3D Object Localization using Forward Looking Sonar (FLS) and Optical Camera via Particle Filter based Calibration and Fusion

National University of Singapore
Bumblebee Autonomous Underwater Vehicle (BBAUV)
Yaadhav Raaj, Alex John, Tan Jin

***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 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, and a hybrid sonar/camera particle filter based calibration step and fusion step.**

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

I. 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 as seen in commercial systems such as the SABB Sabertooth and Seabotix Underwater Vehicles.

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 it's 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 [20]. This makes algorithms for thresholding or segmentation work ineffectively, which makes tracking 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 [3] 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 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 P900-90 [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 100m, 90° and 20° 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. 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.

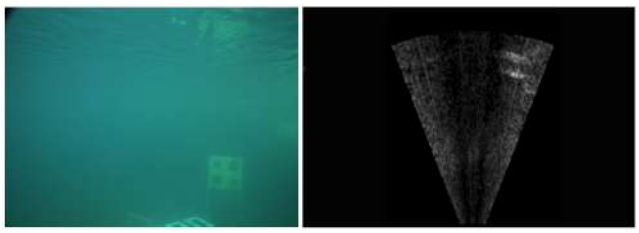


Fig. 1: Objects seen in both the camera and sonar are barely visible

The visibility limitations, intermittent loss of data due to high SNR and lack of 3D information of the object make localizing and tracking these objects difficult. In this paper, we propose a novel method for calibrating the sonar and camera, and fusing information from them along with odometry information from the vehicle using particle filter based methods, which we use to localize multiple objects in the aquatic 3D space.

II. Preliminaries

A. Literature Review

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 seabed but not ahead [22] [23]. In the seminal paper describing 3D mapping of the great barrier reef [22], 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 objects are later used in this paper.

Fig. 2: 3D map generation using sonar and camera on the Great Barrier Reef

Since the FLS does not provide elevation information, methods such as projecting the sonar ambiguity space has also been explored. In Anthony Spears paper [18], 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.

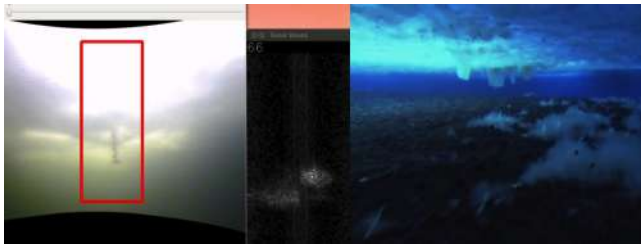


Fig. 3: Projecting sonar image into camera space on under ice topography in Spear's paper

Motion tracking approaches fusing information from sonar and camera data using a Joint Probabilistic Data Association (JDPA) fusion approach has also been studied [10]. 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 doesn't allow for depth correction.

SLAM approaches using particle filters in underwater environment [4], and AUV Rigid body dynamics [13] 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.

Lastly, state of the art calibration techniques between sonar and camera spaces by S. Negahdaripour have also been extensively studied [15] [14]. 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 [11] for non-linear optimization in order to solve for the unknown transformation parameters, and also shows the use of a special calibration grid. These methods are extremely accurate, but require elaborate setups to get working correctly, are challenging to setup in the field and occasionally require human input for matching keypoints in noisy environments. A simplified model derived from these are later used, with a particle filter based approach, easing the limitations above.

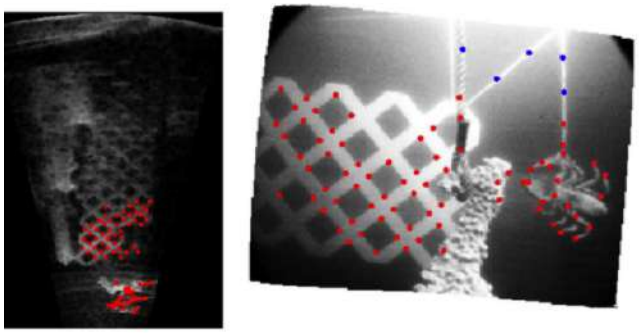


Fig. 4: Corresponding points between sonar and camera images in Negahdaripour's paper

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.

B. Vehicle Used

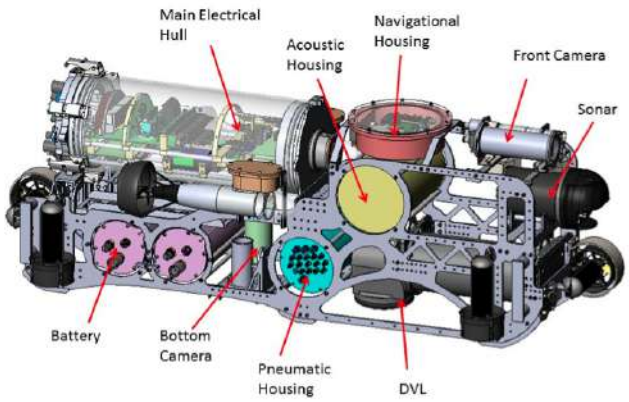


Fig. 5: BBAUV 3.0

Weight	52 kg
Dimensions	0.6m X 1.2m X 0.6m
Single Board Computer	Intel Core i5-4402E 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 AHRS-8 and STIM 300 IMU US300 Pressure/Depth Sensor
Vision Sensors	AVT Guppy Pro (Front) AVT Guppy (Bottom)
Sonar	BlueView P900-90 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) Gentoo GNU/Linux x64

TABLE I: BumbleBee AUV 3.0 Specifications

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 and similar bottom camera, Blueview P900-90 Imaging Sonar for vision, Teledyne TDI Explorer DVL (Doppler Velocity Log), Sparton AHRS-8 and STIM 300 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.

