to ± 90° to leave room for calibration and our Control Strategy uses the same range for control. These servos are a 200% increase in torque over the DA-26 servos used in 2018,

eliminating a previous issue with holding strength. The DA-36 servos are environmentally rated to withstand salt water and uses the Glenair Superfly 881-001PB-H10W-M200J5-24 connector which provides excellent water resistance. For sending commands to the servo, Minion sends 6 Byte serial commands via the RS-485 communication protocol at a baud rate of 115200 bits/s with the following structure:

Byte #	Description
1	Command / Response-Code
2	Actuator ID
3	Argument 1
4	Argument 2
5	CRC High Byte
6	CRC Low Byte

IV. Control System Architecture

The control system architecture for Minion has been significantly refined to address the challenges of autonomous navigation and precise vessel maneuvering. The architecture integrates feedback control, inverse kinematics, and over-actuation via independent motor steering to provide enhanced control over the vessel's three degrees of freedom, specifically in **surge, sway, and yaw** directions. This section outlines the main components of the control system, detailing the key advancements implemented for the competition.

A. Control System Overview

The 2024 control system for Minion is designed around a fully over-actuated configuration that allows for independent control of each propulsion pod. This configuration enables flexible maneuvering by independently steering each motor pod, as well as providing differential thrust control. The architecture leverages multiple control modes such as pose control, path following, and transitional states, enabling the system to adapt dynamically to different operational scenarios.

The propulsion system consists of two motor pods, each capable of being steered azimuthally along with the motors being capable of providing thrust in both forward and reverse directions, providing full 360-degree thrust vectoring. This setup gives the control system the ability to generate forces and moments in all directions (surge, sway, and yaw) while optimizing energy efficiency and ensuring precise path tracking.

B. Inverse Kinematics and Thrust Allocation

The control system uses an inverse kinematic model to translate the desired forces and moments into steering angles and thrust commands for each motor pod. The inverse kinematic equations calculate the exact angles and thrusts needed to achieve the desired sway force, surge force and yaw moment, accounting for the asymmetrical capabilities of the propulsion devices, such as differences in forward and reverse thrust limits.

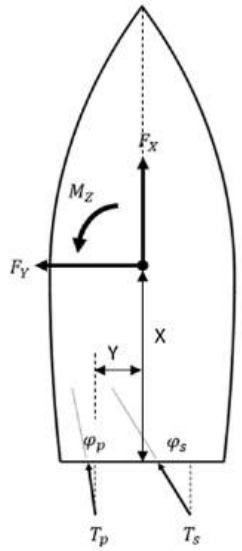


Fig. 2. Forces and Moment Definition

The key relationship governing the inverse kinematic model is expressed as follows:

$$\begin{bmatrix} F_x \\ F_y \\ (M_z + F_y X)/Y \end{bmatrix} = \begin{bmatrix} T_p \cos(\phi_p) + T_s \cos(\phi_s) \\ T_p \sin(\phi_p) + T_s \sin(\phi_s) \\ -T_p \cos(\phi_p) + T_s \cos(\phi_s) \end{bmatrix} \tag{1}$$

where F_x, F_y, represent the forces in the surge and sway directions and M_z being the yaw moment, respectively. T_p and T_s are the thrusts of the port and starboard motors, and ϕ_p and ϕ_s are the steering angles. Y is the distance between the motor and the center of mass of the vessel. This inverse kinematic model ensures that the propulsion commands respect the physical limits of the system, while providing the necessary forces and moments for the vessel's movement.

These three equations relate the thrust forces F_x and F_y (in the surge and sway directions) and the yaw moment M_z to the motor pod thrusts T_p and T_s and their steering angles ϕ_p and ϕ_s. However, since we are controlling four variables $(T_p, T_s, \phi_p, \phi_s)$ and we only have three equations, the system is under-constrained. This means that there is no unique solution unless we provide an additional constraint or equation to fully specify the system. These equations are selected from the workspace classifier explained in the Force/Moment Workspace and Classification section.

C. Motor Thrust Modelling

This section presents the motor thrust modeling for Minion based on experimental data collected during on-water testing. Figure 3 illustrates the relationship between the motor command (throttle) and the average current drawn by the motors, with both port and starboard motors set to a zero-degree azimuth angle.

From the data, it is observed that the current draw, which is directly related to the thrust produced by the motors, exhibits a nonlinear relationship with the motor command. Both forward (FWD) and reverse (REV) thrusts are represented. Interestingly, the current draw in reverse is consistently higher than in forward, indicating that Minion generates greater thrust when operating in reverse. This asymmetry is likely due to the skin drag effect between the pontoons and the water, which alters the hydrodynamic behavior of the vessel.

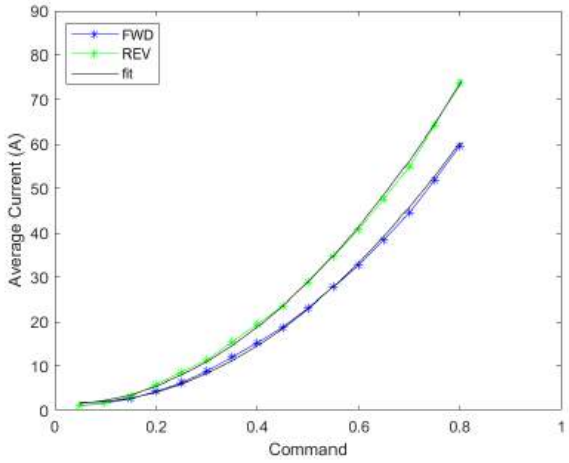


Fig. 3. Comparison of average current vs. command for forward (FWD) and reverse (REV) thrust. The fitted curve is shown for reference.

The relationship shown in this figure is specific to the zero-degree azimuth configuration. It is important to note that as the azimuth angle of the motor changes, the thrust characteristics will also vary. The non-linear nature of the current-to-thrust relationship suggests that when the motors are angled (e.g., during maneuvers), the efficiency and current draw will differ from the data shown here. This will be taken into account in the control algorithms, ensuring that the dynamic thrust is adjusted appropriately for different motor orientations to maintain optimal maneuverability.

The fitted curve for both forward and reverse thrust serves as the basis for generating a control model, which can predict motor performance based on input commands. These curves will be utilized in the controller to calculate thrust outputs, ensuring accurate and efficient control of Minion during navigation.

D. Feedback Control Design

The control system relies on feedback control algorithms to regulate the vessel's motion. For each degree of freedom, a Proportional-Integral-Derivative (PID) controller is employed to minimize error between the desired and actual vessel state. These controllers are tuned for optimal performance under various operating conditions.

1) **Surge and Sway Control:** The primary objectives in surge and sway control are to ensure accurate forward/backward movement and lateral stability, respectively. The system dynamically adjusts thrust commands to maintain the desired velocities while minimizing errors in path following mode.
2) **Yaw Control:** Yaw control is managed by independently steering the motor pods, allowing precise heading adjustments. The feedback system calculates the required yaw moment and distributes the necessary thrust commands accordingly.

The controllers are designed to switch seamlessly between different modes, ensuring smooth transitions between dynamic and stationary states, such as moving from path following to pose control mode when approaching a target.

E. Operating Modes

The control system operates in several distinct modes to handle different mission requirements:

1) **Station-Keeping Mode:** This mode is responsible for maintaining the vessel's position and orientation when in station-keeping mode. The control system simultaneously manages surge, sway, and yaw motions to minimize positional error.
2) **Path Following Mode:** In this mode, the vessel follows a predefined path with a specific requirements. The system prioritizes controlling surge and yaw, while allowing sway to be minimally controlled or ignored when the vessel is far from the target. The control system uses a trajectory planning algorithm to determine the optimal path between waypoints.

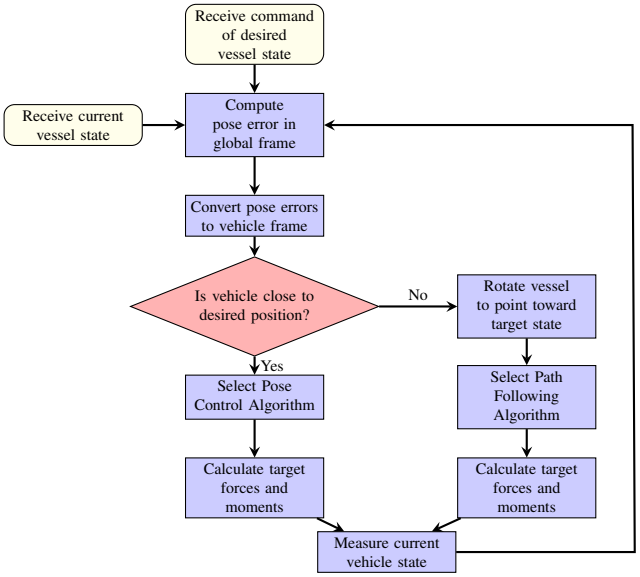


Fig. 4. Mode Selection Flowchart

F. Efficiency and Actuator Management

To ensure optimal performance and durability of Minion's actuators, the control system allows the use of any cost function, including those prioritizing energy efficiency or minimizing motor stress. However, for Minion, the chosen cost function primarily aims to avoid switching between positive

and negative thrust, as abrupt thrust reversals can introduce inefficiencies and increase wear on the actuators. The second priority is to minimize changes in the azimuth direction, reducing unnecessary steering adjustments and mechanical strain on the motors.

The control system applies a set of constraints within the force/moment workspace to ensure that the solutions fall within feasible operating limits. When multiple solutions exist, the system selects the solution that results in the smallest change in azimuth angle, balancing maneuver smoothness and operational effectiveness. This method prioritizes smooth transitions and stability, ensuring the vessel operates efficiently without placing excessive demand on the propulsion system.

$$\begin{aligned}\text{Cost} = a_1(T_p^+ - T_p^-)^2 + a_2(T_s^+ - T_s^-)^2 \\ +a_3(\phi_p - \phi_{p,\text{current}})^2 + a_4(\phi_s - \phi_{s,\text{current}})^2\end{aligned} \tag{2}$$

The tunable coefficients a_1, a_2, a_3 and a_4 allow the system to emphasize different priorities, such as avoiding thrust reversals or minimizing azimuth changes, depending on Minion's current mission requirements.

V. Force/Moment Workspace and Classification

The control system for Minion is designed to manage the complex dynamics of an over-actuated vessel by calculating the forces and moments necessary to achieve the desired motion in all degrees of freedom. The key to successful maneuvering lies in understanding the limits of the forces and moments the propulsion system can generate, which is defined by the Force/Moment workspace. This section outlines the characterization of this workspace, the classification of operating modes, and the strategy for selecting feasible propulsion configurations.

A. Achievable Force/Moment Regions

The force/moment workspace can be visualized as a volume in three-dimensional space, with the axes corresponding to surge force F_x, sway force F_y, and yaw moment M_z. The boundary of this workspace represents the maximum attainable forces and moments assuming equal amounts of forward and reverse thrust, and given the actuator constraints of the propulsion system.

- **Thrust Limits**: The forward and reverse thrust capabilities of the motors impose limits on the magnitude of the achievable forces in the surge direction. Typically, the forward thrust is greater than reverse thrust, introducing an asymmetry in the workspace.
- **Steering Angle Limits**: The azimuth steering range of each motor pod also defines the shape of the workspace. When the steering angles are constrained to particular values (such as $0°$, $\pm 90°$, or $180°$), certain forces and moments may become unattainable.

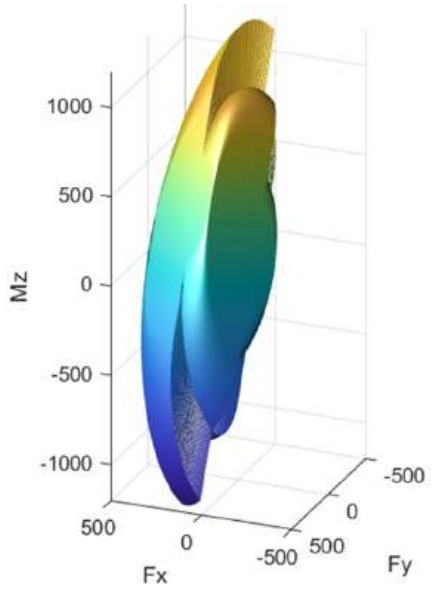


Fig. 5. Force/Moment Workspace

B. Classification of Operating Classes

To efficiently manage the vessel's motion within the limits of its force/moment workspace, the control system classifies the operating conditions into distinct *operating classes*. Each class represents a specific combination of thrust and steering angle constraints which when added to the kinematics of Equation 1 creates a deterministic solution for T_p, T_s, ϕ_s and ϕ_p.

The key operating classes are defined based on the relationship between the port and starboard thrusts and steering angles. The five primary classes that span the workspace are as follows:

- **Class 1: Port Steering Angle =** $0°$
  In this class, the port motor is fixed in a straight-ahead position ($\phi_p = 0°$), while the starboard motor is free to steer. This class simplifies control by reducing one degree of freedom, often used in forward motion where asymmetrical steering is not required.
- **Class 2: Starboard Steering Angle =** $0°$
  In this class, the starboard motor is fixed at $0°$, and the port motor is allowed to steer. This class is useful for maneuvers where the port motor needs to handle more complex motions, such as in tight turns or when compensating for disturbances from the starboard side.
- **Class 3: Equal Thrust on Both Motors**
  Both motors generate equal thrust, but their steering angles can vary independently. This class provides symmetrical force generation and is typically employed for surge motion or balanced yaw control.
- **Class 4: Port Thrust = Starboard Thrust** $\times\alpha$
  Here, the port and starboard thrusts are related by a constant ratio α, where α is defined by the maximum forward/reverse thrust limits. This class is useful for balancing forces during complex maneuvers, especially

when the system needs to maintain thrust asymmetry for fine control of yaw.

- **Class 5: Starboard Thrust = Port Thrust** $\times\alpha$
  Similar to Class 4, but with the roles of port and starboard motors reversed. This class is typically selected when a higher forward thrust is needed on one side of the vessel to maintain balance during tight maneuvers.

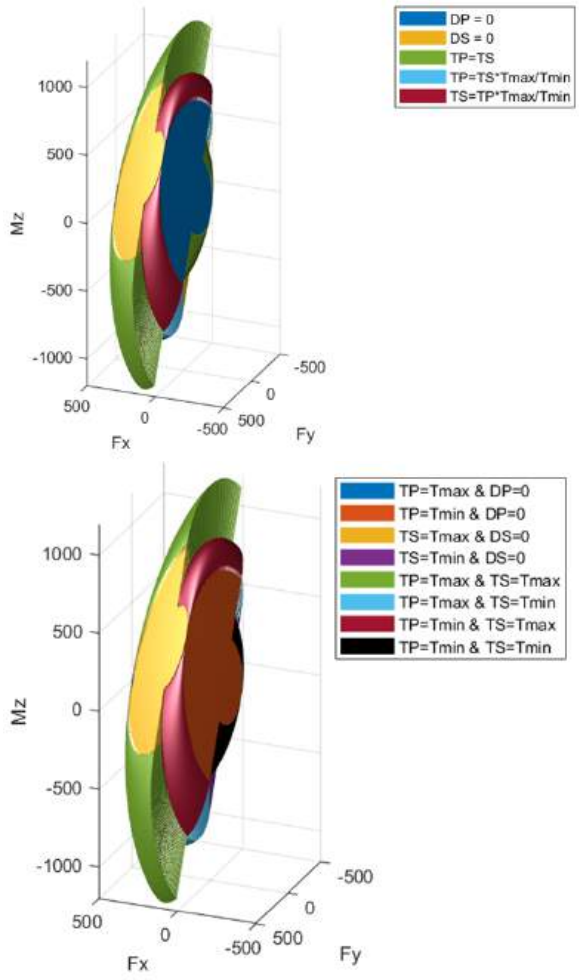


Fig. 6. Total Workspace Separated by Class Definition and Limiting Thrust

C. Feasibility and Selection of Operating Class

During operation, the control system continuously calculates the required force/moment vector based on the desired vessel state (pose, velocity, etc.). Once the required forces and moments are determined, the system checks whether these demands can be met by any of the predefined operating classes.

The *feasibility check* involves ensuring that the calculated forces and moments lie within the achievable region of the force/moment workspace. If the required forces/moments exceed the physical limits, the system scales down the commands proportionally to ensure that they fit within the workspace.

Once multiple feasible operating classes are identified, the control system uses a *cost function* to select the most efficient class. The cost function evaluates factors such as:

The cost function evaluates the following factors:

- **Avoiding Thrust Reversals**: Minimizing the transitions between positive and negative thrust, which helps to reduce energy loss and wear on the actuators.
- **Minimizing Azimuth Change**: Reducing the amount of steering angle change required, decreasing mechanical wear and actuator strain.

The operating class that minimizes the cost function while ensuring feasibility is selected for implementation.

VI. Closed Form Solutions for Operating Classes

The operating classes for Minion's propulsion system can be categorized into two primary groups: those that operate within the force/moment workspace and those that provide additional functionality such as handling failures or enabling specialized maneuvering. These groups are:

- **Workspace Classes**: The five main classes as discussed in Section V-B cover the fundamental operating conditions and are designed to ensure optimal control of the vessel within the defined force/moment workspace. They manage the propulsion system's performance for both symmetric and asymmetric thrust configurations, enabling precise and efficient control in various operational modes.
- **Specialized and Fault Mitigation Classes**: These additional classes are designed for specific scenarios, such as handling actuator failures or providing enhanced maneuvering capabilities. When one or more actuators fail, these classes allow the system to continue operating with reduced functionality. They may also be used to improve efficiency or reduce mechanical strain during specific maneuvers, such as tight yaw control or minimal-thrust operations.

The following subsections describe the closed-form solutions for each operating class within these groups.

A. Primary Classes

The primary classes provide the core functionality for controlling the vessel under normal conditions. These classes are used for general maneuvering and are applicable to a wide range of tasks.

