BeaverAUV

Sam Steere, Andre Gordon, Milin Chabhra, Kyle Chiu, Emma Thiebault, Macy McKinney, Ben Klinman, Bradley Stone, Kyle Benton, Matthew Baltay, Ava Restuccia, Abigail Coe-Rohl, Max DiCarlo, Matthew McAdam

Anchovy

08.2025

BEAVER COUNTRY DAY SCHOOL

ABSTRACT - This paper will overview BeaverAUV's competition strategy and technical design. The team's AUV, Anchovy, has been upgraded and outfitted with parts to perform the team's competition strategy. Anchovy will complete the same tasks as attempted last year, and will utilize major improvements such as computer vision and a DVL to complete tasks that Anchovy was not capable of completing in the past. BeaverAUV's testing strategy involved integrating these new components into Anchovy and refining Anchovy's movement using the team's hovercraft.

COMPETITION STRATEGY

BeaverAUV includes students with different levels of robotics knowledge, with many different backgrounds in all aspects. As a result, the team's competition strategy balances task performance with program time management, which will allow for Anchovy to complete many tasks, while also leaving time knowing that Anchovy may require multiple runs to complete the ideal program. Anchovy will complete four tasks, being the coin flip, the gate, the slalom, and the octagon. Anchovy will also aim to complete the torpedoes tasks, if the team is able to design a working torpedo system prior to the competition, and the return home mission, Anchovy is able to complete the rest of the set program quickly. Anchovy's program starts with the coin flip. Anchovy's IMU is calibrated on the dock facing the gate, and then is placed in the water at the correct orientation. The IMU is then able to correct the orientation and head towards the gate. Anchovy will pass through the middle of the gate, and will yaw 720 degrees midway through. Anchovy will then proceed to the slalom, where Anchovy will use computer vision in order to navigate between the red and the white pipes. From there, Anchovy will continue to the octagon, where it will use it's DVL to estimate the distance from the slalom to the center of the octagon, where Anchovy will surface. From there, Anchovy's run will either be over, or continue and attempt to pass back through the gate and return home.

DESIGN STRATEGY

MECHANICAL DESIGN

1. THE FRAME

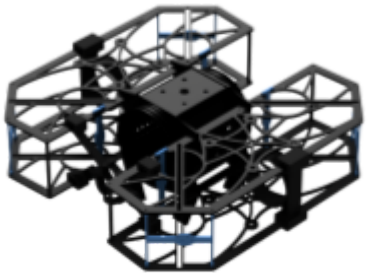

Fig. 1: Anchovy's frame with 3D-printed attachments

Anchovy's frame has remained the same for the past few years. The frame, made of ¼ inch anodized aluminum and originally designed with modularity as a focus, has many attachment points and extra hull penetrations. Because of this modularity, the team has continued to use the same main frame, as it works soundly with the Anchovy's competition strategy, and continually supports yearly additions and iterations. For the 2025 competition, Anchovy's frame has received multiple upgrades, including revamped 3D printed battery pod holders, a camera mount (not pictured below), and a DVL mount, while a torpedo system is currently being designed as an attachment as well.

2. HULL

Fig. 2: 7.5" ID acrylic tube and aluminum end cap assembled.

Anchovy's main hull is made up of two ¼ inch thick acrylic tubes with a 7.5" ID. O-ring's seal the central hub with the hull and the end caps. The hull is vacuum sealed prior to being placed in the water which lowers the pressure within the sub to below ambient air pressure, and pulls the hull sections and end caps closer together towards the central hub. This ensures a tight, waterproof seal in Anchovy. BeaverAUV chose to seal Anchovy in this way because both of the hull and end cap assemblies can be removed and replaced without disconnecting any electronics or removing any screws or clamps, which provides a relatively simple waterproofing system for the submarine.

3. THE CENTRAL HUB

Fig. 3: The central hub with hull penetrators attached.

