Project Beluga: Perception and State Estimation in a GPS-Denied Environment

Final Report
Harvey Mudd College Engineering Clinic

Project Team:
Evan Chapman
Austin Chun
John Lee
Zayra Lobo
Dominic Frempong
Nancy Wei

Project Advisor:
Professor Anthony Bright

Project Liaisons:
Cyrus Huang
Vaibhav Viswanathan
Jerry Hsiung
Benjamin Chasnov

May 2017

Abstract

The HMC Techmation Clinic Team improved the state estimation of the Beluga autonomous underwater vehicle (AUV) in a GPS-denied environment by developing a perception system and state estimation algorithm for the robot.

The team narrowed their sensor options down to an acoustic positioning system (APS), camera, and sonar, with the APS ultimately being selected due to the drift of the other two sensors. The APS was integrated into the Beluga's perception system through a pipeline involving sampling the hydrophones at 62.5 kHz, performing a matched filter algorithm on the data to calculate time-of-flight of the signal, and then sending those calculations to the AUV's main computer. The two main issues with this pipeline were the signal attenuation at distances greater than 2 m, which could be fixed by a variable gain amplifier, and the spatial resolution (± 0.5 m) of the hydrophone data collected, which was due to the high velocity of sound in water (1500 m/s).

The team also developed a 2D Extended Kalman Filter (EKF) to fuse the Inertial Measurement Unit data, time-of-flight calculations, and dynamic system model in order to perform the most accurate state estimation possible given the sensor data and robot dynamics. Due to the hydrophone pipeline issues, the EKF could not be run on experimental IMU and hydrophone data this year. Instead, as a proof-of-concept, the team ran the EKF offline on time-of-flight measurements and IMU data with added Gaussian noise, and they observed significant improvements from the naive direct integration algorithm used for the Beluga's state estimation before.

Acronyms

APS	Acoustic Positioning System
AUV	Autonomous Underwater Vehicle
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
GPS	Global Positioning System
KF	Kalman Filter
IMU	Inertial Measurement Unit
LBL	Long baseline
OCXO	Oven-controlled crystal oscillator
OWTT	One-way-travel-time
ROV	Remotely operated underwater vehicle
SBL	Short baseline
ToA	Time of Arrival
TDoA	Time Difference of Arrival
TWTT	Two-way-travel-time
USBL	Ultra-short baseline

Table of Contents

Table of Figures

List of Tables

1. Introduction

1.1. Techmation

Techmation Co., LTD. is a manufacturer of control systems in Taipei, Taiwan and is sponsoring a 2017-2018 clinic project to improve the performance of low-cost Autonomous Underwater Vehicle (AUV), affectionately named the Beluga. The company was founded in 1984 and originally focused on the production of controllers for injection molding machines. In recent years, the company has changed its focus to robotics, sustainable technology, and "Internet of Things" products. Techmation has service members across the world in countries such as Brazil, India, Turkey, and Denmark, and it has produced over 150,000 controllers in the past three years.

The Beluga v1.0, depicted in Ffigure 1, is an autonomous underwater vehicle (AUV) created by Harvey Mudd College alumni in the summer of 2016. Last year, the Techmation clinic team focused on creating the control system for the Beluga, but ran into troubles with localization underwater without GPS. Thus the goal for this years clinic team is to improve the state estimation system, following the objectives, constraints, and functions described in Section 1.2.

Figure 1: Fully assembled Beluga v1.0.

1.2. Project Statement

The HMC Techmation Clinic Team will develop an exteroceptive perception system for the Beluga AUV v1.0 and v2.0 and further enhance the system's functionality by improving the current state estimation algorithm.

1.2.1. Objectives

Objectives of the perception system and state estimation algorithm include:

- Minimizing cost
- Estimating the robot's state with minimal error
- Maximizing ease of integration with the current system

1.2.2. Constraints

The design must meet the following constraints:

- Be exteroceptive
- Be waterproof, stable, and failsafe
- Run entirely on-board the Beluga AUV
- Operate underwater in a GPS denied-environment
- Cost less than $10,000

1.2.3. Functions

The functions of perception system and state estimation algorithm include:

- Collecting time-stamped data
- Reducing the error of the current state estimation algorithm
- Being applicable to other robotics systems

- Tracking the robot in the presence of disturbances (e.g. waves, currents)

1.3. Deliverables

By the end of the Fall 2017 semester, the team will have:

- Ordered and tested the selected sensors for the perception system
- Created a sensor comparison document to inform the final sensor selection
- Implemented a Kalman Filter (KF) algorithm to estimate the robot's position at least as well as direct integration from accelerometer data
- Learned how to operate the Beluga by performing a wet test
- Identified the leak in the shell of Beluga v2.0

By the end of the Spring 2018 semester, the team will have:

- Designed and integrated the selected sensor with the current Beluga hardware
- Improved and tested the state estimation algorithm further to completely fulfill the objectives, functions, and constraints listed in the above sections

1.4. Project Status

At the end of the Spring 2018 semester, the team has:

- Designed and integrated the selected sensor with the the robot's perception system, with some limitations and bugs in the system
- Improved the state estimation algorithm for 2D localization and tested with simulated sensor data to fulfill the objectives, functions, and constraints listed in the above sections

The limitations of the selected sensor and its data collection pipeline are discussed in Section 8. Due to these limitations, the team was unable to test the state estimation algorithm

with experimental sensor data and instead verified the algorithm's performance using simulated sensor data.

2. Background

The Beluga v1.0 was made by Harvey Mudd alumni in the summer of 2016. One of the goals of the Beluga project is to create a cost effective AUV, as the average unit costs around $70,000 to build and individual sensors can range from $5,000 to $100,000 each.[1] Another goal of the Beluga project is to create a working state estimator, which would allow the Beluga to determine its current state from a sequence of inputs and outputs. A desirable state estimation system would integrate information from sensors such as the IMU (inertial measurement unit) and pressure sensor to track the position, orientation, linear velocity, and angular velocity of the

[1] Reference (from work plan)

AUV. This would require proper implementation of a Kalman Filter and calibration of all the sensors used.

A major issue in the current state estimation system is IMU drift. The current state estimation system relies on an off-the-shelf Kalman Filter. In preliminary tests, the state estimation system showed a significant drift of 300 meters in 15 seconds. The test was conducted by keeping the IMU on the Beluga stationary while collecting data for 15 seconds, shown in Figure 2.

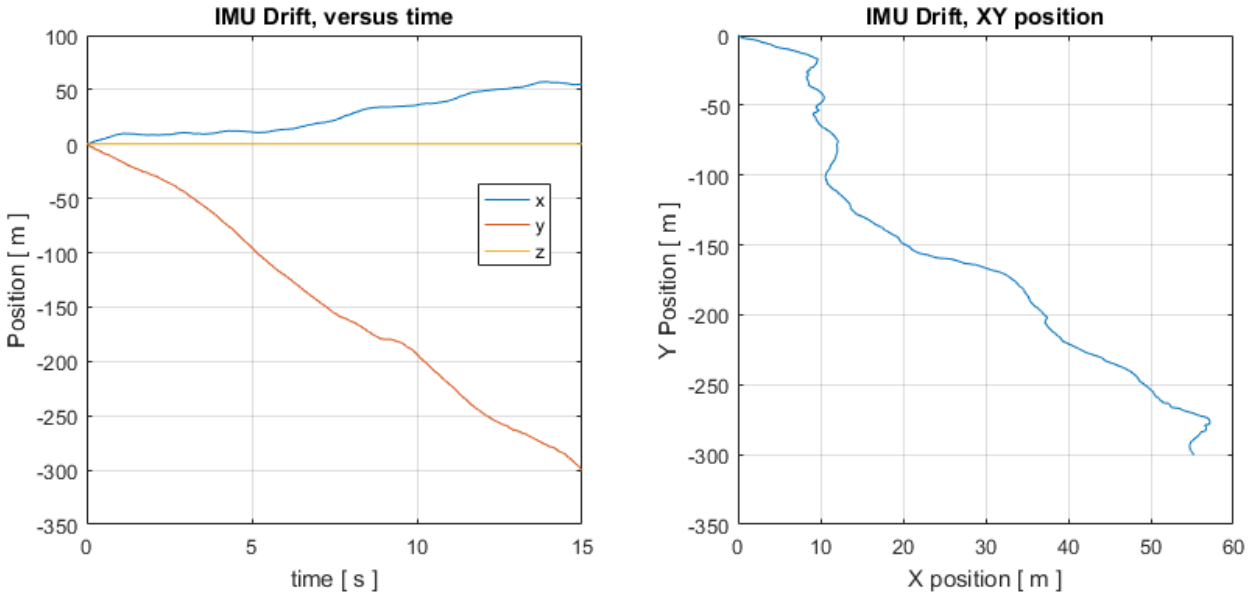


Figure 2: IMU drift data.

To improve the state estimation system and eliminate drift, it is necessary to improve or replace the off-the-shelf Kalman Filter. It is also possible to bound the IMU drift by incorporating other sensors into the system. GPS signals, while effective in giving absolute position data and constraining error within a given radius, cannot be received underwater. Therefore, the team must consider alternative methods to improve the state estimator. Exteroceptive sensors are ideal as they rely on information from the AUV's environment rather

than the current state of the AUV. The exteroceptive sensors that the team considered are discussed in Section 5.1.

3. Impact

The motivation behind this project is to develop a cheaper, more modular, and more versatile AUV than those commercially available today. By implementing a more effective state estimation system on the robot, the team is improving the system to make it comparable to more expensive AUV's on the market. Making AUVs cheaper would also make the technology more affordable and thus more accessible to the general public.

State estimation itself is still an area of active research within robotics, especially within the subfield of underwater robotics. Because these robots have to operate underwater, they cannot use common state estimation sensors such as GPS or radar. Because accurate state estimation is necessary for robots to perform other tasks such as path planning or data collection, this research topic is of extreme importance to the developing Beluga system. Without a precise state estimation subsystem, the robot will be unable to navigate to defined waypoints, know the global location of underwater features, or accurately control its motors underwater. Because of the importance of this system, the team has been tasked with improving the hardware and software components of the Beluga in order to enhance its state estimation.

4. State Estimation

State estimation is a crucial component for any robotics endeavor because at the core state estimation is simply knowing where the robot is and how it is oriented. Though state estimation can be applied to any arbitrary state of the robot, not limited to position and orientation, the Techmation Clinic team is focused on localization of the Beluga AUV. Specifically, the team ultimately aims to measure and control 12 states: xyz-position and velocity, along with roll-pitch-yaw and angular velocity.

Figure 3 illustrates the Beluga and its environment. Currently the Beluga is equipped with the following sensors

- 9 DOF IMU (Accelerometer, Gyroscope, Magnetometer)
- Pressure Sensor
- GPS
- Temperature Sensor

Since the goal is for operation in a GPS-denied environment (underwater), and there is no easy conversion of temperature to any of the states, the Beluga is left with only the IMU and pressure sensor. The issue with this is that there is no direct measurement of the xy-position. The only way to obtain the xy-position is through double integration (or similar) from the acceleration data which introduces an impractical amount of drift error. For outdoor land/air vehicles, the GPS measurements provide absolute position measurements with bounded error.

Thus, the team plans on introducing a new exteroceptive sensor that can provide a better measurement of the robot's position to minimize the drift error.

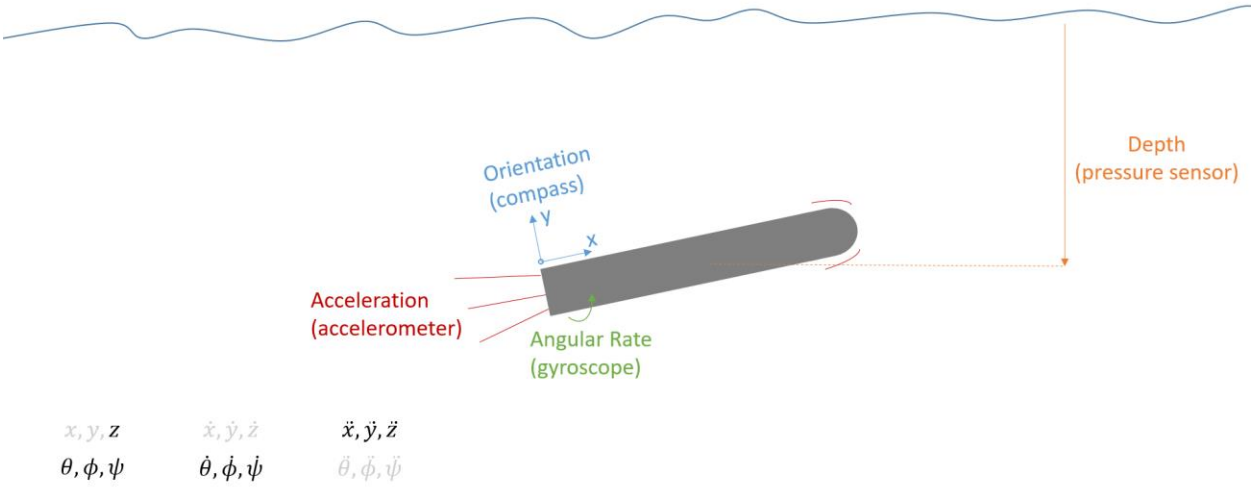


Figure 3: Illustration of AUV environment, with current sensors and states

4.1. Kalman Filter

The industry standard for robot localization algorithms and sensor fusion is the Kalman Filter/estimator. A Kalman Filter produces the optimal estimate of the desired states by leveraging not only the sensor measurements, but also a dynamic model of the system. The dynamic model can be thought of as the predicted behavior of the system given known inputs. For example, if you step on the gas of a car, you would expect it to move forwards, and depending on how hard you step on the gas dictates how far/fast you travel. Thus the dynamic model helps convert these control inputs (stepping on the gas) to a prediction of the states (position, velocity).

The crux of the Kalman Filter is the Kalman gain, which essentially represents the confidence level in either the model prediction or the sensor measurements. This can be illustrated with two scenarios.

Scenario 1: The IMU measures a constant acceleration, resulting in quadratic displacement such that the robot is at 200m after 10s despite never commanding the motors to move.

In this scenario, the IMU is most likely wrong, possibly due to biasing in the accelerometer. Thus the Kalman Filter would recognize that the IMU is inaccurate, and trust the model predictions more.

Scenario 2: The robot receives a command to turn on the motors to move forward, however there is a strong wind and both IMU and GPS do not measure any displacement.

This time it is the model prediction that is most likely inaccurate, since it does not know that the wind may be buffeting the movement. Therefore the Kalman Filter would shift confidence towards the sensor measurements.

The Kalman Filter shifts this confidence rating by keeping track of the errors, then propagating and updating the expected error through each time step. More information on Kalman Filters can be found at these helpful tutorials: Kalman Filter with Pictures[2], Kalman Filter Interactive Tutorial[3].

Ultimately, the team has chosen to utilize an Extended Kalman Filter to provide accurate estimates of the AUV location and orientation. Details on the team's implementation of a Kalman Filter can be found in Section 7.

---

[2] Babb, Tim. "How a Kalman Filter Works, in Pictures"

[3] "The Extended Kalman Filter: An Interactive Tutorial for Non-Experts"

5. Design Alternatives

The team is working towards improving the state estimation system by adding an exteroceptive sensor to the AUV. The sensor measurements will be incorporated into the Kalman Filter. We investigated several exteroceptive sensors, including sonar, camera, and acoustic beacons. The sensors are described in the following subsections.

5.1. Sensor Selection

5.1.1. Sonar

Sonar is a technology commonly used in AUV state estimation applications[4,5,6]. By tracking how features on the ocean floor move with respect to the robot over time, the acceleration, velocity, and position of the robot can be estimated through a process called odometry[7]. Because sonar relies on sound and not light to sense features in the robot's environment, it can sense objects much farther away than a camera can due to the relative impermeability of light through water. However, odometry is a process that accumulates error, meaning that the longer the robot has no ground truth estimate of its state, the more incorrect the odometry estimate is[4]. Another disadvantage of sonar is that the devices tend to be expensive, placing them out of budget for this project since one of the constraints is that the entire project must cost less than $10,000.

---

[4] M.F. Fallon et al., "Efficient AUV Navigation Fusing Acoustic Ranging and Side-scan Sonar"
[5] L. Paull et al., "AUV Navigation and Localization: A Review"
[6] J. Zhou, "State Estimation Strategies for Autonomous Underwater Vehicle Fish Tracking Applications"
[7] K. Chong et. al, "Sonar based map building for a mobile robot"

The primary sonar device considered for this project was the Humminbird Helix 7 pictured in Ffigure 4 below. Because this device only cost $800[8], it was a cost-effective option for this project. However, the Helix 7 was developed by Humminbird to be used by fisherpeople to locate fish or underwater features where fish typically reside, and not to interface with an AUV in real-time. Thus, the team determined that accessing the side scan sonar images from the device for use in real-time state estimation would require time-intensive hardware hacking, as the data was written to an SD card and could not be exported to a computer while the sonar runs. With the only cost-effective option eliminated, the team determined that sonar was not a viable sensor option for this project.

Figure 4: The Humminbird Helix 7 CHIRP SI GPS G2N.

5.1.2. Camera

Cameras are used in visual odometry to determine the position and orientation of an AUV. The process works by taking images of the AUV's surrounds and tracking features in consecutive images to determine the pose of the AUV.

The camera setup could be either monocular (with one lens) or stereo (multiple lenses). We chose the latter case because while a monocular setup might be less expensive, stereo

[8] https://www.humminbird.com/Products/HELIX-7-CHIRP-SI-GPS-G2N/

cameras can obtain absolute distances without the need for a fiducial marker. This reduces the complexity of the setup and allows the camera to function in regions where it would otherwise be impossible to places artificial markers. It is possible to create a stereo setup with two webcams for approximately $100.

Once an imaging system has been setup, there are multiple odometry libraries, such as libviso, that could be used to perform the processes needed to extract the position of a device. These work by calculating the distance moved by features in the images and add that to the last know position of the device. Therefore, while cameras might be very cost effective, they are also subject to drift and are rather unreliable in the long run.

5.1.3. Acoustic Positioning System

Underwater acoustic positioning systems are another method of determining an AUV's position. These systems use acoustic distance measurements from multiple beacons to triangulate the position of an AUV. The distance measurements can provide the AUV's absolute position when coupled with GPS signals to the beacons, to supplement relative positions of the AUV to the beacons. There are three main categories: long-baseline (LBL), short-baseline (SBL), and ultra-short-baseline (USBL).

In LBL systems, baseline transponders are mounted on the seafloor and act as reference points for navigation. The AUV sends out an interrogation signal and receives a response from the beacons. The two-way travel time of the acoustic signal is used to determine the distance between the AUV and beacons. LBL systems are expensive and time-consuming to set up, as they rely on an established network of beacons. The range of the AUV is also limited by the

range of the beacons. However, LBL systems are robust and can be accurate to between 0.01 to 1 meters.[9]

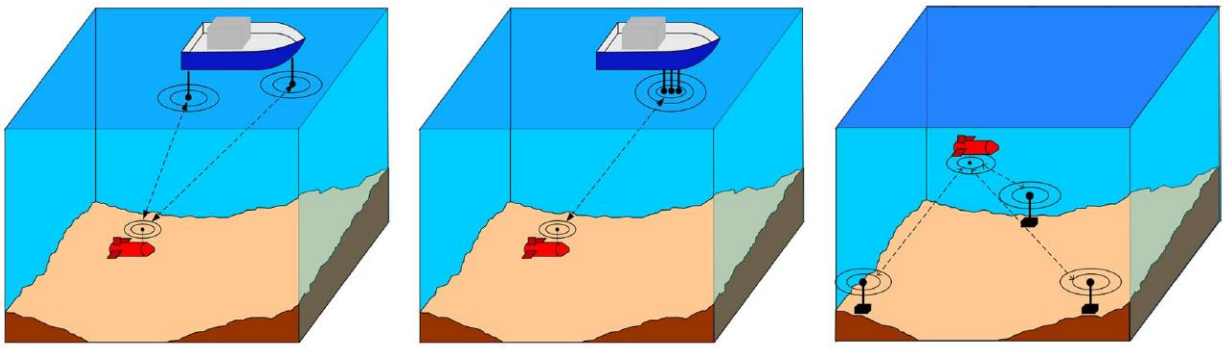


Figure 5: Graphical Depiction of Acoustic Beacon Systems. Right to left: SBL, USBL, LBL. [9]

SBL systems use an array of beacons spread over a short distance, such as from the ends of a ship. The accuracy of the system is reduced as the distance between the beacons decreases.

USBL systems use a single beacon with multiple transducers spaced around 10 cm apart. The position of the AUV relative to the beacon is calculated using time of flight measurements, and the orientation of the AUV is calculated using the phase differences of the signal arriving the multiple transducers. USBL systems are limited by the range of the single beacon and are less accurate than LBL systems.

Between these options, we chose to pursue USBL solutions as they are the most cost effective. A LBL system would require installation of multiple beacons on the seafloor--the added benefit of increased accuracy does not outweigh the installation costs. Similarly, a SBL system would not significantly increase accuracy, as the Beluga AUV is only a few feet long.

Acoustic beacons achieve localization by using time of flight measurements, which can be two-way-travel-time (TWTT) or one-way-travel-time (OWTT). In TWTT systems, the AUV

[9] En.wikipedia.org. (2017). Underwater acoustic positioning system.

sends an interrogation signal to the beacon, which responds with a signal that the AUV processes. This requires the AUV to have an active acoustic system, which consumes power and adds cost. In OWTT systems, the beacon and the AUV are time-synchronized such that only the beacon needs to send a signal to the AUV. OWTT systems require less power, but need high-resolution clocks on the beacon and AUV.

Acoustic positioning system solutions have bounded error in their localization measurements, since each measurement is taken with respect to the position of the beacon rather than relying on previous measurements. This is a huge benefit to decreasing drift over time, as the sensor will not accumulate error.

5.1.4. Other Options

The team also considered alternative sensor solutions which were dismissed early on for various reasons.

5.1.4.1. Moving Beacon

A stationary acoustic beacon limits the operation range of the AUV. This is a major drawback to introducing a separate system that is not onboard the AUV, compared to a camera or sonar. However, if the surface beacon could move around on its own, then the range of operation would be unbounded.

Specifically, the buoyed beacon could have its own propulsion system and receive GPS signals. Localization of the buoy itself is not required, since the GPS provides location with bounded error. Depending on the desired use, the AUV-beacon could have two way communication, allowing the beacon to be very simple, and simply transmit its current location. Using the same methods of ranging, the AUV would then deduce its location relative to the beacon, and thus the global position. Communication could be done through special encoding in

the acoustic signals, such as frequency modulation, phase modulation, or others. More research would be required if the team decides to pursue this approach.

Adding mobility to the beacon overcomes the problem of operation range, but clearly increases complexity of the system. Due to the time constraint of this project, the team has decided not to implement this design. However, this illustrates the expansion possibilities of the acoustic beacon sensor, and could be explored in later years.

5.1.4.2. Tethered Beacon

The critical obstacle to creating a network of nodes between the AUV and the beacon is communication. Water is a difficult medium to deal with, since the usual communication methods in air do not work (radio waves, wifi, etc.). Acoustic communication is very common underwater, but it is limited in bandwidth (< 100 kHz).[10] Additionally, implementing any wireless communication would require significant signal processing, such as analog filtering, peak/edge detection, convolution etc.

One option to overcome the difficulties of underwater communication is tethering the beacon to the AUV. This would allow for drastically higher data rates with no need for any signal processing. The beacon could then transmit data as usual through air using conventional methods such as WiFi or Bluetooth. The beacon could potentially even stream live video from a camera on the AUV.

How is this different from a remotely controlled underwater vehicle (ROV)? Although there is a feature to remotely control the AUV, the AUV itself is still capable of autonomous

[10] Kaushal, Hemani. "Underwater Optical Wireless Communication."

navigation. If the tethered beacon were also mobile, the AUV could navigate itself with no need to resurface and dump data or reorient.

The main drawback to this approach is the feasibility of tethering the beacon to the AUV. Tethering the AUV limits the depth of operation, but a long wire could get tangled or caught on objects in the water (or eaten).

The team believes tethering the AUV is rather impractical, and thus plans on investigating the regular acoustic beacon first. If communication proves to be too great an issue, this approach may be reassessed.

5.1.4.3. Optical Communication

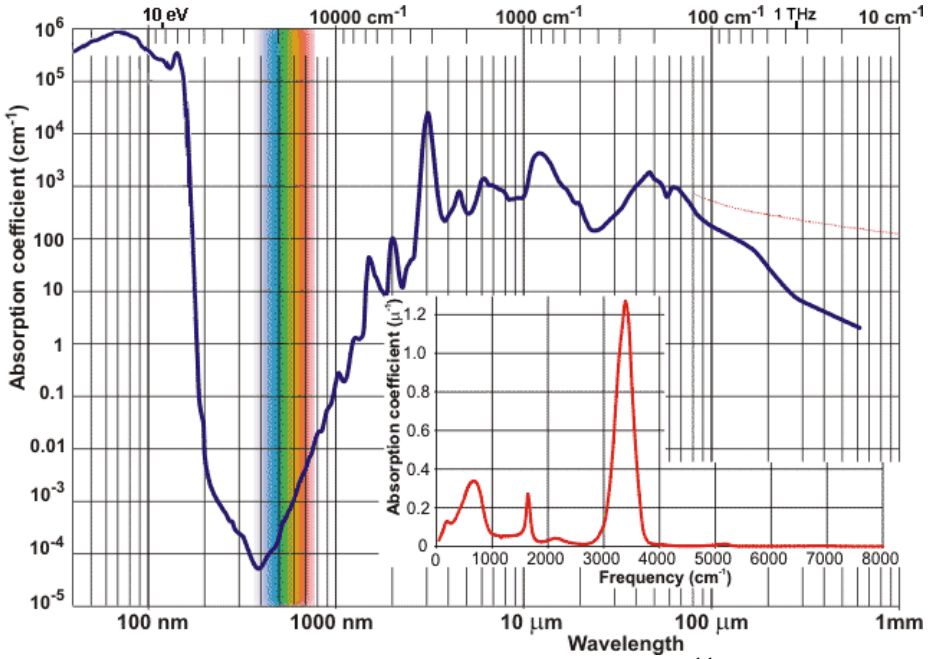


Figure 6: Absorption of light in water[11]

A new area of research in underwater wireless communication uses optics, specifically blue light. Across the electromagnetic spectrum, blue light has the best transmission (lowest

[11] Chaplin, Martin. "Water Absorption Spectrum."

absorption rate) in water. On the other hand, as mentioned before, infrared and radio waves are greatly absorbed underwater: microwaves use this property to heat food.

Using optical communications can lead to drastic improvements, with Sonardyne's BlueComm 200 advertising 12.5 Mbps up to 150 m.[12] Since commercial products are very expensive, the team is would need to implement a custom communication system. However, the feasibility of creating the system would be unrealistic given the time span, and there are also extensive OSHA safety regulations for high power lasers that would complicate the development.

5.1.4.4. Salinity, Temperature, and Turbidity

The team has also researched other various sensors such as temperature, salinity, and turbidity sensors, but there is no indication that a location could be reliably determined from the information. Instead, these sensors could be useful in calibrating for the speed of sound if acoustic communication is used.

5.1.5. Sensor Comparison

To determine which sensors to test in the fall, the team compiled characteristics of the most feasible sensor alternatives, as shown in Table 1.

Product	Sensor Type	Approximate Cost	Implementation Difficulty	Bounded Error?
HELIX 7 CHIRP SI GPS	Sonar (cheap)	$800	Hard	No

[12] Control, B. (2017). BlueComm 200 - Wireless Underwater Video and Vehicle Control - Sonardyne.

G2N				
Hydro Box HD	Sonar (expensive)	$6,000	Medium	No
Logitech C920 Pro Webcam	Camera	$100	Easy	No
HTI-96-Min Hydrophone, UW30 Underwater Speaker	Acoustic Beacon	$2,000	Medium	Yes

Table 1: Sensor Comparison Table

The team considered each sensor alternative's cost, difficulty of implementation, and drift. While the sensors have other important characteristics, such as the range of the sensor, we considered these factors to be the most critical. The cost of the sensor has a hard limit of $10,000, which ruled out blue laser optical communication options. Whether the sensor has unbounded error is another important factor that determines how much the exteroceptive sensor can improve the current state estimation. Sensors with unbounded error will drift significantly over time, as shown by the IMU drift in Section 2, so a sensor with bounded error is ideal. The difficulty of implementation is a rough estimate of the feasibility of implementing the sensor.

