Design and Implementation of UFRJ Nautilus' AUV: BrHUE 2020

Gustavo R. Villela, Ana Clara L. Cruz, Felipe B. Costa, Ramon Christian Mendes,
Lara F. de Amorim, Vitor A. Pavani, Claudio M. de Farias

***Abstract*— This paper discusses the development and implementation of UFRJ Nautilus' latest AUV (Autonomous Underwater Vehicle) design: BrHUE 2020. Through the competition strategy it is clear that the team philosophy this year was to take the sub's main weaknesses and turn them into its main strengths: a reinvented SLAM based navigation system. With that in mind, our main goal is thus to execute all tasks that depend solely on the robot's localization system. Through this processes, not only was the localization improved as a whole but there were also significant subsequent improvements in our beamforming algorithm since last year.**

UFRJ Nautilus is a team of undergraduate students from the Federal University of Rio de Janeiro focused on building low-budget and off-the-shelf autonomous vehicles.

I. INTRODUCTION

When developing a low-cost Autonomous Vehicle, several factors come into play, not only on the materials needed for its construction but also on the services and algorithms utilised by it. This situation proves to be even more delicate when dealing with an AUV. Given the constant risk of a possible flooding of internal electrical components, special care is needed when building its hardware. This framework, combined with the current COVID-19 pandemic, calls for imaginative solutions in order to be able to accomplish the team goal of improving the current AUV project with as few expenses as possible. As a result, the utilisation of open-source drivers and algorithms as well as the constant use and optimization of simulation software, prove to be good alternatives in order to test all possible improvements even remotely.

Although several efforts were made in order to compensate for the lack of funding that, not only prevents us from developing a new project but also limits our possibilities for acquiring new and better sensors, we were able to make significant improvements in our current project. This evolution can be seen throughout all the team's areas and, combined with our improved and more focused strategy, allowed us to expand even more on our AUV's capabilities.

II. COMPETITION STRATEGY

This year, given our restricted funding and time to physically work on our robot, we decided to focus our efforts on getting the most points possible from all tasks that don't involve any object manipulation.

Since the competition won't happen on a physical environment, we decided to simulate it on Gazebo with all the tasks that we would have performed. By doing so, we were able to make significant progress on the performance of our sub, without the need to make direct changes to its hardware.

First, our proposed path initiates with the Coin Flip, followed by passing through the Gate into the G-man side. The choice will be predefined by our State Machine, which will look for this image. After passing the gate, BrHUE will perform a 720º yaw turn around its center in order to gain the maximum Style Points and will proceed to follow the Path using its bottom cameras. Then, it will go towards the buoys in order to hit the Badge buoy given that we chose to follow the G-man side of the gate. Finally, it concludes by following the second Path and proceeds to the course by going directly bellow the Octagon and emerges inside its boundaries. After each try, BrHUE remains on, except for its motors, enabling for better mapping. Both of these concepts will be further expanded upon in future section of this technical report.

It is worth to mention that during all tasks, BrHUE will be receiving precise control commands by its State Machine. The latter will determine its decisions by using SLAM information that, combined with the image recognition algorithms of its Neural Network, will be able to determine where the robot is located as well as which task to perform next.

III. DESIGN

Several factors contributed to the improvement of the latest model of BrHUE. Though they may vary a lot in nature, we can attribute them mainly to the acquisition of new resources in the form of sensors or other types of hardware and - most importantly - to a better and more efficient application of them throughout all main areas of the team. Among them, we can highlight the acquirement of a new IMU and the implementation of four hydrophones provided by the Brazilian navy, both used for our new and improved SLAM. We maintained the general structure of last year's version of the sub in order to prioritize spending time testing it so that we could improve the AUV's software, which makes this year's version of the sub much more competitive.

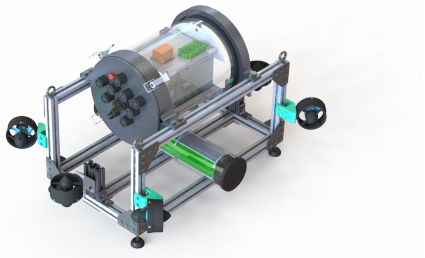


Fig. 1: BrHUE 2020's SolidWorks rendering .