C. Sensor and System Models

The vehicle has front and bottom facing cameras about frames c and b. The camera intrinsics are calibrated via Zhangs method [19] using a checkerboard pattern. This allows us to map 3D coordinates about frame c to camera pixels (U_c, V_c) by accounting for it's focal lengths and principal points Eq. [2]. To account for camera distortion, the radial coefficients (K Values) are also estimated using Brown-Conrady model Eq. [3]

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

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

$$\begin{bmatrix} U_U \\ V_U \end{bmatrix} = \begin{bmatrix} f_x * (((U_D - c_x)/f_x) * (1 + K_1 * r^2 + K_2 * r^4)) + c_x \\ f_y * (((V_D - c_y)/f_y) * (1 + K_1 * r^2 + K_2 * r^4)) + c_y \end{bmatrix} \quad [3]$$

A Forward Looking Sonar (FLS) is also available. An FLS forms 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. Objects that are further away have a lower intensity due to signal attenuation in the medium. The FLS Model is usually represented in spherical coordinates Eq. [4] of Range, Azimuth and Elevation (R_s, Θ_s, Φ_s) but with Φ_s unknown. Hence, 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 [5] in the forward looking space.

$$\begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} = \begin{bmatrix} R_s \cdot \cos(\phi_s) \cdot \sin(\theta_s) \\ R_s \cdot \sin(\phi_s) \\ R_s \cdot \cos(\phi_s) \cdot \cos(\theta_s) \end{bmatrix} ; \begin{bmatrix} R_s \\ \Theta_s \\ \Phi_s \end{bmatrix} = \begin{bmatrix} \sqrt{X_s^2 + Y_s^2 + Z_s^2} \\ arctan(\frac{X_s}{Z_s}) \\ arcsin(\frac{Y_s}{\sqrt{X_s^2 + Y_s^2 + Z_s^2}}) \end{bmatrix} \quad [4]$$

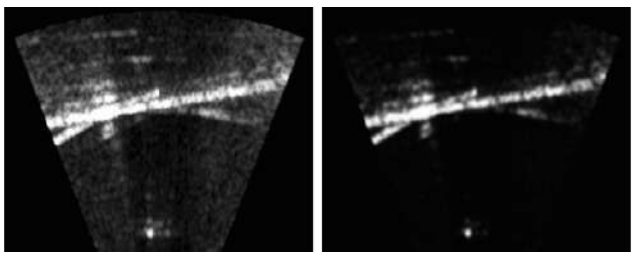


Fig. 6: Before and after applying image processing techniques on sonar image

Sonar imagery is extracted via the Blueview API. Several pre-processing steps are applied to the stream, referenced from Kim's paper on radar processing techniques [8]. A convolution with a Gaussian low pass kernel, application of power law normalization and averaging the image over 2 frames is used. These steps help reduce some of the sparse electrical or acoustic noise in the image, and are represented by Eq. [5] onwards. From there we are able to map pixels (U_s, V_s) to (R_s, Θ_s).

$$g(x,y) = e^{\frac{-\left((x-c_x)^2+(y-c_y)^2\right)}{(2\sigma)^2}} \qquad I(x,y) = I(x,y) \otimes g(x,y) \quad [5]$$

$$I(x,y) = \left(\frac{I(x,y)}{255}\right)^2 \cdot 255 \quad [6]$$

$$I(x,y) = \frac{\sum_i^2 I(x,y)}{2} \quad [7]$$

The vehicle is also equipped with a DVL (Doppler Velocity Log) which provides the velocity ($\dot{X}_v$, $\dot{Y}_v$, $\dot{Z}_v$) and altitude A_v from the sea-floor of the vehicle relative to the vehicle frame v, and can output data at up to 10Hz. It's also equipped with an IMU that provides both orientation (α_v, β_v, γ_v) and angular velocity ($\dot{\alpha_v}$, $\dot{\beta_v}$, $\dot{\gamma_v}$) about v, and a depth sensor that provides the depth D_v from the water surface, both at up to 50Hz.

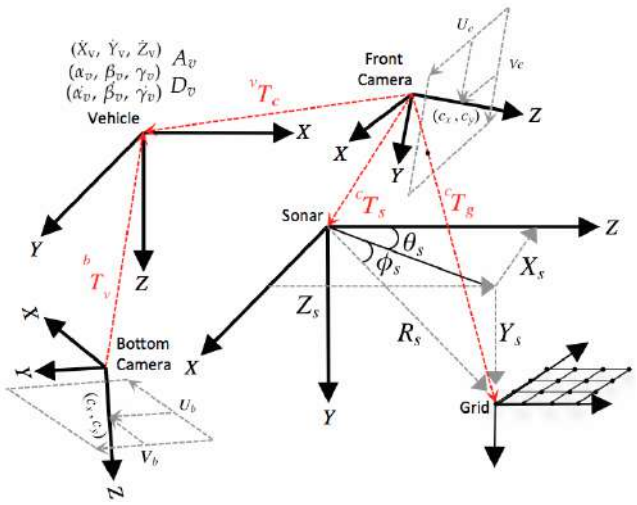


Fig. 7: System Frames

An additional grid about frame g is included which is used for the calibration step later. The vehicle to camera transforms ${}^{v}T_c$, ${}^{b}T_c$ are known from the solidworks model, and camera extrinsic calibration [Fig. 7]. However, since ${}^{c}T_s$ and ${}^{c}T_g$ directly affect the position computed for an object, our focus is on extracting this information and calibrating it precisely.

