AN ABSTRACT OF THE DISSERTATION OF

Robert DeBortoli for the degree of Doctor of Philosophy in Robotics presented on December 6, 2021.

Title: Deep Perception Without a Camera: Enabling 3D Reconstruction and Object Recognition using Lidar and Sonar Sensing

Abstract approved: ____

Geoffrey A. Hollinger

Deep learning has recently revolutionized robot perception in many canonical robotic applications, such as autonomous driving. However, a similar transformation has yet to occur in more harsh environments including underwater and underground. This is due in part to the difficulty in deploying robots in these environments, which lack large real training datasets and often necessitate the use of non-traditional sensors for deep learning (e.g. imaging sonars and lidars). In this dissertation we demonstrate that by explicitly accounting for the sensor noise beget by challenging environments and by incorporating synthetic data in the training process, the power of deep learning can be leveraged for deployment in these harsh environments.

In our first contribution we develop a framework that enables the real-time 3D reconstruction of underwater environments using features from 2D sonar images. Due to noisy and low-resolution imagery as compared with standard cameras, accurate sonar image analysis necessitates the explicit consideration of noise. While deep learning by using Convolutional Neural Networks (CNNs) has been leveraged on sonar images, previous CNN-based methods do not explicitly consider the noise (from factors such as multi-pathing or irregular surfaces) often present in the images. In this contribution our key insight is to use atrous convolution, which has a larger field of context than standard convolution and is thus not misled as much by localized noise. We demonstrate that atrous convolution, as well as human-in-the-loop feature annotation, provides real-time reconstruction capability on datasets captured onboard our underwater vehicle while operating in a variety of environments.

In our second contribution we remove the human from the loop and develop an approach which leverages deep learning for a fully automated 3D underwater reconstruction algorithm using 2D sonar images as input. Our algorithm is able to produce accurate estimates even when common physical models break down due to phenomena such as non-diffuse reflections. Inspired by our success in the previous contribution, we propose the utilization of CNNs as a powerful method to extract meaningful information without being misled by noisy data. To ensure training convergence, 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 information for training. Our method can produce accurate 3D estimates given only a single image. We demonstrate that our method produces 3D reconstructions with an 80% reduction in Root Mean Square Error compared to previous approaches, both in simulation and on real data.

We then extend this approach to leverage the series of images the robot collects as it moves through the environment. Specifically, we develop two CNNs that take as input multiple images captured at different points in time and output a more accurate prediction than just using a single image as input. To our knowledge this is the first such multi-sonar-image CNN designed for the 3D underwater reconstruction task. We validate this extension on synthetic and real data and show up to a 5% improvement over competing methods.

Finally, we develop an improved method for incorporating synthetic data into the training process. This takes our previous contribution a step further by more tightly coupling synthetic and real point cloud feature extraction. We develop an adversarial training technique, which along with the standard object detection loss provides a training signal that encourages similar feature extraction from both synthetic and real clouds. This brings the training process closer to the preferred scenario: where the synthetic point clouds contain features that are very similar to those found in the real scans. We validate our approach in the context of the data-limited DARPA Subterranean Challenge and demonstrate that our 3D adversarial training architecture improves 3D object detection performance by up to 15% depending on the data representation.

Deep Perception Without a Camera: Enabling 3D Reconstruction and Object Recognition using Lidar and Sonar Sensing

by
Robert DeBortoli

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented December 6, 2021
Commencement June 2022

ACKNOWLEDGEMENTS

I would like to thank the members of RDML past and present for the enormous amount of assistance in the development of this thesis, particularly: Graeme Best, Jeff Caley, Jen Jen Chung, Dylan Jones, Nick Lawrance, Seth M^{c}Cammon, Tim Player, Ian Rankin, Manish Saroya, Andrew Smith, and Thane Somers for help developing research ideas, deploying robots, and moving heavy concrete sonar targets. Thank you also to Neale Ratzlaff and Lawrence Neal of the Deep Machine Vision group for their helpful theoretical discussions. Thank you to my collaborators at CMU for allowing me the opportunity to gain experience developing and deploying a robotic system from the ground up. Vasu Agrawal, Bacon, Bill Drozd, and Henry Zhang were particularly helpful in providing technical expertise and training which made me a better real-world roboticist.

I would like to thank my advisors Geoff and Fuxin for their consistent training in conducting high-quality research as well as their guidance on identifying promising research directions. I would like to thank my committee members Cindy Grimm, Michael Olsen, and Bill Smart for offering their diverse expertise and being available whenever I sought advice. I would like to thank Ashish Kapoor for his assistance in learning to conduct high-impact research in an environment outside of academia.

I would like to thank my family: Mark, Julie, Mitchell, and Olivia for their support in more ways than I can count, both before the start of my graduate education and during. I would like to thank Chris Lee for his friendship and support both inside and outside the lab. Finally, I would like to thank Ariel for the immense amount of encouragement and support, without her this thesis would not have been possible.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

Robots have the potential to monitor and inspect environments often considered too harsh or unsafe for human intervention. For example, underwater robots can be used to drastically reduce the cost and time associated with tasks such as ship hull inspection [34, 37] and aquatic wildlife monitoring [7]. Underground, robots can search for survivors in scenarios where fire, smoke, or unstable structures prohibit human rescuers from safely searching the environment [1]. Oftentimes in these harsh environments, cameras have reduced visibility as low as five meters or less. This is due to a variety of factors such as dust in the air or sand particulates in the water, in addition to the lack of environmental lighting. Therefore, in this thesis we develop approaches to leverage powerful deep learning architectures with non-traditional sensors that provide increased visibility, namely 2D underwater sonar and 3D LiDAR.

Alternative sensing modalities such as underwater sonar, radar, or LiDARs are often used for perception tasks in harsh environments [103, 104]. These sensors provide increased visibility up to 100 meters or more; however deploying deep learning models for such sensors has been difficult for two primary reasons: *1) large datasets do not exist (and are often extremely difficult to obtain) for harsh environments* and *2) sensor input is typically noisy enough to demand explicit attention*. This noise can take many forms but loosely refers to the fact that multiple sensors measurements of the same object or environment, that should theoretically be the same, can be significantly different.

These challenges have prevented deep learning models from being deployed in harsh environments, precluding a perception transformation similar to that recently achieved in terrestrial applications [85].

In applications such as autonomous driving, large corporations such as Audi [27] and Waymo [91] have supported the generation of large datasets (on the order of hundreds of thousands or millions of examples). However, such datasets do not exist for underwater or underground applications. This is due in part to the unique output data structures of these alternative sensing modalities, which limit the number of people who can label such data. Additionally, these harsh environments are difficult and expensive to access, often requiring specific ship time or environmental permitting accommodations to be made.

Simulated data can be used as a supplement for real data, however a sim-to-real gap often exists which must be addressed before deployment in the real world [76]. To combat this lack of real data, in this thesis we develop sim-to-real and self-supervised techniques that enable the training of deep learned models in data-limited harsh environments. We specifically focus on improving 3D reconstruction from 2D underwater sonar images and 3D underground object detection from LiDAR point clouds.

In addition to leveraging large real datasets, many previous deep learning approaches are able achieve accurate results while not directly considering sensor noise. However, such approaches are limited in harsh environments. For example, underwater sonar images of the same object can appear very different depending on the local surface roughness of the object, surrounding reflectors such as the seafloor, or small variations in the pose of the robot (Fig. 1.1). Similar effects can be seen in other applications using

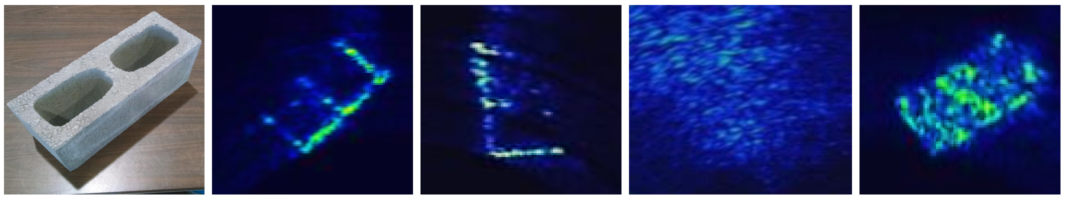


Figure 1.1: Examples of difficult-to-analyze 2D sonar images. Speckle noise, such as in (e) is extremely difficult to physically model and makes local feature extractors unreliable. In this thesis we develop feature extractors and architectures which take a higher-level view than traditional methods and are less likely to be affected by such noise.

alternative sensing such as radar for autonomous driving [104]. To better leverage the power of deep learning in harsh environments, in this thesis we develop models that are explicitly noise-robust.

Towards improving robotic perception capabilities in harsh environments, this thesis presents techniques for leveraging the power of deep learning in scenarios that lack large-scale realistic datasets and often are very difficult to model using physical models. While we investigate underwater and underground environments specifically, such techniques could be extended for many challenging robotic perception tasks (e.g. autonomous cars operating in poor weather). Specifically, our contributions are:

1. A human-in-the-loop method that leverages deep learning for reliable 3D reconstruction of underwater environments in scenarios where traditional feature extractors fail due to insonification inconsistencies. Our approach uses *atrous convolution* to analyze the imagery which has a larger field of view than standard

convolution. This increased context results in a more noise-robust architecture that overfits less to the limited training data.

2. A deep-learned architecture which removes the human from the loop and enables automatically generating 3D underwater reconstructions. Previous methods rely on physics-based models which become unreliable as insonification inconsistencies corrupt the sonar image data. In terrestrial applications, deep learning has shown the ability to gain a higher-level context, from training with large amounts of data and architecture design, in making accurate predictions in spite of sensor noise. However, leveraging deep learning for underwater 3D reconstruction has until now not been prevalent due to the lack of large underwater 3D sonar datasets. In this contribution we propose using synthetic data intelligently in our training data to utilize the power of deep learning.

3. An adversarial training technique to better incorporate synthetic data for 3D object detection networks. While mixing synthetic data into the training process enables deep network deployment in environments without a lot of real-world data, our third contribution shows that explicitly reasoning about the features extracted from the synthetic data can improve synthetic data utilization when training deep networks. Specifically, by training the deep network to extract similar features from both synthetic and real data, we move the training of the network closer to the optimal scenario of having large amounts of representative data for training. We demonstrate the efficacy of this approach on the application of 3D object

detection in subterranean environments, where the challenging environment has inhibited the release of large-scale datasets.

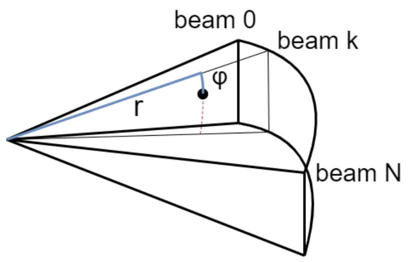
Thesis Roadmap: In Chapter 2 we discuss the background for this thesis including the sonar imaging model, LiDAR point cloud generation, and deep learning. In Chapter 3 we present our first contribution for enabling real-time 3D underwater reconstructions in spite of large amounts of noise. Chapter 4 contains our second contribution for completing dense 3D reconstructions without a human in the loop while still being noise-robust. In Chapter 5 we discuss our third contribution on better utilization of synthetic data during deep network training for the application of 3D object detection in subterranean point clouds. We finally conclude in Chapter 6 with a summary of our contributions and directions for future work.

Chapter 2: Background

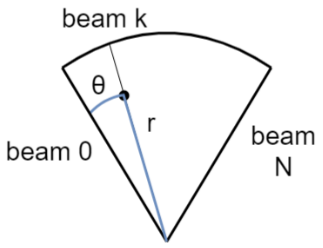
We will next cover the necessary prerequisite material to guide our presentation of this thesis. We will start with the underwater sonar 2D image formation model, then cover generating 3D point clouds using a lidar, and finally finish with an overview of deep learning and its applications to these sensor outputs.

2.1 Underwater Sonar Imaging Model

Turbid waters often limit the range of sensing for conventional cameras and thus underwater 2D imaging sonars are the preferred sensing modality for underwater missions. There are many types of 2D imaging sonars including sidescan [77], multi-beam [38], pencil beam [113], synthetic aperture [106], and mechanically scanning [97]. In general a multi-beam sonar, unlike the other sonar types, offers the ability to generate 2D images very fast (often on the order of tens of Hz) by sending out and receiving multiple acoustic beams from a single static sensor. The use of a single planar array of receiver/transmitter units (along with techniques such as beamforming [33]) leads to several advantages for being deployed on small underwater vehicles including: low weight, small size, and relatively low cost. Due to these advantages, multi-beam sonars have been used in a variety of previous work for underwater inspection tasks and are the sensor we will use in this thesis [38, 43, 55]. 3D sonars do exist, however their size, cost, and slow speed of operation preclude widespread use on research-class Remotely Operated Vehicles (ROVs) such as those used in this thesis [3].



(a) Computation of range and elevation angle



(b) The acquisition of bearing by emitting multiple beams

Figure 2.1: Mapping from Euclidean to polar coordinates.

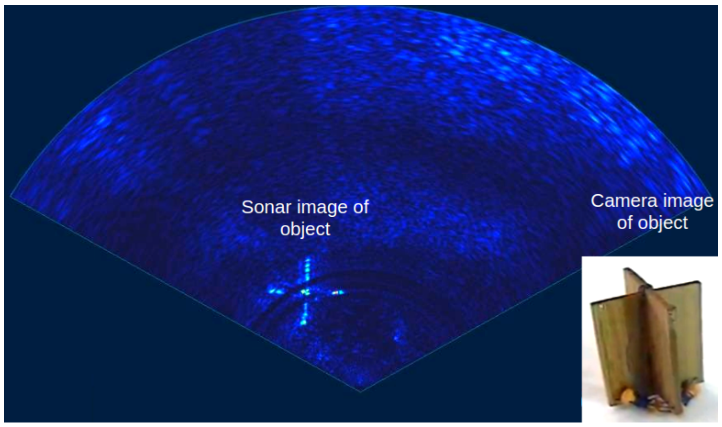


Figure 2.2: Example of a sonar image of an X-shaped object.

We next describe the sonar imaging model for multi-beam sonars. Following traditional notation, a sonar transforms a 3D point (X, Y, Z) into polar coordinates (r, θ, ϕ) for range, bearing angle, and elevation angle respectively. Shown in Fig. 2.1, a multi-beam sonar images by sending out a series of acoustic waves, each at a different bearing angle θ and measuring both the time-of-flight r and intensity of the return I of the reflected wave off of environmental objects. Importantly, the elevation angle ϕ (seen in Fig. 2.1) is lost in the imaging process. Multi-beam sonar images (shown in Fig. 2.1) therefore have pixel dimensions *range* and *bearing*. In other words, every point along the elevation arc has the same range and bearing and thus will reflect energy that is mapped to the same pixel in image space. The collapse of this dimension is analogous to the collapse of the depth dimension in 2D camera images. It is therefore often the goal of underwater 3D reconstruction algorithms to recover this missing dimension. In this thesis we use the Tritech Gemini 720i which is a common multi-beam sonar that has 256 beams creating a swath of 120°, a max range of 120m, and ambiguity in the elevation dimension of 20°.

2.1.1 Lambert's Law

To fully understand the various approaches for converting 2D underwater sonar images into 3D representations, we now describe Lambert's Law, which is a common technique for modeling the image formation process. Lambert's Law for underwater sonar assumes that the reflecting object is *Lambertian*, meaning the reflector spreads the acoustic energy diffusely. In the experimentally validated variant of Lambert's Law, sonar image intensity I depends on the angle between the acoustic wave and the surface normal of the reflector:

$$I = kcos^2(\alpha) \tag{2.1}$$

where k is a normalization term that accounts for factors such as distance to the reflector (farther away objects will reflect less energy back) and α is the angle between the emitted wave and the surface normal of the reflector [6]. This formulation, has been used in many previous works [6] and while accurate in some circumstances, is sensitive to speckle noise and multipath reflections, as noted by the authors in [3]. As an alternative to directly modeling the low-level noise in the sonar imaging process, in this thesis we present deep learning approaches which are a high-level estimators and therefore less sensitive to this noise.

2.1.2 Challenges Analyzing Underwater Sonar Images

Sonar image analysis shares some challenges with standard camera images. For example, occlusion occurs in sonar images when part of the object cannot be reached by the emitted sonar beam (Fig. 1.1c). However, due to the insonification process, there exist many challenges in sonar that are not prevalent in camera imagery. At the forefront of these challenges is insonification inconsistencies which occur for a variety of reasons including the interference of acoustic beams [63] and the large impact on image pixel intensities from small relative changes in pose between the sonar and object being imaged. Speckle noise is also prevalent in sonar images and can make the same object look different in different images. For example, in Fig. 1.1e, even though most likely the side of the cinder block is being imaged, a solid rectangle is not created in image space. Environmental noise can also occur when surfaces other than the object (such as the seafloor) reflect energy back to the sonar (e.g. Fig. 1.1d). Addressing the low signal-to-noise ratio has been the focus of many previous works [9, 40].

Recently, CNNs have been used to analyze sonar imagery. Kim, et al. use CNNs to analyze sonar imagery and track the trajectory of another underwater vehicle [47]. While they are able to track this vehicle accurately, we demonstrate in Chapter 3 that by explicitly accounting for noise in our deep network, our method is more accurately able to analyze sonar images. Williams and Dugelay address the problem of noise in sonar imagery by fusing together multiple views of the object of interest and using a deep network to classify images as either containing a man-made object or a naturally occurring rock [107]. Using techniques such as atrous convolution, in this thesis we develop methods which do not require sets of images for noise-robust analysis.

2.1.3 3D Reconstruction from 2D Sonar Images

There are three basic 3D representations utilized for underwater reconstructions: voxel or occupancy-based maps, meshes, and 3D pointclouds. Voxel-based approaches typically build a large discretized grid of places where sonar returns have been recorded [5, 23, 30, 93, 97]. This is analogous to voxel-based approaches in terrestrial applications [22]. Mesh-based approaches build a consistent set of 3D points, usually using some information about surface normals [4, 34, 66]. Finally, 3D point clouds simply represent the object or features as a set of 3D points [20, 38, 39]. In our first contribution we use point clouds as our final 3D reconstruction output. In our second contribution we use 3D point clouds as the output of our CNN-based elevation angle estimator. Standard methods such as Matlab's *trimesh* function are then used to generate meshes from the point clouds, providing a richer representation for the human operator.

While there exist methods for using multiple sonars to generate these 3D representations [67], here we focus on single sonar formulations which can be deployed on small inspection vehicles. For this case, broadly there are three ways to obtain an underwater 3D representation: modeling the physics of generating the 2D images, using specific motion patterns, or assuming the object is in a certain region of the environment.

