# Technical report: Underwater feature extraction and pillar mapping

Unnar Þór Axelsson and John Folkesson

***Abstract*—A mechanicaly scanned imaging sonar, MSIS, produces a 2D image of the range and bearing of return intensities. The pattern produced in this image depends on the enviormental feature that caused it. These features are very useful for underwater navigation but the inverse mapping of sonar image pattern to environmental feature can be ambiguous. We investigate problems associated with using MSIS for navigation. In particular we show that support vector machines can be used to classify the existance and types of feature in a sonar image. We develop a sonar processing pipleline that can be used for navigation. This is tested on two sonar datasets collected from ROV's.**[1]

## I. Introduction

Navigation and exploration of a previously unknown areas is a necessary capability of a truly autonomous robot. Simultaneous localization and mapping (SLAM) is a well known and researched area aimed at solving this problem[1], [2], [3]. Multiple variations of SLAM have been suggested and shown good results, especially in indoor and outdoor situations where laser scanners or accurate gps measurements can be used. However in underwater situations these sensors are not available making underwater slam a challenging problem, instead most remotely operated underwater vehicles (ROVs) rely on less accurate acoustic sensors such as imaging sonars instead of laser scanners to extract features from the environment [4], [5], [6]

This paper is focused on ROVs navigating in man made underwater environments such as harbours, marinas or around piers. These man made environments are often made up of many repeating structures such and walls or cylindrical pillars that hold up a pier. Being able to detect and correctly classify these different structures can be vital for correct mapping and localization of the environment, however very little work has been done on the subject. In this paper we present a method that extracts and classifies both pillar and wall features, furthermore we present a simple SLAM method based in iterative closest point (ICP) to use the pillar structures extracted from the environment to build a map of the environment and localize to it.

For this purpose we use data from two different datasets, the first dataset [7] was gathered in an abandoned marina near St. Pere Pescador, Spain while the second dataset was gathered at the Stevens pier on the Hudson river. A picture of the environment where these two datasets where gathered can be seen in Figure 2. Two different ROVs with different set of sensors were used to capture the two datasets, for that reason


(a) Marina dataset. 42.203321, 3.105446

(b) Pillar dataset.

Figure 1: Shows the environment for the two different datasets. The marina datasets is in an abandoned marina where the only features are walls while the pillar dataset is a pier on the Hudson river where only pillars are present.

the deadreckoning estimate for them does not work in the same way. However they are both equipped with a Mechanically Scanned Imaging Sonar (MSIS) making the feature extraction comparable. Furthermore since the first dataset contains wall features and the second pillars we can investigate methods of distinguishing these types of features from one another and from the background.

The rest of this paper is organized in the following way. Section II describes the position estimation used for the ROVs in the two datasets. Section III describes the feature extraction from the sonar data. Section IV describes how the extracted features are used to incrementally build up a map of the environment and improve to position estimation. Section VI details the results found and finally Section VII gives a short conclusion and future work.

[1]ISBN 978-91-7595-821-7, KTH

## II. Dead-reckoning Estimation

Since the two different ROVs used to gather the datasets do not contain the same set of sensor two different position estimation methods needed to be implemented. The purpose of the dead-reckoning is to estimate the motion well enough to compensate it during the time of a single 360 degree scan, to give an inital estimate for the icp algorithm used to align the features seen in different scans and finally to constrain the estimates in direction not well constrained by the features them selves, for example the depth or tranlations along a single wall.

(a) Marina ROV. (b) Pillar ROV.

Figure 2: Shows the two ROVs used for the experiments.

### A. Marina dataset position estimation

The position estimation uses an Extended Kalman Filter (EKF) designed in [7] for the same dataset with good results. There a simple constant velocity kinematic model is used for the prediction, a doppler velocity log unit (DVL) provides velocity and depth measurements and a compass provides attitude measurements.

*1) Initialization:* Because the ROV is relatively stable both roll and pitch can be ignored, thus the state $\hat{x}_k$ of the ROV at step $k$ can be defined as

$$\hat{\mathbf{x}}_k = \begin{bmatrix} x & y & z & \psi & u & v & w & r \end{bmatrix}^T, \tag{1}$$

here $(x \;\; y \;\; z \;\; \phi)$ represents the position and attitude and $(u \;\; v \;\; w \;\; r)$ the linear and angular velocities.

When a new run is started a local reference frame L is defined as a base and the robot is assumed to be positioned in the center with zero angular and linear velocity, the state is then initialized as

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} 0 & 0 & 0 & \psi_0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{2}$$

$$\mathbf{P}_0 = \begin{bmatrix} 0_{3\times3} & 0_{3\times1} & 0_{3\times4} \\ 0_{1\times3} & \sigma^2_{\psi_0} & 0_{1\times4} \\ 0_{3\times3} & 0_{4\times1} & 0_{4\times4} \end{bmatrix}. \tag{3}$$

Here $\psi_0$ is the first available measurement from the compass and $\sigma^2_{\psi_0}$ is it's estimated uncertainty.

*2) Prediction step:* For the prediction step we use the standard ekf equations

$$\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1|k-1}, \mathbf{s}) \longrightarrow$$

$$\begin{bmatrix} x \\ y \\ z \\ \psi \\ u \\ v \\ w \\ r \end{bmatrix}_{k|k-1} = \begin{bmatrix} x + (uT + s_u\frac{T^2}{2})\cos(\psi) \\ -(vT + s_v\frac{T^2}{2})\sin(\psi) \\ y + (uT + s_u\frac{T^2}{2})\sin(\psi) \\ +(vT + s_v\frac{T^2}{2})\cos(\psi) \\ z + wT + s_w\frac{T^2}{2} \\ \psi + rT + s_r\frac{T^2}{2} \\ u + s_uT \\ v + s_vT \\ w + s_wT \\ r + s_rT \end{bmatrix}_{k-1|k-1} \tag{4}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{\mathbf{x}}\mathbf{P}_{k-1|k-1}\mathbf{F}^T_{\mathbf{x}} + \mathbf{F}_{\mathbf{s}}\mathbf{Q}\mathbf{F}^T_{\mathbf{s}}. \tag{5}$$

Here $\hat{\mathbf{s}} = \begin{bmatrix} s_u & s_v & s_w & s_r \end{bmatrix}^T$ is an acceleration white noise additive with covariance $\mathbf{Q}$ and $\mathbf{F}_{\hat{\mathbf{x}}}$ and $\mathbf{F}_{\hat{\mathbf{s}}}$ are the Jacobian matrices of the nonlinear function $f$ with respect to $\hat{\mathbf{x}}$ and $\hat{\mathbf{s}}$.

*3) Update step:* Whenever a new measurement from either the DVL or the compass is available the state is updated using the standard Kalman filter equations. The measurements are given by

$$\mathbf{z}_{DVL,k} = \begin{bmatrix} u & v & w & r \end{bmatrix}^T, \qquad \mathbf{z}_{C,k} = \begin{bmatrix} \psi \end{bmatrix} \tag{6}$$

and the measurement model becomes

$$\mathbf{z}_{DVL,k} = H_{DVL}\hat{\mathbf{x}} + w_k, \quad \mathbf{z}_{C,k} = H_C\hat{\mathbf{x}} + w_k \tag{7}$$

$$H_{DVL} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

$$H_C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{9}$$

Here $w_k$ is a white Gaussian measurement noise with zero mean.

