ElevateNet: A Convolutional Neural Network for Estimating the Missing Dimension in 2D Underwater Sonar Images

Robert DeBortoli, Fuxin Li, Geoffrey A. Hollinger

***Abstract*— In this work we address the challenge of predicting the missing dimension (elevation angle) from 2D underwater sonar images. The high noise levels in these images, from phenomena such as non-diffuse reflections, frequently limits the usefulness of physical models. We thus propose the utilization of Convolutional Neural Networks (CNNs) as a powerful method to extract meaningful information without being misled by noisy data. We also introduce a self-supervised method that uses the physics of the sonar sensor to train the network on real data without ground-truth elevation maps. Our method can produce accurate elevation angle estimates given only a single image. Finally, we demonstrate that our method produces more accurate 3D reconstructions than competing methods, both in simulation and on real data.**

I. INTRODUCTION

3D reconstructions of marine environments can provide rich information to human operators in tasks such as ship hull inspection [1], seafloor mapping [2], and pipe inspection [3]. Current methods for obtaining these reconstructions oftentimes involve the use of divers with optical cameras, which is a slow and potentially dangerous process. Robotic underwater vehicles have the potential to obtain these reconstructions in a safer and more time-efficient manner. Due to the turbidity of water, the preferred sensing modality in such applications is often sonar (visibility $\sim$100 m) as opposed to optical cameras (visibility $\sim$1 m) [4].

3D sonars are available to generate these reconstructions, but they are often much more expensive and slower to operate than alternative sensors, in addition to being too heavy to mount on a research-class Remotely Operated Vehicle (ROV) [4]. This limits their application in rapid mapping applications. As a solution to these drawbacks, 2D imaging sonars are relatively inexpensive, provide images at tens of hertz, and can be mounted on small ROVs.

In spite of these advantages, the use of 2D imaging sonars for 3D reconstruction has been limited due to difficulties in recovering the missing dimension, the *elevation angle*, from 2D images. These difficulties often stem from the poor signal-to-noise ratio in these images, which is often much lower than standard camera imagery. This makes the development of an inverse sensor model and tasks such as feature association very difficult [4], [5]. There has been a variety of previous work attempting to estimate the elevation angle from these images, including using priors in the environment such as the shadow of the object on the seafloor [6] or moving the vehicle in specific ways around the object to reduce ambiguity in the elevation [7], [8]. While these methods provide high-quality reconstructions in specific applications, they do not cover the case where the object is in the water column (and no shadow is produced) and the vehicle can only traverse around the object in a constrained manner, as would be the case in a cluttered environment.

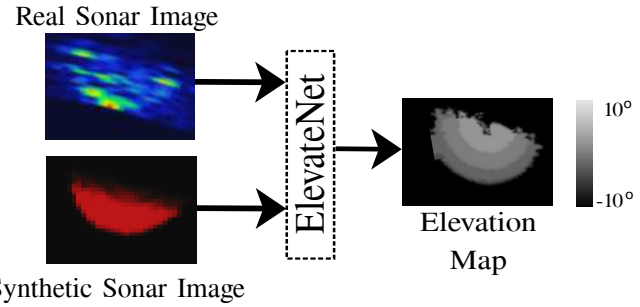


Fig. 1: Inputs and outputs for ElevateNet. ElevateNet takes in 2D sonar images (left) and produces pixelwise estimates of the elevation angle. The network can be trained in a fully supervised manner when ground truth information is available, as is the case with synthetic data. Our method can also train in a self-supervised manner on real sonar images, where ground truth information is not available.

These drawbacks constrain the speed of producing 3D maps and thus limit the usefulness of sonar-equipped underwater vehicles. Our method is a first step towards a more flexible approach to providing dense reconstructions, to enable "fly-through" missions where a robot only has to traverse quickly through the environment to construct a 3D map.

In contrast to the difficulties in underwater applications, terrestrial applications have seen an abundance of methods for obtaining the missing depth dimension from 2D RGB images [9], [10]. In many recent works, Convolutional Neural Networks (CNNs) have proven to be a powerful method to obtain high-quality depth estimates. These methods can even be trained in a *self-supervised* manner, meaning they do not require a ground truth estimate of depth in order to be trained [10]. This is an attractive property for our application, where ground truth information is often unavailable. Self-supervision in terrestrial applications is done by using one image of a scene to generate a depth estimate. This estimate is then used to produce an image of the scene from a different viewpoint [10]. An actual image of that scene from that

This research has been funded in part by U.S. Naval Facilities Engineering Command (NAVFAC) contract N0002418F8702 and the National Science Foundation under award #1751402.

The authors are with the Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis, Oregon 97331 {debortor, fuxin.li, geoff.hollinger}@oregonstate.edu

different viewpoint can then be used to compute an error and train the network. The intuition is that if the depth estimate from the first frame is accurate, the predicted image will also be accurate, and vice-versa.

To enable self-supervision, many RGB-based methods rely on well-established theory, such as pixel disparities and the stereo vision model, to stabilize the training of the network. Such theory does not exist in a general form for underwater sonar images, and thus we propose an approach that does not require these sensor properties for model training.

In this work, we propose a self-supervised CNN for generating estimates of the elevation angle in 2D sonar images. In contrast to previous approaches for estimating the elevation angle, which rely on sensitive physical models or many views taken at specific poses, we use a CNN to obtain a higher-level representation of the image and avoid being misled by large amounts of noise.