The physics-based approach shape-from-shading is a well-established method for reconstructing surfaces by observing the impact of light on the surface [35]. This method can be used for underwater reconstruction applications if the acoustic signal is taken as an analogy to light and is modelled using Lambert's Law. Using this analogy, shape-from-shading approaches typically involve jointly optimizing with respect to image intensities (which correspond to surface normals) and consistencies among estimated surface normals (the surface does not change a lot quickly). For example, Aykin and Negahdaripour [5] use shape-from-shading to generate meshes of complex underwater objects such as rocks and Coiras et al. [15] use shape-from-shading to build 3D pointclouds of the seafloor.

While such approaches perform well, there do exist some drawbacks. Such approaches generally do not consider the case of multiple returns along the same elevation arc, resulting in reconstructions that only have one reflector per elevation arc. This of course may not always be true of the object being imaged. Additionally, speckle noise can make shape-from-shading approaches difficult to properly converge [5]. As we will show, the hierarchical nature of deep learning and the ability to learn from previous data allows deep-learned models to be less sensitive to the noise that often plagues physics-based methods.

Using less sensitive techniques to recover the missing dimension have been achieved using certain motion sequences. As discussed in Section 2.1, the missing dimension in 2D sonar images is the elevation angle, where the ambiguity lies roughly in an arc in the z direction. To reduce this ambiguity, motion-based approaches often rely on $roll$. For example, in [5] roll is used to create a voxel-based representation of underwater objects. Depending on the amount of roll and observation positions, on the order of 10 roll actions per 5-10 observations poses can be necessary to recover the 3D shape. Moving in a vertical direction, almost along the exact plane of ambiguity in sonar measurements, has also been shown to resolve ambiguities in underwater reconstructions [30]. While specific maneuvers can be made in open environments, inspecting cluttered environments or generating 3D reconstructions in a “fly-by” scenario requires a less-constraining method. In Chapters 3 and 4, by leveraging the representative power of deep learning, we present methods for underwater reconstruction which do not require specific motions in order to generate accurate reconstructions.

The third and final class of reconstruction methods presented relies on the assumption of the object being in a certain region of the environment. For example, [37] assumes that the imaged points lie on the plane where the elevation angle ϕ = 0. This works well in their ship hull inspection application, but perhaps would not work as well in an environment with more complex objects of interest. This particular planar assumption is relaxed in [38] using factor graph optimization techniques. More general planar assumptions have also worked well when inspecting submerged structures. For example, while inspecting a flood gate with a very regular structure, [69], fit the 3D sonar points to a plane to complete accurate registration across frames.

The reconstruction of objects sitting on the seafloor, by using the object shadow has also been explored. For example, in [4], the length of the shadow is used to estimate the height of the object, which is used to constrain the optimization on the object mesh. While such approaches work well, they are of course constrained to the case where the object lays on the seafloor. Objects such as dock pilings, moored underwater devices, and other common objects of interest to be mapped are not on the seafloor and thus do not project shadows. As we will show, deep networks can be trained with data of objects in the water column, which enables the accurate generation of 3D reconstructions without the objects being constrained to being on the seafloor.

2.2 Lidar Point Cloud Generation

While cameras provide robots with a cost and weight-efficient manner to capture detailed imagery, they oftentimes lack 360-degree fields of view and do not naturally output 3D information. Light detection and ranging (lidar) is a sensing modality which allows for rapid capturing of 3D environmental information on the order of hundreds of meters.

Lidars work by sending out a set of laser beams at known angles, ϕ_i for beam i, and measuring the time-of-flight from the sensor, to the environment, and back which gives the range dimension r. Lidars typically spin around the upwards-facing axis giving different bearings θ to each laser beam. This process is shown in Fig. 2.3. A typical output of a lidar is a 3D point cloud, an example of which is shown in Fig. 2.4b.

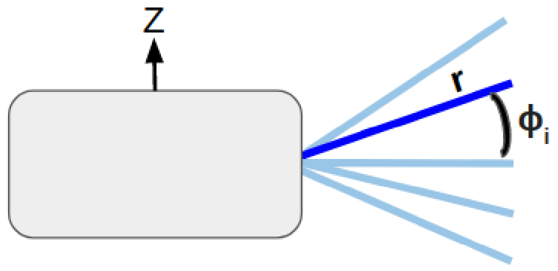
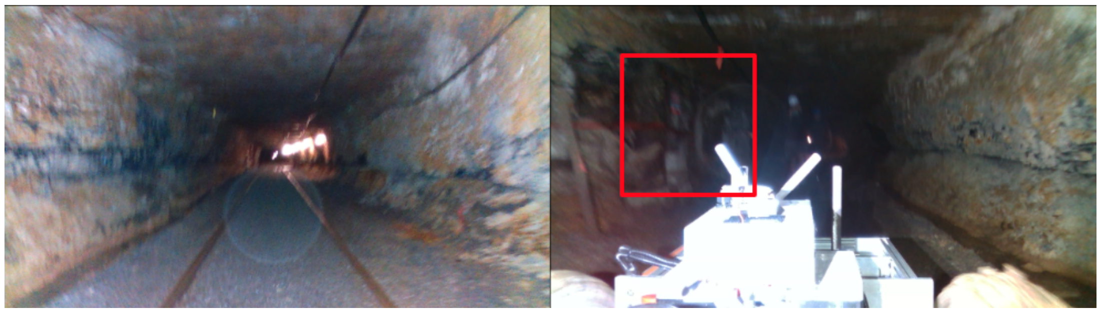


Figure 2.3: Lidar point cloud generation. In this example, a series of 5 laser beams (blue) are sent out at known angles (e.g. ϕ_i) and the range to the environment is measured. The lidar (grey) spins this set of lasers around the Z-axis to obtain 3D information about the environment.

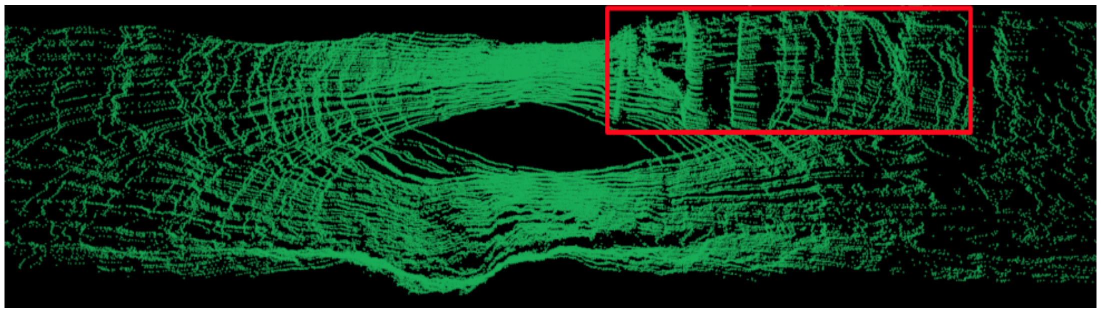
While point clouds generated from lidars contain detailed 3D information, the cloud structure is very environment-dependent. Specifically, point clouds can have widely varying local densities and number of points, depending on factors such as smoothness of the surrounding environments and distance from the robot. These challenges, combined with clouds containing 100s of thousands of points and large physical scales (on the order of dozens of meters), motivates the need for efficient point cloud processing algorithms.

2.3 Deep Learning

The area of deep learning has helped perception researchers achieve state-of-the-art performance in a variety of robot perception tasks including object detection [48], tracking [29], and physical interaction prediction [10] among others. Deep learning in a general sense is the extension of neural networks by adding more layers of neurons [82]. In the past, long training times and the lack of large open-source datasets limited the use of



(a) Left: Forward facing image. Right: Backwards facing image with vertical brace structure in red box, not visible due to lack of light.



(b) Side-view of point cloud data using the Velodyne VLP-16 lidar in the same environment. Vertical brace structure (in red box) is much clearer because lidar does not rely on adequate lighting. Robot is facing to the left in this cloud

Figure 2.4: Example of lidar providing 3D information over long ranges in visually degraded subterranean environment.

deep networks. Advances in GPU computing and techniques such as back-propagation have made the training of these networks much faster. The open availability of large datasets (e.g. KITTI [26] for autonomous driving and NYU Depth [86] for indoor mapping) have made the training of these networks tractable. Unlike for autonomous driving and household scenarios, underwater and underground environments lack representative training sets. We will show that by developing methods for intelligently utilizing simulated data, deep networks can be trained and deployed in environments without a large amount of real training data.

The simplest formulation of deep learning, the neural network, was designed as a way to map input features to a desired output (e.g. classification or control action). The basic building block of a neural network is a neuron. A neuron takes as input a weighted sum of input features and applies an activation function (such as the sigmoid function) which produces the output. In this simple formulation, a neuron can be represented as:

$$output = activation\left(\sum_{i=1}^{K} w_i * x_i\right) \tag{2.2}$$

where K is the number of input features, w is the learned weight vector, and x is the input feature vector. This structure is roughly inspired by neural systems in biology, which found that certain neurons fire given certain stimuli [82].

Neural networks are *layers* of neurons that interact via *connections* where the output of one layer is the input to the subsequent layer. These "vanilla" neural networks are called "fully connected" because every neuron in one layer is connected to every neuron in the subsequent layer. A single hidden-layer neural network can be defined as:

$$output = \mathbf{W}_o f_a(\mathbf{W}_i x + b) \tag{2.3}$$

where $\mathbf{W}_i$ is the input weight matrix, $\mathbf{W}_o$ is the output weight matrix, x is the input feature vector, f_a is the activation function, and b is the bias vector.

Intuitively, with more neurons per layer and/or more layers, more complex expressions can be represented. Indeed, neural networks have been proven to be Universal Function Approximators [36].

Learning a mapping between some input and some desired output is generally the purpose of neural networks (also called models). *Learning* can be thought of as a credit assignment problem: given some error between the desired and actual output of the network, determine how much each weight contributed to this error [82]. This assignment can be done through the well-established stochastic gradient descent (SGD) algorithm, which approximates the gradient of the cost function and takes a small step to minimize it. There has been significant development in devising methods for allowing computers to utilize SGD and similar techniques effectively. For example, momentum, which discourages large changes in the weight update, was developed to enable more stable convergence [71]. Perhaps most important is back-propagation which leverages the chain rule to perform gradient descent, and thus credit assignment, computationally efficiently [79].

2.3.1 Neural Network Models for Robot Perception

We next describe two types of neural networks widely used for robot perception, namely: Convolutional Neural Networks (CNNs) which account for the 2D spatial nature of images and neural networks capable of handling 3D point cloud data.

2.3.1.1 Convolutional Neural Networks

In the early days of image analysis, discriminative features for tasks such as object recognition were extracted using static hand-designed 2D filters such as Sobel filters (which rely on image gradients) [89]. An example Sobel filter is shown in Fig. 2.5. In the following years, perception researchers spent much time and effort designing filters such as SIFT [58] and SURF [8] that were scale-invariant and highly descriptive. Still, the types of features being extracted were pre-defined by the designers of these filters. In recent years, CNNs and similar deep learning structures have provided a way to reduce the amount of effort in feature engineering by *learning* filters for the extraction of features. In a multitude of applications, this has resulted in an increase in visual task performance [54, 90].

The atomic operation for CNNs is the sliding of a convolutional filter (or kernel) over an input image to extract image features (such as image gradients or edges). Convolution can be understood as the weighted sum of pixels in a region where the weights are the coefficients in the filter. Thus, these coefficients prescribe the features to be extracted. In this context, the output of convolution is a 2D activation image (or feature map) containing high pixel intensities in areas where the corresponding features occur, and low pixel intensities where the features are absent. As an example, the filter in Fig. 2.5 will activate highly on vertical edges because the weights (positive vs. negative) are aligned vertically. The use of 2D filters are intuitive because they exploit the inherent 2D structure present in 2D images.

The key behind a CNN is that it learns the coefficients for the filters, thus learning the type of features to extract from an image for a given task. This learning is done very similarly to standard neural networks discussed previously, with weight updates based on a variant of back-propagation. Learning the coefficients on these filters provides a concise method for training because the weights for the filters are not dependent on the image size or location in the image (called weight-sharing). As an alternative, if we naively vectorized the image and input it into a neural network, the number of parameters to learn would grow to an intractable amount (on the order of 1 billion versus 140 million for a standard CNN).

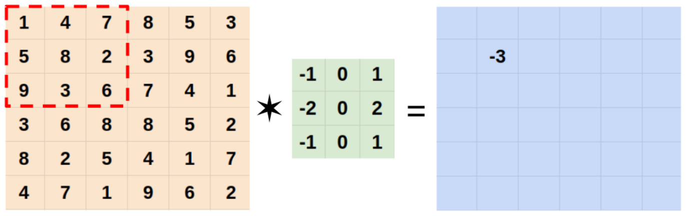


Figure 2.5: Convolution process. The filter (center) is slid over the original image (left) and the resulting feature map is a sum of the element-wise products between the filter and original image.

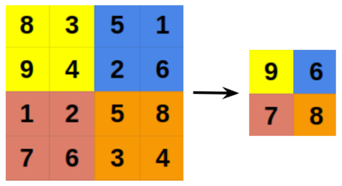


Figure 2.6: Max pooling using a 2D neighborhood

In 2012, the Imagenet computer vision challenge (which includes tasks such as object classification) was won for the first time by a CNN, named AlexNet after its creator

Alex Krizhevsky [49]. This network is widely believed to have launched CNNs into popularity and we will use it as a case-study in CNN architecture design.

AlexNet contains several kinds of layers including **convolutional**, **max pooling**, **activation**, and **normalization**. *Convolutional* layers consist of many filters (on the order of tens or hundreds) for sliding over the image and extracting various types of features. *Max pooling* layers reduce the image size as it goes through the network by keeping only the maximum response in a small neighborhood (typically 2x2). An example of max pooling is shown in Fig. 2.6. This is important because smaller images, when input into the fully connected layers at the end of these networks, reduce the number of parameters (weight connections) to be learned. Non-linear *activation* layers improve the expressibility of the network, because convolution is a linear operator. An example activation is the Rectified Linear Unit (ReLU), which is defined as $f(x) = max(0, x)$. ReLU activation allows for an easy-to-compute gradient (either 0 or 1) and maintains the attractive property of being non-linear; both of which improve training networks [49]. Finally, *normalization* layers such as batch normalization (developed after AlexNet but popular today) allow for faster training by normalizing the weights in the network over a batch of data [41].

After going through a series of convolution/pooling/activation/normalizations, the network has a high-level representation of the image. More specifically, while the features learned at the earlier levels may be simple edges and corners, the features learned by the network at later levels are combinations of these edges and corners, which form features such as eyes, mouths, or faces. In this way, such networks are considered to be hierarchical. Finally, at the end of AlexNet are *fully connected* layers. At a high level, this allows for combinations of features to be learned.

Since AlexNet, the computer vision and deep learning communities have devised a number of ways to improve the learning capabilities of CNNs. VGG-16 showed the importance of a hierarchical structure by making a very deep network (16 layers) [88]. GoogLeNet introduced the Inception module which allowed for different operations to happen to the images in parallel instead of sequentially [92]. Residual networks, which use skip connections between early layers to later layers, have recently allowed for networks on the order of hundreds of layers to be trained [32].

2.3.1.2 Using CNNs to Estimate the Missing Dimension

Specifically relevant to our task of 3D reconstruction underwater, CNNs have also been used to predict the missing dimension in 2D images. To our knowledge, our method in Chapter 4 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 [21, 52]. 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 [21]. 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 [25, 110]. 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 [25]. To take this virtual image, bilinear sampling is used, which after defining sub-gradients is differentiable.

Recently there have been a variety of works which use self-supervision. A distinguishing feature of many of these works is the manner in which the virtual image taken in a differentiable manner (to enable training) [25, 110]. For example, Garg et al. use a Taylor Series expansion is used to linearize around the predicted disparity and move pixels from one image to the other [25]. 2D underwater sonar images have a different geometry and thus such approaches cannot be used for our applications. Instead, we develop more general methods that do not require pixel disparities to be established for training and just use the general geometry of the sonar imaging model to train deep networks.

2.3.1.3 Leveraging Information From Multiple Viewpoints in CNNs

While single-image CNNs have shown great promise in a variety of applications, robotics typically gather many images in a video-like format as they traverse throughout an environment. To leverage the potentially complementary information in these multiple viewpoints, recent research attention has been given to multi-view CNNs, which taken as inputs two or more images.

While there are a variety of ways to handle multi-image inputs, relevant to this thesis are two broad techniques: *early* and *late* fusion. Early fusion involves concatenating the set of input images in the channel dimension, creating an input that is of shape (height, width, num_{images}) assuming a greyscale input [80]. While this can encode information from multiple images with the added benefit of no structural changes to the actual CNN, if the scene changes greatly between consecutive images, it may be difficult for a CNN to correlate changes across images. Late fusion combats this by first extracting features from each input image using a shared set of weights and then concatenates these features for further processing. This has been shown to improve performance in RGB applications such as water pouring [80] and in semantic segmentation of forest environments [95].

2.3.1.4 Processing Point Clouds Using Deep Learning

Point clouds, unlike camera images do not have a defined structure. Camera image pixels lie on a pre-defined 2D grid while points in a point cloud can have varying local densities which are dictated by the surrounding environment. Therefore, CNNs which are well-suited for well-structured, dense signals, make less sense for processing point cloud information than camera images. Specifically, the filters integral to CNNs are traditionally defined with respect to such a regular and 2D structure. Changing these underlying data properties makes the use of these rigid filters much less compelling.

In spite of this, some of the first attempts to leverage the power of deep networks used CNNs directly on point clouds [60, 56]. For example, Li used a 3D convolution to detect cars in an autonomous driving application [56]. However, the use of such dense filters on such a large scene limited the algorithm's speed to 1Hz. To speed up processing while still leveraging CNNs, methods have also been proposed which project the point cloud into a 2D view, such as the bird's eye view. Complex-YOLO was recently introduced as a technique to complete object detection with orientated bounding boxes in bird's eye view images [87]. While such techniques work fairly well in autonomous driving scenarios, where large objects such as cars can be seen easily from top-down views, such methods are not appropriate for applications such as underground exploration, where clutter and large overhangs preclude the use of a top-down projected view.

More natural than using a dense feature extractor or compressing the 3D information is to operate directly on the points of the point clouds data. However, this precludes the use of a traditional CNN. Instead, Qi et al. proposed an architecture called "PointNet", which uses series of fully connected layers operating directly on the point clouds [74, 75]. These fully connected layers along with intelligent sampling enabled a powerful way to extract features directly from the points without compressing information or using inefficient 3D convolutions. Recently Wu et al. proposed PointConv, a deep network which uses fully connected networks to learn a 3D convolution operation that is better-adapted to dealing with the varying densities of point clouds than PointNet and PointNet++ [109].