D. Particle Filter Theory

Particle Filters are used for both the calibration process of pose estimation to find ${}^{c}T_s$, and the tracking of multiple objects. Particle Filters have been used extensively in multi-sensor calibration [7] [17] and in cooperative tracking from multiple sensors. This is because they allow the modelling of complex weights from multiple sensors easily, and allow for multi-modal distributions which make working with image data much easier. The following steps below describe the 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.
$X_{n|n-1}^{(k)} = f(X_{n-1|n-1}^{(k)}, w_{n-1}) \quad k \epsilon \mathbb{N}$

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 [6].

4: *Measurement Model update* :
Compute weights given a measurement Y_n based on a multivariate gaussian distribution model where d represents the difference from mean of a feature.
$P(Y_n|X_{n|n-1}^{(k)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot exp(-\frac{d^2}{2\sigma^2})$
$log(P(Y_n|X_{n|n-1}^{(k)})) = -log(\sigma\sqrt{2\pi}) - \frac{0.5}{\sigma^2} \cdot d^2$

5: *Resample* :
Generate N new particles by resampling with replacement given normalized weight q_n
$q_n = \frac{P(Y_n|X_{n|n-1}^{(k)})}{\sum_k P(Y_n|X_{n|n-1}^{(k)})} \quad k \epsilon \mathbb{N}$
$X_{n+1|n}^{(k)} = h(X_{n|n}^{(k)}|q_n) \quad k \epsilon \mathbb{N}$

6: Go to step 2 and repeat with new inputs

The above steps can be described in [Fig. 8] [16], where prediction is done through the dynamic model update in Step. 2/3, calculation 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.

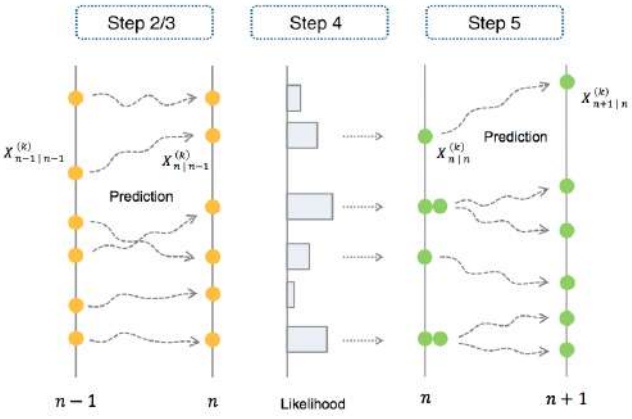


Fig. 8: Particle Filter Stages

III. Calibration

A. Grid Extraction

Our calibration method starts with a minimum of a 3x3 grid array setup, and is done with pearl weights

in our case, with the distance between each weight precisely set and known. A set of points in frame g that have a Z value of 0 are generated, and are multiplied into ${}^{c}T_{g}$ and the camera matrix K_c to give pixel coordinates m_c Eq. [8]. This can be simplified into a Homography matrix H [12], where bundle adjustment can then be used to solve for ${}^{c}T_{g}$ Eq. [9]. This allows us to get a set of 3D grid coordinates in frame c.

$$\tilde{m_c} = K_c {}^{c}\begin{bmatrix} R^1 & R^2 & R^3 & t \end{bmatrix}_g \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}_g = K \begin{bmatrix} R^1 & R^2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad [8]$$

$$H = K \begin{bmatrix} R^1 & R^2 & t \end{bmatrix} => R^3 = R^1 * R^2 \quad [9]$$

B. Particle Filter Optimization

Our objective is to find the transform ${}^{c}T_{s}$, hence our state matrix consists of the translational and rotational components of the transformation matrix with N particles. Updates are performed on the state matrix using a Brownian motion model with a Gaussian distribution w Eq. [10].

$$X_{n|n-1}^{(k)} = \begin{bmatrix} X & Y & Z & \alpha & \beta & \gamma \end{bmatrix} \qquad X_{n|n-1}^{(k)} = X_{n-1|n-1}^{(k)} + w_{n-1} \quad [10]$$

Sonar and camera imagery sets are collected. For each particle n in the weight calculation step, we iterate through each image set, extract the 3D coordinates of the grid in the camera frame using Eq. [9], transform it into the sonar frame using the current particles ${}^{c}T_{s}$ value Eq. [10], and project it into the sonar image based on Eq. [4]. This projection can be visualized.