To train our network, we present a self-supervised method that uses real sonar data. To do this, we leverage the physics of the sensor to predict the next frame and thus enable the training process. Before training on real data we pre-train the model using synthetic data, which provides two benefits. First, it allows us to train a network using only a small amount of real data, which is often scarce for underwater sonar imagery. Second, because this is an under-constrained problem (multiple estimates can lead to the same image), training on synthetic data puts the CNN in an appropriate part of the parameter space. Fine tuning is then done using real data.

To our knowledge, ours is the first method for training a network on real underwater sonar data to predict the pixelwise elevation angle. As we show, by allowing the network to train on real data, it is able to better account for the noise in real imagery, thus improving 3D reconstruction ability in the real world. Fig. 1 shows an overview of our method.

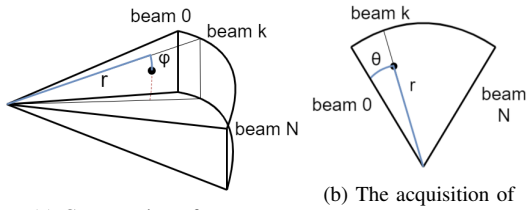
In short, our contributions are as follows:

- A novel CNN for generating dense reconstructions of underwater environments from a single sonar image. Our approach does not require the object to be in a specific part of the environment or specific traversals around the object.
- A self-supervised training scheme that uses the physics of the sonar sensor to enable training on real data without ground truth elevation angle information.

II. RELATED WORK

A. 2D Imaging Sonars

To inform our discussion of related work, we first describe the imaging sonar model shown in Fig. 2. At a high level, 2D imaging sonars operate by sending out acoustic pulses and measuring both the time of flight and intensity of the returned acoustic energy. From the time-of-flight, a range r to the reflector can be obtained (Fig. 2a). From the amount of acoustic energy returned, the pixel value (from 0-255) can be obtained. Bearing to the target is obtained by sending out a series of acoustic pulses along a horizontal swath (Fig. 2b). Thus, the dimensions of a 2D sonar image are range r and bearing θ. Note that the elevation angle ϕ is lost in the imaging process. The purpose of this work is to recover this missing dimension.



(a) Computation of range and elevation angle

(b) The acquisition of bearing by emitting multiple beams

Fig. 2: Mapping from Euclidean to polar coordinates.

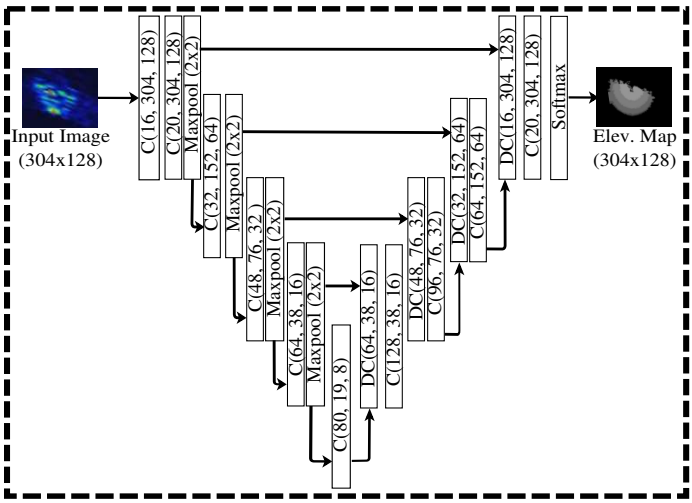
B. Estimating the elevation angle

Previous work on recovering the missing elevation angle from 2D sonar images can be grouped roughly into three categories: feature-based methods, single-view physics-based methods, and multi-view methods.

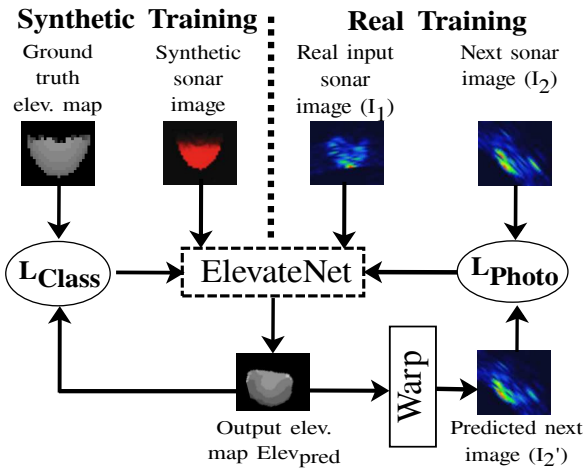
Feature-based methods often use the tracking of image features across multiple frames to estimate the elevation angle. Due to the noise present in these images, feature annotation is often done by hand [11], [12]. Alternatively, Westman et al. use A-KAZE features to identify high-quality image features and automate this process [5]. While these methods produce high-quality results, they are constrained to producing sparse reconstructions. In this work we complete a pixelwise estimation, thus generating dense reconstructions.

Using a physical model of the sonar and determining the elevation angle from the intensity values returned has also been a popular option in previous work [2], [6]. For example, Aykin and Negahdaripour use shape-from-shading to predict the elevation angle of objects on the seafloor [6]. While this method produces promising results for objects on the seafloor, the reconstruction approach requires the object to display a visible shadow on the seafloor for initializing the optimization. Additionally, this and other physics-based methods that attempt to determine the elevation angle from pixel values are often sensitive to noise in the sonar images [6]. To avoid this sensitivity, our method uses a CNN to gain a higher-level representation of the object.