***Class 1: Port Steering Angle** = $0°$:* In this class, the port motor's steering angle is fixed at $0°$, and the starboard motor provides the necessary steering and thrust. The system of equations is:

$$\begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} = \begin{bmatrix} T_p + T_s\cos(\phi_s) \\ T_s\sin(\phi_s) \\ -YT_p + YT_s\cos(\phi_s) \end{bmatrix} \tag{3}$$

Solving for T_s, T_p and ϕ_s:

$$\begin{aligned} T_p &= \left(F_x - \left(M_z + F_y X\right)/Y\right)/2 \\ |T_s| &= \sqrt{\left(F_y\right)^2 + \left(\left(F_x + \left(M_z + F_y X\right)/Y\right)/2\right)^2} \\ \phi_s &= \tan^{-1}\left(\frac{2F_y}{\left(F_x + \left(M_z + F_y X\right)/Y\right)}\right) \end{aligned} \tag{4}$$

***Class 2: Starboard Steering Angle** = 0°:* In this class, the starboard motor is fixed at 0°, and the port motor handles the steering. The system of equations is:

$$\begin{bmatrix} F_x \\ F_y \\ \left(M_z + F_y X\right)/Y \end{bmatrix} = \begin{bmatrix} T_p\cos\left(\phi_p\right) + T_s \\ T_p\sin\left(\phi_p\right) \\ -T_p\cos\left(\phi_p\right) + T_s \end{bmatrix} \tag{5}$$

Solving for T_s, T_p and ϕ_p:

$$\begin{aligned} T_s &= \left(F_x + \left(M_z + F_y X\right)/Y\right)/2 \\ |T_p| &= \sqrt{\left(F_y\right)^2 + \left(\left(F_x - \left(M_z + F_y X\right)/Y\right)/2\right)^2} \\ \phi_p &= \tan^{-1}\left(\frac{2F_y}{\left(F_x - \left(M_z + F_y X\right)/Y\right)}\right) \end{aligned} \tag{6}$$

***Class 3: Equal Thrust on Both Motors**:* In this class, both motors produce equal thrust, but their steering angles can vary. The system of equations is:

$$\begin{bmatrix} F_x \\ F_y \\ \left(M_z + F_y X\right)/Y \end{bmatrix} = \begin{bmatrix} T_{ps}\cos\left(\phi_p\right) + T_{ps}\cos\left(\phi_s\right) \\ T_{ps}\sin\left(\phi_p\right) + T_{ps}\sin\left(\phi_s\right) \\ -T_{ps}\cos\left(\phi_p\right) + T_{ps}\cos\left(\phi_s\right) \end{bmatrix} \tag{7}$$

First, solve for the angles:

$$\begin{aligned} \phi_p &= \cos^{-1}\left(\frac{\left(F_x - \left(M_z + F_y X\right)/Y\right)}{2T_{ps}}\right) \\ \phi_s &= \cos^{-1}\left(\frac{\left(F_x + \left(M_z + F_y X\right)/Y\right)}{2T_{ps}}\right) \end{aligned} \tag{8}$$

Then, solve for T_{ps}:

$$T_{ps} = \pm\sqrt{\frac{\left(c_1^2 + c_2^2 - F_y^2\right)^2 - 4c_1^2c_2^2}{4F_y^2}} \tag{9}$$

where,

$$\begin{aligned} c_1 &= \frac{\left(F_x - \left(M_z + F_y X\right)/Y\right)}{2} \\ c_2 &= \frac{\left(F_x + \left(M_z + F_y X\right)/Y\right)}{2} \end{aligned} \tag{10}$$

***Class 4: Port Thrust = Starboard Thrust** $\times\alpha$:* In this class, the port thrust is related to the starboard thrust by a constant ratio α, i.e., $T_p = \alpha T_s$. The system of equations becomes:

$$\begin{bmatrix} F_x \\ F_y \\ \left(M_z + F_y X\right)/Y \end{bmatrix} = \begin{bmatrix} T_p\cos\left(\phi_p\right) + \alpha T_p\cos\left(\phi_s\right) \\ T_p\sin\left(\phi_p\right) + \alpha T_p\sin\left(\phi_s\right) \\ -T_p\cos\left(\phi_p\right) + \alpha T_p\cos\left(\phi_s\right) \end{bmatrix} \tag{11}$$

Solving for T_p, ϕ_p, and ϕ_s:

$$\begin{aligned} \phi_p &= \cos^{-1}\left(\frac{\left(F_x - \left(M_z + F_y X\right)/Y\right)}{2T_p}\right) \\ \phi_s &= \cos^{-1}\left(\frac{\left(F_x + \left(M_z + F_y X\right)/Y\right)}{2\alpha T_p}\right) \end{aligned} \tag{12}$$

$$\begin{aligned} &\left(\alpha^4 - 2\alpha^2 + 1\right)T_p^4 \\ &+ \left(-2\alpha^2F_y^2 - 2F_y^2 + 2\alpha^2c_1^2 - 2\alpha^2c_2^2 - 2c_1^2 + 2c_2^2\right)T_p^2 \\ &+ \left(F_y^4 + 2F_y^2c_1^2 + 2F_y^2c_2^2 + c_1^4 - 2c_1^2c_2^2 + c_2^4\right) = 0 \end{aligned} \tag{13}$$

where,

$$\begin{aligned} c_1 &= \frac{\left(F_x - \left(M_z + F_y X\right)/Y\right)}{2} \\ c_2 &= \frac{\left(F_x + \left(M_z + F_y X\right)/Y\right)}{2} \end{aligned} \tag{14}$$

The roots of a 4^{th} order polynomial without odd terms can be found using a modified version of the quadratic equation.

$$\begin{aligned} AT_p^4 + BT_p^2 + C &= 0 \\ T_p &= \pm\sqrt{\frac{-B \pm \sqrt{B^2 - 4AC}}{2A}} \end{aligned} \tag{15}$$

***Class 5: Starboard Thrust = Port Thrust** $\times\alpha$:* Class 5 is the reverse of Class 4, where $T_s = \alpha T_p$. The system of equations becomes:

$$\begin{bmatrix} F_x \\ F_y \\ \left(M_z + F_y X\right)/Y \end{bmatrix} = \begin{bmatrix} \alpha T_s\cos\left(\phi_p\right) + T_s\cos\left(\phi_s\right) \\ \alpha T_s\sin\left(\phi_p\right) + T_s\sin\left(\phi_s\right) \\ -\alpha T_s\cos\left(\phi_p\right) + T_s\cos\left(\phi_s\right) \end{bmatrix} \tag{16}$$

Solving for T_p, ϕ_p, and ϕ_s:

$$\begin{aligned} \phi_p &= \cos^{-1}\left(\frac{\left(F_x - \left(M_z + F_y X\right)/Y\right)}{2\alpha T_s}\right) \\ \phi_s &= \cos^{-1}\left(\frac{\left(F_x + \left(M_z + F_y X\right)/Y\right)}{2T_s}\right) \end{aligned} \tag{17}$$

$$\begin{aligned} &\left(\alpha^4 - 2\alpha^2 + 1\right)T_s^4 \\ &+ \left(-2\alpha^2F_y^2 - 2F_y^2 - 2\alpha^2c_1^2 + 2\alpha^2c_2^2 + 2c_1^2 - 2c_2^2\right)T_s^2 \\ &+ \left(F_y^4 + 2F_y^2c_1^2 + 2F_y^2c_2^2 + c_1^4 - 2c_1^2c_2^2 + c_2^4\right) = 0 \end{aligned} \tag{18}$$

where,

$$c_1 = \frac{(F_x - (M_z + F_y X)/Y)}{2}$$
$$c_2 = \frac{(F_x + (M_z + F_y X)/Y)}{2} \tag{19}$$

The roots are found in a similar way as Class 4,

$$AT_s^4 + BT_s^2 + C = 0$$
$$T_s = \pm\sqrt{\frac{-B \pm \sqrt{B^2 - 4AC}}{2A}} \tag{20}$$

B. Efficiency and Specialized Maneuvering Classes

These classes are used for specific types of motion where the control system prioritizes efficiency, energy saving, or fine-tuned steering for precise maneuvers. They are often used in scenarios like straight-line travel or yaw-focused adjustments.

***Class 6: Parallel Steering (Port = Starboard Steering)**:* In this class, the port and starboard steering angles are equal, i.e., $\phi_p = \phi_s$. The system simplifies as:

$$\begin{bmatrix} F_x \\ F_y \\ (M_z + F_y X)/Y \end{bmatrix} = \begin{bmatrix} T_p \cos(\phi_{ps}) + T_s \cos(\phi_{ps}) \\ T_p \sin(\phi_{ps}) + T_s \sin(\phi_{ps}) \\ -T_p \cos(\phi_{ps}) + T_s \cos(\phi_{ps}) \end{bmatrix} \tag{21}$$

$$= \begin{bmatrix} (T_p + T_s)\cos(\phi_{ps}) \\ (T_p + T_s)\sin(\phi_{ps}) \\ (-T_p + T_s)\cos(\phi_{ps}) \end{bmatrix} \tag{22}$$

Solving for $\phi_p and \phi_s$:

$$\phi_{ps} = \tan^{-1}\frac{F_y}{F_x} \tag{23}$$

$$\begin{aligned} T_s &= (F_x + (M_z + F_y X)/Y) / (2\cos(\phi_{ps})) \\ T_p &= (F_x - (M_z + F_y X)/Y) / (2\cos(\phi_{ps})) \\ \phi_p &= \phi_{ps} \\ \phi_s &= \phi_{ps} \end{aligned} \tag{24}$$

***Class 7: Counter Steering (Port = $-$Starboard Steering)**:* In this class, the port and starboard motors are steered in opposite directions, i.e., $\phi_p = -\phi_s$. The system becomes:

$$\begin{bmatrix} F_x \\ F_y \\ (M_z + F_y X)/Y \end{bmatrix} = \begin{bmatrix} T_p \cos(-\phi_{cs}) + T_s \cos(\phi_{cs}) \\ T_p \sin(-\phi_{cs}) + T_s \sin(\phi_{cs}) \\ -T_p \cos(-\phi_{cs}) + T_s \cos(\phi_{cs}) \end{bmatrix} \tag{25}$$

$$= \begin{bmatrix} (T_p + T_s)\cos(\phi_{cs}) \\ (-T_p + T_s)\sin(\phi_{cs}) \\ (-T_p + T_s)\cos(\phi_{cs}) \end{bmatrix} \tag{26}$$

Solve for the steering angles:

$$\phi_{cs} = \tan^{-1}\frac{F_y}{(M_z + F_y X)/Y} \tag{27}$$

$$\begin{aligned} \phi_p &= -\phi_{cs} \\ \phi_s &= \phi_{cs} \end{aligned} \tag{28}$$

And calculate the thrusts:

$$\begin{aligned} T_s &= (F_x + (M_z + F_y X)/Y) / (2\cos(\phi_{cs})) \\ T_p &= (F_x - (M_z + F_y X)/Y) / (2\cos(\phi_{cs})) \end{aligned} \tag{29}$$

***Class 8: Counter Thrust (Port Thrust = $-$Starboard Thrust)**:* In this class, the port and starboard motors generate equal but opposite thrusts, i.e., $T_p = -T_s$. This class is commonly used for pure yaw motion. The system becomes:

$$\begin{bmatrix} F_x \\ F_y \\ (M_z + F_y X)/Y \end{bmatrix} = \begin{bmatrix} -T_s \cos(-\phi_p) + T_s \cos(\phi_s) \\ -T_s \sin(-\phi_p) + T_s \sin(\phi_s) \\ T_s \cos(-\phi_p) + T_s \cos(\phi_s) \end{bmatrix} \tag{30}$$

The steering angles are given by:

$$\begin{aligned} \phi_p &= -\cos^{-1}\left(\frac{(F_x - (M_z + F_y X)/Y)}{-2T_s}\right) \\ \phi_s &= \cos^{-1}\left(\frac{(F_x + (M_z + F_y X)/Y)}{2T_s}\right) \end{aligned} \tag{31}$$

The thrust is given by:

$$T = \pm\sqrt{\frac{\left(c_1^2 + c_2^2 - F_y^2\right)^2 - 4c_1^2 c_2^2}{4F_y^2}} \tag{32}$$

where,

$$c_1 = \frac{(F_x - (M_z + F_y X)/Y)}{2}$$
$$c_2 = \frac{(F_x + (M_z + F_y X)/Y)}{2} \tag{33}$$

C. Fault Mitigation Classes

These classes are used when there is a failure in one of the propulsion pods. They ensure that the vessel can continue operating, albeit with reduced capabilities, using the remaining functional pod.

Class 9: Port Thrust = 0 (Starboard Only)*:* In the event of a failure on the port side, this class allows the starboard motor to generate all thrust. The system becomes:

$$\begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} = \begin{bmatrix} T_s \cos(\phi_s) \\ T_s \sin(\phi_s) \\ -T_s \sin(\phi_s) X + T_s \cos(\phi_s) Y \end{bmatrix} \tag{34}$$

$$\begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} = \begin{bmatrix} T_s \cos(\phi_s) \\ T_s \sin(\phi_s) \\ -F_y X + F_x Y \end{bmatrix}$$

The steering angle ϕ_s is:

$$\begin{bmatrix} \tilde{F}_x \\ \tilde{F}_y \\ \tilde{M}_z \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} - k\hat{n} \tag{35}$$

$$\phi_s = \tan^{-1}\left(\frac{\tilde{F}_y}{\tilde{F}_x}\right) \tag{36}$$

The thrust T_s is:

$$T_s = \tilde{F}_x / \cos(\phi_s) \tag{37}$$

Class 10: Starboard Thrust = 0 (Port Only)*:* Similarly, if the starboard motor fails, the port motor generates all thrust. The system is:

$$\begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} = \begin{bmatrix} T_p \cos(\phi_p) \\ T_p \sin(\phi_p) \\ -T_p \sin(\phi_p) X - T_p \cos(\phi_p) Y \end{bmatrix} \tag{38}$$

$$\begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} = \begin{bmatrix} T_p \cos(\phi_p) \\ T_p \sin(\phi_p) \\ -F_y X - F_x Y \end{bmatrix}$$

The steering angle ϕ_p is:

$$\begin{bmatrix} \tilde{F}_x \\ \tilde{F}_y \\ \tilde{M}_z \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} - k\hat{n} \tag{39}$$

$$\phi_s = \tan^{-1}\left(\frac{\tilde{F}_y}{\tilde{F}_x}\right) \tag{40}$$

The thrust T_p is:

$$T_s = \tilde{F}_x / \cos(\phi_s) \tag{41}$$

VII. Conclusion

In preparing for the 2024 Maritime RobotX competition, Minion has undergone significant control system refinements to enhance its autonomy, agility, and overall efficiency. This appendix has presented an in-depth look at the innovations implemented, including full azimuth control with independent motor steering and an over-actuated system that allows for precise maneuverability in all three degrees of freedom. By leveraging advanced control strategies such as inverse kinematics and feedback control, Minion can dynamically adjust its movement to meet mission demands.

The integration of the azimuth control via Volz DA-36 servo [10] with higher torque and salt-water resistance ensures that the hardware setup is optimized for both performance and reliability in harsh marine environments.

Overall, these advancements in both software and hardware represent a step forward in autonomous vessel control, showcasing Minion's ability to handle complex maritime tasks with precision and minimal human intervention. This work positions Minion to continue pushing the boundaries of autonomous marine technology for the competition.

References

[1] US Patent No. US8777681B1, "Systems and methods for maneuvering a marine vessel," Brunswick Corporation, 2014.

[2] US Patent No. US9132903B1, "Systems and methods for laterally maneuvering marine vessels," Brunswick Corporation, 2015.

[3] US Patent No. US7467595B1, "Joystick method for maneuvering a marine vessel with two or more sterndrive units," Brunswick Corporation, 2008.

[4] US Patent No. US10671073B2, "Station keeping system and method," Brunswick Corporation, 2020.

[5] US Patent No. US20090171520A1, "Marine vessel running controlling apparatus and marine vessel including the same," Yamaha Hatsudoki Kabushiki Kaisha, 2009.

[6] US Patent No. US20150089427A1, "Vessel display system and small vessel including the same," Yamaha Hatsudoki Kabushiki Kaisha, 2015.

[7] Team Minion's Journal for RobotX Maritime Challenge, 2016.

[8] Team Minion's Journal for RobotX Maritime Challenge, 2018.

[9] Team Minion's Journal for RobotX Maritime Challenge, 2022.

[10] Volz Servos GmbH & Co. KG, *DA 36 Low Profile Technical Specification*, Revision D, 2023. Available: https://www.volz-servos.com

Appendix C: Minion Racquetball Launcher

Dutch Holland, Travis Kerridge, Erik Liebergall, James Palmer
holland9@my.erau.edu, kerridgt@my.erau.edu , lieberge@my.erau.edu, palmej26@my.erau.edu
Embry-Riddle Aeronautical University
Daytona Beach, Florida

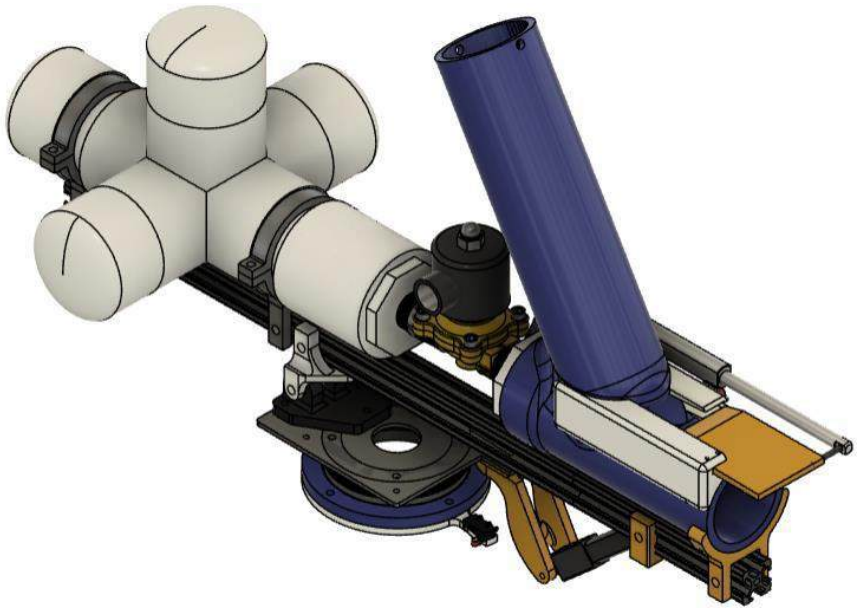


Figure 1: Minion Launcher CAD Model

I. Introduction

Minion Launcher is designed to complete *Task 6: Dock and Deliver* [1, p. 21] for Embry-Riddle's RobotX Team Minion. This system was developed by a group of graduate mechanical engineering students at Embry-Riddle Aeronautical University for an Advanced Mechatronics class, EE 505. This course is project based, meaning the students form teams during the first week of class then each team works on a project throughout the semester. The teams select their own project ideas and work together to complete their project and conduct a final demonstration at the end of the semester. Dutch Holland, Travis Kerridge, Erik Liebergall, and James Palmer all took this Advanced Mechatronics class and were on the same team. For their class project, they chose to help Team Minion by creating a ball launching system to use during the 2024 Maritime RobotX Competition in Sarasota, Florida. This course took place during the Spring of 2024.

To determine a feasible racquetball launching method, a decision matrix was created to compare spring punching (spring powered punch-rod), crossbow, pneumatic launching, pneumatic punching, and roller launching mechanisms. The decision matrix was conducted with a scale of 1-5 with 5 being the least ideal scenario. Although the crossbow mechanism resulted in the lowest overall score, the pneumatic launcher was selected due to its rate of fire, ease of build, and less actuation required.