From these metrics, the team chose to pursue the acoustic beacon solution. The cheap sonar option was ruled out because it would have been too difficult to reverse engineer the product to obtain raw sensor data. The more expensive sonar, while easier to interface with, is significantly more expensive than the other options. The camera sensor option is the easiest to implement and performs similarly in comparison to the sonar. The major drawback of the camera when compared to the sonar is the necessity of clear water to function. Therefore, the expensive sonar solution is ruled out in favor of the cheaper and easier to implement camera option. We chose the acoustic beacon over the camera because it is the only sensor with bounded error, and its cost and difficulty of implementation are not prohibitively high.

While a drawback of the acoustic beacon is the fact that the AUV can only travel within range of the signal from the beacon, the acoustic beacon solution can be improved upon to allow the beacon to move freely above water, as described in Section 5.1.4.1. Because of the low cost and ease of implementation of the camera, the team is considering implementing a camera on the AUV in conjunction with the acoustic beacon, if time permits.

5.2. Pipeline Design

For the one-way ToF acoustic beacon system, a data collection and processing pipeline is necessary to convert raw hydrophone signals to time of flight data. This section covers the different hardware alternatives to acquire and process the hydrophone data. Several factors were considered, including:

1. Cost
2. Size: The pipeline must easily fit within the interior of the Beluga.
3. Sample rate: The sample rate directly corresponds to the time granularity of the measured ToFs.
4. Timing: Uncertainty and drift in time directly correspond to spatial uncertainty and drift respectively
5. Engineering effort

5.2.1. Off-the-shelf DAQs

The team investigated numerous data acquisition units (DAQs) as means of collecting the hydrophone data. Most DAQs were desirable with respect to engineering effort, but the team did not identify any DAQs which could satisfy all the previously identified metrics.

The Dataq Model DI-1110 was purchased and tested extensively as it was cheap, compact, and capable of sampling two hydrophones at 80kHz. However, it had a low-pass

frequency response with a cutoff frequency around 7kHz, which is below the frequency range for the chirp. Additionally, this DAQ was unable to keep time within the required specifications and had no way to interface with an external clock. These shortcomings eliminated it from consideration.

5.2.2. Teensy 3.6 + Raspberry Pi

The team identified microprocessors with ADCs as cheaper, smaller alternatives to DAQs. Cheap microprocessors had similar performance characteristics to commercial DAQs, which costed thousands of dollars. Therefore, the team decided to investigate pipeline designs using either the Teensy 3.6 or the STM32 Nucleo, two microprocessors with which the team and mentors had experience. A challenge with this approach was engineering effort. Significant development time and embedded programming experience was the tradeoff for superior performance in metrics 1 through 4. Ultimately, the Teensy 3.6 was selected due to its simpler interface with an external clock.

Neither microprocessor had sufficient memory to process data onboard, so it was necessary to transfer the raw data for processing. There was no reliable way to transfer raw data to the Intel NUC, so a Raspberry Pi was added between the Teensy and NUC to facilitate communication. The Teensy and Pi would communicate via SPI. The Pi would then process the data to detect the chirp waveform and calculate ToF.

The use of a microprocessor over a DAQ provided great flexibility for the pipeline's capabilities, and may in the future allow for such improvements on the system as adjusting the sampling timing and length based on the robot's state. Therefore, the additional flexibility granted by a microprocessor may add to the sensor's range and robustness.

6. Sensor Implementation

Section 6 describes sensor implementation in depth, including the Acoustic Positioning System with the implemented hardware and software.

6.1. Acoustic Positioning System (APS) Details

The team looked at various research papers and products to determine what type of hydrophone and underwater speakers to obtain. The team chose to use the HTI-96-Min hydrophone with a frequency response of 2 Hz to 30 kHz and a maximum operating depth of 10,000 feet, which is more than enough for our purposes.[13] The dimensions of the hydrophone are 2.50" length x 0.75" diameter, which is small enough to fit on a rig attached to the AUV. For the underwater speaker the team is using the Electro-Voice UW30, which has a frequency response of 100 Hz to 10 kHz.[14] The underwater speaker delivers uniform sound up to a 30 x 30 feet area, which is sufficient for testing.

The team chose to use a linear up-chirp for the signal. Linear up-chirp signals are sinusoidal and linearly increase in frequency over time. A sinusoidal wave is preferable to a square wave, as the sudden ramp-up of a square wave could damage the speaker. Linear up-chirp signals are commonly used in sonar and radio applications and are useful for matched filtering.

In a linear chirp, the instantaneous frequency *f(t)* varies linearly with time, so

---

[13] "HTI-96-Min Hydrophone." High Tech, Inc. USA, High Tech Inc, www.hightechincusa.com/products/hydrophones/hti96min.html.

[14] "EV UW30 Underwater Speaker for Marine Biology." Lubell Labs Inc. http://www.lubell.com/UW30.html

$$f(t) = f_0 + kt, k = \frac{f_1 - f_0}{T}$$

where f_o is the initial frequency, k is the rate of change in frequency, f_1 is the final frequency and T is the length of the signal. The chirp is characterized by the equations:

$$\phi(t) = \phi_0 + 2\pi \left(f_0 t + \frac{k}{2} t^2 \right)$$

$$x(t) = \sin \left[\phi_0 + 2\pi \left(f_0 t + \frac{k}{2} t^2 \right) \right]$$

where ϕ_0 is the initial phase. We are using a 10 ms chirp with a frequency of 7 to 10 kHz. The signal is pre-recorded and saved as a .wav file, which is played off a computer connected to the underwater speaker.

To determine the distance between the AUV and the beacon, the team used matched filtering with measurements from each of the beacons. A matched filter can be obtained by cross-correlating the received signal $y[t]$ with the transmitted signal $x(t)$. In discrete-time, the cross-correlation function is

$$\phi_{xy}[n] = \sum_{-\infty}^{\infty} y[n+k]x[k]$$

For time of flight t_0, sampling frequency F_s, and $n_0 = F_s t_0$,

$$\phi_{xy}[n_0] = max(\phi_{xy}[n])$$

and the distance between the AUV and the beacon can be found through the equation

$$d = ct_0 = \frac{cn_0}{F_s}$$

where c is the speed of sound in water. The speed of sound in water is typically around 1500 m/s.[15]

The team chose to implement a one-way-travel-time (OWTT) system. The OWTT only requires one signal source and two hydrophones, removing the need to provide power for a speaker on the AUV.

The two hydrophones are mounted to the head and the tail of the Beluga. Each of hydrophones provides a distance estimate, which results in a circle of possible locations for each hydrophone as shown in Ffigure 7. In the figure, r_1 is equal to the calculated distance between the acoustic beacon and the head hydrophone, while r_2 is equal to the distance between the beacon and the tail hydrophone. Since the distance between the two mounted hydrophones is known and the magnetometer sensor can provide the heading for the AUV, there are only two possible locations for the AUV. The measurements from the IMU are then used to eliminate one of the possible locations.

In a 3D system, it would be necessary to have four hydrophones to provide an exact location for the AUV, just as four satellites are necessary for GPS localization. Each of the hydrophones provides a distance estimate, which is the radius of the sphere of possible hydrophone locations. The distance between each of the hydrophones relative to each other is known, and the heading of the AUV in 3D space is provided by the magnetometer, so there is only one possible location for the AUV.

[15] Elert, Glenn. "Speed of Sound in Water." The Physics Factbook, hypertextbook.com/facts/2000/NickyDu.shtml.

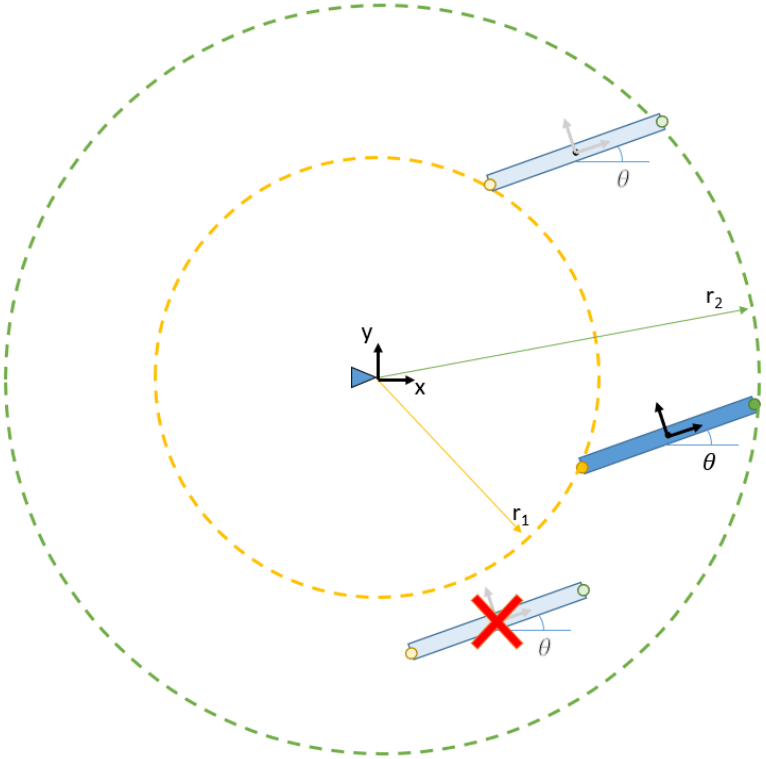


Figure 7: 2D localization with 2 hydrophones.

6.2. Pipeline Hardware

The figure below gives an overview of the implemented hydrophone pipeline. The pipeline ensures that data from the two hydrophones is collected, processed, and transferred to the main Beluga computer, which then combines all sensor measurements with the Kalman Filter.

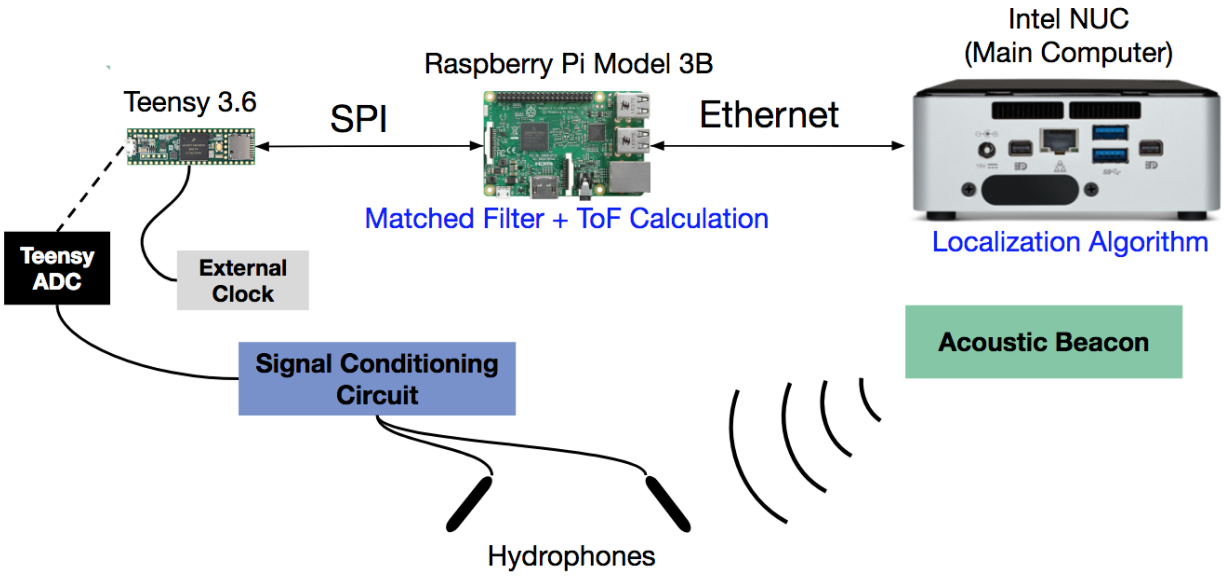


Figure 8Fig: Hydrophone pipeline overview

The acoustic beacon first sends a 10 ms chirp at a frequency of 2 Hz, or twice a second. The Teensy microcontroller samples the chirp data from the hydrophones at the start of every second for 40 ms. The microcontroller only samples once per second even though the chirp itself occurs twice a second because pipeline uses the remaining time to process the collected data. To ensure that the sampling window captures the signal in the first sample, the microcontroller has an initialization step in which it samples data for 500 ms. The chirp occurs twice a second

because the microcontroller can only store 500 ms of data and does not have enough space to store a full second of data. This initialization step is described in more depth in Section 6.3.1.

A signal conditioning circuit is used to ensure the peak-to-peak voltage of the hydrophone signal is within the 0-3.3V tolerances of the Teensy analog input pins. The Teensy is connected to an external oven-controlled crystal oscillator that allows it to sample in sync with the 2 Hz acoustic beacon signal.

Once the Teensy has sampled for that short window, it passes the raw hydrophone data to the Raspberry Pi, which applies the matched filter algorithm to extract time of flight information. The time of flight measurements are then transmitted through Ethernet to the main Beluga computer (an Intel NUC). The Beluga computer then applies the Extended Kalman Filter, which combines all the relevant sensor measurements to provide a location estimate of the Beluga.

6.2.1. Signal Conditioning Circuit

Each hydrophone is powered by a 12V output from the Beluga and outputs a maximum signal with a 12V peak to peak value and a varying DC offset. Without this offset, the signal oscillates between +6V and -6V.

In order to have the output from the hydrophones meet the specifications of the Teensy the team built a conditioning circuit for each hydrophone, as shown in the figure below.

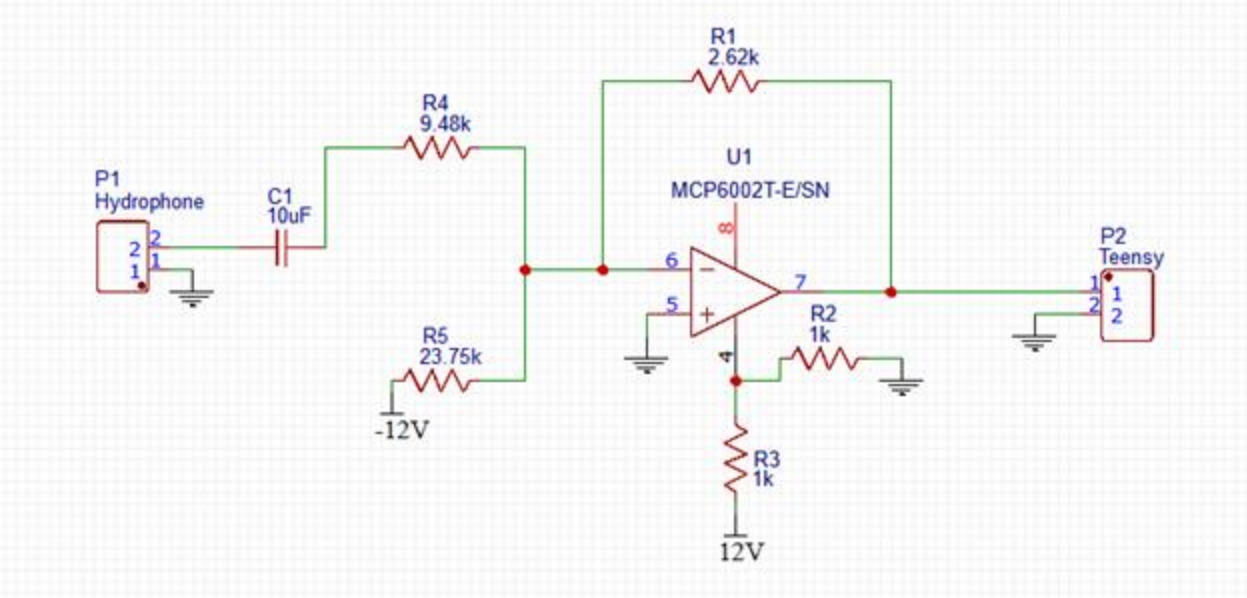


Figure 9: The schematic for the signal conditioning circuit

The capacitor removes the aforementioned DC offset, and the rest of circuit acts as a summing amplifier to offset and scale our signal to between 0V and 3.3V.

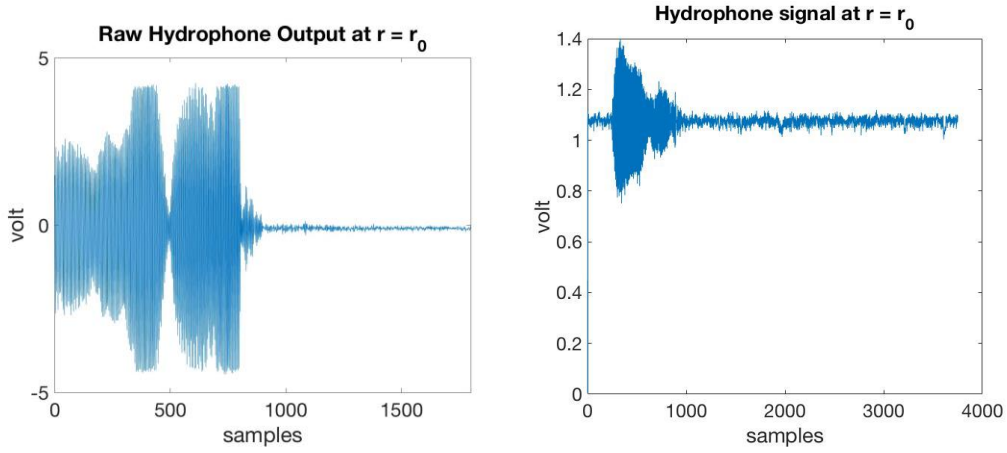


Figure ??. Unconditioned vs conditioned hydrophone signal.

6.2.2. External Clock Specifications

The crystal oscillator used is the Connor-Winfield OH100-61003CF-010.0M[16]. It requires a 3.3V power supply, and it operates between -40°C ~ 85°C at a 10MHz frequency. It has a 10 ppb precision which corresponds to a drift of 50 mm/hr. The clock ensures that the teensy stays in sync with the hydrophones, which is required in a one way travel time system.

6.2.3. Teensy Hardware

This section describes some of the Teensy's hardware specifics and how they influence design aspects of the pipeline.

The Teensy has 25 analog input pins with 13-bit resolution. The maximum sampling rate on the Teensy ADC is 125 kHz.[17] To get the best possible location resolution, the hydrophones should be sampled at the highest rate possible. Since the Teensy is using one ADC to sample from both hydrophones, the sampling rate on the Teensy ADC is set to 125 kHz / 2 = 62.5 kHz. This sampling rate could be increased in future iterations of the hydrophone pipeline by decreasing the resolution on the ADC, or configuring the Teensy to use both of its two ADCs in parallel.

Although the Teensy has a built-in clock that governs its ADC sampling frequency, it is far too inconsistent to be used in this application. In order to get a highly precise and consistent sampling frequency, the Teensy needs a reference to a more precise clock. The Flexible Timing Module, or FTM, on the Teensy can be used to integrate an external clock. The FTM is a 16-bit counter module on the Teensy that uses an external clock input to generate a pulse width

[16] "OH100-61003CF-010.0M." www.digikey.com/short/j4chr8.

[17] "Fast sampling ADC." https://forum.arduino.cc/index.php?topic=65192.0

modulated signal.[18] By triggering the teensy with the new pulse width modulated signal, the FTM controls the ADC such that it samples at the desired 62.5 kHz. An interrupt service routine is used to sample from the ADC on every rising edge of the signal.

In addition to keeping sampling frequency consistent, the Teensy also needs to keep track the 2 Hz hydrophone signal so that it samples at the right time. Therefore, with the aid of the 10 ppb external clock, the FTM is also used to ensure that the Teensy starts sampling at the beginning of every second. It will count up to one second worth of ticks and raise a FTM READY flag when a second has elapsed, which triggers the Teensy to start sampling for the acoustic chirps.

6.3. Pipeline Software

The hydrophone pipeline requires communication between the Teensy and the Raspberry Pi, as well as between the Raspberry Pi and the main Beluga computer. State machines are implemented on the Teensy and the Raspberry Pi to enable these components to process data and communicate with each other. Communication between the Teensy and the Raspberry Pi is handled via SPI, with the Raspberry Pi being the SPI master. Communication between the Raspberry Pi and the main Beluga computer is handled over an ethernet cable.

6.3.1. Teensy State Machine

The Teensy has a set of states that allows it to interface with the Raspberry Pi. The purpose of the states is to help the Teensy sample from the hydrophones at the correct times and

[18] *Features of the Flexible Timing Module*. Freescale Semiconductor, Inc. 2015. https://www.nxp.com/docs/en/application-note/AN5142.pdf

coordinate SPI communication with the Pi. Figure 10 below presents an overview of the various states on the Teensy and the flags that transition the states.

An important aspect of the Teensy State Machine is its ability to update the sampling window of the Teensy. The Teensy is not initially synced to the chirp from the acoustic beacon. Therefore, it has a initialization sequence to help locate the acoustic chirp. On startup, the Teensy will first sample for about 500 ms. The chirp occurs twice a second, so sampling for 500 ms ensures that the entire signal will be sampled in that window. It will transmit the data over the Pi and the Pi will calculate the offset and send it back to the Teensy. Upon receiving the chirp, the Teensy then updates the next sampling window to ensure that the chirp is captured within the 40 ms sampling window. The Teensy then enters a looped sequence that is illustrated in Ffigure 11.

A description of each of the states is listed below.

- ***Synchronization***: On startup, the Teensy always begins in the synchronization state. This state is visited once per loop. This state ensures that the SPI communication between the Teensy and the Raspberry Pi is working correctly. The Pi sends a known sequence to the Teensy, which responds with the same sequence. If the Pi receives the wrong sequence, it will adjust accordingly.
- ***Ready to Sample***: In this state, the Teensy waits until the FTM READY flag is set. The flag is set when a full second has elapsed since the last sample.
- ***Sampling***: In this state, the Teensy samples data from the two hydrophones. If the init flag is set, the Teensy samples about 500 ms worth of data to ensure the chirp is captured within the sample. Otherwise, the Teensy assumes the sampling window is synced already, and samples hydrophone data at a rate of 62.5 kHz for 40 ms.

- ***Dump Data***: The Teensy transfers the sampled hydrophone data to the Raspberry Pi during this state. If the init flag is set, it transfers 500 ms worth of data. Otherwise, it transfers 40 ms worth of data.
- ***Wait for Time Delta***: The Teensy waits for the Raspberry Pi to process the hydrophone data. The Teensy sets a pin high to indicate to the Raspberry Pi that it is ready to receive for the time delta that the Pi calculates from the given data. The Teensy then adjusts its sampling window using the time delta.

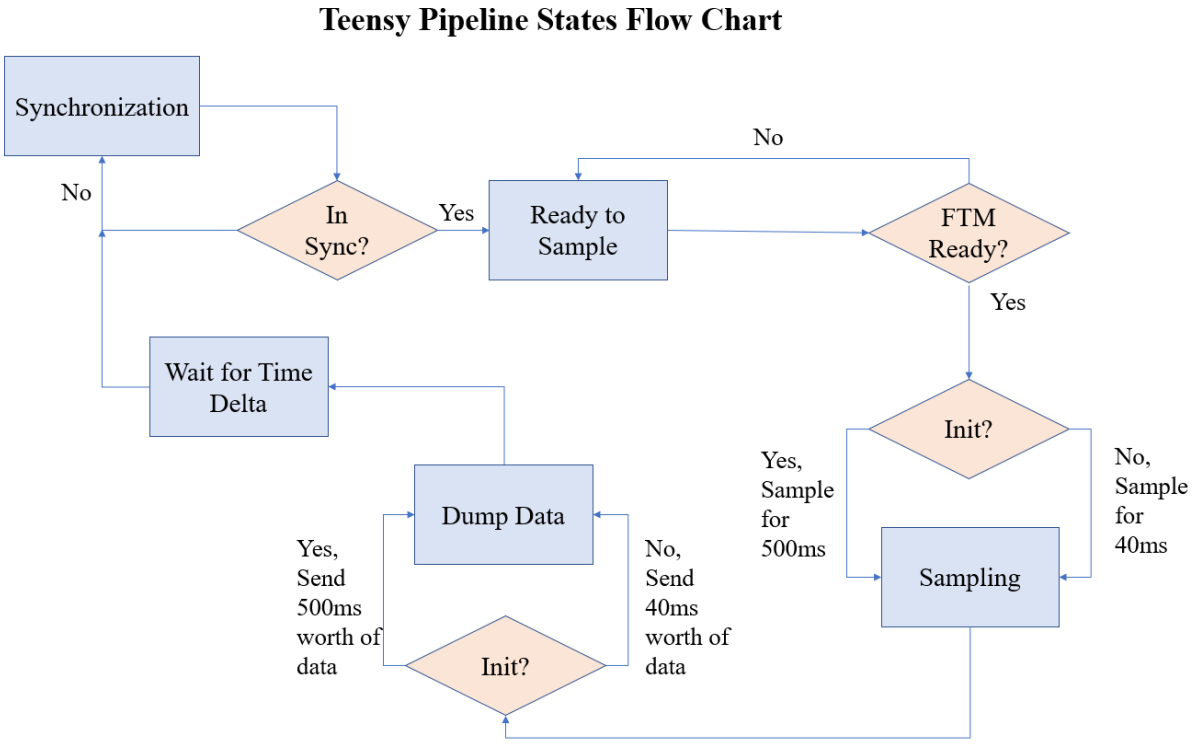


Figure 10: Teensy Pipeline Flow Chart

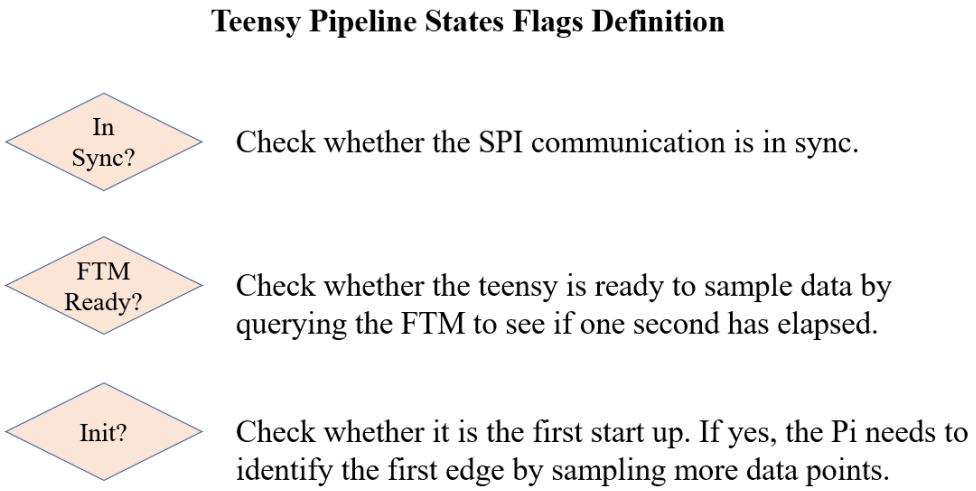


Figure 11: Teensy Pipeline Flag Definition

6.3.2. Raspberry Pi State Machine

As on the teensy, the raspberry Pi also has a sets of states that it follows to correctly interface with the teensy. Figure 13 below lists all the states (box) that the Pi has and the flags (diamond) that directs the program flow. Figure 12 lists all the flags and their definition. The details of the states are summarized in the following list:

- ***Synchronization***: At startup, the Pi always begin in the synchronization state. This state is visited once per loop. It checks whether the SPI is working properly by sending a known checking sequence to the teensy, and the teensy responds with the same sequence. If the checking sequences don't match, the Pi will adjust until the checking sequence agrees.
- ***Wait for Data:*** Sampling a window of data points on teensy requires time, and when it completes sampling it will set a GPIO pin to high on the Pi. Before the read ready pin is high, the Pi will stay in this state indefinitely until the pin becomes high.

- ***Read Data:*** During this state, the Pi will check the init flag to see if this is the first start up. If it is, the Pi needs to identify the first chirp by sampling 500 milliseconds worth of data. If it is not, it will sample only 40 milliseconds worth of data.
- ***Process Data:*** Once the Pi finish reading the data from the teensy, it will run the match filter on the data to determine when the signal arrives. Then it will find the time difference between the arrival time and the center of the sampling window. The goal is to always keep the chirp at the midpoint of the 20 milliseconds window so that the Pi doesn't lose track of it. If this data sample is 500 milliseconds, the resulting time delta will be used to help the teensy determine when to start sampling.
- ***Wait to Send Time:*** The Pi needs to wait for the teensy to be ready to receive the latest time data. The teensy will set a corresponding pin on the Pi to high if it is ready, and the Pi will stay in this state indefinitely until the teensy has turn the pin on.
- ***Send Time:*** In this state, the Pi sends the newly calculated time delta to the teensy and then return to the synchronization phase to double check on connection.