Armed with powerful feature extractors operating directly on 3D points, research attention has very recently been given to completing object detection in 3D point clouds while operating directly on the points. For our applications, the object detection problem is: Given a set of 3D points, output a set of bounding boxes with center (x, y, z), dimensions ($length$, $width$, $height$), and orientation (yaw) corresponding to the objects of interest in the scene. To achieve this, Qi et al. proposed VoteNet [73], which uses PointNet++ to extract feature vectors from the point cloud and uses these feature vectors to predict an object center. Clustering around a predicted object center is then used to predict a corresponding object bounding box.

2.3.1.5 Closing the Sim-to-Real Gap for Point Clouds

Reducing the sim-to-real gap between simulated and real point clouds has received only a small amount of research attention. Lai and Fox demonstrated the use of synthetic Google 3D warehouse models for detecting objects in real-world autonomous driving scenes [51]. While they developed an accurate model, it took on the order of 80 seconds to analyze a single scene. In this thesis we are aimed at real-time point cloud processing on the order of 5-10 Hz. SqueezeSegV2 is an alternate approach which projects point clouds into sets of 2D images and then learns to add realistic point intensities to synthetic point clouds from Grand Theft Auto V [108]. While training with these added intensities improves segmentation performance on the KITTI dataset, projecting into 2D images may warp the input scene and inherently encourages neighboring pixels to have some correspondence, even if they are at very different distance. In this thesis we develop a method for intelligently incorporating synthetic data during the training of an object detector which operates on the raw points and is thus efficient without compressing information.

For robot perception in harsh environments, previous work has left multiple research gaps which we make progress towards in this dissertation. Instead of relying on specific underwater vehicle motions or relying on objects being reconstructed to be on the seafloor, we leverage deep learning to generate accurate 3D reconstructions in more general scenarios. We demonstrate by using the naturally hierarchical and powerful construction of deep learning architectures, we are less sensitive than the previously-favored physics-based methods for generating 3D reconstructions. We also demonstrate that by intelligently incorporating synthetic data into the training process, deep networks can be trained for use in harsh environments with limited amounts of training data. These ideas cut across applications as we show the efficacy of using synthetic data both underwater and underground.

Chapter 3: Real-Time Underwater 3D Reconstruction Using Global Context

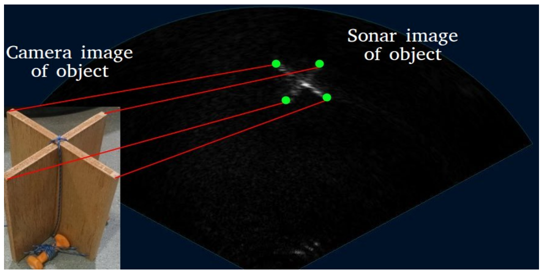
In Chapter 2 we outlined the state-of-the-art in 3D underwater reconstruction from 2D sonar images. One large research gap identified was the sensitivity of previous methods to the noise present in sonar images. In this chapter we present a novel deep network for analyzing noisy underwater sonar images, which leverages *atrous convolution* for more noise-robust image analysis. We compare to architectures that do not build in such noise-robust provisions and demonstrate that by explicitly considering noise, our method is more accurate at detecting useful images for 3D reconstruction.

Because of the large amounts of noise in sonar imagery, human input is often necessary to hand-select features in sonar imagery when completing environment reconstructions [38]. Specifically in this contribution we look at 3D point cloud reconstructions of the point features from underwater sonar images. The high framerate of imaging sonars, on the order of 20Hz, precludes such methods from generating reconstructions in real-time. In this chapter we present a CNN which enables a framework for producing accurate and real-time reconstructions. Our key insight for the CNN is the use of *atrous* convolution, which has a greater field of view than standard convolutional filters and thus makes the network better at overcoming the noise present in underwater sonar images. We show that by leveraging the power of deep learning while explicitly incorporating noise-robust atrous convolution, we can generate faster and more accurate reconstructions than previous methods in a variety of environments. [1]

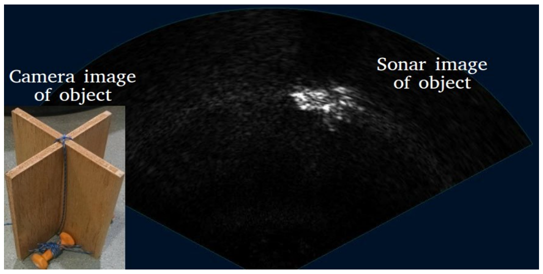
The 3D reconstruction of underwater environments has proven useful in a variety of applications, including ship hull inspection and underwater surveying tasks [37, 44]. As we have seen, sonar image noise comes in a variety of forms and strains image processing algorithms. One such example is multipath returns which occur when the projected beam completes a sequence of reflections other than simply to the object and back to the sonar (e.g., from the sonar to the seafloor, to the object, and then back to the sonar). Due to the increased length of time for the reflected wave to arrive back at the sonar, these reflections are often imaged at a range beyond the object. This can result in an image similar to Fig. 3.1b, where the horizontal component of the X appears thicker, due to delayed reflections from multipath reflections. Non-diffuse or specular reflection occurs when an object is smooth and fails to reflect the beam back in the direction it was emitted from. An example of this can be seen in Fig. 3.7i, where only the corners of the triangle are shown and the flat top surface does not return energy back to the sonar.

The substantial amount of noise present in sonar images is problematic for two reasons. First, noise will often corrupt an image so much that analyzing it further would waste time and computational resources. Our method provides a way to spend these resources more efficiently by only providing quality imagery to perception algorithms. Second, noise combined with the low-resolution of sonar imagery hampers the develop-

[1]We note that this work was published in a paper co-authored with A. Nicolai in R. DeBortoli, A. Nicolai, F. Li, and G. Hollinger, ”Realtime underwater 3D reconstruction using global context and active labeling,” in Proc. IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, May 2018, pp. 6204-6211. This thesis is concerned mainly with the image processing and atrous convolution aspect of the work.



(a) Frame with well-defined features (informative)



(b) Frame with object lacking well-defined features (non-informative)

Figure 3.1: Examples of informative (a human can confidently identify features) and non-informative frames while inspecting a target in the shape of an X. The sub-image in both figures is a camera image of the X shaped target that is insonified. In (a) feature correspondences (red) can be made between the camera image and sonar features (green). In our experiments we found on average only about 39% of the captured frames to be informative.

ment of an automatic feature extractor for sonar images. For example, feature extractors that use image gradients would incorrectly attempt to extract features from the gradients present in Fig. 3.1b.

The lack of an automatic feature extractor for sonar images has often necessitated human annotation of the features for 3D reconstructions [38]. Due to the high data rate of imaging sonars (on the order of 20 Hz) humans cannot annotate the frames in real-time. Additionally, we show that naively sub-sampling the data (to allow for real-time annotation) does not result in high-quality reconstructions because noise corrupts many of the images sampled.

The need for human annotation creates a long delay between data collection and environment reconstruction. To eliminate this delay and enable real-time reconstruction, we develop a method to recognize and propose only *informative* frames to the human

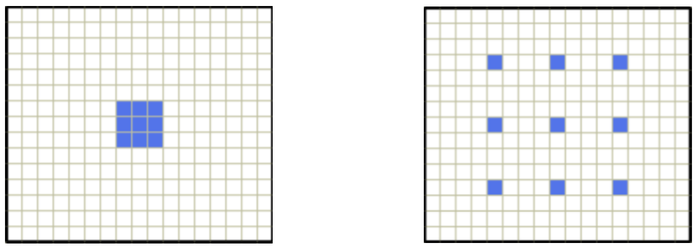

(a) Dilation rate of 1 (b) Dilation rate of 4

Figure 3.2: Example of traditional 3x3 filter (left) and a 3x3 dilated filter (right). The dilated filter uses a larger neighborhood, which compensates for the lack of strong local features in sonar imagery.

for annotation. We define an informative image as one that contains a set of clear and distinguishing features for environment reconstruction (as seen in Fig. 3.1a). This automatic identification, which is done in real-time, allows for a small set of frames to be presented to the operator. This set is small enough to be annotated in real-time.

To identify informative images in noisy and low-resolution sonar imagery we use dilated filters (e.g. Fig. 3.2b) in the atrous convolution architecture, previously used in the computer vision community for image segmentation tasks [13, 114]. Such filters use a large neighborhood of context in analyzing low-resolution and noise-filled sonar images.

We also focus on an approach that is flexible. Given the difficulty in obtaining and labeling sonar imagery, we pay special attention to not overfitting to the objects we train with. This is done by using the context afforded by dilated filters. To demonstrate this, we conduct experiments on imagery of objects not seen in the training set. We thus show that our approach is useful in the mapping or inspecting of objects not recorded in sonar previously.

In summary, for this first contribution we present a novel framework for underwater 3D reconstruction that:

- Automatically selects informative frames for an operator to annotate features, enabling the real-time reconstruction of underwater environments where the features must be manually selected.
- Utilizes the atrous convolution architecture to classify sonar images where features are often not well-defined or well-localized.
- Identifies objects not seen in the training set with a higher average precision than previous approaches.

We validate this framework on real sonar imagery containing a variety of objects in different environments.

The remainder of this chapter is organized as follows. We first discuss related work in underwater Structure From Motion in Section 3.1. We then discuss the reconstruction formulation in Section 3.2 and our method in Section 3.3. Next, we present our experiments and their results in Section 3.4. We conclude and propose areas for future work in Section 3.5. We note that this work appeared previously in [19], [20], and [53].

3.1 Background

3.1.1 Underwater Acoustic Structure From Motion

There has been much previous work in the area of using an imaging sonar for underwater 3D environment reconstruction. One popular approach in solving the underwater SLAM problem involves feature correspondence between pairs of sonar images. Hover et al. extract features from sonar images by clustering points in the image with large gradients [37]. A Normal Distribution Transform (NDT) is used for image registration, and the vehicle trajectory is optimized using a pose-graph. However, due to the unknown elevation angle of sonar image features, a planar assumption is used and the points are projected into the same plane as the vehicle. While this may work well for environments with large objects (e.g., sea floor, ship hull), this assumption is broken in more complex and unstructured environments.

The Acoustic Structure From Motion (ASFM) method proposed by Huang and Kaess provides two main advantages over other approaches [38]. First, ASFM relaxes the planar assumption for projecting sonar features into 3D. Second, their approach is capable of utilizing more than single pairs of sonar images to reconstruct 3D points. They formulate the reconstruction as a factor graph optimization problem. With the assumption of Gaussian noise, this can be formulated as a nonlinear least-squares problem. The factor graph is initialized by setting the sonar feature's unknown elevation angle to 0. Their proposed algorithm searches over a tree of potential feature correspondences, similar to Joint Compatibility Branch and Bound [64]. We leverage this work here to perform automatic feature correspondence and 3D reconstructions from hand annotated images.

We contribute a complementary framework for analyzing incoming imagery to allow for the human to select the features in real-time.

3.2 Reconstruction Formulation

We formulate the Acoustic Structure From Motion (ASFM) problem using a factor graph as proposed by Huang and Kaess [38]. An overview of the process is provided here, with a graphical representation shown in Fig. 3.3. For full details, please refer to [38, 45].

In formulating the reconstruction objective, we seek to find the maximal probability set of poses, $\Theta = \{b_i, l_j\}$, and observations, $Z = \{u_i, m_k\}$, where b_i is a robot pose, l_j is a landmark, u_i is a navigation measurement, and m_k is a landmark measurement. Assuming Gaussian noise, this can be formulated as a nonlinear least-squares problem:

$$\begin{aligned}\Theta^* = \underset{\Theta, Z}{\text{argmin}}[\ & \|b_0\|^2_{\Lambda} + \sum_{k=1}^{M} \|h(b_i, l_j) - m_k\|^2_{\Xi_k} \\ & + \sum_{i=1}^{N} \|g(b_i, b_{i-1}) - u_i\|^2_{\Lambda_i}\ ]\,,\end{aligned} \tag{3.1}$$

where $\|b\|^2_{\Sigma} = b^T \Sigma^{-1} b$ is the Mahalanobis distance squared and *M* and *N* are the number of sensor and odometry measurements respectively. The sensor model is defined as $h(b_i, l_j) + \mathcal{N}(0, \Xi_k)$ and the odometry model is defined as $g(b_i, b_{i-1}) + \mathcal{N}(0, \Lambda_i)$, where Ξ_k is the variance of sensor measurement k and Λ_i is the variance in odometry measurement i.

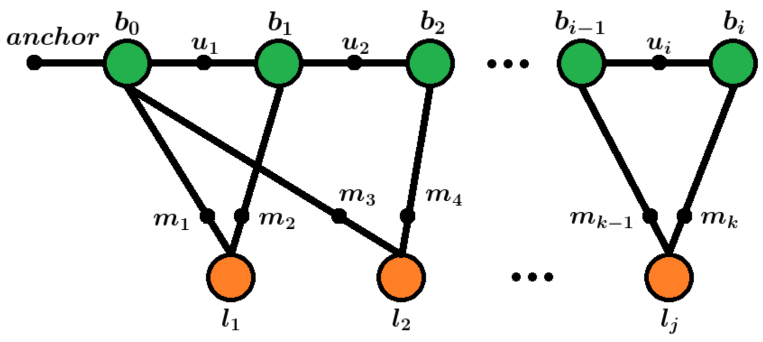


Figure 3.3: Factor graph representation. l, m, b, and u are the landmark positions, landmark measurements, robot pose, and navigation measurements respectively. The landmarks are hand-labeled features in sonar images (green dots in Fig. 3.1a).

In this formulation, there are six unknowns $(x, y, z, \alpha, \beta, \gamma)$ for every pose and three unknowns (x, y, z) for every landmark where α, β, and γ are the roll, pitch, and yaw respectively. With all landmarks visible from every pose, fixing the first frame yields a fully constrained system iff: $6(n-1) + 3m \leq 2mn$, where n is the number of robot poses and m is the number of landmarks in the factor graph. Rearranging this, we derive n, the number of frames necessary for reconstruction, to be

$$n \geq \frac{3m-6}{2m-6}. \tag{3.2}$$

While this formulation is well-suited to the reconstruction process, it does not directly support performing these reconstructions in real time. In developing such a solution, we introduce an image proposal system which gives a human operator a small set of frames containing information useful for reconstruction.

3.3 Method

3.3.1 Proposing Informative Frames

To select informative frames automatically we leverage the atrous convolution architecture. We motivate and briefly summarize our approach here, which first appeared in our prior workshop paper [19].

Due to the lack of strong local features in sonar images, the CNN method of analyzing sonar imagery can be greatly improved by using dilated filters (shown in Fig. 3.2b). Dilated filters use the same number of pixels as a standard filter; however because they are distributed, they can alleviate the effects of noise and low-resolution in sonar images.

The ability to ignore and not overfit to local features also improves the transfer learning capabilities in identifying informative sonar images. That is, given an image of an object not seen in training data, the atrous CNN is able to classify informative images with a higher average precision than a standard CNN or frequency-component based method. This ability is particularly attractive when using an underwater sonar, where there is a lack of training data as compared with terrestrial environments, and various sources of noise make similar looking objects appear different in sonar imagery.

3.3.2 Architecture Choice

To determine the appropriate architecture for configuring these dilated filters for use in analyzing sonar imagery, we test multiple architecture configurations and parameter choices. Full details of our selection process can be found in our workshop paper; a summary is below [19].

For the general architecture choice we evaluated three approaches. The first is similar to a normal CNN except the convolutional layers beyond the first have dilated filters instead of standard filters. This is inspired by Yu and Koltun, who use this configuration of dilated filters for dense prediction in image segmentation tasks [114]. The second pretrains two networks: one a standard CNN, the other an atrous network. The dense layers are then popped off of each and combined into a single dense layer for training again. This is meant to capture the expressibility of CNNs with the generalization capabilities of atrous networks. Finally, we test an architecture that uses filters of different dilation rates in the same layer. This is inspired by Chen, et al. who use this configuration for semantic image segmentation [13]. We complete a full parameter sweep on each architecture, including number of layers, number of filters, kernel size, dilation rate, and dense layer size. Average precision was used as the metric for comparing each architecture and parameter configuration, defined as:

$$AP = \sum_{n}(R_n - R_{n-1})P_n \tag{3.3}$$

where P_n and R_n are the precision and recall respectively at the nth detection threshold. To implement this metric computation, we use the scikit-learn $average\_precision\_score$ function [70]. To account for variability in initialization, the results for each were averaged over 20 runs. We found the architecture in Fig. 3.4 to achieve the highest average precision. Due to the large dilation rate, this network starts to extract non-local features as early as the second convolutional layer. We note that when describing features that the network extracts these are not necessarily the same as what a human would extract (e.g. points) but rather features the CNN derives based on the data it is given. The use of atrous filters allows the chosen network to avoid overfitting to local patterns in the training set and has better generalization capabilities.

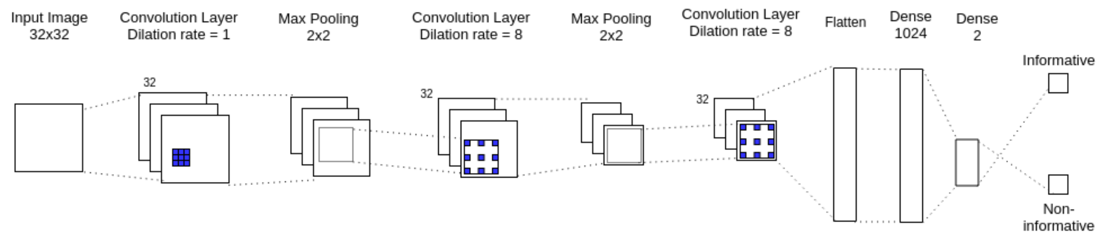


Figure 3.4: The atrous convolutional network used. The parameters and filter configuration were tuned by testing discrete options.

3.3.3 Frame Proposal Process

We treat the real-time frame proposal problem as a single-shot information-greedy selection. That is, every time a frame is proposed to the human operator, the most informative frame is selected from the set of candidate frames, $max(n_{info}) \in N$. The informativeness, n_{info}, of a frame is determined directly from the sigmoid output of our atrous convolution based deep network [19].

