Robosub 2021 Technical Report

Underwater Robotics at Berkeley
communications@urobotics.berkeley.edu

***Abstract*—Underwater Robotics at Berkeley presents the *Bearacuda*, our latest AUV designed for the 2021 Robosub competition. The *Bearacuda* features several major innovations to improve its performance, including a custom-built carbon fiber chassis, two manipulator arms, and an adaptable perception system. Our mechanical engineers used CAD software to optimize the AUV's shape and components to minimize drag and complete mission tasks effectively. CAD models and analyses allowed team members to tailor the *Bearacuda* for its mission while working remotely. An all-new approach to perception also maximizes the AUV's efficiency: the AUV can actively choose between its perception algorithms depending on which is most effective under the given conditions. Additionally, we implemented a cascaded PID system for our controllers, and tested our entire software loop in Gazebo simulation to ensure our software stack was robust. These innovations consolidate to form an AUV prepared to excel in the 2021 competition.**

I. Competition Strategy

Our AUV was designed to complete all of the competition tasks. While this required a wider range of functions, the benefit of higher point potential outweighed the risk of complexity. We emphasized simplicity and efficiency throughout the design process, making parts as compact and straightforward as possible to maximize reliability.

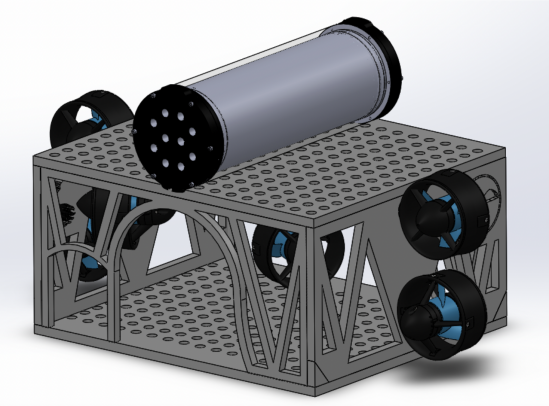
We designed the AUV with originality as a central focus, creating custom designs for as many features as possible (excluding foundational parts like the motors and battery). Customization of key components such as the chassis, gripper, and torpedo launcher allowed us to continuously iterate on our designs throughout the year in CAD software and tailor each detail to the task at hand.

The team is divided into three departments: hardware, software, and operations. The hardware department includes mechanical and electrical engineers who planned and constructed the AUV frame, payload tools, and electrical components. The software department conducted simulations and enabled the AUV to sense and respond to its environment. The operations department led recruitment efforts, organized finances, and managed public relations. Each department met weekly to delegate tasks and exchange feedback on progress, and all of the departments convened together for a separate weekly meeting to stay informed and integrate the AUV system. With this organization, each department independently focused on product development while maintaining interdepartmental communication, leading to an improved and well-unified vehicle.

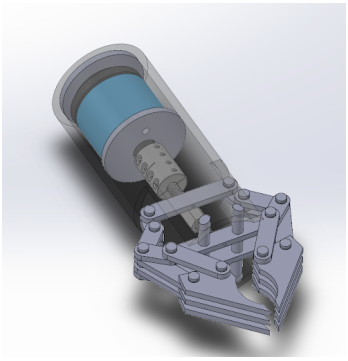
Our engineers adopted a large-to-small scale approach to construction, starting with general tasks like frame design before honing in on payload tools and mission-specific features like the torpedo launcher. The first several months of production were dedicated to creating online 3D CAD models of the ROV and payload tools to evaluate effectiveness and feasibility of the product before manufacturing. Because of the COVID-19 pandemic, we spent much of the year perfecting these 3D designs and conducting computerized analyses such as stress tests, as well as carrying out control simulations of the full AUV; this collective focus on remote design and simulations optimized our ability to assess the vehicle without in-person testing. As we approached the competition deadline, members received parts to work with remotely for construction and experimentation.