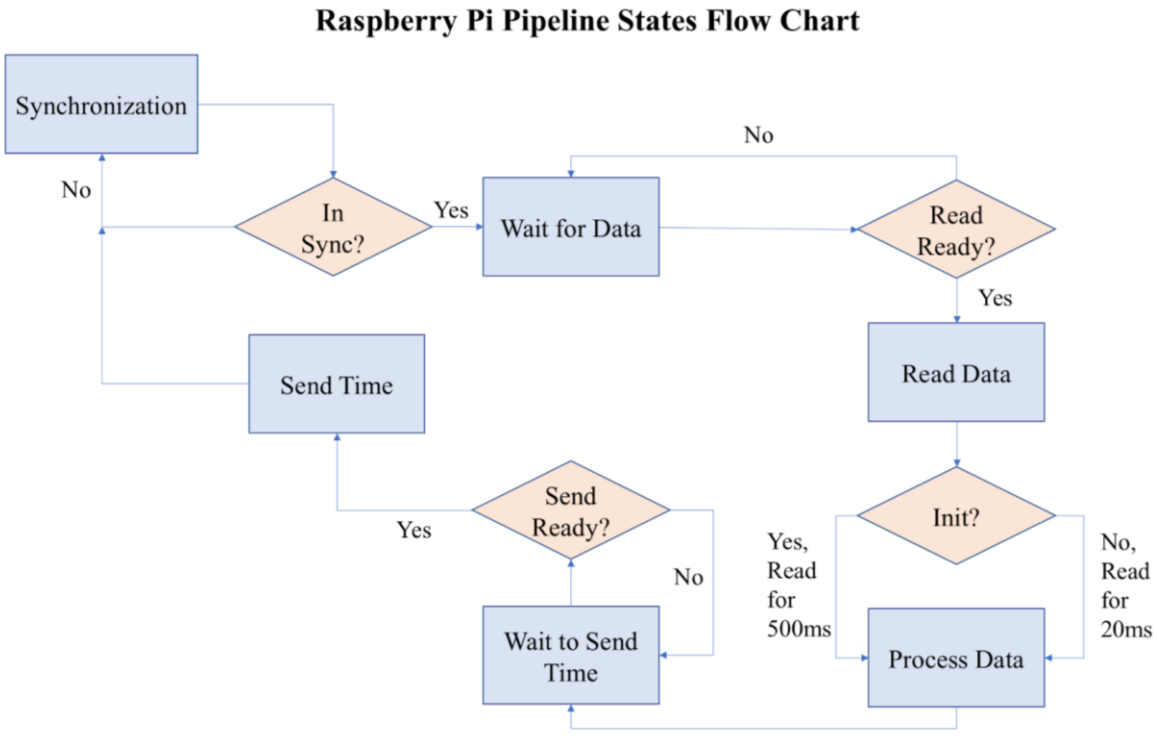


Figure 12: Raspberry Pi Pipeline Flow Chart

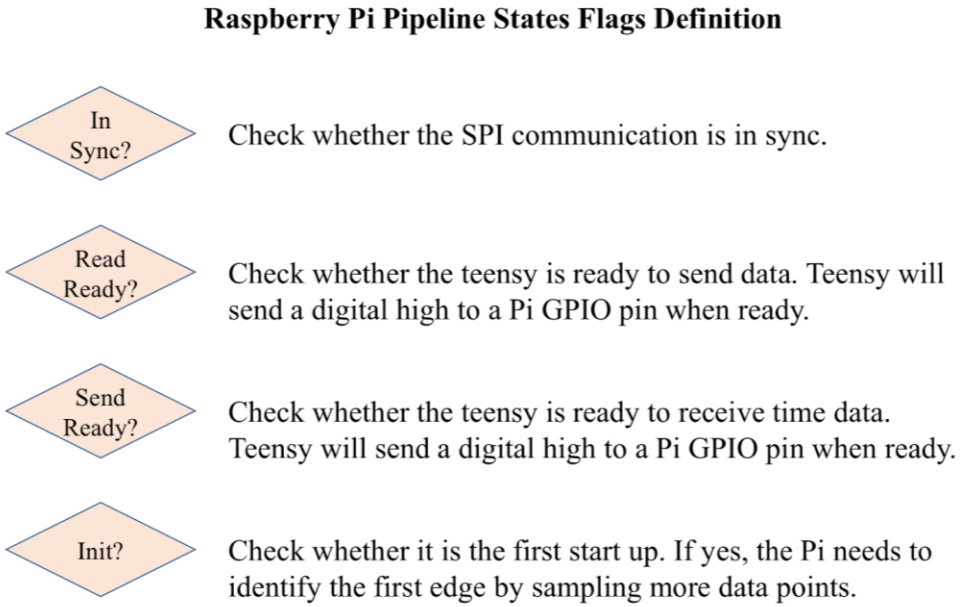
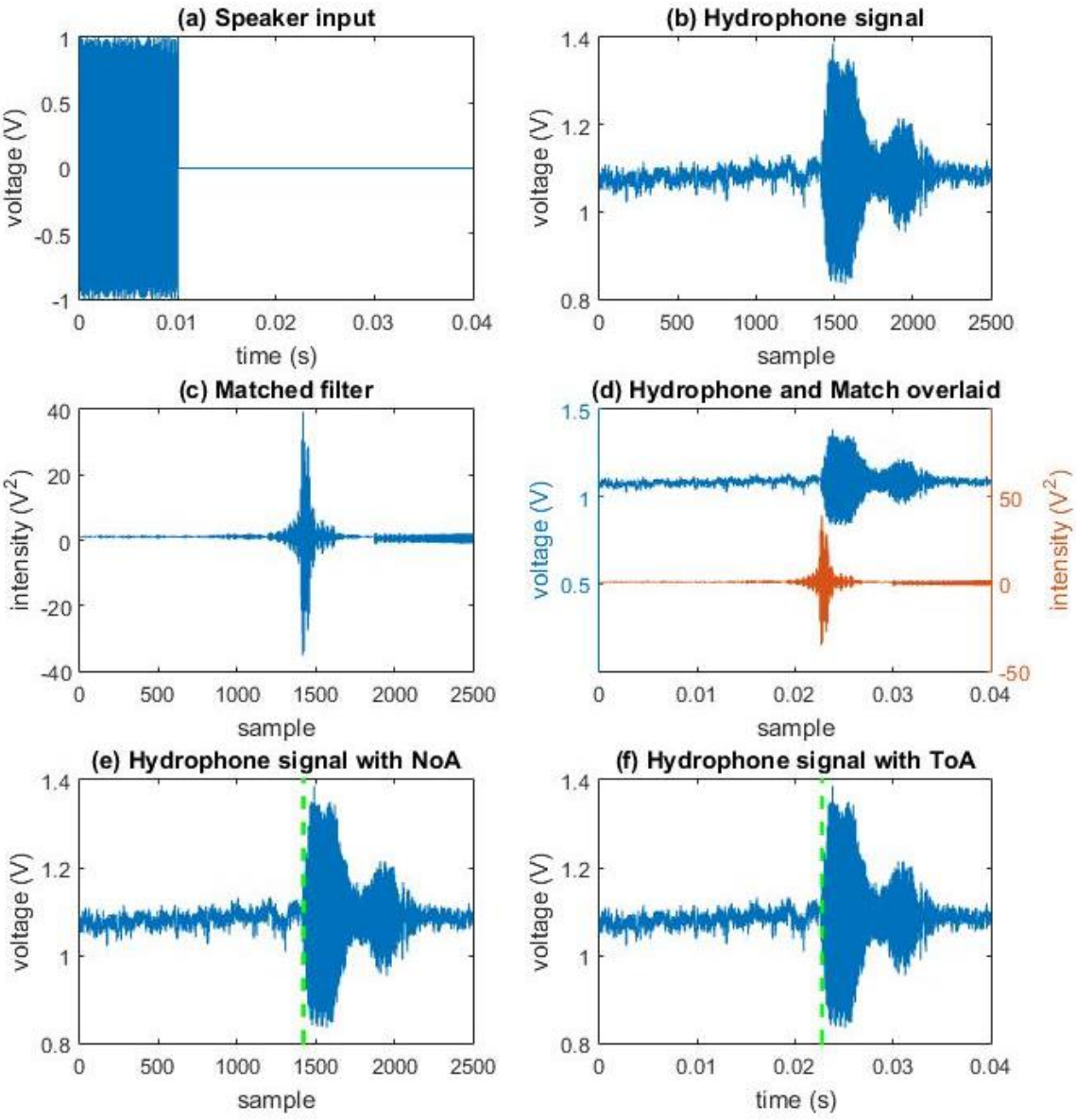


Figure 13: Raspberry Pi Pipeline Flag Definition

6.3.3. Determining Time of Arrival

As has been mentioned in this section, the Raspberry Pi is responsible for processing the hydrophone data. Much of this processing involves extracting time of flight measurements using the matched filter, as is shown in Figure 14. The speaker input (a) and the hydrophone signal (b) are autocorrelated to produce the matched filter output, shown in (c). The index of arrival occurs when the matched filter is maximized, as shown in (d). In (e), the index of arrival is shown to correspond with the front edge of the chirp, and (f) shows the conversion of index to time of arrival by dividing by the sampling frequency. From time of arrival, time of flight and distance can be calculated trivally.



Ffigure 14: Hydrophone data processing to extract time of arrival

7. State Estimation Algorithm

As mentioned before, the team used an Extended Kalman Filter for the localization algorithm. This leverages both predictions of the system and corrections from sensor measurements. Section 7 describes the Extended Kalman Filter in depth.

To denote certain states, the following variables will be used:

Variable	Description	Units
x, y, z	3D position in Cartesian coordinates	m
v_x, v_y, v_z	Linear velocity in the respective direction (x,y,z direction)	m/s
${}^{G}a_x,\ {}^{G}a_y,\ {}^{G}a_z$	Linear acceleration in the respective direction in the global (inertial) frame	m/s^2
${}^{R}a_x,\ {}^{R}a_y,\ {}^{R}a_z$	Linear acceleration in the respective direction in the robot (local) frame (Accelerometer data is in the robot frame)	m/s^2
θ, Φ, Ψ	Angle about the z-axis (in the xy-plane), y-axis, and x-axis respectively	rad
$\omega_x, \omega_y, \omega_z$	Angular velocity about the respective axis	rad/s
r_f, r_b	Range to the front/back hydrophones	m
ToA	Time of Arrival: Actually treated as range measurement (assume conversion of time to distance). Specified as distance from beacon to front hydrophone (equal to r_f)	m
$TDoA$	Time Difference of Arrival: Similar to ToA, treated as range measurement. Specified as the Difference between the range measurements ($TDoA = r_f - r_b$)	m
u_R, u_L	Motor commands to left/right motor	N

Table 2: Nomenclature for EKF formulation

7.1. Kalman Filter Algorithm

The following equations depict the linear model, the prediction step, and the update step of a linear Kalman Filter.

Model:

$$x_k = Ax_{k-1} + Bu_k$$

$$z_k = Cx_k + v_k$$

Predict:

$$\hat{x}_k = A\hat{x}_{k-1} + Bu_k$$

$$P_k = AP_{k-1}A^T + Q$$

Update:

$$G_k = P_kC^T(CP_kC^T + R)^{-1}$$

$$\hat{x}_k \leftarrow \hat{x}_k + G_k(z_k - C\hat{x}_k)$$

$$P_k \leftarrow (I - G_kC)P_k$$

Figure 15: Linear Kalman Filter Equations[19]

□ = states of the AUV (□, □, □, □, □, □, □, □, □, □, □, □)
□ = control inputs (motor commands)
□ = system matrix (linearized model of the Beluga)
□ = control matrix (transformation of control inputs to the states)
□ = sensor matrix (transformation of states to compare with sensors)
□ = prediction covariance matrix
□ = Kalman Gain
□= covariance of sensors (sensor noise)
Q = process noise covariance (inherent noise of the system)

The important point to keep in mind is that there are two main steps for a Kalman Filter:

1) *Prediction*: leverage knowledge of the system
2) *Correction*/*Update*: incorporate sensor measurements

7.2. 1D Line Test

In the first semester, the team developed a 1D Kalman Filter (linear position and velocity) to get started with the Kalman Filter. Real-time data was logged as the Beluga was wheeled with a table in a straight line. This was merely a starting point of the Kalman Filter

[19] https://home.wlu.edu/~levys/kalman_tutorial/

development and served as an introduction to the Kalman Filtering. More details of this specific implementation can be found in the Appendix.

7.3. 2D Localization and the Extended Kalman Filter

The logical next step after implementing the 1D Kalman Filter is to implement a 2D Kalman Filter that can localize the AUV in a 2D plane. However, localizing the AUV in a 2D plane will make the system nonlinear, as translating the IMU reading from robot frame to global frame required trigonometry. Furthermore, the drag in the water will also make the AUV response nonlinear. Since a Kalman Filter is only suitable for linear systems, an Extended Kalman Filter (EKF) has to be used in place of the Kalman Filter.

The EKF differs from the Kalman Filter in that the system model is non-linear. For the regular Kalman Filter, the prediction step is linear and thus can be written in matrix form as follows. The following equation updates the state space vector *x* with matrix *A* and accounts for system input *u* with matrix B. Assuming that the sensor measurements also has a linear relationship with state space *x*, the matrix *C* is used to transform sensor measurements into the state space vector $\vec{x}$.

$$\vec{x}_k = A\vec{x}_{k-1} + B\vec{u}$$

$$\vec{z}_k = C\vec{x}_k$$

whereas an Extended Kalman Filter allows for nonlinear systems uses the general prediction function *f* and sensor transformation function *h* to transform *x* into sensor measurements *z*.

$$\vec{x}_k = f(x_{k-1}, u_k)$$

$$\vec{z}_k = h(\vec{x}_k)$$

In order to update the covariance matrix, the EKF linearize the prediction function f around the current estimate by taking its Jacobian F. The covariance prediction step from figure 15 is then converted to:

$$P_k = F_{k-1} P_{k-1} F_{k-1}^T + Q$$

When calculating the new covariance matrix, the function becomes:

$$P_k = (1 - G_k H_k) P_k$$

Similarly, to compute the Kalman gain, the EKF takes the Jacobian H of the sensor transformation function h. The Kalman gain calculation hence becomes:

$$G_k = P_k H_k^T (H_k P_k H_k^T + R)^{-1}$$

7.4. Beluga 2D EKF

Recall from Section 7.1 that the Kalman Filter can be broken down into two major steps: *Prediction* and *Correction*. In total, the team implemented 2 different kinds of 2D EKF: one that performs *Prediction* with a kinematic model and the other that performs *Prediction* with a dynamic model. They also have a slightly different *Correction* step depending on the information available. This section will first go over the available sensor and input information, followed by a high level description of each model, and lastly a run through of the program flow. For detailed derivation of each formulation, please consult Appendix 14.

7.4.1. Sensors and Inputs

Before going into details about the dynamic and kinematic model, it is necessary to consider and define the different sensor measurements and inputs of the Beluga that can be used in the EKF. As noted before, the Beluga has access to two different sources of sensor

information underwater: the IMU and the hydrophones. It also has information about its motor input at the current sample time.

In a 2D plane, the IMU provides the following 4 measurements denoted as $\vec{z}_{IMU}$

$$\vec{z}_{IMU} = [\ {}^{R}a_x\ ,\ {}^{R}a_y\ ,\theta_{mag}\ ,\omega_{gyro}\ ]^T$$

In this case, $\theta_{mag} = \theta$ and $\omega_{gyro} = \omega_z$. The state space estimation at time step k can be converted to an expected IMU measurement through the function $h_{IMU}(\vec{x})$ with Jacobian H_{IMU}

$$\vec{z}_{exp,IMU,k} = h_{IMU}(\vec{x}_k)$$

On the other side, the hydrophone provides 2 measurements, denoted as a vector $\vec{z}_{hp}$

$$\vec{z}_{hp} = [r_f\ ,r_b]^T$$

Alternatively, the hydrophone measurements can be encoded as

$$\vec{z}_{hp} = [ToA\ ,TDoA]^T$$

The state space estimation at time step k can be converted to an expected hydrophone measurement through the function $h_{hp}(\vec{x}_k)$ and its Jacobian H_{hp}

$$\vec{z}_{exp,hp,k} = h_{hp}(\vec{x}_k)$$

At any given moment, the Beluga has access to the command that it is sending to the motor. The EKF can leverage this information along with a dynamic motion model to predict movement. Since the Beluga has two motors (in 2D), the motor inputs can be defined as u_L and u_R .

$$\vec{u} = [u_L\ ,u_R]$$

Depending on the formulation of the EKF, those sensor measurements are going to be leveraged differently.

7.4.2. State Space Vector

Compared to the generalized 3D state vector shown in Section 7.1, since the localization is simplified to 2D, the robot's state is described by the following state vector (see Table 2 for definitions of variables).

$$\vec{x} = [x\ , y\ , \theta\ , v_x\ , v_y\ , \omega_z\ , {}^{G}a_x\ , {}^{G}a_y\ ]^T$$

To avoid confusion with the x position of the Beluga, the state space vector x from the previous sections is denoted with a vector sign $\vec{x}$. Depending on the formulation (dynamic v.s. kinematic), the state space will either include or exclude ${}^{G}a_x$ and ${}^{G}a_y$.

7.4.3. Prediction with Kinematic Model

The kinematic model does not take into account the drag of water. It uses a set of basic kinematic equations to predict the robot's motion (see Appendix B.1). The state vector for the 2D kinematic formulation consists of six states (as defined in Table 2):

$$\vec{x} = [x\ \ y\ \ \theta\ \ v_x\ \ v_y\ \ \omega_z\ ]^T$$

This formulation does not assume any motion model of the Beluga and therefore, uses acceleration as input instead of the motor input. It is important to note that the IMU accelerometer data is relative to the robot frame (denoted with the pre-superscript R). Since the state variables are in the Global frame, the accelerometer data must be transformed using a rotation matrix. The input for the prediction step is thus the acceleration of the Beluga in its own frame (see section 7.4.1):

$$\vec{u} = [{}^{R}a_x\ \ {}^{R}a_y]$$

For the sensor, this formulation utilizes both the IMU and hydrophone measurement. Since the accelerometer is already used as inputs for predicting, the sensor measurements for the IMU is reduced to magnetometer and gyroscope. IMU magnetometer and gyroscope that update at 50 Hz, and the newly designed hydrophone data updating at 1 Hz. As defined in section 7.4.1, the IMU measurements $\vec{z}_I$ and hydrophone measurements $\vec{z}_{hp}$ are defined as:

$$\vec{z}_I = [\theta_{mag} \quad \omega_{gyro}]^T \qquad \text{(50 Hz update rate)}$$

$$\vec{z}_{hp} = [r_f \quad r_d]^T \qquad \text{(1 Hz update rate)}$$

7.4.4. Prediction with Dynamic Model

The dynamic model has the advantage of the information about the dynamic of the Beluga with a set of more accurate prediction function. In comparison with the kinematic model, the notable differences between the Kinematic EKF and the Dynamic EKF are:

- Prediction step uses a dynamic model rather than a kinematic model
- The control inputs are motor commands
- The accelerometer data is moved into the update step
- X and Y acceleration are included as state variables
- The hydrophone data is formulated as Time of Flight (ToF) and Time Difference of Flight (TDoF)

The full formulation of the Dynamic EKF can be found in Appendix B.2. In brief, the state vector for this formulation of the EKF now contains eight states:

$$\vec{x} = [x \quad y \quad \theta \quad v_x \quad v_y \quad \omega \quad {}^G a_x \quad {}^G a_y]^T$$

Note that all state variables are in the Global frame (not the robot frame). The prediction state uses the motor commands as control inputs (rather than accelerometer measurements).

$$\vec{u} = [u_L \ u_R]$$

Where u_L and u_R are the motor commands (in units of thrust, converted from PWM using the specifications of the motors).[20] Specifically, the state transition function $\vec{s}_{k+1} = f(\vec{s}_k, \vec{u})$ is derived from the experimental step tests. Note that the team assumed the linear and angular responses are independent for simplicity. The dynamic relationship is illustrated in the flowchart below.

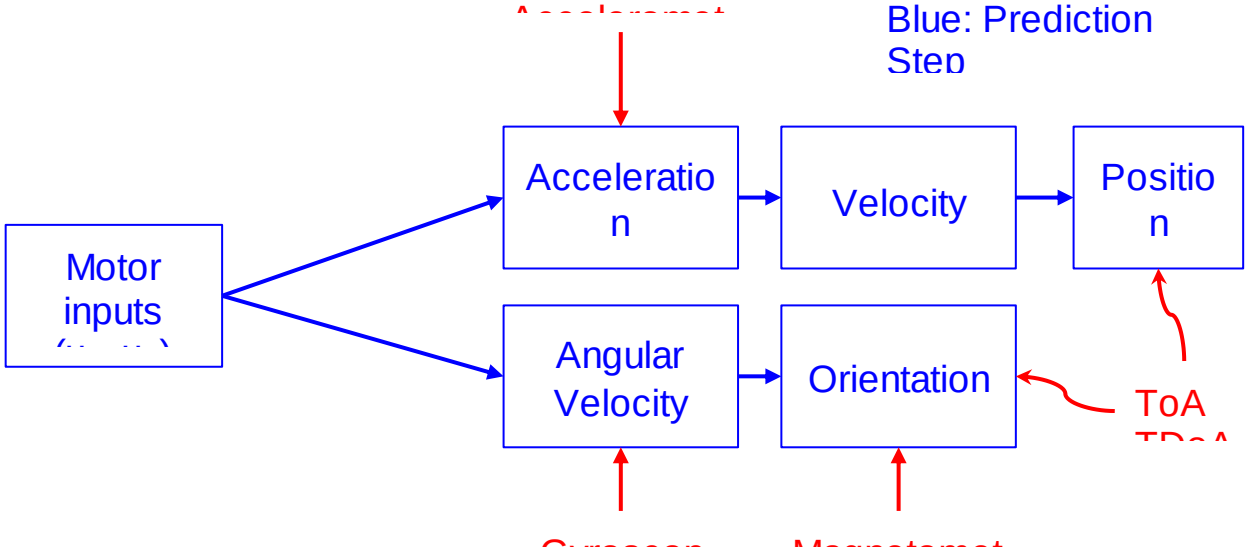


Figure 16: Flowchart of Dynamic Model

Just like the Kinematic 2D EKF, there are two correction steps, one at 50 Hz and one at 1 Hz. Also, the hydrophone data is encoded as Time of Flight (ToF) and Time Difference of Flight (TDoF). The reason for the change in hydrophone data encoding is to accurately reflect the data

[20] "Thrusters & Motors." T200 Thruster Documentation, Blue Robotics, docs.bluerobotics.com/thrusters/t200/.

pipeline characteristics, specifically that the hydrophone pipeline has very accurate measurements of the TDoF, but sees larger error in the ToA. As in section 7.4.1, the IMU measurements $\vec{z}_{IMU}$ and hydrophone measurements $\vec{z}_{hp}$ are defined as:

$$\vec{z}_I = [{}^R a_x \;\; {}^R a_y \;\; \theta_{mag} \;\; \omega_{gyro}\;]^T \qquad \text{(50 Hz update rate)}$$

$$\vec{z}_{hp} = [TOA \;\; TDoA]^T \qquad \text{(1 Hz update rate)}$$

7.4.5. Correction

Regardless of the model, both models need some way to leverage the different sensor measurements. Consider a general sensor measurement $\vec{z}$ that is related to the states $\vec{x}$ by the following equation:

$$\vec{z}_k = h(\vec{x}_k)$$

In the correction step for the EKF, the Jacobian of the function, *H*, is used to replace the linear sensor matrix C (see Ffigure 15) because the function is nonlinear . Hence the correction steps become:

$$G_k = P_k H_k^T (H_k P_k H_k^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k + G_k(\vec{z}_k - h(\hat{x}_k))$$

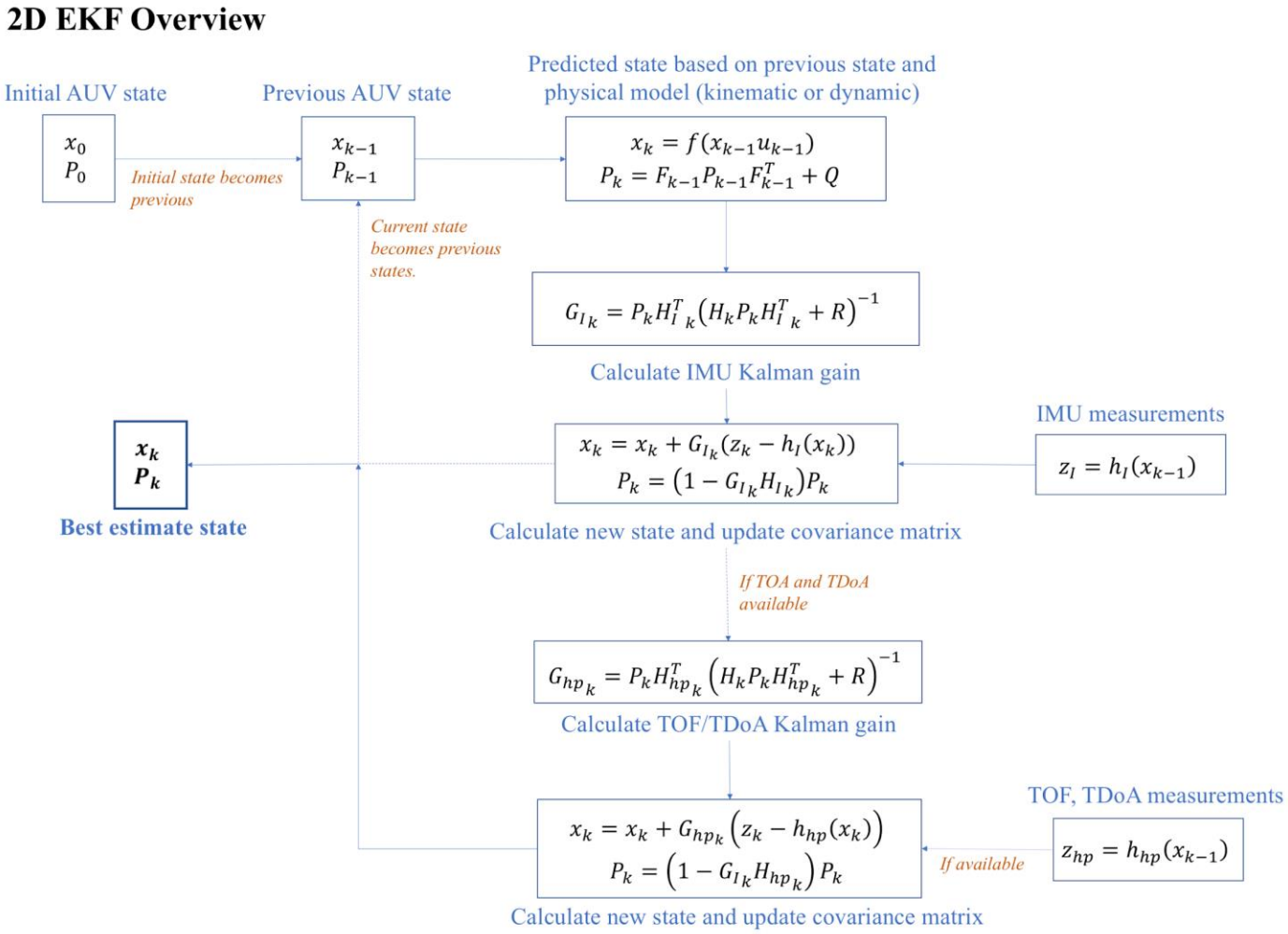
$$P_k = (I - G_k H_k) P_k$$

The equation above applies to both the hydrophone and IMU measurements.

7.4.6. EKF Flow Chart

Upon completing all aspects of the Kalman F, the next step is to integrate both the hydrophone sensor correction and the IMU sensor correction. Figure 17 illustrates the high level EKF process. To begin the Beluga always starts at a known location (right next to the acoustic beacon). As the Beluga begins running, it will update its current estimate using prediction

(kinematic or dynamic). As mentioned before, the IMU is updating at a significantly faster 50Hz than the 1hz hydrophone update. To work around this, the AUV only enters the hydrophone



update step at a 1 hz as illustrated below in the flow chart.

Figure 17: Extended Kalman Filter Overview

8. Pipeline Results

8.1. Field Test Setup

To test the hydrophone pipeline, the team performed a variety of field tests and lab tests. The lab tests involved connecting a function generator, usually producing a sine wave or square wave of a known frequency and amplitude, to the Teensy 3.6 and observing the output on the Intel NUC. Air tests were also conducted in order to check if peak detection was working by taping a hydrophone to the underwater speaker to sense the speaker's vibrations. Field tests were performed either in the large test tank at Harvey Mudd College or in the pool at Pitzer College. Tests at both locations involved lowering the speaker to the same depth as the hydrophone(s) in the water and observing the output of the hydrophones on an oscilloscope or offline after using the pipeline for data collection.