A. Mechanical System

Our robot was built mainly aiming to be a low cost version of an easy assembly and disassembly project. Therefore, the mechanical team adopted the concept of modularity so that we could also achieve a cheaper and easier way of transporting it. Nowadays it uses six Blue Robotics T200 thrusters in order to enable a total of 5 degrees of freedom: x, y, z, pitch and yaw. Material-wise, it consists of an aluminum frame, a naval aluminum cover, an acrylic main hull ,attachments with UHMW covers and 3d printed thrusters holders made of re-utilized ABS filament. Those materials were chosen to achieve the lightest model while maintaining high resistance and always being careful with our tightness. As for a tightness improvement we added a pressure valve that will ensure our hardware safety and guarantee an easiest and fastest way to open BrHUE.

Furthermore, the team started using ANSYS software to analyze different aspects of our AUV with both structural and CFD(Computational Fluid Dynamics) simulations.

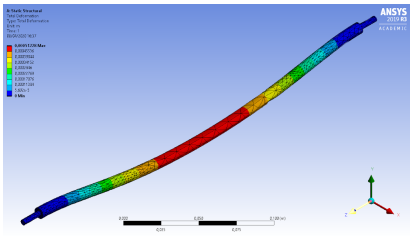


Fig. 2: Obtaining the deformation caused by an oversized load applied evenly on the rods.

One that had direct influence on our current model was the structural simulation of the rods in which we simulated different materials with the purpose of achieving the highest cost-benefit taking into account the least possible deformation. The simulations showed us that the best materials would be Stainless Steel or AISI 1020 Carbon Steel, which presented very close deformations when loads equivalent to the electronics weight were applied. Therefore, we opted for using the latter material, since it has a lower cost. As for electrical and software demands two new parts were implemented:

1) Hydrophones: To improve our spatial arrangement and better accommodate the hydrophone plate, we decided to add an attachment to our frame, for the purpose of housing this system. Also, in order to successfully dispose our hydrophones in a way that would reduce the vibration noise, a holder composed of a junction of 3d printing and resin was made.

2) New kill Switch System: With the addition of a new kill switch, we were able to turn off the AUV's thrusters without turning off our main system. Also, this year the model was modified to achieve a more optimized version, that would have our two magnets out of the acrylic instead of one outside and one inside like we had last year. This change brought us a faster and safer way to assembly.

(a) Hydrophone Attachment (b) Kill Switch

Fig. 3: 3D Models generated with SolidWorks.

B. Electrical System

The Electrical project of BrHUE was made to be functional and low-cost. Since the priority of our team this year was to spend the most time we could testing our AUV, we decided not to make great changes on its electrical system. So, we prioritized to improve the existing systems, making the sub easier to operate. The AUV is made by the sum of the following parts:

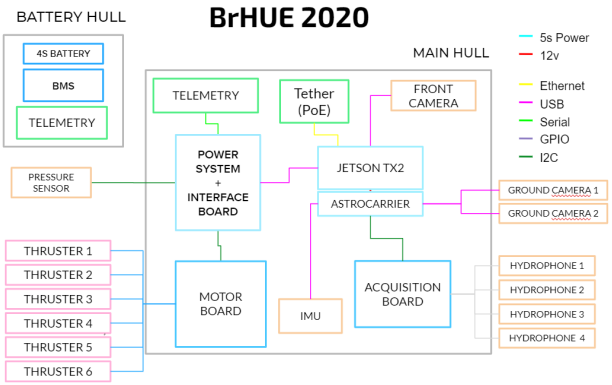


Fig. 4: Diagram of BrHUE's Hardware.

1) Battery and Power Management: We use a 4S Lipo battery and, in order to make it safe and stable, we make use of a BMS (Battery Management System) partly fabricated indoors. Besides using an Protection Circuit Module to balance the cells, this year we built a board to indicate the battery's status and temperature both to the main computer of the sub and the person operating it. This change made the sub more secure and easier to operate.

2) Internal Communication: In order to integrate all of the sensors and actuators to the main computer, we created a system of internal communication. The motors and the data acquisition boards have each an ATMEGA controller, that communicate to the main computer by using the i2C protocol. Some of the sensors such as the cameras are directly connected to the main computer via USB.

3) Propulsion System: This system is responsible for controlling and delivering power in a secure way to the thrusters. This year, we upgraded this system by adding a more robust security protocol against over-current and over-voltage as well as a circuit able of measuring the current that is going through the Electronic Speed Controllers.

