

Underwater Object Localization using Forward Looking Sonar (FLS) and Optical Camera via Calibration and Particle Filter Fusion

National University of Singapore
Yaadhav Raaj, Dr. Ng Tek Khim

November 14, 2015

Abstract

Underwater Object Localization is widely used in the industry in Autonomous Underwater Vehicles (AUV), both in sea and lake environments for various applications. Sonars and Cameras are popular choices for this, but each sensor alone poses several problems. Data extraction from Optical Cameras underwater is a challenge due to poor lighting conditions, hazing over large distances and spatio-temporal irradiate (flickering), while Sonars tend to have coarser sensor resolution and a lower signal-to-noise ratio (SNR) making it difficult to extract data. This makes false positives more likely. In this paper, we present a robust method to localize stationary objects in front of an AUV in 3D space, using camera imagery, sonar imagery and odometry information from onboard sensors. This is done through various image processing techniques, a hybrid sonar/camera calibration step, and finally a particle filter based approach for data fusion.

Keywords: *Underwater, Localization, Particle Filters, Forward Looking Sonar, Camera*

Acknowledgements

I would like to express my deepest gratitude to my project supervisor, Associate Professor Ng Teck Khim, who has given me considerable guidance in the direction of this project as well the writing of this report.

I would also like to thank the Bumblebee AUV (BBAUV) team for providing the hardware platform and testing avenues to work on, and their continuous support throughout this project.

Table of Contents

1	Introduction	6
1.1	Motivation	6
1.2	Objective	7
2	Literature Review	7
2.1	Existing Approaches	7
2.2	Vehicle Used	9
2.3	Sonar Model	10
2.4	Camera Model	11
2.5	Estimation Theory	12
2.6	Particle Filter Algorithm	13
3	Calibration	14
3.1	Calibration Model	14
3.2	Data Collection	16
3.3	Calibration process	17
3.4	Mapping process	18
4	Particle Filter	19
4.1	Dynamic Model	19
4.2	Measurement Model	20
4.3	Particle Filter Results	22
5	Conclusion	22
6	Figures	23

List of Figures

1	SAAB Sabertooth	6
2	Objects seen in both the camera and sonar are barely visible	6
3	Great Barrier Reef 3D mapping in William's paper	7
4	Projecting sonar image into camera space on under ice topography in Spear's paper	8
5	JDPA tracking in Krout's paper	8
6	AUV Dynamic model in Miller's paper	8
7	Corresponding points between sonar and camera images in Negahdaripour's paper	9
8	BumbleBee AUV 2.5	9
9	FLS Image Formation 1	10
10	FLS Image Formation 2	11
11	Sonar Frame	11
12	Camera Calibration	11
13	Particle filter estimating pdf	13
14	Particle Filter Stages	14
15	Sonar Camera Model	15
16	Sonar Vehicle Model	15
17	Before and after applying image processing techniques on sonar image	16
18	Calibration buoy seen in the camera frame	17
19	Matching objects in the sonar and camera by projecting ambiguity lines into the camera	19
20	Sonar Features used	21
21	Large errors in pixel space (left) and 3D space (right) before calibration	23
22	Reduced errors in pixel space (left) and 3D space (right) after calibration	23
23	3D Euclidean error between resolved 3D coordinates from sonar and camera data vs ground truth in metres .	24
24	2D Pixel Euclidean error between projected pixels and ground truth before and after calibration	25
26	Testing calibrated parameters derived from training set against 4 test sets	27
27	Object not visible in camera, hence bigger uncertainty in elevation as seen in the camera image	28
28	Elevation ambiguity corrected via camera. Also loss of sonar information at that instant, but position still maintained through camera and odometry fusion but with increased covariance	28
29	Object not visible in sonar, hence ambiguity in range	28
30	Objective is to localize the task board located nearly 15m away in murky water and is barely visible in either sensor	29
31	Particle Filter still localizes the object in 3D space	29
32	Particle Filter localizes the object despite noisy data in both sensors when close to a wall	29
34	Comparison of object coordinates derived through the mapping method vs particle filter	30

1 Introduction

Vision plays a key role in object localization in underwater environments. Object localization in such environments is used extensively in AUV's, for commercial applications such as undersea pipeline analysis for the oil and gas industry, or scientific research such as the study of marine life. Many of these applications require the vehicle to move towards a target and get close enough before performing more complex tasks. In order to perform this localization step, having AUV's equipped with sonars and optical cameras have become a standard practice. An example of this is the SAAB Sabertooth.



Figure 1: SAAB Sabertooth

In our case, we wish to design a localization method for our vehicle, the Bumblebee AUV 2.5. This is then tested in multiple AUV competitions such as the Singapore AUV Challenge (SAUVC) and Robosub in San Diego, where tasks simulating industry and military use cases are required to be performed. Some of these tasks include having to localize and approach objects up to 15m away in murky water conditions, such as buoys which simulate mines and target boards which simulate items that an AUV has to get close enough to perform a task with its end-effector.

1.1 Motivation

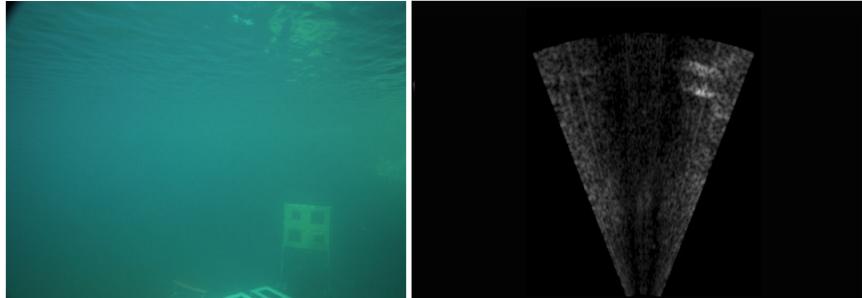


Figure 2: Objects seen in both the camera and sonar are barely visible

Optical cameras are extensively used for this purpose, where one can make use of various image processing algorithms to find a region of interest (ROI) and its centroid (U_c, V_c) to track. However, finding this from the image alone is difficult. In shallow environments, noise can be attributed to spatio-temporal irradiance (flickering) and scattering [23]. This makes algorithms for thresholding or segmentation work ineffectively, which makes localization difficult. In deep environments, hazing due to low light conditions and suspended particles in the water make segmentation via Edge based or HSV color space based segmentation difficult. Bianco [5] uses the dark channel prior estimate to dehaze the scene, and get a depth map as well, but this method cannot run in real time. Camera's are unable to perceive depth or bearing without resorting to Structure from Motion (SFM) or Stereo Camera techniques, both which require good features to track and hence are susceptible to the issues above.

Sonars are also extensively used for this purpose. Two popular models in ROV's/AUV's are the DIDSON (Dual-Frequency Identification Sonar) and the Teledyne Blueview series, based on blazed array technology. The sonar used here is the Blueview P450-E [2]. It is able to provide video imagery at up to 12Hz, and has a maximum Range (R_s), Azimuth

(Θ_s) , and Elevation (Φ_s) of 200m, 45° and 15° respectively. Similar algorithms are used to extract Range (R_s) and Azimuth (Θ_s) of objects from the sonar imagery. However, the coarser sensor resolution combined with the lower signal-to-noise ratio (SNR) make data extraction challenging. Reflections from walls, or waves on the water surface produce large amount of noise. Objects out of the frustum of the sonar look noisy. Furthermore, most mid-range systems such as the one used here are unable to provide Elevation (Φ_s) data, making it impossible to compute 3D coordinates, and hence correct for depth.

1.2 Objective

The visibility limitations, intermittent loss of data due to high SNR and lack of 3D information of the object make localization difficult. In this paper, we propose a novel method for fusing information from the camera and sonar imagery, along with odometry information from the vehicle using particle filters to localize objects, assuming prior information such as the dimensions and color of said object is provided.

This paper assumes that both sensors are arranged with overlapping views, and hence it would be possible to model the transformational geometry between both spaces. It assumes that sonar is factory calibrated, and that a pre-calibration step on the camera is performed to find its intrinsic matrix. The premise is to acquire (R_s, Θ_s) and (U_c, V_c) from the sonar and camera data to compute the 3D coordinates (X_s, Y_s, Z_s) in the sonar frame, which is then used for path planning purposes. Lastly, it assumes that odometry information, such as angles of rotation $(\alpha_v, \beta_v, \gamma_v)$, angular velocities $(\dot{\alpha}_v, \dot{\beta}_v, \dot{\gamma}_v)$, translational velocities $(\dot{X}_v, \dot{Y}_v, \dot{Z}_v)$, and depth (D_D) from the water surface are known, possibly through the use of a Doppler Velocity Log (DVL), Inertial Measurement Unit (IMU), and pressure sensor about the vehicle center of mass (COM). These are assumed to be calibrated as well.

2 Literature Review

2.1 Existing Approaches

Many previous sensor fusion approaches that used range and bearing (LIDAR, RADAR, SONAR) with an optical camera have been tested and used well-defined environments with many features to track, where the sonar points towards the surface but not ahead [25] [26]. In the seminal paper describing 3D mapping of the great barrier reef [25], the transformations between the sensors appear to be already known/calibrated, and particle filters are used here in the SLAM process, where the motion model of the particles use vehicle odometry data. The bottom camera feed has significant features to track in order to perceive range and is at close proximity to the objects of interest. This might not be the case in the forward range where objects are much further away with poor visibility. However, many useful ideas, such as projection of the sonar features into the camera frame, and involvement of vehicular odometry in localizing stationary objects are later used in this paper.



Figure 3: Great Barrier Reef 3D mapping in William's paper

Since the FLS does not provide elevation information, methods such as projecting the sonar ambiguity space has also been explored. In Anthony Spears paper [21], where the use case was for localizing underwater ice systems in the Antarctic regions, objects detected in the sonar data are mapped directly into the camera image as an area of interest in which to search for landmarks. Although a precise calibration method is not used here, the field of view of both sensors are used to determine a bounding box where the object may exist in the camera frame. Such ideas were used in the paper as well. However, this method may not have been enough to localize precisely or acquire said objects 3D coordinates.

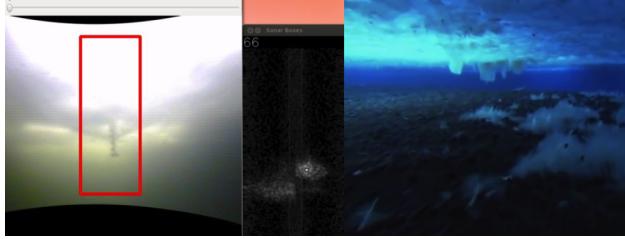


Figure 4: Projecting sonar image into camera space on under ice topography in Spear's paper

Motion tracking approaches fusing information from sonar and camera data using a Joint Probabilistic Data Association (JPDA) fusion approach has also been studied [14]. Information from the camera is used to correct the position of the floating buoy in a planar space. This method works well as the camera is on the surface, allowing better tracking on that space. However, our use case requires the system to localize objects in 3D space while fully submerged, and fusion in 2D space may only aid in visual servoing which is not the intention.

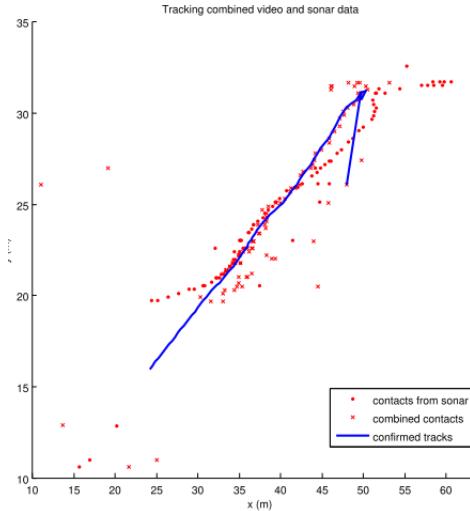


Figure 5: JPDA tracking in Krout's paper

SLAM approaches using particle filters in underwater environment [6], and AUV Rigid body dynamics [16] have also been studied. Clark's paper talks about the use of a PHD particle filter on FLS data, incorporating a dynamic model based on velocity and a measurement model based on intensity and centroids of thresholded objects in the sonar imagery. Miller's paper talks about the dynamic model used in a particle filter to incorporate data from the AUV's pressure sensors, DVL, IMU and an Acoustic Long Baseline (ALB). Ideas from the measurement models used in Clark's paper and vehicular dynamic models in Miller's paper are later used.

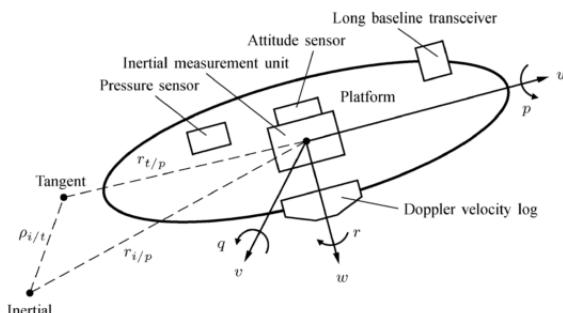


Figure 6: AUV Dynamic model in Miller's paper

Lastly, state of the art calibration techniques between sonar and camera spaces by S. Negahdaripour have also been extensively studied [18] [17]. In these papers, opti-acoustic feature matching and the epipolar geometry between both sensors is discussed. The equations describing both sensor spaces include the polar to cartesian conversions of the sonar space, transformation matrix between both sensors, perspective projection of 3D points into the camera space as pixels and a closed formed solution solving for Φ_s . The paper describes the use of the Levenberg Marquardt algorithm [15] for non-linear optimization in order to solve for the unknown transformation parameters, and also shows the use of a special calibration grid. Although a much simplified model is used later on for our use case, many ideas are taken from these two papers.