### B. Pillar dataset position estimation

This dataset was gathered on the Hudson river outside Stevens Institute of Technology. A VideoRay 4 Pro ROV equipped with a Tritech Micron DST Scanning Sonar, Micron-Nav USBL positioning system, depth sensor and a compass. The ROV can be seen in Figure 2b.

A dead-reckoning position estimation system is design for this ROV and setup based on a Kalman filter.

*1) Initialization:* Due to differences in both the design of the ROVs and the Hudson river being a more difficult environment with stronger currents than the marina the ROV is not as stable, for this reason roll and pitch can't be ignored in the position estimation. To estimate the position two separate Kalman filters were designed, one to estimate the (x,y) position

and one to estimate the orientation (roll, pitch, yaw). The state of the ROV at step k can then be written as

$$\hat{\mathbf{x}}_{P,k} = [\,\dot{x}\ \dot{y}\ x\ y\,]^T \qquad \hat{\mathbf{x}}_{O,k} = \left[\,\dot{\psi}\ \dot{\theta}\ \dot{\gamma}\ \psi\ \theta\ \gamma\,\right]^T \tag{10}$$

where $(x\ y\ \psi\ \theta\ \gamma)$ represent position and orientation and $(\dot{x}\ \dot{y}\ \dot{\psi}\ \dot{\theta}\ \dot{\gamma})$ represent linear and angular velocities. As before when a new run is started a local reference frame L is defined and the robot is placed at the origin with zero linear and angular velocity and the heading initialized as the first available measurement from the compass. This can be written as

$$\hat{\mathbf{x}}_{P,0} = [\,0\ 0\ 0\ 0\,]^T \qquad \hat{\mathbf{x}}_{O,0} = \left[\,0\ 0\ 0\ \sigma^2_{\psi_0}\ 0\ 0\,\right]^T \tag{11}$$

$$\mathbf{P}_{P,0} = \left[0_{2\times 2}\right]\ \mathbf{P}_{O,0} = \begin{bmatrix} 0_{3\times 3} & 0_{3\times 1} & 0_{3\times 1} & 0_{3\times 1} \\ 0_{1\times 3} & \sigma^2_{\psi_0} & 0 & 0 \\ 0_{1\times 3} & 0 & \sigma^2_{\theta_0} & 0 \\ 0_{1\times 3} & 0 & 0 & \sigma^2_{\gamma_0} \end{bmatrix} \tag{12}$$

where $(\sigma^2_{\psi_0}\ \sigma^2_{\theta_0}\ \sigma^2_{\gamma_0})$ are the estimated uncertainties in the orientation.

*2) Prediction step:* For the prediction we use the standard Kalman filter equation

$$\mathbf{x}_{P,k|k-1} = \mathbf{A_P}\mathbf{x}_{P,k-1|k-1} + \omega_{\mathbf{P}} \tag{13}$$

$$\mathbf{x}_{O,k|k-1} = \mathbf{A_O}\mathbf{x}_{O,k-1|k-1} + \omega_{\mathbf{O}} \tag{14}$$

with

$$\mathbf{A}_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ dt & 0 & 1 & 0 \\ 0 & dt & 0 & 1 \end{bmatrix} \quad \omega_P = \begin{bmatrix} 0 \\ 0 \\ \omega_x \\ \omega_y \end{bmatrix} \tag{15}$$

$$\mathbf{A}_O = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ dt & 0 & 0 & 1 & 0 & 0 \\ 0 & dt & 0 & 0 & 1 & 0 \\ 0 & 0 & dt & 0 & 0 & 1 \end{bmatrix} \quad \omega_O = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \omega_\psi \\ \omega_\theta \\ \omega_\gamma \end{bmatrix}. \tag{16}$$

Here $dt$ is the duration from the last prediction. The covariance can then be updated with

$$\mathbf{P}_{P,k|k-1} = \mathbf{A}_P\mathbf{P}_{P,k-1|k-1}\mathbf{A}_P^T + \mathbf{Q}_P \tag{17}$$

$$\mathbf{P}_{O,k|k-1} = \mathbf{A}_O\mathbf{P}_{O,k-1|k-1}\mathbf{A}_O^T + \mathbf{Q}_O \tag{18}$$

where $\omega_P$ and $\omega_O$ is a gaussian white noise with zero mean and $\mathbf{Q}_P$ and $\mathbf{Q}_O$ there respective covariance matrices.

*3) Update step:* Whenever a new measurement is available from the USBL, compass or the sonar the state is updated using the standard Kalman filter equations. However since the USBL and the compass provide absolute measurements and the sonar provides incremental measurements they need to be made compatible for the sonar to have any influence on the estimation.

In order to do this we define a bias for the position and yaw

$$\beta_{P,k} = [x_\beta\ y_\beta]^T \qquad \beta_{O,k} = [\psi_\beta]. \tag{19}$$

The bias is then added to the USBL and compass measurements and updated whenever a new sonar measurement is received.

The measurements from the USBL and the compass at step k are given by

$$\mathbf{z}_{USBL,k} = [\,x\ y\,]^T \qquad \mathbf{z}_{C,k} = [\,\psi\,] \tag{20}$$

and after the bias is taken into account the measurement model becomes

$$\mathbf{z}_{USBL,k} + \beta_{P,k} = H_{USBL}\hat{\mathbf{x}}_{P,k} + w_k \tag{21}$$

$$\mathbf{z}_{C,k} + \beta_{O,k} = H_C\hat{\mathbf{x}}_{O,k} + w_k \tag{22}$$

with

$$H_{USBL} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

$$H_C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{24}$$

As before $w_k$ is Gaussian white noise with zero mean.

For the sonar the measurements are in the form of a transformation matrix $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} \cos\theta' & \sin\theta' & tx \\ -\sin\theta' & \cos\theta' & tx \\ 0 & 0 & 1 \end{bmatrix}. \tag{25}$$

The total rotation is then given by $\theta'$ and the position estimation can be found from the transformation matrix by multiplying it with the current x and y estimation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{26}$$

Since we have two Kalman filters, one for position and one for orientation we look at it as two separate measurements

$$z_{SP,k} = [\,x'\ y'\,]^T \qquad \mathbf{z}_{SO,k} = [\,\theta'\,] \tag{27}$$

and the measurement model becomes

$$\mathbf{z}_{SP,k} = H_{SP}\hat{\mathbf{x}}_{P,k} + w_k \tag{28}$$

$$\mathbf{z}_{SO,k} = H_{SO}\hat{\mathbf{x}}_{O,k} + w_k \tag{29}$$

with

$$H_{SP} = H_{USBL} \qquad H_{SO} = H_C. \tag{30}$$

Finally the bias can be updated after each sonar update using

$$\beta_{P,k} = \beta_{P,k-1} + H_{\beta O}\left(\hat{\mathbf{x}}_{P,k|k} - \hat{\mathbf{x}}_{P,k|k-1}\right) \tag{31}$$

$$\beta_{O,k} = \beta_{O,k-1} + H_{\beta P}\left(\hat{\mathbf{x}}_{O,k|k} - \hat{\mathbf{x}}_{O,k|k-1}\right) \tag{32}$$

with

$$H_{\beta O} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{33}$$

$$H_{\beta P} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{34}$$

## III. MSIS Data Processing

Mechanically Scanned Imaging Sonar (MSIS) has been a very popular choice for underwater vehicles for many years. The sonar can scan a full 360-deg plane around the ROV by using a mechanically rotating head that sends out an acoustic beam at fixed angular intervals to get an intensity profile of the environment.