II. Design Creativity

A. Hardware



*1) **Mechanical**:* The Robosub's chassis consists of custom designed high-density polyethylene components, with cut-outs for the main electrical housing, the battery compartment, the manipulator, and the torpedo launcher. The chassis was made to fit around the central electrical components, with design focuses being mounts for the stereovision camera setup and 8 Blue Robotics T200 vertical and horizontal thrusters. The truss design style was meant for maximum stability while maintaining good manufacturability, and the large plates have holes for water flow and modularity. The front-side electronics plates were added later on in the engineering process to stabilize and reinforce the frame, and still feature the truss design. There are spots for the exterior thrusters on the main side components, and have two thrusters per side. The battery enclosure on top of the frame adds to the buoyancy of the design. Brackets to attach components together are planned to be 3D printed.



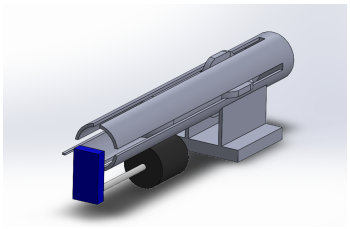
Our gripper is a relatively simple design; it is powered by a threaded rod drive which converts the rotary motion of a Blue Robotics M200 motor into linear motion, causing an assembly of aluminum plates to open and close, actuating the claw.

In more detail, a threaded rod is attached to the output shaft of the M200 motor via a long shaft collar. This rod is then threaded through a nut, which is attached to a thin machined aluminum plate via a 3D printed plate. This plate is attached to a series of other machined plates via a series of non-threaded shafts, which are held in place via either shaft collars or snap rings. The specific configuration of plates results in the open-close actuation of the claw when the threaded rod is spun by the motor. Instead of being machined out of aluminum, the plates may also be 3D printed; however, this will result in a much weaker claw that is more prone to breakage, and is not recommended. The entire assembly is housed in a 1.9" OD (1.5" ID) PVC pipe, and is held in place using plates and non-moving shafts.

Various compromises and trade-offs were made in this design. First, the decision to house it inside a relatively small PVC pipe was made in the interest of ensuring it could fit on the AUV and be attached with relative ease, as well as detached easily for servicing. However, it led to many trade-offs in regards to the internal components which could be used; only parts less than 1.5" in two perpendicular axes of their dimensions could fit inside the PVC pipe. Thus, many design considerations, such as the use of a worm gear system or a planetary gear system were discarded due to their space requirements.

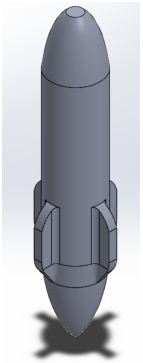
In addition, compromises were made in regards to cost. Underwater Robotics @ Berkeley is a student club, reliant on university funding; as a result, we are unable to procure extremely expensive materials or source extremely expensive parts. As a result, decisions were made to greatly simplify the interior from its first design. In prototype designs, the interior used a system of spur gears to reduce the gear ratio; this was removed because the gears included in the design were prohibitively expensive, and replaced with a screw drive.

Finally, compromises were made in regards to safety. The Blue Robotics M200 motor was selected for use because of its ability to operate underwater, despite its lack of precise position control. While other motors such as precise position-control stepper motors would have been more desirable in this role, they are not waterproof and would have required extensive, careful waterproofing of the claw. This would be difficult and expensive to completely achieve; in addition, in case of a failure, the entire motor would fail, which is both dangerous and expensive. Another trade-off made with regards to safety was the shift away from a gear-driven claw design to a plate-driven one. The original design was somewhat easier to machine since it included far less aluminum plates; however, it made use of 3D printed parts in critical load-bearing areas, which would have led to potential failure. Thus, the decision was made to move to a design using metal in the critical places, improving strength.



The torpedo launcher is designed to accurately launch a torpedo while maintaining modularity and compactness, allowing for flexible mounting on the AUV. Due to COVID limited testing has been conducted, so inspiration for the torpedo design was taken from military torpedoes. The launching system is spring powered and utilizes pre-owned motors to launch. Though simple, the launching system has very few areas for error.

The torpedo has been designed to emulate military torpedoes, since the COVID-19 pandemic has greatly impacted our ability to manufacture and prototype designs. There are four fins on the torpedo for horizontal and vertical stability, helping it launch straight. The torpedo is neutrally buoyant, decreasing the buoyant force and increasing accuracy. The curvature of the torpedo body is designed with hydrodynamics in mind, limiting the wake left behind as the torpedo travels through the water.