Multi-view methods for recovering the elevation angle often involve the development of a framework that leverages a specific trajectory of the vehicle around the object. Guerneve et al. develop a method that relies on the vehicle moving in the direction of ambiguity (in the z-direction) [8]. Aykin and Negahdaripour use a state-of-the-art space carving method to leverage roll of the vehicle at various points around the object and generate a 3D reconstruction [7]. At each pose, points that do not give returns in image space are "carved" away from the final object mesh. As we will show, such multi-view methods do not perform well on cases where only part of the object is seen, as is the case for applications that require rapid mapping.



(a) Learned part of ElevateNet. C = Convolution/Batch normalization/ReLU layer. DC = Deconvolutional layers. C(a,b,c) are number of filters, output image width, output image height respectively for a convolutional layer.



(b) Training methods for ElevateNet. Training on synthetic data (left loop) can be done by directly comparing the output with the ground truth elevation map. Training on real data (right loop) uses the output to predict the next image and a photometric loss is applied between the prediction and the actual next sonar image.

Fig. 3: Our overall model architecture. (a) shows detailed model parameters of ElevateNet which can be used directly on synthetic data. (b) presents the two training methods used (on synthetic and real data, respectively). Note that the dotted box in both Fig. (a) and (b) represents the same model.

C. Using CNNs to estimate missing dimension

To our knowledge, ours is the first CNN used for directly estimating the missing dimension in 2D sonar images. Therefore, we will review terrestrial applications in this section. There are two broad groups of methods for training these CNNs: supervised and self-supervised training schemes.

Training CNNs to predict depth from 2D RGB images in a supervised manner has received much research attention [9], [13]. For example, Eigen et al. use a Kinect sensor to obtain ground truth depth maps, which are used to train a CNN that achieves excellent estimation performance [9]. Given that there is no such 3D sensor widely available for underwater vehicles, we are interested in methods that do not require a ground-truth sensor to be used.

Self-supervised methods for training CNNs do not require a 3D sensor for training [10], [14]. Instead, given an initial image, such methods estimate the pixelwise depth, producing a predicted point cloud. A virtual image is then taken of the estimated point cloud at a different pose. This virtual image can then be compared to an actual image at that different pose and the pixelwise difference between the two can be used for training [10]. To take this virtual image, bilinear sampling is used, which after defining sub-gradients is differentiable.

III. METHODS

In this section we first discuss our architecture, describe how we train on synthetic data, and finally outline an extension for training with real data. For notation, we denote the learned part of the CNN (Fig. 3a) as the $model$ and the process by which to incorporate real data (Fig. 3b) as the $architecture$.

A. Model

We first discuss the model, shown in Fig. 3a. Our network takes in a 2D sonar image and outputs an elevation angle estimate for every pixel in the image. To facilitate training, we treat the estimation as a classification and not a regression problem. Thus, we discretize the elevation angle into 20 bins over the range from $-10°$ to $10°$. Because we are completing a pixelwise classification, we adopt a network architecture similar to the U-Net CNN for pixelwise prediction tasks [15].

At this point, the model can be trained on synthetic data where we have ground truth information. However, because synthetic data does not capture all sources of noise, it would be beneficial to train using real sonar images as well.

B. Architecture

To incorporate real data into the training process, we present a method that predicts the next frame given the current frame and the estimated elevation map. This process is shown in the right loop of Fig. 3b. To start, the output elevation map from the model ($Elev_{pred}$) is a polar point cloud with coordinates (R, θ, ϕ) or range, bearing, elevation angle respectively. We convert this into Cartesian coordinates using the transformation for a polar point (R, θ, ϕ):

$$X_c = R\cos\phi\sin\theta, \tag{1}$$

$$Y_c = R\cos\phi\cos\theta, \tag{2}$$

$$Z_c = R\sin\phi, \tag{3}$$

where (X_c, Y_c, Z_c) are the Cartesian coordinates of the point, R is the range to the point from the sonar, ϕ is the elevation angle to the point, and θ is the bearing to the point.

We denote the resulting point cloud as P_{est}. We then take a simulated sonar image of P_{est} from the reference frame of the next image I_2 (also called image warping). This produces a predicted image I_2'. To produce the simulated sonar image we use the differentiable bilinear sampling module by Jaderberg et al. [16].

This predicted image can then be compared pixelwise to the actual image I_2. This difference is called the *photometric loss* and can be used to train the model. The intuition behind this is that an accurate estimate of the elevation angle will lead to an accurate estimate of the next frame and vice versa. In this manner we embed the physics of the sonar sensor into the training process.

C. Training with synthetic data

We first train our model with synthetic data because we have the ability to generate large amounts of training data, and the ground truth elevation maps provide a clear training signal. We used a weighted cross entropy loss function (as implemented in PyTorch [17]) where we weight pixels with sonar returns 5x more than those without returns. We term this the L_{Class} loss. Because the ground truth elevation map is such a strong signal, we did not find self-supervision to help when training on synthetic data.

To choose the best model based on synthetic data, we complete a parameter sweep over the number of layers, filter size, number of filters, learning rate, and batch size. We found the number of elevation bins did not affect performance greatly. We trained each model in the sweep for 120 epochs. We found a batch size of 9, a filter size of 5x5, and a learning rate of 1E-03 to achieve the lowest validation loss. The final model and its parameters can be seen in Fig. 3a.