The main disadvantages with the sonar is the noisy measurements that they get and unlike laser scanners that are popular on mobile robots where a full 360-deg scan can be acquired almost instantly, the slow mechanical rotation of the sonar results in a scan time of several seconds. For this reason the ROV can move significantly during the time that a full scan is acquired which means that the movement of the ROV needs to be taken into account.

In the this section we introduce a feature extraction method that is able to detect two types of features from the environment (walls and pillars). First a beam segmentation is performed on each acoustic beam to extract possible hits on real objects, then each hit is placed in a buffer covering a 360 degree scan where the movement of the ROV has been accounted for and finally individual features are extracted from the buffer using a machine learning method.

### A. Beam segmentation

Each beam arriving from the sonar is associated with a specific bearing angle with respect to the sonar and is given by a set of intensity values where each value represents a fixed distance from the sonar. The method that we use to locate possible hits is threefold. First we try to extract regions of interest from the noise by applying a fixed threshold on both the intensity and the distance. By applying a threshold on the intensity most of the background noise can be removed and by thresholding the distance, regions with higher noise level can be ignored, for example far away from the sonar or very close.

Next the best local maximums are extracted from the remaining data under the criteria that they can not be closer to each other than a certain distance $d_m$. That means that if two local maxima exist where the distance between them is smaller than $d_m$ the smaller local maxima is ignored. Finally for each detected maxima a Gaussian function is fitted to a small area around the maxima and made sure that the variance $\sigma^2$ falls with in a predefined range $\sigma_l^2 \leq \sigma^2 \leq \sigma_h^2$. This helps with removing spurious and nosy measurements that the thresholding didn't remove. A picture showing a typical measurement from the sonar can be seen in Figure 3.

### B. Buffer propagation

To be able to do feature extraction and localization the hits from the beam segmentation need to be placed in a buffer to be precessed along with consecutive hits. However due to the slow scan time of the sonar the hits can not be stored in the frame of the sonar. For that reason the hits are transferred to a global coordinate frame (odom) that takes into account the estimated position of the robot at all times (See Section II for detailed description of the position estimation). Each hit is then stored in the buffer for the duration that it takes the sonar to scan a full 360-deg circle and then thrown away.

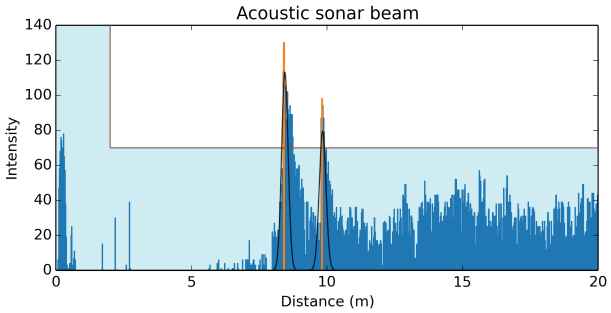



Figure 3: Shows an example of a typical beam reading from the MSIS. Here two pillars were present and can be seen by the two orange peaks in the data. The shaded area represents and example of thresholding to remove background noise and unwanted areas. The black curves represent Gaussians fit to the peak and used to place requirements on the minimum and maximum width of a peak. This futher removes spurious detections.

### C. Feature extraction

The feature extraction method assumes there are two different types of features in the environment, walls and pillars. First a naive method is used to find possible wall and pillar candidates from the buffer, random sampling consensus (RANSAC) is first used to extract possible walls and then euclidean cluster extraction is used to extract possible pillars from the remaining data. The candidate features are then fed to the SVM to do the actual the classification as wall, pillar or nothing. The result of this method can be seen in Figure 5. Both wall and pillar candidate detection are explained in more detail below. Note that some wall candidates can be later classified as pillars and visa versa.

*1) Wall candidate detection:* Walls in 3D can most often be modeled using planar segments, that means that in a 2D sonar scan they will appear as a straight line. To detect these lines in the buffer RANSAC is used, in it's simplest form using RANSAC to detect multiple lines in the buffer works in the following way.

1) For a fixed number of times two points are selected at random, a line is drawn through them and the number of points within a fixed distance $\epsilon$ from the line counted.
2) The line with the highest number of inliers is removed from the buffer.
3) Steps 1 and 2 are repeated until all lines have been found.

*2) Pillar candidate detection:* Pillars are more difficult to detect reliably than walls. This is mainly due to the size difference between walls and pillars, while walls often take up significant portion of the robot's surrounding pillars are often very small and so get few hits from the beam segmentation,

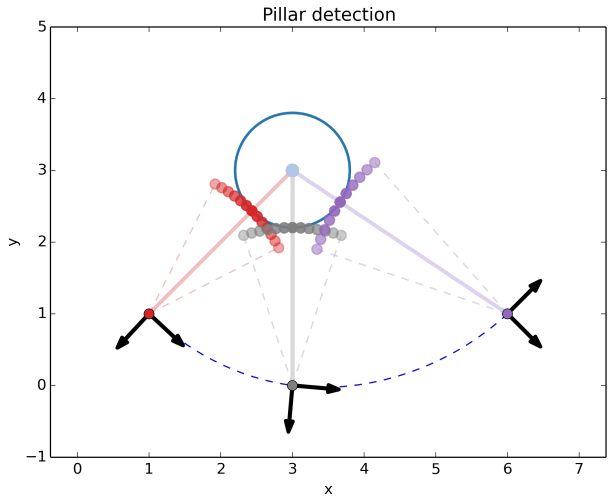



Figure 4: Shows how the detected features change and follow the outline of the pillar as the robot moves. For this reason the pillar is represented using a single feature positioned at the estimeted center of the pillar.

this makes them harder to detect and separate from noise. Another problem with detecting pillars is that the shape and location of the extracted shape. This can be seen in Figure 4 where we see that when the ROV moves around the pillar the features move along the border of the pillar. For this reason a pillar feature is best represented by a single point that estimates the center of the pillar. Another problem with detecting pillars is that the sonar is not perfect, side lobes create hits where no object exists. For this reason the extracted feature looks wider than the pillar itself and does not represent the shape of the pillar, instead the pillars look more similar to a short line segment than a pillar.

Because of the limited number of hits associated to each pillar line detection methods like RANSAC are not well suited to find them. Instead we use Euclidean Cluster Extraction to group individual hits into features. Euclidean clustering groups hits together based on proximity and is explained in Algorithm 1.

*3) Pillar Radius Model:* The final stage is a SVM classification to be described later. If the classification says that a pillar has been found then the beam with the highest intensity amoung the individual beams of the feature is projected a distance $p_r$ further away from the ROV to account for the pillar radius. This point is then used as the 'measured' center point of the pillar. This radius is assumed known here. We will show results for various values of this assumed radius.