4) Passive Sonar (Hydrophones): Acquiring the signal of our hydrophones was the most challenging electronic project on this year's version of our sub. Since the acquisition boards

on the market are too expensive and most of them are not compatible with our software, we decided to make the whole data acquisition system (filter, amplifier and A/D converter) indoors.

This solution was not only cheaper than buying an acquisition board, but was also simpler and more efficient.

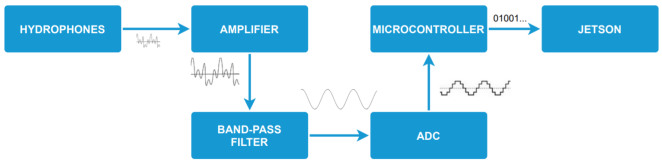


Fig. 5: Acoustic data acquisition.

We made it by first processing the raw data (amplifying and filtering the sign to the frequency of the pins on the pool), then converting the data to be digital using a Analog Digital Converter. The digital data is then processed by the micro controller that sends the information to the main computer by the internal communication system.

5) Kill Switch System: In order to guarantee the security of the sub and the people around it without harming the integrity of the whole electronic structure, we developed the kill switch system. This system includes a magnetic key of 3 parts: the first "ON", in which the sub is operating normally, the second "HALF", in which the motors are off but the the computer is on, and the third "OFF", in which the bot is all shut down. This year's implementation of the "HALF" part of the system, enabled people around the vehicle to be safe manipulating it while all of the instances of the bot are working. By doing so, no time is wasted re-initializing the main computer between races and the cameras can still map the environment even when the sub is not with it's motors on.

C. Software

This year, the main focus of development from the Software team was to implement ROS-based SLAM algorithms and, most importantly, integrate them with the rest of our repository. In order to achieve these improvements, we also implemented several changes in the Software team as a whole, restructuring it in three main areas: SLAM, AI and Simulation. Even though the main improvements can be seen in the SLAM area, the team as a whole underwent a relevant evolution, specially compared to last year.

Therefore, we were able to develop an absolute control system based on our localization, which, when combined with our image recognition neural network and state machine, enables a reliable autonomous navigation.

1) SLAM: In previous iterations of BrHUE, we based our localization system solely on the sensor fusion provided by the robot_localization package without taking into account the frame of each sensor. This made it impossible for us to truly have a reliable source of Odometry data, therefore rendering our localization system extremely ineffective.

However, this scenario has completely changed since last year. Firstly, we concentrated on correctly making use of ROS' TF library in order to properly establish all relevant frames. Using the robot model's urdf file, we are now able to easily define all transforms without having to write multiple broadcasters and listeners. This way, we have a lightweight solution to our frame problems, given that ROS has optimized functions to accomplish these tasks.

Furthermore, we expanded on this study by applying the final urdf model (with the complete TF tree) on RTAB-Map for ROS [1]. This consisted in loading the model in this assortment of SLAM algorithms. To load the former, it was necessary to research the inner workings of RTAB and specially how it developed its SLAM. The algorithm consists in developing a point cloud map - which is then stored and can be later recognized in closed-loop scenarios - via camera input and then fusing it with the Odometry data. The latter can be provided either exclusively by the camera system or by any other Odometry sensor - or via sensor fusion algorithms - in order to perform the localization.

We therefore utilized the robot_localization package provided by ROS to accomplish the data fusion from our localization sensors. This package consists of two types of Kalman Filters which can be chosen by the user: the Extended Kalman Filter(EKF) and the Unscented Kalman Filter(UKF). As it is shown by [2], Kalman Filters are state estimators used, in general, to calculate values for which we don't have direct measurements to and to fuse sensor data by giving different weights to them and the state matrix model. It is also known that EKFs are used for slightly nonlinear systems - given their use of Jacobians - with Gaussian error whereas UKFs can be used for even less linear systems but requires an error of the same nature as the former, though it can lead to high computing requirements [3]. Given its flexibility and the low difference in computational power demand in our case, we then opted for the UKF as our estimation and sensor fusion algorithm.

However, in order to achieve the best performance possible, all sensor messages had to be adjusted to ROS' standards, for which we came up with fast and effective solutions. Since both depth sensor and hydrophones (as an ensemble) provided their data in a non-standardized manner - the first being floating point number data and the second being an array of two angles: the robot's azimuth and elevation angles in respect to the pinger - the first step we had to take was to republish both as Pose with Covariance Stamped ROS messages in their original frames. An explanation of how we fixed the depth frame can be found in the Appendices I.