D. Training with real data

After training with synthetic data, the model can be fine-tuned on real data, which improves its performance by adjusting to realistic noise. By integrating the physics of the sonar sensor into the training process, we are able to train without ground truth elevation maps.

For training on real data, we use a photometric loss, defined as:

$$Loss_{photometric} = \sum_{i=0}^{Y} \sum_{j=0}^{X} M \left| I_2'[i,j] - I_2[i,j] \right|, \qquad (4)$$

where M is a scaling coefficient (to be on the same order as the synthetic loss) and X and Y are the image width and height respectively. We found M = 14 to bring the reconstruction loss to the same order as the synthetic loss. This loss function allows for training in a self-supervised manner, which is necessary for real data where we do not have ground truth elevation angle information.

IV. EXPERIMENTS AND RESULTS

In this section we first present the overall robotic system used in this work. We then demonstrate that our approach produces more accurate elevation angle estimates and 3D reconstructions than baseline methods in simulation. Finally, we demonstrate that training on real-world underwater dataset enables more accurate 3D reconstructions on real data.

In all of our experiments we test the approaches in runs where only part of the object is visible. This is motivated by applications where the vehicle must operate in a cluttered environment, or is only completing a "fly-through" mapping mission. Therefore, the reconstructions estimated are over the part of the object actually seen, not the entire object.

Fig. 4: The Seabotix vLBV300 and Gemini 720i imaging sonar (white rectangle in the lower right) used in experiments.

A. System Overview

The vehicle used in this work is a Seabotix vLBV300 Remotely Operated Vehicle (Fig. 4). The vehicle is tethered, allowing for real-time data to be streamed back to a basestation. To estimate the 6D pose of the vehicle, it is equipped with a Doppler Velocity Logger (DVL) and Inertial Measurement Unit (IMU) which are fused together by a Greensea Inertial Navigation System (INS) [1].

Onboard the vehicle is a Tritech Gemini 720i imaging sonar which operates at 720kHz and uses 256 beams to sweep a 120° swath in the bearing direction. Images are collected at 20 Hz. The range-resolution is range dependent but is roughly 8 mm. Images we collected had on the order of 350 pixels in the range direction. These parameters are roughly equivalent to those found in sonars used in previous work, including the SoundMetrics DIDSON, BlueView M900-90, and the Aris Explorer 3000 [18], [19], [20].

B. Performance on synthetic data

We first describe the generation of our synthetic dataset and then show that we estimate the pixelwise elevation angle accurately. We then validate that these accurate estimations lead to more accurate 3D reconstructions.

1) Synthetic dataset: We generate synthetic data using the simulator by by Cerqueira et al. which models the same Tritech Gemini 720i sonar on our vehicle [21]. We generate a dataset of 8454 synthetic images of spheres, cylinders, and cubes which are often the shapes of underwater objects of interest (e.g. mines, pipes, etc.) [22], [23]. The sphere has a radius of 19 cm, the cylinder has a radius of 5 cm and height 10 cm, and the cube a side length of 30 cm.

We take care to avoid overfitting in our training set by imaging objects at a variety of poses and from a variety of ranges. We also take images with varying maximum ranges (from 1-5 meters).

[1] https://greenseainc.com/products/ins

TABLE I: Synthetic Pixelwise Elevation Angle MAE (m)

Min. dist. to training ex. (cm)	Entire test set			Sphere			Cube			Cylinder		
	<10	10-20	20-30	<10	10-20	20-30	<10	10-20	20-30	<10	10-20	20-30
$Pred_{random}$	0.160	0.170	0.180	0.200	0.190	0.183	0.213	0.211	0.206	0.0673	0.068	0.187
$Pred_{zero}$	0.114	0.112	**0.120**	0.140	0.120	**0.114**	0.145	0.135	0.135	**0.0527**	**0.0545**	**0.146**
CNN (ours)	**0.087**	**0.105**	**0.120**	**0.100**	**0.110**	0.126	**0.105**	**0.109**	**0.105**	0.0635	0.0657	0.186

From the simulator we get images of size 600x256 pixels. Images are downsampled to 304x128 pixels and normalized before being fed into the network. Images are normalized according to:

$$I_n = (I - \mu)/\sigma, \tag{5}$$

where I_n is the normalized image, I is the original image and μ and σ are the mean and standard deviation of the dataset respectively.

We break up the dataset into 75% for training, 15% for validation and 10% for test. We ensure that the test data contains images of all three shapes (sphere, cylinder, cube). The network is then trained according to the procedure described in Section III-C.

2) Evaluating pixelwise error: We first show that our method can produce more accurate elevation maps than competing methods. We compare against two baselines. The first, $Pred_{random}$ predicts a random elevation angle for every pixel. This baseline gives a sense of the overall amount of ambiguity in the elevation dimension. The next baseline is $Pred_{zero}$, which predicts an elevation angle equal to zero for all of the pixels in the image. We do not compare against physics-based methods because oftentimes they require an environmental prior (such as the shadow of an object lying on the seafloor) and thus are not applicable to the general case of an object in the water column [6].

For our performance metric we use the Mean Absolute Error (MAE) in elevation angle estimation. To give context to this error, we convert the error into meters, thus resulting in the final reported metric, defined as:

$$MAE = \frac{1}{|N|} \sum_{n \in N} |r_n \sin(\phi_{p,n} - \phi_{t,n})|, \tag{6}$$

where N is the set of pixels whose elevation angles are estimated and r_n, $\phi_{p,n}$, and $\phi_{t,n}$ are the range, predicted elevation angle, and ground truth elevation angle to point n.