Anchovy revolves around a central hub, which is also made out of aluminum. The two acrylic hulls attach to the ends of the central hub, where much of the motor wiring, kill switch, and arduino are stored. The central hub has many penetrators along the sides, many of which are not currently in use, but exist to ensure that modifications and attachments can always be made for Anchovy. The central hub also has a SubConn four-contact underwater connector (not pictured below), which allows for task information to be passed through the sub even while completely sealed. Lastly, there is also a downwards-facing camera mount on the bottom of the central hub, which gives an easy way to follow path markers along the floor of the pool.

ELECTRICAL DESIGN

Anchovy's electrical system helps to turn the information from the software into actual movements of Anchovy. A custom designed PCB attaches to an Arduino Nano, and relays commands from the Raspberry Pi, which is Anchovy's main computer, to drive 8 ESC's, which then give commands to the motors. The ESC's also control the current directly from one of Anchovy's two batteries, which powers all eight motors on the sub. The second battery is dedicated to powering the computer stack. The computer module is centered around a network switch, which interlinks the onboard NVIDIA Jetson TX2, onboard Raspberry Pi, and any external computer. The computer module is both connected and tethered with a skeleton-like rack to essentially sandwich the parts together while still allowing airflow and access to the components.

SOFTWARE DESIGN

One of the main goals of this year's design was to have multiple hardware platforms (the main sub and the hovercraft test bed, as well as a currently-incomplete mini-sub). To complete that goal, BeaverAUV created the Python library EZAUV which is published publicly on PyPi and available through pip. This will allow for our code to be used by other teams, especially beginning teams who do not yet have a mature codebase. The library's main feature is to take in a set of standardized functions to read/write sensors and thrusters, and from that automatically move at a set of expected accelerations (both translational and rotational).

The core principle of the library is to abstract as much of the hardware away from the user as possible throughout the code, such that the user only has to provide base hardware interfaces in

the initialization of the program and from then on can interact with the sub through only high-level functions. In pursuit of that, the library must first take into account the base hardware pieces so it can calculate the rest internally; it requires the moment of inertia tensor, relative thruster locations, thruster thrust vectors, and functions to set those thrusters to given speeds. It also accepts per-thruster outer bounds (positive and negative, representing the maximum percentage allowed; for example, the hovercraft is not permitted to go above 20% due to safety concerns) and dead zones (also both positive and negative, representing the minimum percentage before the static friction prevents initial rotation from a stopped position). Once the needed data is provided, an acceleration in terms of translation and rotation on each axis can be given to set all the thrusters to the needed speeds. Generally, these accelerations are passed in terms of a Task, a standardized class which gives accelerations over time.

1. INERTIA BUILDER

To simplify the process of creating the needed inertia data, EZAUV provides an InertiaBuilder class. This class takes in a set of InertiaGeometry objects, and computes a total moment of inertia tensor. InertiaGeometry is an abstract class with some built-in helper methods to assist in creating new types of geometries, and EZAUV also provides a few pre-made subclasses: Sphere, HollowCylinder, and Cuboid. The sub can be ‘modeled’ out of these geometries, and the final tensor will be computed automatically.

2. MOTOR CONTROLLER

One of the most important parts of the code is the motor controller, which solves for the needed motor speeds and drives the motors at those speeds. To solve for the motor speeds given, it should first be considered that the problem which must be solved is

$$\begin{bmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} \tau_{wanted} \\ \alpha_{wanted} \end{bmatrix}$$

Where τ_n represents the thrust vector of the nth motor, α_n represents the angular acceleration of the nth motor, and a_n represents the speed the nth motor must go at to reach the desired total acceleration. The matrix on the far left with all motor information will from now on be referred to as the motor matrix, or M. The problem must be solved such that each a is within $[b_{nl}, d_{nl}] \cup [0] \cup [d_{nu}, b_{nu}]$, where b_{nl} represents the lower bound, b_{nu} represents the upper bound, d_{nl} represents the lower deadzone, and d_{nu} represents the upper deadzone. The solution

should also be exactly at or at least near the minimum of the sum of squares of a; it would not be acceptable if, for example, all motors turned on at maximum speed when trying to stay still. Although it would technically be correct, it would obviously be an unwanted outcome. To solve this, the problem is formulated as the following mixed-integer quadratic programming problem (or MIQP):