	Minion Launcher Decision Matrix				
	Spring Puncher	Crossbow	Pneumatic Launcher	Pneumatic Puncher	Roller Launcher
Rate of Fire	5	2	1	1	1
Ease of Build	4	4	3	2	3
Power Consumption	4	3	3	3	4
Computational Expense	2	3	2	2	3
Number of Motors	2	4	3	3	5
Minion Integration Feasibility	3	1	2	2	3
Weight	2	2	4	5	5
Size	1	4	3	3	3
Testability	3	1	5	4	1
Totals	26	24	26	25	28

Table 1: Minion Launcher decision matrix

II. General Solution

To help Team Minion achieve Task 6, the Minion Launcher team developed a pneumatic ball launching system that can aim at the target and fire four consecutive shots without manual intervention. The system is comprised of a primary pressure tank, secondary pressure tank, barrel, reloader, solenoid valves, stepper motor, linear actuator, a Raspberry Pi 4B, Arduino MEGA 2560, LED panel, and emergency stop button as shown in Figure 2. The Raspberry Pi 4B will listen to ROS messages from Minion that contain the target location and distance. When given the ROS command from Minion, the Raspberry Pi will send serial USB commands to the Arduino, which will control the launchers angle and firing sequence. During the firing sequence, the primary tank will pressurize the secondary tank, then the secondary tank pressure is released into the barrel, launching the ball. After launching, the system will reload and repeat the sequence 3 more times. Team Minion will attempt to make all four racquetballs in the smaller of the designated targets, though the team has the option to shoot at the larger target instead.

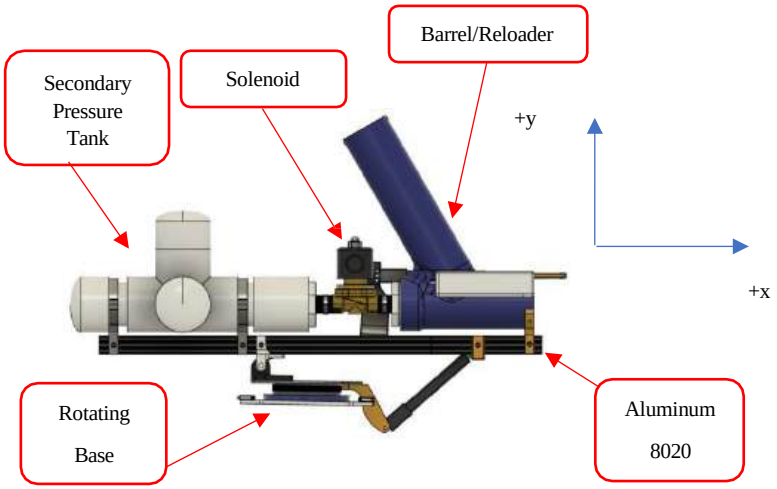


Figure 2: Side view of the Minion Launcher with associated reference frame.

III. Pressure, Barrel Length, and Range Estimation

Since this involves kinematic projectiles, the projectile's motion given various initial conditions was modeled. A MATLAB script was created to run through multiple combinations of barrel lengths and initial pressures that the projectile could experience. The barrel lengths considered were 1 to 8 inches with 1-inch increments, and the pressures were 20 to 80 psi in increments of 20. The kinematic relationship used in this MATLAB script is the set of standard projectile motion equations shown below:

Eq. 1:

$$y = y_o + v_{oy} * t - \frac{1}{2} * 32.2 \frac{ft}{s^2} * t^2$$

Eq. 2:

$$x = x_o + v_{ox} * t$$

In these equations, x and y are the final height and horizontal position of the projectile, x_o and y_o are the initial height and horizontal position of the projectile, v_{ox} and v_{oy} are the initial velocity components of the projectile once it leaves the barrel, t is the time, and a is the acceleration of gravity. To calculate the initial velocity of the projectile as it left the barrel, an equation that relates the pressure, barrel length, and barrel's cross-sectional area was used [2]. This equation is shown below:

Eq. 3:

$$v_o = \sqrt{\frac{2 * P * A * L}{m}}$$

For this calculation, P is the initial pressure of the compressed air tank, A is the cross-sectional area of the barrel, L is the length of the barrel, and m is the mass of the projectile.

When calculating the motion of a projectile, ball drop must be considered when aiming. Ball drop is the difference between the ideal vertical position that the projectile should hit and the actual vertical position that the projectile hits. If a laser is used to indicate the aiming point of the targeting system, the projectile will always hit lower than that point due to gravity acting on the ball and causing a parabolic trajectory, this is affectively ball drop. Ball drop will increase as distance from the target increases due to the longer amount of time that gravity will act on the ball. To account for this, the initial pressure can be increased, which according to Equation 3, will increase the initial velocity, effectively creating a straighter projectile path between the barrel and the target. To calculate the ball drop, a simple difference equation is used. To ensure better targeting, the top edge of the target is considered the ideal vertical position, and the bottom edge of the target is the actual vertical position. The ball drop equation is shown below:

Eq. 4:

$$Ball_{Drop} = y_{ideal} - y_{actual}$$

For projectile motion, the initial angle that the projectile is launched is extremely important to determining the x and y components of the initial velocity, which are needed according to Equation 1. Simply using inverse trigonometric identities is not enough to determine the appropriate initial angle due to the expected drop of the projectile during its flight. To help increase accuracy and decrease ball drop, an adaptation must be made to the angle calculation to account for the increase in ball drop as the distance from the target increases. This is shown in the following equation:

Eq. 5:

$$\theta = \tan^{-1}\left(\frac{y + 2 * Ball\_Drop}{x}\right)$$

The expected inverse tangent equation is used when the y and x locations of the target box are known, but an adaptation to shoot higher than the top of the target based on the allowed ball drop is used for the height of the target location. The angle, θ, can be converted to degrees by multiplying by $\frac{180}{\pi}$. The equation will be changed later to also consider the pressure and barrel length used as those variables will change the initial velocity of the projectile.

To visualize the data, the projectile motion was graphed for the different barrel lengths at the various pressure levels. A conservative estimation of the target location was made with the target height being 2 feet above the exit point of the barrel and 20 feet from the end of the barrel in the lateral direction. The graphs are shown below with the top of the target being indicated by the top black circle, and the bottom black circle indicating the allowed ball drop from the top of the target. The projectile motions that go between those two black circles are considered acceptable combination values for the barrel length and pressure.

Figure 3: Ball Trajectories for 20 PSI with black circles indicating upper and lower target limits.

From the graph in Figure 3, all trials of barrel lengths for a pressure of 20 psi do not fall in the acceptable range between the black circles. This means, that 20 psi is expected to be an insufficient pressure for launching the projectile.

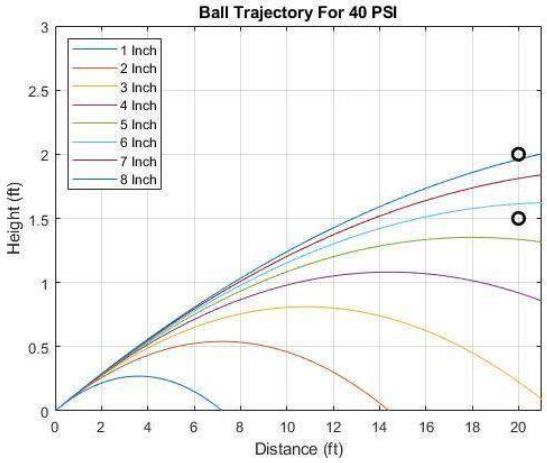


Figure 4: Ball Trajectories for 40 PSI with black circles indicating upper and lower target limits

For a pressure of 40 psi, the barrel lengths of 6, 7, and 8 inches provide a sufficient initial velocity to make the projectile reach the desired locations. This is considered the minimum pressure needed for the launcher.

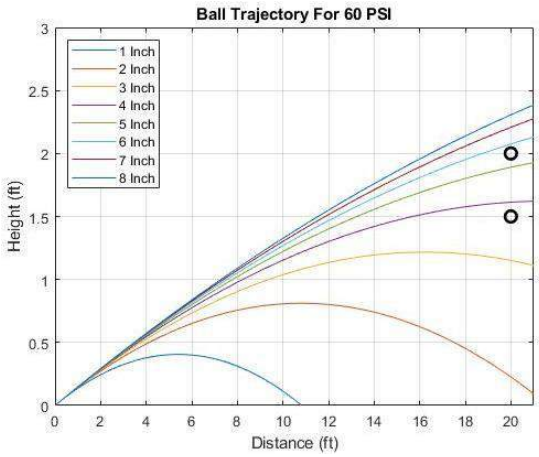


Figure 5: Ball Trajectories for 60 PSI with black circles indicating upper and lower target limits

As seen above, the pressure of 60 psi allows for the projectile to reach the desired location with a barrel length of at least 4 inches. The 6, 7, and 8-inch barrels are overshooting the target, but this error is due to the angle equation needing to be lowered for this combination of pressure and barrel length.

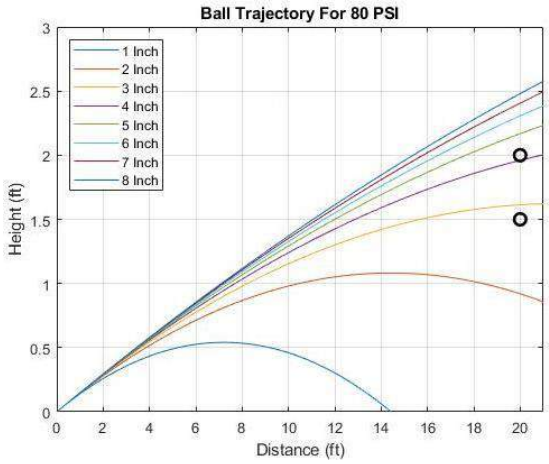


Figure 6: Ball Trajectories for 80 PSI with black circles indicating upper and lower target limits

For 80 psi, half of the barrel lengths result in an overshoot of the target which can be accounted for in the angle equation and the minimum barrel length becomes 3 inches.

After analyzing the initial estimates, a test was conducted with variable barrel lengths and pressures to confirm which combination would provide the most repeatable results. The tested pressures ranged from 30-45 PSI and the barrel lengths ranged from 4-6 inches. The test results, shown in Table 2, indicated that 45 PSI resulted in the least ball drop, as expected.

Table 2: Minion Launcher pressure, barrel, and ball drop test results

Test	Barrel Length (in)	Pressure (PSI)	Measured Impact Height (in.)	Actual Impact Height (in.)	Ball Drop (in.)
1	4	26.5	51	52.000	12.750
2	4	27	48.5	49.500	15.250
3	4	27.5	51	52.000	12.750
4	4	28	51.5	52.500	12.250
5	4	25	47	48.000	16.750
6	4	33	56	57.000	7.750
7	4	31.5	48.5	49.500	15.250
8	4	30	53	54.000	10.750
9	4	33	56.5	57.500	7.250
10	4	32.5	56	57.000	7.750
11	4	38	61	62.000	2.750
12	4	36.5	59	60.000	4.750
13	4	35	60	61.000	3.750
14	4	35	58	59.000	5.750
15	4	35	57	58.000	6.750
16	4	41.5	61	62.000	2.750
17	4	43	62	63.000	1.750
18	4	43	61.5	62.500	2.250
19	4	43	61.5	62.500	2.250
20	4	43	62	63.000	1.750
21	4	45	62.5	63.500	1.250
22	4	47.5	63.5	64.500	0.250
23	4	45	63	64.000	0.750
24	4	47	61.5	62.500	2.250
25	4	47	63.25	64.250	0.500
26	5	25	48	49.000	15.750
27	5	28.5	54	55.000	9.750
28	5	25	47.5	48.500	16.250
29	5	25	48.5	49.500	15.250
30	5	28	50.5	51.500	13.250
31	5	30	54	55.000	9.750
32	5	32.5	56	57.000	7.750
33	5	30	53.5	54.500	10.250
34	5	33	55.5	56.500	8.250
35	5	32	55	56.000	8.750
36	5	37	56.5	57.500	7.250
37	5	36.5	58	59.000	5.750
38	5	37	59	60.000	4.750
39	5	38	59.5	60.500	4.250
40	5	34	57	58.000	6.750
41	5	41	60.5	61.500	3.250
42	5	40.5	61	62.000	2.750
43	5	40	59	60.000	4.750

44	5	41	60	61.000	3.750	60	6	33	58.5	59.500	5.250
45	5	41	61	62.000	2.750	61	6	35	58	59.000	5.750
46	5	45	63.5	64.500	0.250	62	6	35	58	59.000	5.750
47	5	45	63	64.000	0.750	63	6	35	59.75	60.750	4.000
48	5	45	62.5	63.500	1.250	64	6	37.5	60	61.000	3.750
49	5	45	62	63.000	1.750	65	6	35	57.75	58.750	6.000
50	5	44.5	63	64.000	0.750	66	6	40	60.5	61.500	3.250
51	6	25	47	48.000	16.750	67	6	40	62	63.000	1.750
52	6	25	42.5	43.500	21.250	68	6	40	62.6	63.600	1.150
53	6	25	48.5	49.500	15.250	69	6	41	61	62.000	2.750
54	6	25	46	47.000	17.750	70	6	40	61	62.000	2.750
55	6	25	43	44.000	20.750	71	6	45	62	63.000	1.750
56	6	32.5	56.5	57.500	7.250	72	6	45	62.25	63.250	1.500
57	6	30	53	54.000	10.750	73	6	45	63.5	64.500	0.250
58	6	31.5	56.5	57.500	7.250	74	6	45	62.25	63.250	1.500
59	6	30	56.5	57.500	7.250	75	6	45	62.5	63.500	1.250

IV. PRIMARY AND SECONDARY TANK DESIGN

Since four projectiles must be fired without re-pressurizing the primary tank, an analysis is used to determine the primary tank required starting pressure, as using the primary tank to fill the secondary tank will reduce the pressure of the primary tank. This requires the primary tank to be greater than or equal to the required launch pressure in the secondary tank to ensure all shots are fired. The first step in determining the total pressure of the system is to calculate the required pressure for one shot. The volume of the secondary tank is 79.3 in^3 and the volume of the primary tank is approximately 68 in^3. As previously noted, the secondary volume needs to be pressurized to 45 PSI to launch a single projectile. The Combined Gas Law, shown below, is then used to determine the required starting pressure of the primary tank to ensure all four shots can be completed without re-pressurization.

Eq. 8:

$$V_S * P_S = V_P * P_P$$

The subscript P indicates the primary tank property and the subscript S indicates the secondary tank property. This equation shows that for 45psi in the secondary tank, the primary tank must be pressurized to 52.48 psi. The total number of shots is 4, so the total primary pressure needed for all four shots needs to be calculated. The equation to calculate the total primary tank pressure is as follows:

Eq. 9:

$$P_{total} = (Shots + 1) * Primary\_Pressure$$

The reason the equation multiples the primary pressure for one shot by $(Shots + 1)$ is because the primary pressure calculated for 1 shot is really the pressure

required to keep a 45-psi pressure in the secondary tank. Since the secondary tank is initially empty, pressure equalization must be considered which means there will be pressure remaining in the primary tank that won't be released during the 4-shot firing sequence. To fire the number of shots desired, an additional shot must be added to the desired shot count to allow for pressure equalization on the final shot which results in the equation having $(Shots + 1)$. The total primary tank pressure that is needed based on the parameters provided is 262.39 psi.

A paintball air tank was selected for the primary tank because this tank can withstand up to 4500 PSI. This tank's rated pressure is over 17 times greater than the total pressure needed to fire all four projectiles, indicating a high safety factor. The secondary tank is comprised of Schedule 40 PVC piping, PVC fittings, and PVC grade bonding agents. Schedule 40 PVC was chosen because it is rated to 280 PSI, which is more than 6 times the required launch pressure of 45 PSI. Additionally, a one-way pressure valve in conjunction with a digital pressure sensor are implemented to ensure the secondary tank will not be over pressurized.

V. Barrel and Reloader

The barrel/reloader was 3D printed with PLA due to its ease of access, affordability, and repeatability. Utilizing Fusion 360 and 3D printers sped up the iterative process of barrel design, printing, testing, and redesigning; ultimately allowing the group to narrow down optimal barrel dimensions within two weeks. The barrel/reloader, seen in Figure 8, features a ball magazine offset 30 degrees from the vertical, and a reloading door offset 4 degrees from the vertical. The reloading door is necessary to close off the magazine while launching the ball, which ensures most of the air will travel behind the ball down the barrel. Early iterations of the reloader introduced ball binding which caused the system to jam (see Figure 7). The offsets shown in Figure 8 were implemented and reduced the ball binding in the reloader. In this configuration, the contact angle between the ball and the door caused the ball to hit the wall behind it. This angle was reduced in the final version shown in Figure 8, reducing the ball binding.

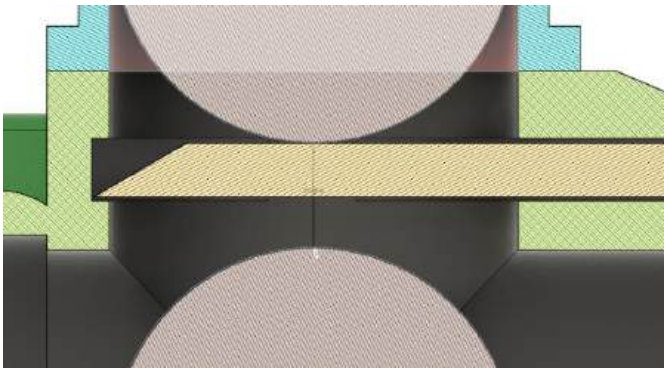


Figure 7: Initial reloading mechanism chamber/barrel interface.

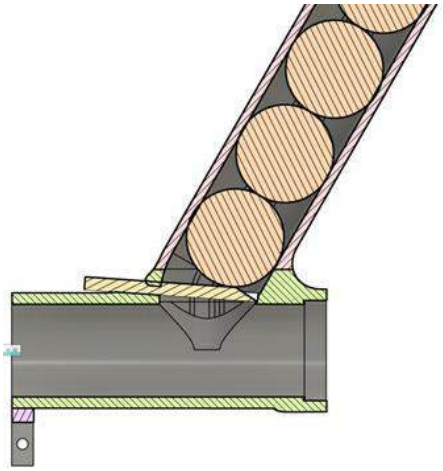


Figure 8: Final reloading mechanism without door guides or linear actuator.

However, the new design was not without flaws. The door did not have enough guidance and would slip out of position when attempting to reload the barrel. So, guides were needed to keep the door aligned with the barrel. With the addition of a thinner, angled, door and

door guides the final design iteration, seen in Figure 9, successfully reloads the ball when commanded.

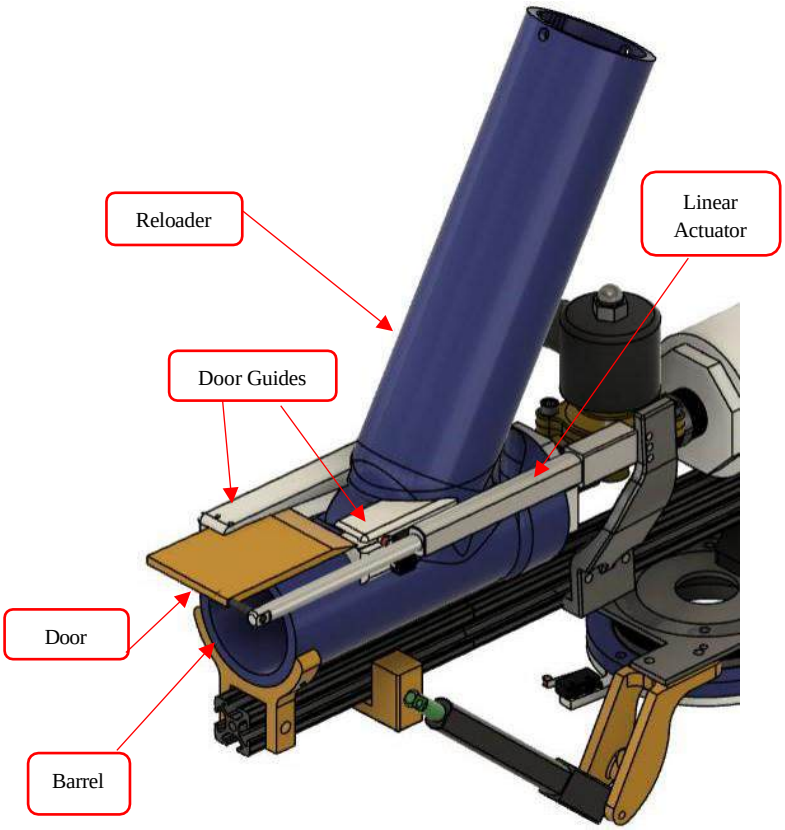


Figure 9: Complete Reloading Mechanism.