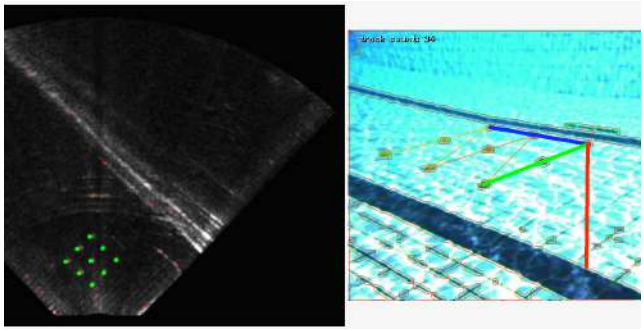


Fig. 9: Projection of calibration grid points into sonar image

The cost function of each particle k is computed to maximize the intensity of the projected points (U_s, V_s) on each sonar image i for a set of images M using our multivariate gaussian distribution model Eq. [11].

$$q_n^{(k)} = \sum_{i}^{M} \left(-log(\sigma_{Intensity}\sqrt{2\pi}) - \frac{0.5}{\sigma_{Intensity}^2} \cdot ((U_s, V_s)_{Intensity} - 255)^2 \right) \quad [11]$$

The calibration cost is minimized in the sonar image space using multiple captured sets with the planar constraints in place, and the sonar images are a representation of points in the R_s, Θ_s space, allowing a full 6DOF transform optimization without the need for manual point/feature matching.

C. Optimization Results

Our calibration method allows for convergence with initialization over a large solution space, or a localized initial but perturbed solution from the CAD model. [Fig. 10] shows the convergence before and after in the former case.

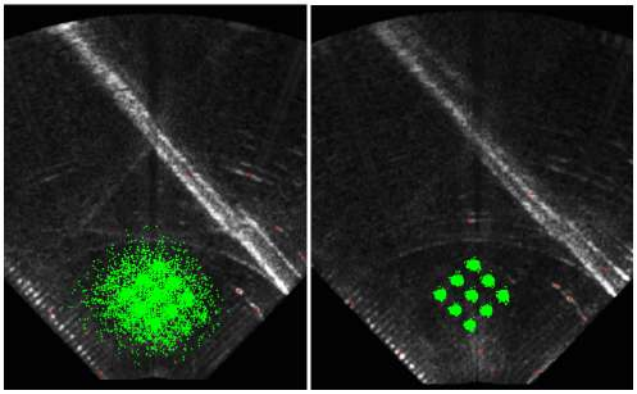


Fig. 10: Convergence of Particle Filter

Negahdaripour's paper [15] had good results, with a standard deviation of roughly 0.01m, in X, 0.005m in Z and 0.03m in Y translational components (converted to our frame conventions). This is very similar to our results as seen in [Fig. 17a]. This makes sense since the sonar image only has R_s, Θ_s mappings embedded in it, and changes of an object's position in the Y axis only affects R_s based on Eq. [4]. We also had significantly more uncertainty in the rotation component about the Z axis (which translates to the rolling component in the vehicle frame).

In his paper however, it is necessary to perform feature correspondence or matching between the sonar and camera images, which can be computationally expensive. Since a gradient descent method based on the Levenberg Marquardt algorithm [11] is used, it is necessary to explicitly pass in correct data sets for optimization. To ensure no incorrect correspondences are passed in, many pre-processing methods must be employed, and a good initial condition is required to prevent convergence to a wrong local minima. With our particle filter based method, our solution can be directly computed in the image space, and can begin with a very large uncertainty, without the need for any thresholding or feature extraction methods. It is also able to achieve similar results at a much larger range of 3 to 5m.

IV. Tracking

A. Tracking Initialization

Our filter's state vector is defined in 3D cartesian coordinates and 3D velocities in the sonar frame.

$$\begin{bmatrix} X_s & Y_s & Z_s & \dot{X}_s & \dot{Y}_s & \dot{Z}_s \end{bmatrix} \quad [12]$$

Our tracking method works on knowing the object's dimensions, and trained camera model through machine

learning techniques such as CNNs (Convolutional Neural Networks) [9] or color histogram as priors. Tracking can be initialized either in the sonar frame or camera frame. In the sonar frame, prior filtering, adaptive thresholding and morphology has been applied [21] and the Lucas Kanade method is used to remove the spurious thresholded objects. In the sonar frame, one may select their desired object size via a major and minor axis range. We use this to compute the search space given by the object's size and project that into the camera image [18] since ${}^{c}T_{s}$ is known [Fig. 11], and determine if there is an object to be tracked in the camera frame. From there, we initialize our filter in the sonar frame in polar coordinates, with R_s, Θ_s with a small uncertainty, and Φ_s with a large one with a random distribution, and then convert it to cartesian coordinates.