Another great improvement since last year was an improved version - created by our team members [4] - of our Beamforming algorithm. A more in depth discussion o the algorithm can be found in the Appendix II.

Finally, our tests on how well our localization improved consisted in progressively adding sensors in our Unscented Kalman Filter algorithm in our simulation. To guarantee the consistency of the tests we made sure the robot performed the same path (by recording a rosbag and playing it with the different sensors setup) and then compared the resulting map and location between each try and the exact values provided

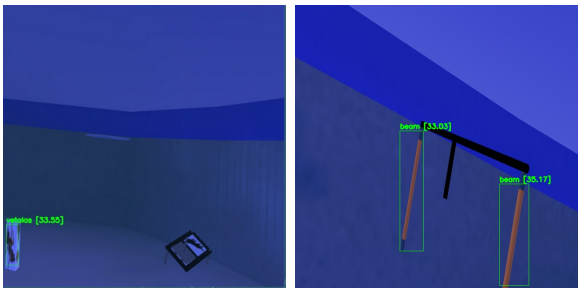
by the simulation. By doing so, we were able to more clearly measure the influence each sensor provided to the final SLAM. As a result, it became easier to tune the covariance matrixes in our UKF, given that it was now possible to isolate the source of drifts and/or noisy data sources. Although this method provided us with relevant results, we were unable to perform them in a real pool given the global pandemic scenario. Because of that, some fine adjustments will still be in need until the quarantine is finally lifted.

2) AI: We consider our robot's intelligence everything related to its decision making: from how much force it decides to output in each thruster in order to move in a certain way, to where it wants to go. Consequently, this encompasses everything related to its Control System, Neural Network and State Machine.

Last year, a great part of the problems related to the AI area were due to the lack of a reliable localization system which prevented us from performing an Absolute Control System, which can be described by [5]. Therefore, by implementing our SLAM we already had a better intelligence as a whole, but more changes were made in order to further this progress.

The current State Machine employed in the 2020 iteration of BRHue was completely redesigned to incorporate a more modular approach, while simultaneously making the access to sensory data a lot more robust and reliable. To achieve these goals, we have developed a system of hierarchical discrete State Machines, implementing the ROS smach package. In practice, this means we now have a main State Machine whose states are other independent ones, each with the ability to solve the different tasks of Robosub. This approach allows for greater abstraction, since we are able to implement very complex logic for a given task,without making the main State Machine more convoluted than it needs to be. The only states in the main State Machine that are not themselves other State Machines are the navigation states, which are responsible for taking the robot from one task to the next. The other front in which this year's implementation has matured greatly is how information generated by sensors, SLAM algorithms and the Neural Network is accessed from within the state's code. Said improvement was possible due to the creation of a wrapper that stores the most recent data from each input source from which all individual states inherit from, and, therefore, have access to. This eliminates completely the need of separate ROS subscribers, for each state and simultaneously gives the data gathered a greater sense of consistency. This occurs because all states interact with the same data coming from the same instance of the wrapper, and thus, agree on what information is available, hence preventing possible inconsistencies.

Furthermore, for this iteration of Robosub, we decided to drop the strategy of using our Neural Network as an input to our main localization system due to inconsistencies originated from different bounding box orientations. This problem originates from the fact that, when recognizing an object that is rotated, the dimension of the bounding box may not match those of the recognized task, leading to an erroneous size measurement, causing an inconsistency in the size-distance ratio. Therefore, this year we decided to maintain our object detection Neural Network YOLOv3 due to its lightweight processing power and high accuracy, using it only as an input to our State Machine to recognize whether or not BrHUE has arrived on the next task, as represented by Fig. 6 shown bellow.



(a) Vetalas (b) Beam

Fig. 6: Tasks recognized in simulated environment.

3) Simulation: In order to fill the gap created by a lack of physical tests due to the COVID-19 pandemic, great efforts were made by our team to optimize our simulation environment. This changes enabled us to continue improving our SLAM algorithms during the quarantine.

We began working with the migration of our simulation software from Gazebo 7 to Gazebo 9, this change also allowed us to migrate our ROS version from Kinetic Kame to Melodic Morenia. The next change we worked on was the 3D model of our sub, previously all 3D mesh used by our simulation and in our visualization software (Rviz) was direct exported from Solid Works, in terms of model mesh fidelity this is the best workflow. Since the model exported is too heavy to allow the simulation software to run in real time, to overcome this problem we started using the 3D creation tool Blender to optimize these models.