VI. Conclusion

The Minion Launcher system proved its capabilities during course demonstration by showing its ability to hit three different targets when given a command without human intervention. This demonstration also served as a proof of concept for integration on Embry-Riddle's RobotX platform, Minion. The launching system will be mounted on the front port corner of Minion's deck using Aluminum 8020 rails. For the 2024 competition, Minion will dock in the required docking bay and hold its position while Minion Launcher attempts to hit the smaller target.

References

[1] *RobotX 2024 Team Handbook*, v2.0. RoboNation. (2024). Accessed: October 1, 2024. [Online]. Available: https://robonation.org/app/uploads/sites/2/2024/09/2024-RobotX_Team-Handbook_v2.0.pdf

[2] Yost, M., Martin, Z., & Odon, L., "The Design and Fabrication of a Compressed Air Cannon," George Washington University, Washington, D.C., 2019. https://scholarspace.library.gwu.edu/downloads/kh04dq41t?locale=it

Appendix D: Vision System

Dan Lane, Dr. Eric Coyle

I. Introduction

In the field of robotics, computer vision is an important tool for perception, but also challenging. Integrating the spatial precision of LiDAR data and the dense pattern information and color discernment of camera data can facilitate a much more accurate interpretation of the system environment. However, to fuse data from both modalities, these sensors must be calibrated to through geometric transforms and have their measurements time synchronized. This ensures that anything in view of either sensor can be easily cross-referenced by the other for additional information.

For the 2024 RobotX Challenge, four of the eight available tasks require a team's unmanned surface vehicle (USV) to make decisions based on an object's color. Luckily, one of the major focus areas of Team Minion in 2024 has been to improve the spatial and temporal accuracy of camera data within the vision system. First, this process entailed precise intrinsic and extrinsic calibrations to reduce the geometric error between the LiDAR and camera reference frames. Second, custom software was written to stream the video data to overcome network latency issues and achieve the levels of temporal accuracy required. This software encodes the system time that each frame is captured into the video data, which is then streamed using the real-time protocol (RTP) from the camera enclosure to Minion's main CPU. On Minion, the video stream is decoded and the image frame and embedded timestamp are published as a Robotic Operating System (ROS) message. These enhancements enable the mapping of information between camera and LiDAR frames at frame rates over 15fps, allowing for the swift addition of color labels to detected objects. This improvement enhances Team Minion's ability to plan tasks, make informed decisions, and achieve success at this year's competition and beyond.

II. Process

For this year's competition, Minion's vision system utilizes four high-definition cameras housed within an enclosure with three LiDAR scanners mounted on its lid as seen in figure 1. The specific hardware is listed in table I. Three 4K cameras with a resolution of 4000 x 2000 and an HDR camera with a resolution of 2880 x 1860 provide a 160-degree view of the bow of the USV. The Livox LiDAR scanners each have a horizontal FOV of 81.7^o which covers the same area as the cameras and is shown in Figure 2. The LiDAR units are connected directly to the Minion's local area network (LAN) and managed by the manufacturer's firmware. The cameras are connected to an NVIDIA Jetson which is responsible for encoding and streaming video to Minion's network and is synchronized to the internal clock on Minion's main computer. This system is designed to be modular and largely self-contained as it is used for research purposes. The port and starboard Livox Horizon LiDAR devices are calibrated to return data in the reference frame of the forward-facing LiDAR, so that if an individual component gets replaced or updated, Minion only needs the calibrated transform to be updated for the system to be functional.

Fig. 1. 3D render of Minion's Camera Enclosure with labeled camera and LiDAR hardware

TABLE I
Vision System Hardware

Device	Make	Model	Qty.
CPU	NVIDIA	Jetson AGX Xavier	1
Lidar	Livox	Horizon	3
4K Camera	FLIR	Blackfly S 120S4C	3
HDR Camera	Leopard Imaging /Sony	LI-USB30-IMX490-GW5400-GMSL2-065H	1

A. Spatial Calibration

Before the data from the camera and LiDAR sensors can be combined, each device needs to be spatially calibrated through extrinsic and intrinsic transforms. This ensures that the scale and position of objects seen by either sensor type can be placed into a common reference frame. Additionally, the data received from each device should agree upon what moment in time the data represents. Prior to this year's competition, latency in the video signal was compensated for by adjusting its timestamp with a constant offset, and while this method was sufficient when the USV was stationary, it created significant errors during rapid motion, especially turning.

1) Camera Intrinsics: Camera intrinsics refer to a camera's unique properties which define the path of light through its optical path. These properties can functionally define the path of any ray of light through the camera lens to the exact location on the camera sensor and provide a geometric translation

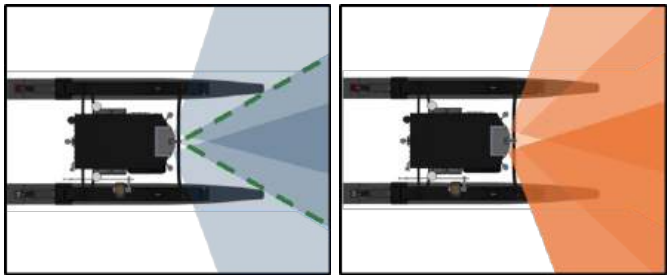


Fig. 2. FOV for camera (left) and LiDAR (right) sensors. 4K FOV is represented by overlapping shaded areas and the HDR camera FOV is shown with dashed lines. Center LiDAR overlaps with HDR and center 4K cameras, as well as a large portion of the port and starboard 4K cameras.

from a position in the 3-dimensional camera frame to a 2-dimensional pixel location in the image frame. It is generally represented as a 3×3 matrix as:

$$\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where (f_x, f_y) is the focal length of the camera lens in pixels, (c_x, c_y) is the (x, y) position (measured in pixels) in the image frame that is centered in the optical path, known as the principal point, and skew s is the angle between the vertical and horizontal axis of the image frame that is generally assumed to be $s = 0$. Radial distortion is the spherical abortion caused by the camera lens and is represented as either one, two, or three constants of a polynomial k_1, k_2, k_3, where:

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$r^2 = x^2 + y^2$$

While these values are generally published as part of the camera specifications, calibration is still required to account for manufacturing tolerances. A common method to perform this calibration is through homography, and involves capturing several images of a checkerboard pattern in multiple positions within the image frame [1]. MATLAB's Camera Calibration application uses an algorithm based on this technique which identifies intersecting points made by the checkerboard squares and the measured distance in pixels. This information is cross-referenced with the physical dimensions of the checkerboard [2]. This tool estimates the camera's intrinsic properties by minimizing the error between the observed checkerboard patterns and projected points through a pin-hole camera model [3]. However, the accuracy of this method is dependent on the quality of the calibration data used. Minion's calibration dataset consisted of 23 instances of simultaneous lidar scans and camera recordings with the checkerboard placed at various locations and distances within each camera's view. While only only data from the camera sensors are required to calibrate intrinsics, the LiDAR data is captured simultaneously for extrinsic calibration. A composite of several images taken during Minion's camera calibration is provided in figure 3.

2) Extrinsic Transform: To identify object positions in a different reference frame than it is measured requires an

Fig. 3. A composite image of multiple calibration images containing checkerboards.

extrinsic transform between the frames. This transform defines the translation and rotation (yaw, pitch, roll, x, y, z) to make one frame match another and is represented by quaternions or Euler angles. For Minion, an extrinsic transform is used between the port and starboard facing Livox devices to the central forward facing Livox to create a unified point cloud. Each camera also receives an extrinsic transform to the Livox reference frame. A flowchart of these transforms is provided in Figure 4.

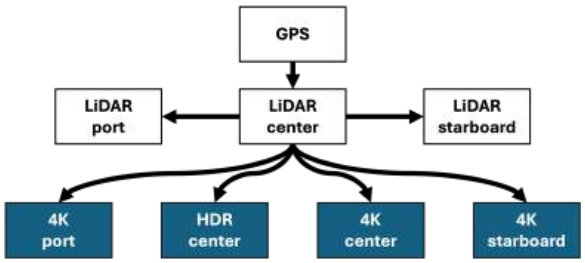


Fig. 4. A flowchart of transforms between reference frames. Arrows indicate extrinsic transformations, and dark boxes indicate an additional intrinsic transform.

LiDAR-to-LiDAR calibration is performed with the Livox Viewer software that ships with the sensors. This software identifies and co-locates features in the overlapping scanning patterns to obtain the translation and rotation between the two sensors. For camera-to-LiDAR calibration, the checkerboard calibration dataset is reused from the camera intrinsic calibration. Three seconds of LiDAR scan data are isolated for each position of the checkerboard pattern to obtain a more dense point cloud and then loaded into the Lidar Calibration application within MATLAB. This function co-locates the three-dimensional checkerboard pattern from the image frame using the identified camera intrinsics and the point cloud data and then estimates the transform between them [4]. This initial calibration is shown in Figure 5 by projecting the outermost checkerboard corners detected in the images (red) into the LiDAR point cloud data.

The results from both intrinsic and extrinsic calibrations were examined by projecting LiDAR data onto the image

frame. This allowed us to manually refine the transforms by comparing how other spatial features, such as walls, trees and lamp posts, align in the image frame. The final extrinsic transform is provided in Table II.

TABLE II
FINAL TRANSFORMS TO CENTER LIVOX FRAME

Sensor	roll	pitch	yaw	x	y	z
Livox (port)	-4.14	1.47	42.97	-0.057	0.169	0
Livox (stbd)	4.25	1.85	-42.6	-0.057	-0.169	0
4K (port)	-88.35	0.84	-97.16	-0.031	0.155	0.159
HDR (fwd)	103.85	-0.69	88.83	-0.032	0.115	0.098
4K (fwd)	-88.31	0.98	-97.33	-0.023	0.095	0.19
4K (stbd)	-88.35	0.84	-97.16	-0.32	0.15	0.16

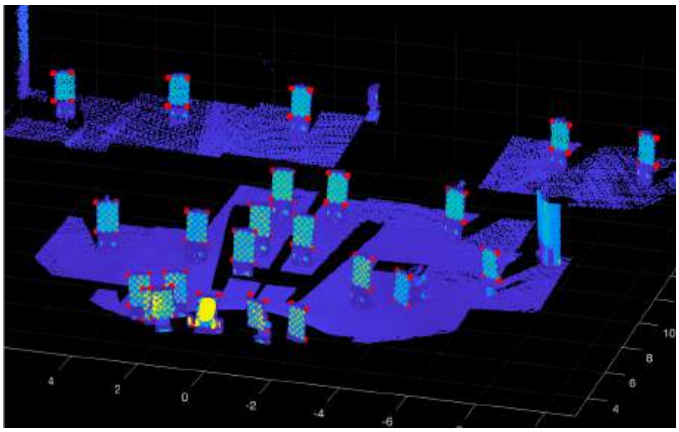


Fig. 5. Result of the LiDAR Calibration. Outer corners of the checkerboard pattern are transformed using Intrinsic and Extrinsic transforms and projected (red) onto LiDAR point cloud (blue and yellow)of the same ROIs.

B. Temporal Synchronization

While Minion's camera and LiDAR systems have been spatially calibrated for previous competitions, the video signal can be delayed due to inefficient encoding or network latency. If unaddressed, this time lag creates a temporal misalignment between the LiDAR and camera frames while the vessel is in motion.

On Minion, object detection is performed with GB-CACHE [5] using LiDAR data. Before this year's competition, any vision-based object or color detection was performed in parallel to the LiDAR-based perception. Video latency was addressed by adding a constant offset to the video timestamp. While this was sufficient for tasks where the USV was stationary, such as detecting the color code sequence, the uncertainty between the time of actual and estimated image capture limited the system's use while in motion.

To remedy this, a custom script was written to record the system time whenever a new video frame was captured by the camera, and simultaneously encode this timestamp with the image frame as supplemental encoded information (SEI) in the video stream. An additional script was written as a ROS node which decodes the video frame and timestamp and then publishes that information as a ROS message. Figure 6 provides a block diagram of the software and hardware for this synchronization process.

Adopting this solution required that the system clock of the Jetson computer responsible for encoding and transmitting the video feeds from the camera enclosure is in near-perfect synchronization with Minion's onboard PC that hosts the ROS functionality. This is handled in Linux with the system utility chrony, which aligns the system clock in the enclosure with Minion's main PC every time the system powers on. It also monitors the clocks via a network time protocol (NTP) for any drift or error introduced through network traffic.

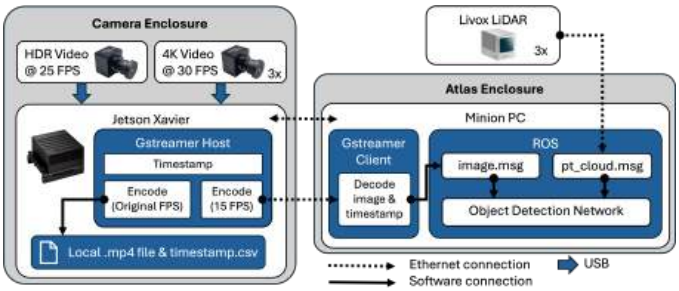


Fig. 6. Block diagram of the vision system's hardware and video pipeline software.

This approach allows for granular customization of the video stream size, frame rate, network bit rate, and compression method. Additionally, because of the structure of ROS, any ROS module that needs to subscribe to the video stream is able to receive the imagery with the correct timestamp. However, to minimize network load, modules only subscribe to this stream when necessary, such as when a specific task is being conducted. While open-source plugins like GSCam [6] and DeepStream [7] exist to incorporate video into ROS, Minion's solution is flexible and better adapted to address the network latency seen with Minion's specific hardware.

C. Object Detection and Color Classification

With an accurate spatial transform and temporal alignment between the LiDAR and video data streams, Minion can use LiDAR to locate objects within the camera frame. Using the extrinsic and intrinsic transforms discussed above, Minion is able to project the bounds of newly detected objects into the image frame to capture a specific region of interest (ROI). This provides a reference image for confirmation of object class and extracting color. However, not all detected objects require a color label, and not all colors are applicable to all objects. Minion's color classification module contains logic to apply specific algorithms based on object class. It should be noted that this algorithmic approach to identifying object color was chosen over a camera-based object detection network like YOLO because the existing LiDAR-based object detection and classification is robust and trusted to classify objects with significantly less training data and time. This only leaves the task of classifying five colors to the vision system, and an algorithmic approach was deemed more reliable and a less resource-intensive approach. The LiDAR approach does this based on the unique geometry and near-infrared reflectivity

for each desired class [8]. These algorithms are provided and discussed below.

Once objects are classified and located using GB-CACHE, a virtual polygon is drawn around the top-down bounds of the object and assumed to be extruded vertically between the top and bottom of the object. When projected onto the 2D image, this creates a number of image frame vertices that can be searched for the minimum and maximum x, y camera frame coordinates. These coordinates form the two-dimensional image ROI. As long as enough of this ROI is within the camera field of view (typically 90%), the ROI is passed onto the color classifier. THis process is diagramed in fig. 7, and example ROI imagery from VRX is provided in fig. 8.

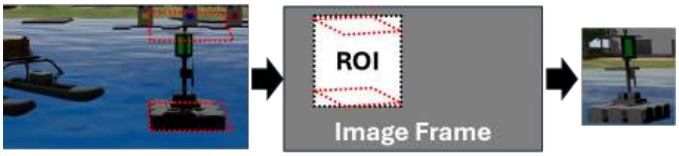


Fig. 7. Flow chart of object detection to ROI image capture process.

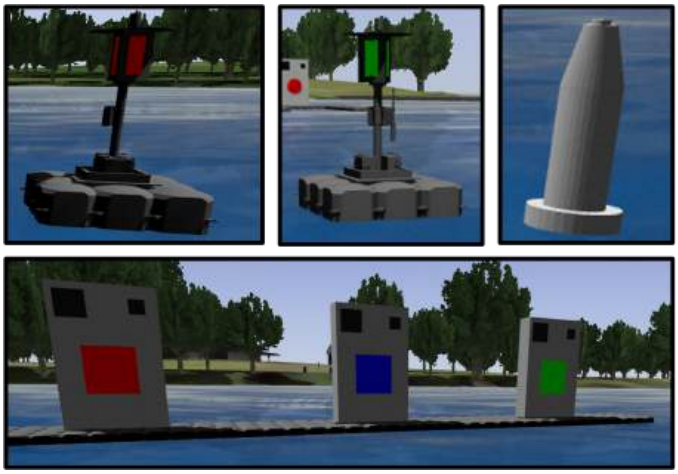


Fig. 8. ROIs from LiDAR-based object detection used to extract relevant objects from video frames.

The color classifier runs one of two algorithms depending on the class of the detected object. One algorithm is used for most objects and is robust under most conditions. This algorithm has been specialized further to handle multi-colored buoys as well as the LED panel mounted on the light tower.

The generalized approach (provided in Algorithm 1) first creates a mask to isolate the object from the background within the ROI. To do this, a slight blur is applied and a second image is created to mimic the background by repeating the first column of the ROI image for 50% of the image, and the last column of the ROI image for the remaining columns. When this filter is subtracted from the original, the areas with the largest difference are identified and used to create a binary image mask to isolate the desired object as seen in fig. 9. The image is then converted from the default red-green-blue (RGB) to hue-saturation-value (HSV) color space. RGB colors are predefined to fall within specific hue values and white and black colors are defined by saturation. The object's color is then defined by the average hue and saturation of the masked object. Examples of this method are shown in fig. 9

Additional steps are required to handle color detection for the tall buoys and light tower panels. Color analysis of tall buoys is performed after splitting the final masked image into an upper and lower region, and individual colors are returned for each. This is due to the inclusion of the multi-color sock buoys in the Follow the Path task as seen in the bottom of fig. 9. Algorithm 1 includes an if statement to handle the return of separate top and bottom colors if the ROI has a class of 'TallBuoy'. The light tower panel color algorithm is slightly more complicated and is provided in Algorithm 2. First, the original ROI image is cropped to a smaller ROI focused on just the light panel. This is done based on the known geometry of the panel relative to the entire geometry of the light tower. The second modification occurs after the background is removed and involves removing the white panel border by applying a threshold to the intensity of the remaining image.

This entire process is repeated for each object found, and the object class and color information are recorded as a vote. Once enough votes are received, the object's class and color are labeled within Minion's map.