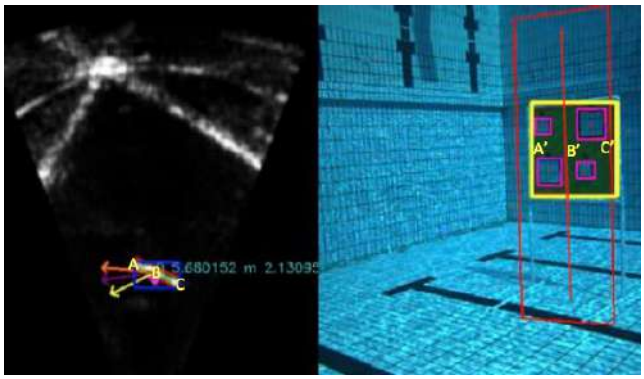


Fig. 11: Projecting sonar object into camera image

We can also begin initialization in the camera frame. If an object is detected in the camera frame, we can compute the object's Θ_c, Φ_c from (U_c, V_c), Eq. [14], initialize polar coordinates in the camera frame with a larger uncertainty in R_c, convert it to cartesian coordinates, and transform it to the sonar frame via ${}^{c}T_{s}$. One can observe uncertainty in R_s vs uncertainty in Θ_s in [Fig. 12].

$$\Theta_c = atan(\frac{U_c - c_x}{f_x}) \quad [13]$$

$$\Phi_c = atan(\frac{(V_c - c_y).cosd(\Theta_c)}{f_y}) \quad [14]$$

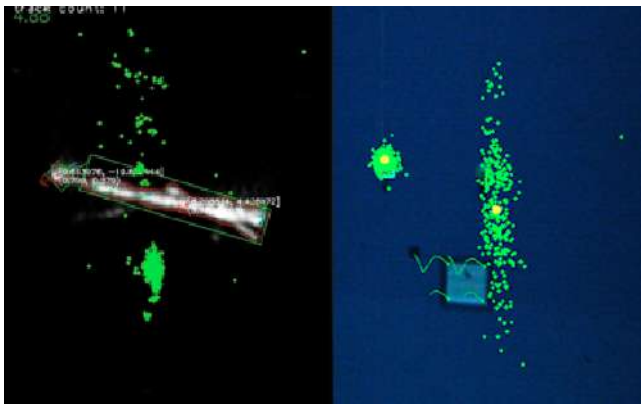


Fig. 12: Uncertainty in Range or Elevation

B. Dynamic Model

The state matrix and dynamic model Eq. [15] 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 gaussian process noise w added.

$$X_{n|n-1}^{(k)} = \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ \dot{X}_s \\ \dot{Y}_s \\ \dot{Z}_s \end{bmatrix} \quad X_{n|n-1}^{(k)} = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * X_{n-1|n-1}^{(k)} + w_{n-1} \quad [15]$$

This model is ideal for tracking, and ensures that objects that move quickly in the image are statistically more likely to regenerate velocities matching the object velocity after resampling, and allow for some level of prediction if there is information loss in some frames. However, in most scenarios, the vehicle is going to be moving significantly faster than the object is when it is approaching the target. Sudden movements such as pitching, yawing or surging forward can cause the system to lose track of the object. Hence, the vehicle's relative velocity $\dot{P}_v$ and angular velocity $\dot{\omega}_v$ from the DVL and IMU are added onto $(\dot{X}_s, \dot{Y}_s, \dot{Z}_s)$ before computing the update matrix Eq. [17].

$$\dot{P}_s = \dot{P}_s + {}^{s}R_v.(\dot{P}_v + \dot{\omega}_v \times ({}^{v}T_s.P_s)) \quad [16]$$

$$\begin{bmatrix} \dot{X}_s \\ \dot{Y}_s \\ \dot{Z}_s \end{bmatrix} = \begin{bmatrix} \dot{X}_s \\ \dot{Y}_s \\ \dot{Z}_s \end{bmatrix} + {}^{s}R_v * \left(\begin{bmatrix} -\dot{X}_v \\ -\dot{Y}_v \\ -\dot{Z}_v \end{bmatrix} + \begin{bmatrix} 0 & -\dot{\gamma}_v & \dot{\beta}_v \\ \dot{\gamma}_v & 0 & -\dot{\alpha}_v \\ -\dot{\beta}_v & \dot{\alpha}_v & 0 \end{bmatrix} * {}^{v}T_s * \begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} \right) \quad [17]$$

When the vehicle is too close to the sea floor, the DVL may not produce any data. However, features on the sea floor become alot more visible in the bottom camera. We also know the rough distance to the sea floor based on the previously known altitude A_v, and rate of change of it $\dot{A}_v$ from the DVL and updates on that from the US300 pressure sensor. With the assumption that A_v, $\dot{A}_v$ is correct, and rate of change in depth is minimal, we can track features extracted via the Lucas Kanade method in the bottom camera frame b, and compute the vehicle relative velocity. This produces a matrix with a vector containing the feature pixel position and pixel velocity. The velocity extracted here is a addition C of both the vehicle's relative translational velocity T and it's rotational velocity R.

$$\begin{bmatrix} U_b & V_b & \dot{U}_b & \dot{V}_b \end{bmatrix}^{C} \quad [18]$$

We need to compute the rotational component pixel velocity to remove it from C. This can be done by backprojecting the above pixels into 3D coordinates P_b in the bottom camera frame with the camera parameters

known Eq. [19], and the Z component known from A_v with the offsets made. We then compute the 3D velocity $\dot{P_b^R}$ of those features in the rotational component Eq. [20], and then compute the pixel velocity $(\dot{U_b}, \dot{V_b})$ by differentiating the camera projection equations wrt time Eq. [21].