In order to develop a dynamic model for the EKF, the team performed several linear and angular step response tests in the Pitzer College pool with the Beluga. For the linear step response, the team stretched a rope across the length of the pool with a mark at approximately every half meter (the distance between each mark was measured and documented precisely for calculation purposes, but was not the same between each pair of marks). This setup can be seen in the figure below. While logging data from all ROS nodes on the Beluga (particularly the IMU and GPS nodes) and filming the robot, the team set the left and right motor speeds to 35 RPM and left them running for ~20 seconds. By later reviewing the video to see when the Beluga reached each mark, the team could use the video and GPS data to determine the gain of the linear step response of the Beluga. Unfortunately, due to the low frequency of the GPS data (1 Hz) and video data, an accurate time constant for the step response could not be obtained. The IMU data

was demeaned before integration in an attempt to "calibrate" it after it was collected and use it to determine the time constant of the linear step response. However, as can be seen below, it appeared to be too noisy and required more calibration to be accurate and useful. The gain for the linear step response was determined to be 0.92 m/sec per 35 RPM, or $0.03\frac{m/s}{RPM}$ as shown in the figure below.

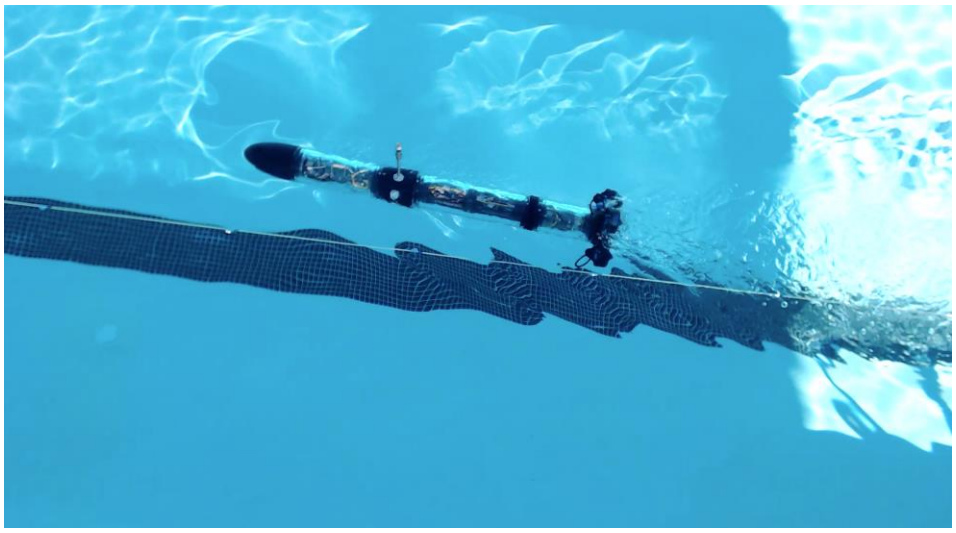


Figure 18: A screenshot of the linear step response test

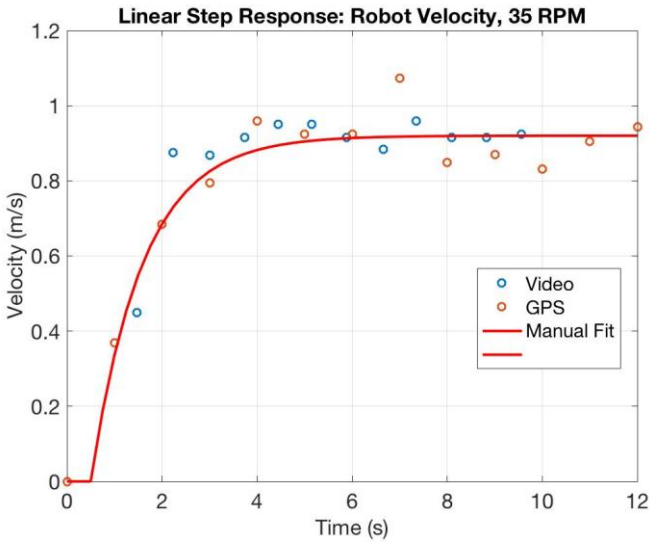


Figure 19: Velocity results from the linear step response test

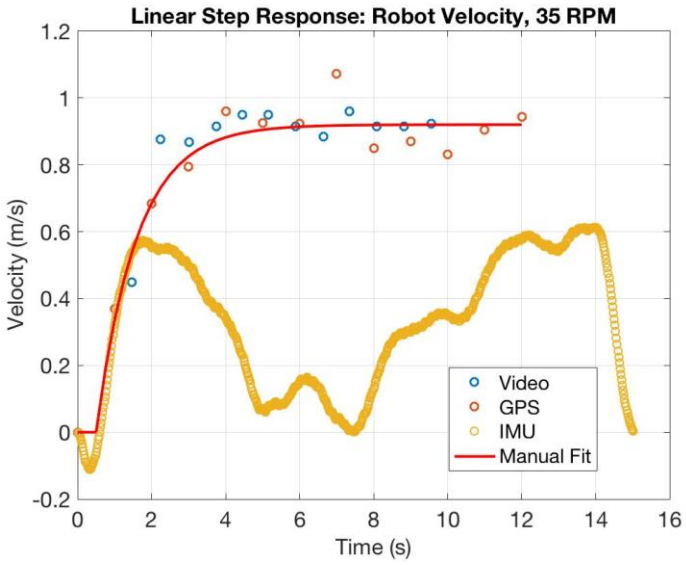


Figure 20: Velocity results from the linear step response test with demeaned IMU data

For the angular step response tests, the team determined through trial and error what the speeds of the left and right motors should be in order for the Beluga to spin about its center of mass (~27 RPM forwards for one motor, 35 RPM backwards for the other, difference of 8 RPM). The motors were set to run at those speeds while the team recorded video of the Beluga spinning and logged data from all its ROS nodes for ~1 revolution. Using the gyroscope IMU data, the

team determined the gain of the angular step response. The gain for the angular step response was determined to be ~0.4 rad/s per 8 RPM, or $0.05\frac{rad/s}{RPM}$ as shown in the figure below.

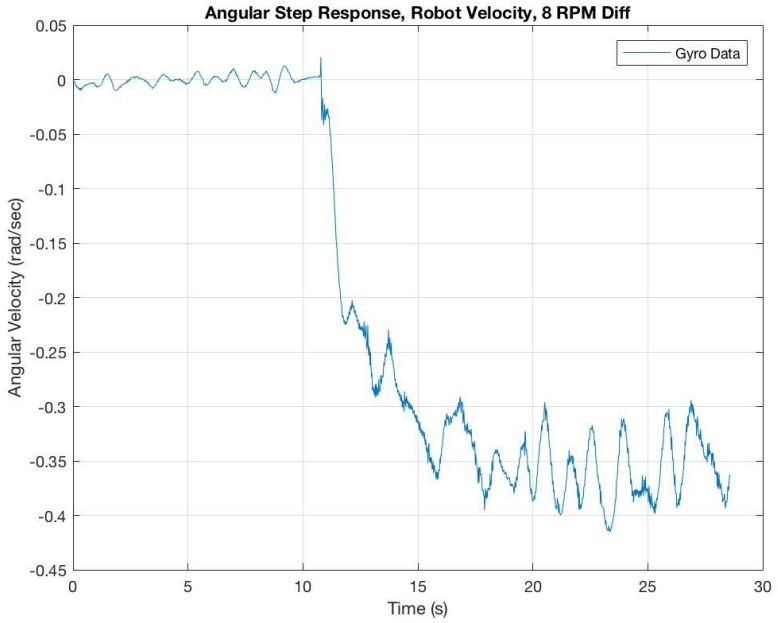


Figure 21: Velocity results from the angular step response test

8.2. Signal Attenuation

As mentioned in Section 6.2, the Teensy analog pins cannot receive a signal outside of the 0-3.3V range. However, the hydrophone raw output signal ranges from 4.5V below ground to 4.5V above ground. So, the team used a signal conditioning circuit to ensure that the maximum amplitude of any signal going into the Teensy from the hydrophones was 3.3V. This maximum occurs when the hydrophone is closest to the acoustic beacon, but as the hydrophone moves away from the beacon, the power of the signal attenuates at a rate of $1/r^2$ where r is the distance away from the beacon. Water transmits audio far more efficiently than air[21], so assuming perfect

[21] Liebermann, L.N. "The Origin of Sound Absorption in Water and in Sea Water"

power transfer through the water, the signal strength of the sound waves would still have to spread out at a rate of $1/r^2$ with the distance from the transmitting source.

Due to this attenuation, the signal becomes small in amplitude the farther the hydrophone gets and thus more noisy due to circuit noise. Experimentally, the hydrophone signal cannot be detected by the matched filter when the hydrophone is greater than 2 m away from the acoustic beacon. If the maximum hydrophone recorded voltage signal $V_0 = \pm 0.25V$ (from the figure below) when the hydrophone is next to the beacon, then at r = 2 m, the signal will drop to $V_0 / 2^2 = 0.06V$, which is indistinguishable from the current noise.

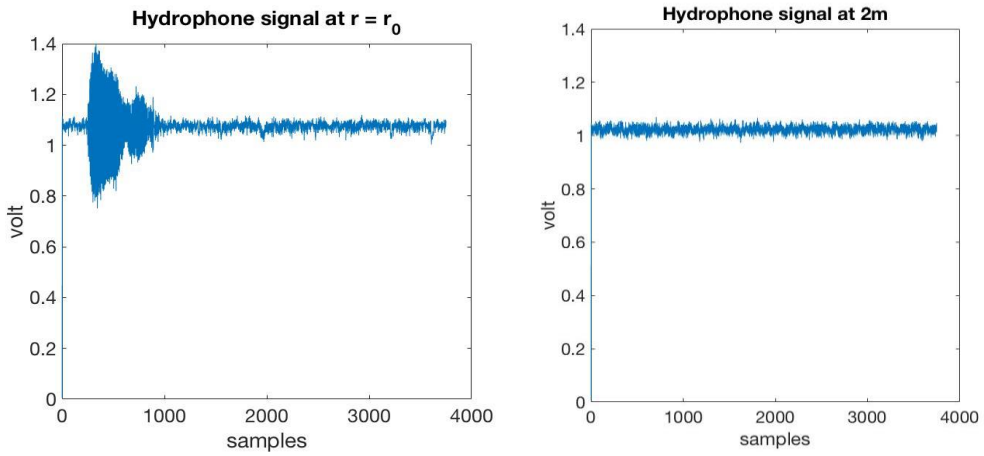


Figure 22: The hydrophone signal as recorded by the Teensy at different distances from the beacon

The signal conditioning circuit also had the wrong values for the resistors, making the signal not centered around 1.65V but at 1.09V, as seen in the plots above.

To fix this signal attenuation issue, a variable gain amplifier could be added to the hydrophone pipeline. This component could vary the gain it provides to an input voltage, such as the hydrophone signal, depending on a given control voltage[22].

8.3. Drift

The hydrophone pipeline system is subject to a linear drift as well as oscillations. When the robot is stationary, the robot perceived a drift of 3 cm/s as well as a 1 m amplitude sawtooth with a period of approximately 12 seconds. The inconsistent nature of the sawtooth makes it difficult to calibrate out. While further investigation is necessary to determine the source of this drift and sawtooth oscillation, the team determined that this error is due to timing irregularities in the pipeline and not due to false detection or multipathing, as naive edge finding, matched filter, and manual edge checking all yielded similar times of flight. The team suspects the audio player to be responsible for much of the drift, since the beacon's chirp was played using an audio file from a personal computer. Testing of different audio sources will likely be necessary to reduce this error.

[22] Tuite, Don. "Variable Gain Amplifiers"

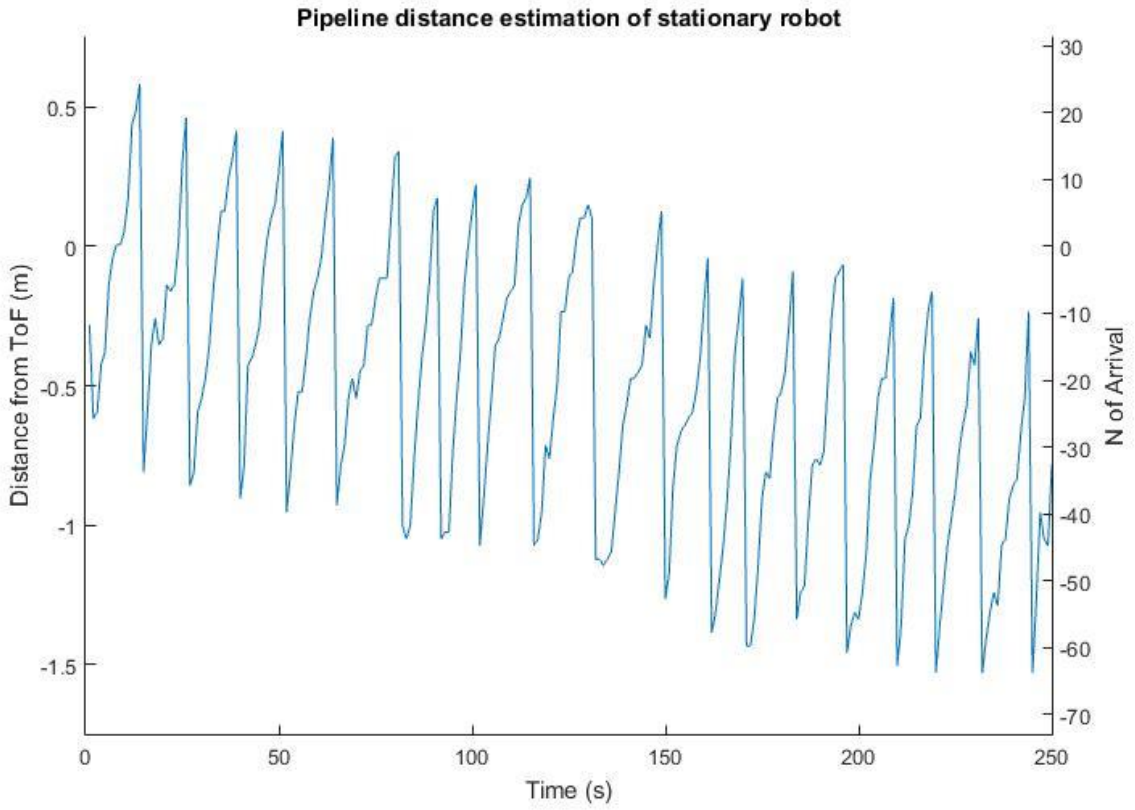


Figure 23: Distance estimates of stationary robot

9. State Estimation Algorithm Results

As mentioned earlier, the team chose to implement an Extended Kalman Filter (EKF) for the localization algorithm. Given the issues with logging viable data from the hydrophone pipeline, to test the effectiveness of the EKF the team simulated data from hypothetical test cases, adding realistic noise to sensor measurements to accurately resemble real life testing. Ultimately, this provided freedom for testing the localization algorithm under many different scenarios and allowed for rapid development.

To describe the statistical properties of the noise, the following symbols are used:

σ_{sensor_x} denotes the "standard deviation of sensor_x"

μ_{sensor_x} denotes the "mean of sensor_x"

These parameters are adjusted to determine the sensitivity of the estimation algorithm to errors.

9.1. Direct Integration

As a reminder, the only localization algorithm available to the team last year was direct integration. For the 2D case, this means converting the accelerometer data into the global from (from the robot/local frame) using angular measurements, then integrating to get velocity and position in the global frame. This method of sensor fusion is not robust to sensor noise or error. Any error in the acceleration will accumulate in the position estimates, leading to unrealistic drifts in the position estimate. This is becomes a problem especially when the accelerometer data has a non-zero mean error (ie. not perfectly calibrated).

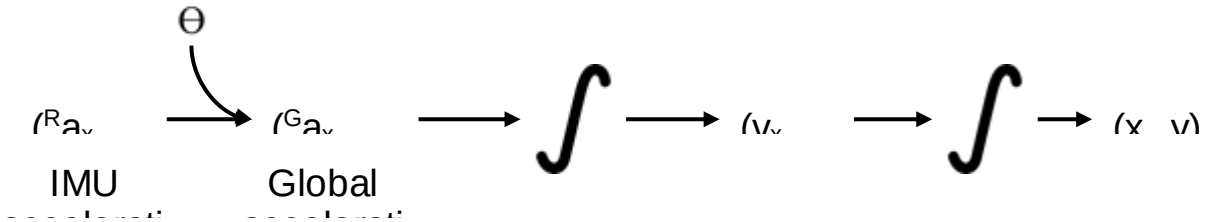


Figure 24: Flowchart of Direct integration sensor fusion method.
Pre-superscripts denote the global (G) or robot (R) frame

9.2. Data Generation (Using Dynamic Model)

In order to simulate data for a hypothetical trajectory, a SimuLink model was created based on the linear and angular step response of the Beluga. Thus, for a given left and right motor input (in units of thrust), the model will output all relevant states (acceleration, velocity, position, angular velocity, orientation).

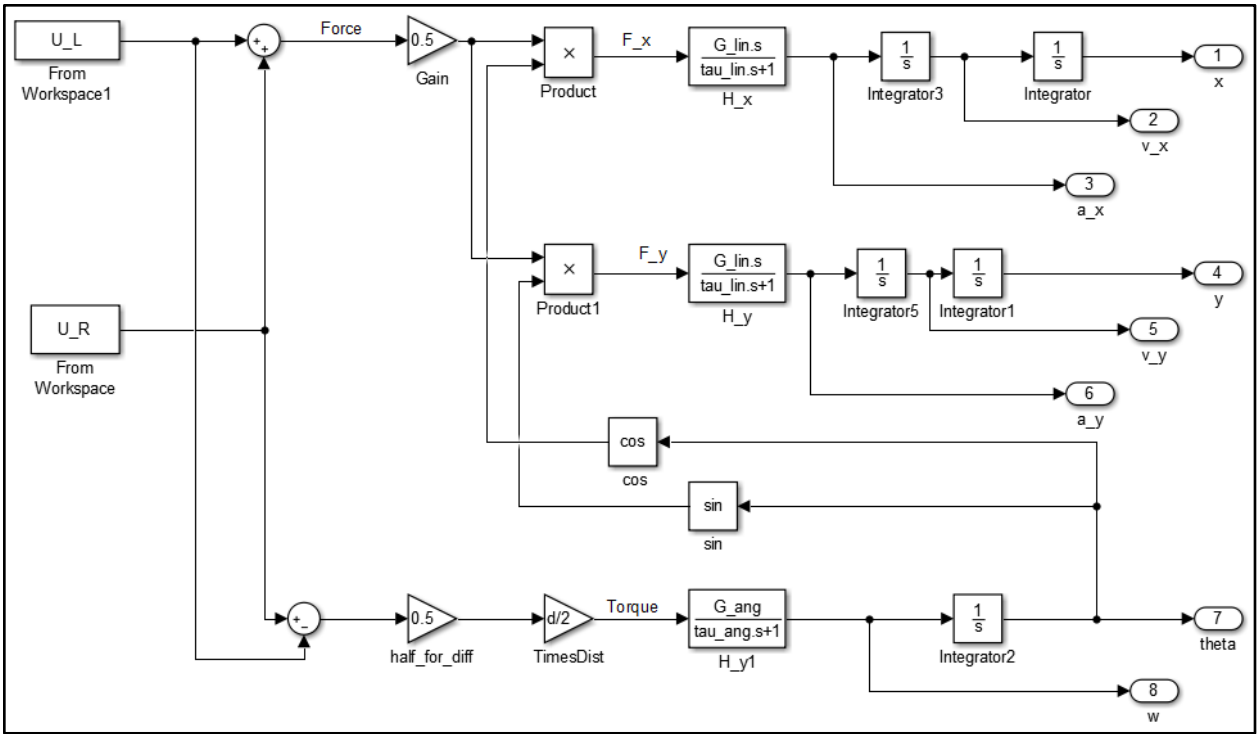


Figure 25: SimuLink dynamic model of Beluga. Motor commands as inputs, and state variables as outputs

Thus for a given motor input (figure 26), a trajectory is defined (figure 27) along with the corresponding data. From the ground truth data, additive white gaussian noise is used to produce sensor measurements (figure 28).

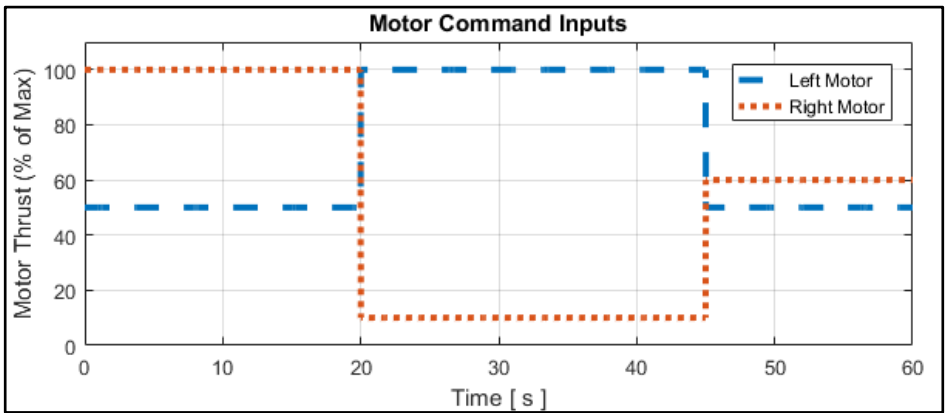


Figure 26: Sample motor command inputs

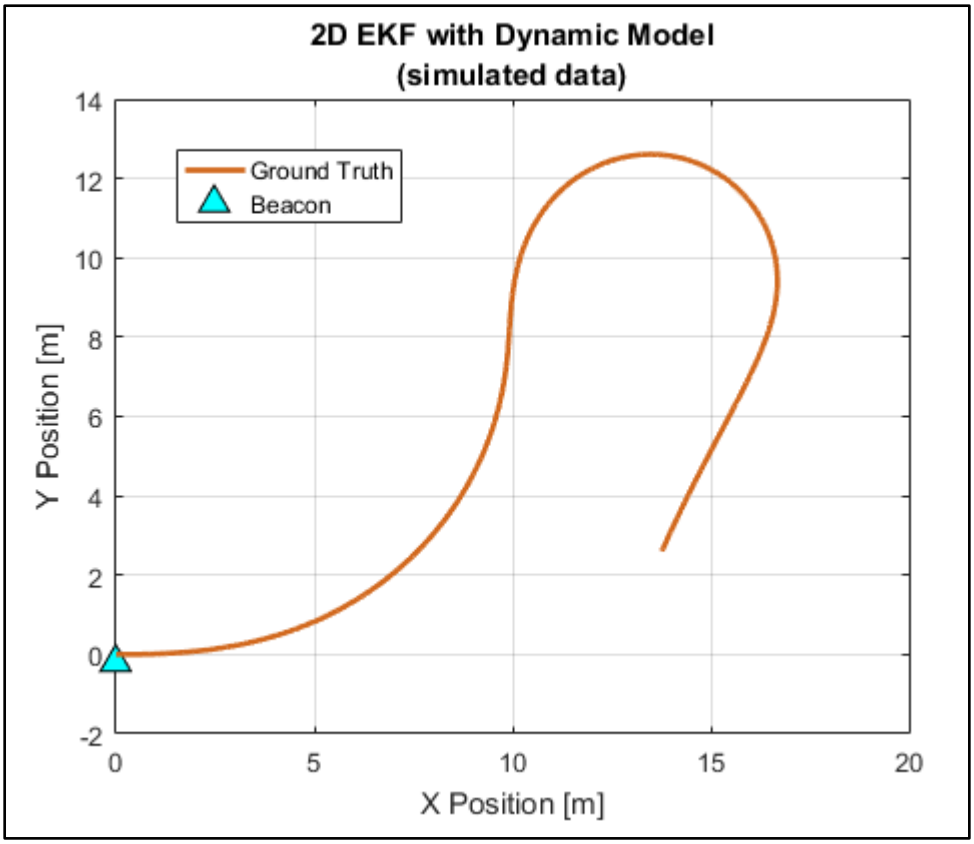


Figure 27: Sample trajectory of Beluga, given motor commands

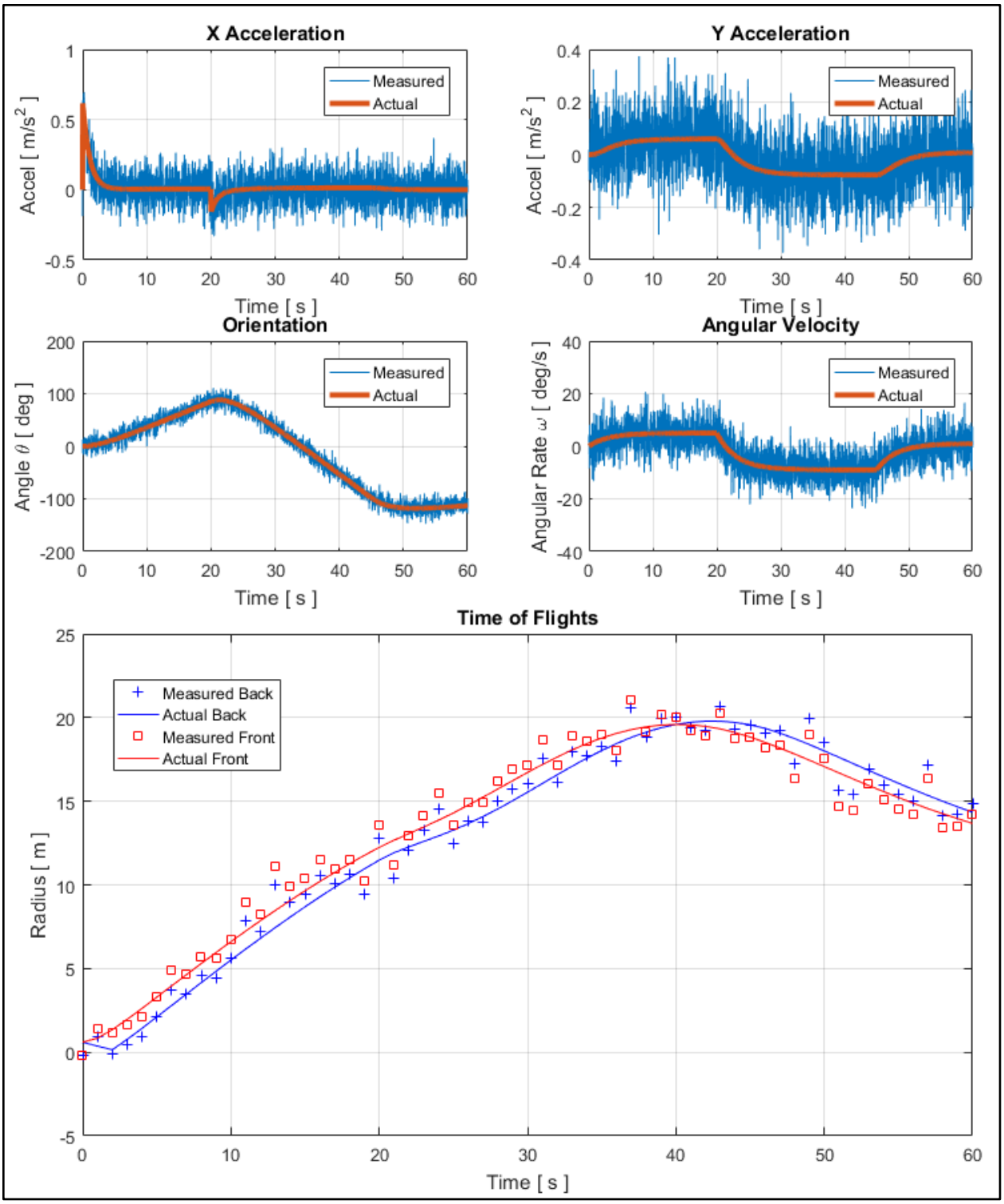


Figure 28: Measured and Actual data, using $\sigma_{ax} = \sigma_{ay} = 0.1 m/s^2$, $\sigma_{\theta} = 10\ deg$, $\sigma_{\omega} = 5\ deg/s$ $\sigma_{ToF} = 1\ m$, and $\sigma_{TDoF} = 0.01\ m$ for sample trajectory

As can be seen in the figure above, the accelerometer error is much larger in magnitude than the error of the other sensors. Compared to this error, the magnetometer and gyroscope errors appear small, which is due to the IMU's internal EKF for its Attitude Heading Reference System (AHRS).

There is also a large difference in the magnitude of ToF error and TDof error, as the $\sigma_{ToF} = 1\ m$and $\sigma_{TDoF} = 0.01\ m$in the above figure. This difference in error accurately reflects the hydrophone pipeline characteristics, because the system should be able to more accurately determine the difference in the times of flight than the times of flight themselves.