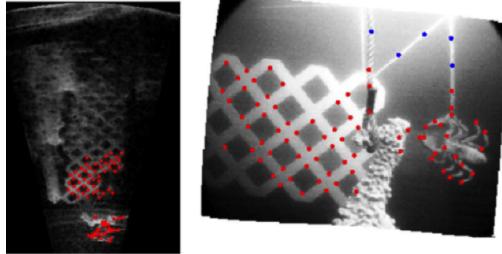


Figure 7: Corresponding points between sonar and camera images in Negahdaripour's paper

In the above approaches, methods that involved fusion of both sensors had the availability of a rich set of features to track, due to certain sensors being out of the water, or being in close proximity with the floor. In our scenario however, we are required to localize objects in front of the AUV at ranges of up to 15m, and we may not have this luxury. Furthermore, at those ranges, the minimal features that we do track in both sensors may have large errors without proper calibration. In all of the above approaches, we are yet to find a method that achieves centimetre level accuracy at those ranges, where camera, sonar and vehicular dynamics are all used to localize objects in front of the AUV, through the use of statistical filters. In order to achieve our objectives, we aim to fill this gap.

2.2 Vehicle Used

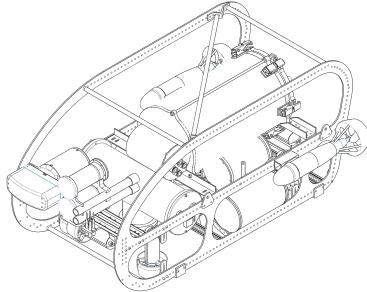


Figure 8: BumbleBee AUV 2.5

The vehicle used for testing is the Bumblebee Autonomous Underwater Vehicle (BBAUV) [1]. BBAUV is equipped with a Guppy Pro F-503 5 Megapixel Color CMOS Camera, Blueview P450-E Imaging Sonar for vision, Teledyne TDI Explorer DVL (Doppler Velocity Log), Sparton AHRS-8 IMU, Reson TC4013 hydrophones and a depth sensor for navigation and acoustics, and 2 VideoRay and 6 SeaBotix Thrusters for up to 6DOF. Fusion of the DVL and IMU via an Extended Kalman Filter provides odometry information and allows the vehicle to move to any 3D Coordinate. The US300 pressure depth sensor provides depth with an accuracy of +/-1cm. The sonar, camera and odometry sensors lie in the same rigid body frame where the transformations between them is known through the mechanical model. This information will be later used as initial values for the calibration process.

Weight	50 kg
Dimensions	0.7m X 1.1m X 0.5m
Single Board Computer	Core i7 - 3610QE Aaeon EMB-QM77 8GB DDR3 RAM 512GB SATA3 SSD
Embedded System	Arduino Mega 2560 Xilinx Spartan-3 on NI sbRIO 9602
Propulsion	6 SeaBotix BTD150 2 VideoRay Surge Thrusters
Navigation	Teledyne RDI Explorer DVL Sparton GEDC-6 IMU US300 Pressure/Depth Sensor
Vision Sensors	AVT Guppy Pro AVT Guppy
Sonar	BlueView P450 Imaging Sonar 4 Teledyne Reson TC4013 Hydrophones
Manipulators	Festo Pneumatics Systems
Power Supply	22.2V 10000mAh LiPo Battery (x2)
Underwater Connectors	SubConn Micro and Low Profile Series
Software Architecture	Robot Operating System (ROS) Debian GNU/Linux x64

Table 1: BumbleBee AUV 2.5 Specifications

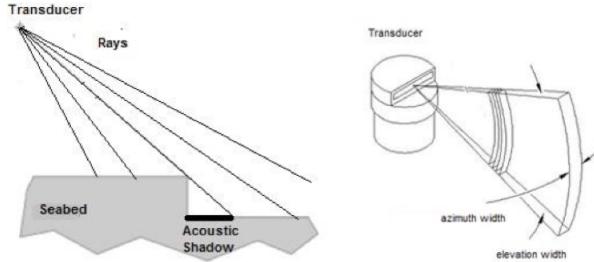


Figure 9: FLS Image Formation 1

2.3 Sonar Model

Concepts and equations behind the FLS are based off the Springer handbook of robotics [4] while understanding of formation of FLS imagery is based off [7]. An FLS is made of an array of vertical and horizontal transducers, which generate pulses of high frequency noise (450 KHz for the P450-E). The FLS Model is usually represented in spherical coordinates of azimuth, elevation angles, and range. They are also equipped with multiple transducers (The P450-E has 256 transducers). Each individual transducer can be represented as an optical ray, where the azimuth and elevation are 1° and 15° respectively as seen from Fig. 9. Hence, the horizontal beam spacing is about $256/45=0.18^\circ$.

Sonars form images using time of flight of pulses. Each transducer output/ray can be plotted as a function of time which consists of both Intensity and Range as seen from Fig. 10. Objects that are further away have a lower intensity due to signal attenuation in the medium. Also, there is a possibility that two objects at different Φ_s but same R_s may overlap in the image causing it to be impossible to perceive both objects separately. Furthermore, reflections from walls may result in objects appearing further than they actually are.

Intensity of the object is also dependant on the angular deviation from the line-of-sight of the transducer [8]. λ is the sound wavelength, a and b are the length and the width of the aperture of the transducer respectively, α and β are the coordinates of the vector of angular deviation and $Io(R)$ is the sound intensity at range R on the beam axis as seen from Eq. 1. Assuming the vehicle has an operating depth of less than 100m, based on the depth profile, it would have a λ of 450khz/1500

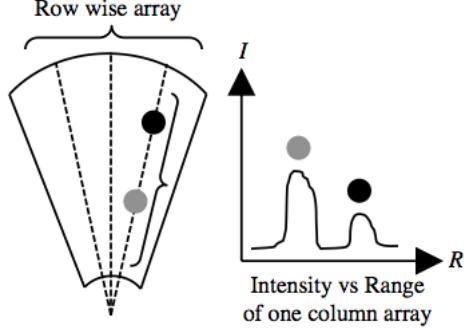


Figure 10: FLS Image Formation 2

$= 0.003\text{m}$.

$$I(R, \theta) = I_0(R) \left(\frac{\sin p}{p} \right)^2 \left(\frac{\sin q}{q} \right)^2 \quad (1)$$

$$p = \frac{\pi a \alpha}{\lambda} \quad q = \frac{\pi b \beta}{\lambda} \quad (2)$$

Lastly, the geometry of the object can be represented by Fig. 11 and Eq. 3. Here, we represent the sonar frame and the object coordinates in that frame in both cartesian and polar form. An alternative form is also used in this case, as the sensor inputs to form the 3D position of the object happen to be R_s , Θ_s and Y_s . Φ_s is not a sensor output provided by the sonar and is actually a value we wish to solve for later in the paper.

$$\begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} = \begin{bmatrix} R_s \cdot \cos(\phi_s) \cdot \sin(\theta_s) \\ R_s \cdot \sin(\phi_s) \\ R_s \cdot \cos(\phi_s) \cdot \cos(\theta_s) \end{bmatrix} = \begin{bmatrix} \sqrt{R_s^2 - Y_s^2} \cdot \sin(\theta_s) \\ Y_s \\ \sqrt{R_s^2 - Y_s^2} \cdot \cos(\theta_s) \end{bmatrix} \quad (3)$$

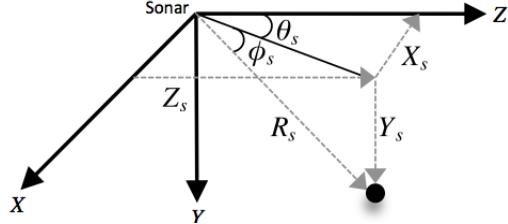


Figure 11: Sonar Frame

2.4 Camera Model

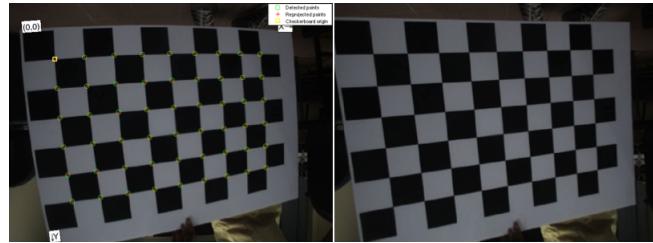


Figure 12: Camera Calibration

The camera is geometrically calibrated via Zhangs method [22] using a checkerboard pattern. Multiple checkboards images are used, and are used to estimate the cameras distortion parameters and intrinsic matrix. This allows us to map

real world coordinates onto the camera plane by finding the focal lengths and principal points as seen in Eq. 4. To account for camera distortion, the radial coefficients (K Values) are also estimated using Brown-Conrady model as seen in Eq. 6. The results and the equations used are shown below:

$$\begin{bmatrix} U_c \\ V_c \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{(f_x*X_c)+(c_x*Z_c)}{Z_c} \\ \frac{(f_y*Y_c)+(c_y*Z_c)}{Z_c} \\ 1 \end{bmatrix} \quad (4)$$

$$r = \sqrt{((U_D - c_x)/f_x)^2 + ((V_D - c_y)/f_y)^2} \quad (5)$$

$$\begin{bmatrix} U_U \\ V_U \end{bmatrix} = \begin{bmatrix} f_x * (((U_D - c_x)/f_x) * (1 + K_1 * r^2 + K_2 * r^4)) + c_x \\ f_y * (((V_D - c_y)/f_y) * (1 + K_1 * r^2 + K_2 * r^4)) + c_y \end{bmatrix} \quad (6)$$

2.5 Estimation Theory

Estimation theory is widely used in many robotics applications. The problem is to sequentially estimate the state of a dynamic system (in this case our object to be localized) using noisy measurements from our various sensors. We have a state vector that contains all information required to describe the system (Eg. position and velocity of the object w.r.t the vehicle in 3D space), and we have a measurement vector that contain observations that are related to the state vector through some model (Eg. range and azimuth from the sonar image). A dynamic model is used to describe the system dynamics, such as predicting the next position of an object, and a measurement model is used to relate the measurement and state vectors. Much of the theory and equations used is extracted from Trieb's paper [24].

One way to describe these models is through a Recursive Bayesian Filter, which consists of a prediction and an update stage and assumes the state's uncertainty is distributed through a Gaussian distribution. The prediction step predicts the next pdf using the dynamic model, but uncertainty is added due to system noise (Eg. using odometry data to predict the object's next position has noise), hence stretches the distribution. The update stage then squeezes the distribution using Bayes' theorem. These steps can be described in Eq. 7/8 where s_k is the state of the system at time k , f is the dynamic model with process noise w_k that is used to predict the next state s_{k+1} . h is the measurement model with measurement noise v_k that is used to correct s_k .

$$s_{k+1} = f(s_k, w_k) \quad k \in \mathbb{N} \quad (7)$$

$$z_k = h(s_k, v_k) \quad k \in \mathbb{N} \quad (8)$$

Given a set of prior observations up to a time k , we wish to find the optimal estimate for s_k . If $p(s_k|Z_{k-1})$ is known, the correction step of the pdf can be derived as seen in Eq. 10. If $p(s_k|Z_k)$ is known, the prediction step of the pdf can be derived as seen in Eq. 11.

$$Z_k = \{z_0, \dots, z_k\} = \{Z_{k-1}, z_k\} \quad (9)$$

$$\begin{aligned} p(s_k|Z_k) &= \frac{p(Z_k|s_k)p(s_k)}{p(Z_k)} = \frac{p(z_k, Z_{k-1}|s_k)p(s_k)}{p(z_k, Z_{k-1})} \\ &= \frac{p(z_k|s_k, Z_{k-1})p(Z_{k-1}|s_k)p(s_k)}{p(z_k|Z_{k-1})p(Z_{k-1})} = \frac{p(z_k|s_k)p(Z_{k-1}|s_k)p(s_k)}{p(z_k|Z_{k-1})p(Z_{k-1})} \\ &= \frac{p(z_k|s_k)p(s_k|Z_{k-1})}{p(z_k|Z_{k-1})} \end{aligned} \quad (10)$$

$$\begin{aligned} p(s_{k+1}, s_k|Z_k) &= p(s_{k+1}|s_k, Z_k)p(s_k|Z_k) = p(s_{k+1}|s_k)p(s_k|Z_k) \\ p(s_{k+1}|Z_k) &= \int_{\mathbb{R}^n} p(s_{k+1}|s_k)p(s_k|Z_k)ds_k \end{aligned} \quad (11)$$

The recursive propagation of the pdf can only be done if the system is linear and gaussian, which might not be the case

for us. To account for this, one way would be to linearise the system then applying a Kalman Filter (Extended Kalman Filter EKF).