## IV. Pillar Mapping

Here we present a method that takes in the pillar features from the scan buffer, uses ICP to align the buffer to the map and by that localizes the ROV to the map and incrementally builds up the map at the same time.

Although the ROV is traveling in a 3D space the sonar only scans a 2D plane around the ROV resulting in a scarce 3D feature set making any 3D scan matching very difficult. For this reason all the features from the buffer are first projected on to the xy-plane and only a simple 2D map of the features is maintained. This can be done since the features of interest (pillars, walls) tend to be uniform with regard to the depth, that is neither size, shape or location of the features change depending on the depth.

**Algorithm 1** Euclidean cluster extraction

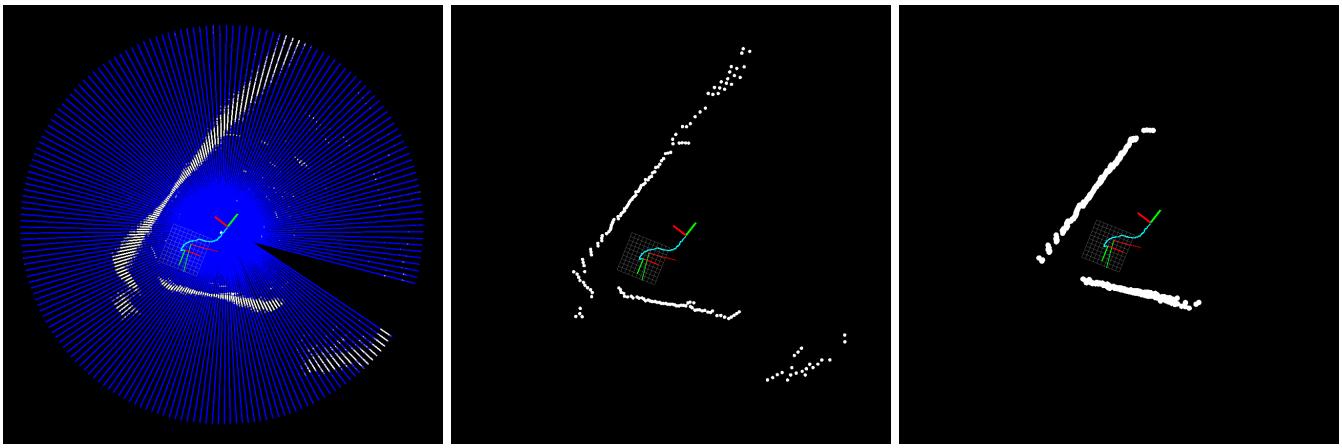
**Require:** $C \leftarrow \varnothing$, {empty list of clusters}.
**Require:** $Q \leftarrow \varnothing$, {empty list of points}.
**Require:** $P$, {list of points}.
  **for** every point $p_i \in P$ **do**
    $Q \leftarrow Q \cup p_i$.
    **for** every point $p_i \in Q$ **do**
      Find set $P_k^i$ within radius $r < d_{th}$ around $p_j$.
      **for** every point $p_i^k \in P_k^i$ **do**
        **if** $p_i^k$ hasn't been processed before **then**
          $Q \leftarrow Q \cup p_i^k$.
        **end if**
      **end for**
    **end for**
    $C \leftarrow C \cup Q$.
    $Q \leftarrow \varnothing$.
  **end for**
  **return** $C$

The map is stored as a set of point features where each feature represents the center of a pillar, each feature is associated with a position (x,y) and an intensity value representing the confidence that we have in the feature. There are two types of features in the map depending on the intensity value. A feature with intensity value higher than a threshold $\varepsilon_h$ is considered a fixed feature that can no longer be removed from the map. All other features are a so called pillar hypothesis where the intensity is updated with each new sonar scan and can be removed from the map if the intensity goes below a threshold $\varepsilon_l$.

Figure 6 gives an overview over how the mapping algorithm works. First a new scan is received from the buffer, a data association is performed between the map and the sonar scan and the best transformation between the map and the sonar data based on the data association is calculated. Finally the transformation is used to update the map and the robot position. Now each step will be described in more detail.

### A. Data Association

The data association step is a preprocessing step that selects part from both the map and the sonar data so that one point in the map is associated to one point in the sonar data. First all points in the map where the intensity is lower than a certain threshold $\epsilon_m$ are removed, this is done to reduce the chance that ICP can align to new map features created from noise or spurious measurements. Next all features further from a point in the map than a threshold $d_m$ is removed from the sonar data since it is unlikely to belong to an existing map feature.



(a) Wall data.

(b) Pillar data.

Figure 5: This figure shows how the feature extraction method works for both pillars and walls. The first image shows the raw sonar data where blue represents low intensity and white high intensity. The second image shows the result from the beam segmentation algorithm and the last image shows the results after the feature extraction.

Then we find a point to point data association between the two sets so that the total distance between all associated points is minimized, then all points that did not get associated are removed.

### B. ICP

After the data association step ICP is used to find the best transformation that minimizes the difference between the map and the sonar data.

ICP is an iterative algorithm that can essentially be broken down into the following steps.

- For each feature in the map find the closest feature in the sonar data.
- Us a mean square error cost function to estimate best transformation from the sonar data to the map.
- Transform the sonar data.
- Iterate.

The algorithm terminates either after a fixed number of iterations or if the distance between the map and sonar data becomes small enough.

If ICP is a success, that is if the difference between the two is sufficiently small after transformation the transformation is sent to the position estimation and used to update the map.

### C. Map update

Given a successful ICP alignment the map can be updated. First the original sonar scan is aligned with the map using the found transformation, then another data association is performed where one map feature is associated with one scan feature given that the distance between them is less than $d_a$. Then the map is updated using the following criteria

- The intensity for all associated pillar hypotheses is increased.
- The intensity for all unassociated pillar hypotheses where the distance to the closest sonar feature is less than $d_u$ is decreased.
- All unassociated sonar features are added to the map as pillar hypothesis with intensity $\varepsilon_l < I < \varepsilon_h$.