9.3. Comparison of Designs

Using ideal sensor measurements as input to the correction step of the EKF, the algorithm appears to estimate the robot's state in 2D fairly accurately. This ideal sensor data assumes a perfectly calibrated IMU, no sensor biases, and good ToA measurements. Below is a plot of the same trajectory shown before with the same errors to the sensor measurements applied.

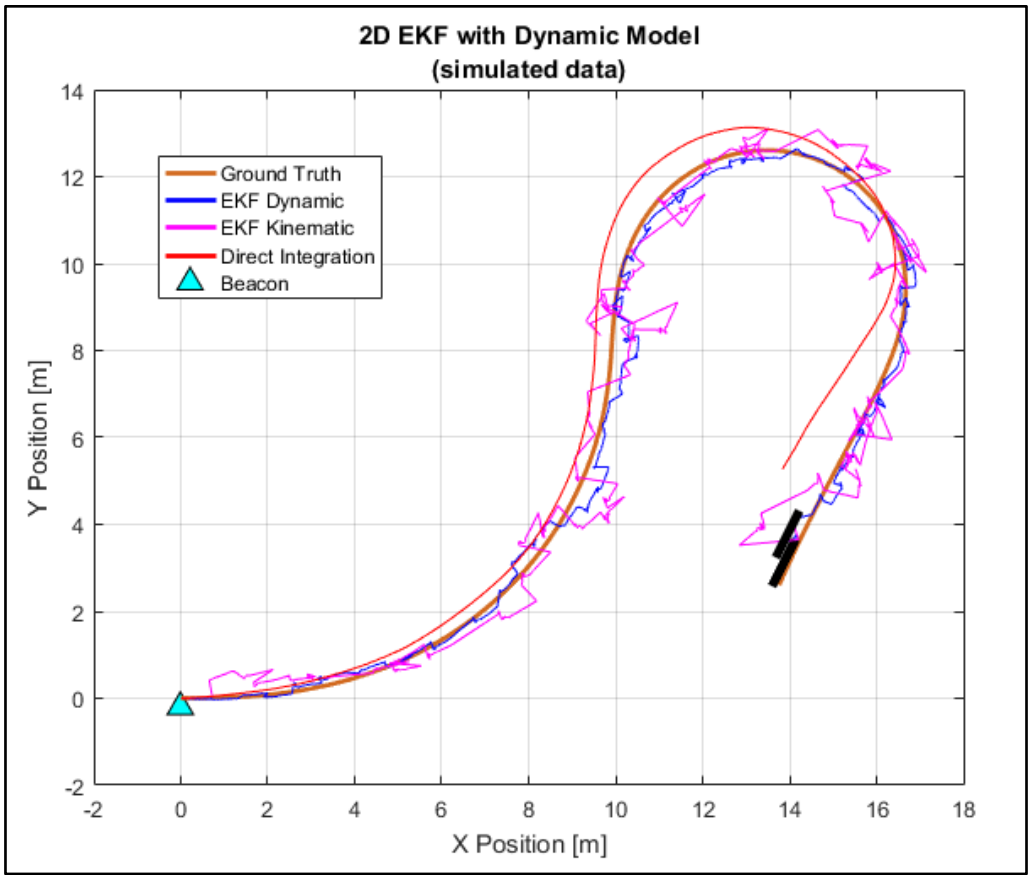


Figure 29: All three localization algorithms with simulated data

9.3.1. Issues with Direct Integration

While all three algorithms seem to perform relatively well under ideal cases, the method of direct integration is extremely sensitive to non-zero mean error in the accelerometer. The below plots illustrate this fact, using a non-zero mean acceleration error of $\mu_{ax} = 0.1 m/s^2$. This value was chosen to reflect the drift seen in actual testing of the accelerometer. According to Vector Nav, the bias can be calibrated to less than $\mu_{ax} = 0.01 m/s^2$ using a simple axial calibration (checking the min/max along each axis) but the team could not get the bias down to that level.

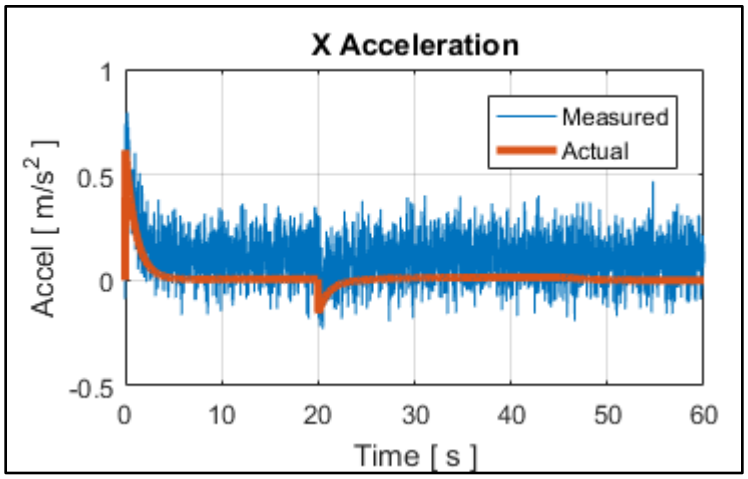


Figure 30: Non-zero mean acceleration error ($\sigma_{ax} = 0.1 m/s^2$)

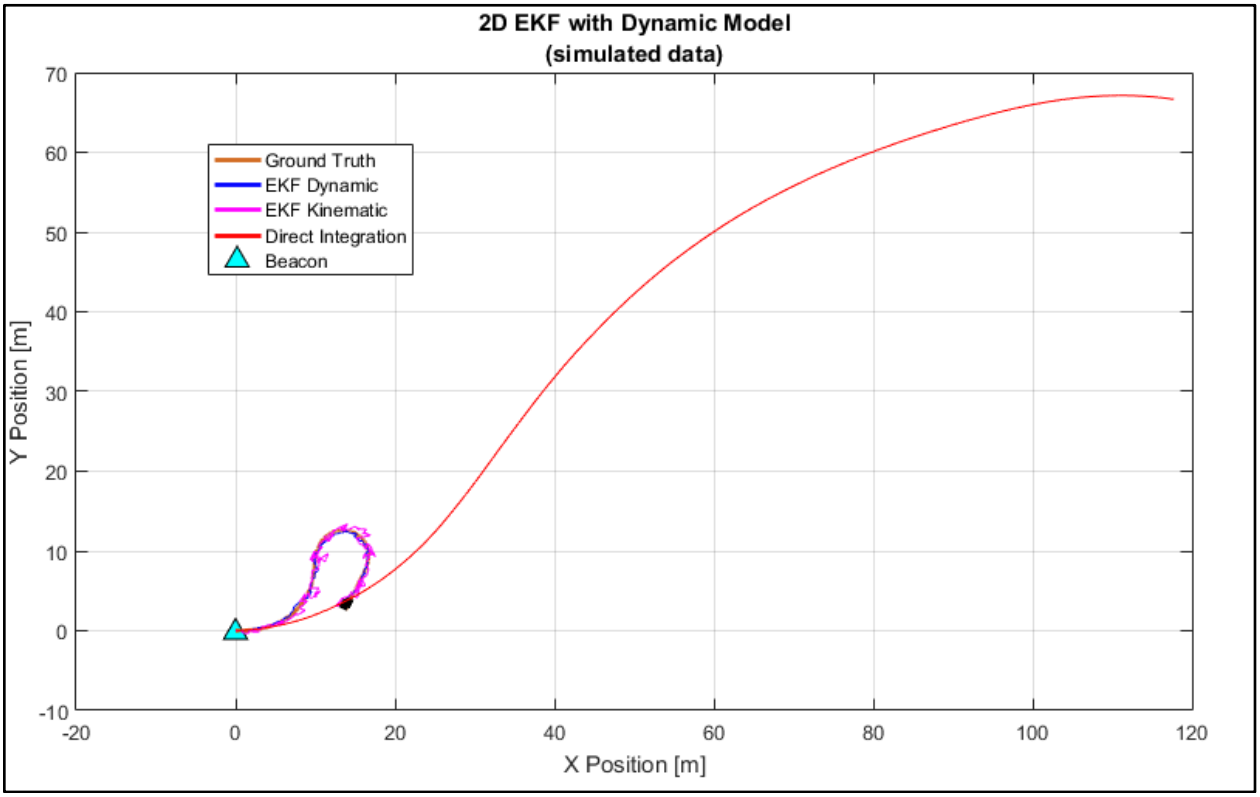


Figure 31: Sensitivity of Direct Integration to non-zero mean error in acceleration data

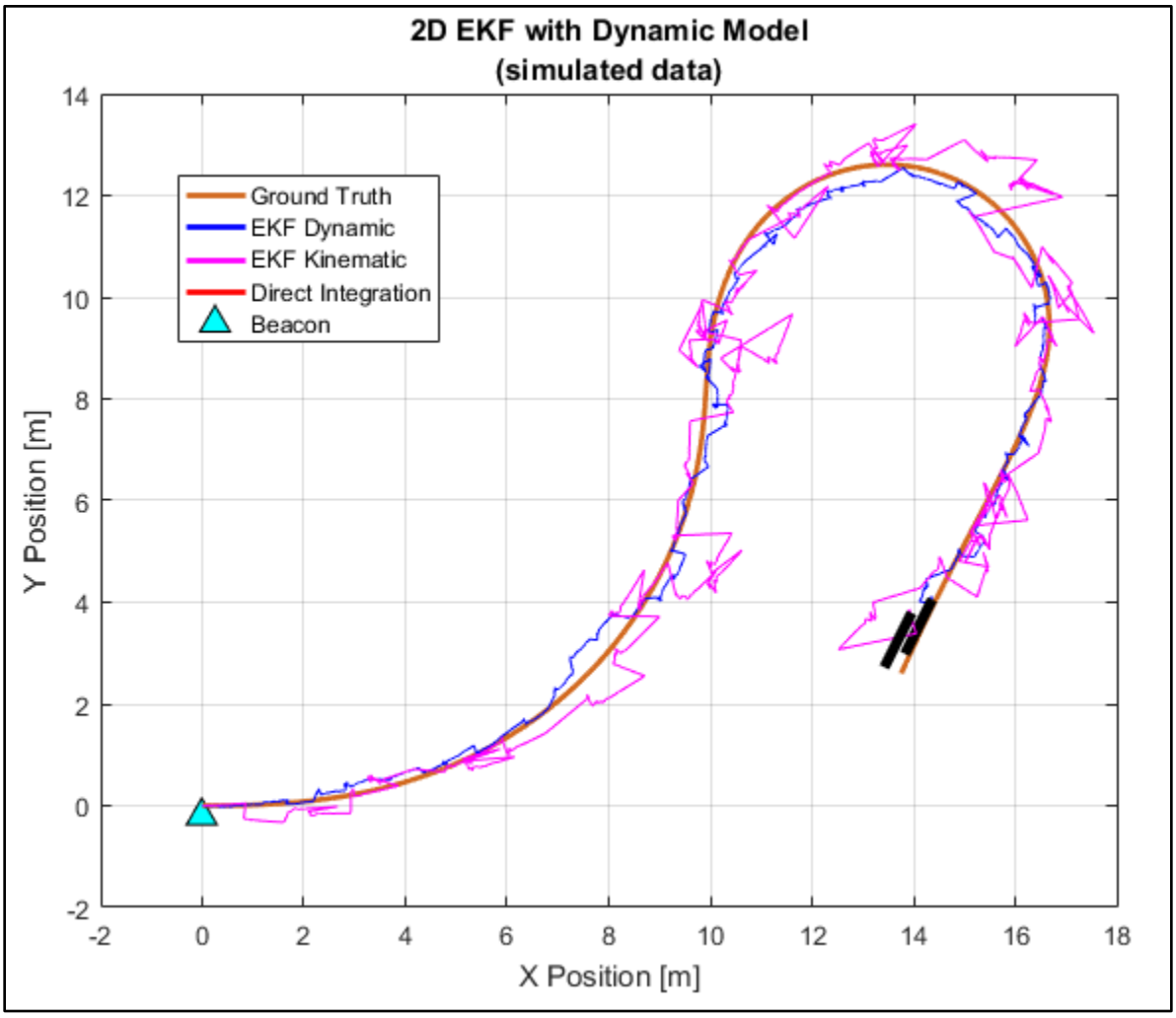


Figure 32: EKF estimates are not nearly as sensitive to non-zero mean errors

As demonstrated in the plot above, direct integration of the IMU accelerometer results in a very poor 2D state estimate. This error is due to the bias in the accelerometer which results in accumulation of error over time and leads to the terrible state estimates described in Section 2. This bias is not an unreasonable expectation for an actual accelerometer, as the team had difficulties calibrating the bias of their IMU's accelerometers out for this project. Thus direct integration is not an viable option for localization. On the other hand, both EKF estimates are accurate, and resemble the ground truth (albeit with some extra noise).

9.3.2. Issues with Kinematic Model

The system model that the kinematic EKF relies on is not as realistic as the dynamic model because it does not rely on the true dynamics of the robot moving through water. Thus, the prediction step of the kinematic EKF is not as accurate as the dynamic EKF's prediction step. So, when the sensors also provide biased, noisy data (particularly ToF data), the state estimation is erratic and more inaccurate than the state estimation of the dynamic EKF.

By changing the standard deviation of the ToF data to be 10 m (compared to the default 1 m), the Kinematic EKF struggles in converging to the location. The plot below illustrates how bad the ToF data is.

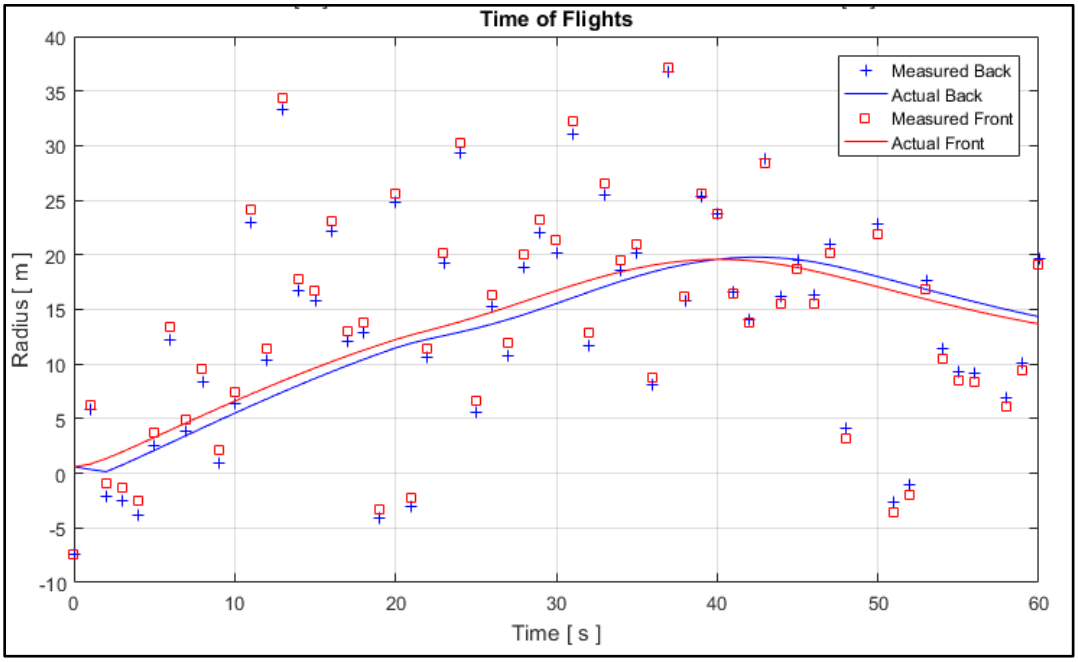


Figure 33: Plot of measured and expected ToF data with an error with 10m standard deviation

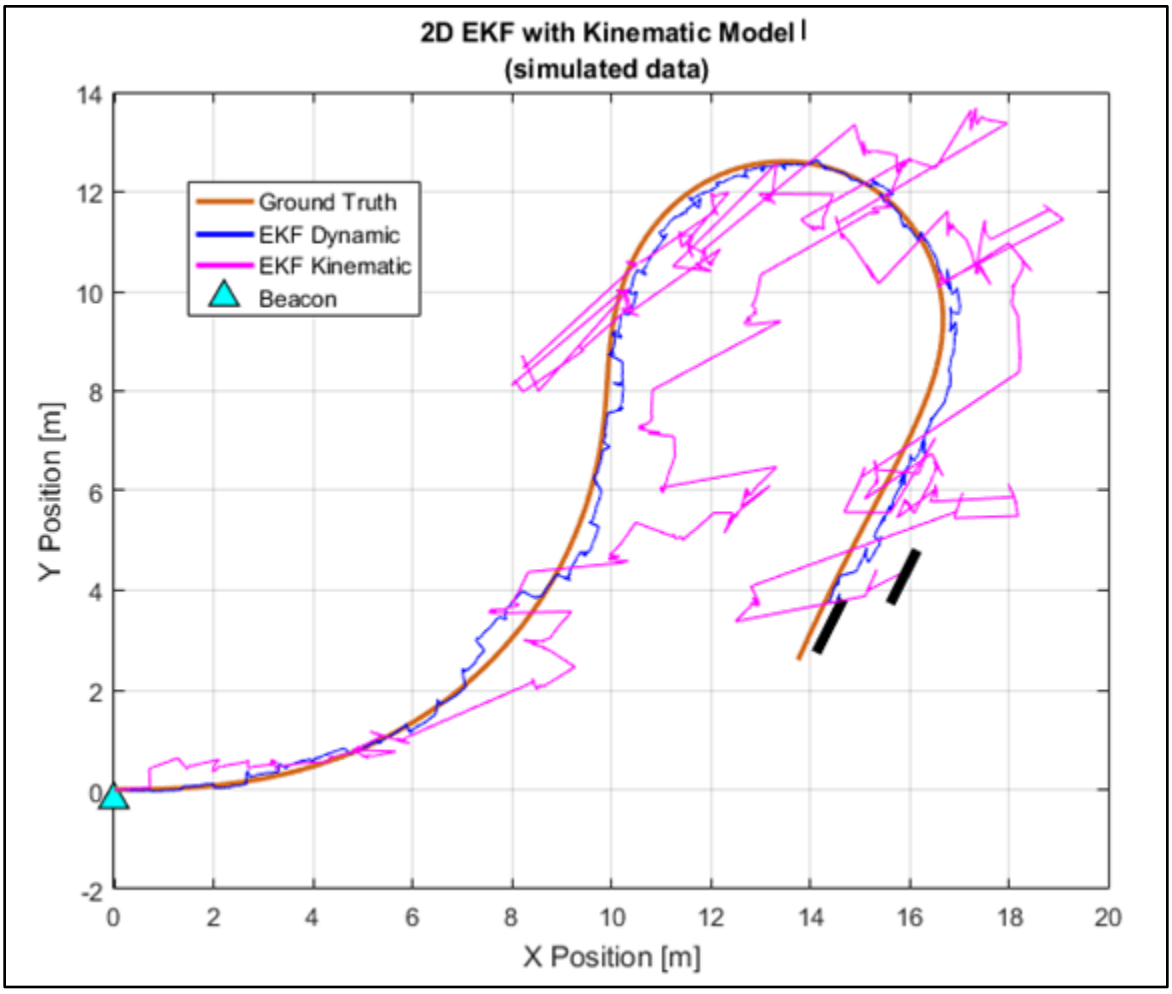


Figure 34: Performance of both EKFs with poor ToF measurements

We can see in this plot that the Kinematic EKF does not track the trajectory truthfully. There are many large jumps between location estimates, thus it would be exceedingly difficult to design a control system around this localization algorithm. On the other hand, the Dynamic EKF still tracks the ground truth very well.

To test both algorithms further, both a poor ToF measurement ($\sigma_{ToF} = 10\ m$) and non-zero mean accelerometer error ($\mu_{ax} = 0.1\ m/s^2$) are applied.

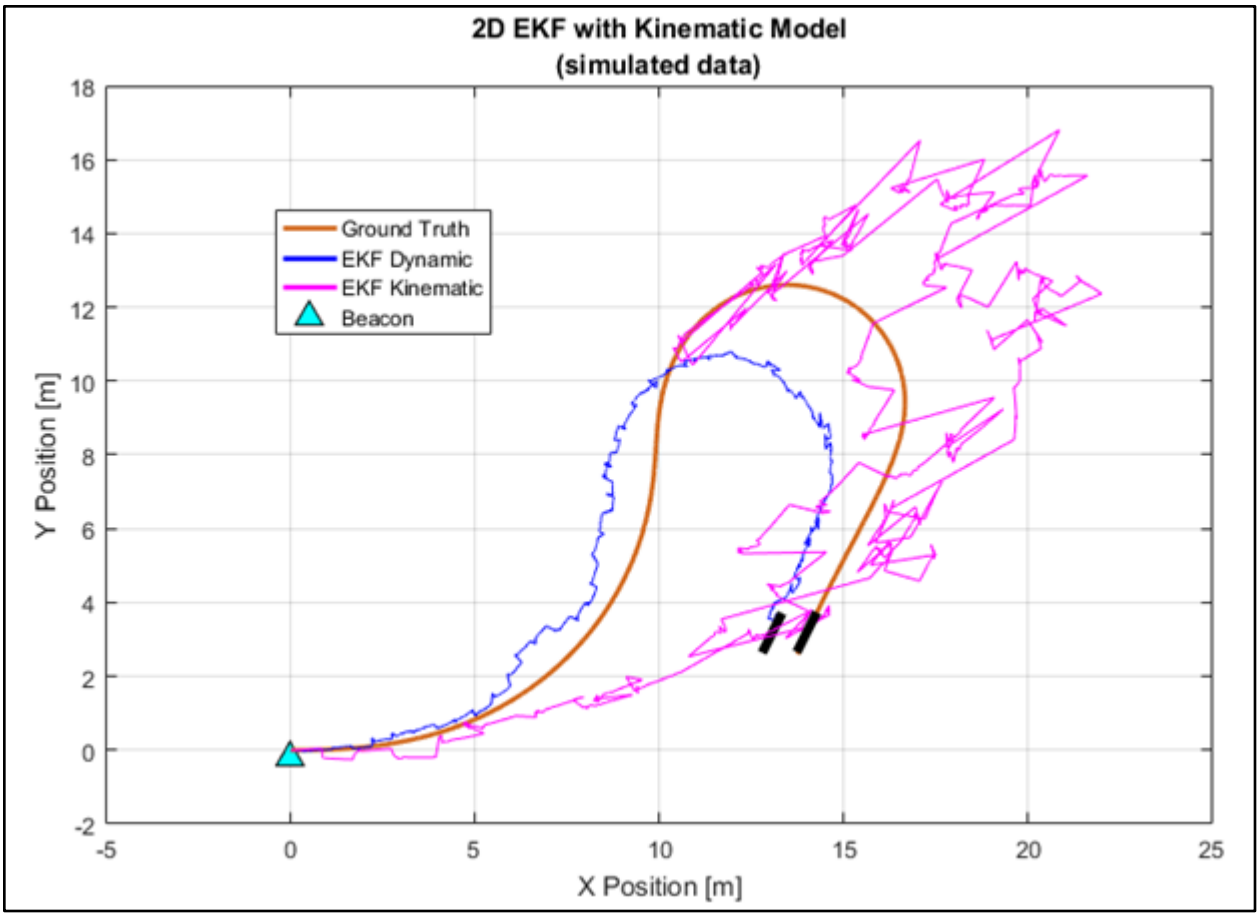


Figure 35: Performance of both EKFs with poor ToF measurements and accelerometer offset

Both the dynamic and kinematic EKF estimates do not closely follow the ground truth trajectory, but the dynamic EKF's estimate still resembles the shape of the ground truth while the kinematic EKF is far more erratic.

As illustrated by revealing these issues with the kinematic EKF, only the dynamic EKF can accurately, robustly localize in spite of sensor bias and noise. The next section will quantify how robust this state estimation is.

9.4. Characterization of 2D Dynamic EKF

With a final fully developed version of the Dynamic EKF, the team then focused on characterizing the performance. Since the algorithm could only be tested in simulation, due to the

difficulties encountered with the hardware, characterizing the Dynamic EKF is important so that ultimate integration with the hardware goes smoothly.

9.4.1. Performance with Standard Trajectory

First, consider the standard trajectory used in the past sections and generated in Section 9.2. Also, consider the same measurement characteristics (standard deviations, means, etc).

The performance of the Dynamic EKF is illustrated below. The dashed ellipses represent the covariance with respect to the x-y position. This can be interpreted as a metric for how confident the algorithm is in the estimate. Another way to interpret the covariance ellipses are a slice of the 2D gaussian distribution surrounding the estimate, specifically at the one-sigma value, such that we are 68% confident the true position is within the ellipse. Note that the ellipses are only plotted every 5 seconds.

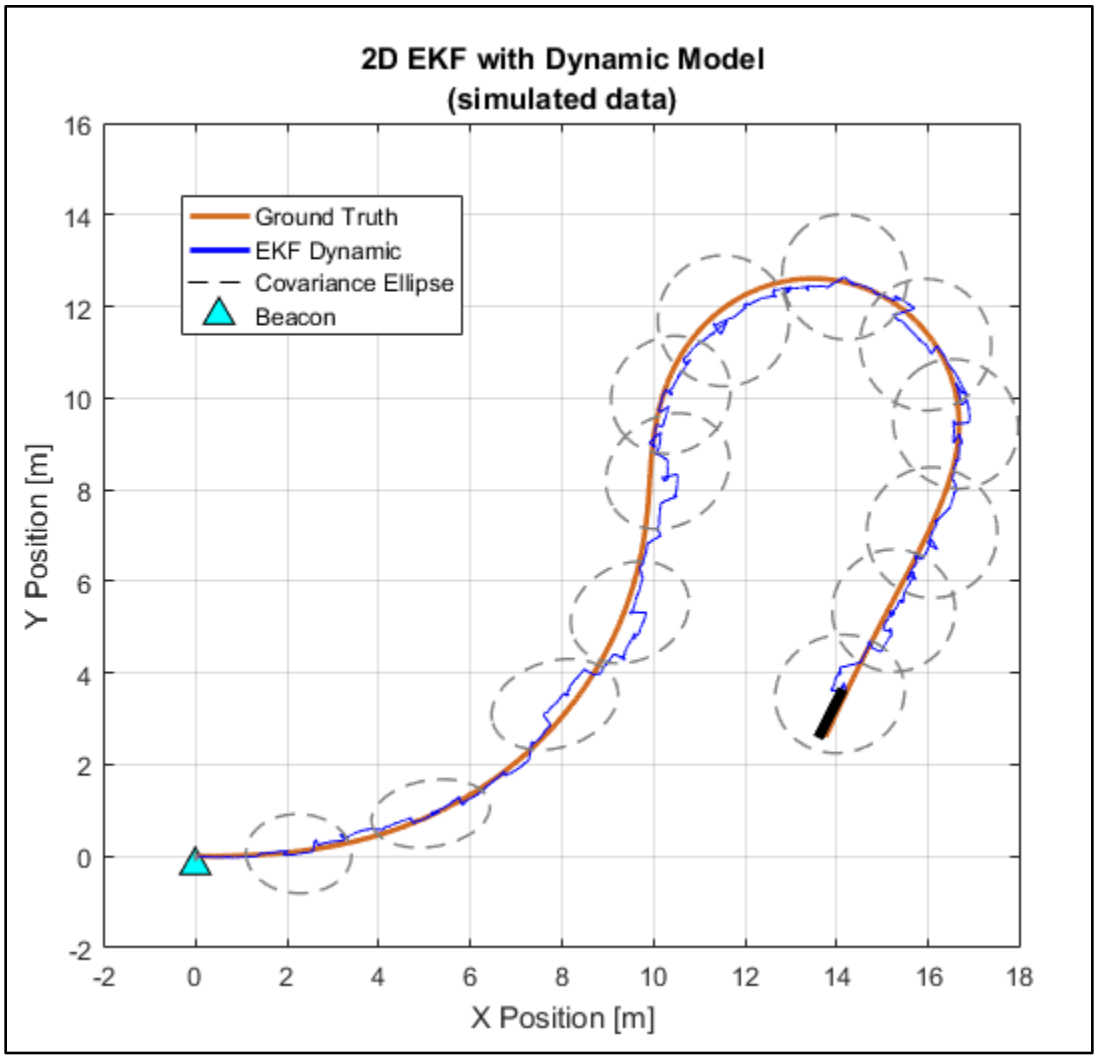


Figure 36: Performance of the Dynamic EKF on the example trajectory, including covariance ellipses

For a clearer illustration, below is a plot of the positional covariances versus time.