The Kalman Filter assumes that the pdf at every time step is Gaussian, and can be described by a mean and covariance. If we assume the dynamic and measurement models are linear with some gaussian noise added, the filter can be described based on these Eq. 12/13 where F_k and H_k describe the dynamic and measurement models as linear functions, w_k , v_k and s_0 are normally distributed based on Eq. 14.

$$s_{k+1} = F_k s_k + G_k w_k \quad k \in \mathbb{N} \quad (12)$$

$$z_k = H_k s_k + v_k \quad k \in \mathbb{N} \quad (13)$$

$$w_k \sim N(0, Q_k), \quad v_k \sim N(0, R_k), \quad s_0 \sim N(\hat{x}_0, P_0) \quad (14)$$

The Kalman Filter proves that when Z_k is given, s_k and s_{k+1} are distributed according to Eq. 15/16, and the filter steps is given by Eq. 17. This allows one to recursively compute the mean and covariance of the next state given that the system is linear and gaussian to begin with.

$$p(s_k | Z_k) \sim N(\hat{s}_{k|k}, P_{k|k}) \quad (15)$$

$$p(s_{k+1} | Z_k) \sim N(\hat{s}_{k+1|k}, P_{k+1|k}) \quad (16)$$

$$\begin{aligned} \hat{s}_{k|k} &= \hat{s}_{k|k-1} + P_{k|k-1} H_k^T S_k^{-1} (z_k - H_k \hat{s}_{k|k-1}) \\ P_{k|k} &= P_{k|k-1} - P_{k|k-1} H_k^T S_k^{-1} H_k P_{k|k-1} \\ S_k &= R_k + H_k P_{k|k-1} H_k^T \\ \hat{s}_{k+1|k} &= F_k \hat{s}_{k|k} \\ P_{k+1|k} &= F_k P_{k|k} F_k^T + W_k Q_k W_k^T \end{aligned} \quad (17)$$

In a nonlinear non-gaussian model, one approach would be to use a particle filter. Here, the pdf is represented as a set of samples with weights. Similar to the first two methods, a dynamic model and a measurement model is used, along with the process and measurement noise. Here however, Monte Carlo (MC) methods are used where the objective is to approximate integrals of the form based on Eq. 18 where $p(s)$ represents a probability function that sums to 1. If we can draw N samples according to $p(s)$, then the integral can be approximated by the sum based on Eq. 19. As N approaches infinity, we can describe the pdf more precisely and we approach an optimal Bayesian estimate. As seen in Fig. 13, particles are able to model multivariate distributions through uneven partitioning, and regions of high/low density.

$$I = \int g(s)p(s)ds \quad (18)$$

$$\hat{I} = \frac{1}{N} \sum_{i=1}^N g(s^i) \quad (19)$$

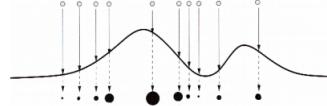


Figure 13: Particle filter estimating pdf

2.6 Particle Filter Algorithm

1: *Initialisation :*

Generate N particles according to an initial distribution $p_0(s_0)$ set by the user. If not set, assume a uniform distribution over a bounded space.

2: *Dynamic Model update* :

Time evolution of particles occurs through the dynamic model, with process noise added in.

$$X_{n|n-1}^{(k)} = f(X_{n-1|n-1}^{(k)}, w_{n-1}) \quad k \in \mathbb{N}$$

3: *Redistribution* :

A certain percentage of particles are randomly selected and then uniformly distributed over the workspace to solve the captured robot problem as suggested by Sebastian Thrun's paper [11].

4: *Measurement Model update* :

Compute weights for each particle given a measurement Y_n and normalize

$$q_n = \frac{P(Y_n | X_{n|n-1}^{(k)})}{\sum_k P(Y_n | X_{n|n-1}^{(k)})} \quad k \in \mathbb{N}$$

5: *Resample* :

Generate N new particles by resampling with replacement given a weight q_n

$$X_{n+1|n}^{(k)} = h(X_{n|n}^{(k)} | q_n) \quad k \in \mathbb{N}$$

6: Go to step 2 and repeat with new inputs

The above steps can be described in Fig. 14 [19], where prediction is done through the dynamic model update in Step 2/3, calculation and normalization of weights based on the measurement model in Step 4, and resampling in Step 5, in this case done using the computed likelihood histogram based on the normalized weights.

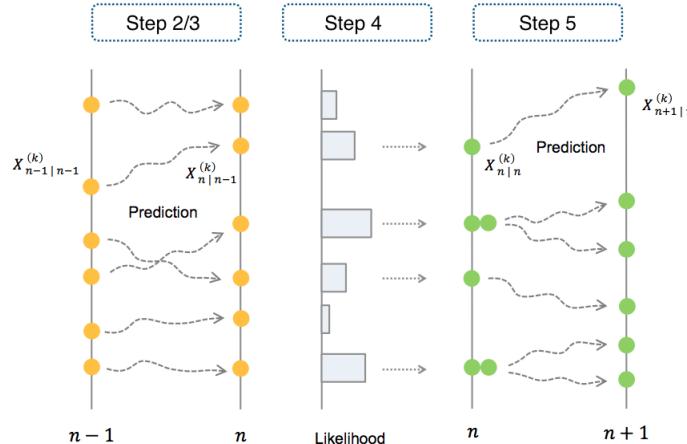


Figure 14: Particle Filter Stages

3 Calibration

3.1 Calibration Model

Now that we have an equation that can map 3D coordinates in the camera frame to camera pixels, we are able to combine both spaces as seen in Fig. 15 and Eq. 20. This assumes that we know cT_s which may be roughly estimated from physical measurements.

$$\begin{bmatrix} U_c \\ V_c \end{bmatrix} \leq \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} {}^cT_{s11} & {}^cT_{s12} & {}^cT_{s13} & {}^cT_{s14} \\ {}^cT_{s21} & {}^cT_{s22} & {}^cT_{s23} & {}^cT_{s24} \\ {}^cT_{s31} & {}^cT_{s32} & {}^cT_{s33} & {}^cT_{s34} \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \sqrt{R_s^2 - Y_s^2} \cdot \sin(\theta_s) \\ Y_s \\ \sqrt{R_s^2 - Y_s^2} \cdot \cos(\theta_s) \\ 1 \end{bmatrix} \quad (20)$$

The depth sensor is located about the vehicle center of mass (COM). Assuming the depth of the object of interest is measured, and the depth of the vehicle (D_D) is known, we can easily derive the depth of the object relative to the vehicle

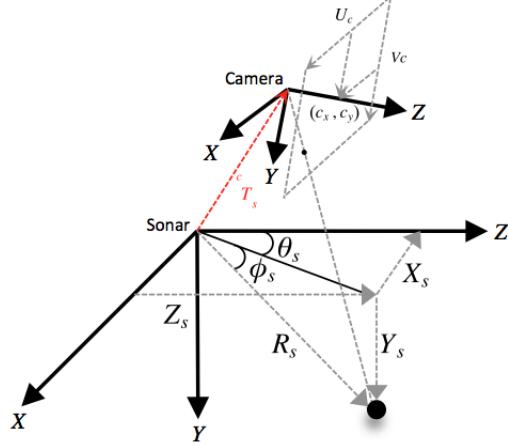


Figure 15: Sonar Camera Model

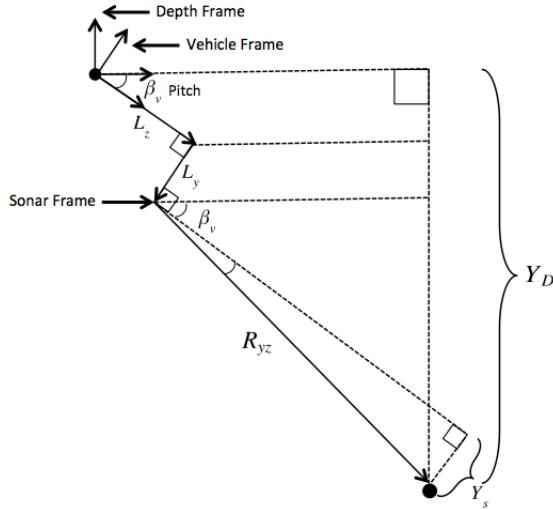


Figure 16: Sonar Vehicle Model

COM aligned to the water surface (Y_D). The depth sensor's output is assumed to be impervious to vehicular dynamics as it is placed at the vehicle COM, and its frame is assumed to be aligned to the water surface. Using geometrical methods, we derived an equation that approximately maps Y_D to Y_s and vice-versa based on Fig. 16 and Eq. 22/23 using R_s and β_v . L_z and L_y can also be estimated from physical measurements. (R_{yz}) is the flattened range of the Y_s and Z_s axis on the sonar frame and is approximately equal to ($R_s * \Theta_s$). Based on a workspace of a max R of 15m and the sonar parameters listed in Eq. 21, this approximation yields a maximum error of 1.35cm and is deemed acceptable. Also, this model assumes that the roll effect is minimal which was the case in our vehicle. The downside of this model is that we cannot estimate L_x , but we do not need it for our calibration step.

$$\begin{bmatrix} \max(R_s) = 15 \\ \max(\theta_s) = 45/2 \\ \max(\phi_s) = 15/2 \end{bmatrix} \quad R_{yz} = R_s * \cos(\Theta_s) \quad (21)$$

$$Y_D = L_y * \cos(\beta_v) + L_z * \sin(\beta_v) + R_{yz} * \sin\left(\beta_v + \arcsin\left(\frac{Y_s}{R_{yz}}\right)\right) \quad (22)$$

$$Y_s = -R_{yz} * \sin\left(\beta_v + \arcsin\left(\frac{(L_y * \cos(\beta_v) - Y_D + L_z * \sin(\beta_v))}{R_{yz}}\right)\right) \quad (23)$$

From these models, we are able to derive a closed form solution that allows one to compute Y_s and hence calculate Φ_s from $(\beta_v, R_s, \Theta_s, U_c$ and $V_c)$. It also allows us to compute U_c and V_c from Y_D, β_v, R_s and Θ_s if the depth of the object is known. Lastly, if the depth of the object is unknown, it allows us to compute a possible solution space since Y_s has a min/max elevation of 15° .

However, this assumes that L_z, L_y and cT_s are precise. This might not be the case from physical measurements alone. This is why calibration techniques are later used to estimate these parameters more precisely.

3.2 Data Collection

In order to collect data to fit into our model, we wish to extract $(R_s, \Theta_s, U_c, V_c, \beta_v, Y_D)$ over a large range. Hence, we have to come up with an algorithm to automatically detect and collect data. This means extracting data from both the camera and the sonar imagery without human intervention using computer vision techniques. This speeds up the data collection process, and results in more accurate data. The test object used here is a solid fluid-filled yellow sphere/buoy with a radius of 0.2m. Solid sphere's are normally used in sonar calibration due to their high levels of insonification and the fact that we can take it's centroid as a single point [9]. Also, the yellow sphere can be easily segmented in the camera image as it's hue is significantly different from the surrounding pool. The buoy's precise depth is measured by placing the depth sensor on it, allowing us to calculate Y_D . β_v is available from the vehicle's on-board IMU. The other information can be extracted from the images. Software used in this entire system consists of Numpy/Scipy for matrix computations [10], ROS (Robot Operating System) for communication [20], OpenCV for image processing functions [3] and MATLAB for algorithm development.

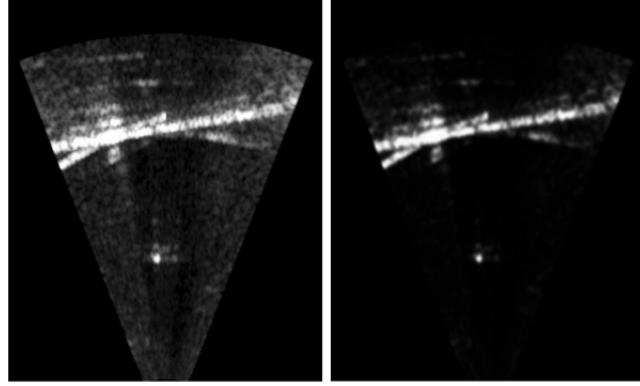


Figure 17: Before and after applying image processing techniques on sonar image

Figure 17 shows the image direct from the sensor and after various filters are applied. Several methods used here are referenced from Kim's paper on radar processing techniques [12]. The first stage was to convolve the image with a 2-D isotropic Gaussian low pass kernel with a σ of 5 pixels based on Eq. 24. N and M represent the width and height of the sonar image respectively. This helps remove some of the sparse electrical or acoustic noise in the image. Power law is then applied to the normalized image to enhance brighter regions based on Eq. 25. Finally, image averaging based on Eq. 26 is applied, and helps lower the intensity of sporadic noise that appears in and out of subsequent frames.

$$g(x, y) = e^{-\frac{(x-c_x)^2 + (y-c_y)^2}{(2\sigma)^2}} \quad (24)$$

$$c_x = 0.5 \cdot N \quad c_y = 0.5 \cdot M \quad \sigma = 5$$

$$I(x, y) = I(x, y) \otimes g(x, y)$$

$$I(x, y) = \left(\frac{I(x, y)}{255} \right)^2 \cdot 255 \quad (25)$$

$$I(x, y) = \frac{\sum_i^2 I(x, y)}{2} \quad (26)$$

After these image processing methods, a combination of thresholding, morphology and contour detection is used to extract objects from the image. We can easily extract and track the calibration buoy using these methods as they are distinct from other objects in the image. These pre-processing techniques will also be later applied before more complex filters such as particle filters are applied. Once the coordinates in the sonar image are extracted, they are passed to the Blueview API to extract R_s and Θ_s .

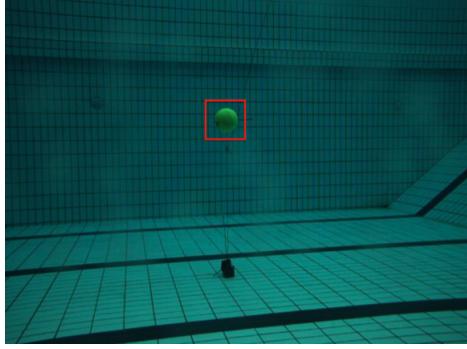


Figure 18: Calibration buoy seen in the camera frame

Camera points are extracted using contrast enhancement, HSV Thresholding to pick out yellow and morphology. This way, we can precisely extract U_c and V_c automatically. A data collection UI has also been written to extract all this information into a MATLAB friendly format.