The set of candidate frames at any given time, N, is determined by two factors: n_{info} and the time since the last frame proposal, t_{motion}. Frame n_i is added to the set of candidate frames, $N \cup \{n_i\}$ if:

$$\begin{aligned} n_{info} &\geq T_{info} \\ t_{motion} &\geq T_{motion}. \end{aligned} \tag{3.4}$$

The threshold T_{info} allows for the tradeoff between frame information quality and frame quantity. The threshold T_{motion} is tuned to allow for view diversity while considering potential vehicle motion and sonar field-of-view.

Frames are proposed to the human operator sequentially as soon as the labeling of the previous frame is complete. When a frame is proposed, the set of candidate frames, N, is cleared. Frame proposal stops when either an appropriate number of frames, N_{thresh}, have been labeled or the data stream ends (e.g., the deployment concludes).

Once a frame is proposed, the human operator selects the features to be reconstructed by simply clicking on the image. We found this usually involved a small number of features (3-6), which allowed for fast feature selection and reconstruction. An example of selected features for the "X" shaped object are shown as green dots in Fig. 3.1a. An interesting avenue for future work is the investigation of the effects of the experience or training of the human labeler. We note that given the simplicity of the task, we do not expect large variations across annotators.

3.3.4 Threshold Tuning

We set the threshold for proposal, T_{info} at 0.99. As demonstrated in Eq. 3.2, with objects having on the order of five features, often less than ten frames are needed to

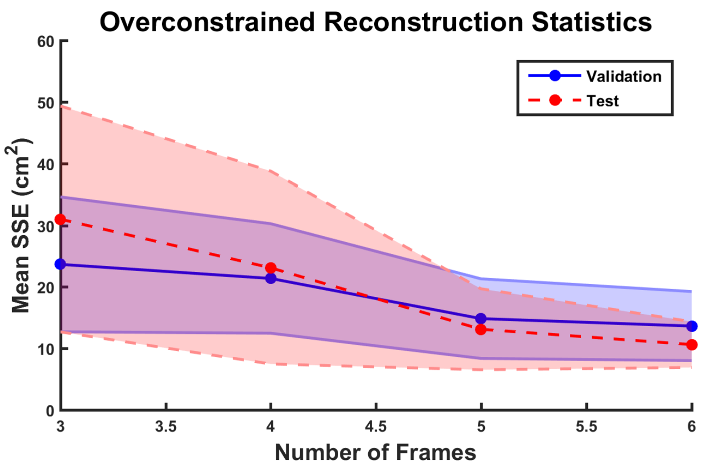


Figure 3.5: Selection of the appropriate number of frames to use for reconstructions. For both validation data (blue line) as well as test data (red line) five frames is at a “knee” point as increasing to six frames gives only a marginal benefit.

complete reconstructions. In these cases it is more beneficial to be selective in choosing frames (high threshold) rather than finding many informative frames. Experimentally this threshold provided more than enough frames to complete the reconstruction.

To determine N_{thresh}, we evaluated the reconstruction error for annotated set sizes of n (the minimum number of frames for reconstruction) to $2n + 1$. To obtain the average and variance on the sum squared reconstruction error (SSE) we use a set of data containing 7 proposed frames. From this set we test all of the combinations of frames for each annotated set size. These combinations are given by $\binom{2n+1}{k}$ where $2n + 1 = 7$ and $k = \{3, 4, 5, 6\}$. The same validation data used in completing the parameter sweep for the atrous network was used. As seen in Fig. 3.5, we find $N_{thresh} = 5$ to provide enough frames for a low reconstruction error/variance, with minor benefits increasing the set size to 6.

For the experiments we set $T_{motion} = 2$ sec. This threshold provided view diversity while allowing the objects of interest to remain in the sonar field-of-view. It also allowed the human operator to annotate each frame in real-time.

3.3.5 Reconstruction Process

Inspired by Huang and Kaess, initial landmark locations in the factor graph are calculated by setting the elevation angle ϕ to 0. The nonlinear least-squares problem is then optimized via iterative linearization using the Levenberg-Marquardt (LM) algorithm.

3.4 Experiments and Results

To demonstrate the capability of our framework to complete accurate reconstructions in real-time, we ran three experiments on real sonar data played back offline. The first shows the transfer learning capabilities of our atrous network when tested on imagery of objects not seen in training. In the second experiment, we show that we select an appropriate number of frames for our reconstruction. That is, we choose enough to fully constrain the optimization but do not propose many superfluous frames. Finally, the third experiment shows that our proposal method chooses a subset of frames that enable real-time reconstructions with errors lower than comparison methods.

Throughout each test we maintain a real-time property by ensuring that at the end of a given experiment, our algorithm has output the reconstructed shape. Experiments 1 and 2 are used only for analyzing components of our process and are thus not timed.

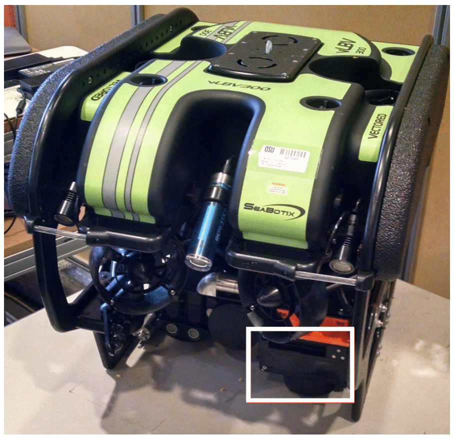


Figure 3.6: The Seabotix vLBV300 and Gemini 720i imaging sonar (designated by the white rectangle in the lower right).

Experiment 3 mimics a deployment and thus in Section 3.4.3.3 we provide runtime from start to finish of our process. For completeness we provide average numbers for each component of our system. Our CNN processes images in 18.1 ± 0.8 ms (55.2 ± 2.41 Hz). We found that a human operator annotated images of the X object in 2.63 ± 0.231 sec. Finally, the factor graph bundle adjustment process took on average 77.1 ± 9.4 ms.

3.4.1 System Overview

For our experiments we use a Tritech Gemini 720i multi-beam sonar onboard a tethered Seabotix vLBV300 Remotely Operated Vehicle (shown in Fig. 3.6).

The Gemini 720i is a standard multi-beam imaging sonar with similar parameters to the SoundMetrics DIDSON, BlueView M900-90, and the Aris Explorer 3000, all of which have been used in previous work [38, 55, 43]. It operates at 720kHz, images a 120° swath horizontally, has a maximum range of 120 m, and produces images at 20Hz.

This data rate, combined with noise from a variety of sources, creates a situation in which the human operator is presented with a large number of frames, many of which will not be useful.

Our vehicle is also equipped with a Doppler Velocity Log and Inertial Navigation System which we use for pose estimates in the factor graph mapping scheme. The vehicle's tether provides continual power as well as data transmission back to an offboard computer which can be used to analyze sonar imagery in real-time. The use of standard underwater sensors demonstrates our approach can be deployed on a variety of platforms for real-time underwater reconstructions.

3.4.2 Dataset Overview

In this work we use four datasets captured onboard our vehicle while it operated in a variety of environments. Each dataset was collected in a passive manner, meaning each dataset is a single contiguous video stream captured while our vehicle moved through underwater environments. The content and number of informative frames varied across datasets and is summarized in Table 3.1.

In dataset X_1 we collect a set of 5000 frames of the X shaped object (Fig. 3.7a) in the Oregon State University pool. In dataset X_2 we collect 1107 frames of the same X object in the same environment on a different date. In dataset $Multi$ we collect a set of 5000 frames containing insonified imagery of the X, Square, Triangle, and T-shaped objects (Figs. 3.7a-3.7d) in the pool. This dataset contains images with just a single object in addition to images containing multiple objects. Finally, in dataset $Cinder$ we

Table 3.1: Summary of the datasets

Dataset	Total number of frames	Percent informative frames
X_1	5000	36%
X_2	1107	56%
$Multi$	5000	39%
$Cinder$	1470	42%

Note: An informative frame is one in which a human operator can clearly identify an object and its features in the sonar image.

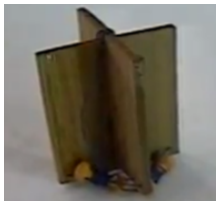
capture 1470 frames of a cinder block target (Fig. 3.7e) in Yaquina Bay, Newport, OR. While in this work we use representative shapes for the underwater domain (resembling objects such as underwater moorings) an interesting avenue for future work involves the use of our architecture on more complex shapes.

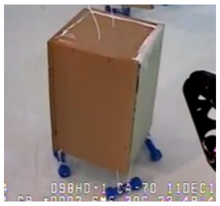
The binary ground truth label (informative or non-informative) is obtained by having an expert evaluate if each image contains enough well-defined features to clearly identify the object. For example, simply seeing two lines intersect does not identify one of the five targets; however observing three well defined lines that each meet in an acute fashion clearly defines the triangle object seen in Fig. 3.7d.

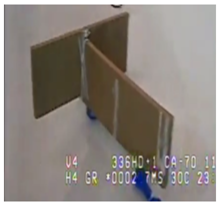
Throughout the three experiments we train primarily using datasets X_1 and X_2 (both only containing data of the X target in Fig. 3.7a). 1000 frames of the $Multi$ dataset are used as validation data for the tuning of parameters.

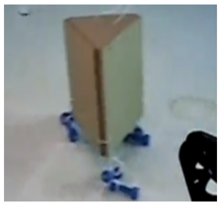
3.4.3 Experiments

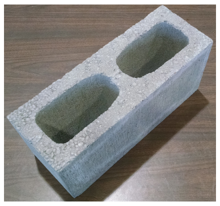
We conduct three experiments to demonstrate that our framework selects informative subsets of images thus enabling real-time reconstructions. In the first experiment we test

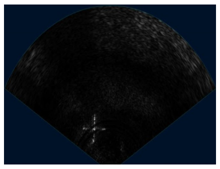

(a) X target

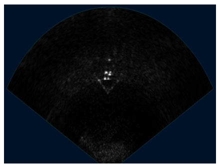

(b) Square target

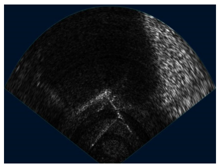

(c) T target

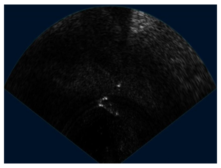

(d) Triangle target


(e) Cinder block target


(f) X in sonar


(g) Square in sonar


(h) T in sonar


(i) Triangle in sonar

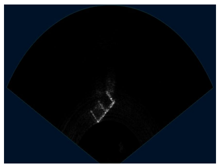

(j) Cinder block in sonar

Figure 3.7: Each target and its representation in sonar image space. (a)-(d) are images taken from the vehicle underwater. As a note, in Fig. 3.7j the second long side of the cinder block has been occluded.

using 4000 frames of the *Multi* dataset (the remaining 1000 frames were used as validation data). In Experiment 2 we use these same frames as well as the *Cinder* dataset. In Experiment 3, we only use data from the *Cinder* dataset. Unless stated otherwise, test data for each experiment contains imagery of objects not trained on, thus demonstrating the flexibility or lack thereof of each approach examined. This also serves to demonstrate the performance of our approach when trained prior to deployments.

3.4.3.1 Validating the transfer learning capabilities of our atrous network

In the first experiment we show that our model is able to identify objects not in the training set with a higher average precision than baseline methods. We are able to do this while also classifying objects seen in the training set with precision comparable to baseline methods. This ability is a result of our atrous network, which balances the power of standard convolutional architectures with the more global context afforded by dilated filters.

We compare against three baselines. The first is the Discrete Cosine Transform (DCT) method of global analysis. Defined for pixels m, n of an image A of size $M \times N$ as:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} cos \frac{\pi(2m+1)p}{2M} cos \frac{\pi(2n+1)q}{2N}, \tag{3.5}$$

where

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, p = 0 \\ \sqrt{\frac{2}{M}}, 1 \leq p \leq M-1 \end{cases} \tag{3.6}$$

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, q = 0 \\ \sqrt{\frac{2}{N}}, 1 \leq q \leq N-1 \, , \end{cases} \tag{3.7}$$

and B_{pq} is the coefficient on the basis cosine function defined by the constant *p* and *q* [59]. The resulting coefficients represent the frequency components contained in the image, which can be used to separate noisy images from those with more complex geometries. We complete this classification with the use of a Support Vector Machine using the frequency components as the input data. This method is natural for sonar imagery because it takes a global image analysis approach to imagery with poorly-defined features. The second baseline is the approach taken by Kim, et al. (called “two-layer CNN”) [47]. The third and final baseline is a network we developed inspired by Kim, et. al that contains an additional convolutional layer (called “three-layer CNN”).

We compare the four approaches in three scenarios, the results of which are shown in Table 3.2. The first scenario allows each classifier to train with 600 frames of multi-object data (called "seeding"). The test data contains imagery of each of the four objects in Figs. 3.7a-3.7d (including the X). These results show that the DCT method of global analysis is not expressive enough to capture the features present in both the training and test data. The two and three-layer CNNs, as well as our method, all perform similarly.

The second scenario still seeds the training of the models, but the test data no longer contains imagery of the X object. We find that our model outperforms the others, which is expected because dilated filters capture a larger neighborhood of influence than localized filters.

Finally, in the third scenario, true transfer learning is evaluated. The models are not given imagery of any object but the X to train on and the test set only contains imagery of the Triangle, Square, and T shaped objects. In this scenario we find that our model significantly outperforms other methods, which is intuitive as dilated filters give a nice balance between the power of deep learning based methods and the larger context of global evaluation methods. As demonstrated in the first scenario, this does not come at the cost of performing worse than baselines in the case where the test data is similar to the training data.

3.4.3.2 Choosing the appropriate number of frames to propose

In the second experiment, we show that our method proposes an appropriate number of frames. That is, our method achieves lower reconstruction error than using the minimum

Table 3.2: Average precision for baselines and our method

Method	Avg. Precision		
	Seeding X present	Seeding No X	No Seeding No X
DCT [59]	0.57±0.01	0.55±0.02	0.48±0.02
Two-layer CNN [47]	0.69±0.04	0.58±0.05	0.42±0.04
Three-layer CNN	0.70±0.05	0.64±0.04	0.42±0.03
Atrous Convolution (ours)	**0.72±0.02**	**0.68±0.03**	**0.54±0.04**

number of frames to fully constrain the reconstruction (n) as well as achieving comparable performance to reconstructions that use a large number of frames ($2n$). Given the lack of global ground truth coordinates, known object dimensions are used to compute the reconstruction error.

The results of this experiment can be seen in Fig. 3.5. As expected, increasing the number of frames used in the reconstruction decreases the reconstruction error. When only using the minimum number of frames required, n, we can see that on average the reconstruction has both the highest sum squared error (SSE) as well as the highest variance. At five frames in both plots we observe a "knee" point, where increasing the size to six gives only marginal benefit.

3.4.3.3 Demonstrating the ability of our network to select informative subsets

In the third experiment, we demonstrate the ability for our method to select informative frames and operate more efficiently than standard baselines. For this experiment we use a subset of 400 contiguous frames. The results of this experiment can be seen in Table

3.3. Statistical bounds have been provided for the comparison to [39] due to the random nature of frame proposals in this baseline. These bounds were found averaging over 5 runs.

The first baseline we compare against is naively presenting the user with every frame for annotation. We note that this method requires the human operator to annotate every frame and thus cannot be completed in real-time. The large amount of error is due to the fact that many poor quality frames are annotated by the user. Such large error demonstrates the low-quality of many of the images, which do not contain enough clear features to be annotated by the human.

The second baseline we test is proposing every informative frame to the user for annotation. The informativeness of a frame is determined by the sigmoid output of our atrous CNN. This baseline achieves a reconstruction error similar to that of our method; however we note that it cannot be run in real-time. The low reconstruction error is expected because the labeled images are all informative frames.

In the third baseline, we naively subsample all frames at an even interval to only present the human operator with the same number of frames as our method. This is equivalent to naively reducing the data rate to allow for real-time labeling. While this allows for real-time performance, we can see that the reconstruction error achieved is poor. This is due to the fact that without first assessing a frame's informativeness, poor quality frames are still presented to the user.

The fourth baseline allows us to compare against a method inspired by the state-of-the-art in underwater reconstruction [39]. In this previous work, Huang and Kaess perform automatic data association between annotated frames to determine whether or

Table 3.3: Evaluation of our reconstruction framework

Method	**Number of frames proposed**	**Reconstruction SSE (cm^2)**
Every frame	400	8761.9
Every informative frame	167	16.37
Naive subsampling	**5**	71.239
Huang and Kaess [39]	6.93±1.91	18.42±6.47
Atrous proposal system (ours)	**5**	**13.12±3.72**

not all annotated feature points can be associated. If they cannot, the frame is rejected and not used for reconstruction purposes. To extend this approach and enable real-time reconstructions, we randomly select frames to propose to the human operator. If an association exists between the annotated feature points, the frame is used in the reconstruction. If not, it is discarded and another is proposed. The proposal process runs until the minimum number of frames required, n, have been successfully annotated. While this method is able to achieve high quality reconstructions, the reconstruction error is both higher and more variable than our method. This is due to the fact that only the minimum number of frames required are used in the reconstruction process. We also note that since frames are not evaluated before proposal, this baseline typically proposes a larger number of frames than our method.

The last row of the table presents the results of our method. We note that we achieve the lowest average reconstruction error and variability, while simultaneously proposing the lowest number of sonar frames to the user. This low reconstruction error is achieved while using N_{thresh} = 5 frames. We also maintain real-time performance during the 20 second experiment. Our method, from start to finish, took on average 14.88 ± 0.49 seconds to complete.

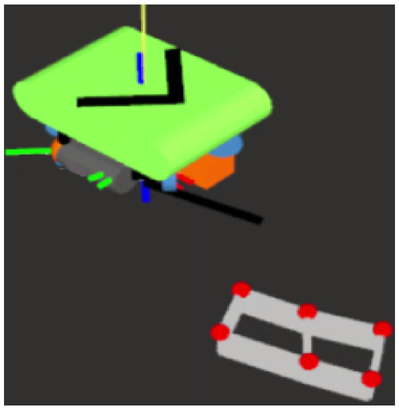


Figure 3.8: The vehicle and reconstruction of the cinder block in the RViz simulation environment. Our method outputs the red dots (chosen as features by the operator) and the gray section was extrapolated.

3.5 Conclusion

In this chapter, we presented a novel framework that enables accurate underwater 3D reconstructions to be generated in real-time and overcome the large amounts of noise present in sonar imagery. We achieve this functionality by identifying a small subset of informative frames for the human to annotate. Through experiments on real sonar images we show our atrous network outperformed other classifiers in identifying informative frames for proposal. We also experimentally validate the ability of our network to derive non-local features to complete transfer learning.

