NYUAD Robosub: Automated Underwater Vehicle Adaptation and Control

Minh Quan Nham, Rami Richani Hamdan, Kirubel Solomon Tesfaye, Aira Khaliq, Sherifa Yakubu, Julio Zuazola, Lukelo Luoga, Obed Morrison, Baoyuan Zhang, Aya El Mir, Ibrahim Nayfeh

***Abstract* - NYUAD Robosub is a team of undergraduate students at NYU Abu Dhabi developing a fully autonomous underwater vehicle to compete in Robosub for the first time. Besides a detailed review and analysis of past competition documentation, our team focused on creating simple but modular and effective subsystems that contribute to the overall realization of the robot in a short time. From mechanical design to electrical and software development, we chose and also created products that can enable our AUV to complete its missions. This paper will highlight our strategies, system architecture, and designs in each subsystem that were essential for us to have a complete and fully autonomous vehicle. In addition, it will underscore the developments, simulations, and experiments that we have done that will allow us to complete the tasks in the competition.**

I. Competition Strategy

Since it is our team's inaugural competition and we are competing against much more sophisticated teams with years of experience, it became our primary priority to research past reports thoroughly and take notes of previous teams' strategies, technology, and drawbacks to form our knowledge background. Our key goal was modularity: in our small team, each person was assigned a specific subsection pertaining to their strong points, allowing each of us the ability to deeply research & develop in our respective categories. The modularity goal has been crucial to allow seamless integration, and for our work to be eventually assembled into a working robot.

As we also started late into the spring semester, our focus was on working with off-the-shelf components and software, only developing new concepts whenever necessary. Our literature review of past reports helped greatly in giving us a head start on available technology, but we also considered a large selection of commercially available solutions to find those that best fit our needs. With our current development, we have planned to attempt all the tasks in the competition. We determined that object detection, high-level control of the robot, and acoustic pinger detection as some of the focal points needed to guide us and complete the tasks. Initially, we spent a lot of time figuring out how the general architecture of our sub-systems should look like, which algorithms and open-source software to use such as for vision, acoustics, and for achieving a fully autonomous run. While doing this, we identified products out on the market that best fit our interest, which ones could be modified, and which ones needed to be made from scratch, such as the torpedo. Throughout the development process, we have been using simulations to test for different scenarios and optimization before testing in the pool.

II. Design Creativity

Our team was mainly divided into three sub-teams (mechanical, electrical, and software) with each sub-team also having subdivisions. Each sub-team has come up with simple and effective plans that would make our complete system fully functional and ready to complete the tasks.

A. Mechanical Subsystem

1) Frame: We decided to work with the BlueROV2 as our base robot as it is one of the most successful commercial ROVs to date. Due to our late entry into the competition, it also helped us move beyond designing the frames and mechanical

layout, which normally takes up a lot of time. The ballast system also helped us in trimming the vehicle with the newly added components.

2) Central Enclosure: With our camera and computational units (Jetson TX2, Raspberry Pi) running at full speed heat became a serious issue. We decided to substitute the traditional transparent container for an aluminum one as a huge heatsink, taking advantage of the ambient water temperature to cool the system. Extra fans were also fitted in the chamber to improve convection.

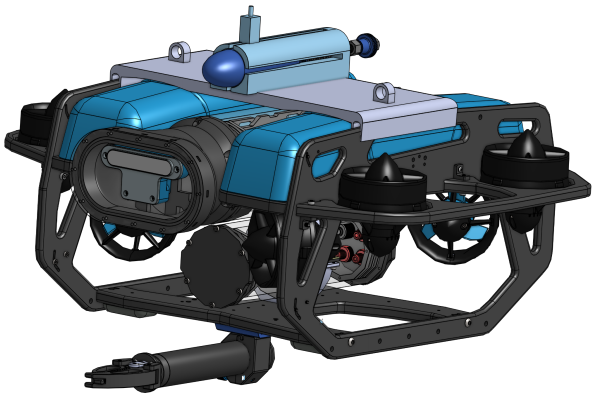

Fig. 1. 3D model of the AUV

3) Camera Housing: We identified early on that the benefits of having a stereo camera exceed its drawbacks, and focused on finding a watertight housing for the camera. Existing off-the-shelf housing solutions, however, were too large and clunky for the BlueROV2. We made use of the traditional oblong shape of stereo cameras to develop a racetrack-shaped transparent window, fitted with a custom O-ring to make it watertight. The custom housing comes with a disadvantage in the field of view, as a curved surface would allow a wider range for the cameras. However, the ease of manufacturing a flat plate instead of transparent domes exceeds its drawbacks. The housing also allowed some more space for the Jetson and additional components to be fitted.