All our design choices maximize the torpedo's ability to shoot as far and straight as possible. The spring-powered launcher provides a reliable and cost effective source of force to eject the torpedo. To initiate a launch, the tab that holds the torpedo in place is rotated, allowing the spring to extend and launch the torpedo. Although the launcher's design is straightforward, it is compact, cost effective, and easily mountable anywhere on the AUV.

*2) **Electrical**:* Learning from the experiences of developing our past vehicles, we strove to create an architecture for our electrical system that was unified, modular, and repairable. In order to facilitate quickly adding and removing components such as thrusters, end effectors, and sensors, we chose a standard set of signals that would serve as the bus connecting all of them together. Reviewing our use cases, we chose to provide 12V, 5V, and CAN bus connections. The CAN bus is widely used in the automotive industry as a noise-resistant, reasonably high-speed interconnect which is why we chose it for our architecture. With one bus connecting everything, we greatly reduced the amount of cabling used and even enabled daisy chaining modules together. Of course, there are certain components with requirements vastly exceeding the capabilities of our bus, such as the high current draw of our thrusters or the high speed communication of our cameras. In those cases, we had dedicated connections to meet those

needs.

To maximize repairability and modularity, we split our electrical system across three enclosures. The main tube is flanked by both the battery tube and camera tube. The main tube contains the entirety of the computation and motor drivers. It has a single backplane with connectors for the various devices we have. Cables run out to our cameras, battery, end effectors, and each of our thrusters. One side of the tube has what we call a "sideplane" to facilitate removing the entire backplane assembly for repairs. The battery and camera tubes are simpler in construction and just have mechanical mounts for their respective components.

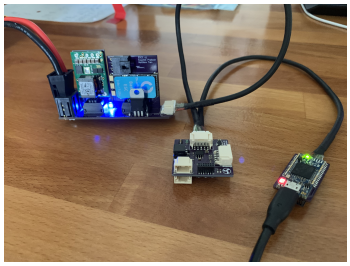


Fig. 1: Daisy chain test

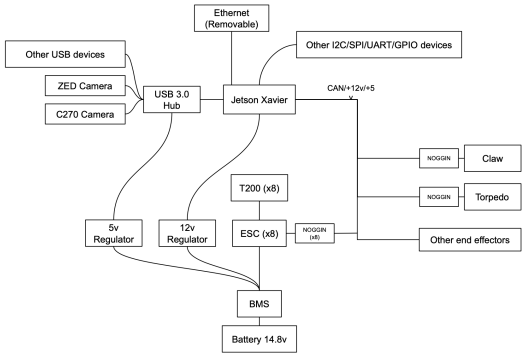


Fig. 2: Electrical connection diagram

For firmware development, we chose the Mbed platform due to its ease of use, feature set, and community support. We chose Mbed Studio as our IDE to make it easier for new members to learn. Getting Git working with Mbed Studio so we could do version control on our firmware took some tricks with symlinks, but at the end we had a monorepo to easily distribute and develop firmware with.

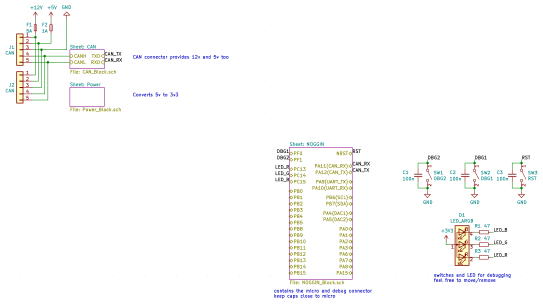


Fig. 3: Template schematic

For hardware development, we developed the NOGGIN system which consists of both a general purpose board that is good enough for around 80% of our tasks and a KiCad template that people can build on top of. The general purpose board contains a brushed motor driver, CAN bus connections, current and voltage sensing for debugging/telemetry purposes, and I2C, UART, DAC, and PWM breakouts. The main idea of NOGGIN was to shift focus away from the details of microcontroller supporting circuitry over to the actual task the board is accomplishing, whether it be reading from a sensor or controlling a motor.