By validating the benefits of atrous convolution for sonar image analysis, we demonstrated that to perform perception tasks in harsh environments, it is beneficial to explicitly account for the sensor noise present in alternative sensing modalities. We also demonstrated that the speed of deep learning can enable real-time 3D underwater reconstructions. In the next chapter we will develop techniques for removing the human from the loop and allowing them to focus on more high-level mission objectives.

Chapter 4: Fully Automated Real-Time Underwater 3D Reconstruction

In our first contribution we demonstrated the power of deep learning in being a noise-robust image analysis tool for 3D underwater reconstruction with a human-in-the-loop. In our next contribution, we propose an alternative approach that enables the human to be removed from the reconstruction process and focus on higher-level mission objectives. We propose the use of a deep network as a powerful and noise-robust estimator for completing 3D reconstructions. Specifically in this contribution we address the problem of *dense* 3D reconstruction, that is, for every pixel in the 2D underwater sonar image, output an estimate of its location in the 3D world. Our approach generates estimates at a pixel level and thus avoids the use of shape priors or explicitly detecting the shape. We note that our approach does not dismiss the benefits of atrous convolution demonstrated in the next section, but rather provides an alternative solution to being noise robust. An interesting future research direction would be to combine atrous filters with our dense reconstruction approach presented in this chapter.

Removing the human from the loop forces the deep network to complete more of the reconstruction task, which consequently requires more data for training convergence. Unlike in terrestrial applications, such as autonomous driving, training data for underwater reconstructions is extremely scarce. Therefore in this contribution, our key insight is to intelligently incorporate synthetic data and the physics of the sonar sensor into the training process. [1]

4.1 Background

In spite of the advantages of 2D imaging sonars, including long-range visibility, low weight, and fast data capture, the use of such sensors for 3D reconstruction has been limited due to difficulties in recovering the missing dimension, the *elevation angle*, from 2D sonar 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 [3, 102]. 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 angle dimension [5, 30]. 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.

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

[1]This work initially appeared in our prior conference paper: R. DeBortoli, L. Fuxin, and G. Hollinger, ”ElevateNet: A convolutional neural network for estimating the missing dimension in 2D underwater sonar images,” in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, China, Nov. 2019.

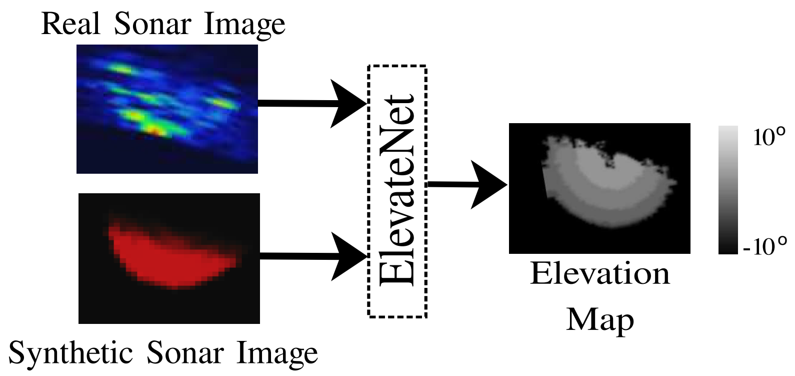


Figure 4.1: Inputs and outputs for ElevateNet. ElevateNet takes in 2D sonar images (left) and produces pixelwise estimates of the elevation angle. The network can be trained in a fully supervised manner when ground truth information is available, as is the case with synthetic data. Our method can also train in a self-supervised manner on real sonar images, where ground truth information is not available.

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 [21, 25]. In many recent works, 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 [25]. 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 [25]. An actual image of that scene from that 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. We first propose a method which takes as input a single sonar image and produces and elevation angle estimate. We then propose an extension that can leverage the information contained in multiple input images taken at different viewpoints to produce more accurate elevation angle estimates. 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 as the backbone for this work 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. 4.1 shows an overview of our method named "ElevateNet".

In short, in this contribution we develop:

- 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.
- Two multi-view extensions to ElevateNet each leveraging the additional information provided by multi-image inputs to produce more accurate elevation angle estimates than its single-view counterpart.

4.2 Elevation Angle Estimates from a Single Viewpoint

We first describe our single-view approach, called *ElevateNet*, for estimating the missing elevation angle given a single 2D underwater sonar image. In this section we develop a

CNN and training architecture which produces accurate 3D reconstructions and which becomes the basis for our multi-view approach described in Section 4.3.

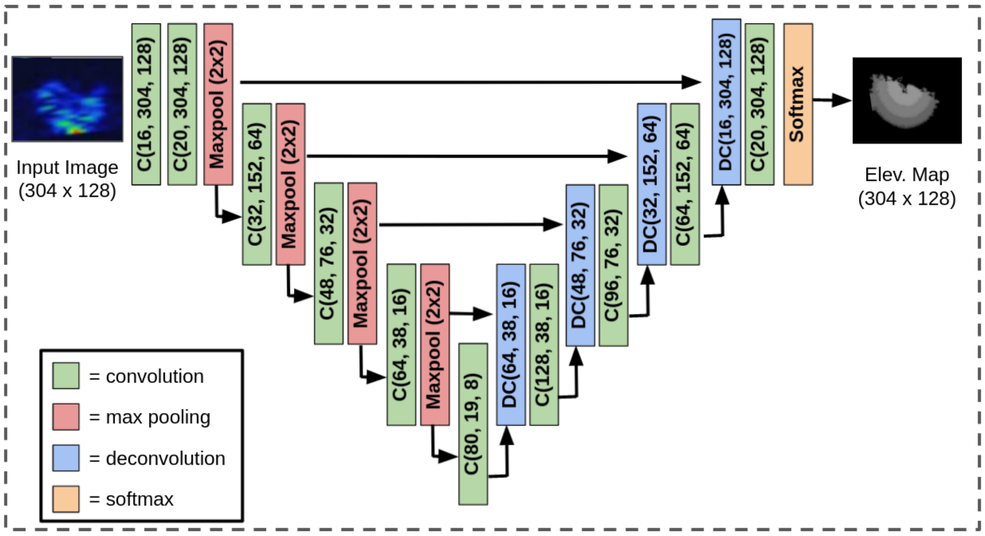
4.2.1 Methods

We now 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. 4.2a) as the $model$ and the process by which to incorporate real data (Fig. 4.2b) as the $architecture$.

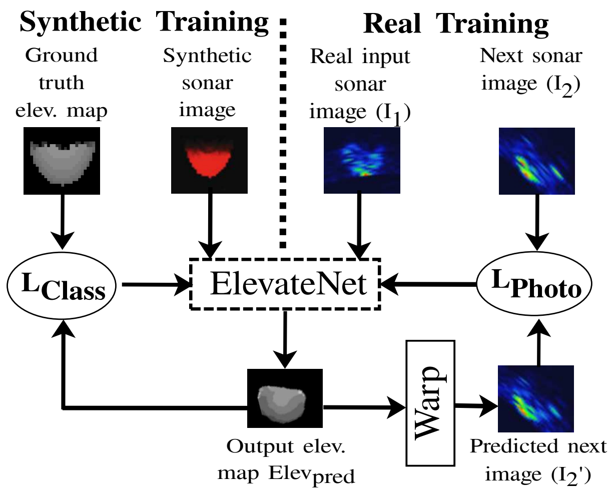
4.2.1.1 Model

We first discuss the model, shown in Fig. 4.2a. 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 [78].

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.



(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.

Figure 4.2: 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.

4.2.1.2 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. 4.2b. 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):

$$X_c = R \cos\phi \sin\theta, \tag{4.1}$$

$$Y_c = R \cos\phi \cos\theta, \tag{4.2}$$

$$Z_c = R \sin\phi, \tag{4.3}$$

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. [42].

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.

4.2.1.3 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 [68]) 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. 4.2a.

4.2.1.4 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:

$$Loss_{photometric} = \sum_{i=0}^{Y} \sum_{j=0}^{X} M \left| I_2'[i,j] - I_2[i,j] \right|, \tag{4.4}$$

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.

4.2.2 Experiments and Results

In this section we first 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 data enables more accurate 3D reconstructions on real data. In this contribution's experiments we use the same Seabotix vLBV300 vehicle as our first contribution; an overview of which can be found in Chapter 3.4.1.

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.

4.2.2.1 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.

Synthetic dataset We generate synthetic data using the simulator by Cerqueira et al. which models the same Tritech Gemini 720i sonar on our vehicle [11]. 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.) [61, 105]. 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).

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:

$$I_n = (I - \mu)/\sigma, \tag{4.5}$$

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 4.2.1.3.

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:

$$MAE = \frac{1}{|N|} \sum_{n \in N} |r_n \sin(\phi_{p,n} - \phi_{t,n})|, \tag{4.6}$$

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

Table 4.1: Synthetic Pixelwise Elevation Angle MAE (m)

	Entire test set			Sphere		
Min. dist. to training ex. (cm)	<10	10-20	20-30	<10	10-20	20-30
$Pred_{random}$	0.160	0.170	0.180	0.200	0.190	0.183
$Pred_{zero}$	0.114	0.112	**0.120**	0.140	0.120	**0.114**
CNN (ours)	**0.087**	**0.105**	**0.120**	**0.100**	**0.110**	0.126

	Cube			Cylinder		
Min. dist. to training ex. (cm)	<10	10-20	20-30	<10	10-20	20-30
$Pred_{random}$	0.213	0.211	0.206	0.0673	0.068	0.187
$Pred_{zero}$	0.145	0.135	0.135	**0.0527**	**0.0545**	**0.146**
CNN (ours)	**0.105**	**0.109**	**0.105**	0.0635	0.0657	0.186

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 4.1. 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.

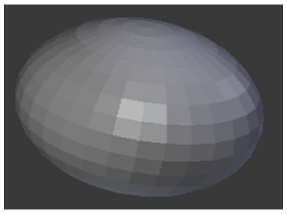
For the sphere, our method performs well when the test poses are close to the training data. Interestingly, it maintains its performance even up to 0.2 m away from any training pose. As the test data gets far away from the training data, $Pred_{zero}$ performs better, suggesting that we are starting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

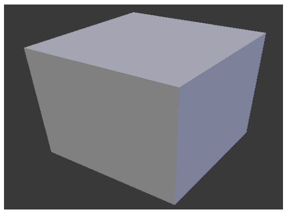
Our low error on the cube demonstrates the ability for our method to perform well on simple objects. Even as we move into test views that are far away from any given training view (up to 0.3 m difference), our method still outperforms competing methods by a large margin.

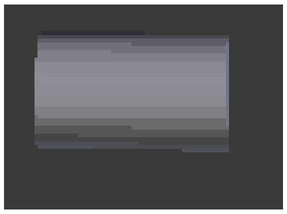
Finally, we find that our method is unable to beat the $Pred_{zero}$ baseline on the cylinder. Further analysis suggests that the network appears to confuse the cylinder with a cube and thus predicts a plane instead of a curved surface. The reasoning for this could be that the cylinder and cube objects (Figs. 4.3e and 4.3f) look very similar in image space.

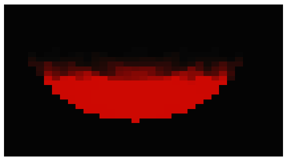
Synthetic 3D Reconstruction Results We next demonstrate that our method can produce high-quality reconstructions using the estimated elevation maps. We compare our method and the $Pred_{zero}$ baseline on their 3D reconstruction abilities in a manner similar to previous work [5]. For brevity, we do not include results from $Pred_{random}$.

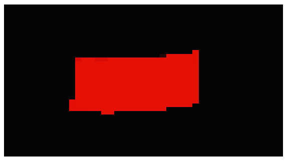
In this experiment we also add the aforementioned state-of-the-art space carving method proposed by Aykin and Negahdaripour [5]. While this method relies on multiple views, it is to our knowledge the only method for dense reconstruction that does not require an environmental prior (e.g. shadow on the 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 which is the same order of magnitude as the results presented in the original paper and is thus representative of a working method.

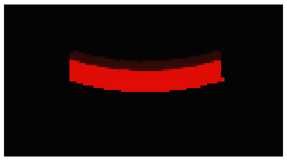

(a) Sphere target

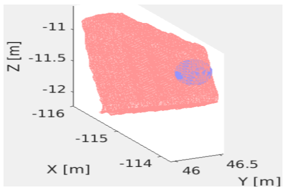

(b) Cube target


(c) Cylinder target

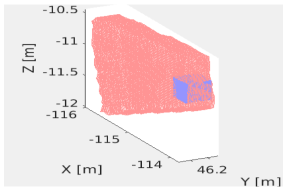

(d) Sphere image


(e) Cube image

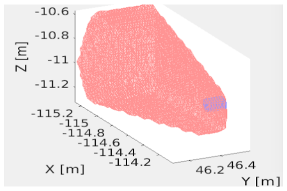

(f) Cylinder image



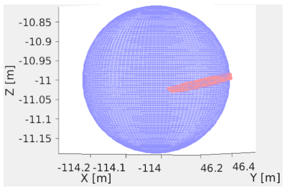
(g) Reconstruction of sphere [5]



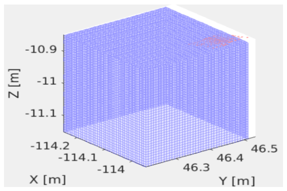
(h) Reconstruction of cube [5]



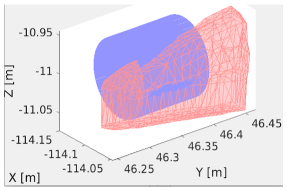
(i) Reconstruction of cylinder [5]



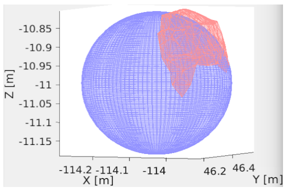
(j) Reconstruction of sphere (zero)



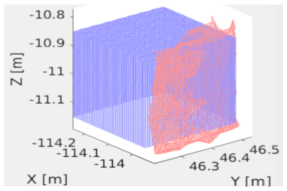
(k) Reconstruction of cube (zero)



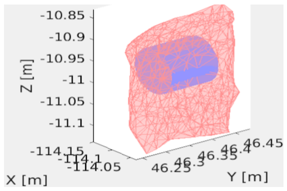
(l) Reconstruction of cylinder (zero)



(m) Reconstruction of sphere (ours)



(n) Reconstruction of cube (ours)



(o) Reconstruction of cylinder (ours)

Figure 4.3: 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. 4.3g-4.3o, the ground truth meshes are blue and the predicted meshes are red. The predicted mesh in Fig. 4.3k 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.

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 generated meshes are for visualization purposes only and quantitative metrics for comparing methods are computed over point cloud representations only. From these limited viewpoints 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. 4.3g-4.3o. Across all three shapes, the space carving method is unable 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 4.2: Synthetic Reconstruction RMSE Results (m)

	Sphere	Cube	Cylinder
Space carving method [5]	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 generate a ground truth point cloud we sample the surface of the ground truth CAD model uniformly. To compute point correspondences nearest-neighbor search is used. 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 4.2. As can be seen in Figs. 4.3j-4.3l, $Pred_{zero}$ 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.

4.2.2.2 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.

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. 4.4a-4.4b) using Quikrete concrete mix. The sphere has radius 19 cm, the cylinder radius 5 cm and height 10 cm, and the cube side length of 29 cm. We note that variations in size between the simulated and real datasets were due to the mold sizes available for constructing the real targets. 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 contribution using a CNN, could be incorporated in the future [20]. We then resize images to 304x128 pixels and normalize according to Eq. 4.5. 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. 4.4d-4.4e.

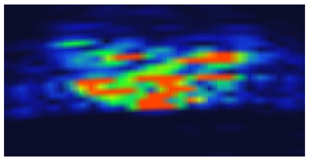
For training and final testing, we break up the dataset into 70% for training, 15% for validation and 15% for test. To ensure independence between validation and testing, the test set contains data from a separate deployment (same day) as the training and validation data.

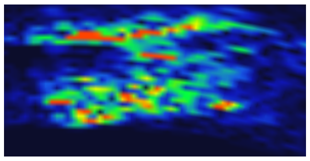
During training on real data, we investigated the effects of the training schedule (what epochs to train with real or synthetic data). We found that training for 15 epochs

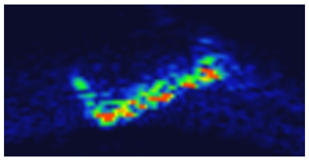
(a) Real sphere target

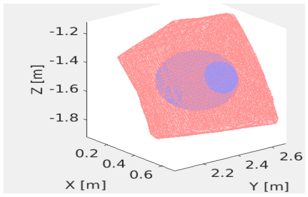
(b) Real cube target

(c) Real cylinder target

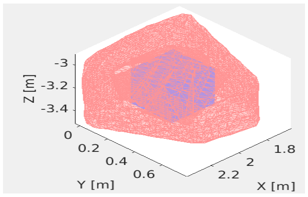

(d) Real sphere image


(e) Real cube image

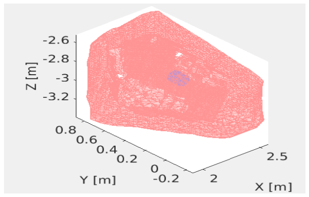

(f) Real cylinder image



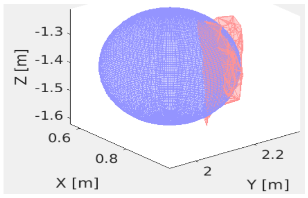
(g) Reconstruction of sphere Aykin [5]



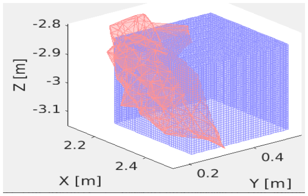
(h) Reconstruction of cube Aykin [5]



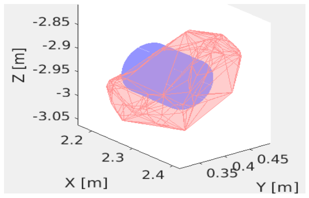
(i) Reconstruction of cylinder Aykin [5]



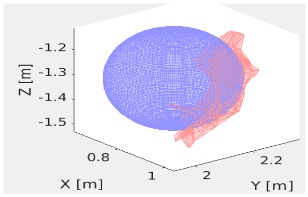
(j) Reconstruction of sphere ($ours_{syn}$)



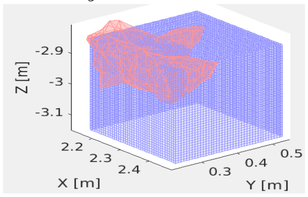
(k) Reconstruction of cube ($ours_{syn}$)



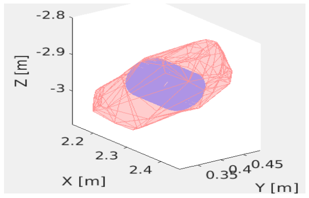
(l) Reconstruction of cylinder ($ours_{syn}$)



(m) Reconstruction of sphere ($ours_{real}$)



(n) Reconstruction of cube ($ours_{real}$)



(o) Reconstruction of cylinder ($ours_{real}$)

Figure 4.4: The real sonar targets (a)-(c), their respective sonar images (d)-(f), and the reconstructed meshes using Aykin [5] (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.

of synthetic data was enough to start pre-training, and then 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 4.2.1.3 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. 4.2a) 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 4.2.1.4.