4) Torpedo Launcher: Our torpedo launcher had to be designed from scratch. Before starting, a simple simulator was built on Geogebra to test the best conditions under which the model could be planned to be refined later. The movement source

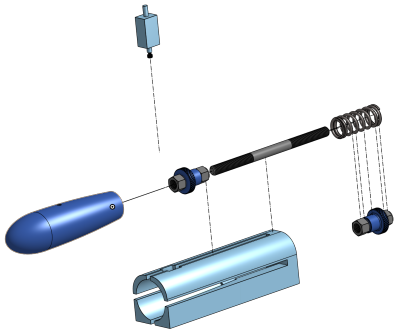

Fig. 2. Exploded-view model of torpedo in launcher

was chosen to be a spring-loaded mechanism activated by a solenoid rather than using a servo motor to reduce weight and the energy needed to activate it. The torpedo itself was projected to have a streamlined exterior, an internal chamber that allowed for a density balance using a water chamber, and a nylon screw to control its size.

B. Software Subsystem

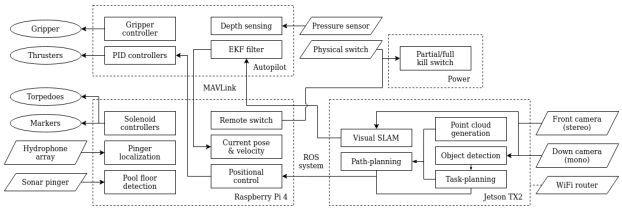

Fig. 3. Overall system architecture

1) ROS: The goal of modularity was met chiefly by our use of the Robot Operating System, which allowed submodules to be connected in a context-agnostic manner. This aspect was essential as we experiment with different camera types and software-in-the-loop schemes. Task planning, controls, and pinger echolocation were implemented using ROS nodes and can be started all within a launch file.

2) Autopilot: In order to avoid unnecessary reimplementation of Kalman filters and PID controllers in ROS, we developed on the ArduSub system as it incorporates both the traditional tools as well as support for underwater-specific components, such as the pressure sensor. Running the EKF filter and PID position-velocity controllers on the Pixhawk also helped alleviate the computational burden put on the Raspberry Pi and

the Jetson, allowing them to focus on other tasks. The ArduSub system also has a great advantage in that it supports the BlueROV2 natively, reducing the need for us to customize and debug a new frame.

To work efficiently with the ArduSub system, instead of using the inflexible mavros package that did not allow much customization and optimization for our tasks, we chose to use the pymavlink library to write custom Python ROS nodes that work with both ROS and MAVLink. Although having a steep learning curve, this allowed much better control over the information sent by ArduSub as well as the messages we send to the system. A ROS publisher was written to send current pose information of the Pixhawk as a relative frame to the world, and the control nodes relied on this to form occupancy grids, send precise location control, and transform objects seen by the camera into the world frame.

3) SLAM: An issue we found with the ArduSub system was that there was no precise control of the position without extra add-ons like a GPS (incompatible with our competition setting) or a DVL (exceeding our price range). The newest ArduSub version, however, allowed us to send a custom velocity stream to the EKF filter via MAVLink; this led us to implement localization algorithms and send changes of pose data to the Pixhawk. The RoboSub competition setting this year was a large factor in our decision to use visual and inertial SLAM; a clear pool with marked walls would allow much better feature detection than a murky, deep open-water pool. The specific SLAM algorithm chosen was ORB-SLAM 3, which provided visual-inertial SLAM with loop closure while providing global optimization and extensive library support. [2]

4) Controls: With the aforementioned SLAM in place, ArduSub enabled precise position control. We made sure to only move the ROV only a small distance at a time to avoid losing track, which would cause detriment both to the SLAM algorithm as well as the ability of the Pixhawk to generate local position data. After it is in place, the control is simply commanding the Pixhawk to go to specific locations. Tuning is done mostly in simulation, but the provided PID gains for the BlueROV2 are already close enough to meet our needs.

5) Path Planning: We evaluated many modern obstacle-avoidance algorithms for path planning of the ROV, and any-angle algorithms showed great promise in their efficiency and adaptability to 3D space. RRT-based algorithms are not very well-optimized for 3D spaces and incremental path planning, and subgoal-graph algorithms are nearly impossible to generalize to 3D. We also looked into A*-based algorithms such as Block and Field A*, but they are both slow and inefficient in 3D space. We settled on Lazy Theta* [1], an algorithm descendant of A* that is very fast in 3D path planning. For the algorithm, space was partitioned into a 26-neighbors occupancy grid, and we used an unordered set of 3D integer vectors to increase speed and memory efficiency.