For integration of our electrical system, we chose a standard set of connectors for various applications. DuraClik connectors are adequate and compact enough for all low current tasks. Molex Ultra-Fit is good for medium current use cases such as motors. Molex Super Sabre is our battery connector due to its high current handling. It was difficult finding a small enough connector for our board-to-board connections such as those on the backplane, but we eventually settled on the new Samtec mPOWER connectors which can handle a surprising amount of current in a small package. Unifying our connectors meant that we didn't waste time deciding on and buying different connectors for each part of our system.

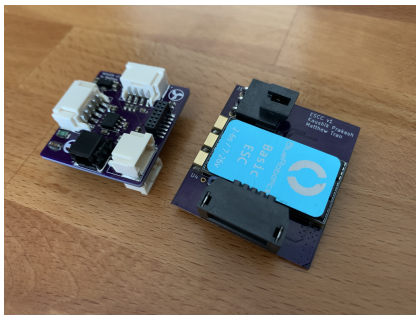


Fig. 4: Two NOGGIN boards

B. *Software*

At the start of the school year, we wanted to have a software system that is easily adaptable to different robot hardware, and also be robustly tested in simulation. For this reason, we decided to upgrade from our old ROS 1 system to ROS 2, which was becoming the predominant robotics middleware. We used an NVIDIA Jetson AGX Xavier as our main compute with a ZED 1 stereo camera as our front camera, a Logitech C920 as our bottom camera, and a Adafruit 9-DoF IMU fusion sensor.

1) ***Controls****:* Our main sensor is a 9-DoF IMU that gives us accelerometer, magnetometer, and gyroscope readings. Using the measurements, we have 2 separate PID systems: one for velocity control and one for orientation control. For going to a desired velocity, we utilize acceleration-based control through a PID controller. We provide a desired velocity vector in the x, y, z directions and the PID outputs the desired acceleration of the sub. Then, we use the moment of inertia matrix to go from body acceleration to body force, and a precomputed Jacobian transpose matrix to go from the desired body force to the individual thrusts of each motor. These motor thrusts are then converted to a PWM signal through linear interpolation and sent to the motor controllers.

The orientation controller behaves similarly and takes a desired orientation parameterized by yaw, pitch, and roll axes

as the input and processed this input through a set of cascaded PID controllers. In this case, we empirically tuned gains of the first PID, which took in a desired orientation and output a desired angular velocity. We specifically tuned for stable rotation along the yaw axis, as we will almost always want our roll and pitch to be at 0 radians to stabilize the sub. The second PID takes angular velocity and converts it to angular acceleration.

To perform route planning for the course, we used a behavior tree provided by the BehaviorTreeCPP library. We used a behavior tree instead of a state machine as behavior trees allow for a simpler graphical structure which allow reuse of components. Each behavior tree node had a corresponding ROS node that it would use to send requests, as well as an action server or service that would process these requests and return the result to the node. Our demo behavior tree to go through the gate is shown below.