$$P_b = \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = \begin{bmatrix} (U_b - c_x).Z_b.f_x \\ (V_b - c_y).Z_b.f_y \\ A_v - {}^v Zoffset_b \end{bmatrix} \quad [19]$$

$$\dot{P_b^R} = {}^b R_v.(({}^b R_v.\dot{\omega_v}) \times ({}^v T_b.P_b)) \quad [20]$$

$$\begin{bmatrix} \dot{U_b} \\ \dot{V_b} \end{bmatrix}^R = \begin{bmatrix} f_x.(\frac{\dot{X_b}.Z_b - \dot{X_b}.Z_b}{Z_b^2}) \\ f_y.(\frac{\dot{Y_b}.Z_b - \dot{Y_b}.Z_b}{Z_b^2}) \end{bmatrix} \quad [21]$$

From there, we can remove the rotational component of the pixel velocity Eq. [22], and then compute the vehicles relative tangential velocity in the bottom camera frame Eq. [23]. We can transform this back into the vehicle frame for use in Eq. [15]

$$\begin{bmatrix} \dot{U_b} \\ \dot{V_b} \end{bmatrix}^T = \begin{bmatrix} \dot{U_b} \\ \dot{V_b} \end{bmatrix}^C - \begin{bmatrix} \dot{U_b} \\ \dot{V_b} \end{bmatrix}^R \quad [22]$$

$$\begin{bmatrix} \dot{X_b} \\ \dot{Y_b} \\ \dot{Z_b} \end{bmatrix}^T = \begin{bmatrix} \frac{f_x.X_b.\dot{Z_b} - \dot{U_b}.Z_b^2}{f_x.Z_b} \\ \frac{f_y.Y_b.\dot{Z_b} - \dot{V_b}.Z_b^2}{f_y.Z_b} \\ \dot{A_v} \end{bmatrix} \quad [23]$$

C. Measurement Model

For the weight computation, we use various heuristics, such as the intensity of the sonar image, the major minor axis size in the sonar image, color histogram in the camera frame or testing a subsection of the image where a particle is against our CNN filter. This is done by projecting the 3D coordinate in a particle state into both the sonar and camera frames, and testing if the particle is inside the image space, or inside a thresholded contour in the case of the major minor axis computation.

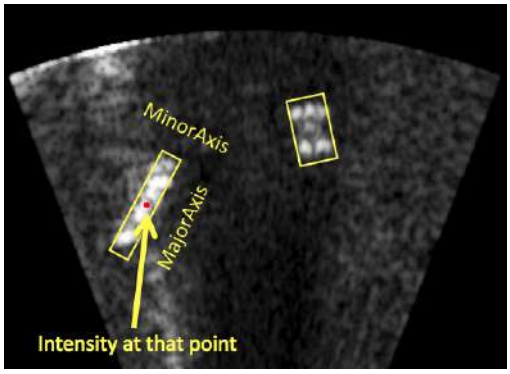


Fig. 13: Major Minor axis distance in sonar image

We continue to use our multivariate gaussian distribution model, where weights are computed for every particle in both the sonar and camera image, for each parameter described above with a user defined ideal parameter value $User_{Param})^2$ with it's own standard deviation σ_{Param}, with the transformation constraint ${}^c\mathrm{T_s}$ in place. Resampling is then done, before the update model is applied again.

$$q_n^{camera} = -log(\sigma_{Param}\sqrt{2\pi}) - \frac{0.5}{\sigma_{Param}^2} \cdot ((U_c, V_c)_{Param} - User_{Param})^2 \quad [24]$$

$$q_n^{sonar} = -log(\sigma_{Param}\sqrt{2\pi}) - \frac{0.5}{\sigma_{Param}^2} \cdot ((U_s, V_s)_{Param} - User_{Param})^2 \quad [25]$$

$$P(Y_n | X_{n|n-1}^{(k)}) = q_n^{sonar} + q_n^{camera} \quad [26]$$

Our particles will then converge over our object of interest, and even if there is temporary occlusion, loss of information in a sensor or if the object we are tracking goes out of the sensor's field of view, prediction with our dynamic model ensures that the object's position gets correct accordingly. With our system, you can also track multiple objects. We make use of the Mahalanobis distance Eq. [27] to ensure that the same object isn't added to our tracker. Since detection begins with image processing and extraction of sonar and camera pixels, this is done on a pixel space in both the sonar and camera space, with the particle filter's state matrix $\overrightarrow{x}$, covariance C and mean $\overrightarrow{u}$ extracted. If this distance exceeds a defined threshold in both cases, we add a new tracker.

$$D = \sqrt{(\overrightarrow{x} - \overrightarrow{u})^T.S^{-1}.(\overrightarrow{x} - \overrightarrow{u})} \quad [27]$$