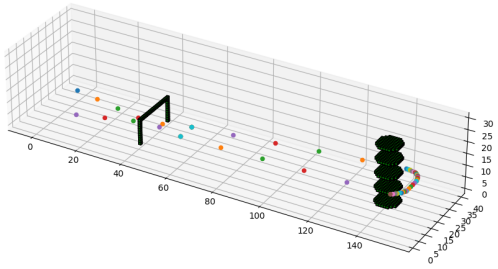

Fig. 4. Lazy Theta* navigation around the prequalification task

6) Computer Vision: We evaluated a diverse range of underwater computer vision architectures in an attempt to curate a vision system capable of recognizing obstacles despite underwater image degradation. Due to image distortion underwater, conventional OpenCV techniques were insufficient in detecting objects. Therefore, we settled on using machine learning algorithms that were compatible with the OAK-D camera. We heavily relied on DepthAI's documentation for fulfilling the aforementioned details. After comparing the efficacy of YoloV3, YoloV4 tiny, and MobileNetV2 SSD [3][4], our team finalized MobileNetV2 SSD for its speed in real-time object detection situations. We also used neural inference in conjunction with a depth map to determine object coordinates in 3D space. Therefore, our computer vision architecture

was not only able to detect and draw bounding boxes around objects of interest but also undertook their 3D localization.

C. Electrical Subsystem

1) Computational Units: With the powerful OAK cameras, we decided that the Jetson TX2 is a good tradeoff between power consumption and computing performance for our needs. We added a Raspberry Pi 4 to help with fast audio analysis and miscellaneous tasks, while also incorporating additional, application-specific boards as needed.

2) Power: We used a single battery to power up the whole system. We decided to fit it into a second water-tight tube at the bottom, which would be connected to the main tube where all the other components are found. We used regulators to supply the required voltage and current to each component. After much deliberation, we decided to use a MOSFET with a lanyard switch as a kill switch instead of using switches fitted to the AUV, as they would require very close-range action if the operators wanted to switch the AUV off.

3) Hydrophones: The hydrophones were placed in a static non-planar array and connected to a quad-channeled analog-to-digital converter attached to an evaluation board. The board would send the data to a processor, which would then communicate with the Jetson and provide the necessary information to properly navigate.

4) Communication Network: We integrated ethernet and wifi to our system that were used not only for completing the tasks but also modified for testing purposes. As we have different devices, such as the Jetson TX2, Raspberry Pi 4, and Pixhawk, that need to run on the same ROS master, we used ethernet to set them to the same network. An ethernet splitter with four ports is used to connect the wifi, Jetson, and Raspberry Pi. The other port is occasionally used by the Fathom-X tether interface board during testing procedures, for connecting another device - most often a laptop outside the water - to the ROS master. This gave us access to the Jetson via SSH to monitor and control the ROS master. After testing, we used the wifi hotspot to communicate with the system whenever the ROV surfaced from the pool.

5) Sensors: The Pixhawk Cube autopilot provided us with three sets of IMU sensors that are fused giving us enhanced accuracy with the redundancy. Besides the IMU, we added a pressure sensor and sonar ping for detecting the depth and the bottom of the pool respectively. From the pressure sensors available, we used the Bar30 pressure sensor since it was supported by the ArduSub firmware. We decided to use a sonar pinger to detect the bottom of the pool and avoid a collision. For this, we created a ROS package that can read and publish the depth and confidence rating of the depth value as ROS topics, which can be used for further applications.

III. Experimental Results

Testing our ideas and designs has been an important step to finalize our AUV. We have focused on testing each of the subsystems individually before testing the completed AUV underwater. Simulations were continuously utilized to assess and improve our designs. The last months before the competition have been dedicated to pool tests for us to complete our vehicle.

A. Mechanical Subsystem

The BlueROV2 has been used in previous competitions, and its main components (enclosures, ballasts, and turbines) showed promising results. The turbines added to its upgrade have been tested individually and worked similarly to the ones previously used. The torpedo design has been tested and the exterior design was hydrodynamic, even with the uneven surface created while 3d printing. However, the density adjustment revealed to require precise control during the design process and printing, and the fine-tuning to make the torpedo neutrally buoyant was only possible after careful balancing. The solenoid revealed itself waterproof, but the locking mechanism is still under testing. Once it is properly set, the marker dropper can be made with a smaller torpedo that has the same aerodynamic front with a thinner and more dense design.