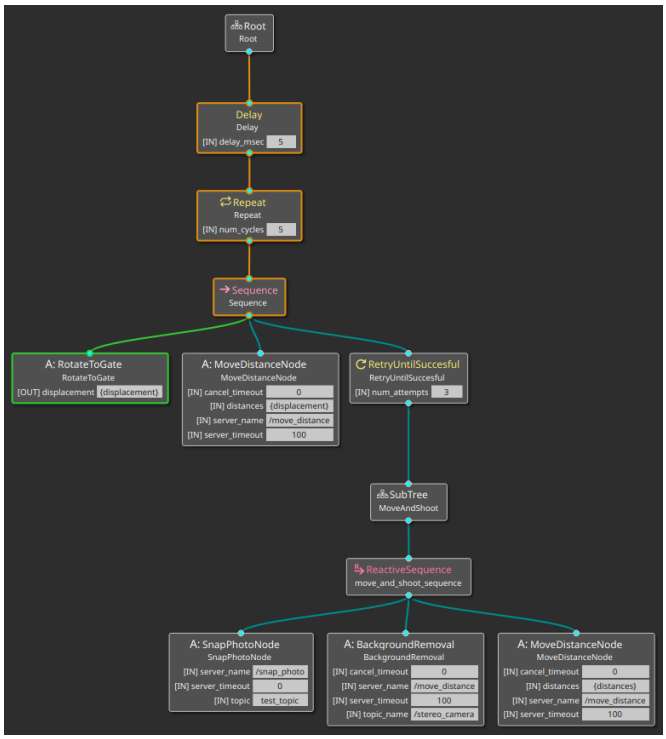


Fig. 5: Behavior tree for going through the gate

2) ***Perception****:* In general, our algorithms for identifying objects of interest (such as the gate with two posts) in the input video feed from the sub involved thresholding out the background water. Usually, the threshold was based on color, i.e. a certain shade of blue. This would isolate the object and simplify the process of identifying its location based on the contours of the thresholded image. This method, while somewhat effective for each task individually, wasn't very reliable and scalable since we would manually hard code the color threshold for each task. In addition, thresholding by color often wasn't enough to get a stable contour of the object of interest, and thus noise was a glaring issue.

Fig. 6: Gate task which involves detecting two posts and maneuvering in between them. This is our naive color thresholding, which introduces noise and breaks down in various light levels.

Our first approach to detect tasks distinguished by specific colors was analyzing histograms of various color spaces and utilizing the locations of the peaks in the distribution. The implementation involved changing the original RGB color space into other color spaces such as LAB and then running Otsu's binarization algorithm to threshold the desired peak in the distribution. By selecting the appropriate peak, the correct color would be chosen. This approach relied on the shape of the distribution and was more robust than fixed color thresholding.

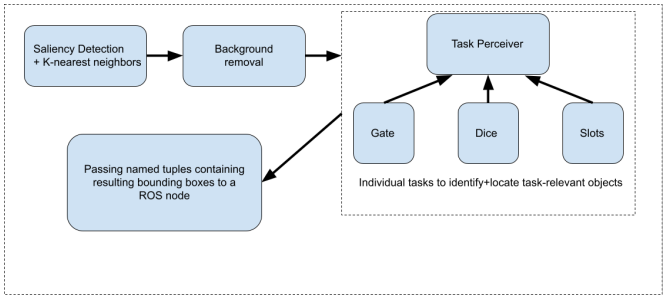


Fig. 7: Code pipeline that shows each step of the process from a raw video feed to resulting bounding boxes around task-relevant objects.

Accurate and efficient background removal is instrumental in our vision pipeline. To negate the downfalls of using one algorithm which only excelled in certain situations, we designed an intelligent switching methodology which selects which of two different algorithms to use. To determine when to switch algorithms, we analyzed contour centroid stability and contour area. Contour centroid stability can be defined as the tendency for the centroid of a contour to move. Generally, the centroid should be stable, as rapidly translating contours mean that a given algorithm does not focus on a specific object or feature in the frame. In general, the contour area cannot cover significant portions of the image (e.g. the whole frame), as it would not provide any useful information about particular objects in the image. One of the two algorithms used is a K-Nearest Neighbors algorithm, as described in [1] and implemented in OpenCV. This algorithm tends to get full detection of objects and is more efficient. The other algorithm is an optimized implementation of "Minimum Barrier Saliency Detection" in [2], which tends to be robust to noise, but is more

computationally expensive.

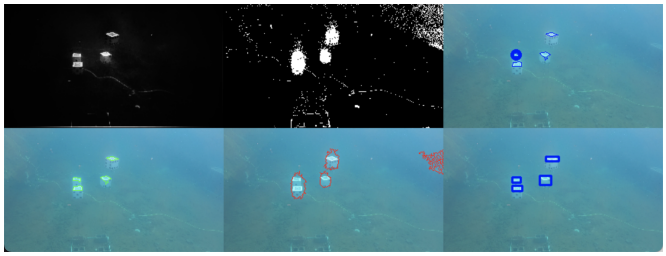


Fig. 8: This showcases the different components of our much more advanced combined background removal algorithm. Top left image shows the output of the Minimum Barrier Saliency Detection algorithm, while the bottom left image shows the extracted contours. The top middle image showcases the output of the K-Nearest Neighbors algorithm, and the bottom middle image shows the output contours from that. The top right image shows the contours of the algorithm that is currently selected, and the bottom right image demonstrates a bounding box representation of the output.