3.3 Calibration process

A set of data points with $(R_s, \Theta_s, U_c, V_c), (\alpha_v, \beta_v, \gamma_v), (\dot{\alpha}_v, \dot{\beta}_v, \dot{\gamma}_v), (\dot{X}_v, \dot{Y}_v, \dot{Z}_v)$ and Y_D is collected over a space of $10 \times 2.5 \times 12$ metres. A total of 10,000 points was collected over this space and split into 5 datasets consisting of 1 training set and 4 test sets. Camera calibration is conducted to solve for $(f_x, f_y, c_x, c_y, K_1, K_2)$ on both land and water, and tested to see that (f_x, f_y) differ by 1.33. L_z , L_y and cT_s are given initial values which are guessed from the CAD model of the system. The results of this calibration and initial estimation is given in Eq. 27/28. We assume identity for the rotation as it is hard to measure initially.

$$F = \begin{bmatrix} 735.4809 & 0 & 388.9467 & 0 \\ 0 & 733.6047 & 292.0895 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} K_1 = 0.0369 \\ K_2 = -0.3870 \end{bmatrix} \quad (27)$$

$${}^cT_s = \begin{bmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & 0.3 \\ 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L_y = 0.2 \\ L_z = 0.4 \end{bmatrix} \quad (28)$$

The Levenberg Marquardt algorithm [10] for non-linear optimization available in MATLAB is used to minimize Eq. 29 where (U_G, V_G) is the ground truth of the yellow buoy, while (U_M, V_M) is the camera position calculated from $(R_s, \Theta_s, Y_D, \beta_v)$ based on Eq. 20/23, and initial parameters from Eq. 27/28.

$$\min((U_G - U_M)^2 + (V_G - V_M)^2) \quad (29)$$

Testing parameters from Eq. 28 against the training set gave results as seen in Fig. 21. The figure on the left shows pixel error from ground truth projection of the yellow buoy vs the combined $(R_s, \Theta_s, Y_D, \beta_v)$ projection into the camera. The figure on the right, shows the error in the ground truth 3D coordinates in the sonar frame derived from $(R_s, \Theta_s, Y_D, \beta_v)$, vs that derived from $(R_s, \Theta_s, U_c, V_c)$. One can observe significant error in Y direction. After calibration is performed, the

newly estimated L_z , L_y and cT_s seen in Eq. 30 is tested against the training set. Fig. 22 shows significant reduction in error against the ground truth. These results can be better observed in Fig. 23, which shows the 3D error (Euclidean Distance between ground truth and sonar camera estimate 3D coordinates) in metres before and after calibration. Fig. 24 shows pixel error (Euclidean Distance between camera pixel coordinates and sonar depth projected pixel coordinates) before and after calibration.

Lastly, the newly estimated L_z , L_y and cT_s is also tested against 4 other test sets. Fig. 26 shows the 3D error in metres for all 4 test sets. Table. 2 shows the mean and variance of the errors before and after calibration of both training and test sets in 3D and Pixel errors. We can observe a 4 time reduction in mean pixel error, and a 6 time reduction in mean 3D error.

Data	Pixel Error Mean / Var	3D Error Mean / Var
Before Calibration Training Set	32.8057 / 43.1141	0.1478 / 0.0068
After Calibration Training Set	8.2440 / 22.7498	0.0258 / 0.0007
After Calibration Test Set 2	8.1873 / 22.3888	0.0257 / 0.0007
After Calibration Test Set 3	8.2730 / 35.4033	0.0259 / 0.0008
After Calibration Test Set 4	8.1897 / 22.2684	0.0255 / 0.0007
After Calibration Test Set 5	8.1486 / 21.5721	0.0254 / 0.0007

Table 2: Pixel and 3D Euclidean errors before and after calibration

$${}^cT_s = \begin{bmatrix} 0.9993 & 0.0311 & 0.0219 & 0.1401 \\ -0.0303 & 0.9989 & -0.0363 & 0.2671 \\ -0.0230 & 0.0356 & 0.9991 & 0.1407 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L_y = 0.1631 \\ L_z = 0.2473 \end{bmatrix} \quad (30)$$

3.4 Mapping process

Now that we have the calibration parameters, we can use (R_s, Θ_s) from the sonar image and (U_c, V_c) from the camera image to solve for Y_s using Eq. 20. From there, we may compute (X_s, Y_s, Z_s) and we can use (L_z, L_y, β_v) to compute (X_v, Y_v, Z_v) and Y_D that may be passed to the control system to actuate the vehicle.

However, we need to relate a blob in the sonar image with a region of interest (ROI) in the camera image in order to compute this. This is done by projecting the corner points and centroid of the sonar blob into the camera as seen in Fig. 19, by extracting (R_s, Θ_s) for these 3 points (A, B, C) , getting the upper and lower Y_s bounds based on the sonar's min/max elevation, then projecting this as a ambiguity lines into the camera. Then, the ROI from the camera has lines (A', B', C') extracted. Finally, we attempt to iteratively minimize Eq. 31, and selecting the best match. Once the match is known, we can compute the object's coordinates.

$$\begin{aligned} f(A, B) &= \text{Distance between lines} \\ \min \left(f(A, A')^2 + f(B, B')^2 + f(C, C')^2 \right) \end{aligned} \quad (31)$$

This method has several drawbacks since matching both sonar and camera objects requires the ROI in the camera to be extracted accurately. While this method worked in pool conditions, it tended to fail in sea/lake conditions, due to particulate matter in water over large distances, or through temporal irradiance due to sunlight filtering through the water. Also, the vehicle's pitching motion tended to cause the intensity of the sonar object to drop as the object would go out of its frustum, causing brief losses in detection. A framework was needed allow for consistent sensing of the object in 3D space.

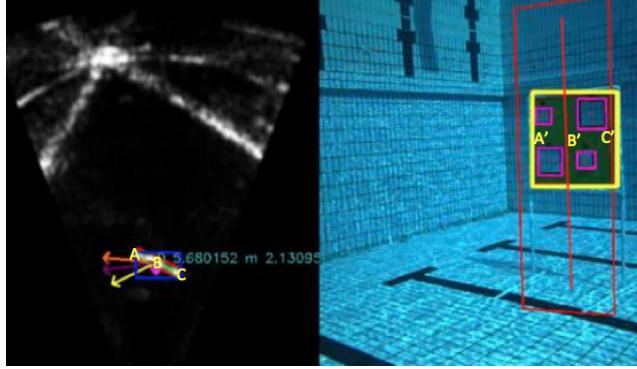


Figure 19: Matching objects in the sonar and camera by projecting ambiguity lines into the camera

4 Particle Filter

This is why the need of a statistical filter to track and localize these objects, constrained by the calibrated transformation between both the sonar and camera based on Eq. 20 is proposed and used. Since the objects being localized are stationary, it is also possible to involve the vehicle's odometry data in order to predict the position of these objects if loss in data occurred in the sonar or camera.

4.1 Dynamic Model

The state matrix and dynamic model is given by Eq. 32 which represents the object position and velocity in 3D space in the sonar frame. N number of particles representing possible solutions to where the object is are initially distributed over the workspace as per step 1. Then each particles position is updated based on a simple velocity displacement update matrix with some process noise w_k added.

$$X_{n|n-1}^{(k)} = \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ \dot{X}_s \\ \dot{Y}_s \\ \dot{Z}_s \end{bmatrix} \quad X_{n|n-1}^{(k)} = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * X_{n-1|n-1}^{(k)} + w_{n-1} \quad (32)$$

Since the object we are localizing is stationary w.r.t the world frame, we can incorporate the vehicle odometry into the object motion. This is so that if we are unable to measure (X_s, Y_s, Z_s) in the measurement step, we can predict the object's next position based on the previous position and current velocity. To do this, we first transform $(X_s, Y_s, Z_s)_k$ at that time instant into the vehicle frame with the known (L_y, L_z) , and convert it to polar coordinates, based on Eq. 33. We assume the sonar and vehicle frames have an identity rotational component here, and that (L_x) which cannot be estimated is minimal.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & L_y \\ 0 & 0 & 1 & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix}_k = \begin{bmatrix} X_v \\ Y_v \\ Z_v \\ 1 \end{bmatrix}_k = \begin{bmatrix} R_v \cdot \cos(\phi_v) \cdot \sin(\theta_v) \\ R_v \cdot \sin(\phi_v) \\ R_v \cdot \cos(\phi_v) \cdot \cos(\theta_v) \\ 1 \end{bmatrix} \quad (33)$$

We then differentiate the polar representation of the object in the vehicle frame in Eq. 33 to yield the Jacobian matrix in

Eq. 34.

$$\begin{bmatrix} \dot{X}_v \\ \dot{Y}_v \\ \dot{Z}_v \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} R_v \cdot \cos(\phi_v) \cdot \sin(\theta_v) \\ R_v \cdot \sin(\phi_v) \\ R_v \cdot \cos(\phi_v) \cdot \cos(\theta_v) \end{bmatrix} = \begin{bmatrix} (\dot{R}_v \cdot \cos(\phi_v) \cdot \sin(\theta_v)) - (R_v \cdot \dot{\phi}_v \cdot \sin(\phi_v) \cdot \sin(\theta_v)) + (R_v \cdot \dot{\theta}_v \cdot \cos(\phi_v) \cdot \cos(\theta_v)) \\ (\dot{R}_s \cdot \sin(\phi_v)) + (R_v \cdot \dot{\phi}_v \cdot \cos(\phi_v)) \\ (R_v \cdot \cos(\phi_v) \cdot \cos(\theta_v)) - (R_v \cdot \dot{\phi}_v \cdot \sin(\phi_v) \cdot \cos(\theta_v)) - (R_v \cdot \dot{\theta}_v \cdot \cos(\phi_v) \cdot \sin(\theta_v)) \end{bmatrix} \quad (34)$$

Next, some components in the Jacobian matrix are replaced with their equivalent readings of other sensors as seen in Eq. 35. The \dot{R}_v components are equal to the negative vehicular translational velocity components, and the $\dot{\theta}_v$ and $\dot{\phi}_v$ are equal to the negative angular velocities of pitch and yaw. This yields Eq. 36, which shows the replaced values.

$$\begin{bmatrix} (\dot{R}_v \cdot \cos(\phi_v) \cdot \sin(\theta_v)) \\ (\dot{R}_v \cdot \sin(\phi_v)) \\ (\dot{R}_v \cdot \cos(\phi_v) \cdot \cos(\theta_v)) \end{bmatrix} = \begin{bmatrix} -\dot{X}_v \\ -\dot{Y}_v \\ -\dot{Z}_v \end{bmatrix} \begin{bmatrix} \dot{\theta}_v \\ \dot{\phi}_v \end{bmatrix} = \begin{bmatrix} -\dot{\gamma}_v \\ -\dot{\beta}_v \end{bmatrix} \quad (35)$$

$$\begin{bmatrix} \dot{X}_v \\ \dot{Y}_v \\ \dot{Z}_v \end{bmatrix} = \begin{bmatrix} (-\dot{X}_v) - (R_v \cdot -\dot{\beta}_v \cdot \sin(\phi_v) \cdot \sin(\theta_v)) + (R_v \cdot -\dot{\gamma}_v \cdot \cos(\phi_v) \cdot \cos(\theta_v)) \\ (-\dot{Y}_v) + (R_v \cdot -\dot{\beta}_v \cdot \cos(\phi_v)) \\ (-\dot{Z}_v) - (R_v \cdot -\dot{\beta}_v \cdot \sin(\phi_v) \cdot \cos(\theta_v)) - (R_v \cdot -\dot{\gamma}_v \cdot \cos(\phi_v) \cdot \sin(\theta_v)) \end{bmatrix} \quad (36)$$

We then perform the update/prediction step based on the velocity model in Eq. 32, then transform the newly estimated $(X_v, Y_v, Z_v)_{k+1}$ back into the sonar frame $(X_s, Y_s, Z_s)_{k+1}$ as seen in Eq. 37.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -L_y \\ 0 & 0 & 1 & -L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X_v \\ Y_v \\ Z_v \\ 1 \end{bmatrix}_{k+1} = \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix}_{k+1} \quad (37)$$

4.2 Measurement Model

Now that we have predicted the new position of the object (X_s, Y_s, Z_s) using the dynamic model above, we wish to correct this with measurements from the sonar and camera. In order to do this, we project this data into their respective image pixels since their calibration data is known. Particles in the sonar frame are projected into the sonar image by converting (X_s, Y_s, Z_s) into (R_s, Θ_s, Φ_s) as seen in Eq. 38. R_s and Θ_s may then be matched into the sonar image pixel coordinates provided by the Blueview API. (X_s, Y_s, Z_s) are then transformed into the camera frame and projected into the camera based on the calibration data derived as seen in Eq. 39. From here, we can extract (U_s, V_s) and (U_c, V_c) pixel coordinates in both sensors from the particle state.

$$\begin{bmatrix} \sqrt{X_s^2 + Y_s^2 + Z_s^2} \\ \arctan\left(\frac{X_s}{Y_s}\right) \\ \arctan\left(\frac{Y_s}{\sqrt{X_s^2 + Y_s^2 + Z_s^2}}\right) \end{bmatrix} = \begin{bmatrix} R_s \\ \Theta_s \\ \Phi_s \end{bmatrix} \Rightarrow \begin{bmatrix} U_s \\ V_s \end{bmatrix} \quad (38)$$

$$\begin{bmatrix} 735.4809 & 0 & 388.9467 & 0 \\ 0 & 733.6047 & 292.0895 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0.9993 & 0.0311 & 0.0219 & 0.1401 \\ -0.0303 & 0.9989 & -0.0363 & 0.2671 \\ -0.0230 & 0.0356 & 0.9991 & 0.1407 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} U_c \\ V_c \end{bmatrix} \quad (39)$$

Computing the weight of the particles from the sonar image is done by taking (U_s, V_s) and testing that pixel against several features, such as brightness/intensity of that spot and major/minor axis of the shape of the object that pixel is in.

This shape information such as the major/minor axis length is extracted by performing contour analysis, morphology and oriented bounding box fitting via Principle Component Analysis (PCA) [13].