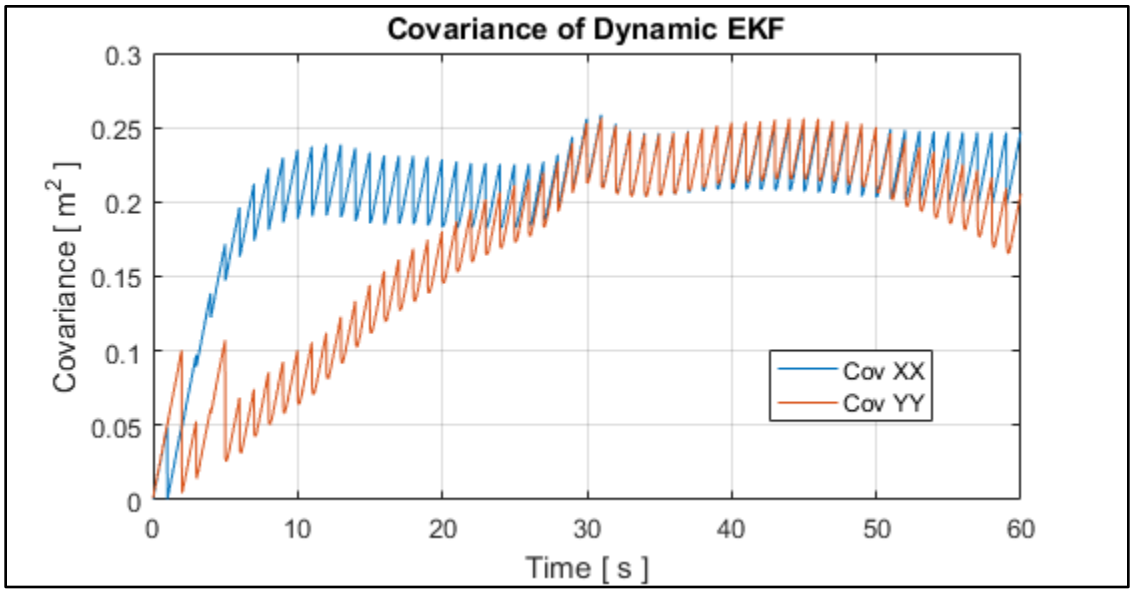


Figure 37: Positional Covariance of the Dynamic EKF versus time

The relative magnitude of the covariances are less than a meter. There are signs of convergence as time progresses.The jagged behavior is due to the prediction-correction steps of the Kalman Filter. Specifically, every 1 second, the covariance shrinks when the algorithm receives a new hydrophone measurement. Between the hydrophone measurements, the states are merely propagated with the dynamic model, and corrected with some IMU measurements. The covariance increases in the propagation phases because there is no direct measurement of the position. The process noise, Q, can be adjusted for slower/faster growth of the covariance in the propagation steps.

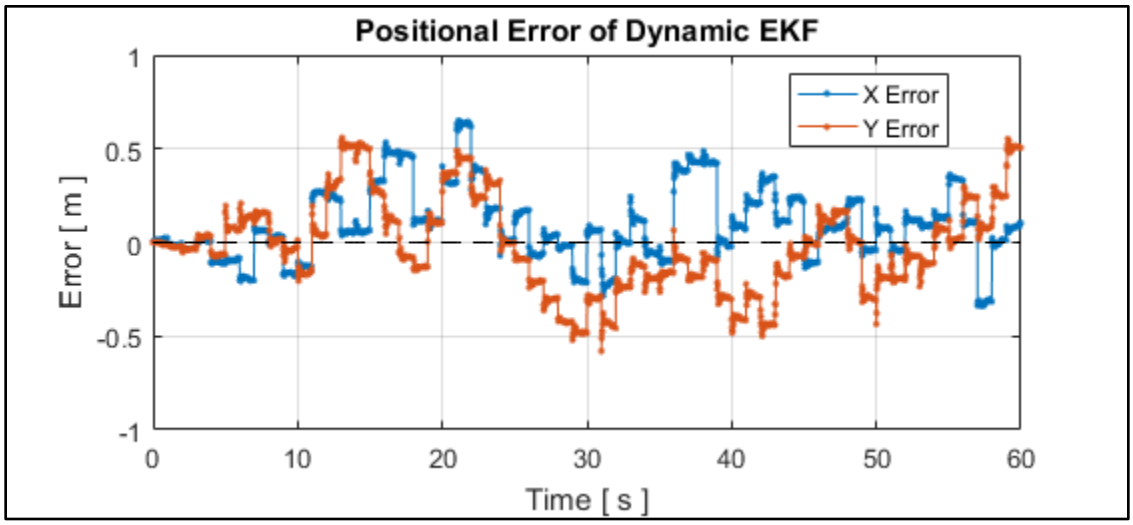


Figure 38: Positional Error of Dynamic EKF on the sample trajectory

Above is a plot of the positional error of the EKF. Note that the error does not grow over time (as would be expected for odometry systems). The error for this setup is less than 1 meter.

9.4.2. Covariance Dependence on Distance

Changing the trajectory illustrates the sensitivity of the covariance to range. In the plot below is a different trajectory than previously used. The same measurement characteristics (standard deviations, etc.) remain unchanged. The trajectory runs for 200 seconds (compared to the 60 second simulation before). The Beluga reaches a distance of ~150m away from the beacon.

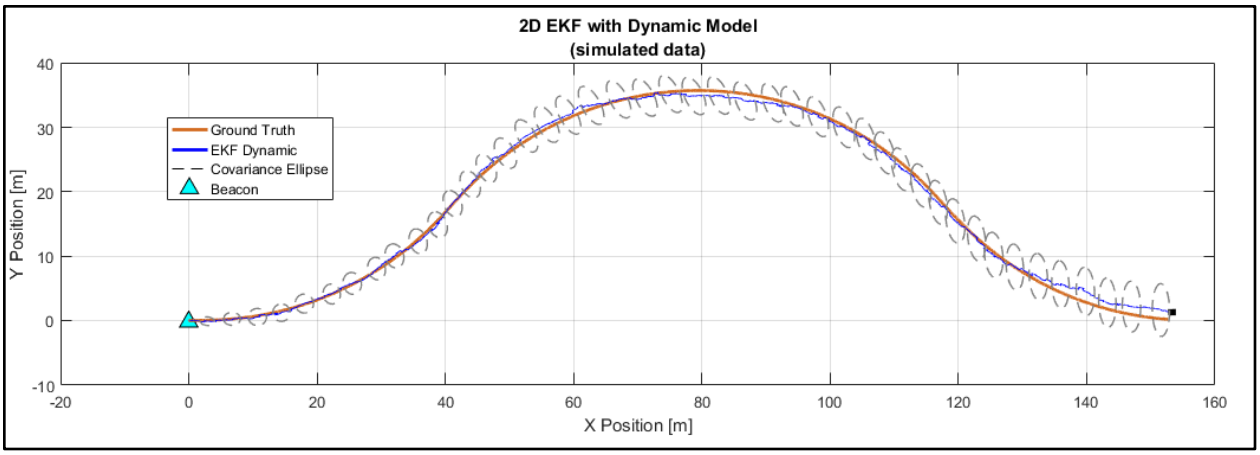


Figure 39: Different trajectory and EKF response

Even for this longer simulation that covers more distance, the EKF is still able to track the motion well. One thing to point out is the range dependence of the covariances.

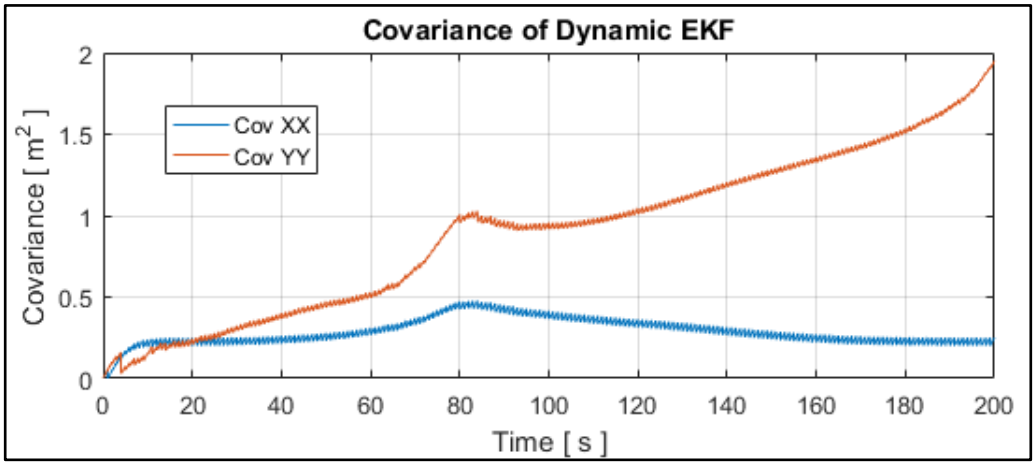


Figure 40: Covariance growth with increases distances from beacon

In the above plot it is clear that the covariance, particularly in the Y direction, grows significantly This is due to the geometric properties of the acoustic localization. At larger ranges, the distance between the hydrophones (mounted on the Beluga) becomes significantly smaller than the distance to beacon. As such, it is more difficult to determine the bearing relative to the beacon. One can contextualize this growth in error as a constant uncertainty in angle relative to the beacon, which correlates to increase positional error as the radius of the range circles (see Figure 7) increase.

9.4.3. Sensitivity to System Disturbances

Continuing with the characterization of the Dynamic EKF, the team investigated the algorithms sensitivity to disturbances on the system. Random disturbance blocks were included in the SimuLink model with specifiable mean and variances (both linear and angular disturbances). Below is a plot depicting the actual force/torque resulting from the disturbances compared to the ideal or commanded motor inputs using the default trajectory generated in Section 9.2.

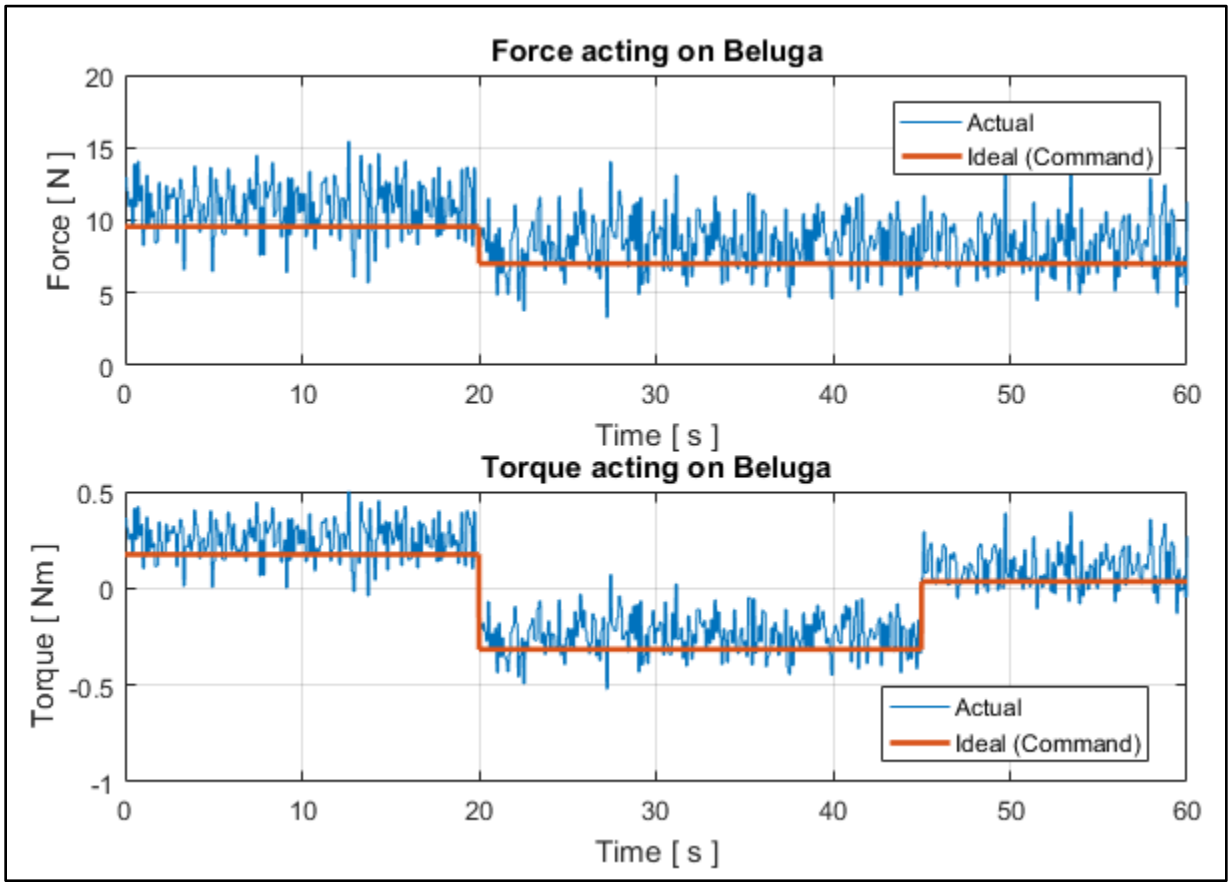


Figure 41: Ideal and Noisy Force/Torque from including system disturbances

The magnitude of the disturbances are specified as a percentage of the max force/torque that the Beluga can produce (based on the motor characteristics). A non-zero mean disturbance

can be contextualized as a water current, in that it produces a net force on the Beluga that is unaccounted for in the system. The above plot disturbances have the following characteristics:

Linear Force Disturbance:
Mean = 10%
Var = 50%
Angular Torque Disturbance
Mean = 10%
Var = 50%

With the default trajectory, and the default measurement characteristics, the following trajectory, and resulting Dynamic EKF performance are plotted below using the noisy force and torques.

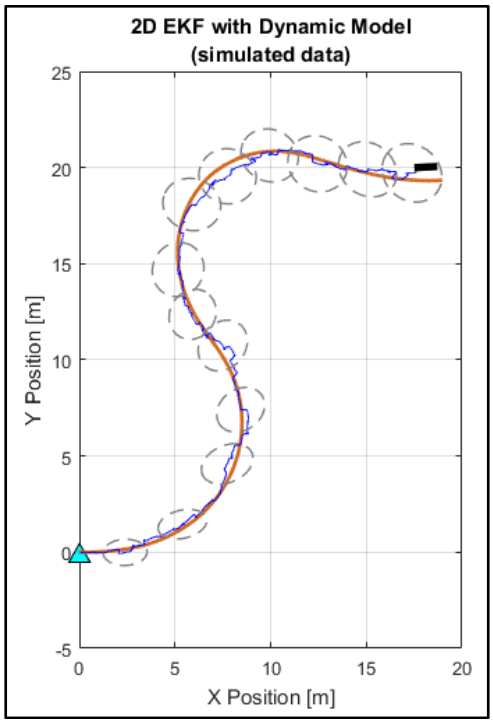


Figure 42: Trajectory and EKF performance from the noisy Force/Torques using system disturbances

Note that the trajectory is different from the desired default trajectory. Despite the change in trajectory, the EKF is still able to track the Beluga well. To reiterate, this example is meant to depict the sensitivity of the algorithm to system disturbances, assuming the sensors are working

properly. This means that the prediction step is not completely accurate, and thus has to rely more on the correction steps.

Taking this to even more extreme values, below are results from incorporating a much larger mean disturbance.

Linear Force Disturbance:
Mean = 100% (rather than 10%)
Var = 50%

Angular disturbances remain the same.

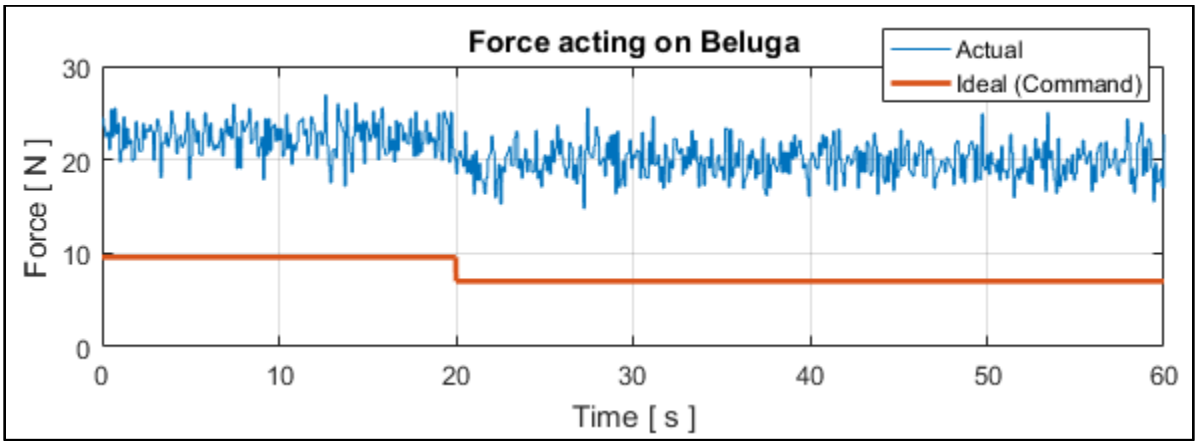


Figure 43: Noisy Force due to large non-zero meaned system disturbance

The resulting trajectory and EKF performance are shown below.

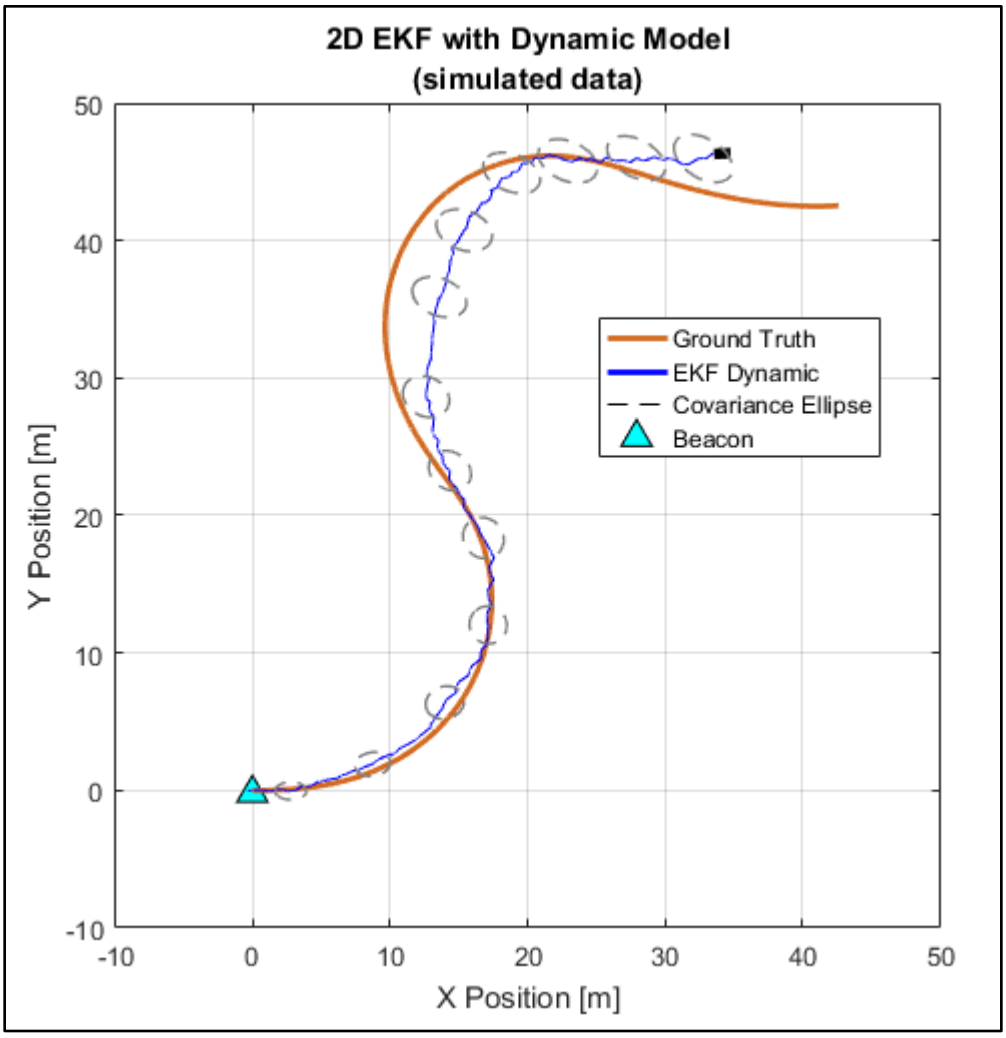


Figure 44: EKF performance with 100% Meaned Force Disturbance

Here we see that the EKF does not perform as well with the large (100%) non-zero meaned disturbance.

9.4.4. Sensitivity to Sensor Error

In a similar sense, the EKF can break down with non-zero mean sensor measurement characteristics. Most notably is non-zero mean orientation measurements. Below are two plots for when the orientation measurements are off by a constant angle (left plot = 10 degrees, right plot = 90 degrees)

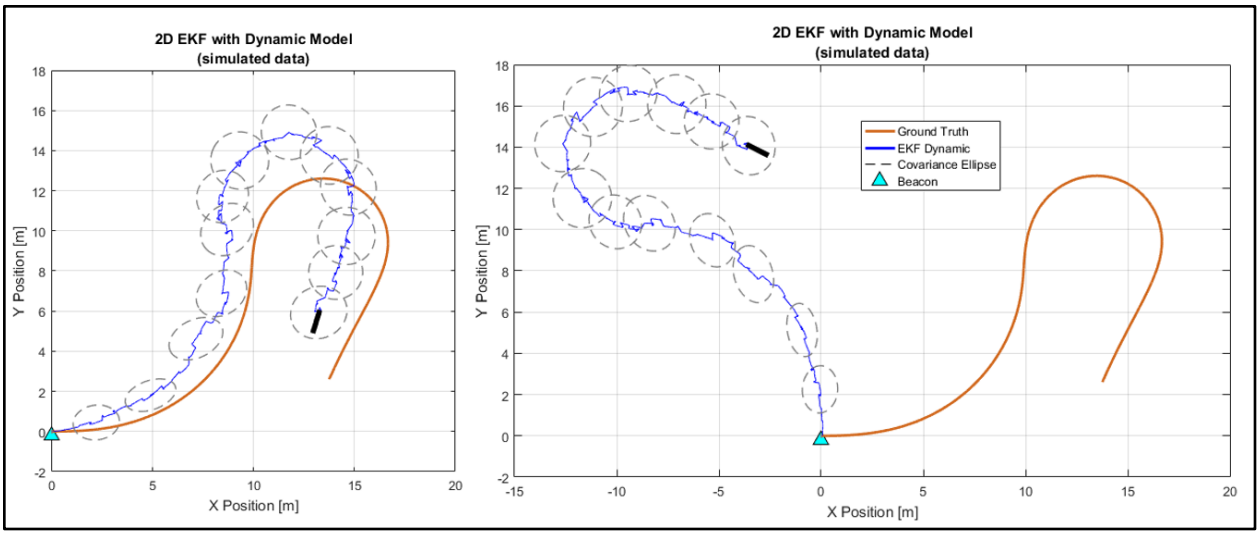


Figure 45: EKF Sensitivity to non-zero mean orientation error
(Left = 10 deg, Right = 90 deg)

These results point to one drawback of the EKF, which is the implicit assumption that errors in the system and sensors are strictly zero mean gaussian errors. Therefore the estimates cannot handle any non-zero meaned errors. Obviously the EKF is more robust to non-zero meaned errors than the direct integration method, but ultimately the algorithm struggles with reconciling the errors between measurement and system without knowing which to believe.

It is important to note that in reality, the orientation measurements of the Beluga have been very reliable since the IMU measurements are actually already filtered. The VN100 runs an internal EKF for Attitude Heading Reference System (AHRS) estimation. Thus it is not realistic to see drastic biases in the orientation measurements.

9.5. Future Work for EKF

To combat the issue with non-zero mean errors, a common practice in EKF localization is to include an estimate of the bias in the state variable. This is particularly useful for correcting for gyroscope biasing.

Additionally, the next steps for the EKF would be extending the algorithm to 3D. This would make use of all sensors onboard, including the previously unused pressure sensor.

10. Conclusion

The team's goal was to select and integrate an exteroceptive sensor into the Beluga's perception system and develop a state estimation algorithm that leveraged that sensor's data and the IMU's data. The acoustic positioning system was selected and integrated with the perception system by developing the hydrophone pipeline, though there are still some issues with that pipeline that require further work to resolve. For the signal attenuation issue, team recommends using a variable gain amplifier to increase the hydrophone signal's amplitude as the hydrophone moves farther from the acoustic beacon. A more consistent periodic chirp from the acoustic beacon would also help in reducing error in the time-of-flight calculations. Ideally, the chirp would be perfectly periodic, but a maximum error of ±0.000001 Hz in the periodicity would ensure a drift in the position estimate of no more than 0.0015 m/s, which would be a reasonable drift for our system to have.

The 2D EKF works well while running offline with simulated sensor data, but future work could involve implementing the algorithm on the Intel NUC such that it can run in real-time on sensor data given by the hydrophones and IMU.

Other future improvements could include encoding the GPS readings from the beacon into the chirp sent to the Beluga or making the beacon follow the robot as it moves, thus expanding the range of the state estimation system. Closing the control loop so that the robot corrects its motion when the state estimation system reports a state different from what the Beluga should be targeting would also be an excellent future goal.

11. Project Management

The team encountered a variety of roadblocks, hardware issues, and conceptual misunderstandings in the process of trying to reach the project goals, and so the time management of this project did not develop as anticipated at the start of the semester. The addition of two new team members also changed the team and time management of the project. Section 11.1 provides an overview of the anticipated project goals and what the team actually achieved within the given project timeline. Section 11.2 details the work breakdown structure and prediction of hours required for each task. Section 11.3 includes a Gantt chart, which helps visualize the critical path of the project this past semester. Section 11.4 describes the division of labor between team members this past semester.

11.1. Spring Overview

The main objective of this project was to develop the hydrophone pipeline as well as possible given the project's time constraint and get the 2D EKF working well enough to run on the IMU and hydrophone data as a proof-of-concept.

In the fall, the team evaluated the different exteroceptive sensor alternatives and began developing a Kalman Filter. In the spring, the team developed the hydrophone pipeline, though with some issues still limiting its performance, and the dynamic 2D EKF, though it ran offline on simulated sensor data. The final result of this project was the overall improvement of the state estimation system of the Beluga, and the team is certain about what the next steps would be to get the pipeline working more accurately and within a much wider range of the beacon and to get the 2D EKF running in real-time on the NUC. These next steps could not be completed due to the time constraints of the project, but they are detailed earlier in this report.

11.2. Work Breakdown Structure

Table 3 below depicts the work breakdown structure, which divides the major objectives of the project into smaller tasks. Each task has the approximate number of hours that were committed to the task.

Tasks	# of People	# of Hours	Total Hours
Acoustic Positioning System			
Perform initial tests to characterize the hydrophone signal	3	10	30
Drill holes in sensor ports for hydrophone wires & epoxy	2	2	4
Develop & debug signal conditioning circuit	3	15	45
Implement & debug Teensy FTM module	3	20	60
Implement & debug SPI communication	3	20	60
Implement & debug Ethernet communication	2	10	20
Develop & debug matched filter	2	20	40
Integrate entire pipeline together	4	50	200
Integrate APS with Beluga	6	3	18
Field tests	6	15	90
Extended Kalman Filter			
Develop 2D kinematic model	3	5	15
Implement 2D kinematic model	2	5	10
Research how to incorporate ToF measurements	1	5	5
Implement 2D EKF with simulated ToF measurements	1	30	30
Develop final 2D EKF for hydrophone and IMU data	2	10	20
Team Meetings			
Internal meetings	6	14	84
Advisor meetings	6	7	42
Liaison telecons	6	15	90

Reports, Documentation, & Presentation			
Spring Presentation	6	7	42
Poster	6	5	30
Final Presentation	6	7	42
Final Report	6	10	60
Other documentation	6	5	30
GRAND TOTAL			1067

Table 3: Work breakdown structure for the spring semester

11.3. Schedule Summary

The Gantt chart in Figure 45 depicts the rough workflow for the spring semester. The schedule includes the major tasks for the semester and the approximate amount of time each task took to complete.