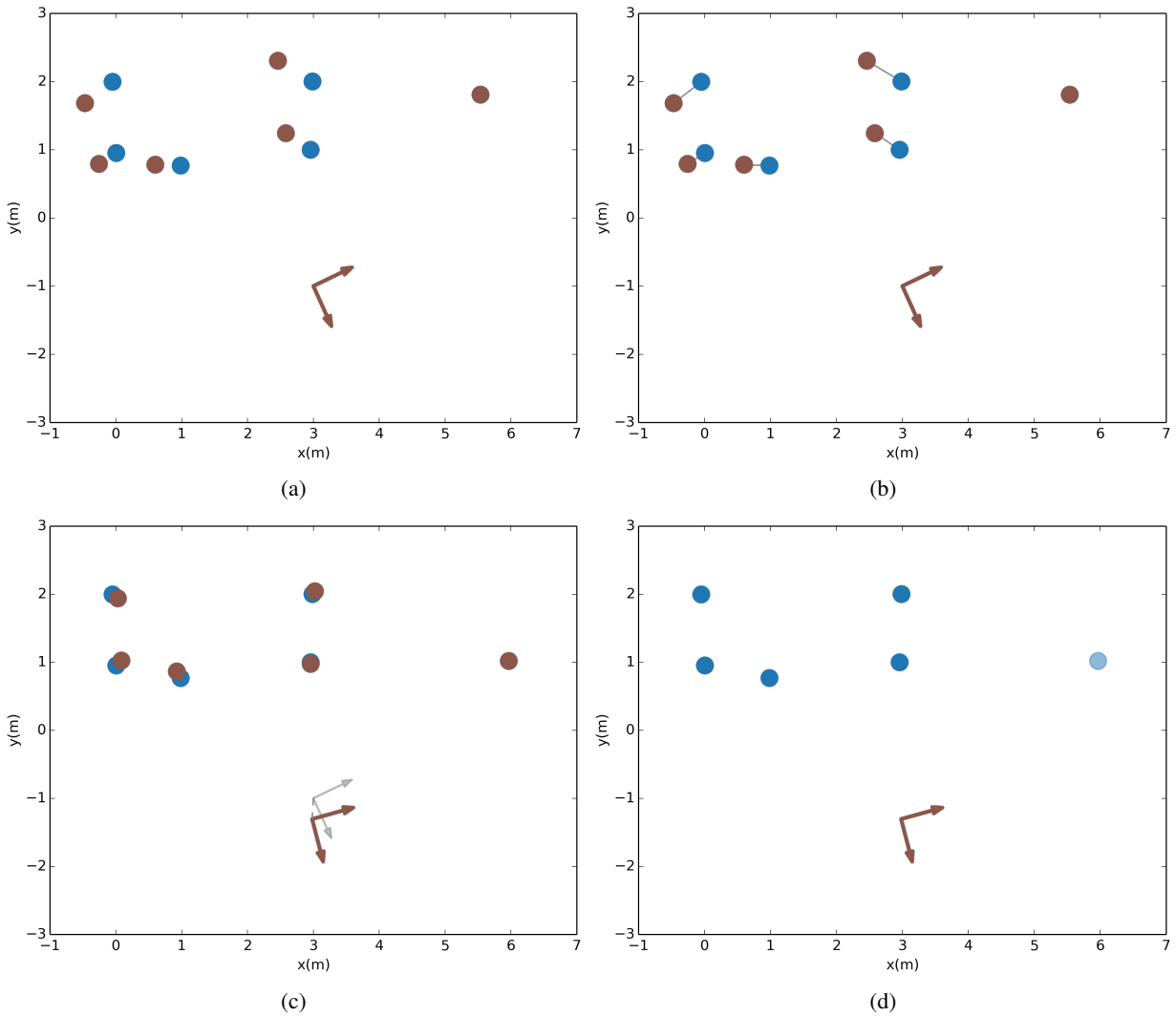



Figure 6: Shows how the mapping algorithm uses ICP to incrementally update the map. The first image shows the current map in blue and a new sonar reading in red. The second image shows the data association between the map and the sonar scan used to find the best transformation between them. The third image shows the sonar scan and the robot position after the transformation has been applied and the last image shows the updated map where a new feature has been added.

## V. Support Vector Machines

Support vector machines (SVM) is a supervised machine learning method mainly used for binary classification and regression analysis. Using a set of training examples and corresponding true labels for each training example the SVM training algorithm creates a model that classifies models into one of two classes. SVM is essentially a linear classifier separating two clusters with a simple hyperplane. Given a dataset with $N$ points

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\,|\mathbf{x}_i \in \mathcal{R}^p, y_i \in \{-1, 1\}\}_{i=1}^{N} \tag{35}$$

where $\mathbf{x}_i$ represents the input and $y_i$ its corresponding label. If the two classes are linearly separable then a hyperplane can be defined as

$$\mathbf{w} \cdot \mathbf{x} - b = 0 \tag{36}$$

with an empty margin of width $2/||\mathbf{w}||$ around the hyperplane defined by the boundaries

$$|\mathbf{w} \cdot \mathbf{x} - b\,| = 1 \tag{37}$$

The algorithm then tries to find $\mathbf{w}$ and $b$ that maximize the distance $2/||\mathbf{w}||$ and satisfy the following criteria

$$\mathbf{w} \cdot \mathbf{x_i} - b = \begin{cases} \text{-1} & \text{if } y_i \text{ is -1} \\ 1 & \text{if } y_i \text{ is } 1 \end{cases} \tag{38}$$

SVM can also perform a non-linear classification by using a so called kernel trick where the input is mapped to a higher dimensional space where it can be linearly separated. The SVM algorithm is unchanged except that every dot product is replaced by the kernel function.

The kernel functions tried in this paper are:

- Polynomial function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\gamma(\mathbf{x}_i \cdot \mathbf{x}_j) + \beta)^d \tag{39}$$

- Radial basis function (RBF):

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2\right) \quad , \gamma > 0 \tag{40}$$

In this project we have three classes (Wall, Pillar and Nothing) and thus a normal binary SVM is not sufficient. Instead we use a multi class version of SVM where one binary classifier is created for each pair of classes, then a one-versus-one strategy is employed where each binary classifier classifies the data and the class with the most wins is the final classification.

### A. SVM setup

In this paper our dataset is on the form

$$\mathcal{D} = \left\{(\mathbf{x}_i, y_i) \,|\mathbf{x}_i \in \mathcal{R}^{40}, y_i \in \{-1, 0, 1\}\right\}_{i=1}^{N} \tag{41}$$

where $x_i$ is on the form.

$$x_i = [I_1, \ldots, I_{20}, D_1, \ldots, D_{20}] \tag{42}$$

Here $I_i$ represents the intensity of a single sonar hit and $D_i$ it's corresponding distance from the ROV at the time it was first seen. If an extracted feature contains more than 20 hits the edges are removed to fit and if the feature contains less than 20 hits, hits with zero intensity and zero distance are added on both ends to fill up the vector.

Before the data can be fed to the SVM it has to be normalized, here we normalize the intensity and the distance differently. Using a training set we find the mean and standard deviation for each intensity dimension. Then for all remaining inputs the intensity is scaled by subtracting it's corresponding mean and dividing by twice the standard deviation. The distance however uses a simpler normalization where all distances $D_i$ are divided by the maximum distance $D_{max}$ for that input $x_i$.

## VI. Results

Multiple experiments were performed in order to test our method. There are two main factors that need to be tested. The first is how well the classification using SVM works and the second is to test our pillar mapping algorithm.

### A. SVM

In order to evaluate the performance of the SVM design the dataset had to be labeled by hand. This was done by taking the output from the feature extraction method without performing the SVM classification step. Each feature was then transformed into the input format described in subsection V-A and assigned a label depending on the correct classification. This was done for both datasets resulting in a dataset comprised of 305 wall features, 240 pillar features and 525 objects that are neither. This dataset was then randomized and split evenly into a training and testing dataset.

| **Linear kernel** | **Polynomial kernel** | | | **RBF kernel** |
|---|---|---|---|---|
| $C$ | $\gamma$ | $\beta$ | $p$ | $\gamma$ |
| 443 | 0.6 | 1 | 2 | 0.05 |

Table I: Shows the best parameters found for each kernel using grid search and five fold cross-validation.

| **Linear Kernel** | Nothing | Wall | Pillar |
|---|---|---|---|
| Nothing | 226 | 3 | 37 |
| Wall | 3 | 141 | 7 |
| Pillar | 33 | 1 | 86 |
| **Polynomial Kernel** | Nothing | Wall | Pillar |
| Nothing | 252 | 4 | 10 |
| Wall | 8 | 140 | 3 |
| Pillar | 34 | 1 | 85 |
| **RBF Kernel** | Nothing | Wall | Pillar |
| Nothing | 253 | 4 | 9 |
| Wall | 3 | 144 | 4 |
| Pillar | 35 | 1 | 84 |

Table II: Shows classification results using the parameters given in Table I as the confusion matrix. Each row represent the correct class and each column the classification result.