B. Software Subsystem

1) Blender: For a machine learning model to work successfully, the dataset must be accurate and meaningful. To this end, we considered a variety of simulation techniques - initially, we even considered taking the pictures manually inside the school pool. However, due to time and financial constraints, Blender became the optimal choice. Since Blender could facilitate the use of caustics and diverse lighting environments, the renderings obtained were varied and mimic reality closely. A rendering of our pre-qualification gate being detected is shown below.

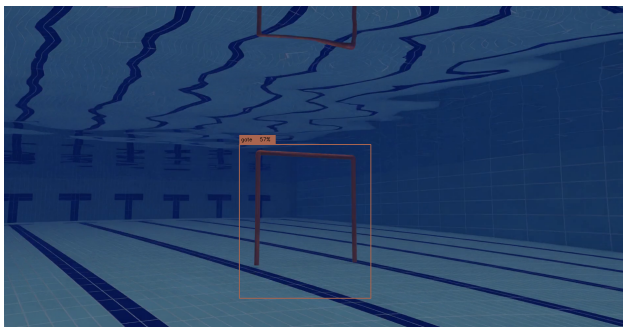

Fig. 5. Rendering in Blender and detection of gate using MobileNet

2) Gazebo: We have also identified early on that a reliable simulation environment is fundamental in developing our vision and control algorithms, as developing the BlueROV2 while submerged underwater is highly inefficient and cumbersome. We utilized Gazebo with the UUV Simulator [5] plugin for the simulation, as it has been a proven platform used with success by multiple teams in the past. With a fusion of manufacturer-supplied integration of ArduSub SITL with Gazebo [6] and a Github repository implementing UUV Simulator capabilities for the BlueROV2 [7], we produced a resilient, physics-based simulation environment to test our control and vision algorithms.

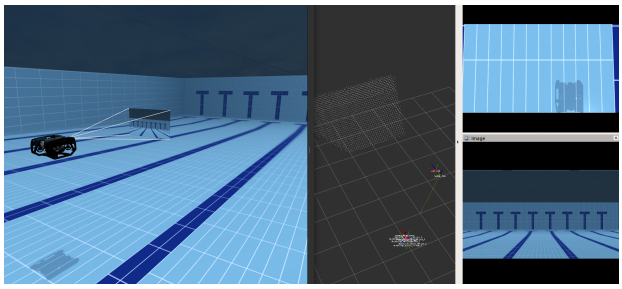

Fig. 6. BlueROV2 in the simulated environment with filtered camera point cloud, downward and forward-facing camera streams

C. Electrical Subsystem

We have tested the components that need to be powered by the battery and the connections across the components. All the components, such as the thrusters, the Fathom tether interface, and the Pixhawk are working properly. We have also set up the network interface, such as among the Jetson, wifi, Fathom-X tether, and external laptop for monitoring purposes in testing. We were successful in running ROS using multiple devices on the same ROS master. The pressure sensor is integrated into the autopilot through I2C and the sonar ping is also publishing its data for further use.

IV. Acknowledgments

NYUAD Robosub team would like to thank everyone who supported us through our journey. We would like to express our gratitude to our university NYU Abu Dhabi, especially the Engineering Design Studio, for providing us with the necessary financial and material support for building and testing our robot. We appreciate NYUAD pool lifeguards for their assistance during our pool tests. We would also like to offer our deepest gratitude to our advisor Matthew Karau for his guidance and technical support throughout our journey. Last but not least, we want to thank RoboNation for providing us with this opportunity to explore our interests and develop our knowledge in robotics.

REFERENCES

[1] A. Nash, S. Koenig, and C. Tovey, "Lazy Theta*: Any-Angle Path Planning and Path Length Analysis in 3D," *AAAI-10 24th Conference on Artificial Intelligence*, 2010. [Online]. Available: http://idm-lab.org/bib/abstracts/papers/aaai10b.pdf. [Accessed: 12-Jun-2022].

[2] C. Campos, R. Elvira, J. J. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[3] P. Adarsh, P. Rathi, and M. Kumar, "Yolo v3-Tiny: Object Detection and recognition using one stage improved model," *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Mar. 2020.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018.

[5] M. M. Manhaes, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV Simulator: A Gazebo-Based Package for Underwater Intervention and Multi-Robot Simulation," *OCEANS 2016 MTS/IEEE Monterey*, 2016.

[6] P. Pereira, "Scripts to help Bluerov integration with ROS," *GitHub*, 30-Jan-2018. [Online]. Available: https://github.com/patrickelectric/bluerov_ros_playground. [Accessed: 12-Jun-2022].

[7] FletcherFT, "Scripts to help bluerov2 integration with ROS and UUV Simulator," *GitHub*, 17-May-2021. [Online]. Available: https://github.com/FletcherFT/bluerov2. [Accessed: 12-Jun-2022].