To give insight towards the ability for our method to extend to views unseen, we compare each method across subsets of data, each with vehicle poses at varying amounts of distance away from the training set. Specifically, for each shape we generate 3 subsets: one with poses greater than 0 and $<$ 0.1 m away from the closest training example, one with poses in the range 0.1-0.2 m to the closest training example, and finally a set with poses 0.2-0.3 m away from the closest training example.

The results can be seen in Table I. Over the entire test set, our method performs well, reaching errors below 0.1 m when the poses are near training data. Naturally, as the poses become increasingly further away from the training set, our method performs worse, but still achieves results competitive with or better than baselines.

For the sphere, our method performs well when the test poses are close to the training data. Interestingly, it maintains its performance even up to 0.2 m away from any training pose. As the test data gets far away from the training data, $Pred_{zero}$ performs better, suggesting that we are starting to observe the object closer to the ϕ = 0 plane.

Our low error on the cube demonstrates the ability for our method to perform well on simple objects. Even as we move into test views that are far away from any given training view (up to 0.3 m difference), our method still outperforms competing methods by a large margin.

Finally, we find that our method is unable to beat the $Pred_{zero}$ baseline on the cylinder. Further analysis suggests that the network appears to confuse the cylinder with a cube and thus predicts a plane instead of a curved surface. The reasoning for this could be that the cylinder and cube objects (Figs. 5e and 5f) look very similar in image space.

3) Synthetic 3D Reconstruction Results: We next demonstrate that our method can produce high-quality reconstructions using the estimated elevation maps. We compare our method and the $Pred_{zero}$ baseline on their 3D reconstruction abilities in a manner similar to previous work [7]. For brevity, we do not include results from $Pred_{random}$.

In this experiment we also add the aforementioned state-of-the-art space carving method proposed by Aykin and Negahdaripour [7]. While this method relies on multiple views, it is to our knowledge the only method for dense reconstruction that does not require an environmental prior (e.g. shadow on the seafloor) to compute a reconstruction. To ensure this method is working on our data, we test it on images of the sphere taken from a variety of views and roll angles, to ensure convergence. We note that this set of viewpoints is not indicative of the rapid mapping scenario targeted in our experiments. We achieve an RMSE of 0.021 m, indicating a working method.

Because a single view does not offer much coverage over the entire object, we use 20 views of the object to generate a mesh for each method. We note that entire coverage of the object may not still be attained and thus the generated meshes are only over a portion of the ground truth objects. This highlights the ability of our method to generate estimates even when a diverse set of views are not possible.

We first compare the methods qualitatively. The output of $Pred_{zero}$ and our approach is a 3D pointcloud with dimensions range, bearing, elevation. To generate meshes from pixel based estimates we use the standard Matlab trimesh function. The results can be seen in Figs. 5g-5o. Across all three shapes, the space carving method is unable

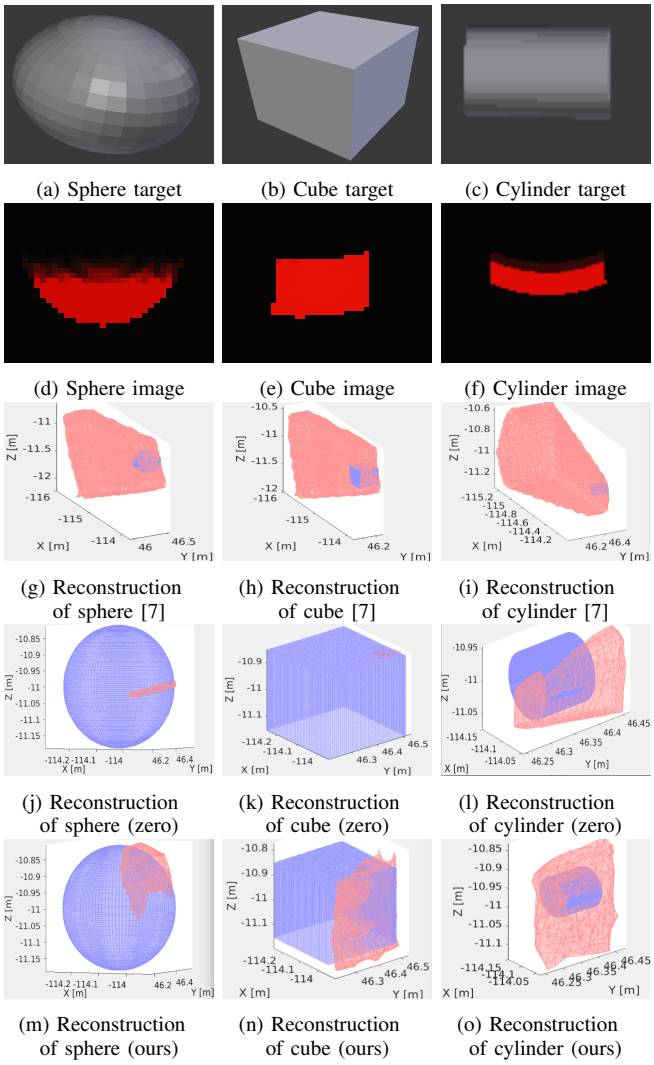


Fig. 5: The synthetic sonar targets (a)-(c), their respective synthetic sonar images(d)-(f), and the reconstructed meshes(g)-(o) from a small subset of images. For Figs. 5g-5o, the ground truth meshes are blue and the predicted meshes are red. The predicted mesh in Fig. 5k is small, but is in the upper right corner of the image. In these experiments, only part of the ground truth objects are seen and thus the predicted meshes should only cover one part of the object. For the sphere and cube, our method is uniquely able to capture the shape of the object being imaged. The CNN confuses the cylinder for a cube and thus predicts a vertical plane. Best viewed in color.