*1) Parameter tuning:* As was explained in Section V three different kernels were tried for the best possible performance, a linear kernel, RBF kernel and a polynomial kernel. The linear kernel depends on a single parameter C that controls the trade-off between classification error and maximizing the margin around the hyperplane. Similarly the polynomial kernel depends on three parameters $\gamma$, $\beta$ and $p$ and the RBF kernel depends of a single parameter $\gamma$, they can be seen in Equations 39 and 40 respectively.

In order to find the pest parameters for each kernel a simple grid search over the the parameters was performed using a five fold cross-validation as the performance measure. The best parameters found using this method for each kernel can be seen in Table I.

*2) Performance evaluation:* The result from the classification for each kernel using the parameters in Table I can be seen in Table III. Notice that the classification errors that have the most significant effect are false positives of walls and pillars. Mistaking a pillar as nothing does much less harm to the navigation estimate. So for example for the RBF kernel there were only 18 such mistakes or about $3\%$ of the total feature candidates. To avoid having these harm the estimate they would need to be subjected to further checking by matching accross time for persistance and consistancy of the detections.

We use three performance measures (Precision, Recall and Accuracy) to evaluate the classification for each individual class:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{43}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{44}$$

| **Linear Kernel** | Precision | Recall | Accuracy |
|---|---|---|---|
| Nothing | 0.86 | 0.85 | 0.89 |
| Wall | 0.93 | 0.97 | 0.97 |
| Pillar | 0.72 | 0.66 | 0.85 |
| All | - | - | 0.84 |

| **Polynomial Kernel** | Precision | Recall | Accuracy |
|---|---|---|---|
| Nothing | 0.95 | 0.86 | 0.90 |
| Wall | 0.93 | 0.97 | 0.97 |
| Pillar | 0.71 | 0.87 | 0.91 |
| All | - | - | 0.89 |

| **RBF Kernel** | Precision | Recall | Accuracy |
|---|---|---|---|
| Nothing | 0.95 | 0.87 | 0.91 |
| Wall | 0.95 | 0.97 | 0.98 |
| Pillar | 0.7 | 0.87 | 0.91 |
| All | - | - | 0.89 |

Table III: Shows Precision, Recall and Accuracy for for the three types of kernels using the parameters given in Table I.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{45}$$

Here TP, TN, FP and FN stand for true positive, true negative, false positive and false negative respectively.

The results from applying these performance measures on the data can be seen in Table III, there we can see that both the polynomial kernel and the RBF kernel are give almost identical results with $\approx 90\%$, it can also be seen that the linear kernel gives significantly worse results than the other two.

From the results it can also be seen that classification of wall features gives much higher accuracy than the other two. This is as expected since walls most often include many more hits in a single feature than a pillar and any feature extracted from pure noise is likely to be small and be much closer to a pillar in both shape and size.

### *B. Pillar mapping*

As previously said the dataset was gathered at the Stevens pier on the Hudson river. The pier is shown in Figure 2a where it can be seen that it consists of pairs of cylindrical pillars extending down into the river spaced approximately 8-9 m apart. The trajectory of the ROV consisted of multiple way-points. On each way-point the ROV was stopped to allow the sonar to scan at least two full scans, this was also repeated for three different depths on each way-point. Using this method the ROV was driven approximately 15 m along the pier and then back to the starting point stopping at multiple way-points on the way.

The results from using only the USBL positioning system and the compass for positioning and mapping every hit from the feature extraction can be seen in Figure 7. There it can clearly be seen that the data is split into four different clusters, however due to high level of noise and position errors each cluster is spread out over a large area making it impossible to say how many pillars if any belong to each cluster.

A map for the same data when using our mapping method can be seen in Figure 8. There we can see that it gives a vast improvement and that it correctly captures all eight pillars present in the data as well as the USBL stabilization system as the ninth pillar. Here we have assumed that the pillars them self are point features, that is that they have radius of 0 m.

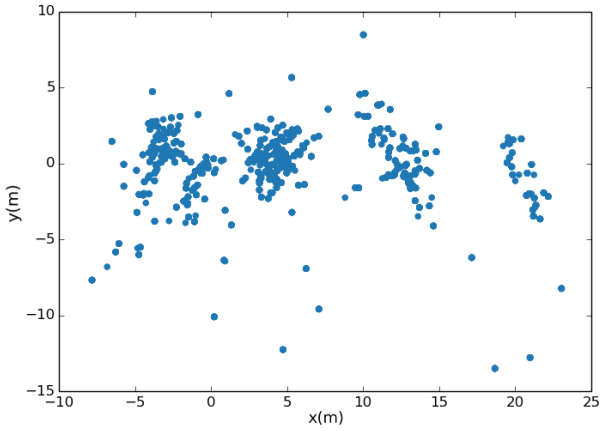



Figure 7: Shows the result when position estimation depends only on the USBL and the compass and every hit from the feature extraction is placed in the map without any modification.

In Figure 8 the estimated center of each pillar is shown with a blue dot and the 95% confidence ellipse and one standard deviation for each pillar are drawn as well. This information is also shown in Table IV displaying the number of hits belonging to each pillar, the standard deviation and the confidence ellipse. The average values are calculated using the formula

$$\alpha_{avg} = \sum_{i=0}^{N} \left( \frac{\text{Hits}_i}{\sum_{j=0}^{N} \text{Hits}_j} \cdot \alpha_i \right) \tag{46}$$

where $N$ is the number of pillars found, $\text{Hits}_i$ is the number of hits associated with the i-th pillar and $\alpha_i$ is either the RMS or the ellipses values for the i-th pillar.

When looking at the 95% confidence ellipses of the pillars we can see that the shape of the ellipses is elongated meaning that the position uncertainty depends on the direction. This can be explained when looking at Figure 4. There it can be seen that due to the sidelobes of the sonar the extracted feature looks wider than it is, this means that extracting the correct center of the feature is more difficult than finding the correct distance to it. This means that the uncertainty is higher along the travel direction of the ROV as can be seen in Figure 8.