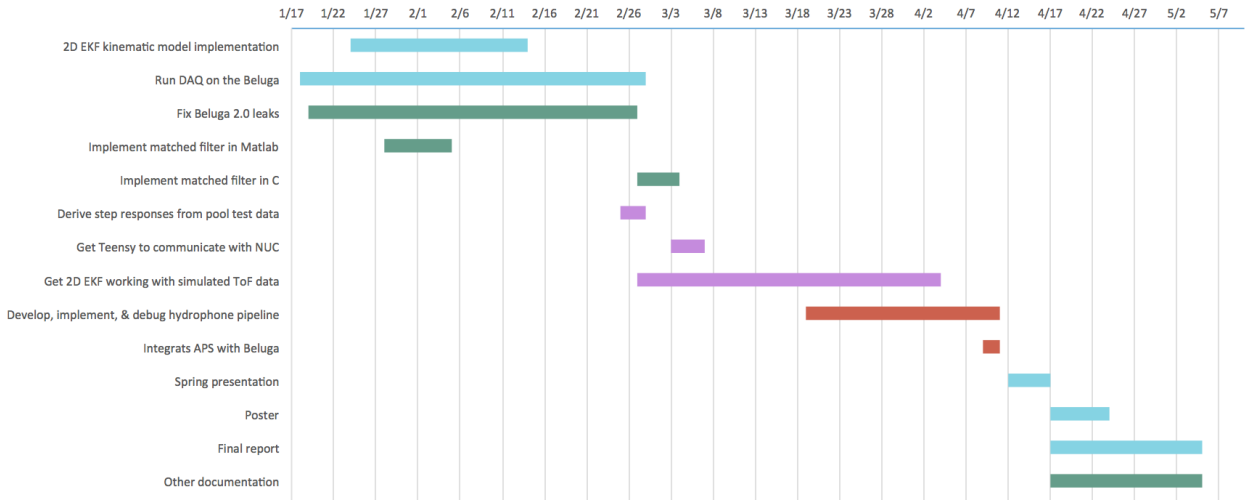


Figure 46: Gantt chart of the major tasks worked on throughout the spring semester

11.4. Division of Labor

Each team member worked on a wide variety of tasks throughout the semester, but Table 4 below summarizes the main tasks that each person worked on.

Tasks	**Owner**
Acoustic Positioning System	
Develop, implement, and debug the matched filter	Evan
Develop, implement, and debug the Ethernet communication	John
Develop, implement, and debug the SPI communication	Austin
Develop, implement, and debug the Teensy FTM module	Nancy
Develop, implement, and debug the signal conditioning circuit	Dom, Zayra
Integrate the APS with the Beluga	Dom, Zayra
Integrate the hydrophone pipeline together	Team
Perform field tests	Team
Extended Kalman Filter	
Derive the step response gains from the field test data	Dom, Zayra
Implement 2D kinematic model	John, Austin
Implement 2D EKF with simulated ToF measurements	Austin
Develop final 2D EKF for hydrophone and IMU data	John, Austin
Reports, Documentation, & Presentation	Team

Table 4: Division of labor for the spring semester

12. Acknowledgments

This project would not have been possible without the constant support and guidance of our liaisons, Jerry Hsiung (‘16), Cyrus Huang (‘16), Benjamin Chasnov (‘16), and Vaibhav Viswanathan (‘17), and our faculty advisor Professor Anthony Bright. We would also like to acknowledge the assistance of Professor Christopher Clark and Professor David Harris in the development of the Extended Kalman Filter and acoustic system pipeline, respectively. Lastly, we would like to thank the Engineering Department staff, especially Samuel Abdelmuati, Lorena Gonzalez, and Sydney Torrey, and Clinic Director Professor Kash Gokli for their management of the Harvey Mudd College Clinic program.

13. References

[1] Techmation Clinic Workplan

[2] Babb, Tim. "How a Kalman Filter Works, in Pictures." Bzarg, www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/.

[3] "The Extended Kalman Filter: An Interactive Tutorial for Non-Experts". home.wlu.edu/~levys/kalman_tutorial/.

[4] M. F. Fallon, M. Kaess, H. Johannsson and J. J. Leonard, "Efficient AUV navigation fusing acoustic ranging and side-scan sonar," *2011 IEEE International Conference on Robotics and Automation*, Shanghai, 2011, pp. 2398-2405.

[5] L. Paull, S. Saeedi, M. Seto and H. Li, "AUV Navigation and Localization: A Review," in *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131-149, Jan. 2014.

[6] Jun Jay Zhou (2008). State Estimation Strategies for Autonomous Underwater Vehicle Fish Tracking Applications. UWSpace. http://hdl.handle.net/10012/3520

[7] Kok Seng Chong and L. Kleeman, "Sonar based map building for a mobile robot," *Proceedings of International Conference on Robotics and Automation*, Albuquerque, NM, 1997, pp. 1700-1705 vol.2.

[8] Humminbird.com. (2017). *Helix Series* [online] Available at: https://www.humminbird.com/Products/HELIX-7-CHIRP-SI-GPS-G2N/ [Accessed 29 Nov. 2017].

[9] En.wikipedia.org. (2017). Underwater acoustic positioning system. [online] Available at: https://en.wikipedia.org/wiki/Underwater_acoustic_positioning_system [Accessed 25 Sep. 2017].

[10] Kaushal, Hemani. "Underwater Optical Wireless Communication." IEEE Access, vol. 4, 11 Apr. 2016. IEEEXplore.

[11] Chaplin, Martin. "Water Absorption Spectrum." Water Absorption Spectrum, 10 Nov. 2017, www1.lsbu.ac.uk/water/water_vibrational_spectrum.html.

[12] Control, B. (2017). BlueComm 200 - Wireless Underwater Video and Vehicle Control - Sonardyne. [online] Sonardyne. Available at: https://www.sonardyne.com/product/bluecomm-200-wireless-underwater-video/ [Accessed 25 Sep. 2017].

[13] "HTI-96-Min Hydrophone." High Tech, Inc. USA, High Tech Inc, www.hightechincusa.com/products/hydrophones/hti96min.html.

[14] "EV UW30 Underwater Speaker for Marine Biology." Lubell Labs Inc. http://www.lubell.com/UW30.html

[15] N. R. Rypkema, E. M. Fischell and H. Schmidt, "One-way travel-time inverted ultra-short baseline localization for low-cost autonomous underwater vehicles," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 4920-4926.

[16] Elert, Glenn. "Speed of Sound in Water." The Physics Factbook, hypertextbook.com/facts/2000/NickyDu.shtml.

[17] "OH100-61003CF-010.0M." Connor-Winfield | Crystals, Oscillators, Resonators, Digikey, www.digikey.com/short/j4chr8.

[18] "Fast sampling ADC." https://forum.arduino.cc/index.php?topic=65192.0

[19] *Features of the Flexible Timing Module*. Freescale Semiconductor, Inc. 2015. https://www.nxp.com/docs/en/application-note/AN5142.pdf

[20] "Thrusters & Motors." T200 Thruster Documentation, Blue Robotics, docs.bluerobotics.com/thrusters/t200/.

[21] Liebermann, L. N. "The Origin of Sound Absorption in Water and in Sea Water." *The Journal of the Acoustical Society of America*, vol. 20, no. 6, 1948, pp. 868–873., doi:10.1121/1.1906450.

[22] Tuite, Don. "Variable Gain Amplifiers." Electronic Design, 1 Nov. 2012, www.electronicdesign.com/analog/variable-gain-amplifiers.

[23] Lazebnik, Svetlana. "Feature extraction: Corners and Blobs." http://www.cs.unc.edu/~lazebnik/spring09/lec07_corner_blob.pdf

[24] J. Mrovlje and D. Vrančić, "Distance measuring based on stereoscopic pictures," *Proceedings of the 9th International PhD Workshop on Systems and Control, October 1-3, 2008, Izola, Simonov zaliv, Slovenia*, 2008.

Appendices

A. Camera Odometry

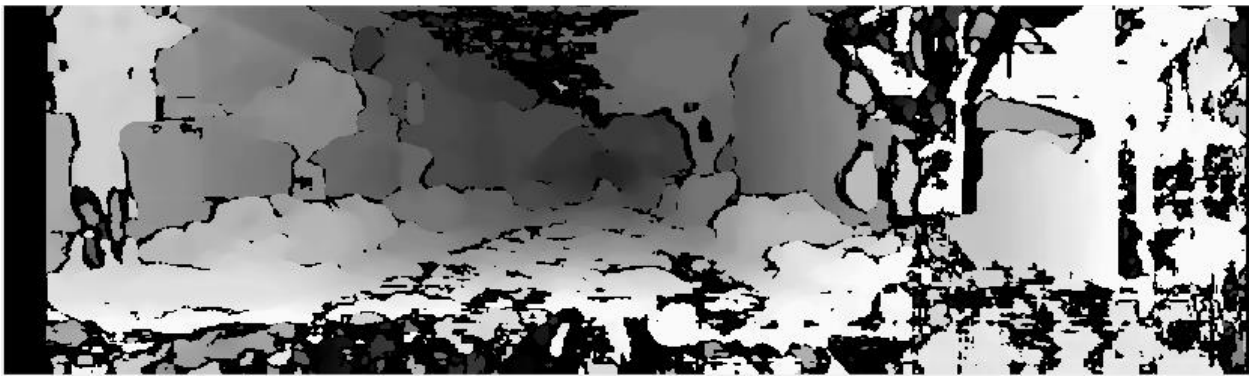
Camera odometry is effectively the same above and underwater. It can be broken down into four steps: preprocessing, creating a disparity map, detecting and tracking features, and calculating a position estimate.

In preprocessing, image processing algorithms remove all the blemishes due to taking a photos. The most predominant of these is distortion which occurs when some parts of an image are bent either away or towards the center of the camera.

The other preprocessing step is rectification which is unique to stereo camera setups. Rectification aims to reduce the separation of pixels in a pair of images to one dimension, as opposed to two. This can be achieved by projecting both images onto the same plane and extrapolating the respective position of pixels from the new plane, onto their original images.

Figure 47: Two processed images from the left and right cameras (respectively) in a stereo setup.

Once the images have been rectified a disparity map can be created. This shows the difference between each pixel value at every point in the two images. The disparity map helps highlight and preserve the 3D aspect of the image as close objects (which seem to have moved more) would have very different values from those further away.



Ffigure 48: A disparity map created from the two images in Figure 46.

After creating a disparity map for a single frame, we can detect objects in the image. This is done by searching for edges, corners, and blobs.[23] Once the features in a disparity map have

[23] Lazebnik, Svetlana. "Feature extraction: Corners and Blobs."

http://www.cs.unc.edu/~lazebnik/spring09/lec07_corner_blob.pdf

been identified, they are compared to the disparity map from the next frame to obtain an approximate distance by which they have moved. This distance would also be the distance that the AUV would have moved in the time between frames.

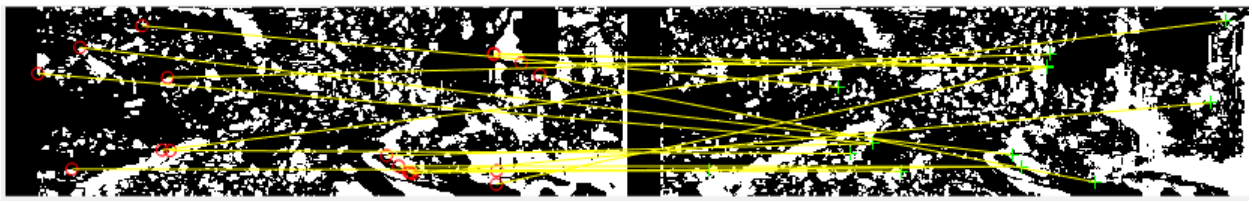


Figure 49: Feature detection and tracking between two disparity maps.

The following images illustrate how stereo cameras can be used to obtain the absolute distances of object in images[24]. The known variables are

B: the distances between the two images

x_L-x_D: horizontal difference between the same object on both pictures

x_0: the pixel width of the images

x_1: the pixel distance to the object in the image from the left camera
x_2: the pixel distance to the object in the image from the right camera
$\square_0$: the angular field of view of the cameras

[24] J. Mrovlje and D. Vrančić, "Distance measuring based on stereoscopic pictures," *Proceedings of the 9th International PhD Workshop on Systems and Control, October 1-3, 2008, Izola, Simonov zaliv, Slovenia*, 2008.

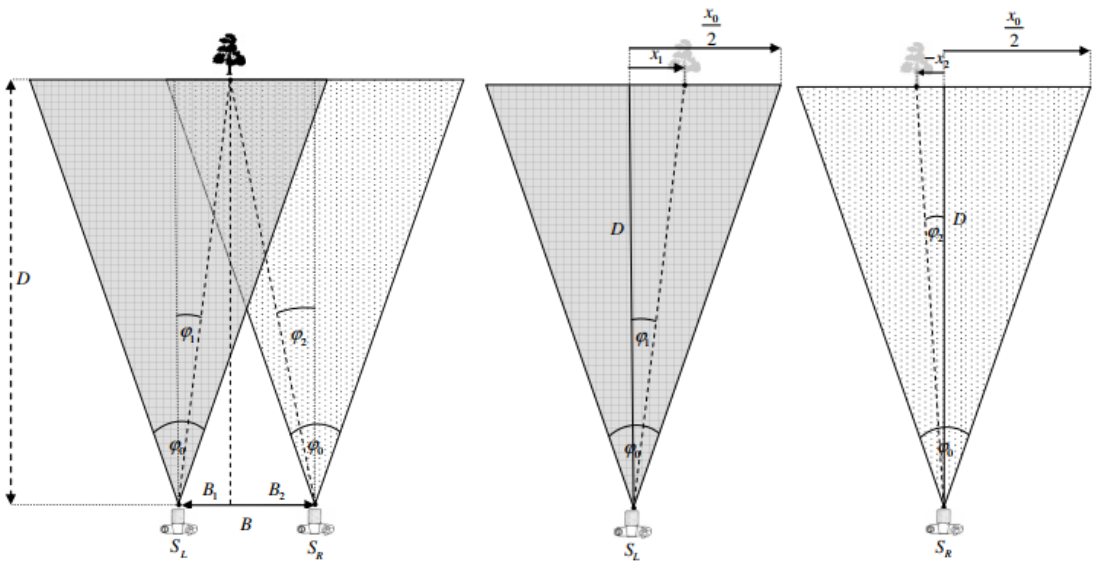


Figure 50: Diagrams showing the trigonometry involved in obtaining distances in stereo camera setups.

To find D, we can use the following set of equations:

$$B = B_1 + B_2 = D\tan\varphi_1 + D\tan\varphi_2,$$

$$D = \frac{B}{\tan\varphi_1 + \tan\varphi_2}$$

$$\frac{x_1}{\frac{x_0}{2}} = \frac{\tan\varphi_1}{\tan\left(\frac{\varphi_0}{2}\right)}, \quad \frac{-x_2}{\frac{x_0}{2}} = \frac{\tan\varphi_2}{\tan\left(\frac{\varphi_0}{2}\right)}$$

$$D = \frac{Bx_0}{2\tan\left(\frac{\varphi_0}{2}\right)(x_L - x_D)}$$

Therefore given the features of a stereo camera setup (i.e. the camera separation and focal lengths) it is always possible to find the distance to an object in an image.

Libviso is visual odometry library capable of performing all the processes outlined in this section. There is a ROS wrapper know as viso_ros, this is what the team will be implementing on the Beluga for visual odometry.

A.1. 1D Model (Line Test)

As a starting point, the team focused on a 1D Kalman Filter to become more familiar with the algorithm. The team implemented the algorithm in MATLAB for a line test that was performed with the Beluga v1.0. The team performed the test by starting the IMU and GPS data collection when the Beluga was stationary for approximately 5 seconds, then rolling the Beluga approximately 15 m in a roughly straight line at roughly constant velocity, stopping the Beluga for approximately 5 seconds, and then stopping the data collection.

Since the Beluga was merely pushed along the line, there was no motor inputs. A simple kinematic model was used, assuming a constant velocity and no acceleration throughout the test. For this formulation, the state vector is

$$\vec{x} = \begin{bmatrix} x & v & a \end{bmatrix}^T$$

There are two sensors, the accelerometer (sensor 1) and GPS (sensor 2). The sensor matrix C is

$$C = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

the following kinematic equations were used to determine the state transition matrix for the Kalman Filter:

$$d_i = d_{i-1} + v_{i-1}\Delta t + \frac{1}{2}a_{i-1}\Delta t^2$$

$$v_i = v_{i-1} + a_{i-1}\Delta t$$

$$a_i = a_{i-1}$$

Equations 1 & 2. Kinematic equations for developing *A* matrix for KF.

Converting these kinematic equations to matrix form with the state vector yields the following state-transition matrix

$$A = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

Equation 3. State transition *A* matrix for line test KF, based on Eqns. 1 & 2.

In this case, there are no control inputs so

$$B = [0]$$

The prediction covariance matrix was initialized as the identity matrix. The process noise was set to zero and the sensor covariance matrix $\underline{R}$ was adjusted to see the effects of on the estimate.

$$P = [I]$$

$$Q = [0]$$

$$R = \begin{bmatrix} \sigma_{IMU}^2 & 0 \\ 0 & \sigma_{GPS}^2 \end{bmatrix} \text{ for varying } \sigma_{IMU}^2 \text{ and } \sigma_{GPS}^2$$

The GPS on the Beluga v1.0 has a sampling rate of 1 Hz, but the IMU has a sample rate of 50 Hz, so initially the team linearly interpolated the GPS data to plot the following KF state estimates of the line test:

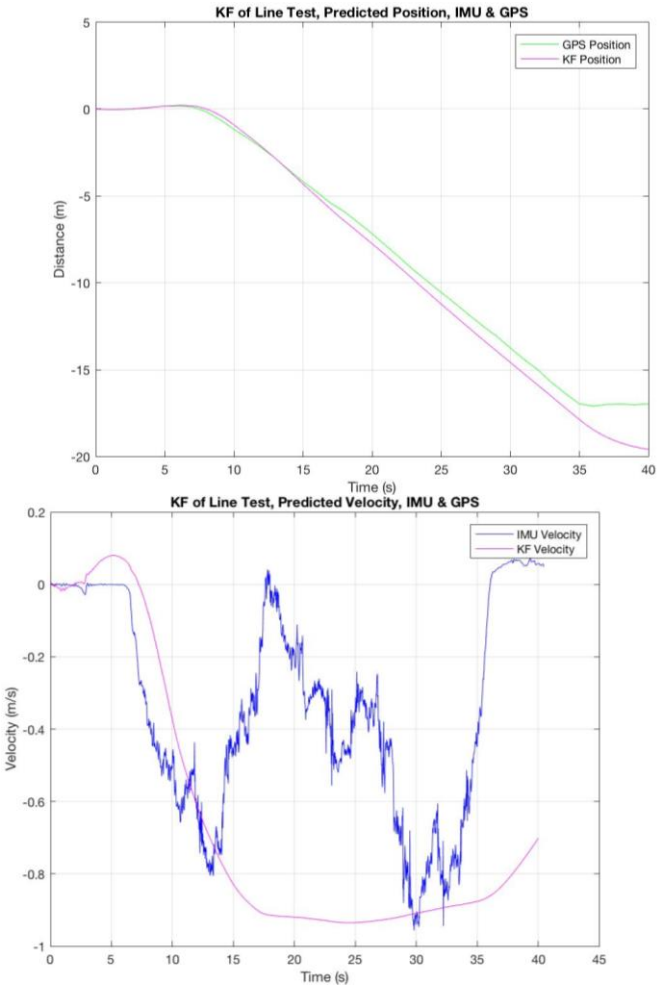


Figure 51: Position and velocity KF estimates, with GPS and IMU data.

The KF estimate is closer to the expected velocity trend than the integrated IMU acceleration data was (relatively constant velocity in the middle of the test). However, the position estimate did not appear as good as the GPS estimate (no indication of stop at end of test), regardless of how the team weighted the covariance matrices of the two sensors. The position error can be explained by the kinematic model, in that the system had no reason to expect a stop, so the model dominated the estimate such that there was constant velocity, as had been accurate for the entire middle of the test. Therefore the sensor measurements were

determined to be inaccurate, and the Kalman Gain converged towards zero (ignoring the sensors). The plot below illustrates this trend, with the Kalman Gain associated with the IMU related to acceleration, and GPS related to position (plotted logarithmically since gain converges to zero quickly).

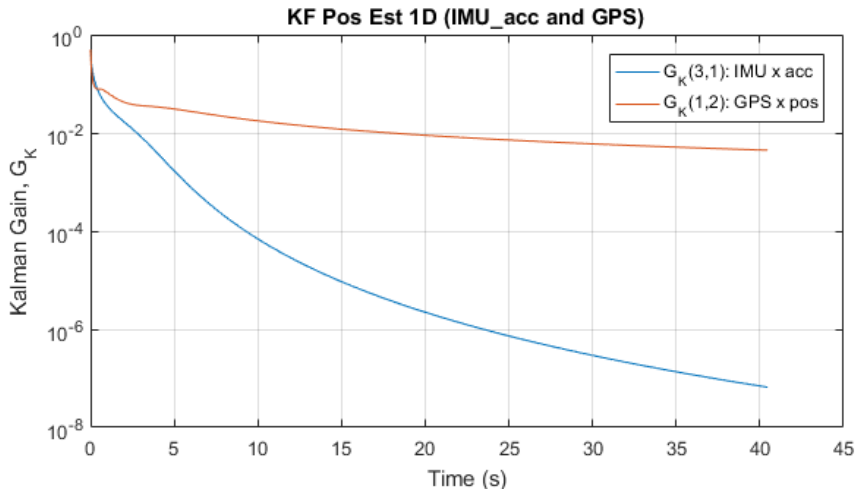


Figure 52: Kalman Gain converges towards zero, ignoring sensors

Another issue with this initial implementation was that the GPS data was linearly interpolated. Instead, the algorithm should correct its estimate with the GPS data only at timesteps for which a GPS datapoint was recorded. Therefore the algorithm was altered to have multiple predict steps, only entering the update step once a GPS data point is loaded in. Due to the difference in update rates, this means the update step happens every fifty predict steps.

This alteration means that the state vector is only position and velocity, with the acceleration treated as a control input. Thus GPS is the only sensor used in the update step (IMU acceleration measurements taken in the predict step).

$$\vec{x} = [x \; v]^T$$

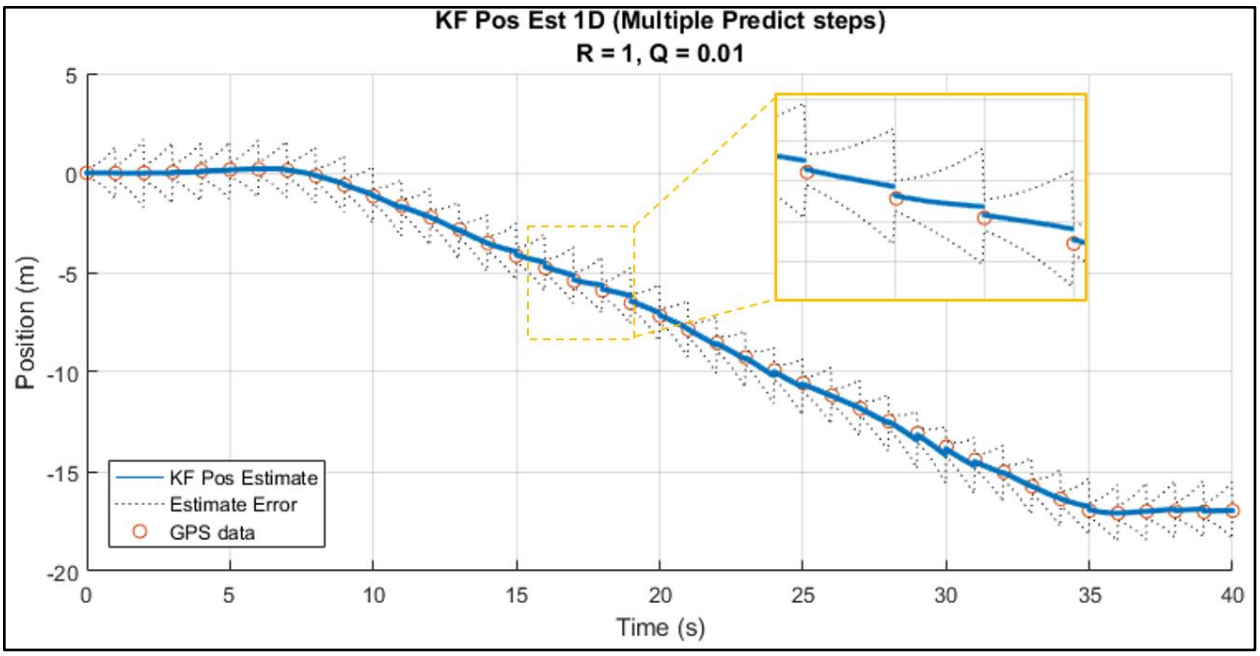
$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t^2 \end{bmatrix} \quad C = [1 \; 0]$$

As before, the prediction covariance matrix starts as the identity, and the sensor covariance is toggled. However this time, process noise is non-zero.

For the most realistic results, the team uses the RMS value of the accelerometer while stationary to estimate the process noise (~0.01 m/s^2), and the covariance of the GPS is approximately 1 m^2 (IMU covariance not included here since measurement now in predict step).

$$R = [1] \quad Q = [0.01]$$

Figure 52 illustrates the position and velocity estimates, along with the associated error at each time step (dashed black line). With this formulation, it is clear that the position estimate very closely tracks the GPS data. In between GPS measurements, the error quickly grows, but is corrected once the GPS measurement comes in again.



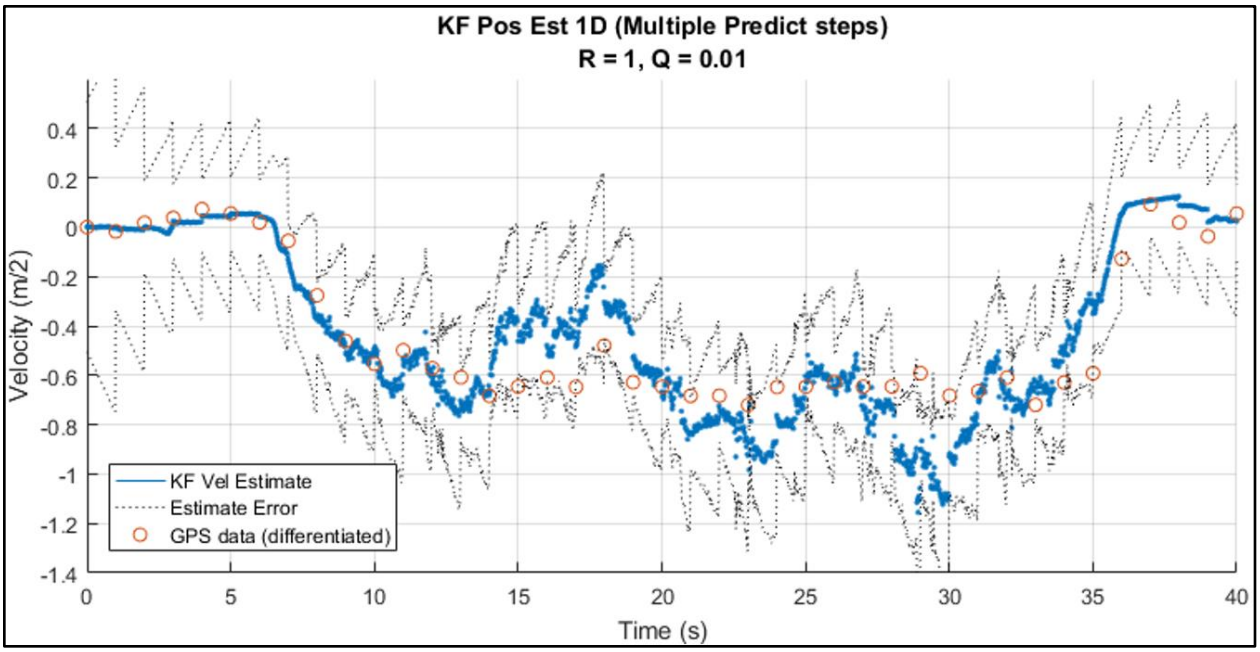
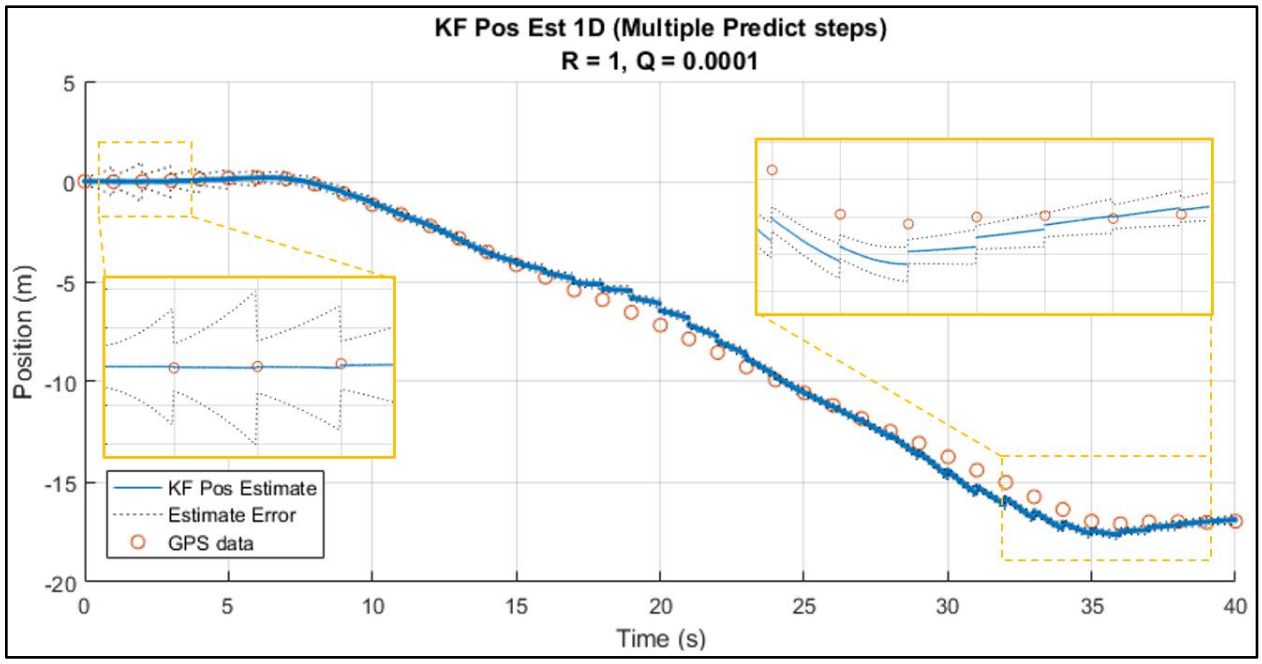


Figure 53: Kalman Filter Estimate with multiple predict steps. Top: Position vs Time. Bottom: Velocity vs Time.