Appendix A - Component specifications

Component	Vendor	Model/Type	Specs	Custom/ Purchased	Cost	Year of Purchase
Buoyancy Control	Blue Robotics	BlueROV2 Heavy Machined Buoyancy Foam x2	https://bluerobotics.com/store/rov/bluerov2-components-spares/float-r3318-brov2-heavy-r1-rp/	Purchased	$100	2022
Frame	Blue Robotics	BlueROV2	https://bluerobotics.com/wp-content/uploads/2020/02/br_bluerov2_datasheet_rev6.pdf	Custom (Significantly modified)	$3,950	2017
Additional Frame	Blue Robotics	BlueROV2 Heavy R2	8 turbines, 6 DOF movements	Purchased	$740	2022
Waterproof Housing	Blue Robotics	3’’ enclosure 4’’ enclosure	https://bluerobotics.com/store/watertight-enclosures/wte-vp/#tube	Purchased	$200	2022
Waterproof Connectors	Blue Robotics	Penetrators x4	https://bluerobotics.com/store/cables-connectors/penetrators/penetrator-vp/	Purchased	$36	2022
Gripper	Blue Robotics	Newton Subsea Gripper	6.2 cm aperture	Purchased	$590	2022
Torpedo Launcher	-	-	Max speed 3m/s	Custom	-	2022
Dropper	-	-	-	Custom	-	2022
Sonar	Blue Robotics	Ping Sonar	https://bluerobotics.com/store/watertight-enclosures/buoyancy/float-r1/	Purchased	$360	2022
Pressure Sensor	Blue Robotics	Bar30 High-Resolution 300m Depth/Pressure Sensor	30 Bar (300m depth) with a depth resolution of 2mm	Purchased	$85	2022
Thrusters	Blue Robotics	T200 Thrusters x8	https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t20	Purchased	Included in Frame + Additional	2017

			0-thruster-r2-rp/		frame	
Motor Control	Blue Robotics	Basic ESC	https://bluerobotics.com/store/thrusters/speed-controllers/besc30-r3/	Purchased	Included in Frame + Additional frame	2017
High Level Control	HEX	Pixhawk 2 UAV Autopilot Flight Controller	https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk-2.html	Purchased	$260	2022
Propellers	Blue Robotics	T200 propellers	Diameter: 3”	-	Included in Frame + Additional frame	2022
Battery	Blue Robotics	LiPo Battery	10000 mAh 14.8 V LiPo	Purchased	$170	2017
Regulator	Blue Robotics	5V 6A Power Supply	https://bluerobotics.com/store/comm-control-power/control/bec-5v6a-r1/	Purchased	$25	2022
CPU	NVIDIA	Jetson TX2 Module	https://developer.nvidia.com/embedded/jetson-tx2	Purchased	$500	2022
Carrier board	AUVIDEA	J120 carrier board for the NVIDIA Jetson TX1/TX2	https://auvidea.eu/download/manual/J120/J120_technical_reference_1.6.pdf	Purchased	$300	2022
Companion Computer	Raspberry Pi	Raspberry Pi 4	https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/	Purchased	$160	2022
Internal Comm Network	BotBlox	GigaBlox + Nano PicoConn	https://botblox.io/products/gigablox-small-gigabit-switch	Purchased	$125	2022
External Comm Interface	Blue Robotics	Fathom-X interface board set	https://bluerobotics.com/store/comm-control-power/tether-interface/fathom-x-r1/	-	Included in Frame	2017
Acoustic	Aquarian	Aquarian H1C hydrophone	https://www.aquarianaudio.com/AqAudDocs/H1c%20manual.pdf	Purchased	$600	2022

Front Camera	Luxonis	OAK-D	https://github.com/luxonis/depthai-hardware/blob/master/BW1098OAK_USB3C/Datasheet/OAK-D_Datasheet.pdf	Purchased	$200	2022
Inertial Measure-ment Unit (IMU)	HEX	Pixhawk 2 UAV Autopilot Flight Controller	InvenSense MPU9250, ICM20948 and/or ICM20648	Purchased	$260	2022
Doppler Velocity Log (DVL)	-	-	-	-	-	-
Algorithms: Vision	-	MobileNetV2, YoloV3-tiny	-	-	-	-
Algorithms: Localization and Mapping	-	ORB-SLAM 3	-	-	-	-
Algorithms: Autonomy	ROS Melodic	SMACH	-	-	-	-
Algorithms: Path Finding	-	Lazy Theta*	-	-	-	-
Open-Source Software	-	ROS Melodic, Blender	-	-	-	-