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 4.3 shows our results. As shown in Eq. 4.4, we scale the photometric loss to be roughly the same as the synthetic loss; consequently the photometric loss is unitless.

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

Table 4.3: Evaluation of predicting the next frame

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

Table 4.4: 3D reconstruction accuracy on real data

	RMSE (cm)		
	Sphere	Cube	Cylinder
Space carving [5]	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

ground truth location of the real object, point clouds are first manually hand-aligned to a ground truth model using a rigid transform. Then Iterative Closest Point (ICP) is run and the RMSE is computed using the MATLAB *pcregrigid* library. We note that because we do not know the ground truth location of the object being imaged, metrics comparing position/orientation estimates are not appropriate for our task. We increased the MaximumIterations parameter from 25 to 500 and left all other parameters default.

In addition to the space carving baseline [5] 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 4.4. 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. 4.4. 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.

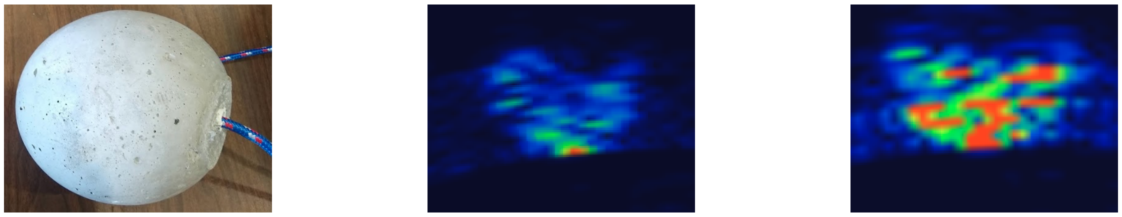
There are interesting qualitative differences between $ours_{syn}$ and $ours_{real}$. For the sphere, $ours_{real}$ better captures the curved shape of the sphere. While estimating the cube surface, $ours_{syn}$ produces a variety of surface shapes. Shown here is what would be expected from training on the synthetic data: a vertical surface. Interestingly, $ours_{real}$ produces many “corner” shaped objects shown in Fig. 4.4n. This makes sense because many of the real images of the cube, including Fig. 4.4e, 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.

4.3 Elevation Angle Estimates from Multiple Viewpoints

While we demonstrated that a single-image input to ElevateNet could produce accurate reconstructions, as the robot traverses the environment, it is collecting images continuously that could potentially have complementary information. In other robotic applications, incorporating multiple frames as input into deep learning models has been shown to improve estimation abilities over just using a single frame. For example, Schenck and Fox use a multi-frame RGB CNN to improve liquid detection performance during a robotic liquid pouring task [80].

Underwater, we find a similar high-level scenario: as the robot traverses the environment and collects information, different images can provide complementary information. For example, in Fig. 4.5 we can see that different images taken at different poses have different overall pixel intensities, suggesting the object being imaged is at a certain location in the environment. However, different than applications using RGB images, there is a distinct lack of training data available for 2D underwater sonar images.

Therefore, in this section we leverage the ElevateNet architecture for training and develop approaches to leverage the additional information contained in multi-view inputs to produce more accurate 3D underwater reconstructions. We first describe our two approaches, the training approach for each, and finally our results on simulated and real sonar data.



(a) Image of sphere object

(b) Sphere away from center of elevation arc

(c) Sphere close to center of elevation arc

Figure 4.5: Intuition for leveraging multiple frames. In Fig. 4.5b the sphere does have strong returns in the image, indicating that the object is not near the center of elevation angle arc. In Fig. 4.5c the sphere shows up very strong in the image, indicating that that object is near the center of the elevation arc. Known poses between these images can help to reduce ambiguities in the elevation angle estimation.

4.3.1 Methods

In this section we first describe an extension to ElevateNet called *Multi-View Multi-Channel (MVMC) ElevateNet* which only adapts the model input format and keeps the overall model structure very similar. We then describe an extension called *Multi-View Late-Fusion (MV-LateFusion) ElevateNet* which adapts both the input data format and the actual model architecture.

4.3.1.1 Multi-View Multi-Channel Inputs

Given the success of the ElevateNet model architecture on 3D reconstructions, we first present a method that keeps the overall model structure the same and augments the only the input data format. To incorporate the information from multiple images, we take the images from different times and viewpoints and stack them together into a multi-channel input. Thus, while ElevateNet takes inputs of shape (*height*, *width*, *1*), our

MVMC approach takes inputs of shape (*height*, *width*, *num_images*) where *num_images* is a parameter that can be tuned during training. We note that each image is spaced evenly *step_size* timesteps apart, where 1 timestep is 1/20 seconds.

An overview of our MVMC approach can be seen in Fig. 4.6a. This approach only requires the concatenation of input images, making it simple to implement and avoids complex model structures which may be difficult to implement or may overfit.

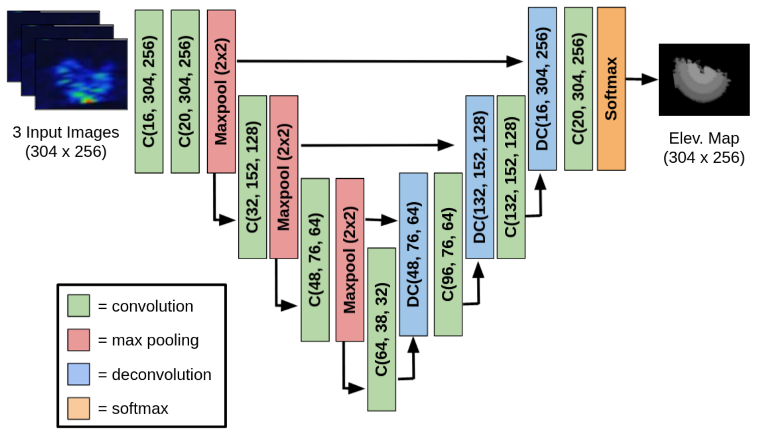
4.3.1.2 Multi-View Late-Fusion Network

Previous work has demonstrated that in applications working primarily with RGB images, *late-fusion*, or extracting images features from individual images before combining the feature, achieves better performance than just channel stacking the images. Inspired by these benefits shown in ther applications, we develop a Multi-View Late-Fusion (MV-LateFusion) adaptation of ElevateNet.

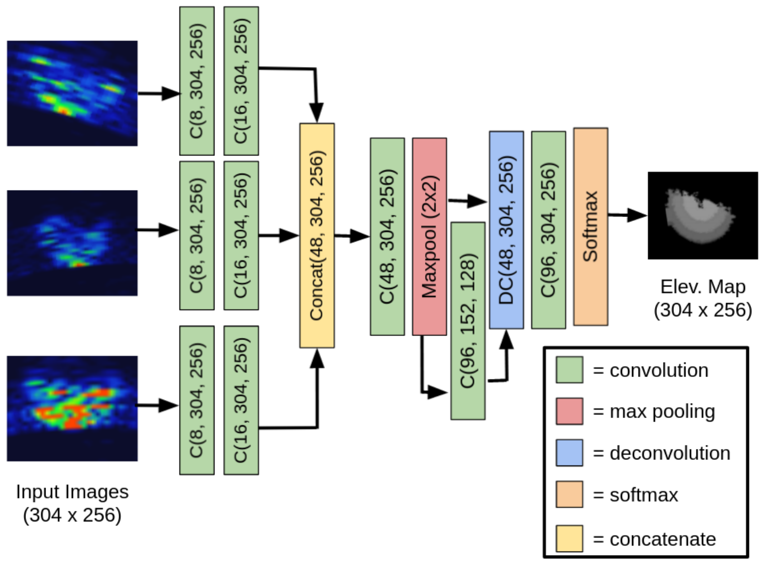
An overview of our MV-LateFusion approach is shown in Fig, 4.6b. Each input image is first passed through a series input convolution blocks after which the features are concatenated, processed further by convolutional block, and finally upsampled to produce an elevation angle estimation.

4.3.2 Multi-View Training Process

We next describe the overall training process for both the MVMC and MV-LateFusion models on synthetic and real data. We note that at a high-level the overall training



(a) MVMC model. 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) Multi-View Late Fusion Model. 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.

Figure 4.6: Our two proposed multi-view models. In the top figure, the overall MVMC model structure is very similar to ElevateNet, with the only large difference being the input data being a 3-channel image. In the bottom figure one can see the structural differences between MV-LateFusion as compared with the standard ElevateNet approach.

process is similar to that of ElevateNet (Sections 4.2.1.3 and 4.2.1.4) and therefore we focus mostly on the differences between the single and multi-view training processes.

4.3.2.1 Training with Synthetic Data

We start the training process by using synthetic data which has ground truth elevation angle information available, which provides a strong training signal. To compute a loss function and enable training, we designate one of the ground truth images from the set of input image/ground truth elevation map pairs to be used as the goal of the final prediction. We denote this ground truth elevation map $Elev_k$ where $0 \leq k \leq num\_images$. In practice we used $k = num\_images/2$ so there is image information both before and after the time of estimation. For both the MVMC and MV-LateFusion loss functions we use the L1 difference between the predicted and real elevation maps, defined as

$$Loss_{syn} = \sum_{i=0}^{Y} \sum_{j=0}^{X} \left| Elev'_k[i,j] - Elev_k[i,j] \right|, \tag{4.7}$$

where $Elev_k$ is the ground truth elevation map and $Elev'_k$ is the predicted elevation map, and X and Y are the image height and width respectively. We note that we found the L1 loss function to achieve better validation accuracy than the cross entropy loss function used in SV-ElevateNet.

For both MVMC and MV-LateFusion we completed a parameter sweep over the batch size, learning rate, filter size, number of convolutional layers, pooling method, input image size, number of layers and *num_images* values. For both MVMC and

MV-LateFusion we found a learning rate of 5E-4, filter size of 3x3, batch size of 128, *num_images* of 3, $step_{size}$ of 5, input image size of 304x256 and 2x2 average pooling to work best on validation data. It is interesting to note that we experimented with larger and deeper networks, but this did not improve performance, which is intuitive given the low resolution of underwater sonar images. The final parameter choices for model structure are presented in Fig. 4.6.

4.3.2.2 Training with Real Data

Due to the high-levels of noise present in real underwater sonar imagery, we also train with real data. We follow the same overall training architecture as in ElevateNet (Fig. 4.2b) where we first use synthetic data for *num_syn_only* epochs to bring the model into an appropriate part of the parameter space, and then alternate real and synthetic training. While training with real data there is no ground truth elevation angle information and therefore we once again use the L1 difference between the predicted next image (I'_2) and the actual next image (I_2) where I2 is an image *step_size_real* timesteps away from the current input.

Once again we completed a parameter sweep over the batch size, learning rate, filter size, number of convolutional layers, pooling method, *step_size_real* and *num_images* values. We trained each model for 300 epochs or until convergence, whichever came later. For $MVMC_{real}$ (i.e. the MVMC architecture trained with synthetic and real data) we found a *step_size_real* value of 9 and a *num_syn_only* value of 20 to work best and all other major parameter choices matching that of $MVMC_{syn}$. For MV-LateFusion we found all major parameter choices to be the same as MVMC except for the weight_decay term which we set to 0.2. This high value was required due to the high-potential for overfitting by this model. As with ElevateNet, we multiplied the real loss by a constant to bring it to the same scale as the synthetic loss, in this case that value was 8.

4.3.3 Experiments and Results

We next validate our proposed multi-view approaches on real and synthetic data. We note that we use and model the same Seabotix vLBV300 system with a Tritech Gemini 720i that was described in the single-view ElevateNet section (Section 3.4.1).

4.3.3.1 Performance on Synthetic Data

We first discuss the generation of our synthetic data, which differs from our earlier single-view work. We then demonstrate that our multi-view approaches are capable of generating more accurate pixel-level elevation angle estimations and resulting 3D reconstructions in simulation.

Simulated dataset Different than our earlier single-view work, for this extension we use a new Gazebo-based simulator by Choi et al. [14][2] which is capable of modeling acoustic signal transmission, different object surface properties, and signal loss through reflections and multi-pathing. We chose this package because unlike the previous Rock-based simulator, the integration with Gazebo and ROS enabled us to more easily see the

[2]https://github.com/Field-Robotics-Lab/dave/wiki/Multibeam-Forward-Looking-Sonar

trajectories taken by the robot in simulation, thus allowing us to more easily develop a dataset of diverse viewpoints and trajectories. The use of such a ubiquitous framework will also allow others to more easily utilize or improve on our synthetic dataset in the future.

Other than the simulator choice, the high-level dataset details are similar to our earlier work. We model three shapes: a sphere of radius of 19 cm, a cylinder with a radius of 5 cm and height 10 cm, and a cube a side length of 30 cm. We generated data by taking diverse trajectories around the object and imaging at different poses at different ranges from 1-5 m. In total we generated 7284, 7636, and 6036 images for the sphere, cube, and cylinder respectively. These sets were split roughly 50/20/30 for the train/val/test sets. Examples of the simulated objects and their respective simulated sonar images can be found in Figs. 4.7a - 4.7f.

Pixelwise Elevation Angle Estimation We next evaluate the use of incorporating the information from multiple viewpoints in the elevation angle estimation process. We compare against three baselines: $Pred_{random}$ predicts a random elevation angle for every pixel which gives an overall idea of the ambiguity of the space, $Pred_{zero}$ predicts the mean (zero) elevation angle for every pixel, and our previous ElevateNet work (SV-ElevateNet) which only uses a single input image [18] .

We follow a similar evaluation process as in Section 4.2.2.1: differences between the predicted and ground truth elevation angle are converted to a height difference to give more context to the metric and each method is evaluated in terms of Mean Absolute Error (MAE) between the predicted and ground truth heights in a pixel-wise manner.

Table 4.5: Synthetic Pixelwise Elevation Angle MAE (cm)

	Sphere	Cube	Cylinder	All
$Pred_{random}$	15.5	15.8	14.7	15.6
$Pred_{zero}$	9.5	10.5	8.5	9.9
SV-ElevateNet [18]	9.13	**8.65**	8.50	8.81
MV-LateFusion (proposed)	**8.56**	8.92	6.85	8.59
MVMC (proposed)	**8.56**	8.79	**6.75**	**8.52**

The results are shown in Table 4.5 and overall are intuitive. $Pred_{random}$ and $Pred_{zero}$ are unable to achieve a low error due to their inability to leverage contextual information to make an informed estimate. SV-ElevateNet achieves a much lower error than the previous two approaches, but is not able to beat the multi-view methods in any category except for the cube, which has the simplest geometry. It is interesting to note that MVMC beats or is competitive with MV-LateFusion for all of the objects. This is possibly due to the increased complexity of MV-LateFusion leading to worse generalization ability on the test set. Overall, MVMC is the best performing method, validating that the additional information contained in multiple images is useful for the elevation angle prediction task.

Synthetic 3D Reconstruction Results Next, we evaluate our proposed approach on its ability to generate 3D reconstructions of the object being imaged. We compare with two baselines including SV-ElevateNet and a more recent work by Wang et al. called Acoustic to Front-View Network (A2FNet) [101]. A2FNet takes a different approach than estimating the missing elevation angle and instead attempts to train a CNN to transform the 2D sonar image into the front depth-view of the object being imaged. The benefit of this approach is that it is not affected by the non-bijective nature of the elevation angle estimation (i.e. multiple reflectors can reflect energy to the same pixel in sonar image space).

To compare against this approach we utilize the open-source version of this model [3] which we tuned to perform well on our dataset. One key difference in our tuning was to add a *weight_decay* term of 0.05 to help alleviate overfitting. We also found just using the L1 difference for a training loss did not work very well because much of the desired output image consistns of pixels with value 0 and therefore the network output did not contain any pixels besides those of depth 0. To avoid this we weighted pixels that should have a non-zero depth 100x higher than those that should have depth equal to 0.

To generate depth data for training this network in simulation, we used the depth cameras from the same simulator described previously. To generate more realistic data we used the CycleGAN network[4] used for style transfer in the original A2FNet paper. To verify that our version of A2FNet was well-tuned, we tested our model on their dataset and achieved a Mean Average Error of 2.7 cm which is even less than their reported 3.3 cm.

To compute quantitative metrics we generate a single 3D reconstruction for every simulated deployment. For each object we have 5 simulated deployments which involve trajectories around the object of interest at various ranges (between 1 and 5 m.) and poses. Similar to our ElevateNet work earlier, we compute the Root Mean Squared Error (RMSE) between the generated point cloud and a ground truth CAD model. Results for every shape are then averaged over the 5 simulated deployments.

[3]https://github.com/sollynoay/A2FNet
[4]https://github.com/junyanz/CycleGAN

Table 4.6: Synthetic Reconstruction RMSE (cm)

	Sphere	Cube	Cylinder
A2FNet [101]	11.88±3.63	12.11±3.27	16.49±3.82
SV-ElevateNet [18]	5.47±1.64	4.60±1.73	5.76±1.85
MV-LateFusion (proposed)	5.07±2.09	4.52±1.67	4.15±1.81
MVMC (proposed)	**5.05**±1.97	**4.45**±1.53	**3.69**±1.89

The results for our multi-view methods and the comparison methods are shown in Table 4.6. It is interesting to note that A2FNet achieves the highest error of all methods being evaluated. This could be due to the fact that it is completing a much more complex task (image prediction) as compared to our ElevateNet-based approaches (pixelwise classification) and thus is more susceptible to overfitting. Once again our MVMC is the most accurate, producing reconstructions with the lowest RMSE. This is intuitive given the extra information it can leverage over the single-view and the relative simplicity over the MV-LateFusion method, which may be more prone to overfitting.

Qualitative results showing 3D reconstructions from every method are shown in Fig. 4.7. The generated meshes are from a single "deployment" or traversal around the shape and thus we may not see every part of the object and are aiming for a partial reconstruction. Meshes are generated using the open3d library function $create\_from\_point\_cloud\_alpha\_shape$ with alpha parameter equal to 0.08.