IV. EXPERIMENTAL RESULTS

A. Beamforming

With the help of the Marine Research Institute (IPqM), tests were carried out where the signals emitted by a pinger were recorded trying to imitate the possible signals that could be emitted by the RoboSub pinger, that is, sine pulses of 40ms every two seconds with frequencies of 25kHz, 30kHz, 35kHz, 40kHz. Four hydrophones were used, with one hydrophone on the x axis, one on the y axis and two on the z axis, in order to measure the azimuth and elevation between the pinger and the hydrophones. For calculating the algorithms times an average of 100 iterations was taken plus standard deviation. The metrics used to evaluate the implementation of Beamforming are: accuracy, precision and execution time. Accuracy is considered good if the value returned by the algorithm is within 5 degrees of the expected value.

The time domain Beamforming, demonstrated good accuracy, but low precision, returning a set of possible angles. Its

run time is acceptable, with an average of 0.5 seconds on the tested hardware. On the other hand, frequency Beamforming has good accuracy and high precision, with the algorithm returning only a couple of angles, referring to the azimuth and elevation of the sound source. Unfortunately, its average speed on the tested hardware is around 1.3 seconds. Considering that the robot must run other algorithms besides the Beamforming using lower end hardware, there is a possibility that the execution time will exceed the 2-second time limit.

For the rapid implementation of the Beamforming we achieved the same accuracy and precision as Beamforming in the frequency domain. Its execution time was approximately 0.03 seconds, being the fastest algorithm among the three. It is possible to see from the data that the rapid implementation of Beamforming was the one that best suited the metrics, having the best precision, accuracy and execution time.

	Time	Frequency	Mixed(n=8)
Execution time	0.57 $\pm$ 0.01	1.35 $\pm$ 0.03	0.032 $\pm$ 0.002

TABLE I: Time $\pm$ standard deviation (σ) for the different types of beamforming.

B. SLAM

The incremental addition of sensors to RTAB-Map proved to be extremely effective. We hoped that, by doing so, we would be able to evaluate how an IMU and depth sensor improve orientation and depth accuracy respectively. The path starting at (0,0) was originally made in Gazebo, for which a rosbag was recorded and then replayed with each sensor implementation as seen in Fig. 7, where V.O stands for the Visual Odometry provided by RTAB.

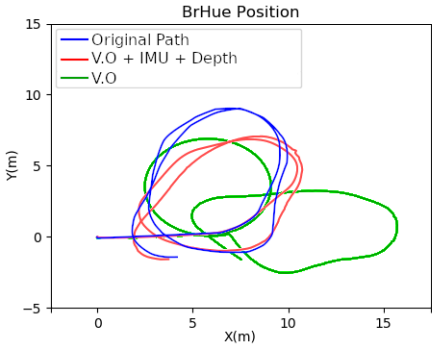


Fig. 7: Path comparison with incremental sensor implementation.

Analysing the different returned paths it is possible to pinpoint the moment of the trajectory where the algorithm was unable to capture points for the map, even with the textures at the bottom, which is where the V.O path describes an elliptical motion. One possible explanation to this scenario is that, before getting lost, the V.O was registering a counterclockwise curve, whose data was then repeated until the algorithm found itself again. From that point on, the V.O then recognizes another curve but, this time, gathering points, which returns a path that is slightly similar to the original path's second curve, besides the poor orientation - a common characteristic of V.O algorithms.

On the other hand, with all sensors fused, it is clear to see a significant improvement, specially orientation-wise. This result confirms our expectations for the IMU's contributions to the final path. Using the same V.O, even with the lost path, the IMU data prevents the sub from completely getting lost and allows for a more accurate localization.

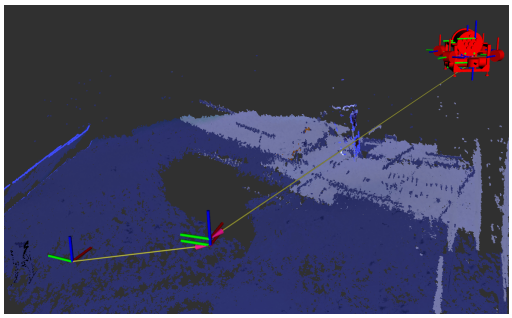


Fig. 8: Mapped path with V.O, IMU and depth sensor.