### *C. Pillar radius*

As has been mentioned earlier pillars are best represented using their center point, in order to do that the radius of the pillar has to be known. Figure 9 shows the result from the mapping algorithm using different estimated pillar radius. Table V then compares these results by showing the average results. There it can be seen that although a small difference

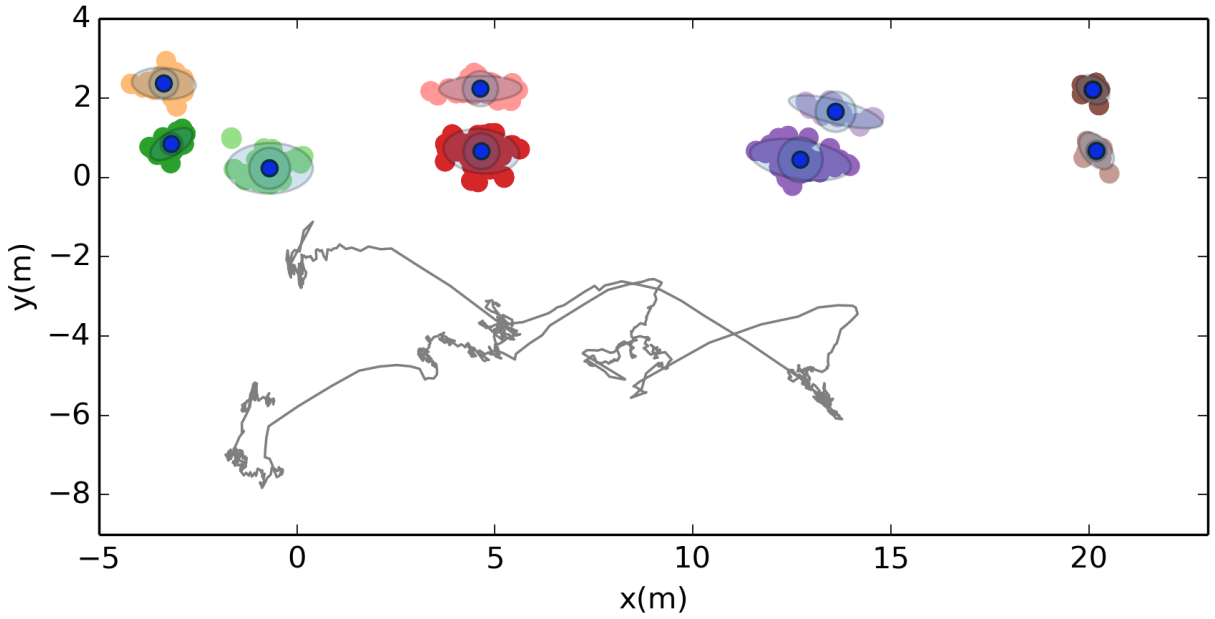



Figure 8: Shows the result from the mapping algorithm on the pillar dataset using pillar radius of 0 m. The gray line shows the robot path.

| Pillar | Hits(#) | RMS(m) | Ellipse x(m) | Ellipse y(m) |
|---|---|---|---|---|
| 1 | 127 | 0.37 | 1.62 | 0.79 |
| 2 | 148 | 0.27 | 1.21 | 0.58 |
| 3 | 215 | 0.52 | 2.19 | 1.28 |
| 4 | 310 | 0.45 | 1.92 | 1.11 |
| 5 | 231 | 0.45 | 2.09 | 0.64 |
| 6 | 184 | 0.56 | 2.57 | 0.99 |
| 7 | 64 | 0.51 | 2.43 | 0.6 |
| 8 | 31 | 0.22 | 0.89 | 0.66 |
| 9 | 38 | 0.26 | 1.08 | 0.68 |
| Average | 149.78 | 0.44 | 1.95 | 0.91 |

Table IV: Shows the result from the mapping algorithm using pillar radius of 0 m.

| Radius(m) | Hits(#) | RMS(m) | Ellipse x(m) | Ellipse y(m) |
|---|---|---|---|---|
| 0.0 | 149.78 | 0.44 | 1.95 | 0.91 |
| 0.1 | 155.0 | 0.44 | 1.91 | 0.94 |
| 0.2 | 154.0 | 0.43 | 1.9 | 0.89 |
| 0.3 | 151.56 | 0.43 | 1.89 | 0.9 |
| 0.4 | 147.78 | 0.44 | 1.94 | 0.93 |
| 0.5 | 157.67 | 0.45 | 1.99 | 0.94 |

Table V: Shows the average results from the mapping algorithm using using different estimated pillar radius.

the uncertainty in the pillar estimation decreases as the radius reaches the true radius of the pillar between 10 and 20 cm and the average number of hits increases.

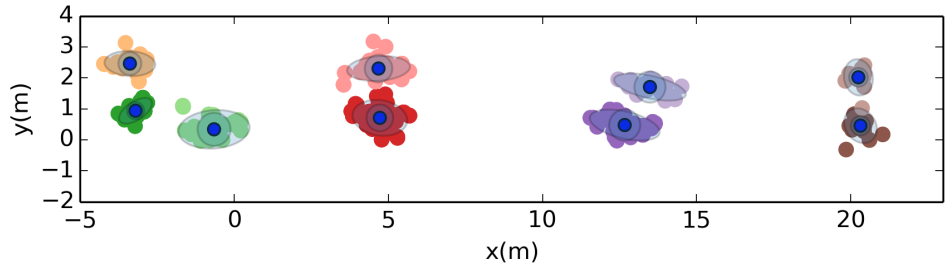
A bigger difference can be seen in Figure 9, there it can be seen that as the estimated pillar radius increases the distance between pillars increases. This means that choosing a radius to small results in a compressed and incorrect map and similarly choosing a radius to large results in an expanded map making correct pillar radius vital for accurate map.

The most significant comparison can be made by noting that although we do not have the exact dimensions of the pier we can reasonably assume that the eight pillars form a regular equal spaced grid and all have the same radius. By least square fitting the centers to a grid we can compare the mapped location to that grid. In doing so we discard the third pillar in the back row as this was not very well observed as can be seen in the resulting maps. We also do not consider the USBL beacon seen near (0,0) as it is not part of the pier. The other pillars do line up well but only when the radius is chosen correctly. The results are shown in Table VI. Compared to Table V which has two parameters per pillar, the model in Table VI estimates only five parameters for the whole pier, the pillar spacing along and across the pier, the angle of the pier to due east, and the coordinates of one of the pillars.

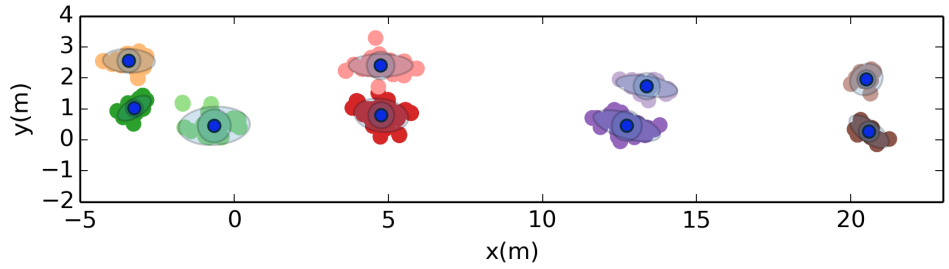
The ratio of the unknown pillar radius to the width between two pillars can be bounded by examining the photo of the pier in Figure 2. This gives

$$0.095 < R/Width < 0.127 \tag{47}$$

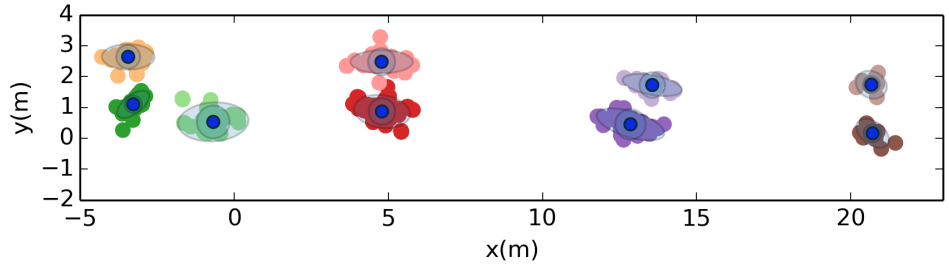
We therefore show the fitted values for the width in Table VI. As you can see the best linearity and overall error corresponds to the values for R/Width close to this bound for the 0.2 m radius model. This allows us to conclude that the radius does make a significant difference in the accuracy of the map. To directly estimate the radius using this data is not possible due to the noise and uncertainty in ROV pose, but if