In Figs. 4.7g - 4.7i we can see that A2FNet can reconstruct the general shape, but the depth prediction is off, resulting in reconstructions with a high error. Our previous method, SV-ElevateNet [18], is able to reduce the error greatly by directly estimating the elevation angle. Our proposed multi-view methods, MV-LateFusion and MVMC, leverage the information from multiple views and achieve the most accurate reconstructions.

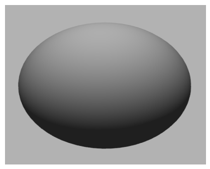
We note MVMC produces less outliers due to its simplified network structure which is less prone to overfitting.

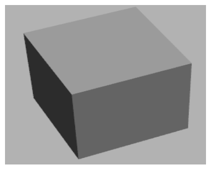
4.3.3.2 Performance on Real Data

We next discuss the generation our real-world dataset and the 3D reconstruction capability of our multi-view approaches on real data.

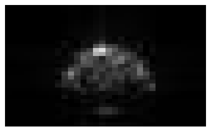
Real Dataset For this work we use data from the same Yaquina Bay, OR deployment as our earlier SV-ElevateNet work (described in Section 4.2.2.2). However, instead of having an expert select high-quality images for reconstruction we take a more programmatic approach to selecting images. From the set of images that contain an object, we first blur the sonar images using a (9, 9) Gaussian filter to reduce the effects of speckle noise and then select images that have at least 30 pixels of intensity greater than or equal to 50. These parameters were tuned on validation data to select images where the object of interest was prominently displayed.

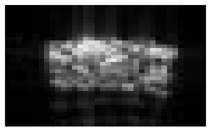
To evaluate the effectiveness of our approach we use the same objects as before: a sphere of radius of 19 cm, a cylinder with a radius of 5 cm and height 10 cm, and a cube a side length of 29 cm. In total we generated 7679, 10586, and 7575 images for the sphere, cube, and cylinder respectively. While it may seem counter-intuitive that we actually have more real data than synthetic data, we note that due to the lack of ground truth information, the real data does not provide as strong a training signal as the synthetic data. These sets were split roughly 55/25/20 for the train/val/test sets.

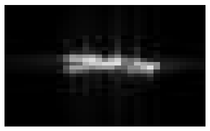

(a) Sphere target

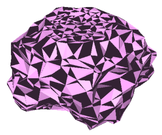

(b) Cube target

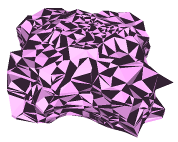
(c) Cyl. target

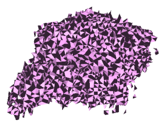

(d) Sphere image

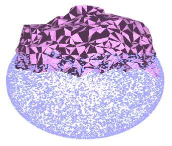

(e) Cube image

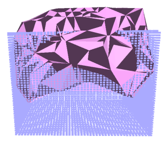

(f) Cyl. image

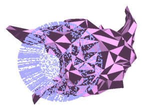

(g) A2FNet sphere [101]

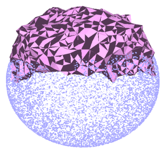

(h) A2FNet cube [101]

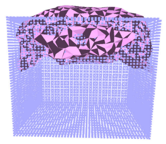

(i) A2FNet cylinder [101]

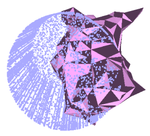

(j) SV-ElevateNet sphere [18]

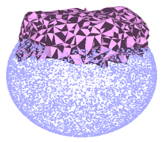

(k) SV-ElevateNet cube [18]

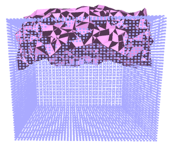

(l) SV-ElevateNet cylinder [18]


(m) MV-LateFusion sphere (ours)


(n) MV-LateFusion cube (ours)


(o) MV-LateFusion cylinder (ours)


(p) MVMC sphere (ours)


(q) MVMC cube (ours)

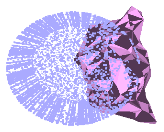

(r) MVMC cylinder (ours)

Figure 4.7: Synthetic sonar targets (a)-(c), respective synthetic sonar images (d)-(f), and reconstructed meshes (g)-(r). Ground truth point clouds are blue and predicted meshes are pink. Incorporating information from multiple viewpoints (MVMC and MV-LateFusion) enables predictions with less outliers that fit the ground truth models (blue) better. Additional results can be found at: https://youtu.be/QhcHExXDQtY

Real 3D Reconstruction Results To evaluate the performance of our multi-view approach on real data, we take a similar approach to our earlier work (Section 4.2.2.2) and generate 10 reconstructions from sets of 3 images taken from various viewpoints. This approach demonstrates the ability of our method to generate reconstructions in "fly-through" scenarios or in cluttered environments where obtaining diverse viewpoints is challenging.

Due to the more diverse and potentially lower image quality of the real dataset, for this extension we filter the outputs of our multi-view CNNs. Specifically, we use the open3d library's $remove\_statistical\_outlier$ function with number of neighbors equal to 150 and standard deviation ratio of 2.0 to remove 3D outliers most often generated by bright pixels in the image that were not the object.

Due to the lack of ground truth elevation angle information for the real data, we once again hand-align the predicted point clouds to their respective ground truth model and complete ICP to rigidly align the point clouds. To complete ICP we use open3d's $registration\_icp$ function with a maximum points-pair distance of 5 cm.

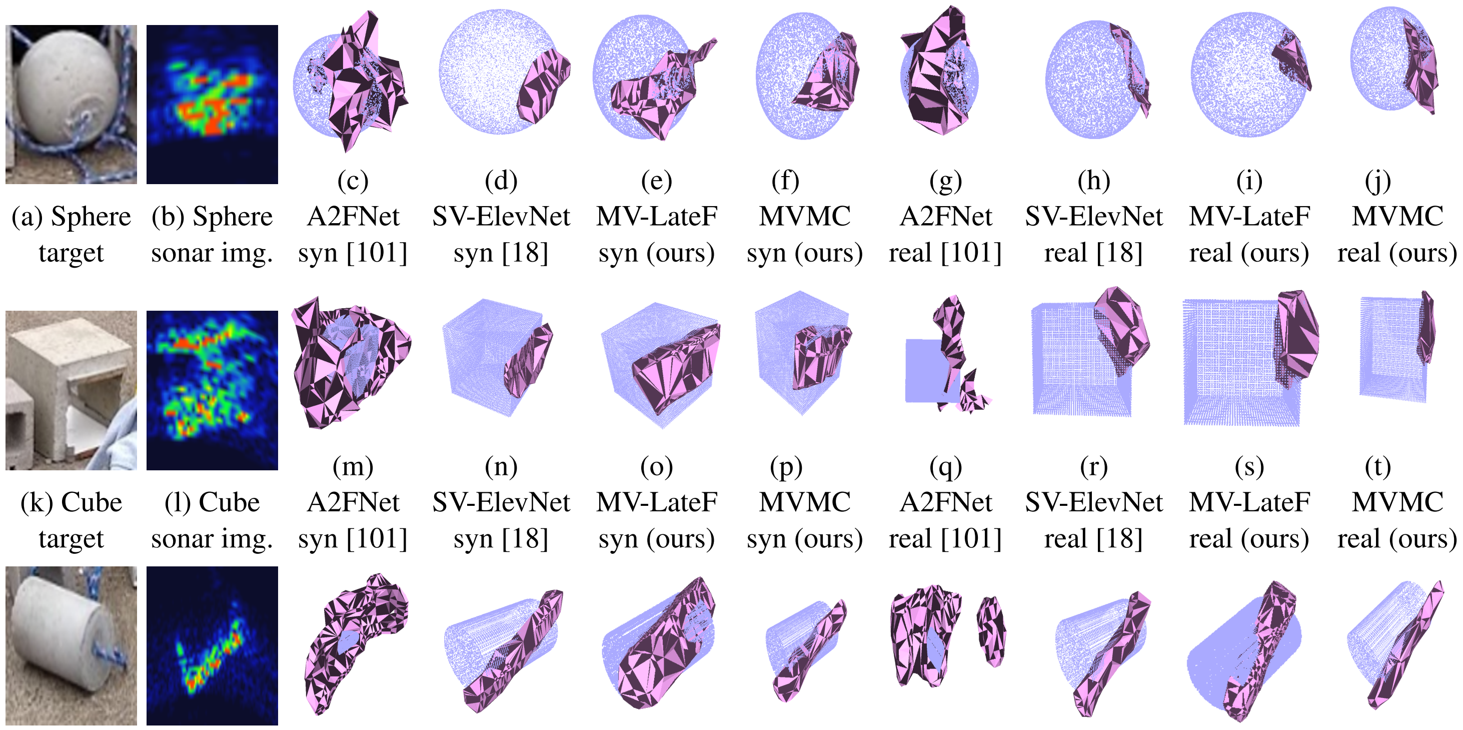
The results averaged over the 10 reconstructions are shown in Table 4.7. The column contains an average over the averages for each shape, representing one number to compare each method. As one can see, leveraging multiple frames (e.g. MVMC vs. SV-ElevateNet) achieves better performance than architectures trained only with single-image inputs. As expected, training with real data also enables more accurate reconstructions for all of the architectures. A2FNet also struggles on the real data, probably due to the fact that the network is required to do much more than ElevateNet and thus cannot properly converge during training. It is interesting to note that the SV-ElevateNet

Table 4.7: Multi-view Real Reconstruction RMSE (cm)

	Train Data	Sphere	Cube	Cylinder	Avg.
A2FNet [101]	Syn	12.96±8.01	18.0±9.57	29.06±4.29	20.0
SV-ElevateNet [18]	Syn	3.21±1.06	1.90±0.63	2.39±0.50	2.50
MV-LateFusion (ours)	Syn	3.31±0.93	1.94±0.71	2.69±0.45	2.65
MVMC (ours)	Syn	3.38±0.83	1.53±0.59	2.31±0.40	2.41
A2FNet [101]	Syn + Real	12.17±9.2	16.35±9.5	29.36±9.7	19.3
SV-ElevateNet [18]	Syn + Real	**2.96**±1.11	1.84±0.55	2.48±0.46	2.43
MV-LateFusion (ours)	Syn + Real	3.32±1.12	1.73±0.63	2.49±0.48	2.51
MVMC- (ours)	Syn + Real	3.26±0.83	**1.40**±0.50	**2.28**±0.41	**2.31**

method achieves the best performance on the sphere object, which could indicate that the multi-view methods overfit slightly on the more complex object. In any case, our MVMC method trained with synthetic and real data achieves the lowest error overall and on the cube and cylinder shapes specifically.

We present qualitative results in Fig 4.8. Once again, we see more accurate reconstructions generated when architectures train with real data and are able to adapt to the noise present in the real imagery. We also note that MVMC and MV-LateFusion are able to leverage the additional information present in their multi-image inputs to achieve reconstruction that better match the shape of the ground truth models. One particular example of this is comparing the reconstructions SV-ElevateNet (Figs. 4.8j, 4.8r, 4.8ab) with the reconstructions of MVMC (Figs. 4.8h, 4.8t, 4.8ad) with the latter reconstructions more closely aligning to the surfaces being imaged.


(a) Sphere target (b) Sphere sonar img. (c) A2FNet syn [101] (d) SV-ElevNet syn [18] (e) MV-LateF syn (ours) (f) MVMC syn (ours) (g) A2FNet real [101] (h) SV-ElevNet real [18] (i) MV-LateF real (ours) (j) MVMC real (ours)

(k) Cube target (l) Cube sonar img. (m) A2FNet syn [101] (n) SV-ElevNet syn [18] (o) MV-LateF syn (ours) (p) MVMC syn (ours) (q) A2FNet real [101] (r) SV-ElevNet real [18] (s) MV-LateF real (ours) (t) MVMC real (ours)

(u) Cylinder target (v) Cylinder sonar img. (w) A2FNet syn [101] (x) SV-ElevNet syn [18] (y) MV-LateF syn (ours) (z) MVMC syn (ours) (aa) A2FNet real [101] (ab) SV-ElevNet real [18] (ac) MV-LateF real (ours) (ad) MVMC real (ours)

Figure 4.8: Real 3D reconstruction results. The first column contains the sonar targets, the second column contains example sonar images and in remaining columns the ground truth models are blue while the estimated reconstructions are pink. Once again not all parts of the object are observed by the robot so a partial reconstruction is expected. Meshes are generated using the open3d library function *create_from_point_cloud_alpha_shape* with alpha parameter equal to 0.08. Additional results can be found at: https://youtu.be/TnxEsXP-p14. Best viewed in color.

4.4 Conclusion

In this chapter we proposed 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.

Additionally, we proposed an extension of this single-view architecture which can accommodate multiple input images, thus reducing the effects of noise and allowing the architecture to leverage more information for its estimation. We demonstrated that utilizing multi-view inputs, MVMC and MV-LateFusion can generate more accurate 3D reconstructions that their single-view counterpart.

In conclusion, in this chapter ElevateNet removed the need for a human to be involved a low-level with the underwater 3D reconstruction loop. This could enable them to focus on higher-level tasks, such as mission planning or vehicle monitoring. While we demonstrated that mixing the synthetic and real data produced more accurate reconstructions on real-world data, in this next chapter we'll take the sim-to-real process a step further by explicitly forcing the feature extractor to focus on features relevant to both synthetic and real data.

Chapter 5: Bridging the Sim-to-Real Gap for 3D Object Detection from Point Clouds

5.1 Background

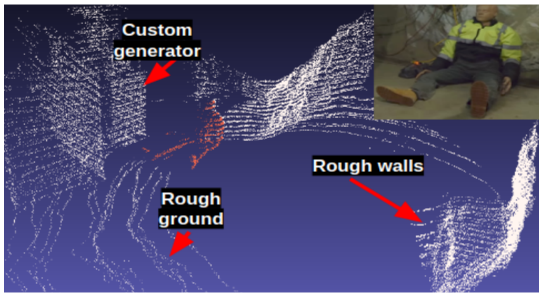
In our second contribution we showed that intelligently mixing synthetic data into the training process helped to train deep networks when there was a lack of available real training data. However, an improved scenario is one in which the learned features in synthetic data would be directly applicable on real data. Therefore, in this contribution we take the incorporation of synthetic data a step further by explicitly tying the training of the network to extract similar features from synthetic and real data. In this contribution we use the exploration of subterranean environments as a motivating application where the lack of real labeled datasets prompts the use of synthetic data when training deep networks.

Scenarios such as search and rescue in poorly-lit underground environments [99] and autonomous driving applications in poor weather or at night [72] can cause the same object or environment to appear differently in camera imagery depending on the conditions. This adversely affects the performance of algorithms extracting information, such as object location, from these images [72]. Light detection and ranging (lidars) are increasingly being used as a complementary sensor to cameras because of their ability to better capture clear sensor data in such low-light or poor weather conditions. In this chapter we address the problem of object detection in point clouds from a lidar. Specifically, given a set of 3D points we develop an approach which outputs a set of bounding boxes with center (x, y, z), dimensions ($length$, $width$, $height$), and orientation (yaw).

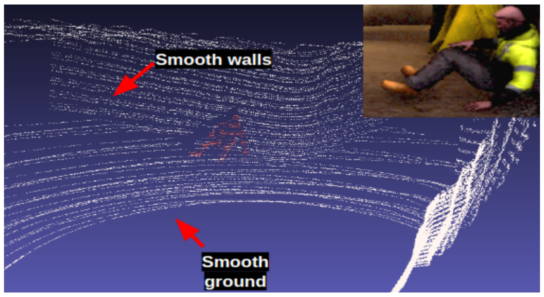
Recently, deep learning has achieved state-of-the-art performance for object detection from point clouds [73]. Oftentimes these results are demonstrated in scenarios where a large corpus of training data in the same domain as the test data can be leveraged. In this contribution we target applications where such a large amount of training data does not exist; such as those motivating the DARPA Subterranean (SubT) Challenge. Even after expensive robot deployments in representative environments, annotating objects in point clouds is difficult due to the relatively small number of humans who are familiar with point cloud data. In addition to all of the challenges of labeling 2D images (time, expertise, and possibly poor annotation quality), 3D point clouds have another dimension thus making object annotation even more difficult than labeling 2D camera images.

Simulators can be used to generate large amounts of training data, however, as shown in Fig. 5.1, a *sim-to-real gap* often exists. This gap limits the performance of networks that naively incorporate simulated data for training [24, 94].

Addressing the sim-to-real gap for point clouds has received limited research attention, often focusing on enforcing similarity constraints between real and synthetic feature extraction. For example, SqueezeSegV2 is a feature comparison approach which uses a geodesic distance function between real and synthetic feature vectors [108]. However, in [108] point clouds were compressed into 2D first which can cause the loss of detailed 3D information [76]. While fully 3D methods exist, these approaches often fo-



(a) Real survivor in point cloud (pink points) and inset image.



(b) Synthetic survivor in point cloud (pink points) and inset image.

Figure 5.1: Example of the sim-to-real gap for point cloud object detection. The real point cloud (top) has rough walls and features as well as objects such as the generator that are not found in simulation. The synthetic point cloud (bottom) has relatively smooth surfaces. Points are colored for visualization purposes only and are not used for object detection.

cus on small-scale applications such as single-object classification, which we will show is not always scalable to larger robotic scenes [76].

Adversarial training is an alternative method for feature comparison which has seen success for comparing features from 2D camera images. For example, Ganin et al. [24] used adversarial training with synthetic road signs to assist with training a network that classifies real road signs. This is accomplished by using an adversarial discriminator to promote extraction of similar features from real and synthetic data.

While such approaches are effective for 2D images, we note that 3D point cloud data is quite different than images particularly with respect to the highly-unstructured nature of point cloud data. Towards developing effective solutions for real-world point cloud object detection, we develop a fully 3D adversarial discriminator which does not compress information into 2D, and is lightweight enough to scale to large robotics scenes. Our adversarial discriminator is not specific to a fixed 3D representation and we analyze its performance with both point and voxel-based detectors [73, 85]. Additionally, for point-based backbones we develop an adaptive sampling module which maintains important data throughout the discriminator.

In summary, in this contribution we:

- Develop a flexible and lightweight 3D adversarial training scheme that leverages an adaptive sampling module to reason over large robotic scenes.
- Demonstrate the utility of this scheme across voxelized and point-based 3D object detection architectures.
- Release our 3D object detection dataset and code as a training resource and benchmark for future work.[1]

We validate our approach on real data collected in a variety of environments as part of the DARPA Subterranean Challenge and demonstrate that our approach increases performance by up to 15%. We note this work also appeared in a journal paper [17].