to see the entire object, and thus there is a lot of ambiguity in the region behind the object (which is un-observable from a limited number of views). $Pred_{zero}$ is only able to predict planar meshes and thus for the sphere and cube especially, the resulting reconstructions bear no resemblance to the ground truth shape. For the cylinder, some views may have been at different z-coordinates and thus the mesh produced by $Pred_{zero}$ was built over planes of varying height.

Our method does well on the sphere and cube estimation, producing meshes that lie on the ground truth surface. For the cylinder, we see results that agree with the elevation angle prediction results previously mentioned. Due to the similarity of the cube and cylinder in image space, the CNN most likely predicts the cylinder to be a cube and thus predicts a vertical plane.

TABLE II: Synthetic Reconstruction RMSE Results (m)

	Sphere	Cube	Cylinder
Space carving method [7]	1.09	1.05	0.815
$Pred_{random}$	0.115	0.160	0.070
CNN (ours)	**0.022**	**0.020**	**0.068**

To compare each method quantitatively, we use the Root Mean Square Error (RMSE) between the two point clouds. To compute point correspondences we use a nearest-neighbor search and then compute the error between the matched points. Because we have the ground truth object pose, we do not apply any affine transformation to any point cloud in this section. Our results can be seen in Table II. As can be seen in Figs. 5j-5l, $Pred_{zero}$ only is able to produce planar estimations and does not generate accurate estimations. Therefore, we do not include it in the quantitative comparisons.

Intuitively, because our method can predict the shapes of the sphere and cube well, it scores best on that object. The space carving method is unable to see the object from many viewpoints, and thus a large amount of ambiguity in its estimation remains. For the cylinder, the results agree with the qualitative analysis, with our method likely confusing the cylinder for the cube and thus producing a planar object.

C. Performance on real data

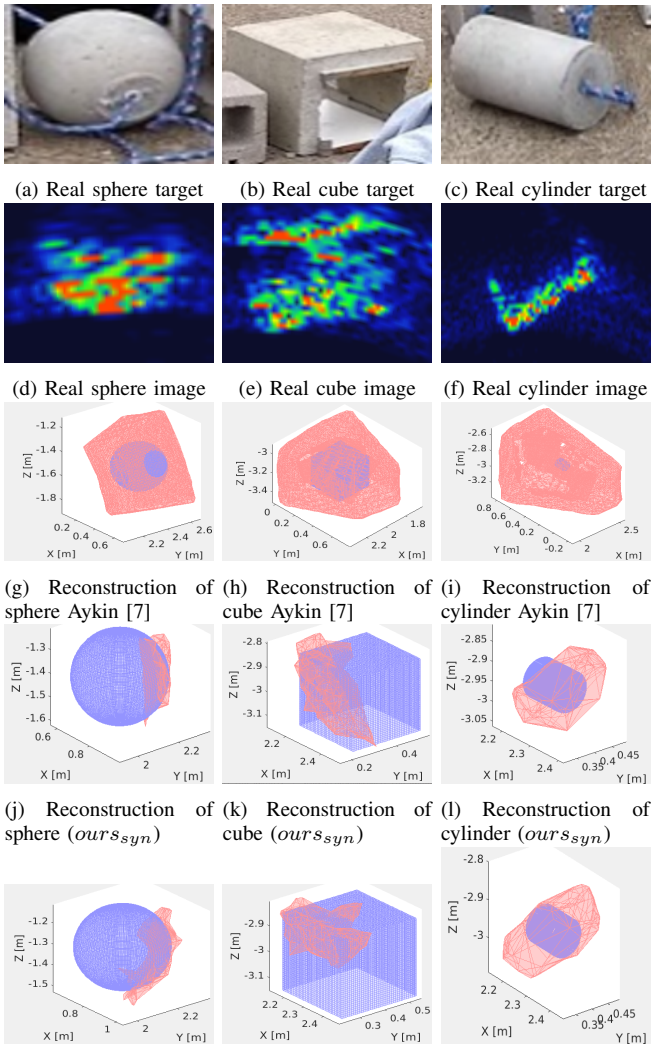
Finally, we demonstrate that our self-supervised method of training produces more accurate 3D reconstructions than just training on synthetic data or doing space carving. We first describe the dataset generated. Then, we describe how fine tuning on real data not only allows our method to predict the next frame better, but also generate more accurate 3D reconstructions than competing methods.

1) Real dataset: For this experiment we use underwater sonar imagery captured in Yaquina Bay, Newport, OR. We constructed concrete sonar targets to be imaged for dataset generation (Figs. 6a-6b). The sphere has radius 19 cm, the cylinder radius 5 cm and height 10 cm, and the cube side length of 29 cm. To capture the images, we drove around the objects at varying poses and ranges (on the order of 2-5 m) and captured images with different maximum ranges.