This is done by calculating the log likelihood of that particle/state where we assume the above features that are measured at a particular state follow a gaussian distribution given a user specified major/minor axis length. This is specified by Eq. 40/41. Here, d represents the difference between a user specified measurement and actual measurement and σ represents the measurement standard deviation.

$$P(Y_n|X_{n|n-1}^{(k)}) = \frac{1}{\sqrt{2\pi}\sigma^2} \cdot \exp\left(-\frac{d^2}{2\sigma^2}\right) \quad (40)$$

$$\log(P(Y_n|X_{n|n-1}^{(k)})) = -\log(\sigma\sqrt{2\pi}) - \frac{0.5}{\sigma^2} \cdot d^2 \quad (41)$$

The final sonar weight of a particle is given by Eq. 42 where Fig. 24 describes the features used.

$$\begin{aligned} q_n^{sonar} = & -\log(\sigma_{Intensity}\sqrt{2\pi}) - \frac{0.5}{\sigma_{Intensity}^2} \cdot ((U_s, V_s)_{Intensity} - 255)^2 + \\ & -\log(\sigma_{MajorAxis}\sqrt{2\pi}) - \frac{0.5}{\sigma_{MajorAxis}^2} \cdot ((U_s, V_s)_{MajorAxis} - User_{MajorAxis})^2 + \\ & -\log(\sigma_{MinorAxis}\sqrt{2\pi}) - \frac{0.5}{\sigma_{MinorAxis}^2} \cdot ((U_s, V_s)_{MinorAxis} - User_{MinorAxis})^2 \end{aligned} \quad (42)$$

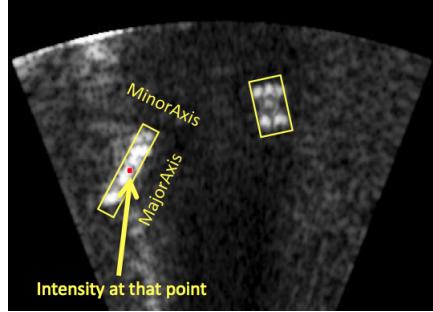


Figure 20: Sonar Features used

Computing the weight of the particles from the camera image is done by taking (U_c, V_c) and testing that pixel against several features, such as hue and shape ratio. The final camera weight of a particle is given by Eq. 43.

$$\begin{aligned} q_n^{camera} = & -\log(\sigma_{Hue}\sqrt{2\pi}) - \frac{0.5}{\sigma_{Hue}^2} \cdot ((U_c, V_c)_{Hue} - User_{Hue})^2 + \\ & -\log(\sigma_{Ratio}\sqrt{2\pi}) - \frac{0.5}{\sigma_{Ratio}^2} \cdot ((U_c, V_c)_{Ratio} - User_{Ratio})^2 \end{aligned} \quad (43)$$

There is also the possibility of computing the Pixel Velocity using the Lucas-Kanade feature tracking method, and giving higher weights to the projected particles that match this velocity, but using the above features were found to be sufficient. We finally add these weights up, perhaps giving more weightage to either sensor in the fusion process if the user requires it, and normalize it as per step 4. Resampling with replacement is then applied as per step 5 and the process is repeated. Finally, in order to extract results from the state matrix, we use GMM (Gaussian Mixture Model) clustering to extract the mean and

covariance of the particle clusters.

$$P(Y_n|X_{n|n-1}^{(k)}) = q_n^{\text{sonar}} + q_n^{\text{camera}} \quad (44)$$

$$q_n = \frac{P(Y_n|X_{n|n-1}^{(k)})}{\sum_k P(Y_n|X_{n|n-1}^{(k)})} \quad k \in \mathbb{N} \quad (45)$$

4.3 Particle Filter Results

Fig. 27 to Fig. 29 shows the particle filter algorithm in action tested in pool conditions. One can observe that initially, there is ambiguity in Φ_s as the object wasn't visible in the camera. As the vehicle moves closer, the ambiguity is corrected. Finally, as the object moves out of the sonar image, we can see the ambiguity in R_s . Also, in instances where there is loss of information from either sensor, we are still able to predict the next position of the object using the vehicle odometry data.

Fig. 30 to Fig. 32 show the particle filter algorithm in action tested in murky water conditions with poor visibility in the camera imagery, and a low SNR in the sonar imagery due to the object being almost out of the sonar frustum as seen in Fig. 31. Despite this, the particles still converge on the correct solution over time. It is also impervious to noise from surrounding wall as seen in Fig. 32.

We can see the mapping vs particle filter localization results on Fig. 34 for both cases. The graph on the left shows the (X_s, Y_s, Z_s) axis measurements gotten from both the mapping algorithm described in section 3.4 and the particle filter algorithm over time. The figure on the right shows the same data but represented in 3D space. The mapping algorithm loses a lot of data points as it requires testing the ROI extracted from the camera against the projected ROI of the sonar object. This method still worked well in the first case in pool conditions, but failed miserably in murky water conditions, with intermittent data loss and noisy data points. Using the particle filter method however, we can observe much better results that fill in a lot of the gaps of the first method and are much smoother.

5 Conclusion

This paper proposed a novel object localization algorithm in the 3D space in front of the AUV using particle filters, where the calibration parameters between the sonar and camera are used to enforce a model constraint. The algorithm was validated through implementation as a real time system on the BumbleBee Autonomous Underwater Vehicle (BBAUV).

The resulting algorithm was found to meet the specifications and objectives stated in the beginning of the project. The localization of objects in 3D space using the sonar and camera outperforms methods using only 1 sensor alone.

Future work can be done to possibly better estimate the complete transformation between all frames in the vehicle, include L_x , by incorporating the vehicular odometry data in the calibration model. Extracting precise ground truth is also a challenge, as GPS is unable to work underwater and odometry data from the DVL may not be precise enough. Rather than relying on data from the camera and depth sensor for ground truth, better methods such as long baseline (LBL) acoustic positioning systems may be considered. Lastly, working from a world frame instead of vehicle/object frame may be considered in the future, so that it may be easily extended to SLAM frameworks.

6 Figures

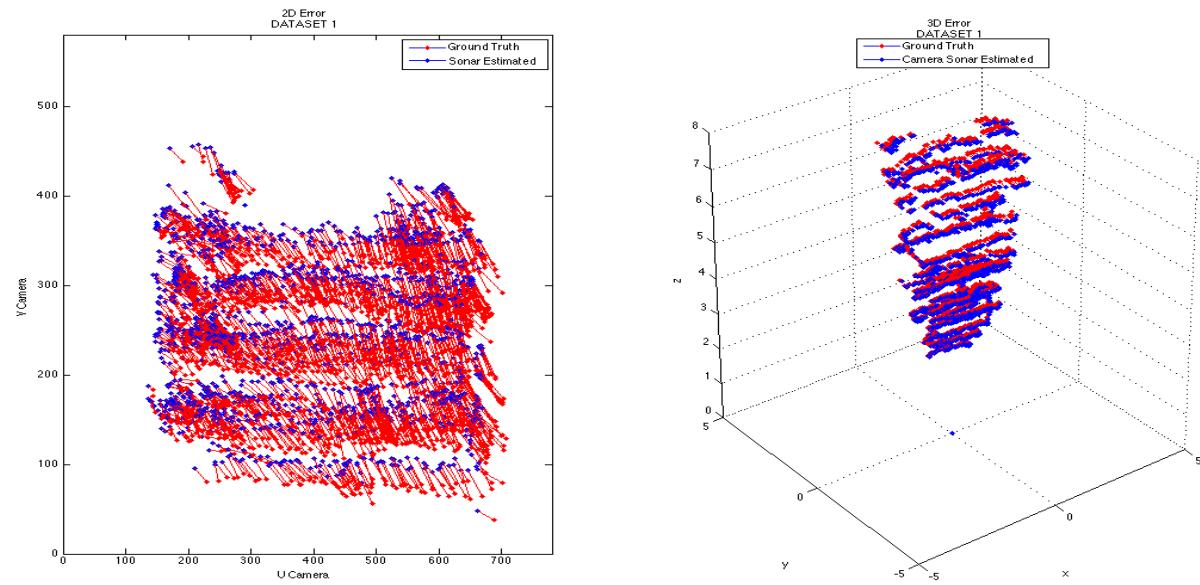


Figure 21: Large errors in pixel space (left) and 3D space (right) before calibration

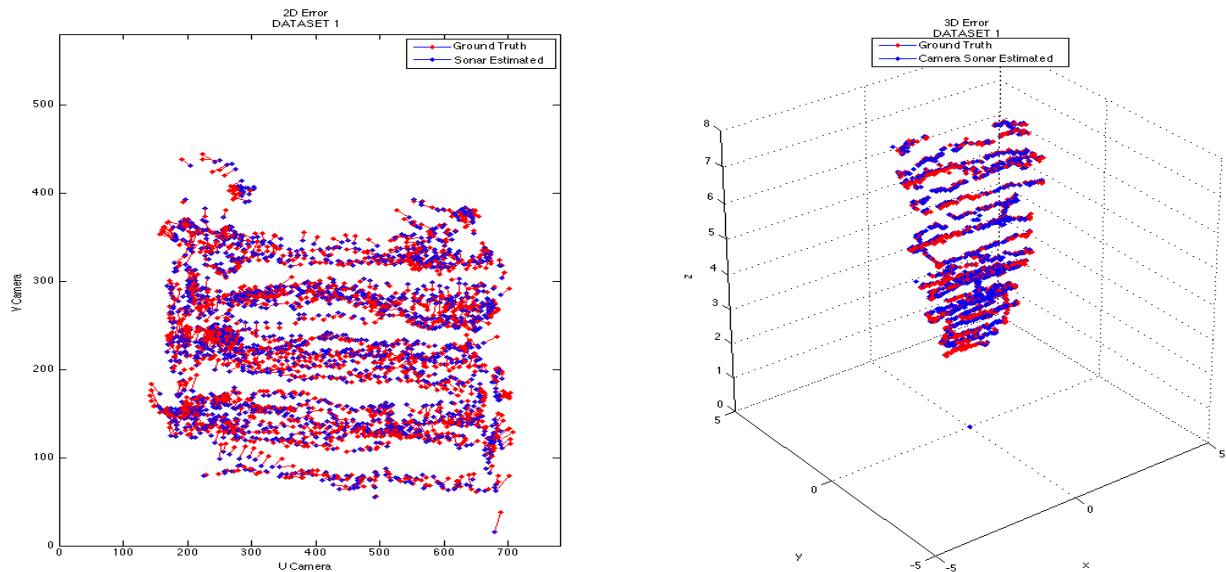
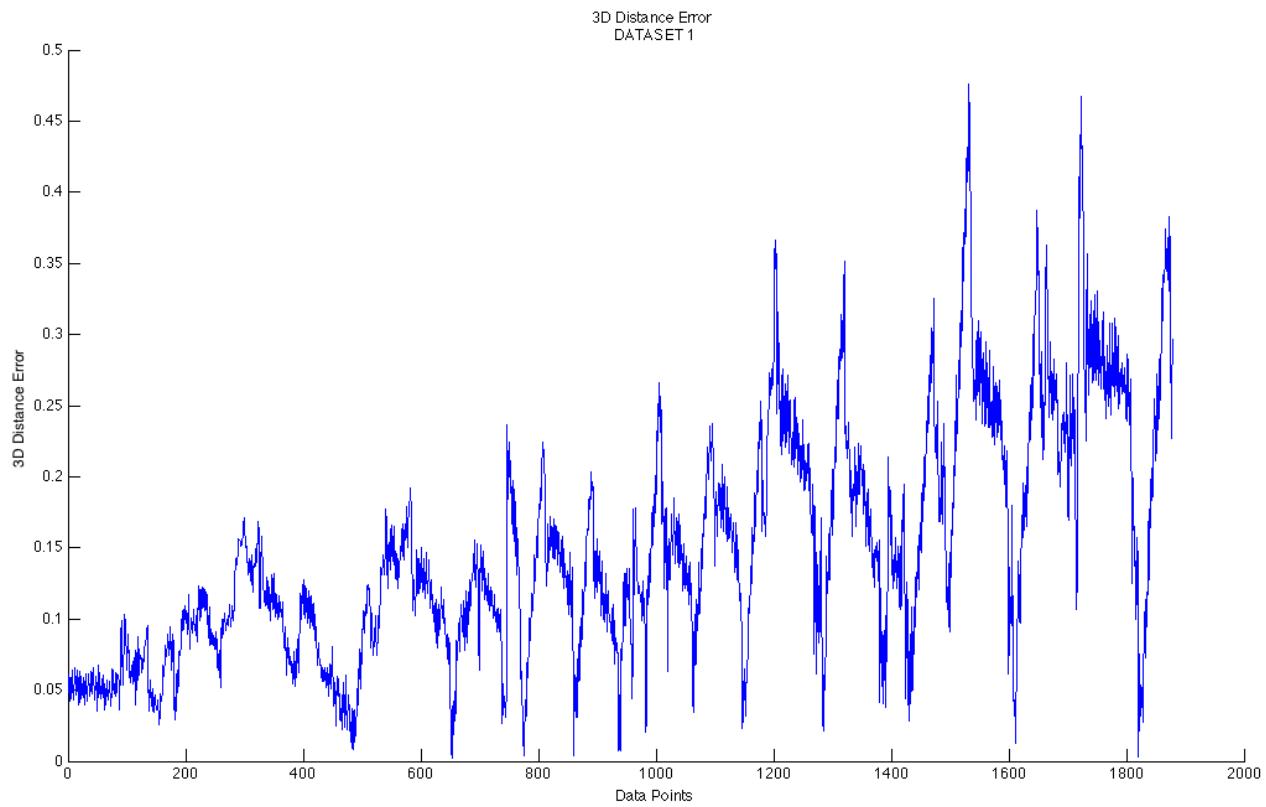
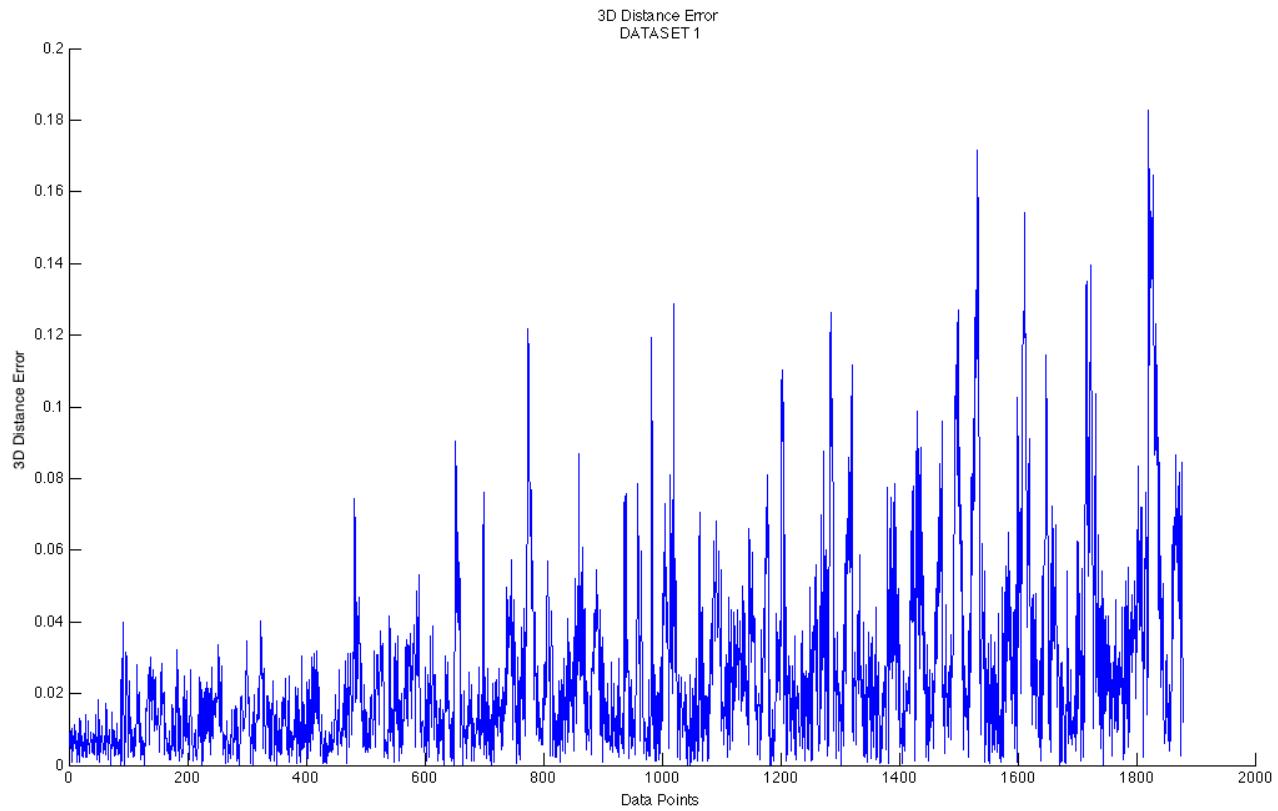