$$
\begin{aligned}
&\textit{Continuous variables } a_n, \textit{ and binary variables } z_n \textit{ and } s_n \\
&\textit{Motor matrix } M \\
&M_0 = \textit{ bound with the highest absolute value } = \max(|b|) \\
&n = \textit{number of motors} \\
\\
&a_n \geq z_n \cdot b_{nl} \\
&a_n \leq z_n \cdot b_{nu} \\
&\textit{The speeds should be in the range of the bounds, unless } z = 0, \textit{ in which case } a = 0 \\
\\
&Ma = V \\
&\textit{Ensure the multiplication is true} \\
\\
&a_n - d_{nu} \cdot s_n + M_0 (1 - s_n) + M_0 (1 - z_n) \geq 0 \\
&a_n - M_0 \cdot s_n - d_{nl} (1 - s_n) - M_0 (1 - z_n) \leq 0 \\
&\textit{Enforces the speed is on the positive or negative side of the deadzone} \\
\\
&\textit{Minimize } \sum_{n=1}^{a} a_n^2
\end{aligned}
$$

Currently, the problem is solved using Gurobi, but we plan to transition away from it. V is the expected vector, which is composed of the translational and rotational acceleration vectors stacked atop each other. In this equation, z_n represents whether $a_n = 0$ exactly, and s_n represents whether a_n is in the positive or negative side of the deadzone (assuming $z \neq 1$).

In practice, there often is not an exact solution considering rotations and highly precise wanted accelerations. Due to this, EZAUV uses a two-step process, where the problem is first solved with the $Ma = V$ term replaced with $Ma + \epsilon = V$ and the objective with $\sum_{n=1}^{\epsilon} \epsilon_n^2$. Then, the problem is solved again with the original objective and the epsilon term as constant. This means that, if the wanted acceleration is not possible, the program will solve for the *nearest* acceleration.

3. CONTROL PROGRAM

To decide the acceleration which will be sent into the motor solver, the program uses a few modules (which are also provided by EZAUV) to create a total wanted acceleration. The program uses a PID controller on depth and angle, and uses simple Task objects to decide

which direction to move, like AccelerateVector to follow a vector over a period of time. Although not currently implemented, BeaverAUV also has plans to attach a DVL to the sub which will allow the program to define tasks in terms of velocity rather than acceleration.

TESTING STRATEGY

The BeaverAUV team built a hovercraft from scratch which is the same size as its aquatic counterpart, Anchovy. The hovercraft was made in order to test the code live without needing access to water or a pool, and ultimately allowed for the team to test coding improvements more frequently. The hovercraft is controlled by an Arduino Nano and Raspberry Pi which mimics the computer setup of Anchovy. The hovercraft also uses the same IMU as Anchovy, which has led to accurate movement testing. Building the hovercraft not only allows for the team to get a better understanding of how Anchovy will move, but also works as a way to develop new improvements for Anchovy's movement code.

SIMULATION TESTING

To help test the code to a base degree, BeaverAUV also created a custom two-dimensional Python simulation to test the code. This simulation is built using Pygame, and simulates input/outputs equivalent to those available on the hovercraft. It uses basic physics, but provides enough simulation to see if the program generally works. BeaverAUV also has an experimental three-dimensional simulation built in Unity.

ACKNOWLEDGEMENTS

BeaverAUV would like to thank Morgan Muschamp, Zoz Brooks, Melinda Ching, and Blake St. Louis, the team's faculty advisors, for their continued, dedicated support of the team. Thanks are also due to Erica DeRosa for her valuable advice regarding sponsorships and outreach. Without her, the team would not have been able to secure thousands of dollars in valuable funding. The team would also like to acknowledge Beaver Country Day School and former Head of School Kim Samson for facilitating and funding the project. Several sponsors also contributed to the BeaverAUV team. The team would like to thank sponsors WaterLinked, Nvidia, BlueRobotics, MathWorks, Connect Tech Inc., and VectorNav for supplying products that are crucial to the performance of Anchovy. Beaver AUV would also like to thank the team's generous donors, who made it possible to purchase critical parts for Anchovy.