Figure 54 shows the position and velocity estimates for lower process noise (Q = 0.0001), and Figure 55 shows the estimates with a more accurate GPS along with the lower process noise. In both cases, the estimate error is smaller, as expected for more accurate sensors and less noise.


KF Pos Est 1D (Multiple Predict steps)
R = 1, Q = 0.0001
Position (m)
Time (s)
KF Pos Estimate
Estimate Error
GPS data

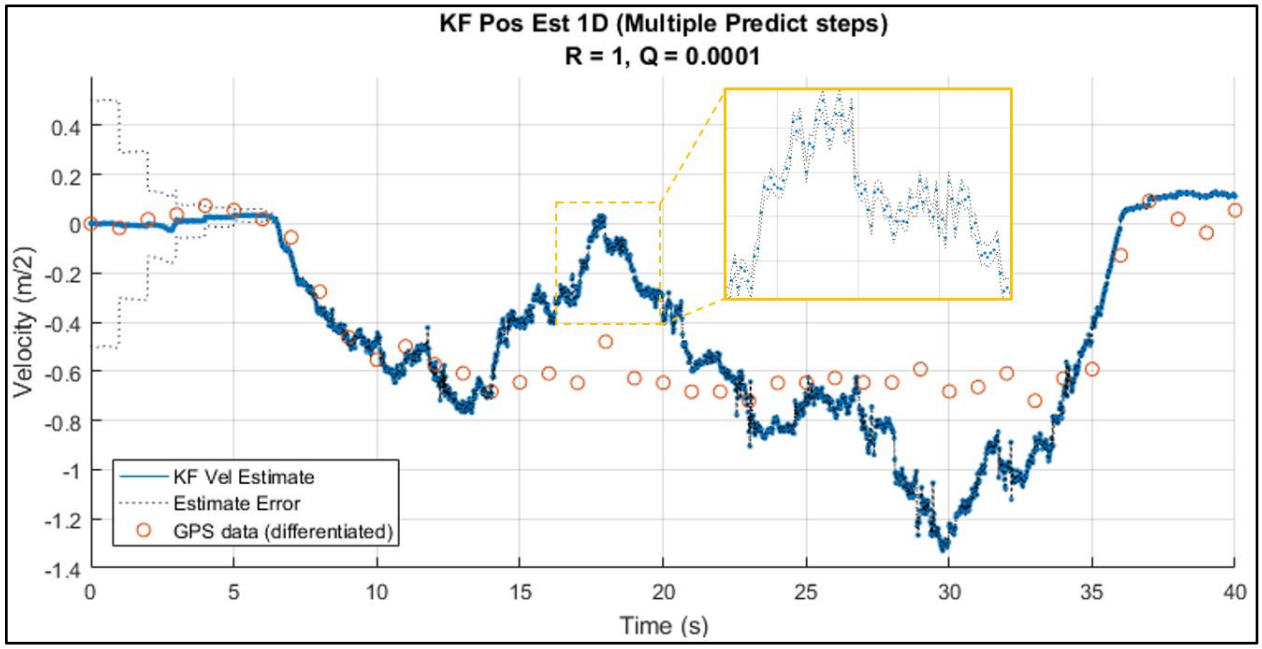

KF Pos Est 1D (Multiple Predict steps)
R = 1, Q = 0.0001
Velocity (m/2)
Time (s)
KF Vel Estimate
Estimate Error
GPS data (differentiated)

Figure 54: Estimates with less process noise (Q = 0.001). Top: Position vs Time. Bottom: Velocity vs Time.

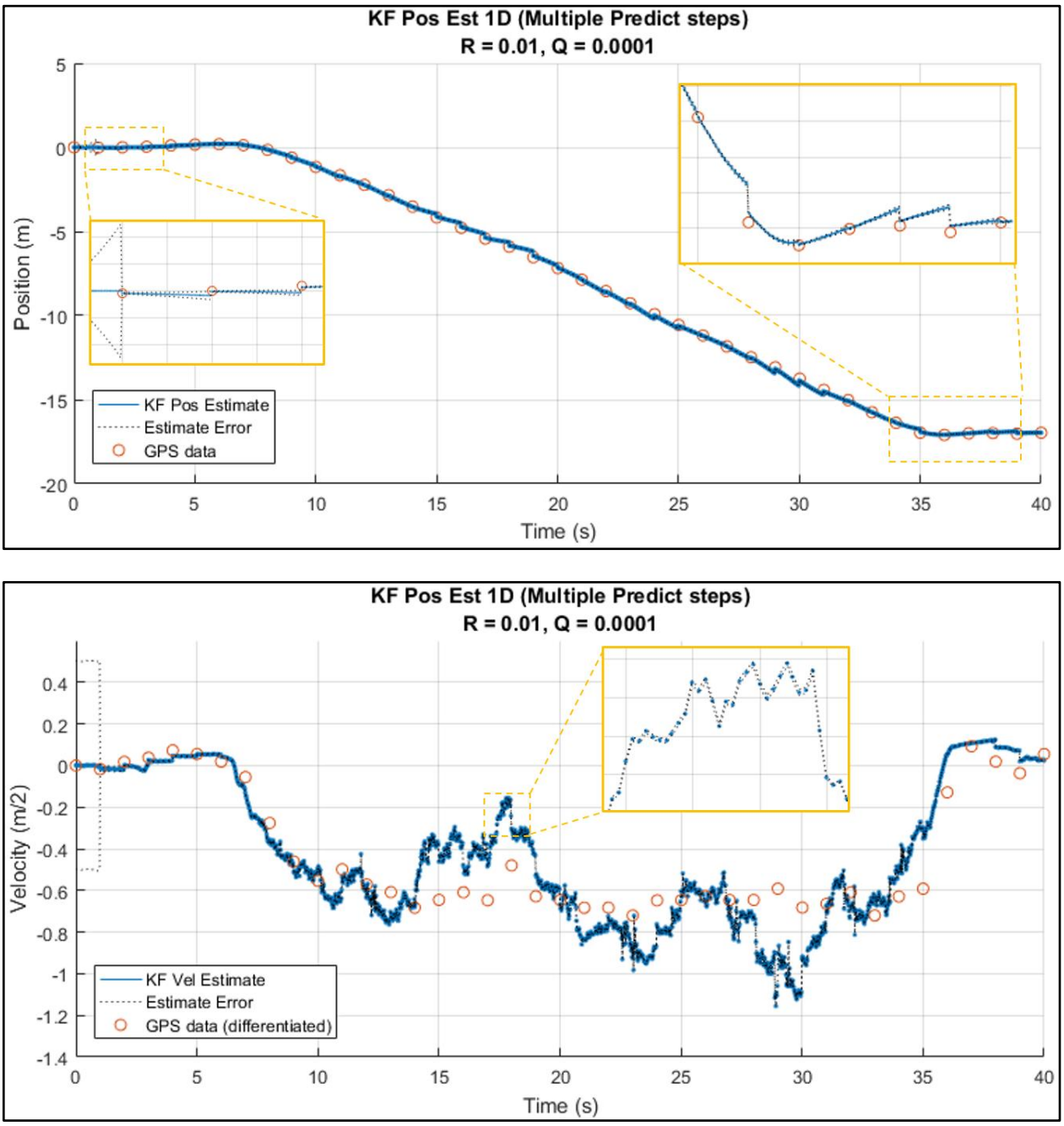


Figure 55: Estimates with less process noise (Q = 0.001) and more accurate sensor (R = 0.01). Top: Position vs Time. Bottom: Velocity vs Time.

These formulations of a 1D Kalman Filter have performed well, and the team has learn a lot in the process. Moving forward, the team plans to integrate the line test KF in the Beluga's GUI and run it in real-time as the Beluga is pushed around the parking lot. This test will verify that the algorithm that the team developed will work on real-time data within the current Beluga system.

B. EKF Derivation

All the variables used in this section can be found in section 7.4.1 for sensor measurements, and 7.4.2 for state variables.

B.1. 2D EKF Kinematic Prediction

In a 2D cartesian plane, the newly defined state space contains variables x, y that denote the position of the AUV. Their derivative with respect to time, v_x and v_y, describes the robot's velocity. Angle $\theta$and its derivative ωrepresents the robot's heading relative to the x axis and the angular velocity.

$$\vec{x} = [x \quad y \quad \theta \quad v_x \quad v_y \quad \omega]$$

Since the system is nonlinear, the prediction step can be generalize as follow, where f is the prediction function and h translates the current states into appropriate measurement.

$$\vec{x}_k = f(\vec{x}_{k-1}, \vec{u})$$
$$\vec{z}_k = h(\vec{x}_k)$$

To predict the next state, consider variable x. With a basic kinematic model, the x at time step $k$with time differenceΔt can be predicted as:

$$x_k = x_{k-1} + v_{x,k-1}\Delta t + \frac{1}{2}^{G} a_x \Delta t^2$$

The challenge here lies in the fact that the IMU measures acceleration in the robot frame. Recall in section 7.4.1, the acceleration in the robot frame is defined as ${}^{R}a_x$ for x direction and ${}^{R}a_y$ for the y direction. A rotation matrix is needed to convert from the acceleration from robot frame R to global frame *G*.

$$ {}^{R}R^{G} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} $$

Applying to the *x* kinematic equation yields prediction function for x. The *y* prediction function can also be derived in a similar manner. Thus we have our first two equations:

$$ x_k = x_{k-1} + v_{x,k-1}\Delta t + \frac{1}{2}\Delta t^2 \left[{}^{R}a_x \cos\theta_{k-1} - {}^{R}a_y \sin\theta_{k-1} \right] $$

$$ y_k = y_{k-1} + v_{y,k-1}\Delta t + \frac{1}{2}\Delta t^2 \left[{}^{R}a_x \sin\theta_{k-1} + {}^{R}a_y \cos\theta_{k-1} \right] $$

Since the θ is measured directly in the global frame, it has the following relationship:

$$ \theta_k = \theta_{k-1} + \omega_{k-1}\Delta t $$

Taking derivative of them with respect to time resulted in:

$$ \dot{x_k} = v_{x,k-1} + \Delta t \left({}^{R}a_x \cos\theta_{k-1} - {}^{R}a_y t \sin\theta_{k-1} \right) $$

$$ \dot{y_k} = v_{y,k-1} + \Delta t \left({}^{R}a_x t \sin\theta_{k-1} + {}^{R}a_y t \cos\theta_{k-1} \right) $$

$$ \dot{\theta_k} = \omega_{k-1} $$

B.2. 2D EKF Dynamic Prediction

In order to understand the robot's dynamic, we conducted several pool tests to determine the robot's linear and angular step responses. The step responses were extracted through video analysis, image processing, GPS measurements, and gyroscope measurements. Since the IMU is no longer being used as a direct measurement of robot state, it can be fused with the prediction of acceleration based on motor input. The state space vector can hence be expanded to eight states:

$$\vec{x} = \begin{bmatrix} x & y & \theta & v_x & v_y & \omega & {}^{G}a_x & {}^{G}a_y \end{bmatrix}^T$$

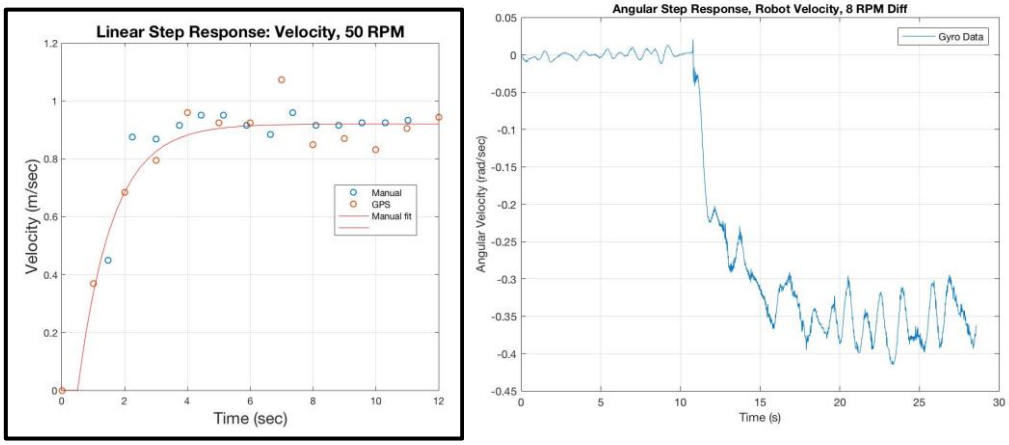


Figure 56: Experimental Step Response data for Linear (left) and Angular (right) to derive dynamic model

The linear and angular data provides linear time constant, τ_{lin}, linear gain, G_{lin}, angular time constant. τ_{lin}, and angular gain G_{ang} of the Beluga AUV. Each of the plots were fitted as a first order system. In the laplace domain, the angular and linear step response of the Beluga can be modeled as step response function for velocity and angular velocity as $Y_v(s)$ and function $Y_\omega(s)$.

$$Y_v(s) = \frac{G_{lin}}{\tau_{lin}s + 1}\frac{1}{s}$$

$$Y_\omega(s) = \frac{G_{ang}}{\tau_{ang}s + 1}\frac{1}{s}$$

Taking their derivative will return the velocity and angular velocity transfer functions H_v and H_ω

$$H_v(s) = \frac{G_{lin}}{\tau_{lin}s + 1}$$

$$H_\omega(s) = \frac{G_{ang}}{\tau_{ang}s + 1}$$

To obtain the transfer function for position, we take the integral of the velocity transfer function and obtain the positional transfer function $H_{lin}(s)$ and $H_{ang}(s)$:

$$H_{lin}(s) = \frac{G_{lin}}{s(\tau_{lin}s + 1)}$$

$$H_{ang}(s) = \frac{G_{ang}}{s(\tau_{ang}s + 1)}$$

Given the transfer functions, we can rewrite them into control canonical form with state space representation. As the angular and linear responses are both of the same form, we will only derive the linear form and apply it to the angular form. The control canonical form of the linear transfer function is listed as below, where $\vec{g_x}(u_R, u_L)$ is a nonlinear function that transform the right and left motor input, u_R and u_L, into the motor command strictly in the *x* direction.

$$\begin{bmatrix} \dot{v_x} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_{lin}} & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ x \end{bmatrix} + \vec{g_x}(u_R, u_L)$$

$$u_x = \vec{g_x}(u_R, u_L) = \frac{u_L + u_R}{2}\cos\theta$$

The equation becomes:

$$\begin{bmatrix} \dot{v_x} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_{lin}} & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ x \end{bmatrix} + \begin{bmatrix} \frac{G_{lin}}{\tau_{lin}} \\ 0 \end{bmatrix} u_x$$

Expanding this form will yield us two equations:

$$\dot{v_x} = -\frac{1}{\tau_{lin}}v_x + \frac{G_{lin}}{\tau_{lin}}u_x$$

$$\dot{x} = v_x$$

We can use digital emulation to convert the above equations into discrete time. Consider the function at time step *n*, the derivative of the variables can be rewritten as the difference between time *n* and *n-1* divided by the time step Δt.

$$\frac{v_x[n] - v_x[n-1]}{\Delta t} = -\frac{1}{\tau_{lin}} v_x[n-1] + \frac{G_{lin}}{\tau_{lin}} u_x[n-1]$$

$$\frac{x[n] - x[n-1]}{\Delta t} = v_x[n-1]$$

Multiplying Δt over and simplifying the equation will yield

$$v_x[n] = (1 - \frac{\Delta t}{\tau_{lin}}) v_x[n-1] + \frac{G_{lin}}{\tau_{lin}} u_x[n-1]\Delta t$$

$$x[n] = x[n-1] + v_x[n-1]$$

Subbing in the expression for u_x for the top equation gives us

$$v_x[n] = (1 - \frac{\Delta t}{\tau_{lin}}) v_x[n-1] + \frac{G_{lin}}{\tau_{lin}} \frac{u_R[n-1] + u_L[n-1]}{2} \cos\theta[n-1]\Delta t$$

By taking the derivative of the velocity, we can obtain the prediction for acceleration:

$${}^{G}a_x[n] = (1 - \frac{1}{\tau_{lin}}) a_x[n-1] + \frac{G_{lin}}{\tau_{lin}} \frac{u_R[n-1] + u_L[n-1]}{2} \cos\theta[n-1]$$

Subbing this back into the velocity equation, we obtain:

$$v_x[n] = (1 - \frac{\Delta t}{\tau_{lin}}) v_x[n-1] + {}^{G} a_x[n-1]\Delta t$$

The rest of the prediction can be developed with the same logic. The entire state prediction function ends up being:

$$\vec{x}_n = f(\vec{x}_{n-1}, \vec{u}) = \begin{bmatrix} v_{x_{n-1}}\Delta t + x_{n-1} \\ v_{y_{n-1}}\Delta t + y_{n-1} \\ \omega_{n-1}\Delta t + \theta_{n-1} \\ (1 - \frac{\Delta t}{\tau_{lin}})v_{x_{n-1}} + {}^{G}a_x \Delta t \cos\theta_{n-1} \\ (1 - \frac{\Delta t}{\tau_{lin}})v_{y_{n-1}} + {}^{G}a_y \Delta t \sin\theta_{n-1} \\ (1 - \frac{\Delta t}{\tau_{ang}})\omega_{n-1} + \frac{G_{ang}}{\tau_{ang}} \frac{u_{R_{n-1}} - u_{L_{n-1}}}{2} \frac{d}{2} \Delta t \\ (1 - \frac{1}{\tau_{lin}})a_{x_{n-1}} + \frac{G_{lin}}{\tau_{lin}} \frac{u_{R_{n-1}} + u_{L_{n-1}}}{2} \cos\theta_{n-1} \\ (1 - \frac{1}{\tau_{lin}})a_{y_{n-1}} + \frac{G_{lin}}{\tau_{lin}} \frac{u_{R_{n-1}} + u_{L_{n-1}}}{2} \sin\theta_{n-1} \end{bmatrix}$$

B.3. Correction (IMU)

Since the IMU are no longer used as a direct substitute of the robot dynamic model, there needs to be a way to fuse its measurements with the robot states. At time step *n*, we will have sensor measurement $\vec{z}_I$ from the IMU, which includes the robot's acceleration measurements in the x, y direction and the angle and angular velocity.

$$\vec{z}_{I_n} = \begin{bmatrix} {}^{R}a_x \\ {}^{R}a_y \\ \theta \\ \omega_z \end{bmatrix}$$

The function $h_I(\hat{x_n})$ that convert this measurement into state space is:

$$h_I(\hat{x_n}) = \begin{bmatrix} {}^{G}a_x \cos\theta + {}^{G}a_y \sin\theta \\ -a_x \sin\theta + a_y \cos\theta \\ \theta \\ \omega_z \end{bmatrix}$$

The elements of the Jacobian $H_I(\hat{x}_n)$ can be summarized as;:

$$\frac{\partial h_1}{\partial \theta} = -{}^{G}a_x \sin\theta + {}^{G}a_y \cos\theta$$

$$\frac{\partial h_1}{\partial {}^{G}a_x} = \frac{\partial h_2}{\partial {}^{G}a_y} = \cos\theta$$

$$\frac{\partial h_1}{\partial {}^{G}a_y} = -\frac{\partial h_2}{\partial {}^{G}a_x} = \sin\theta$$

$$\frac{\partial h_3}{\partial \theta} = \frac{\partial h_4}{\partial \omega} = 1$$

The rest of the *H* equals to zero.

B.4. Correction (Hydrophone)

The hydrophone pipeline returns two kinds of measurement back to the NUC. It first returns the TOF for the front hydrophone, and then it returns the TDoA between the two hydrophone. By adding the TDoA to the TOF of the front hydrophone, the TOF of the second hydrophone can be obtained.

There are two kinds of correction steps for the hydrophone measurement. The first is to convert the time of flight (ToF) hydrophone into an expected robot state and do innovation in state space. The second is to do the opposite and convert the state into expected sensor measurement and do the innovation in sensor space. The first method has more ambiguity as there are multiple possible states given one hydrophone measurements. The second method will eliminate those ambiguity by utilizing the robot current knowledge. Hence, this section will only discuss the transformation from state to sensor space.

To begin, we can first define the hydrophone measurements $\vec{z}_{hp_n}$ at time step *n*. There is the ToF measurement for the front hydrophone , r_f, and the TDoA of the hydrophones, r_d.

$$\vec{z}_{hp_n} = \begin{bmatrix} TOA \\ TDoA \end{bmatrix}$$

The function $h_{HP}(\hat{x_n})$ that convert this measurement into state space can be found with some trigonometry:

$$\vec{h}(\vec{x}_n) = \begin{bmatrix} \sqrt{(x_n + \frac{L}{2}\cos\theta_n)^2 + (y + \frac{L}{2}\sin\theta)^2} \\ \sqrt{(x_n - \frac{L}{2}\cos\theta_n)^2 + (y - \frac{L}{2}\sin\theta)^2} - \sqrt{(x_n + \frac{L}{2}\cos\theta_n)^2 + (y + \frac{L}{2}\sin\theta)^2} \end{bmatrix}$$

The Jacobian $H_{hp}(\hat{x}_n)$ can be summarized as:

$$\frac{\partial h_{HP_1}}{\partial x} = \frac{(x + \frac{1}{2}(L\cos\theta))}{\sqrt{(x + (L\cos\theta)/2)^2 + (y + (L * \sin\theta)/2)^2}}$$

$$\frac{\partial h_{HP_1}}{\partial y} = \frac{(y + \frac{1}{2}(L\sin\theta))}{\sqrt{(x + (L\cos\theta)/2)^2 + (y + (L\sin\theta)/2)^2}}$$

$$\frac{\partial h_{HP_1}}{\partial \theta} = (-(L(x + \frac{1}{2}(L\cos\theta))\sin\theta) + \frac{L\cos\theta(y + \frac{1}{2}(L\sin\theta))}{(2\sqrt{(x + (L\cos(\theta))/2)^2 + (y + (L\sin\theta)/2)^2})}$$

$$\frac{\partial h_{HP_2}}{\partial x} = \frac{(x - \frac{1}{2}(L * \cos\theta))}{\sqrt{(x - (L\cos\theta)/2)^2 + (y - (L\sin\theta)/2)^2}} - \frac{(x + \frac{1}{2}(L * \cos\theta))}{\sqrt{(x + (L\cos\theta)/2)^2 + (y + (L\sin\theta)/2)^2}}$$

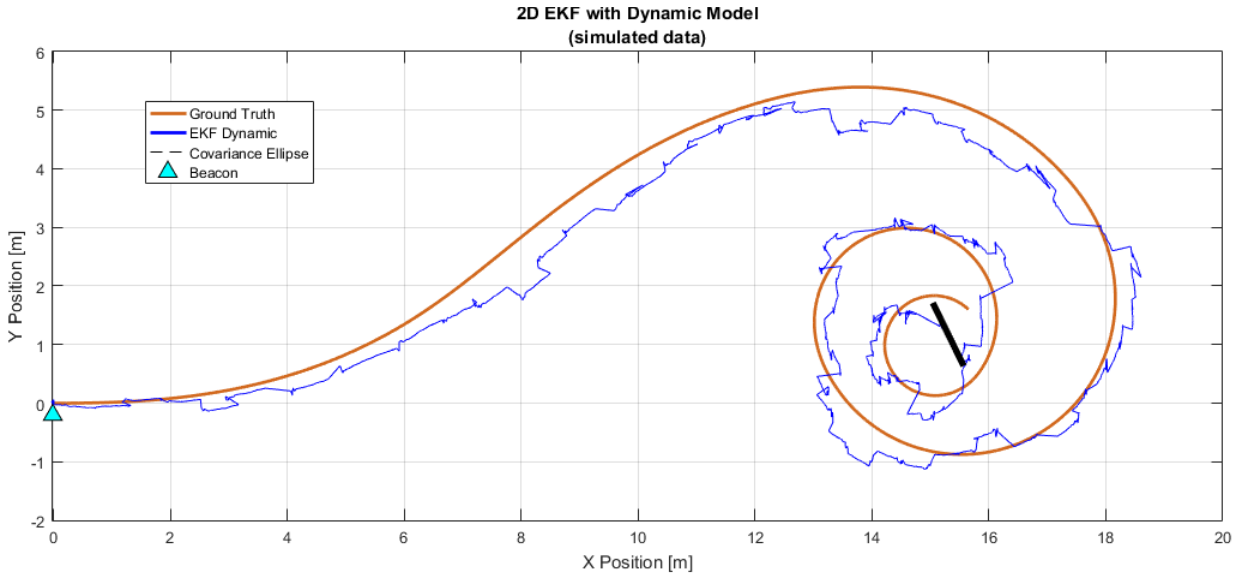
$$\frac{\partial h_{HP_2}}{\partial y} = \frac{(y - \frac{1}{2}(L\sin\theta))}{\sqrt{(x - (L\cos\theta)/2)^2 + (y - (L\sin\theta)/2)^2}} - \frac{(y + \frac{1}{2}(L\sin\theta))}{\sqrt{(x + (L\cos\theta)/2)^2 + (y + (L\sin\theta)/2)^2}}$$

$$\frac{\partial h_{HP_2}}{\partial \theta} = (L(x - \frac{1}{2}(L\cos\theta)\sin\theta) - \frac{L\cos\theta(y - \frac{1}{2}(L\sin\theta))}{(2\sqrt{(x - (L\cos(\theta))/2)^2 + (y - (L\sin\theta)/2)^2})} - \frac{\partial h_{HP_1}}{\partial \theta}$$

The rest of the Jacobian terms equals 0.

C. Extra Dynamic EKF

Below are some results of the Dynamic EKF using different trajectories.



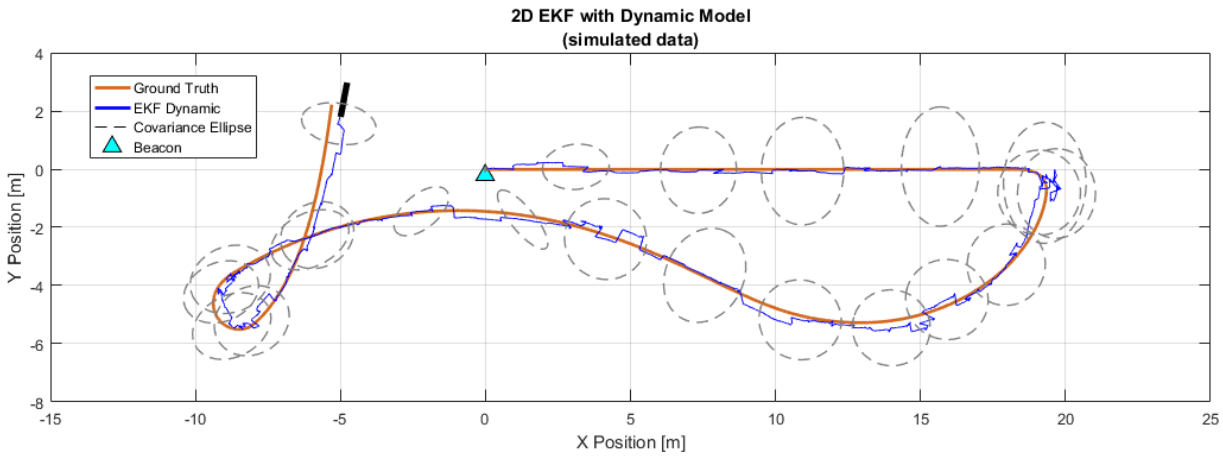


Figure 57: More generated paths and their Dynamic EKF estimates