Figure 22: Reduced errors in pixel space (left) and 3D space (right) after calibration



(a) Mean 3D Euclidean error of approx 15cm on training set before calibration



(b) Mean 3D Euclidean error of approx 2cm on training set after calibration

Figure 23: 3D Euclidean error between resolved 3D coordinates from sonar and camera data vs ground truth in metres

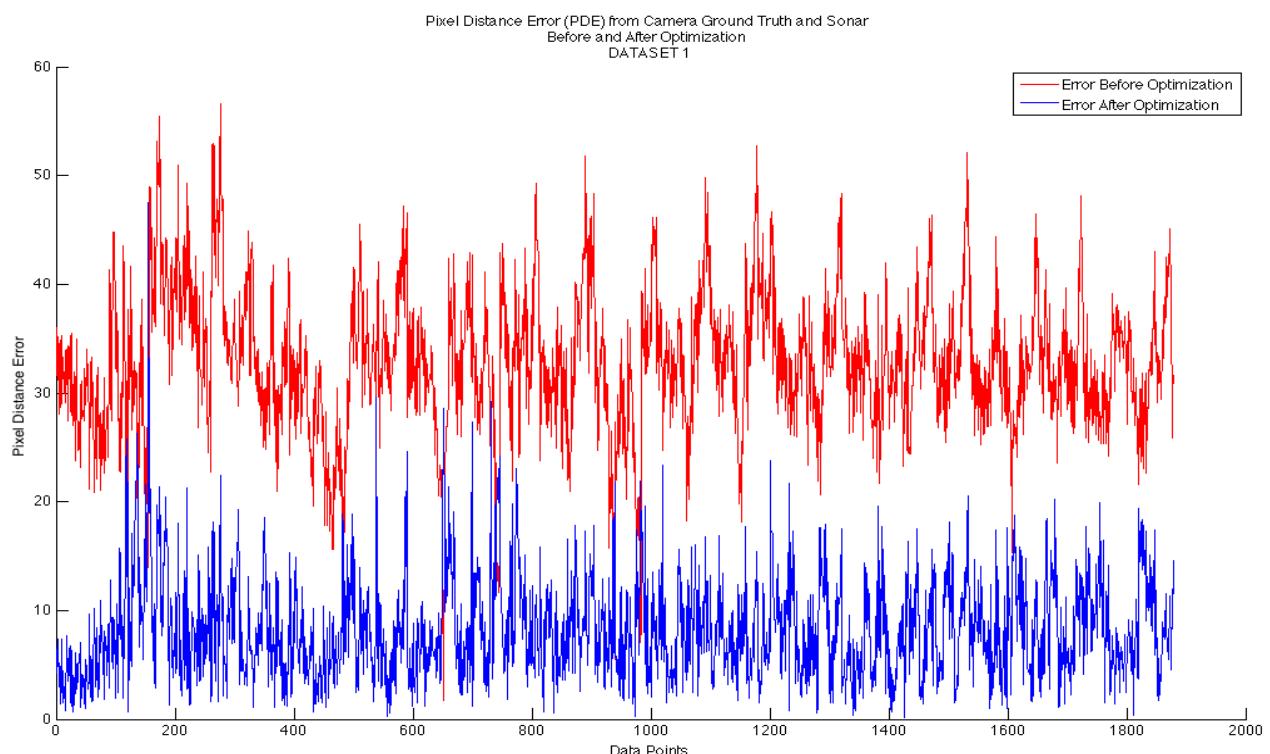
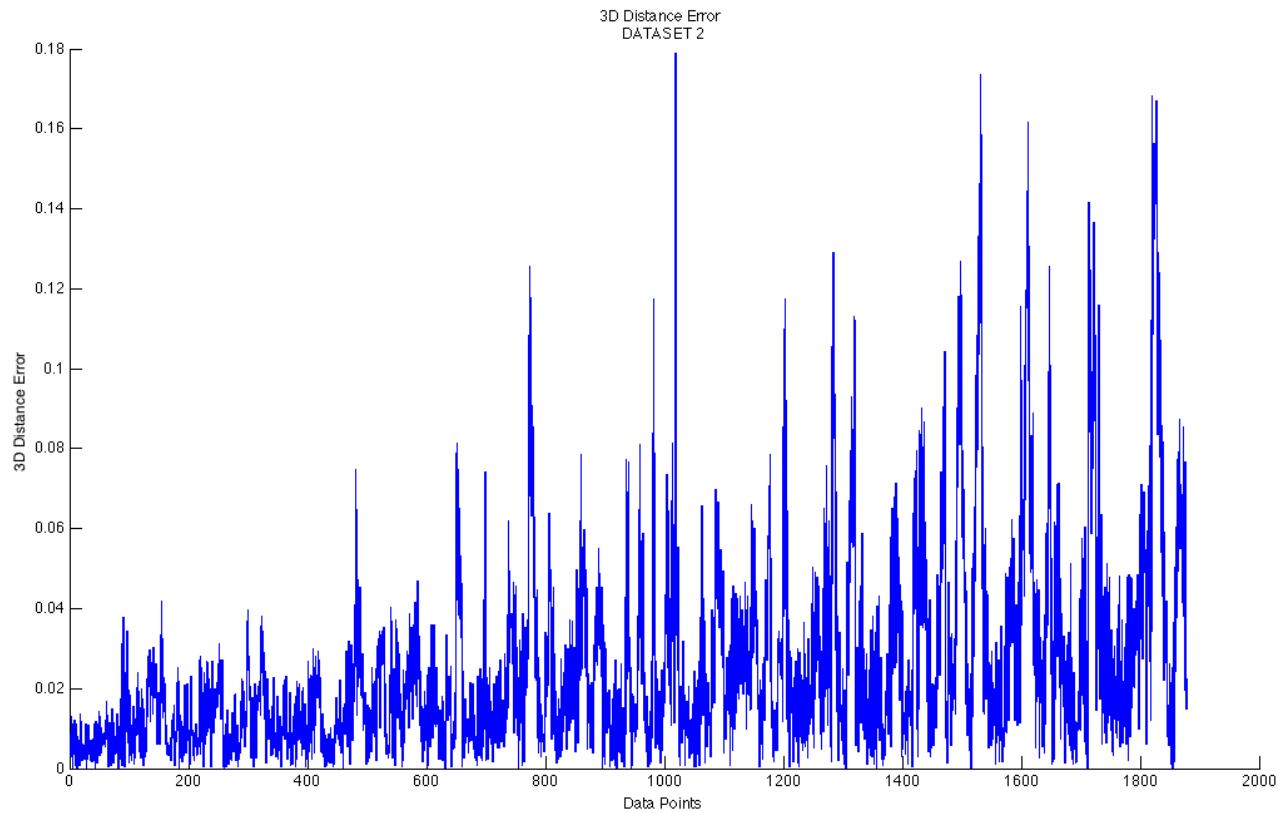
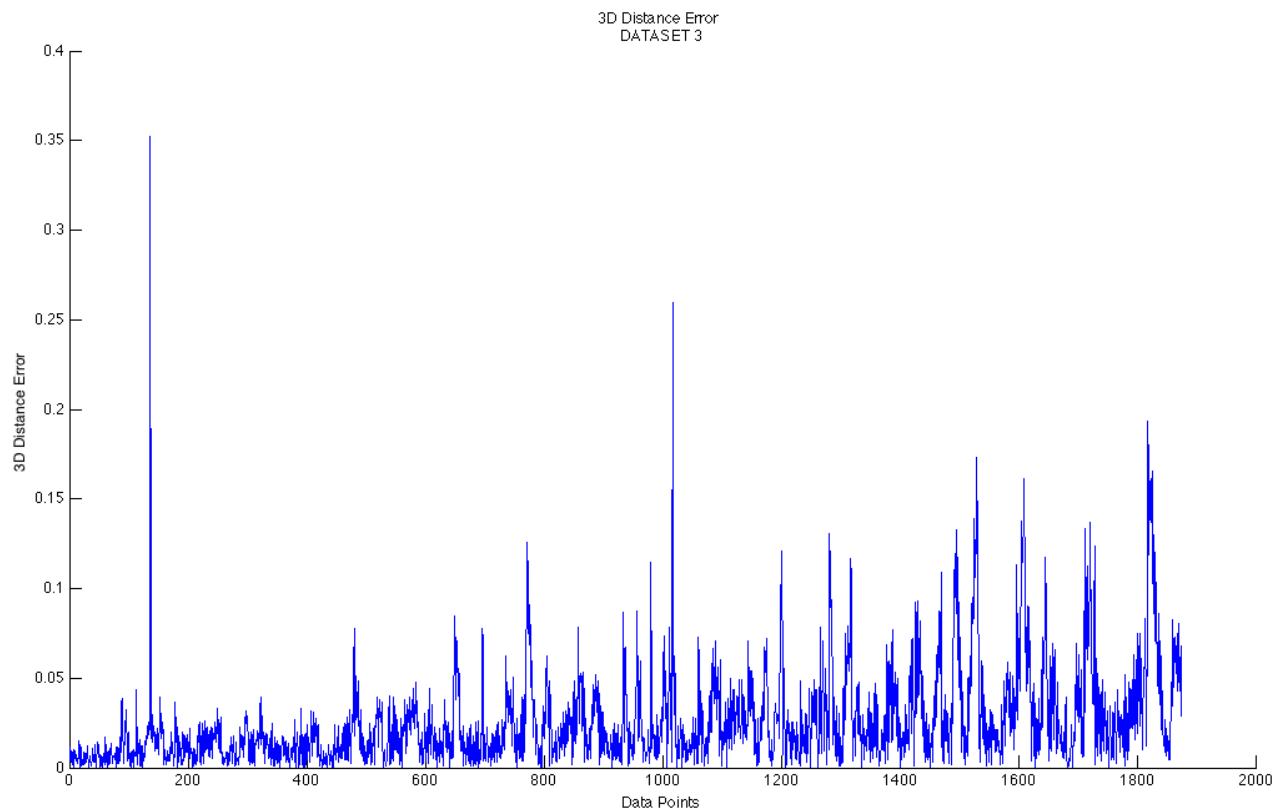