A human expert first hand-labels captured images as high or low-quality, and we use the high-quality images in our dataset. An automatic method, such as our previous work using a CNN, could be incorporated in the future [12]. We then resize images to 304x128 pixels and normalize according to Eq. 5. The final dataset contained 4,667 images with a 40% /30% /30% split for concrete spheres, cylinders, and cubes respectively. Real image examples can be seen in Figs. 6d-6e.

For training and final testing, we break up the dataset into 70% for training, 15% for validation and 15% for test. To ensure independence between validation and testing, the test set contains data from a separate deployment (same day) as the training and validation data.

During training on real data, we investigated the effects of the training schedule (what epochs to train with real or synthetic data). We found that training for 15 epochs of synthetic data was enough to start pre-training, and then



(a) Real sphere target (b) Real cube target (c) Real cylinder target

(d) Real sphere image (e) Real cube image (f) Real cylinder image

(g) Reconstruction of sphere Aykin [7] (h) Reconstruction of cube Aykin [7] (i) Reconstruction of cylinder Aykin [7]

(j) Reconstruction of sphere ($ours_{syn}$) (k) Reconstruction of cube ($ours_{syn}$) (l) Reconstruction of cylinder ($ours_{syn}$)

(m) Reconstruction of sphere ($ours_{real}$) (n) Reconstruction of cube ($ours_{real}$) (o) Reconstruction of cylinder ($ours_{real}$)

Fig. 6: The real sonar targets (a)-(c), their respective sonar images (d)-(f), and the reconstructed meshes using Aykin [7] (g)-(i), our model without real training (j)-(l) and with real training (m)-(o). For the meshes, ground truth is in blue and estimated meshes are in red. Because we only see part of the object, our estimated reconstructions are over only a part of the ground truth object. For the sphere and cube our model is able to capture the shape of the object given only a limited number of views. Best viewed in color.

alternated between 5 epochs of real training and 2 epochs of synthetic training for the remainder of the process (total of 120 epochs). To choose the best model for performance on real data, we complete a parameter sweep over the parameters listed previously in Section III-C for synthetic data, as well as the step size (temporal difference between I_2 and I_2') and the training schedule. We chose the final model based on evaluation of the photometric loss on real validation data. The final model had all of the same parameters as the synthetic model (Fig. 3a) except for the filter size, which was 3x3 for this case. The step size found to perform the best on validation data was 15 frames. The network was then trained using an initial learning rate of 1E-05 which halved every 50 epochs, the Adam optimizer, and a batch size of 9. Training was done using the procedure described in Section III-D.

TABLE III: Evaluation of predicting the next frame

	Photometric loss (unitless)
$Pred_{random}$	1.92
$Pred_{zero}$	2.01
CNN ($ours_{real}$)	**1.80**

TABLE IV: 3D reconstruction accuracy on real data

	RMSE (cm)		
	Sphere	Cube	Cylinder
Space carving [7]	14.8±0.25	17.96±0.18	42.2±0.42
CNN ($ours_{syn}$)	3.57±0.23	3.26±0.10	**6.53±0.91**
CNN ($ours_{real}$)	**3.14±0.29***	**3.09±0.23***	6.64±0.87

* = Statistically significant with p-value <0.05

2) Evaluating next frame prediction: We first tested our architecture's ability to predict the next frame. We compare with the same two baselines as before: $Pred_{random}$ and $Pred_{zero}$. Intuitively, as we learn to produce elevation maps more accurately, from both synthetic and real data, our ability to predict the next frame improves. Table III shows our results. As shown in Eq. 4, we scale the photometric loss to be roughly the same as the synthetic loss; consequently the photometric loss is unitless.

3) Real 3D reconstruction results: Finally, we demonstrate that our CNN-based approaches produce more accurate reconstructions than competing methods and training on real data produces better 3D reconstruction estimates. For each shape we generate reconstructions from 10 random subsets of 2 images. Because we do not know the ground truth location of the real object, point clouds are first hand-aligned to a ground truth model using a rigid transform. Then Iterative Closest Point is run, and finally the RMSE is computed using the MATLAB *pcregrigid* library.

In addition to the space carving baseline [7] from before, we also wanted to examine the benefit of fine tuning the network by training with real data. Thus, we compare our method only trained on synthetic data ($ours_{syn}$) to our method trained on synthetic data and then fine tuned with real images ($ours_{real}$).

We present the mean and standard deviations from the 10 subsets for each shape in Table IV. For each shape, we run a t-test to compute the statistical significance of a difference existing between the means of $ours_{syn}$ and $ours_{real}$. Our CNN approaches reduce the ambiguity given only a limited number of views when compared to the space carving baseline. Fine-tuning with real data produces more accurate meshes than just training with synthetic data for the sphere and cube because the network can adjust to the difference between real and synthetic data. For the cylinder, both CNN methods predict a vertical plane, suggesting that the confusion between the cube and cylinder from the synthetic training remains.

Qualitative reconstruction results are shown in Fig. 6. Similar to the synthetic case, our CNN-based approaches allow us to capture the shape of the sphere and cube much more accurately than the space carving baseline. For the cylinder, our results mirror the synthetic case where the CNNs likely get confused between the cylinder and cube

and produce more planar surfaces than the cylinders being imaged.