Switching between two distinct algorithms can produce vastly different results. A rectangle point averaging scheme is implemented to create a gradual transition between potentially vastly different outputs. In practice, this averaging scheme will also help mitigate the effects of noise that arrive from the two algorithms described above, as outlier artifacts are considered less.

III. Experimental Results

To test our software system, we utilized the Gazebo simulator with the underwater vehicle library UUV Simulator [3] and its ROS 2 port Plankton.

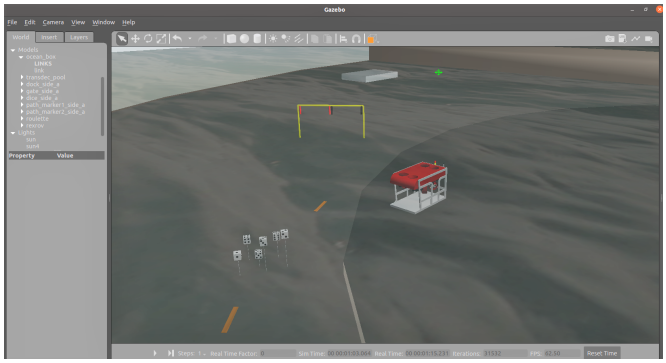


Fig. 9: Image from our Gazebo simulator showing the test sub, Robosub pool, and prior game elements

We developed our controller algorithms in ROS 2 and interfaced them with the existing Plankton codebase to actuate the Gazebo ROS controllers. We additionally added a copy of our ZED camera and IMU onto the simulation sub to emulate the sensor suite of our actual sub. To test our controllers and determine their accuracy, we compared the ending configuration of the sub using our onboard IMU vs. using the ground truth pose and twist data from the simulator and found that both were able to achieve our desired setpoint in a similar amount of time.

The Saliency Detection and K-Nearest Neighbors algorithms described above detect and identify a set of 4 large dice underwater (from the 2018 competition) with 30.2% accuracy. There were two main reasons which we determined were the cause of the discrepancy. The first is the small bit of noise that still exists even after the background removal, which causes the resulting bounding boxes to jump around. This can be mitigated by interpolating between the periods of stability, through the periods of noise. The second reason is the method of measurement itself, the ground truth, is somewhat flawed due to bounding boxes only having been measured 3 times per second for 30 FPS footage. Often, thus, the ground truth lags behind.

Since Minimum Barrier Detection relies on a raster scan of all the pixels in the frame, it underperformed on our onboard computer, NVIDIA Jetson AGX Xavier achieving 12 FPS, which did not give enough room to perform further analysis on the frame. Thus, to increase algorithm efficiency and headroom, we implemented a few major improvements. First, to mitigate the slow performance of Python when compared to C, we used the Cython module, which compiled the code into a C-based interface. This meant accessing low-level memory as frequently as possible, instead of falling back to slower Python implementations and the Global Interpreter Lock (GIL), which limits processes to one thread. We also predefined memory allocation blocks for the code. Finally, we noticed that it was not necessary to analyze every pixel in the raster scan, as we achieved similar classification accuracy. Thus, we chose to run the algorithm skipping to every 5 pixels in the frame, such that every 25 pixels were subsampled to one pixel. These performance improvements led to a 53% FPS gain on the Jetson. This performance gain meant that we were able to more comfortably implement our task algorithms with the remaining overhead.

Acknowledgements

We at the Underwater Robotics Team at Berkeley would like to thank our sponsors:

Sofar Ocean
Omnisci
Nvidia
Solidworks
Berkeley Engineering
Berkeley OCF

References

[1] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction." in *Pattern Recognition Letters 27*, vol. 27, no. 7, 2006, pp. 773–780.

[2] J. Zhang, S. Sclaroff, Z. Lin, X. Shen, B. Price, and R. Mech, "Minimum barrier salient object detection at 80

fps," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1404–1412.

[3] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation," in *OCEANS 2016 MTS/IEEE Monterey*. IEEE, sep 2016. [Online]. Available: https://doi.org/10.1109%2Foceans.2016.7761080