Algorithm 1 Color Detection

global variables
N, ROI image
F, Camera Frame
end global variables
Require: $N \cap F \geq 90\%$ of N
N = bilateralBlur(N)
$filter$ = empty(N.size) ▷ create image filter
$filter$ = fillBG($filter$)
$mask$ = threshold($N - filter,\ 75\%$) ▷ create image mask
n =rgb2hsv(n)
if N.class = TallBuoy **then**
color(top, bottom) = hsvThresh($n \cup mask$)
else
color = hsvThresh($n \cup mask$)
end if

Algorithm 2 Color Detection - Light Tower Panel

global variables
N, Light Tower ROI
n, Cropped Light Tower Panel
F, Camera Frame
end global variables
Require: $N \cap F \geq 90\%$ of N
n = crop(N)
n = biateralBlur(n)
$filter$ = empty(n.size) ▷ create image filter
$filter$ = fillBG($filter$)
$mask$ = threshold($n - filter,\ 75\%$) ▷ create image mask
n =rgb2hsv(n)
color = hsvThresh($n \cup mask$)

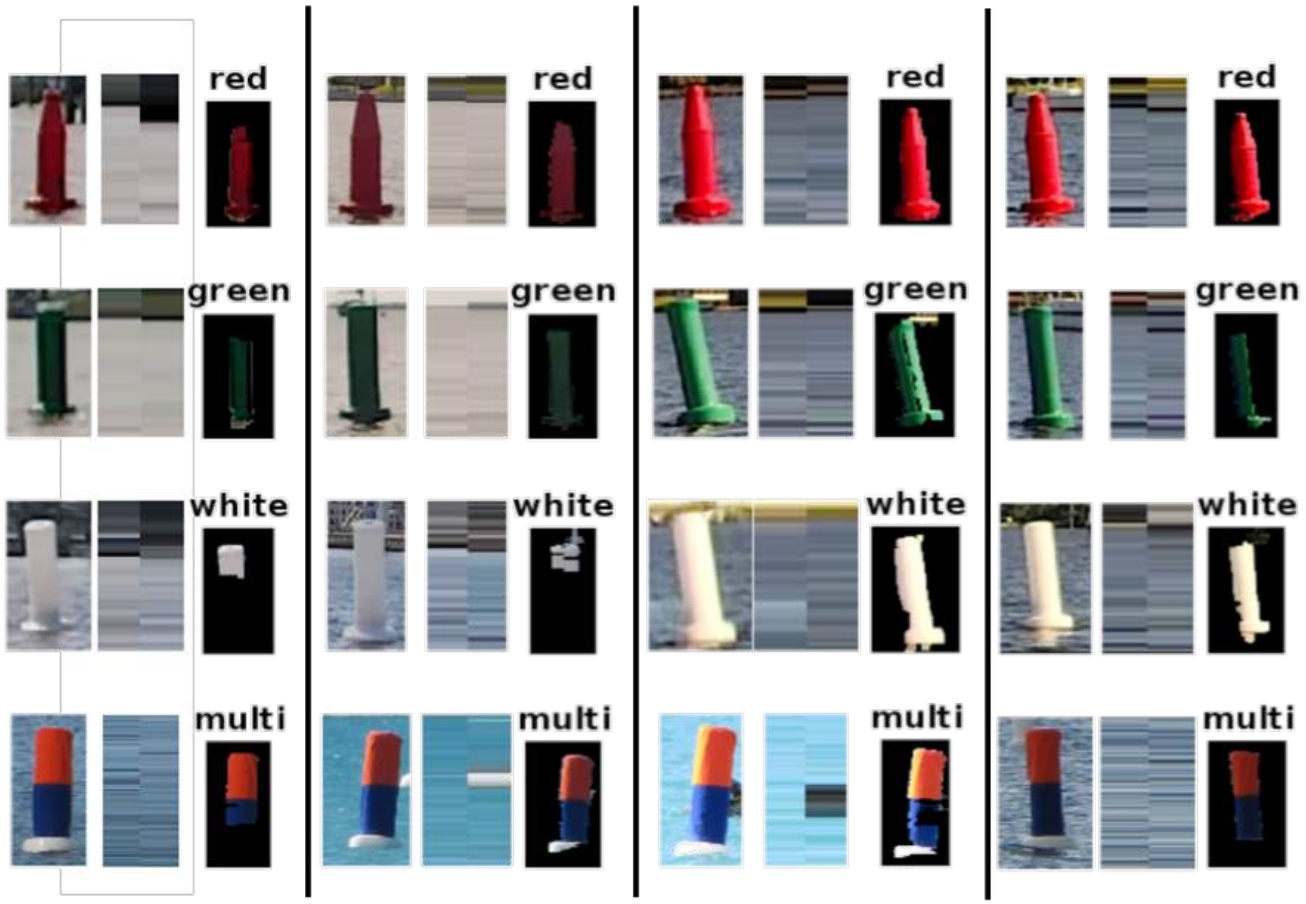


Fig. 9. Tall buoy ROI images, the corresponding generated filter used to remove the background and the color designated by the TallBuoy color algorithm. Each row in the image represents an object class and color pair, e.g. TallBuoy, white. Each column separated by the thick vertical line represents a separate instance of each object/color pair.

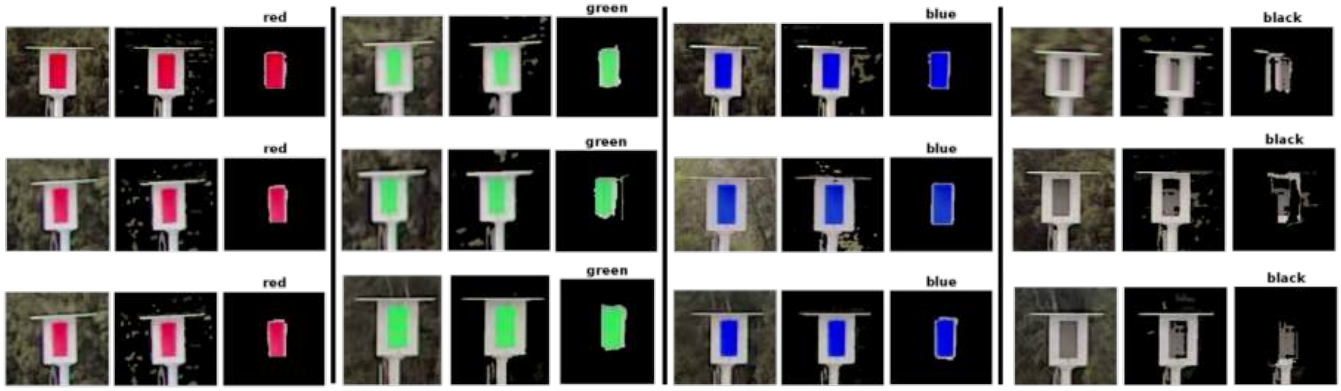


Fig. 10. Light tower ROI images, the corresponding ROI with the background filter applied, and the panel border removed with the color designated by the Light Tower Panel algorithm. Each set of three images represents the ROI of the detected object class 'LightTowerPanel' at various stages in the color detection algorithm. (Left). Each row in the image represents a sample from an expected color, e.g. TallBuoy, white. Each column separated by the thick vertical line represents a separate instance of each object/color pair.

REFERENCES

[1] "Computer vision toolbox documentation," Natick, Massachusetts, United States, 2024. [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html

[2] "Camera calibrator," Natick, Massachusetts, United States, 2024. [Online]. Available: https://www.mathworks.com/help/vision/ref/cameracalibrator-app.html

[3] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[4] "Lidar camera calibrator," Natick, Massachusetts, United States, 2024. [Online]. Available: https://www.mathworks.com/help/lidar/ref/lidarcameracalibrator-app.html

[5] E. Coyle, "Efficient grid-based clustering and concave hull extraction for

unstructured point clouds," journal of Field Robotics, 2024, submitted for publication.
[6] B. R. Lab, "Gscam," 2018. [Online]. Available: https://github.com/ros-drivers/gscam
[7] NVIDIA, "Nvidia deepstream." [Online]. Available: https://developer.nvidia.com/deepstream-sdk
[8] D. Thompson, "Neural network fusion of multi-modal sensor data for autonomous surface vessels," Ph.D. dissertation, Embry-Riddle Aeronautical University, 2023.

Appendix E: UAV Precision Landing Using Image Recognition and Control

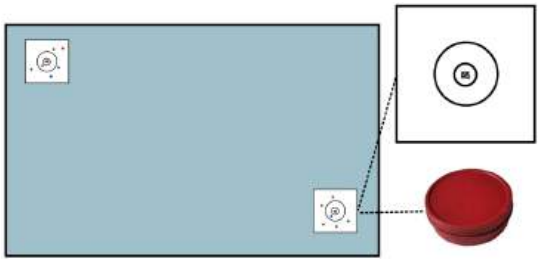
Mathis Abe, Sagar Sarkar, Rohith Vinnakota, Dr. Eric Coyle

I. Introduction

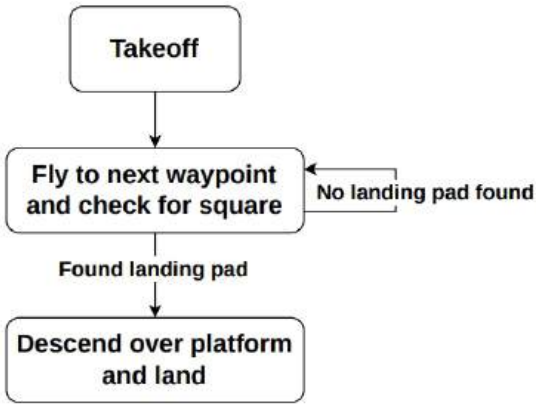
In recent years, UAV technology has become more affordable and accessible, leading to increased usage for tasks such as mapping, surveillance, and delivery. With the rise in autonomous operations, a critical challenge has emerged: landing accurately on a designated target, particularly in environments with poor GPS coverage or when the target's position is not precisely known. While GPS guidance is traditionally used for autonomous landings, it is often less reliable in terms of accuracy, especially when the target is moving. Image recognition offers a solution by improving success rates, as it allows the UAV to locate the target visually, even with less precise GPS data, and continuously update the target's position.

One key application of precision landing using image recognition is the autonomous landing and recharging of UAVs [1], which reduces the need for human intervention. This technology is especially important for landing on moving platforms, such as ships, where UAVs are increasingly used for reconnaissance or dangerous missions. Developing a reliable system for this purpose requires fast, accurate image recognition and tracking to locate the landing spot, as well as a robust control system to guide the UAV. Consistent communication between components is also essential for the system's effectiveness.

The objective of this work is to develop a vision-based system that allows a UAV to autonomously detect and land on a floating helipad over water. This system will be utilized in the international Maritime RobotX competition. The UAV is provided with waypoints indicating the estimated locations of the landing pads, which are assumed to be supplied by the USV. The UAV must take off, fly to these waypoints, verify if a landing pad is within its view, and proceed to land if a suitable pad is detected. Depending on the mode selected by the user, the UAV may also be required to land on the tin closest to the center of the platform. An outline of the task and the overall procedure is depicted in Figure 1, with subfigure 1b providing a high-level overview of the process.



(a) Overview of the task to be done.



(b) Diagram of the process.

Fig. 1: Task Overview.

II. Methodology

This section describes the various aspect of the work, which includes: 1) UAV and Payload, 2) Communication between on-board computer and the autopilot, 3) Image processing to identify the target, and 4) Controller Design.

A. UAV and Payload

The UAV, Figure 2, custom-built on the "Tarot Iron Man 650" frame, weighs 3.9 kg with the onboard computer and camera attached. It measures 90 cm x 90 cm x 35 cm, including the propeller guards. The Pixhawk V6X flight controller, running PX4, is equipped with a compass, accelerometer, and gyroscope to maintain desired attitudes. Sensor fusion combines position estimation from the IMU with GPS and barometer data to hold the current position. The flight controller also features redundant sensors to detect failures automatically.

For precision landing, the UAV uses an onboard computer to process images and controls, with a camera oriented straight down to capture the landing pad. The Raspberry Pi 4B, chosen for its affordability and ease of use, handles the processing tasks. The Raspberry Pi HQ camera, with a high resolution of 12.3 MP and a wide field of view (65° by 52°), is used to track features at lower altitudes, enhancing landing accuracy. At 10 meters altitude, the camera captures an area of 12.74 m by 9.75 m, sufficient to locate a 2 m by 2 m helipad even if the initial waypoint is off by 3.88 m.

To ensure buoyancy in case of a water crash, the UAV is equipped with pool noodles, providing enough flotation to recover the drone. The combined flotation capacity is 4.8 kg,

leaving 0.9 kg of buoyancy after accounting for the computing and camera payload.

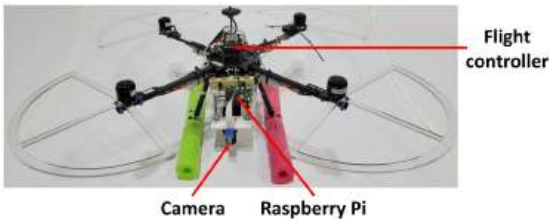


Fig. 2: UAV with camera and onboard computer.

B. Communications

The Unmanned Surface Vessel (USV) is an autonomous boat that provides the UAV with estimated GPS positions of the landing platforms. The Ground Control Station (GCS) monitors the UAV during testing using two data links: a 915 MHz two-way link for receiving flight controller messages, changing parameters, and obtaining telemetry data, and a 2.4 GHz link for communicating with the onboard computer over a Wi-Fi network using ROS (Robot Operating System) messages [7]. This second link also monitors image processing results and other messages from the onboard computer.

The UAV components include a camera connected directly to the onboard computer for image processing. The flight controller and onboard computer communicate via MAVROS, a ROS implementation of the MAVLink messaging protocol, allowing the onboard computer to access sensor data, position estimation, and flight controller states. Commands from the onboard computer to the flight controller are sent through this protocol. Communication between the USV and UAV, although not covered in this report, will use ROS messages over a 2.4 GHz link, providing sufficient speed and range.

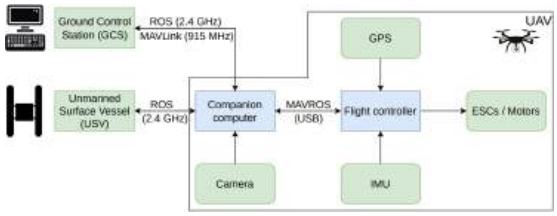


Fig. 3: Basic components and connections on and to the UAV.

1) ROS: The communication between the onboard computer and flight controller, as well as the USV and GCS to the onboard computer, is based on ROS (Robot Operating System). ROS simplifies communications in robotic applications by using nodes, which are programs that handle communication through topics and services. Nodes can publish topics for asynchronous communication, while services allow for direct communication with a request and response system. Communication between the flight controller and the onboard computer is facilitated by MAVROS, a ROS package that uses the MAVLink protocol, allowing for easier integration with the rest of the software running on the Raspberry Pi.

The ”UAV control node” is the main program that receives messages from the MAVROS node and the camera, using local and global positions for waypoint navigation. The Pixhawk state is crucial for switching flight controller states, such as switching to ”Loiter” if the target is lost. For safe testing, the remote pilot in command must have the ability to kill the running script and take over control. The ”UAV control node” subscribes to the Remote Control (RC) channels from the transmitter, which are published at a higher frequency of 30 Hz, allowing for efficient UAV takeover by killing the script. This ensures that the UAV can be controlled safely and effectively during testing.

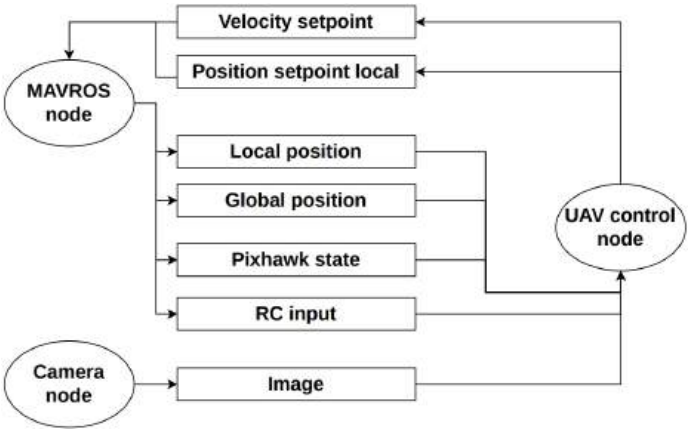


Fig. 4: ROS nodes and topics.

2) States: The landing process for the UAV involves two main subprocesses: flying to waypoints and checking for the landing pad, and descending over the target. Initially, the UAV is armed by the operator and follows a waypoint list, stopping at each waypoint for 3 seconds to check for landing platforms. If detected, the UAV switches to the descent procedure; otherwise, it moves to the next waypoint. During landing, the UAV can either aim for one of the tins or the center of the inner concentric circle, using visual features to guide its descent to 1.5 meters. At this altitude, it maintains its position above the target. If the target is lost, the UAV ascends to regain visual contact and loiters if the target remains lost for too long. This process ensures accurate and safe landings, even in challenging conditions.

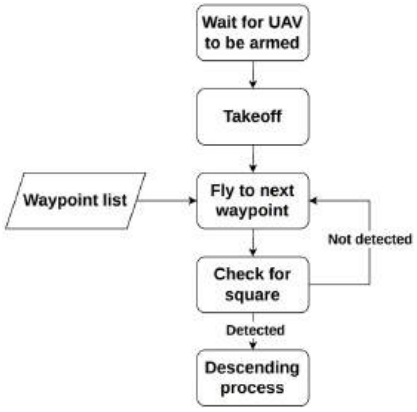
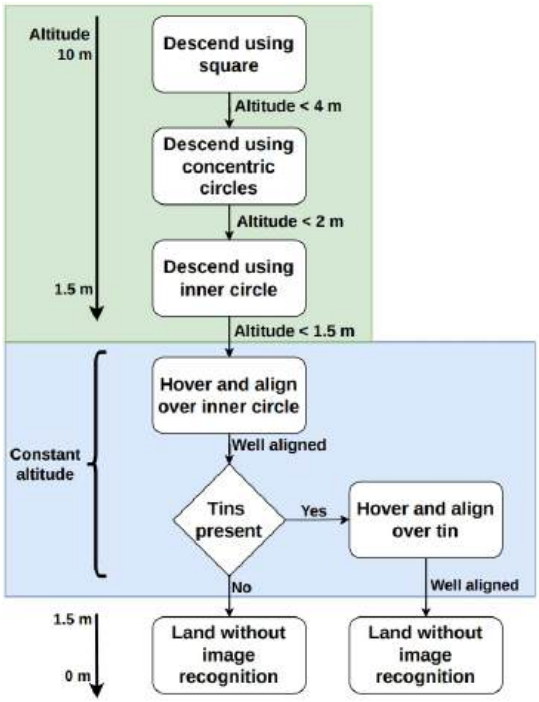
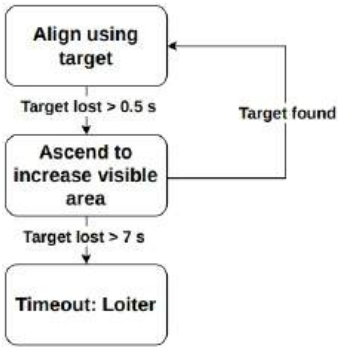


Fig. 5: Process of flying to waypoints to search for the landing pads.