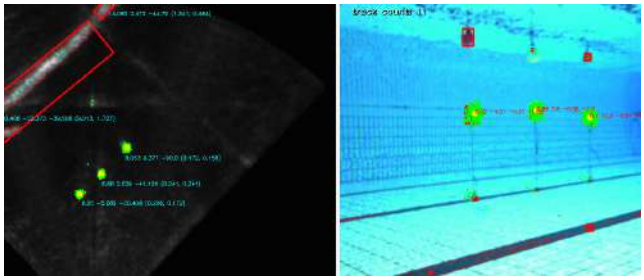


Fig. 14: Tracking of 3 Buoys

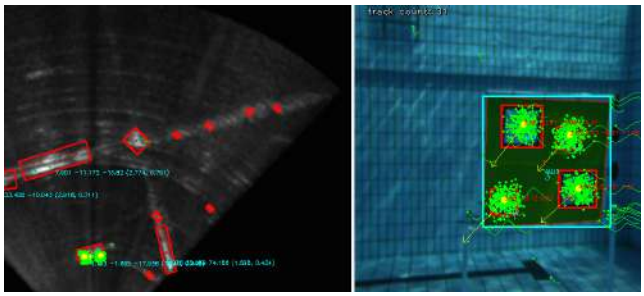


Fig. 15: Tracking of 4 Structures

D. Tracking Results

To test the viability of the filter, we used buoys in both pool conditions and murky water conditions, with a diver to move the buoy to induce non-stationary motion as well, and jerky movement of the vehicle with increased pitching, yawing and forward /backward motion. We use a direct mapping algorithm similar to [Fig. 11] to search for the buoy and extract (R_s,

Θ_s) and (U_c, V_c) to compute the buoy's 3D coordinate using a closed form solution as a ground truth. We also extract the mean result of the particle filter tracker. We can see the results in [Fig. 18a] and [Fig. 18b]. In pool conditions, there is occasional loss of data using a direct mapping approach, especially during sudden motions of the vehicle. However, the particle filter is able to continue to track the object. In murky water conditions, there is significant spurious noise using the direct mapping approach, but the particle filter method is able to maintain a consistent track on the buoy in 3D space, with minimal error from the ground truth.

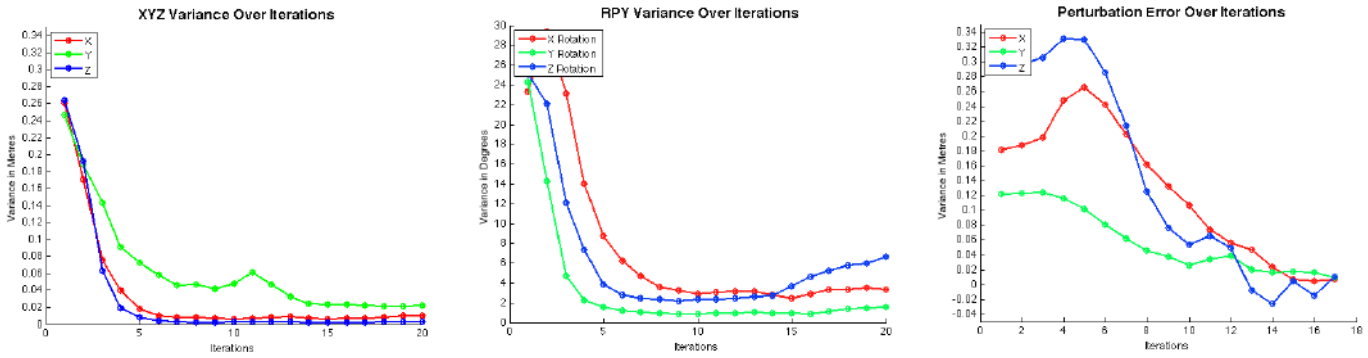
V. Conclusion

Object tracking and localization in the forward looking space at ranges greater than around 5m will always be challenging, due to poor lighting conditions and hazing over large distances with minimal features to track, or due to the much more complex scene due to the nature of forward facing situations since a much larger range is present. Past approaches for tracking have either used the sonar imagery alone, or have used the camera and sonar with methods requiring complex data association techniques. Our solution eliminates the need for this by using a particle filter based tracker, using a dynamic velocity update model making use of vehicular odometry data, and a locking constraint based on prior extrinsic calibration between the sonar and camera. If one sensor becomes more noisy or has information loss, the object to be tracked can still have it's position corrected by another, making our tracking method extremely robust. It also allows one to easily add customizable cost functions in the measurement update step with minimal change to the code base. Lastly, using our method with machine learning techniques gets significantly faster since it eliminates the need for sliding window techniques over an entire image.

Our unique particle filter based sonar camera calibration method is also fully automated, and does not require any user intervention. It is able to achieve similar results to existing methods, and can be further improved with a larger grid.

Fig. 16: BBAUV performing object localization autonomously

Our algorithm was implemented using ROS (Robot Operating System), OpenCV and Numpy, and is able to track up to 8 objects simultaneously. With our tracking method, we were able to track and localize objects up to 15m away in difficult water conditions. Also, with our tracking method, it is possible to further extend this into a SLAM approach to help localize the vehicle's position in the water better.