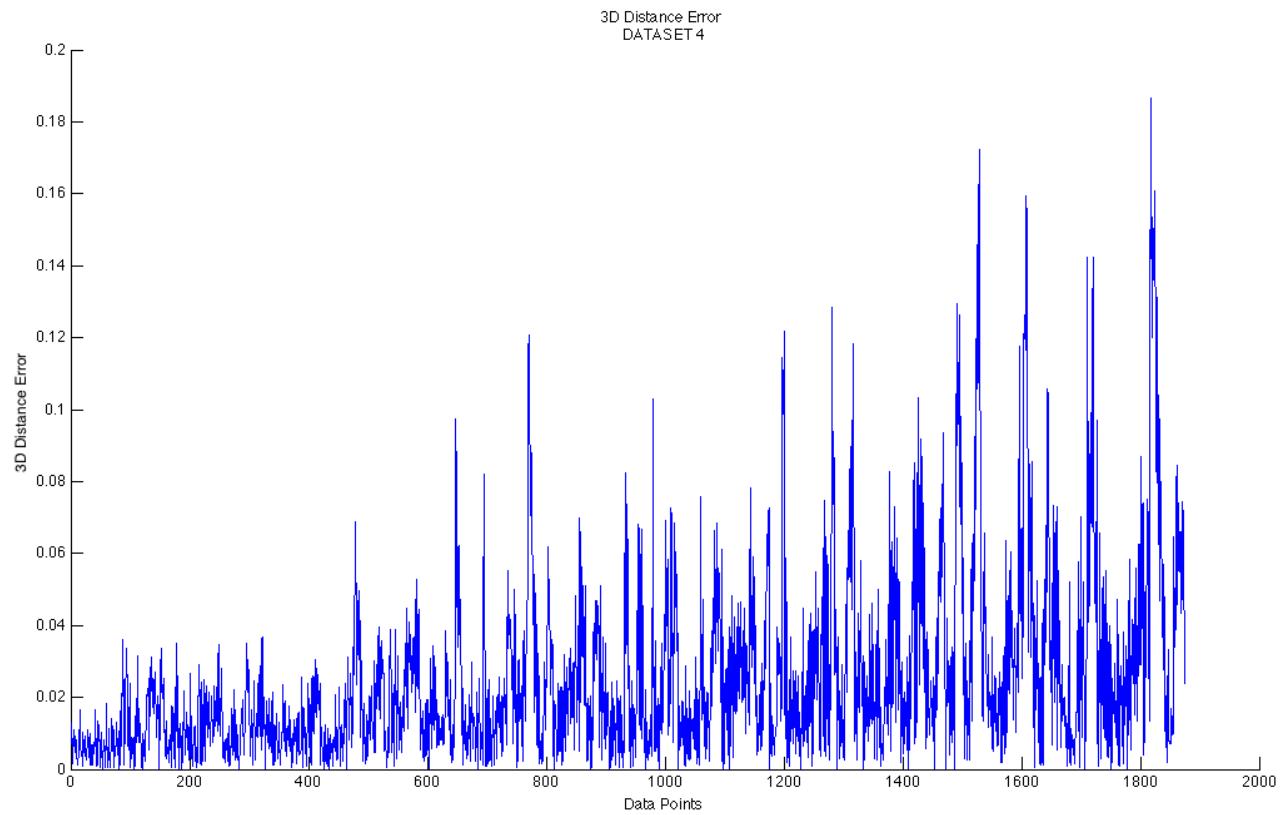
Figure 24: 2D Pixel Euclidean error between projected pixels and ground truth before and after calibration



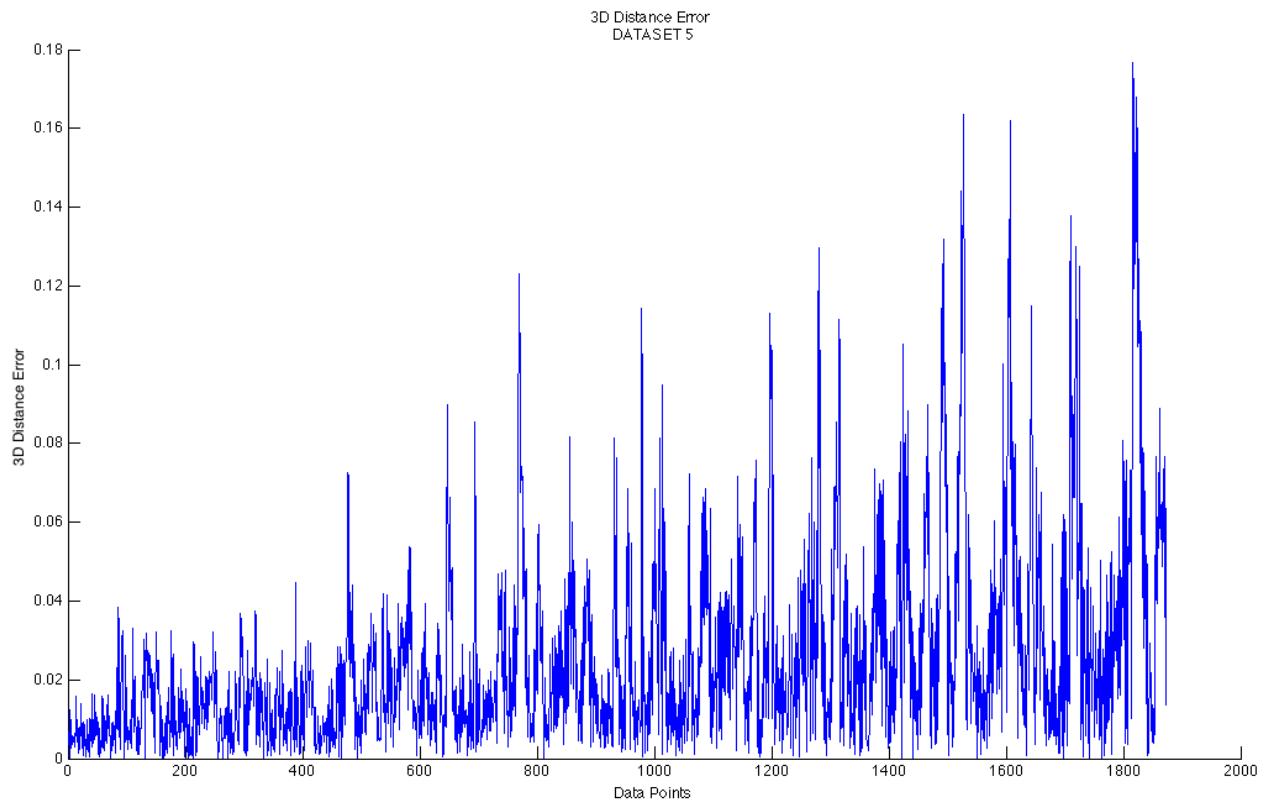
(a) 3D Euclidean error Test Set 2



(b) 3D Euclidean error Test Set 3



(a) 3D Euclidean errorTest Set 4



(b) 3D Euclidean error Test Set 5

Figure 26: Testing calibrated parameters derived from training set against 4 test sets

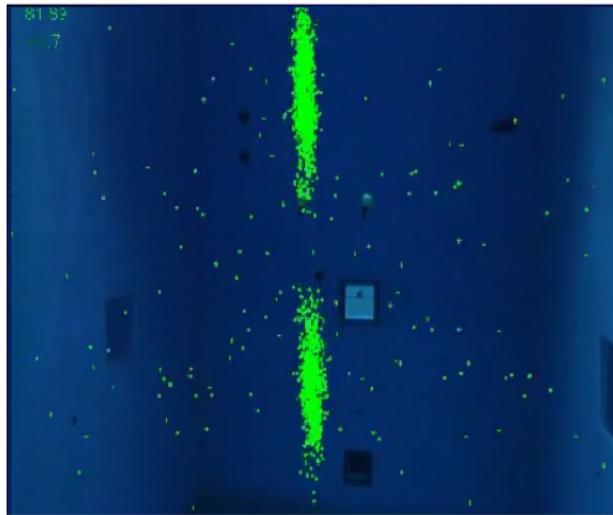


Figure 27: Object not visible in camera, hence bigger uncertainty in elevation as seen in the camera image

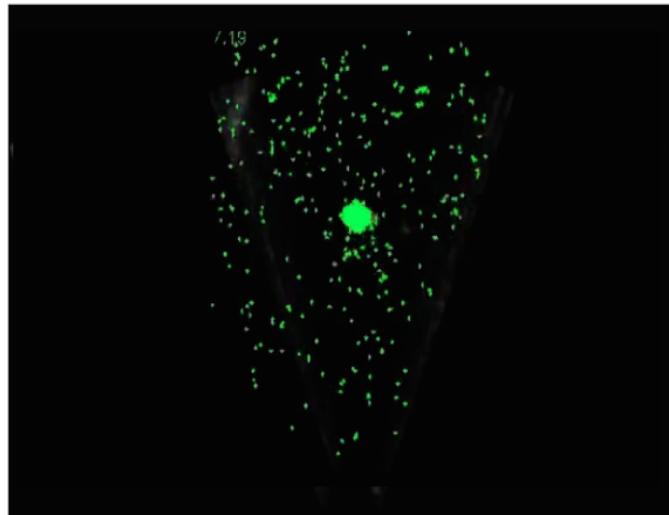
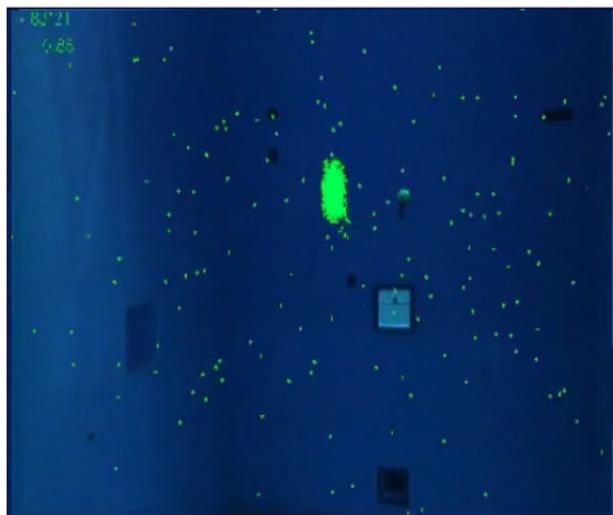


Figure 28: Elevation ambiguity corrected via camera. Also loss of sonar information at that instant, but position still maintained through camera and odometry fusion but with increased covariance

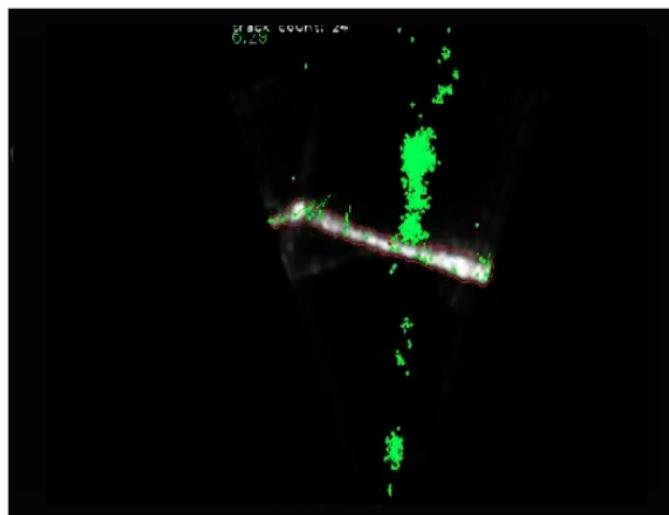
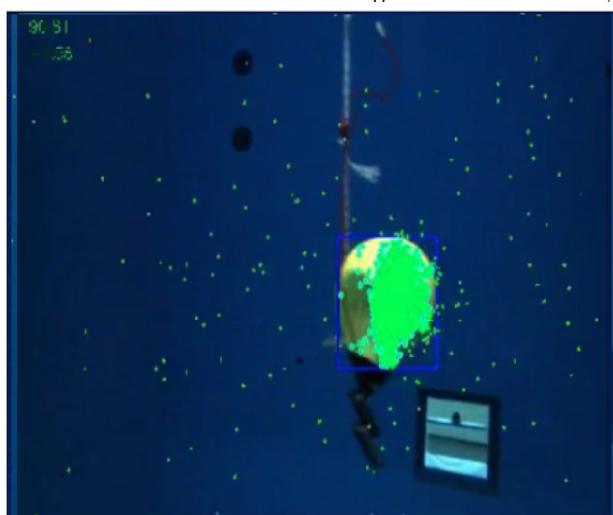


Figure 29: Object not visible in sonar, hence ambiguity in range

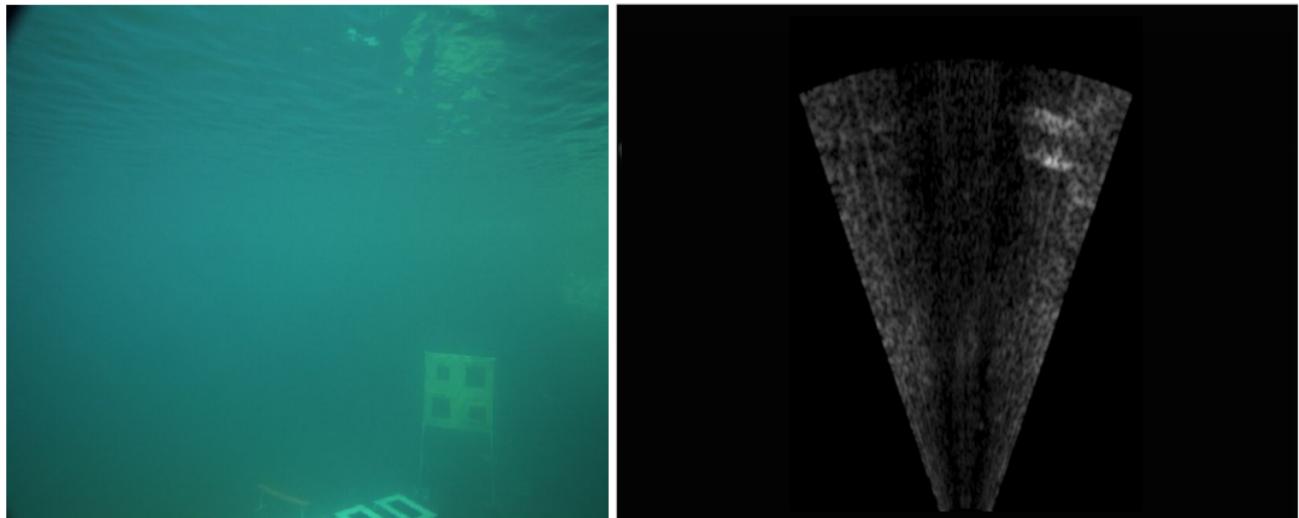


Figure 30: Objective is to localize the task board located nearly 15m away in murky water and is barely visible in either sensor