C. Controlling the UAV



(a) Process of descending and landing on landing pad.



(b) Procedure when target is lost.

Fig. 6: State diagrams for descending and landing on target.

The UAV uses several control modes to navigate to waypoints and descend over the target. For waypoint navigation, the desired position in the ENU (East North Up) coordinate system is sent to the flight controller, which manages the internal controls to reach this position. During the descent from 10 m to 1.5 m, the vertical velocity is kept constant, and a controller aligns the UAV in the horizontal plane. Once the UAV maintains a constant altitude, an active altitude controller is needed. The position estimation from image recognition is used as the error for the control algorithm, with the Raspberry Pi, Pixhawk flight controller, and UAV components working together to manage the controls.

The horizontal velocity setpoints are calculated using image processing to estimate the target's position relative to the UAV. These distances are fed into a P controller, which outputs the horizontal velocity setpoint for the flight controller to manage the UAV's motion. The vertical controller can operate in three states: constant descend, constant ascend, and proportional control. Proportional control is used to hover at a constant altitude, while constant descend is used for most of the descent process. If the target is lost, constant ascend is used to increase the visible area. The maximum speeds commanded by this process are limited to 0.5 m/s to ensure safe operation.

III. Target

TABLE I: Known dimensions of the Landing Pad

Measurement	Dimension(m)
Total Landing Pad	2 x 2
Inner Circle Diameter	0.24
Outer Circle Diameter	0.72

The competition organizers specify the landing pad design, with dimensions of key features listed in Table I. While the background color is unclear, it is assumed to be light gray or white, as it was white in the 2022 RobotX competition. The pad is placed on floating dock units for stability, and due to the dock's size and weight, the angle relative to the horizon is expected to remain stable in calm waters. The pad features various visual markers to assist in descent, though some may be difficult to detect from higher altitudes, as shown in Figure 7a, where concentric circles are not visible from around 15 meters. A replica of the pad, built for testing, is shown in Figure 7b.

(a) Image taken from UAV. The circles are notably not visible.

(b) Landing pad mounted on floating dock.

Fig. 7: Landing pad on a floating dock and image captured from flying UAV.

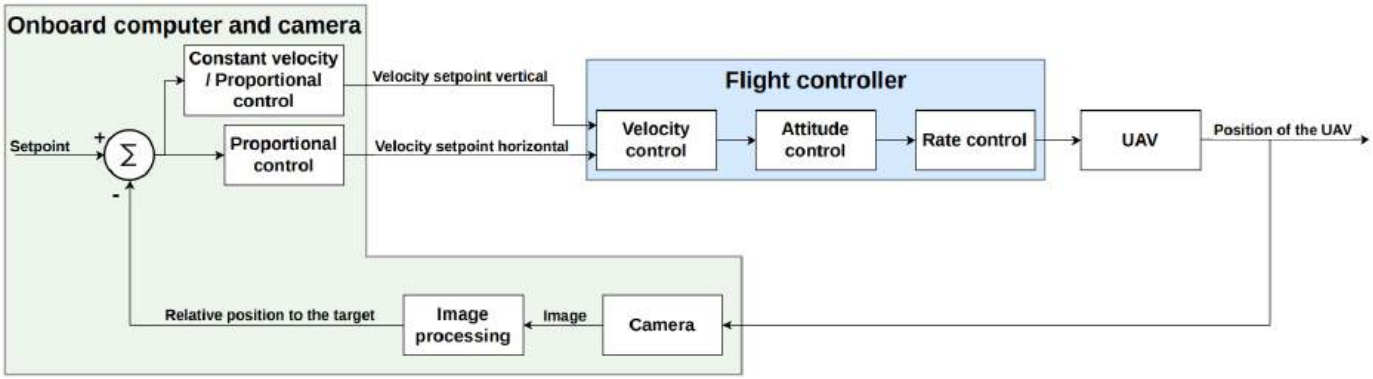


Fig. 8: Process of flying to waypoints to search for the landing pads.

IV. Image Recognition

The most challenging aspect of this task is the reliable detection of the landing target, requiring a robust algorithm that functions across various lighting conditions while minimizing the risk of detecting incorrect targets. Since different visual features of the landing pad are visible at different altitudes, the algorithm must select the appropriate feature to track at each stage of descent, as illustrated in Figure 9. A rule-based approach was chosen for image recognition, as machine learning was deemed impractical due to the lack of sufficient training data and the complexity of flying over water for data collection. Additionally, machine learning would require more computational resources and could complicate control processes, making a hard-coded approach more feasible for this task.

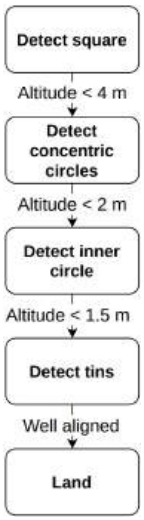


Fig. 9: Overview of the states used for image recognition during the descending process.

A. Pinhole Camera Physics

Since most transitions between states rely on altitude, accurately estimating the altitude above the target is crucial. GPS alone does not provide the necessary precision for this task, so altitude is instead estimated based on known features of the target, including the 2 m by 2 m platform, the 0.72 m outer circle, and the 0.24 m inner circle. The pixel size and location of these features help determine the altitude. This estimation is possible due to the pinhole camera model, where the perceived size of the target in the image depends on the UAV's height. As the UAV descends, the target appears larger, making it easier to identify features, though at very low altitudes, the target may fill the entire image. The formula for calculating the altitude is given by:

$$h = \frac{\frac{l_m w_{px}}{lpx}}{2tan(\frac{HFOV}{2})} \tag{1}$$

Where,

$$\begin{aligned} HFOV &= \text{Horizontal Filed of View} \\ l_m &= \text{Lenght of the object}, m \\ l_{px} &= \text{Length of the object}, px \\ w_{px} &= \text{Total width of the image}, px \end{aligned}$$

The altitude estimate helps identify the most reliable feature to track at any given height and limits the feature search to a specific size range based on this measurement, improving processing speed and reducing false positives. Camera distortion is largely ignored since the target is usually centered, except at the start when it may be near the edge of the frame. Even in this case, the algorithm remains robust, as detecting a square at this stage is relatively easy.

B. Position Estimation

To estimate the relative error to the target in meters, the center of the target must first be detected, using one of several features depending on the current state:

- Square platform
- Concentric circles
- Inner circle
- Closest tin to the center

Once detected, the relative position is converted from pixel error relative to the image center into meters, based on the UAV's height. For simplicity, the UAV's pitch and roll angles are not considered to avoid introducing noise, which might

require more advanced filtering. A 5-point running weighted average filter is applied to both altitude and horizontal error estimations to smooth out noise from faulty detections and external factors, such as wind gusts. This filter gives greater weight to more recent measurements and those taken when the UAV is closer to the target, assuming these are more accurate.

$$w_{dist} = \begin{cases} 10 - d, & \text{if} 10 - d \geq 2 \\ 2, & otherwise \end{cases} \tag{2}$$

$$w_{time} = e^{-\Delta t} \tag{3}$$

$$w = w_{dist}.w_{time} \tag{4}$$

C. Detection of Square Platform

Since the landing pad is bright and contrasts with the darker surrounding water, square detection can be performed by identifying a bright object with the appropriate size and shape. This detection method is used both during the UAV's descent over the landing pad and for the initial identification of the pad's presence.

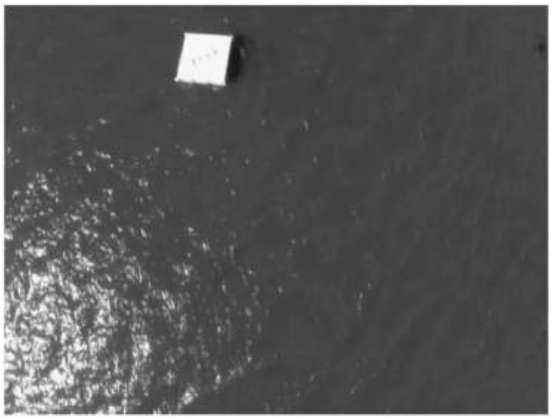
1) Bimodal Image: To understand the thresholding process for square detection, it's important to first grasp the bimodal nature of the expected image. In a bimodal image, the histogram displays two peaks representing two distinct distributions—typically the foreground and background. Figure 10b shows this bimodal distribution from a UAV image over water, where a grayscale image is sufficient since the histogram only contains pixel intensity or brightness. However, unlike typical images where the two peaks are similar in size, here the peak near 250 (representing the landing pad) is smaller due to the pad occupying much less of the image.

2) Otsu's Thresholding: The challenge in thresholding lies in determining a suitable intensity threshold to separate the foreground from the background. For bimodal images, Otsu's method [8] is commonly used, as it divides the histogram into two classes by minimizing within-class variance based on pixel intensity. However, UAV images over water can have bright spots caused by sunlight, complicating the square detection. To address this, Otsu's method can be modified to favor the second peak (the platform), by adjusting the within-class variance function to compute a higher threshold closer to the second peak. Eq. 6 is the modified form of the original eq. 5. Figure 11b shows the result of setting the factor x = 50. The noise caused by the reflection of the sun is drastically reduced and can be filtered out more easily.

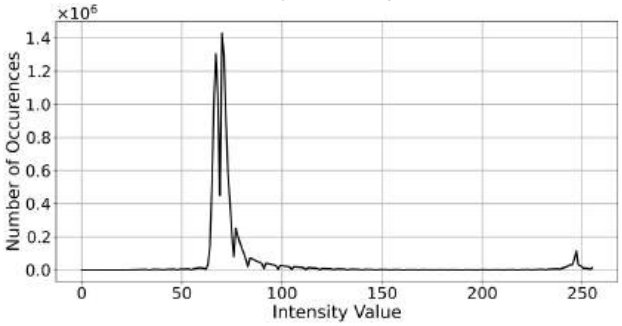
$$\sigma_w^2(t) = w + 0(t).\sigma_0^2(t) + w_1(t).\sigma_1^2(t) \tag{5}$$

$$\sigma_w^2(t) = w + 0(t).\sigma_0^2(t) + w_1(t).\sigma_1^2(t).x \tag{6}$$

3) Morphological Operations: To minimize the presence of small unwanted objects in the image, an erosion process is applied first, followed by a dilation [10]. These operations use a kernel size that is $\frac{1}{38}$ of the image width to compute the local minimum for erosion and the local maximum for dilation. The local extreme values are then applied to the kernel's anchor point, typically its center. This sequence of operations acts as a closing for erosion and an opening for dilation, effectively cleaning up the image.



(a) Grayscale image.

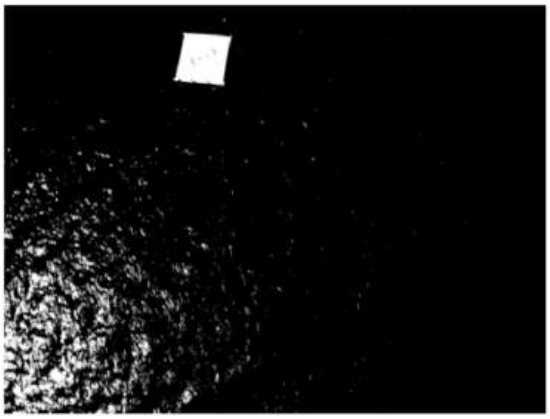


(b) Histogram of blurred grayscale image.

Fig. 10: Grayscale image and corresponding histogram.

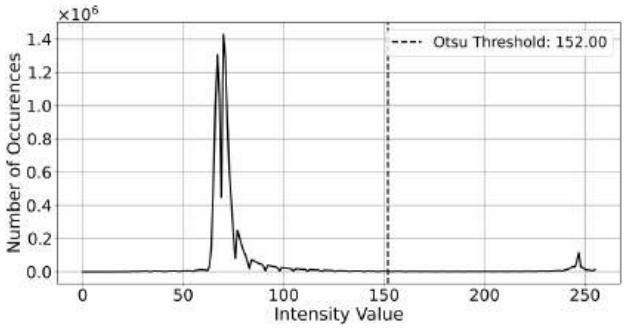
4) Contour Approximation: After performing morphological operations like erosion and dilation, contours are fitted to the image to locate the square and verify that it meets specific criteria. These criteria include the contour having the correct approximate area, four corners, and sides of roughly equal length. The estimated size of the square is initially determined based on the altitude provided by the flight controller. Given that altitude measurements can deviate by up to 2 meters, a tolerance range of ±3.5 meters is added to the estimated size to ensure accurate detection despite potential altitude inaccuracies. This range is broad enough to account for errors, ensuring the square is identified within the calculated size range. Once the square is detected, the altitude derived from this detection is used as the new reference for further square detection, with the same tolerance applied. To confirm the sides are of similar length, the longest and shortest sides are compared, and the contour is only accepted if the difference between them is less than 20% of the longest side's length, as determined by empirical testing. Valid contours, filtered by these constraints, are superimposed on the original image, and the center of the contour is used to estimate its position.

To detect a landing pad, the same method of thresholding and contour approximation is used, and the algorithm runs repeatedly for 3 seconds to account for noise or wind interference. This generates a list of potential landing pad locations relative to the UAV. To determine if one or two platforms are present, a dip test is used to check for bimodal distribution in

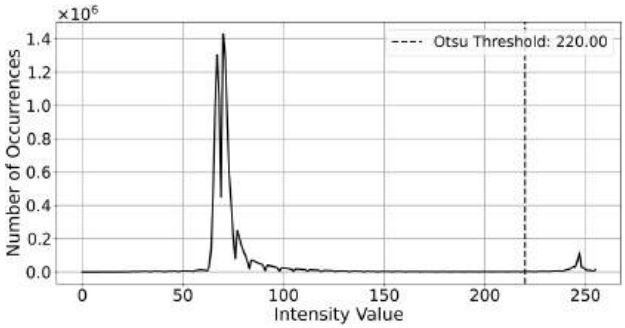


(a) Thresholded image.

(b) Thresholded image using the different modified threshold value.



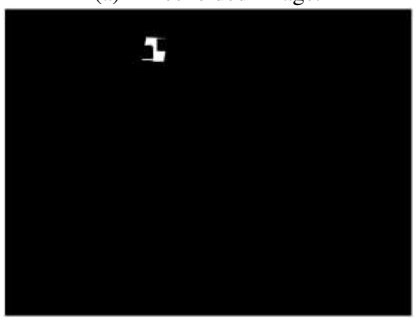
(c) Histogram with threshold indicator.



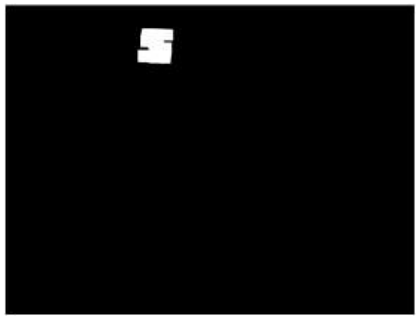
(d) Histogram with threshold indicator at a higher value.

Fig. 11: Comparison result of original and modified Otsu's thresholding method.

(a) Thresholded image.



(b) Result of erosion.



(c) Result of dilation.

Fig. 12: Result of morphological operations with a kernel size of $1/38$ of the image width.

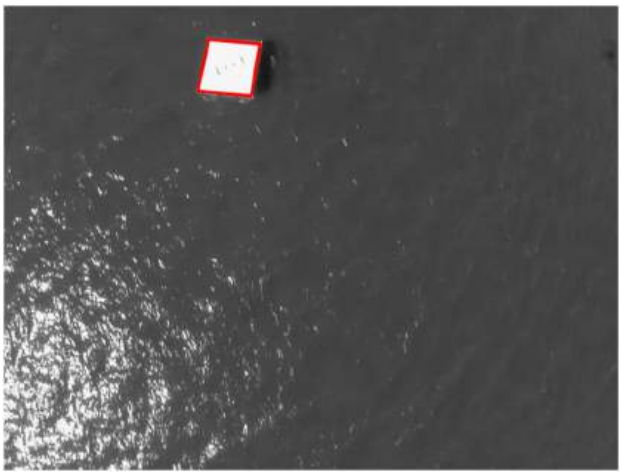
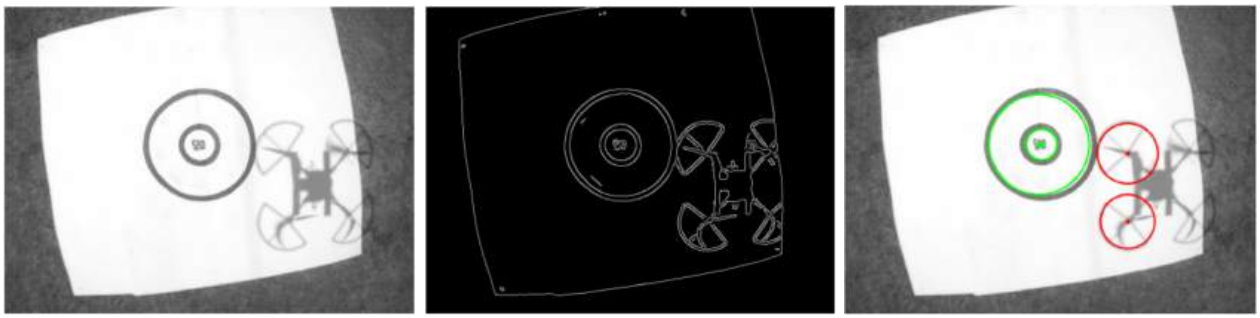


Fig. 13: Fitted contour. For easier viewing the contour is superimposed on the grayscale image.



(a) Grayscale image of the landing pad. (b) Result of canny edge detection. (c) Result of Hough Circle Transform.

Fig. 14: Canny edge detection & Hough-Circle transform performed on a grayscale image. The red circles in c) are unwanted circles.

the horizontal plane. If two platforms are detected, the UAV targets the closer one and saves the location of the other for future reference. If only one platform is found, it becomes the primary target, and the UAV begins descending based on the continuously updated position of the square.

D. Detection of Circles

As the UAV descends over the target and the square reference fills most of the image, it can no longer serve as a guide for the descent. Therefore, the UAV must rely on one or both concentric circles to estimate its relative position. This process involves using Canny edge detection to identify potential contours, followed by the Hough Circle Transformation to locate the most likely circles from the detected edges.

1) Canny edge Detection: The Canny edge detection algorithm involves several steps to identify sharp gradients in an image. It begins by applying a 5x5 Gaussian filter to minimize noise. Then, it calculates the first derivative in both horizontal and vertical directions, followed by additional steps to eliminate unwanted pixels that do not represent edges. The outcome of applying the algorithm to an image of the landing pad is shown in Figure 14b, with the algorithm operating on a grayscale image based on pixel intensity values.

2) Hough Circle Transform: The Hough Circle Transform, an extension of the Hough Transform, is employed to detect circular objects within images. This algorithm evaluates how well a circle of specified radius fits the edge points in the image, searching for circles within a defined range of radii, denoted as r_1 and r_2, which are determined based on the UAV's estimated height. By limiting these radii to a relatively narrow range, the algorithm significantly enhances the speed of circle detection and increases measurement frequency.