Finally, although there definitely is room for improvement, the results showcased by our SLAM system show a clear evolution from last year. Not only that, but is also worth to notice the importance of our new kill switch in this Robosub's edition, given that this closed-loop scenario recorded in Gazebo is only enabled by this new strategy of ours.

V. Acknowledgements

The UFRJ Nautilus team would like to thank all of the people and institutions without whom it wouldn't have been possible to complete this project. Firstly, our thanks go to all the team members, from technical and non-technical departments, for their contributions to the team as whole, be it developing the technology we showcased in this technical report or raising funds and sharing the team's image and vision to the world. Moreover, we would like to thank all of our supporters and sponsors, for giving us the financial and technical support needed for the growth of our team. Finally, we would like to thank our advisor teacher Claudio Miceli de Farias, the Tércio Pacitti's Institute and the UFRJ Polytechnic School that assist us and provides laboratories to work.

References

[1] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ROS-based Visual SLAM methods in homogeneous indoor environment ", IEEE 14th Workshop on Positioning, Navigation and Communications (WPNC), Bremen, 2017.

[2] A. C. Charalampidis and G. P. Papavassilopoulos, "Computationally Efficient Kalman Filtering for a Class of Nonlinear Systems", IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 56, NO. 3, 2011, pp. 483–491.

[3] Y. Hao, Z. Xiong, X. Wang and F. Sun, "Comparison of Unscented Kalman Filters", International Conference on Mechatronics and Automation, 2007.

[4] S. Simplicio, H. Júnior, G. R. Villela, F. Costa, V. Pavani, L. Rodrigues and C. de Faria,"Development of the UFRJ Nautilus' AUV: A Multisensor Data Fusion case study", International Conference on Information Fusion, South Africa, 2020.

[5] V. Rasvan, "Absolute stability of a class of control systems described by functional differential equations of neutral type", Equations Differentielles et Fonctionnelles Non Lineaires, Paris, 1973.

APPENDIX I
FIXING THE DEPTH SENSOR FRAME

When working with a this kind of sensor, one must always remember that, no matter its orientation, it will always return the same depth measurement. Although this is consistent with its purpose, the transform in relation to its parent frame (map) may return a measured depth that does not correspond to the sensor's actual depth in its frame (depth_link), which we will respectively call D_m and D_a. Though this phenomenon, graphically represented in Fig. 9, does not happen with a rotation along its z axis - since the plane generated by the x and y axes will remain parallel to that of the water, considering it completely flat - when there is a change in pitch or roll of θ, the new measured depth D'_m will then be equal to $\cos(\theta) \times D_m$, which will be different from D_a.

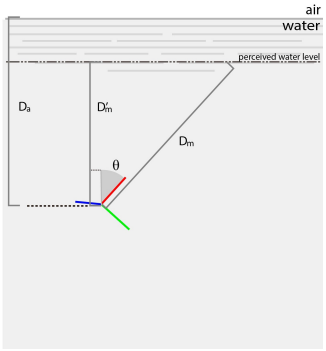


Fig. 9: Rotated frame inconsistency phenomenon.

To prevent this contradiction, a possible solution is to consider the measurement frame as a placeholder (depth_placeholder) while creating an intermediate one (water_footprint) between the former and the map frame. This new frame will act as a projection of the robot's movement on the water's surface, therefore it will always follow the robot's movement and have it's z axis facing up - i.e. perpendicular to the water's surface. After that, we define the transform between depth_placeholder and water_footprint as a translation on the z axis equal to D_m while all other states (x, y, roll, pitch and yaw) remain the same as its parent the latter frame, as it is shown in Fig. 10. By doing so, depth_placeholder becomes a virtual frame whose data will be converted via a Lookup Transform command - i.e. a listener - in order to guarantee that, in all circumstances, $D_a = D'_m = \cos(\theta) \times D_m$ on the depth_link frame. In practice, this method forces the depth sensor frame attached to the robot to be always facing up. That way, it is then possible to correctly implement the final measurements from depth_link to the robot_localization UKF without having frame issues.