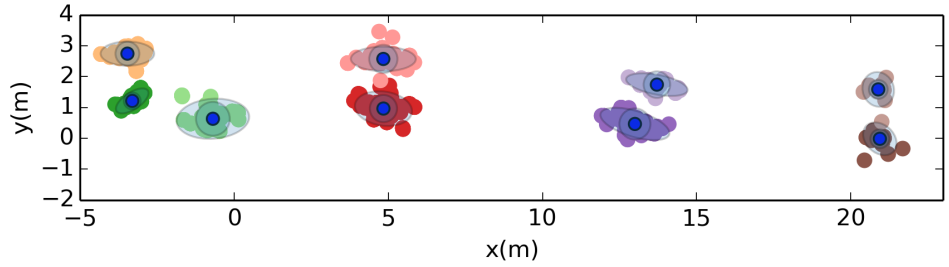

(a) Pillar radius 0.1 m.



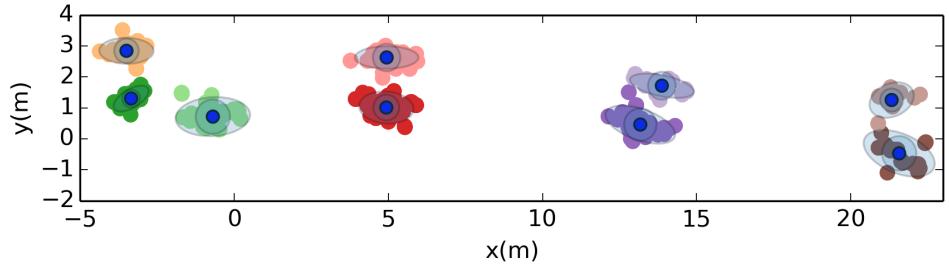

(b) Pillar radius 0.2 m.




(c) Pillar radius 0.3 m.




(d) Pillar radius 0.4 m.




(e) Pillar radius 0.5 m.

Figure 9: Shows the the result from the mapping algorithm when varying the pillar radius. One can see that the pillars do not line up as well for the radius of 0.5 m but rather seem to follow a slight arc from left to right. This is measured as the linearity in Table VI

| Radius(m) | Width (m) | R/Width | RMS Error (m) | Linearity (m) |
|---|---|---|---|---|
| 0.0 | 1.599 | 0.000 | 0.455 | 0.07262 |
| 0.1 | 1.588 | 0.063 | 0.430 | 0.03714 |
| **0.2** | **1.613** | **0.124** | **0.423** | **0.03708** |
| 0.3 | 1.602 | 0.187 | 0.427 | 0.05902 |
| 0.4 | 1.594 | 0.251 | 0.442 | 0.08737 |
| 0.5 | 1.580 | 0.316 | 0.465 | 0.14458 |

Table VI: Results for a least square fit to a 5 parameter model of the pier as a regular rectangular grid of 7 pillars with a fixed radius. The rectangle 'width' is the shortest distance between two pillar centers. Although both the radius and the width are unknown the ratio can be bounded by examining the lower photo in Figure 2. The value consistent with this bound was found by fitting data with a 0.2 m radius model as shown in bold above. The RMS error is over all measurements relative to the fitted grid center positions. The quality of the map is checked by how well the 7 pillar centers line up on the straight lines. The 'linearity' is the RMS error perpendicular to the line of the pier of the 7 centers weighted by the number of measurements of each pillar. Maps where the pillar centers lie along two parallel lines will give low (good) values for 'linearity'.

one has some prior knowledge it can help mapping and motion estimation.

## VII. Conclusion and further work

In this paper we have presented methods for remotely operated underwater vehicles (ROVs) traveling in man-made underwater environments. The main contribution in this paper is a feature extraction and classification method using Support Vector Machines (SVM) as the final classification step allowing a SLAM or localization algorithm to differentiate between walls, pillars and nothing.

Furthermore building on top of the detection method a second contribution is showing the effect of model parameters of the features and map on the accuracy of mapping and localization algorithms. Not surprisingly we show that the radius of pillars in the model effect the accuracy of the map and motion estimates base on the map. Knowing information such as regular repeated patterns in the landmark positions can aid in navigation.

For further work there are many opportunities. Although the feature extraction method gave good results and the SVM classification shows accuracy of up to $90\%$ further testing is still needed preferably using a single dataset where both pillar and wall features are present on the same time. Similarly extending the mapping algorithm to include more types of features such as walls will improve the accuracy of the map and allow the ROV to navigate in more dynamic area. Currently the method assumes that all pillar have the same radius and that radius needs to be known beforehand, this is seldom the case and thus creating a method to automatically optimize the radius of each pillar would be a vast improve over the current method.

## VIII. Acknowledgement

We would like to thank Brendan Eglot at Stevens Institute of technology for allowing us to collect data using his facilities. The work presented in this papers has been funded by the Swedish Defense Material Administration (FMV) through the Cluster for Underwater Technology (CUTe) at KTH. The funding is gratefully acknowledged.

## References

[1] P. Newman and J. Leonard, "Pure range-only sub-sea slam," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1921–1926.

[2] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[3] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in *Springer handbook of robotics*. Springer, 2008, pp. 871–889.

[4] S. B. Williams, O. Pizarro, J. M. Webster, R. J. Beaman, I. Mahon, M. Johnson-Roberson, and T. C. Bridge, "Autonomous underwater vehicle–assisted surveying of drowned reefs on the shelf edge of the great barrier reef, australia," *Journal of Field Robotics*, vol. 27, no. 5, pp. 675–697, 2010.

[5] A. Kim and R. Eustice, "Pose-graph visual slam with geometric model selection for autonomous underwater ship hull inspection," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1559–1565.

[6] A. Mallios, P. Ridao, E. Hernández, D. Ribas, F. Maurelli, and Y. Petillot, "Pose-based slam with probabilistic scan matching algorithm using a mechanical scanned imaging sonar," in *OCEANS 2009-EUROPE*. IEEE, 2009, pp. 1–6.

[7] D. Ribas, P. Ridao, J. D. Tardós, and J. Neira, "Underwater slam in man-made structured environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 898–921, 2008. [Online]. Available: http://dx.doi.org/10.1002/rob.20249