To identify the concentric circles, the Hough Circle Transform is applied with adjusted settings for the minimum and maximum radii. Figure 14c, illustrates the circles detected during this process. To ensure the accurate identification of the landing pad's location, it is essential that one of the smaller circles aligns closely with the center of the larger circle. The center point of the larger circle is then utilized as a reference for the UAV's control algorithm. This method of selecting circles based on their centers allows for lower thresholds in the Hough Circle Transformation, enhancing robustness against noise and image distortions. In contrast, higher thresholds may result in fewer detected circles, which can complicate the identification process, especially under challenging lighting conditions.

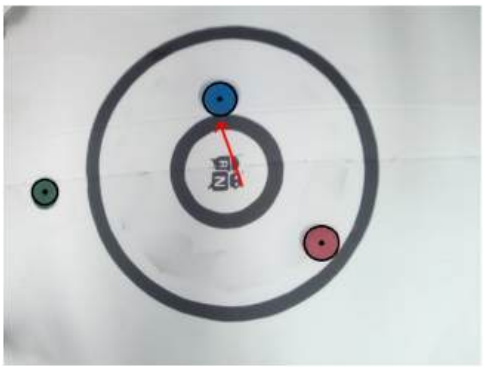
The detection of the inner circle follows a similar approach as that for the concentric circles, but it employs a higher threshold for the Hough Circle Transformation and a narrower range of radii. This increased threshold requires the circle to closely resemble a perfect shape to be considered valid, which helps limit the detection to a single circle. Although this method may be less robust than the concentric circle detection approach, the proximity of the UAV to the target means that noise and distortions have a diminished impact, allowing the feature to occupy a larger portion of the image.

E. Detection of Tins

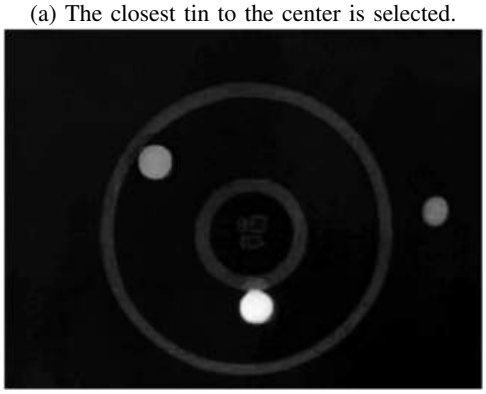
To land on a tin, the UAV hovers at an altitude of 1.5 meters and aligns itself above the closest tin relative to the center of the inner concentric circle. For this alignment, both the tin and the center of the inner circle must be visible, as the distance between them is crucial for selecting the appropriate tin. The process involves several steps: first, the UAV calculates which tin is closest to the inner circle's center, saving the distance and position of the tin. It then checks for additional tins with matching distances and selects the one closest to its previous position in the image frame as a reference for alignment.

The UAV must maintain visibility of both the inner concentric circle and the tin, which constrains its alignment altitude to a minimum height. This ensures adequate space around the features to mitigate the effects of wind gusts that could affect the camera's visible area. As a result, the UAV operates at a relatively high altitude of 1.5 meters to facilitate a stable alignment process. Figure 15a illustrates the detection of the tins, highlighting the closest tin selected for alignment with a red arrow pointing to the desired location.

1) HSV Color Model: The detection of the tins relies on their vibrant colors, which contrast with the predominantly gray, white, and black background of the platform. To enhance this contrast, the image is converted into the hue, saturation, value (HSV) color model. Unlike the red-green-blue (RGB) or cyan-magenta-yellow key (CMYK) models, HSV allows for effective differentiation based on color intensity. In this



(a) The closest tin to the center is selected.



(b) Saturation channel of HSV image.

Fig. 15: Result of Tin Detection. In the saturation channel, the tins clearly stand out as they are the only colorful objects.

model, Hue indicates the perceived color, Value represents brightness, and Saturation denotes the purity of the color. Since perfect white and black have zero saturation, the saturation channel proves particularly effective for detecting colorful objects, such as the tins. Figure 15b displays the original image alongside the grayscale representation of the Saturation channel, highlighting the tins while only faintly revealing the concentric circles. However, this image was captured indoors, so it does not reflect the challenges posed by suboptimal lighting conditions, which will be addressed in a later chapter.

V. Results

A. Offline Testing and Tuning

Figure 16, illustrates a test setup designed to verify the image processing capabilities without the need for actual UAV flight. The target used in the tests is scaled down by a factor of 5, allowing the UAV to be held at a height of 2 meters, which simulates a flying altitude of 10 meters. This scaling allows for a comprehensive testing of the full range of altitudes at a manageable level.

During the testing process, a video stream and raw data from the flight controller are recorded as a Rosbag, facilitating the testing of the code through playback of the captured data. The data obtained from these tests is crucial for fine-tuning the image recognition algorithm. Key parameters that require adjustment include the scaling factor for Otsu's thresholding technique, the threshold settings for Canny edge detection, the thresholds for the Hough Circle Transformation, and the tolerances for size estimation of the square, concentric circles, and tins.

Fig. 16: Collecting sample data on a scaled-down platform to validate the image recognition algorithm.

B. Testing and Simulation

The algorithm is also tested in a simulation environment. A modified Gazebo setup provided by the RobotX competition organizers is utilized for this purpose. The simulation features a 3DR IRIS quadcopter equipped with a downward-facing camera, along with landing pads positioned on the water. Figure 17, illustrates a landing scenario using this simulation setup.

1) Landing on the Center Circle: Figure 19 illustrates the estimated target position derived from image recognition. Subfigure 19a displays a plot of altitude against the total distance to the target in the horizontal plane during a typical trial. Since the x-axis represents the distance rather than a specific position, the UAV does not reach zero and cannot extend into negative distances. The UAV is observed to be well aligned at the beginning of its descent, with the dashed and dotted lines indicating the closest edge of the platform and the outer circle, respectively. These lines serve only as references for the reader and are not actual markers on the landing pad.

The distance to the center of the target is categorized into three safety zones, as depicted in Figure 18. The first category is a safe landing zone, which is within 87.5 cm of the target center. The second category is a gray area where the safety of the landing is conditional, ranging from 87.5 cm to 128.5 cm. Lastly, the third category is considered an unsafe landing if the UAV is positioned beyond 128.5 cm from the target center. These distances are calculated based on the landing gear, which is 25 cm apart, meaning that a leg can fall off

Fig. 17: Photo series of UAV landing on the platform.

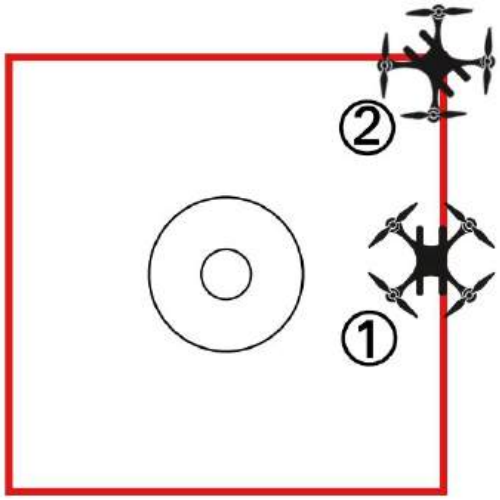


Fig. 18: Edge cases relevant for a safe UAV landing. Edge case 1 represents the border of a safe landing at a distance to the center of 87.5 cm. Between a distance of 87.5 cm to 128.5 cm (edge case 2), the landing is potentially safe or unsafe depending on the position of the UAV. Everything beyond 128.5 cm will fall off the platform and is therefore unsafe.

precisely at 87.5 cm if positioned parallel to the edge of the platform.

In Subfigure 19a, the algorithm effectively maintains the UAV's alignment, keeping it within the outer circle limits. Subfigure 19b illustrates the altitude over time. The UAV uses the square as a reference until approximately 15 seconds into the descent. When the reference switches to the concentric circles, a slight peak appears due to a mismatch in expected feature sizes. After reaching an altitude of 1.5 m, the UAV begins the alignment process at around 21 seconds, resulting in a flattening of the altitude graph. The graph concludes at 28 seconds, as image recognition is not employed during the final unguided portion of the landing.

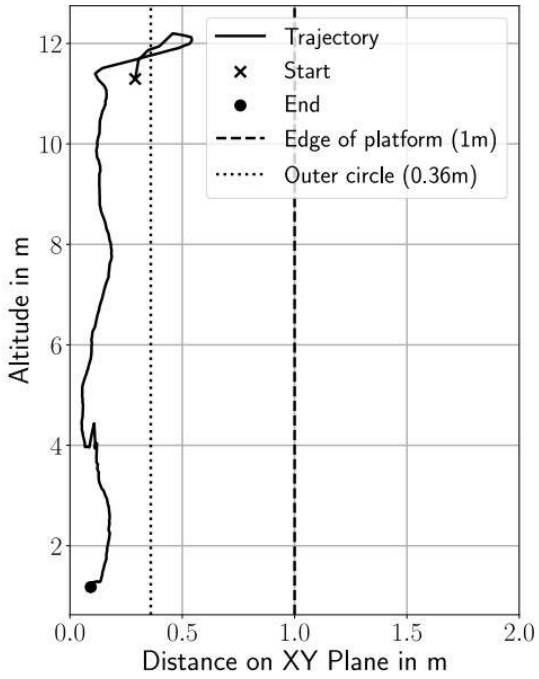
2) Landing on a Tin: The procedure for landing on a tin closely resembles that of landing on a platform; however, it necessitates a longer hovering phase due to the requirement of aligning first with the center of the platform and then with the tin. This is evident in Figure 20, which shows that when the UAV switches to using the tin as a reference, the distance to the target peaks at 39 seconds. In this particular instance, the hovering duration increases by approximately 8 seconds, from the moment the UAV is well aligned with the inner circle at 39 seconds to when it is aligned with the tin and ready to land at 47 seconds. Although the exact increase in hovering time can vary, it consistently extends due to the dual alignment process needed for landing on a tin, which can be challenging, especially under windy conditions.

C. Real-Time Testing

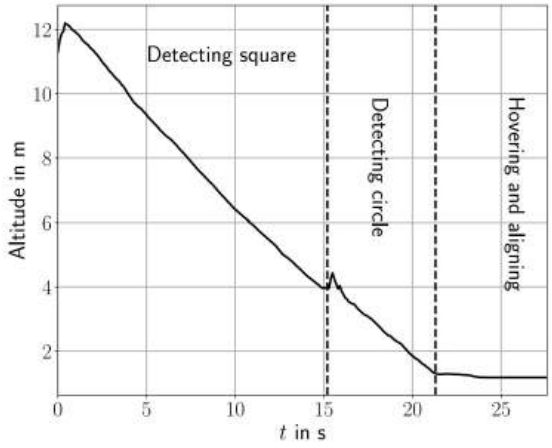
1) Testing on Land: Following the successful validation of the underlying logic and image recognition in simulation, multiple flights were conducted to assess the accuracy and effectiveness of the UAV's landing capabilities on both the tin and the center circle. The initial phase of testing focused on tuning the proportional (P) controller responsible for maintaining the UAV's horizontal position. This tuning process involved adjusting the P controller for both axes in the horizontal plane until oscillations were observed. Figure 21, illustrates the effects of an overly aggressive gain setting, where the camera's position estimate shows a peak at 50 seconds, indicating a switch in tracking from the inner circle to the tin.

However, there are inherent challenges in tuning the controller based solely on camera position estimation. As mentioned earlier, neglecting the UAV's pitch and roll angles can introduce significant errors, particularly when the drone is not level. Figure 22, highlights this issue, as the UAV appears almost directly over the target, yet the estimated position inaccurately indicates the target is to the right. The position estimation, combined with the controller, creates a negative feedback loop; if the target is mistakenly detected to the right, the UAV banks in that direction to accelerate, further compounding the error. Consequently, the slight oscillations observed in the position plot may not reflect actual movement but rather artifacts of the measurement method. To enhance the tuning process, an external position measurement system would be highly beneficial. In contrast, the altitude controller did not require tuning, as the gains from the simulation performed well during real-world testing.

2) Landing on the Circle Center: A total of seven attempts were conducted to assess the algorithm's precision during UAV landings. Although wind speed was not directly measured, it was estimated at around 1 m/s from north to east. Two attempts were prematurely stopped due to difficulties in reaching the first waypoint for target detection. This limitation was



(a) Altitude vs. distance in x-y plane plotted. At the start of the ascent, the UAV is already well aligned.



(b) Altitude vs. time. The different states are marked by vertical lines.

Fig. 19: Position measurements produced by the image processing algorithm

attributed to the UAV being flown in a netted flight cage with a maximum safe testing altitude of approximately 9 m. The failure to reliably ascend to the desired height of 6 to 7 m is likely due to poor GPS vertical positioning, exacerbated by the proximity of the cage to a building. In contrast, tests conducted over water showed more consistent altitude readings.

The results of the five successful landing attempts, measured from the camera's center to the inner circle's center, are summarized in Table II. The weather conditions were sunny, leading to high image exposure, but this did not hinder feature tracking. The UAV successfully landed within the outer circle for all attempts, achieving an average accuracy of 16.23 cm, all within the safe limit of 87.5 cm. Additionally, the UAV landed

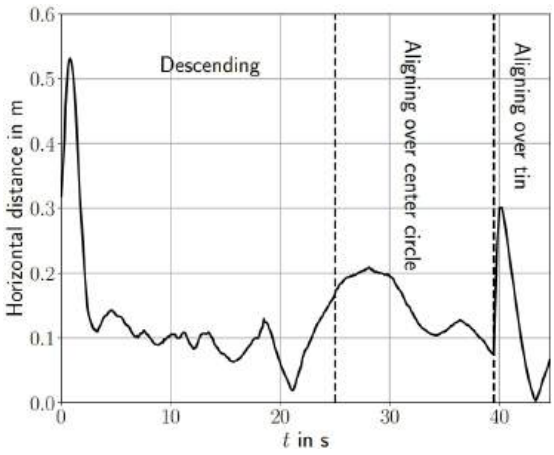


Fig. 20: Distance to object of interest vs. time. The object of interest switches from the middle of the platform to a tin at the second dashed line.

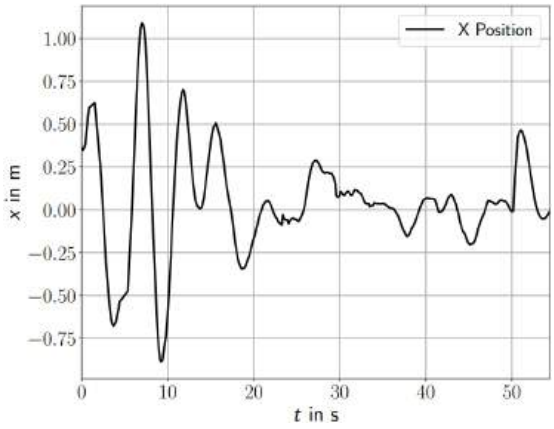


Fig. 21: Distance to the landing pad in x. x refers to the east-west axis, with east being positive x. Oscillatory behavior can be observed especially at the beginning of the alignment process.

inside the inner circle (with a radius of 12 cm) in 2 out of the 5 attempts, demonstrating the method's potential for effective landings over water. The position estimates produced by the image recognition algorithm during the landing procedure are illustrated in Figure 24, with attempt number 5 showing the most significant distance fluctuations to the target. These fluctuations may have been influenced by increased wind gusts during descent; however, further tests are required to validate this observation. Notably, the initial larger errors in distance estimation did not appear to impact the fluctuations later in the descent.

TABLE II: Results of landing on the center circle.

Attempt	Dist. Northward(cm)	Dis. Eastward(cm)	Total Dist.(cm)
1	-26	5.5	26.58
2	-26	-1	26.02
3	3.5	-4	5.32
4	10	-5	11.18
5	8	9	12.04

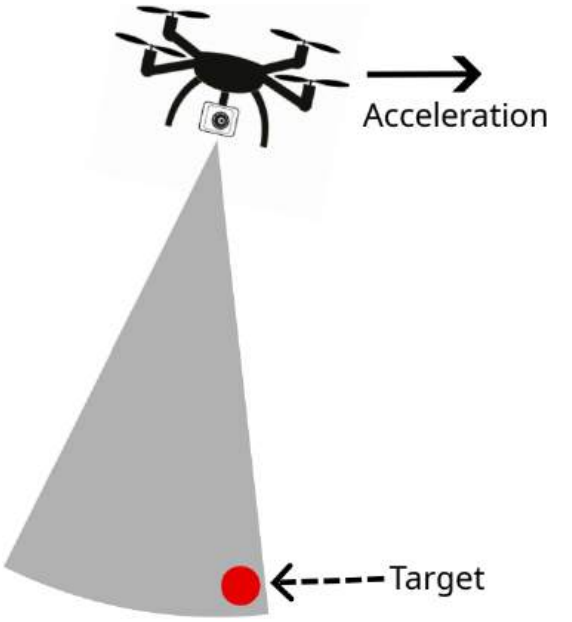


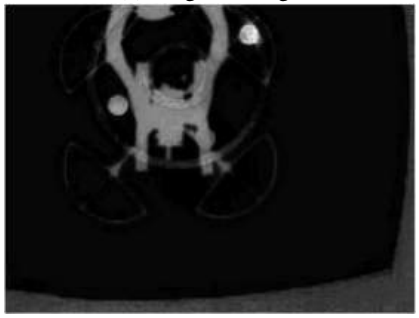
Fig. 22: The position estimation returns poor results if the UAV is not level relative to the horizon, as the camera is not pointed straight down as assumed in the calculations.

3) Landing on Tin: To evaluate the UAV's landing procedure on a tin, three tins—colored blue, green, and red—were strategically placed at varying distances from the center of the landing pad under calm weather conditions with an estimated wind speed of 1 m/s from the north. A total of seven attempts were made, achieving an average accuracy of 22.31 cm from the selected tin. In attempts two through seven, the algorithm consistently selected the tin closest to the center of the platform. However, during the first attempt, the algorithm initially identified the correct tin for tracking but later switched to a nearby one due to intermittent detection of the primary tin, leading to inconsistent results. The distances recorded reflect the UAV's attempt to land on the tin it targeted rather than the initially detected one.

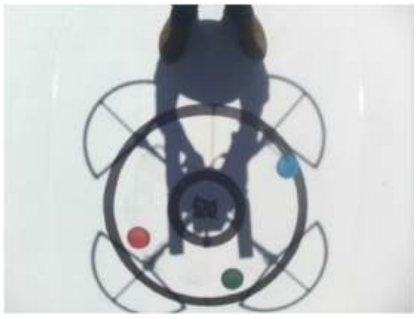
The algorithm's performance was significantly influenced by lighting conditions, particularly due to overexposure, which compromised the effectiveness of the Hough Circle Transformation and Canny edge detection applied to the saturation channel. Subfigures 23a and 23b, illustrates the overexposed image alongside the corresponding saturation channel. In this case, color information was lost as the shadow of the drone appeared as colorful as the tins themselves, primarily caused by an incorrect aperture setting that was too wide open. This highlights the importance of proper lighting settings to ensure reliable detection and tracking of targets. Table III summarizes the result using the default aperture setting.