As an example, let's examine the case in which $D_a = 2m$ while the sub pitches with an angle of $\theta = 60°$. If the

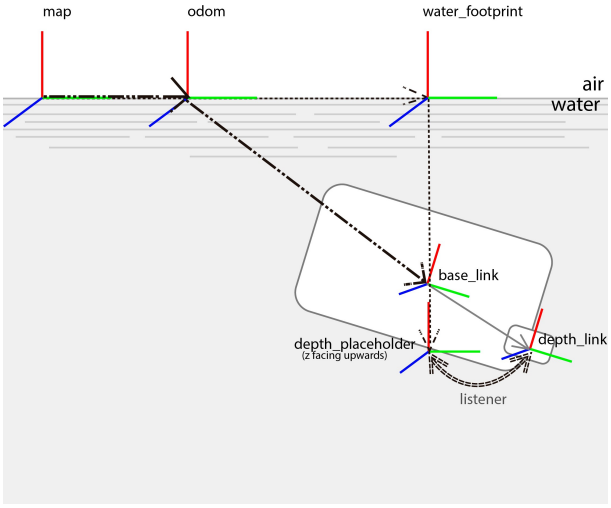


Fig. 10: Frame fixing method via water_footprint.

correction is not made, then $D'_m = \cos(60) \times 2 = 1m$, which is clearly inconsistent with D_a. In contrast, by following the proposed method, the TF system interprets depth_link's D_m as being $D_m = \frac{D_a}{\cos(\theta)} = 4m$. Although this result might seem odd at first, when comparing D_a and the new D'_m, we do see that their values are both equal to $2m$, which means that the depth the rotated sensor frame perceives to be in is equal to the actual value i.e. $D_a = D'_m$.

APPENDIX II
THE BEAMFORMING ALGORITHM

The simplest form of the Beamforming algorithm is to use a multitude of sensors that will receive the signal at different times and calculate its Fast Fourier Transform (Also known as frequency Beamforming). By converting the signals with a delay matrix, based on the configuration of the sensors, it is possible to find the direction of the sound source through the argmax of the square of the Inverse Fourier Transform. For elevation and azimuth angles between 0 and 180 degrees, this whole calculation can be costly enough so that the robot's position will already be out of date by the time the algorithm is finished computing, or that signals emitted by the source will be lost in the meantime, limiting its practical use for real-time localization.

In order to find the direction of the pinger, it was then proposed to combine the Beamforming algorithm in time together with that of the frequency, the former acting as a filter for the latter. The time domain Beamforming is faster than the frequency domain previously discussed, but it is less accurate, returning a region of possible angles, instead of a pair. So we proposed to first use the time domain algorithm and, for the set of angles it returns, it is computed again in the frequency domain. The argmax of the frequency domain Beamforming plus the min(argmax) in time domain is then the angle of the sound source relative to the robot.

APPENDIX III

Table of Components

Component	Vendor	Model/Type	Specs	Cost/If new
Buoyancy Control	-	-	-	-
Frame	Forseti	Aluminum Profile	-	-
WaterProof Housing	Ciplast	Acrylic	-	-
WaterProof Connections	Seacon	IL4MP/IL6MP/IL8MP/IL16MP	-	-
Thrusters	BlueRobotics	T200	5.25 / 4.1 kg f	-
Motor Control	BlueRobotics	BasicESC	R3 30A	-
Controllers	Digikey	ATMEGA 2560	-	-
Actuators	-	-	-	-
Propeller	BlueRobotics	T200 Propeller	-	-
Battery	MaxAmps	5200mah 4S LiPo Battery	-	-
Converter	-	-	-	-
Regulator	-	Custom made	-	-
CPU/GPU	NVIDIA	Jetosn Tegra X2	-	-
Internal Comm Network	-	USB/I2C/TTL SERIAL	-	-
External Comm interface	-	ETHERNET	-	-
Programming Language 1	-	C++	-	-
Programming Language 2	-	Python	-	-
Compass	-	-	-	-
Inertial Measurement Unit	Xsens	MTi-G-AHRS	-	-
Cameras	Logitech	C920	-	-
Hydrophones	Benthowave	Bii-7141	-	-
Manipulator	-	-	-	-
Algorithms: vision	-	ORB	-	-
Algorithms: acoustics	-	Beamforming	-	-
Algorithms: localization and mapping	-	RTAB-Map and Robot_localization	-	-
Algorithms: autonomy	-	PID and FSM	-	-
Open Source Software	-	ROS and Linux	-	-
Testing Time: Simulation	200h	-	-	-
Testing Time: In Water	25h	-	-	-
HW/SW expertise ratio	1/2	-	-	-
Team Size	35	-	-	-