There are interesting qualitative differences between $ours_{syn}$ and $ours_{real}$. For the sphere, $ours_{real}$ better captures the curved shape of the sphere. While estimating the cube surface, $ours_{syn}$ produces a variety of surface shapes. Shown here is what would be expected from training on the synthetic data: a vertical surface. Interestingly, $ours_{real}$ produces many "corner" shaped objects shown in Fig. 6n. This makes sense because many of the real images of the cube, including Fig. 6e, are of the corner of the cube. This is indicated by the horizontal band in the middle of the image, where no reflections come back to the sensor.

V. CONCLUSION

In this work we propose a novel CNN architecture for producing dense 3D reconstructions. We utilize large amounts of synthetic data, where ground truth information is known, to pre-train the network. To train on real data we present a self-supervised training method which utilizes the physics of the sonar sensor to enable training without ground truth information. We show by fine tuning, the network adjusts to the real data and produce more accurate 3D reconstructions.

Our results suggest that if objects are different enough in image space (e.g. the cube and sphere) that the network can generate accurate reconstructions. When objects look similar in image space (e.g. the cube and cylinder) the network struggles more in its estimation. Future work could include incorporating multiple images into the estimation to better distinguish between objects of similar appearance in image space.

It would also be beneficial to test our approach in applications where the object is farther away from the sensor. We anticipate similar elevation angle errors. However, due to the increased range, actual cm accuracy and resulting reconstruction accuracy may decrease.

VI. ACKNOWLEDGMENTS

The authors thank Dr. Murat Aykin for his implementation of the space carving comparison method and Romulo Cerqueira for help installing his simulator. We thank Austin Nicolai for help with figure generation. Additionally, we thank Dylan Jones, Seth M^{c}Cammon, Graeme Best, Scott Chow, Gretchen Rice, and Jeff Caley for their helpful discussions and assistance in deploying the robot. Finally, we thank the anonymous reviewers for their helpful suggestions.

REFERENCES

[1] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, "Advanced perception, navigation and planning for autonomous in-water ship hull inspection," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, 2012.

[2] E. Coiras, Y. Petillot, and D. M. Lane, "Multiresolution 3-D reconstruction from side-scan sonar images," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 382–390, 2007.

[4] M. D. Aykin, "Forward-scan 2D sonar image formation and 3D reconstruction," *University of Miami, PhD Thesis*, 2015.

[3] Y. Petillot, S. Reed, and J. Bell, "Real time AUV pipeline detection and tracking using side scan sonar and multi-beam echo-sounder," in *Proc. IEEE/MTS OCEANS Conference, Biloxi, MI*, 2002.

[5] E. Westman, A. Hinduja, and M. Kaess, "Feature-based slam for imaging sonar with under-constrained landmarks," in *Proc. IEEE International Conference on Robotics and Automation*, 2018, pp. 3629–3636.

[6] M. D. Aykin and S. Negahdaripour, "Modeling 2-D lens-based forward-scan sonar imagery for targets with diffuse reflectance," *IEEE Journal of Oceanic Engineering*, vol. 41, no. 3, pp. 569–582, 2016.

[7] ——, "Three-dimensional target reconstruction from multiple 2D forward-scan sonar views by space carving," *IEEE Journal of Oceanic Engineering*, vol. 42, no. 3, pp. 574–589, 2017.

[8] T. Guerneve and Y. Petillot, "Underwater 3D reconstruction using blueview imaging sonar," in *Proc. IEEE/MTS OCEANS Conference, Genova, IT*, 2015.

[9] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in Neural Information Processing Systems*, 2014, pp. 2366–2374.

[10] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision*. Springer, 2016, pp. 740–756.

[11] T. A. Huang and M. Kaess, "Incremental data association for acoustic structure from motion," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 1334–1341.

[12] R. DeBortoli*, A. Nicolai*, F. Li, and G. A. Hollinger, "Real-time underwater 3D reconstruction using global context and active labeling," in *Proc. IEEE Conference on Robotics and Automation*, 2018, pp. 6204–6211.

[13] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *Proc. Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 239–248.

[14] J. Xie, R. Girshick, and A. Farhadi, "Deep3d: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 842–857.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[16] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," *Advances in neural information processing systems autodiff workshop*, 2017.

[18] T. A. Huang and M. Kaess, "Towards acoustic structure from motion for imaging sonar," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 758–765.

[19] E.-h. Lee and S. Lee, "Development of underwater terrain's depth map representation method based on occupancy grids with 3D point cloud from polar sonar sensor system," in *Proc. IEEE International Conference on Ubiquitous Robots and Ambient Intelligence*, 2016, pp. 497–500.

[20] Y. Ji, S. Kwak, A. Yamashita, and H. Asama, "Acoustic camera-based 3D measurement of underwater objects through automated extraction and association of feature points," in *Proc. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2016, pp. 224–230.

[21] R. Cerqueira, T. Trocoli, G. Neves, S. Joyeux, J. Albiez, and L. Oliveira, "A novel GPU-based sonar simulator for real-time applications," *Computers & Graphics*, vol. 68, pp. 66–76, 2017.

[22] J. McKay, I. Gerg, V. Monga, and R. Raj, "What's mine is yours: Pretrained CNNs for limited training sonar ATR," in *Proc. IEEE/MTS OCEANS Conference, Anchorage, AK*, 2017.

[23] D. P. Williams, "On optimal AUV track-spacing for underwater mine detection," in *Proc. IEEE International Conference on Robotics and Automation*, 2010, pp. 4755–4762.