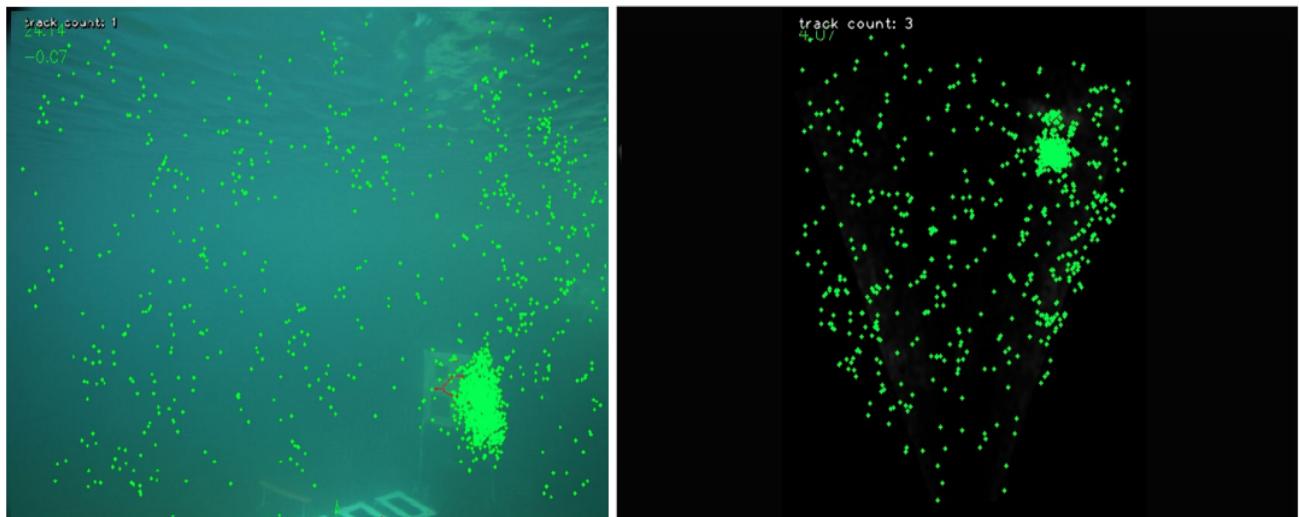


Figure 31: Particle Filter still localizes the object in 3D space

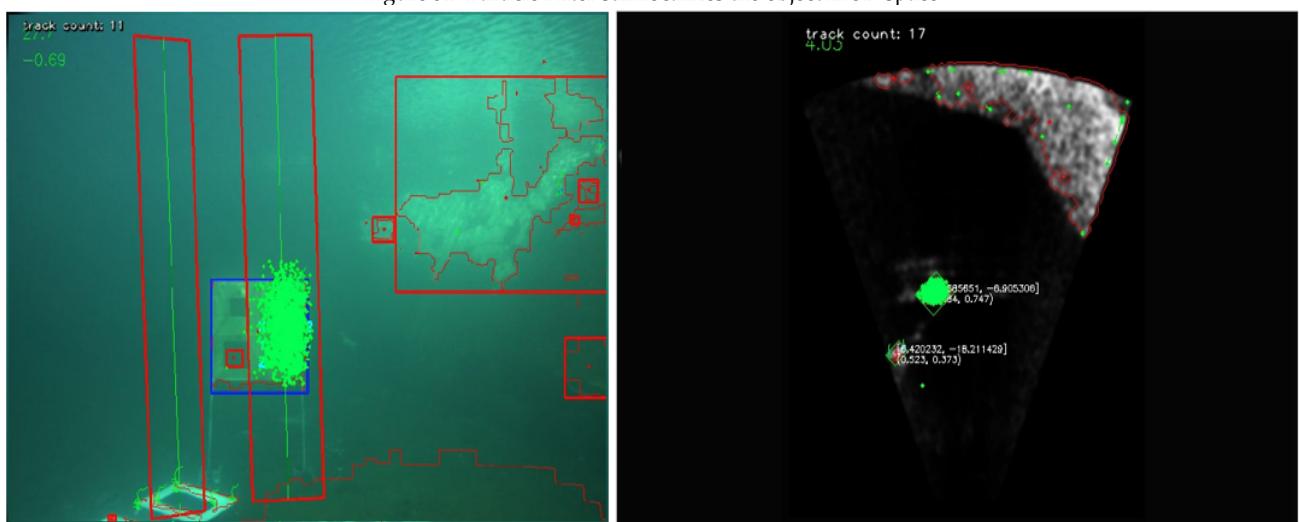
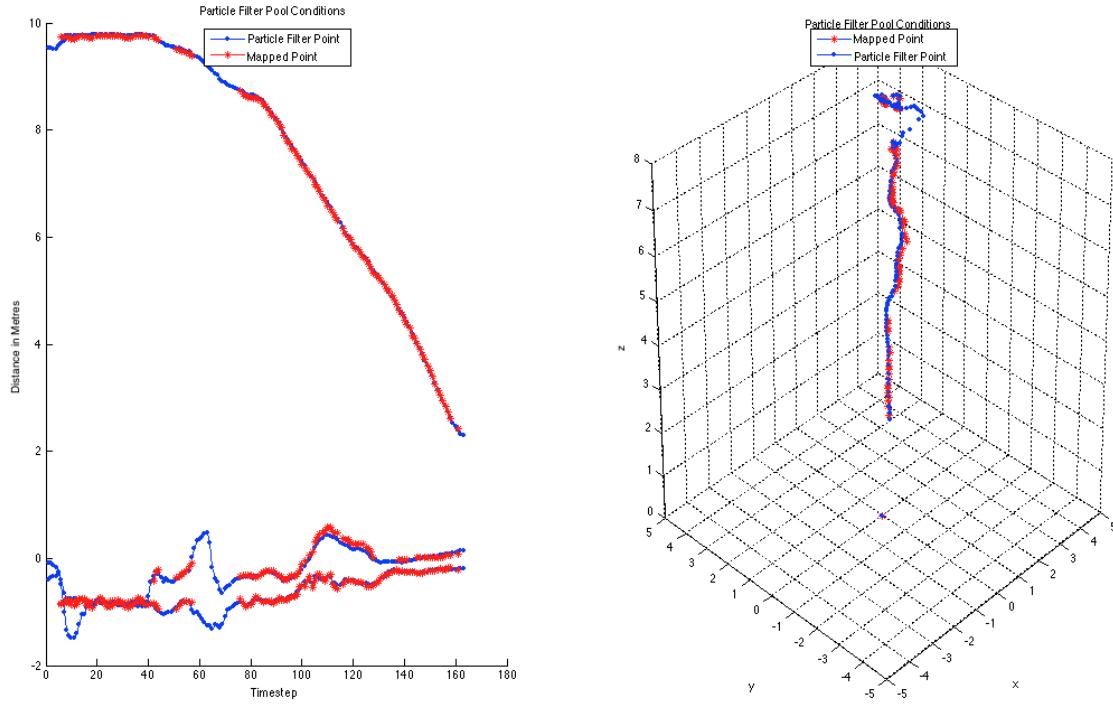
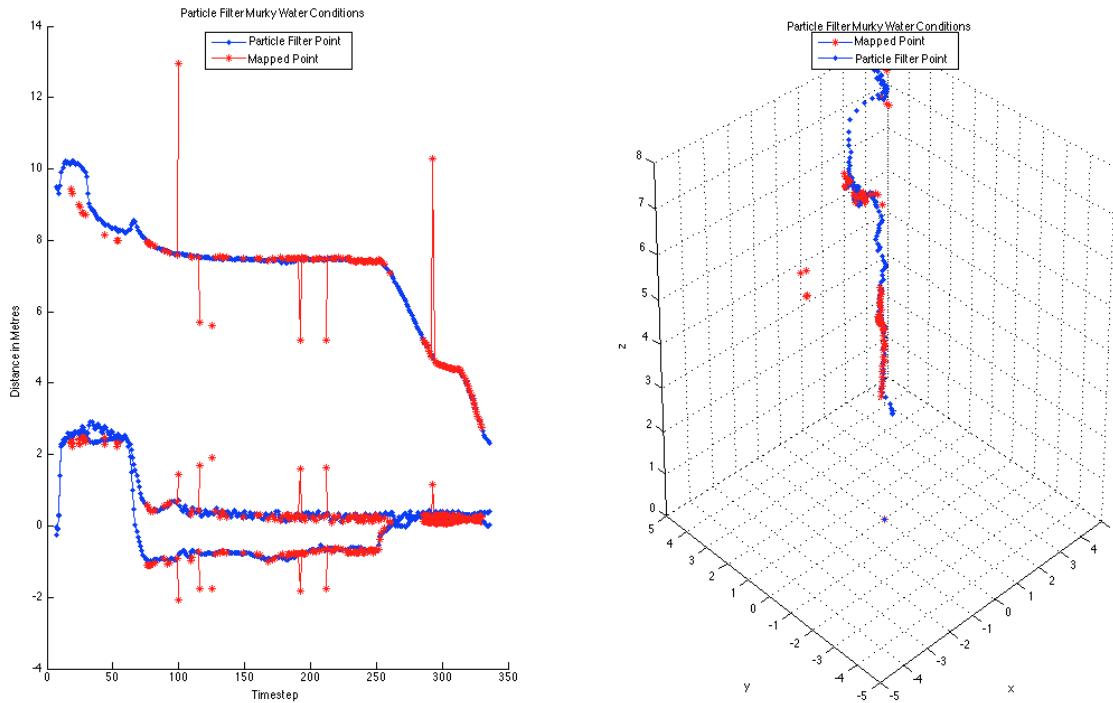


Figure 32: Particle Filter localizes the object despite noisy data in both sensors when close to a wall



(a) Pool conditions



(b) Murky water conditions

Figure 34: Comparison of object coordinates derived through the mapping method vs particle filter

References

- [1] BBAUV. Bbauv 2.5.
- [2] Teledyne Blueview. Blueview imaging sonar.
- [3] G. Bradski. Opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [4] Alberto Broggi, Alexander Zelinsky, Michel Parent, and Charles E. Thorpe. Intelligent Vehicles. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, chapter 51, pages 1175–1198. Springer Berlin Heidelberg, May 2008. ISBN 978-3-540-23957-4.
- [5] Nicholas Carlevaris-Bianco, Anush Mohan, and Ryan M. Eustice. Initial results in underwater single image dehazing. *MTS/IEEE Seattle, OCEANS 2010*, 2010.
- [6] Daniel Clark, Ioseba Tena Ruiz, Yvan Petillot, and Judith Bell. Particle PHD filter multiple target tracking in sonar image. *IEEE Transactions on Aerospace and Electronic Systems*, 43:409–415, 2007.
- [7] E Coiras and J Groen. Simulation and 3d reconstruction of sidelooking sonar images. *Advances in sonar technology. IN-TECH*, (February):1–15, 2009.
- [8] V. Creuze, B. Jouvencel, and P. Baccou. 3D-bottom tracking based on acoustic diffraction for autonomous underwater vehicles. *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pages 3–8, 2005.
- [9] D.M. Deveau and A.P. Lyons. Fluid-filled passive sonar calibration spheres: Design, modeling, and measurement. *Oceanic Engineering, IEEE Journal of*, 34(1):93–100, Jan 2009.
- [10] Paul F. Dubois. Extending python with fortran. *Computing Science and Engineering*, 1999.
- [11] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots Dieter Fox, Wolfram Burgard. *16th National Conference on Artificial Intelligence (AAAI99)*, (Hanschin 1970):343–349, 1999.
- [12] Kt Kim, Dk Seo, and Ht Kim. Radar target identification using one-dimensional scattering centres. *IEE Proceedings-Radar, Sonar and Navigation*, 152(4):285–296, 2001.
- [13] Christian Knauer and Klaus Kriegel. On the Bounding Boxes Obtained by Principal Component Analysis. *2nd European Workshop on Computational Geometry*, pages 193–196, 2006.
- [14] D.W. Krout, G. Okopal, and E. Hanusa. Video data and sonar data : real world data fusion example. *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1 – 5, 2011.
- [15] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.
- [16] Paul a. Miller, Jay a. Farrell, Yuanyuan Zhao, and Vladimir Djapic. Autonomous underwater vehicle navigation. *IEEE Journal of Oceanic Engineering*, 35(3):663–678, 2010.
- [17] S. Negahdaripour. A new method for calibration of an opti-acoustic stereo imaging system. *MTS/IEEE Seattle, OCEANS 2010*, pages 1–7, 2010.
- [18] Shahriar Negahdaripour, Hicham Sekkati, and Hamed Pirsavash. Opti-acoustic stereo imaging: On system calibration and 3-D target reconstruction. *IEEE Transactions on Image Processing*, 18(6):1203–1214, 2009.
- [19] Eiji Ota. Simple particle filter demo.

- [20] Morgan Quigley, Ken Conley, Brian Gerkey, Josh FAust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Mg. ROS: an open-source Robot Operating System. *Icra*, 3(Figure 1):5, 2009.
- [21] Anthony Spears, Ayanna M Howard, Michael West, and Thomas Collins. Acoustic Sonar and Video Sensor Fusion for Landmark Detection in an Under-Ice Environment.
- [22] P.F. Sturm and S.J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, 1(c), 1999.
- [23] Yohay Swirski and Yoav Y. Schechner. 3Deflicker from motion. In *2013 IEEE International Conference on Computational Photography, ICCP 2013*, 2013.
- [24] J. Torstensson and M. Trieb. Particle Filtering for Track Before Detect Applications. *Division of Automatic Control Department of Electrical Engineering Linkoping University, Sweden*, 2005.
- [25] S. Williams and I. Mahon. Simultaneous localisation and mapping on the Great Barrier Reef. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*. 2004, 2, 2004.
- [26] Guo H Xu, Li Y Wu, Kun Yu, Cheng Yang, and Lin Yang. Multi-target Detection of Underwater Vehicle Based on Multi-sensor Data Fusion. 4:467–471, 2012.