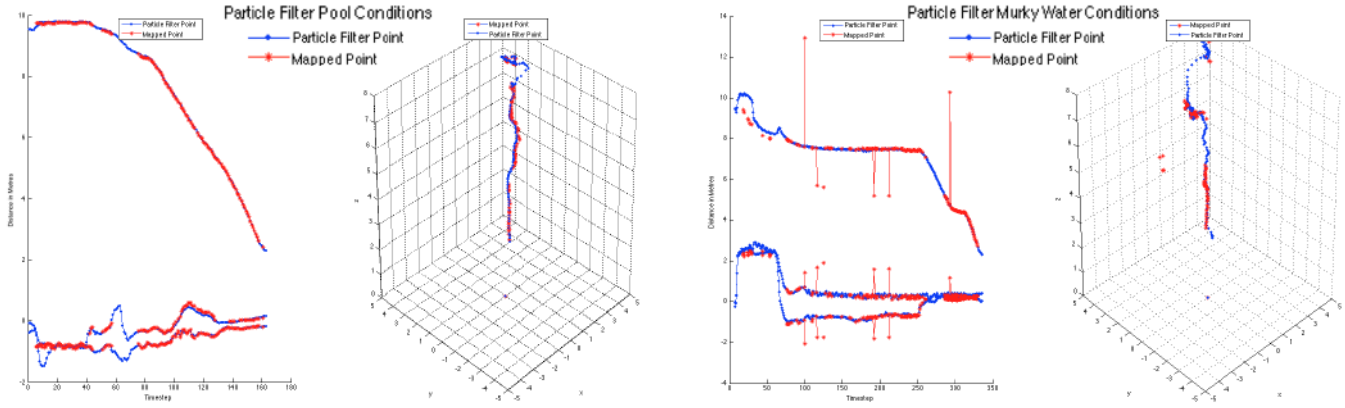


(a) Variance in translational components, with greater uncertainty in the y axis wrt the sonar frame

(b) Variance in rotational components, with greater uncertainty in the Z axis rotation wrt the sonar frame

(c) Error from ground truth due to initial perturbation per iteration in translational components

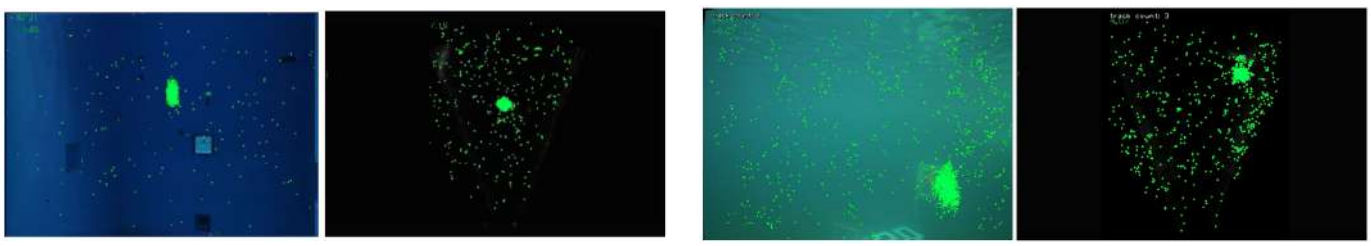
Fig. 17: Graphs showing change in variance in translation and rotational components during calibration per iteration, and a perturbation from the ground truth converging to a zero mean error



(a) Tracking in pool conditions. Occasional loss of data during sudden or fast movements with mapping

(b) Tracking in murky conditions. Significant loss and noise of data during sudden or fast movements

Fig. 18: Direct computation of 3D coordinates using detection and mapping approach, vs our particle filter based approach in all axis while tracking over time



(a) Tracking in pool conditions

(b) Tracking in murky water conditions

Fig. 19: Particle filter convergence and tracking

REFERENCES

[1] BBAUV. Bbauv 3.0.

[2] Teledyne Blueview. Blueview imaging sonar.

[3] Nicholas Carlevaris-Bianco, Anush Mohan, and Ryan M. Eustice. Initial results in underwater single image dehazing. *MTS/IEEE Seattle, OCEANS 2010*, 2010.

[4] 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.

[5] E Coiras and J Groen. Simulation and 3d reconstruction of sidelooking sonar images. *Advances in sonar technology. IN-TECH*, (February):1–15, 2009.

[6] 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)*, (Handschin 1970):343–349, 1999.

[7] F Herranz, K Muthukrishnan, and K Langendoen. Camera pose estimation using particle filters. *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*, pages 1–8, 2011.

[8] 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.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.

[10] 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.

[11] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[12] P. R. S. Mendonça and R. Cipolla. A simple technique for self-calibration. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume I, pages 500–505, Fort Collins, Colorado, 1999.

[13] 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.

[14] S. Negahdaripour. A new method for calibration of an opti-acoustic stereo imaging system. *MTS/IEEE Seattle, OCEANS 2010*, pages 1–7, 2010.

[15] Shahriar Negahdaripour, Hicham Sekkati, and Hamed Pirsiavash. Opti-acoustic stereo imaging: On system calibration and 3-D target reconstruction. *IEEE Transactions on Image Processing*, 18(6):1203–1214, 2009.

[16] Eiji Ota. Simple particle filter demo.

[17] M. Pupilli and a. Calway. Particle Filtering for Robust Single Camera Localisation. *Proc of the 1st International Workshop on Mobile Vision.*, pages 1–14, 2006.

[18] Anthony Spears, Ayanna M Howard, Michael West, and Thomas Collins. Acoustic Sonar and Video Sensor Fusion for Landmark Detection in an Under-Ice Environment.

[19] 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.

[20] Yohay Swirski and Yoav Y. Schechner. 3Deflicker from motion. In *2013 IEEE International Conference on Computational Photography, ICCP 2013*, 2013.

[21] K. Teo, K.W. Ong, and H.C. Lai. Obstacle detection, avoidance and anti collision for MEREDITH AUV. *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges*, 2009.

[22] 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.

[23] 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.