In contrast to the previous tests, subfigures 23c and 23d, illustrates an image captured with a more closed aperture setting. Although the green tin still lacks clarity in the saturation channel, the red and blue tins are more prominent, with brighter spots indicating higher saturation values. With this improved aperture adjustment, further tests were conducted, allowing the Canny edge detection and Hough Circle Trans-

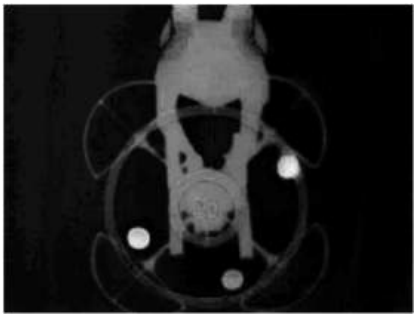
(a) Original Image



(b) Saturation Channel

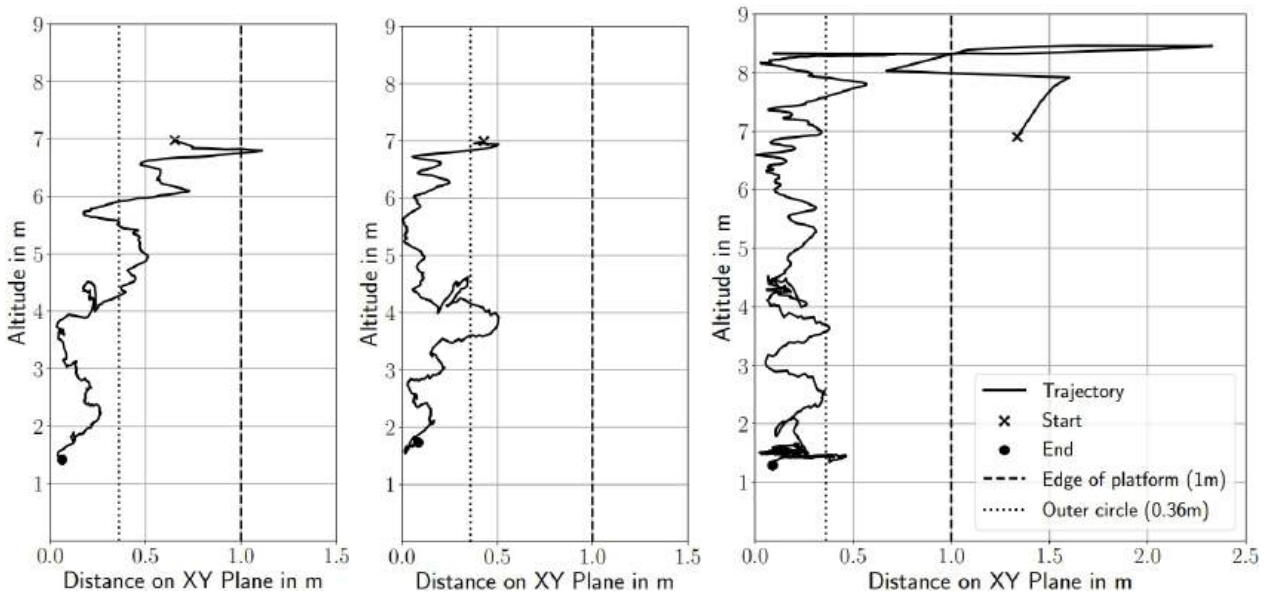


(c) Original Image with Adjusted Aperture



(d) Saturation Channel with adjusted aperture

Fig. 23: Comparison of saturation channel result at different aperture setting.



(a) Attempt 1. (b) Attempt 2. (c) Attempt 5.

Fig. 24: Altitude vs. distance to center of landing pad for three different landing attempts.

TABLE III: Results of landing on the tin.

Attempt	Dist. Northward(cm)	Dis. Eastward(cm)	Total Dist.(cm)
1	6	6	8.49
2	-15	-25	29.15
3	-32	-3	32.14
4	-14	-14	19.80
5	-20	-12	23.32
6	-2	22	22.09
7	7	-20	21.19

TABLE IV: Results of landing on the tin, closed aperture.

Attempt	Dist. Northward(cm)	Dis. Eastward(cm)	Total Dist.(cm)
1	11	26	28.23
2	22	-22	31.11
3	22	-12	25.06
4	-4	-5.5	6.80
5	8	-8	11.31
6	18	-5	18.68
7	5	1	5.10
8	40	20	44.72
9	3	12	12.37
10	24	2	24.08

formation to be fine-tuned for greater robustness. Table IV summarizes the results of testing using a different aperture setting. A total of 12 additional test flights were performed, although two had to be interrupted early due to the UAV flying too close to the top of the flight cage. Notably, both interruptions occurred before the UAV could detect the target. In the sixth attempt, the algorithm once again mistakenly switched to tracking the wrong tin during the alignment process. As in previous instances, the distance measured was relative to this incorrectly tracked tin rather than the intended target. The average distance from the desired landing location was recorded at 20.75 cm, despite the wind speed increasing to approximately 3 m/s from the East-northeast (ENE). This indicates that, even with the increase in wind speed from 1 m/s to 3 m/s, the more reliable visual tracking improved landing accuracy, demonstrating a decrease in average error from the previous trial's 22.31 cm to 20.75 cm.

D. Testing over Water

To test the UAV's landing capabilities over water, the landing pad was mounted on floating dock pieces situated in a tidal creek, where the water flow direction remained relatively constant. The helipad was stabilized using two anchors arranged in a V-shape against the current, minimizing movement caused by changes in flow or wind. Figure 26 illustrates this setup with the UAV positioned over the target. The UAV was launched from an elevated dock on land and manually guided to a position estimated to be above the floating platform, resulting in a total of 12 flights.

TABLE V: Results of landing on the floating helipad.

Attempt	Result
1	Did not fly far enough
2	Flew too far
3	Flew too far
4	Inner circle detection failed
5	Flew too far
6	Square detection failed (glare)
7	Lost connection (flight controller to onboard computer)
8	Landed successfully (29 cm)
9	Inner circle detection failed
10	Detected two squares, aborted
11	Accidental landing (lost connection, distance 43 cm)
12	Square detection failed

The outcomes of these attempts are detailed in Table V.Only two flights successfully landed on the platform, while four attempts were interrupted because the estimated waypoint was too distant from the helipad, rendering it out of the camera's view. Additionally, connection issues between the onboard computer and flight controller triggered a failsafe event. This

automatic failsafe mode switched to manual control when communication was lost, leading to one unintentional landing on the platform due to a low throttle stick position and another abort due to the same communication issues. Image processing challenges also hindered success, with the detection of the landing platform as a square failing three times due to glare on the water and inner circle tracking failing an additional two times, resulting in the UAV losing the target and ascending automatically. Due to these complications, attempts to land on a tin were not conducted over water.

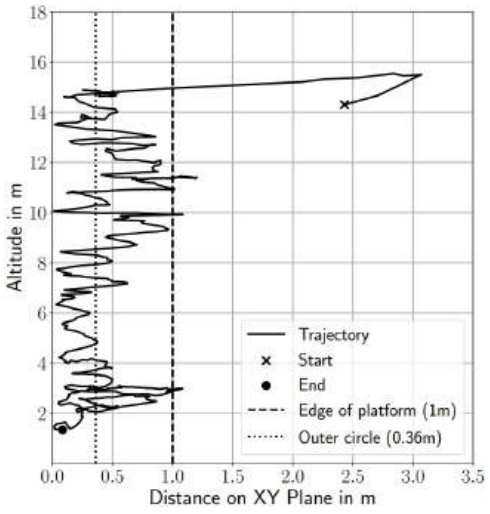
1) Successful Landing: Out of the two successful landings on the helipad, only one was planned, while the other occurred due to a lost connection between the onboard computer and the flight controller. The detailed examination focuses on the successful landing, as depicted in subfigure 25a, which illustrates the UAV's landing path based on camera position estimation. The UAV initiated its descent from 15 meters and aimed to align with the platform. However, during the descent to 1.5 meters, the distance to the center of the platform was frequently greater than observed in previous land-based attempts, likely due to the platform's movement on water and the higher wind speeds of approximately 4 m/s experienced during testing. Additionally, subfigure 25b highlights two instances where the altitude increased at 55 seconds and 62 seconds due to temporary tracking losses of the inner circle, prompting the UAV to ascend.

2) Bad Waypoints: The failure to reach the correct waypoint stemmed from challenges in visually estimating the position from land. However, this issue is anticipated to be resolved in future testing, as the UAV will receive waypoints directly from the Unmanned Surface Vehicle (USV), which offers an accuracy of approximately 20 cm. Consequently, the only potential problem would arise if there is an error in identifying which object corresponds to the landing platform, which could lead to the waypoint being set too far away.

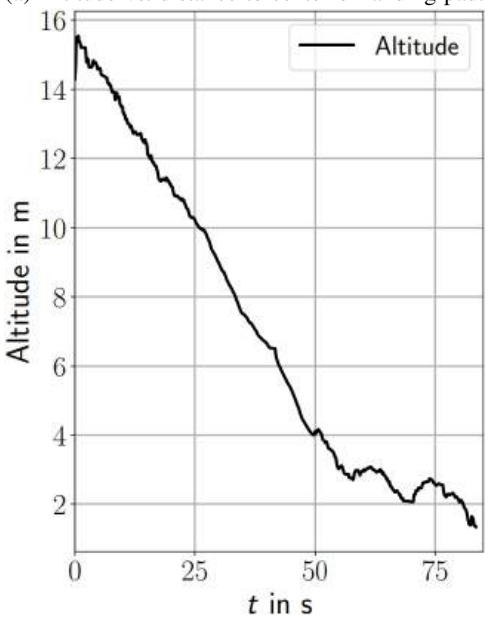
3) Bad Connection: The cause of the connection loss between the onboard computer and the flight controller remains unclear and could not be replicated during testing in the flight cage. Potential explanations for this issue include overheating of the onboard computer or a loose USB connection between the two components. Additional testing is necessary to confirm the exact cause of the disconnection.

4) Failure of Square Detection: The intensity and density of water glare were significantly more pronounced during flight testing compared to scaled-down testing. The increase in altitude from 2 meters during scaled-down tests to approximately 15 meters during actual testing resulted in a wider field of view for the camera. This allowed more sunlight reflections to enter the camera's view, creating a continuous glare pattern instead of isolated spots. Consequently, filtering out the glare using thresholding and morphological operations became considerably more challenging.

Initially, the UAV was flown at an altitude of 10 meters; however, this was exceeded due to the higher launch position relative to the target. At lower altitudes, when the sun was nearly overhead, glare was concentrated near the helipad, distorting its detected shape, as depicted in Figure **??**. To accurately recognize the helipad as a valid landing platform,



(a) Altitude vs. distance to center of landing pad.



(b) Altitude vs. time plot.

Fig. 25: Path captured by the camera on the successful landing attempt.

the detection parameters needed adjustment. The expected size of the area was broadened from ±3.5 meters to ±5 meters, and the accepted angles were modified.

Despite these adjustments, issues arose when glare and the landing platform were within the same field of view. Figure 27 illustrates how glare was incorrectly identified as the landing pad due to the relaxed contour detection parameters. Merely altering the detection parameters was insufficient to resolve this issue. Additionally, varying sun intensities meant that the largest object post-thresholding could not always be assumed to be the landing platform. This is evident when comparing histograms of the landing platform in water at 15 meters with and without glare, where both displayed similar distributions and sharp peaks at 255, complicating glare filtering efforts.

Fig. 26: Test setup on water. The landing target is mounted to floating dock units and held in place by two anchors. The anchors are attached to the blue and pink ropes attached to the corners of the floating dock.

(a) Glare appears right next to the helipad.

(b) Thresholded image with morphological operations applied.

Fig. 27: Original image and the result of thresholding. The glare and platform are directly adjacent.

5) Failure to track the Inner Circle: Two attempts failed due to inadequate tracking of the inner circle, stemming from its distortion in some frames. These distortions likely resulted from vibrations or UAV movement combined with the rolling shutter effect. The distorted appearance of the inner circle, alongside a higher Hough Circle Transformation threshold compared to the concentric circle detection, caused the system to fail in detecting the feature. The use of a higher threshold, as explained in subsection, was necessary but contributed to the detection issues.

Figure 29 illustrates the problem by comparing two frames taken 1/30 seconds apart. In subfigure 29b, distortion is evident, particularly on the inner circle, leading to a failure in detection. In contrast, subfigure 29c shows significantly less distortion, allowing the inner circle to be detected. Although no further improvements were tested over water due to time constraints, possible solutions to these image processing issues are discussed in outlook for future work.

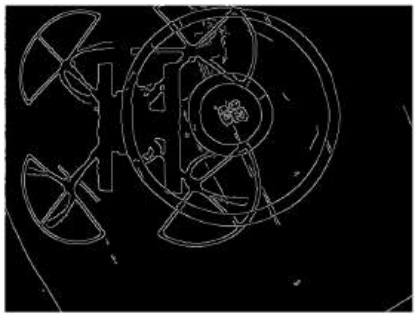
VI. Conclusion

This work introduces a vision and control system designed for autonomous UAV landing on a floating platform over water. A tracking algorithm was developed to accurately locate the target, and this relative position is utilized to control the UAV's movements toward the platform. In a simulated environment, tests showed promising results for landing on both tins and the inner concentric circle.

However, transitioning to real-world applications presented various challenges, such as overexposure and wind conditions that affected the reliability and accuracy of landings. After making some adjustments for land-based operations, the landing success rate improved, achieving an average accuracy of

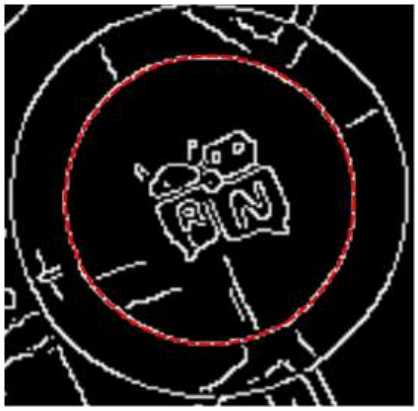
(a) Scaled-down testing. (b) Glare at 6 m. (c) Glare at 15 m.

Fig. 28: Glare seen at different altitudes.



(a) Captured frame with Canny Edge Detection Applied.

(b) Cropped image. Distortion is apparent w.r.t to the red dashed circle.



(c) Frame captured 1/30 s later. Much less distortion.

Fig. 29: Results showing tracking failure of inner circle.

16.23 cm for the inner concentric circle and 20.75 cm for tins at wind speeds of 1 m/s and 3 m/s, respectively. Conversely, the performance over water was significantly hindered, with only two successful landings out of twelve attempts primarily due to incorrectly estimated waypoints. Additionally, issues with image recognition arose, particularly because water reflections complicated the successful detection of the square target.

VII. Learning and Future Work

This research achieved several key milestones, particularly in integrating sensors, the flight controller, onboard computer, and camera. The vision-based tracking system worked effectively on land, allowing the UAV to land on both tins and the inner circle, except in conditions with excessive glare over water. Potential improvements are discussed to address the issues identified during testing.

One recommendation is to adjust the camera's exposure time or aperture settings to reduce glare, potentially improving the distinction between glare and the landing platform in the image histogram. This could help erode glare areas more efficiently. For better inner circle detection, lowering the Hough Circle Transformation threshold and incorporating a tracking system to select the most probable circle based on the UAV's previous location may enhance detection reliability. Another proposed solution is using a global shutter camera, which could reduce distortions caused by vibrations, though at the cost of lower resolution compared to rolling shutter cameras.

Additionally, using a wider lens (90° instead of 65°) would allow the concentric circle detection to work at lower altitudes, making the algorithm more robust and reducing the reliance on inner circle detection. This adjustment would allow for a smoother descent from 10 m to 6.37 m, improving the overall performance of the landing system without significantly increasing glare. These enhancements could lead to more robust image recognition and better UAV landing capabilities.

References

[1] A. S. Nair, P. A. Jeyanthy, L. Ramesh, G. M. Kurian, and S. R. Mohamed, "Autonomous precision landing with uav and auto charging," in 2021 International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT), 2021, pp. 463–466.

[2] A. Khazetdinov, A. Zakiev, T. Tsoy, M. Svinin, and E. Magid, "Embedded aruco: a novel approach for high precision uav landing," in 2021 International Siberian Conference on Control and Communications (SIBCON), 2021, pp. 1–6.
[3] D. Pieczynski, B. Ptak, M. Kraft, M. Piechocki, and P. Aszkowski, "A fast, lightweight deep learning vision pipeline for autonomous uav landing support with added robustness," Engineering Applications of Artificial Intelligence, vol. 131, p.107864, 2024. [Online]. Available: $https$: $//www.sciencedirect.com/science/article/pii/S0952197624000228$
[4] M. Moreira, F. Azevedo, A. Ferreira, D. Pedro, J. Matos-Carvalho, Á. Ramos, R. Loureiro, and L. Campos, "Precision landing for low-maintenance remote operations with uavs," Drones, vol. 5, no. 4, 2021. [Online]. Available: $https$: $//www.mdpi.com/2504 - 446X/5/4/103$
[5] C. Wang, J. Wang, C. Wei, Y. Zhu, D. Yin, and J. Li, "Vision-based deep reinforcement learning of uav-ugv collaborative landing policy using automatic curriculum," Drones, vol. 7, no. 11, 2023. [Online]. Available: $https$: $//www.mdpi.com/2504 - 446X/7/11/676$
[6] C. Castillo, A. Pyattaev, J. Villa, P. Masek, D. Moltchanov, and A. Ometov, Autonomous UAV Landing on a Moving Vessel: Localization Challenges and Implementation Framework, 09 2019, pp. 342–354
[7] ROS Documentation, "Robot operating system (ros)," accessed: 2024-06-21. [Online]. Available: $https$: $//www.ros.org/$
[8] N. Otsu, "A threshold selection method from gray-level histograms," IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62–66, 1979.
[9] OpenCV Documentation, "Tutorial: Image thresholding," accessed: 2024-07-10. [Online]. Available:$https$: $//docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html$
[10] "Erosion and dilation," accessed: 2024-05-21. [Online]. Available: $https$: $//docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html$
[11] J. A. Hartigan and P. M. Hartigan, "The Dip Test of Unimodality," The Annals of Statistics, vol. 13, no. 1, pp. 70 – 84, 1985. [Online]. Available: $https$: $//doi.org/10.1214/aos/1176346577$
[12] J. Canny, "A computational approach to edge detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679–698, 1986.
[13] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," Commun. ACM, vol. 15, no. 1, p. 11–15, jan 1972. [Online]. Available: $https$: $//doi.org/10.1145/361237.361242$
[14] D. Rieck, "Mechanical, rolling, and global shutter cameras for drones," 2021, accessed: 2024-06-21. [Online]. Available: $https$: $//medium.com/@douglasrieck/mechanical - rolling - and - global - shutter - cameras - for - drones - 78460b54e79d$
[15] Raspberry Pi Foundation, "Raspberry pi global shutter camera," accessed: 2024-07-03. [Online]. Available: $https$: $//www.raspberrypi.com/products/raspberry - pi - global - shutter - camera/$