5.1.1 3D Object Detection in Point Clouds

3D object detection from point clouds has received increasing attention from the research community [73, 87]. For example, Qi et al. [73] demonstrated accurate 3D object detection in indoor environments from only point clouds. Methods such as ComplexYOLO have also been proposed which project the point cloud into a 2D view and

[1]https://github.com/debortoli/AdvNet3D

complete object detection from that view [87]. Such techniques work fairly well in autonomous driving scenarios, where large objects such as cars can be seen easily from top-down views. However, they are not appropriate for applications such as underground exploration, where clutter and large overhangs preclude the use of a top-down projection. Additionally, a 2D projection inherently loses valuable 3D information that could be used in downstream tasks and thus many state-of-the-art detectors do not compress clouds [76, 85].

5.1.2 Closing the Sim-to-Real Gap With Point Clouds

Reducing the domain shift between simulated and real point clouds has received increasing research attention. Wu et al. proposed SqueezeSegV2 which projects point clouds into 2D and uses a geodesic distance function to compare synthetic and real features [108]. This geodesic distance function works well in 2D, however we show it does not scale as well as our adversarial approach to fully 3D processing. Alternatively, PointDAN is a method that operates in 3D: by using attention networks on low-level features it improves 3D object classification performance between different domains [76]. We will show that focusing on local features cannot reason over large 3D scenes as well as our adversarial discriminator.

Recently, much work on reducing the domain shift for point cloud processing has focused on autonomous driving. For example, Wang et al. demonstrated that by leveraging knowledge of test dataset car sizes, great improvements could be made in detection performance when testing on a dataset different than training dataset [100]. This method works well for autonomous driving scenarios, however the size of our objects of interest varied considerably less than cars across different continents. Targeting applications without target domain labels, Yang et al. developed a pseudo-labeling scheme for unsupervised domain adaptation and while they achieved great results, we are targeting applications where some target domain labels exist and can be directly leveraged for training [112]. This is especially important in the SubT application, where the domain gap is very large, leading to difficulties for detectors trained on just source data to produce accurate detections or pseudo-labels.

5.1.3 Adaptive Sampling for 3D Point Clouds

Due to the increased dimensionality of 3D point clouds, uniform sampling techniques can remove useful information too early in the processing pipeline [65]. For the point clouds found in SubT a very small number of points represent the object of interest and a large amount of the points represent the walls and floors and therefore a uniform sampling method may not preserve useful object information.

One example of adaptive sampling for point clouds is PointASNL which uses a learned adaptive sampling module to improve sampling of local neighborhoods [111]. However, because this module is learned, it potentially will require a lot of data to train. Alternatively, Nezhadarya et al. propose an adaptive global sampling technique, similar to global max pooling, that does not require any additional training [65]. However, standard architectures such as VoteNet require a fixed number of points after each layer.

To address this [65] copies the downsampled set, which works well for their object classification application, but may destroy too much information in large robotic scenes.

Our method starts similar to [65] and generates a downsampled set based on the maximally activated features from the previous layer. However, this downsampled set may be too small and sparse for entry into the next layer, and therefore we next upsample using nearest neighbor upsampling around the downsampled set. This allows for structural information to be kept and does not require any additional training, an attractive feature in our data-limited application.

5.2 Methods

In this section we first describe the 3D object detection problem and our overall adversarial training architecture. Next, we describe our point-based adversarial training architecture and our adaptive sampling module. Finally, we discuss our voxel-based adversarial training architecture.

5.2.1 Problem Description

Given a raw 3D point cloud, in this work we seek to train a 3D object detector that outputs 3D bounding boxes with center *(x, y, z)*, size *(length, width, height)*, and yaw (θ).

We validate our approach on an application with a particular lack of data: the subterranean environments which motivate the DARPA Subterranean Challenge [1]. Competi-

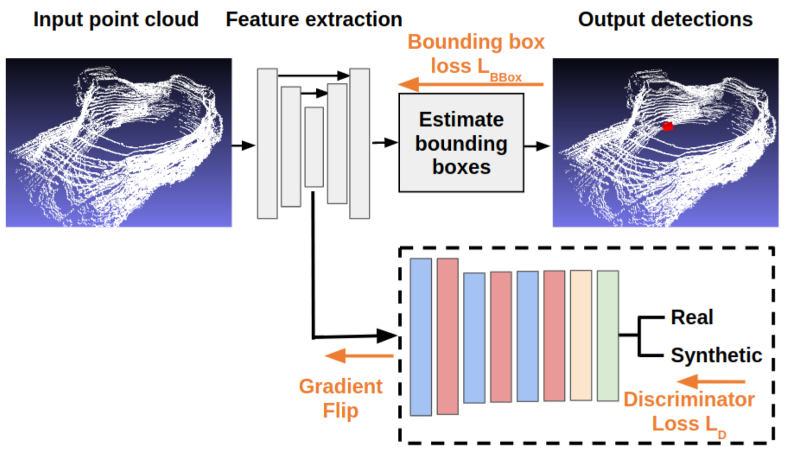


Figure 5.2: Overview of our 3D adversarial training architecture. The top gray pipeline is the backbone network, which can be any standard model for point-based or voxel-based representations. Inside of the dashed box is our adversarial discriminator which leverages intelligent adaptive sampling layers (red) to maintain detailed information throughout the pipeline. By flipping the gradient ("Gradient Flip" in orange above) the feature extractor is encouraged to fool the discriminator by extracting similar features shared across synthetic and real data.

tors in this challenge must develop a robotic system to venture into unexplored underground environments and detect certain objects of interest. In this work we target three object classes that are large enough to appear in point cloud data: survivor, backpack, and fire extinguisher. Examples of these objects and their point cloud representations are shown in Fig. 5.5.

Due to the limited availability of real data, Microsoft's AirSim robot simulator is used to generate large amounts of labeled real data [83]. While the simulator provides a myriad of data, there still exists a sim-to-real gap. Our adversarial training approach reduces the effect of this gap by better leveraging the synthetic data during training.

5.2.2 Training Overview

Algorithm 1 Training algorithm for point-based adversarial discriminator architecture

```
    Input: Gradient penalty coefficient $\lambda$, ratio of discriminator to backbone updates $n_{critic}$, dataset $S$ of real and synthetic data, learning rate $l$, pretrained backbone parameters $\theta_0$, and initial discriminator parameters $w_0$.
 1: while $\theta$ not converged do
 2:     for $j$ = 0,1, .... $n_{critic}$ do
 3:         $f_{real}, f_{syn}, f_{mixed} \leftarrow GetFeatures()$
 4:         ▷ $D$ is discriminator
 5:         $L_D \leftarrow D_w(f_{real}) - D_w(f_{syn}) +$
                  $\lambda(||\nabla_{f_{mixed}} D_w(f_{mixed})||_2 - 1)^2$
 6:         ▷ Update only discriminator
 7:         $w \leftarrow Adam(\nabla_w L_D, w, l)$
 8:     ▷ Now focus on backbone
 9:     $f_{real}, f_{syn}, f_{mixed} \leftarrow GetFeatures()$
10:     $L_{bboxreal} \leftarrow L_{bbox}(bboxes_{real}, target_{real})$
11:     $L_{bboxsyn} \leftarrow L_{bbox}(bboxes_{syn}, target_{syn})$
12:     $L_{bboxmixed} \leftarrow L_{bboxreal} + L_{bboxsyn}$
13:     $L_D \leftarrow D_w(f_{real}) - D_w(f_{syn}) +$
                  $\lambda(||\nabla_{f_{mixed}} D_w(f_{mixed})||_2 - 1)^2$
14:     $\theta \leftarrow Adam(\nabla_\theta L_{bboxmixed} - L_D, w, l)$
15: function GetFeatures
16:     $inp_{real} \leftarrow SampleRealBatch(S)$
17:     ▷ $f_{real}$ is output of FE4 in Fig. 5.2
18:     $bboxes_{real}, f_{real} \leftarrow Backbone(inp_{real})$
19:     $inp_{syn} \leftarrow SampleSynBatch(S)$
20:     $bboxes_{syn}, f_{syn} \leftarrow Backbone(inp_{syn})$
21:     $f_{mixed} \leftarrow \epsilon f_{real} + (1 - \epsilon) f_{syn}$
22:     return $f_{real}, f_{syn}, f_{mixed}$
```

An overview of our training approach can be found in Fig. 5.2 and Algorithm 1; we first start by training the discriminator. We feed input data into the backbone network, take the feature encoding produced and use it as input into our adversarial discriminator which completes a binary classification, either synthetic or real. The Wasserstein-GAN Gradient Penalty Loss (WGAN-GP)[2] is then computed on this output:

$$L_D = D_w(f_{real}) - D_w(f_{syn}) + \lambda(||\nabla_{f_{mixed}} D_w(f_{mixed})||_2 - 1)^2, \quad (5.1)$$

where D and w are the discriminator and its weights, f_{real} and f_{syn} are feature encodings from real and synthetic data respectively, and f_{mixed} is defined as:

$$f_{mixed} = \epsilon f_{real} + (1 - \epsilon) f_{syn}, \quad (5.2)$$

where ϵ is a random number between 0 and 1 [31].

We note that using the WGAN-GP loss function is a departure from the standard adversarial training paradigm [24]. WGAN-GP leverages a gradient penalty term to provide a more stable loss signal than standard distance measures and has typically been used for applications that *generate* data such as 2D images or small single-object 3D point clouds [62]. Towards deployment on large 3D robotic scenes, in this work, we adapt this loss function to the application of adversarial training on point clouds, thereby enabling stable training on high-dimensional and unstructured point clouds.

Next, the weights of the discriminator are updated according to L_D. This loop is run n_{critic} times to ensure the discriminator is trained enough with respect to the backbone. Next, we update the weights of the backbone network. Once again, data is input and now training utilizes both the output bounding boxes as well as the output of the dis-

[2]https://github.com/caogang/wgan-gp

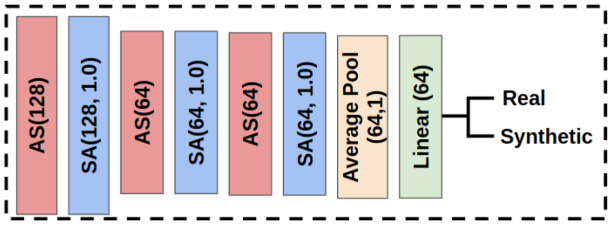


Figure 5.3: Our novel point-based adversarial discriminator which leverages an adaptive sampling layer to maintain object information throughout the processing pipeline. SA(x,y) = VoteNet Set Abstraction layer with x points and y radius. AS(x) = adaptive sampling to x number of points. Average pooling is used to avoid destroying detailed information.

criminator. To compute the bounding box loss, L_{BBox}, we use the standard loss function found in [73] which balances objectness, bounding box location, and bounding box size compared to the ground truth bounding box parameters. During this phase we again compute the discriminator loss L_D, however we now pass the gradients through the discriminator. After the gradients get through the discriminator, they are flipped, which trains the feature extractor layers in the backbone to maximize L_D. The final loss for the backbone is thus:

$$L_{Backbone} = L_{BBox} - \alpha L_D, \tag{5.3}$$

where α is a balancing parameter between the two losses.

By training the backbone network to predict accurate bounding boxes *and* maximize the loss of the discriminator, we enforce similar feature extraction from synthetic and real data, thereby enabling better usage of synthetic data during training. At inference time, the adversarial discriminator is removed and the backbone architecture produces 3D bounding boxes.

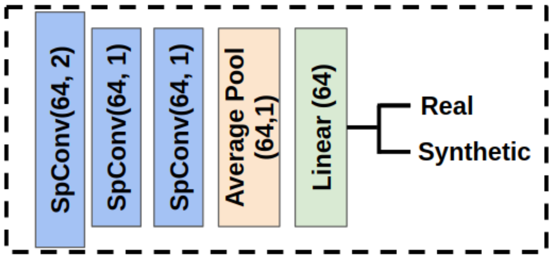


Figure 5.4: Our novel adversarial discriminator used for voxel-based experiments. This discriminator works with the PartA2 backbone which leverages sparse convolutions to operate over voxelized inputs. SpConv(x,y) is a sparse convolution layer with x number of filters and y stride.

5.2.3 Point-Based Architecture

We next discuss our point-based adversarial architecture and our adaptive sampling module. The VoteNet model is used as a backbone for feature extraction and bounding-box generation [73]. VoteNet operates on the raw point clouds without compression into 2D or discretization by voxelization and is a top approach for 3D object detection on the ScanNetV2 dataset [16]. It is thus well-suited for dense environments such as tight underground tunnels.

5.2.4 Dataset Overview

An overview of our point-based discriminator can be seen in Fig. 5.3. For feature extraction, we leverage 3D Set Abstraction layers for seamless integration with VoteNet as well as to support a powerful 3D-aware discriminator. Additionally, we develop an adaptive sampling module that maintains important information throughout the discriminator. This module is a drop-in replacement for the Farthest Point Sampling used in vanilla Set Abstraction layers. To start, similar to [65], we downsample to the set of features that are maximally activated. This downsampled set may be too small for input into the next layer, so to bring it up to the fixed size required by VoteNet, we sample the nearest neighbors of the points in the downsampled set. This is distinctly different than [65] which copies points in the downsampled set to upsample, potentially losing out on valuable neighborhood information which could be leveraged by the discriminator later on for classification.

5.2.5 Voxel-Based Architecture

To demonstrate the flexibility of our 3D adversarial training approach, we also develop a voxel-based adversarial training architecture. For our backbone we leverage the PartA2 architecture; a 3D voxelized model which uses sparse convolution to efficiently extract features [85]. PartA2 has shown high-quality 3D object detection capabilities in outdoor scenes and is one of the best publicly available methods for pedestrian and cyclist detection on the KITTI dataset [26]. PartA2 is thus appropriate for cavernous open-spaces underground such as natural caves.

To integrate effectively with the PartA2 backbone, we use the same type of sparse convolutional feature extractors in our discriminator as in the backbone. Once again, we leverage 3D-aware layers to enable a powerful discriminator which can reason about the incoming features fully in 3D. The discriminator is shown in Fig. 5.4.

For training the voxel-based architecture we use the same overall training structure as the point-based architecture in Algorithm 1. Due to the voxelization process, different batches had different amounts of voxels which made the computation of a gradient penalty term between synthetic and real feature vectors difficult in PyTorch. Therefore, we used the WGAN optimization technique which is similar to WGAN-GP, but does not include the gradient penalty term (i.e. $(||\nabla_{f_{mixed}} D_w(f_{mixed})||_2 - 1)^2$ in Equation 5.1) [2].

5.3 Experiments and Results

In this section, we start with a description of our robot system and point cloud dataset. Next, we demonstrate the benefits of our adversarial training approach on real-world detection datasets. Finally, we explore the effects of feature encoding levels as input to our adversarial discriminator.

5.3.1 System Overview

Real data used for training and testing was captured by Team Explorer's custom-built ground vehicles designed for the DARPA SubT Competition. These vehicles typically travel with speed up to 2 m/s and have a variety of onboard sensors including RGB and thermal cameras as well as a Velodyne VLP-16 used to generate the point clouds for this work. We spin our Velodyne at 10 Hz and use the LOAM SLAM system to concatenate sets of about 5 individual scans for generating high-resolution inputs to our model [115].

To generate synthetic data, we use Microsoft AirSim with a synthetic underground tunnel environment rendered in the Unreal engine [83]. High-resolution surface meshes

Table 5.1: Real dataset composition (# clouds)

	Backpack	Fire Ext.	Survivor	No object
Train	49	88	67	1522
Validation	78	29	26	125
Test	54	53	16	568

Table 5.2: Synthetic dataset composition (# clouds)

	Backpack	Fire Ext.	Survivor	No object
Train	185	225	383	5434
Validation	354	454	180	5229
Test	118	30	163	7069

enable accurate lidar simulation. The simulated vehicle has realistic size, physical capabilities, and lidar parameters.

Fig. 5.5a - 5.5f and Table 5.1 contain an overview of our real dataset. Training data was captured during the DARPA SubT STIX event at the Edgar Experimental Mine, CO, the SubT Tunnel Circuit at the NIOSH Mine, PA, and on the Carnegie Mellon University (CMU) Campus, PA in representative urban environments such as a parking garage. Validation data was captured at Tour-Ed mine, PA. Test data is from the Edgar Mine, the Tour-Ed mine, and the CMU campus in parts of the environments different than those used for capturing training/validation data. After collection, the data was hand-labeled by a human expert.

Fig. 5.5g - 5.5l and Table 5.2 contain an overview of our synthetic dataset. Data was gathered by randomly moving the robot around, collecting scans, and using AirSim's automatic label generation which enabled a large amount of data to be generated cheaply. Training, validation, and test sets were all gathered from different locations in

the simulation. Additionally, due to the realistic nature of the SubT Challenge, often only 10-15% of the clouds in the dataset contain objects.

5.3.2 3D Object Detection Performance

We next discuss pre-processing and comparison methods for both the point and voxel-based experiments. We then present point and voxel-based object detection results.

5.3.2.1 Point Cloud Preprocessing

Before point clouds are input to the model, there are a number of preprocessing steps. Clouds are first downsampled about 20% to 80,000 points to make training and inference faster. Keeping the original number of points did not significantly improve performance. Because of the sparsity of objects to be detected in the environment, clouds are first cropped to 5 meter cubes to make learning easier. We note that this brings our clouds to about the same physical size as other point cloud datasets like ScanNet [16] and that cropping has been used in previous works [50]. To ensure objects were actually observable in the clouds, we did not label objects that did not meet certain size and number of point constraints: The backpack had a minimum (length, height, width) of (10, 10, 10) cm and a minimum of 40 points, fire extinguishers had a minimum size of (5, 20, 5) cm and 60 points, and the survivor had a minimum size of (25, 25, 25) cm and minimum of 80 points.

5.3.2.2 Comparison Methods

We compare our adversarial approach against a number of baselines. The first three methods only use the VoteNet architecture (for point-based experiments) or the PartA2 architecture (for voxel-based experiments) and are dataset composition approaches:

- *OnlySyn* uses only synthetic data for training/validation.
- *OnlyReal* uses only the small amount of real data.
- *SynAndReal* mixes the synthetic and real data for training/validation. To ensure real data is emphasized during learning, it is sampled at a 2:1 rate with synthetic data.

The remaining methods, including our adversarial training approach, all have access to synthetic and real data.

- *GeodesicLoss* (GeoLoss) is based on the work by [108] which utilizes the geodesic distance between the covariance matrices of the real and synthetic feature vectors from 2D point cloud images[3]. While this method is similar to ours in that it encourages synthetic and real feature vectors to be similar, our learned adversarial discriminator is able to scale better to 3D environments.
- *PointDAN* uses Maximum Mean Discrepancy to align local features for point cloud object classification[4] [76]. Focusing on fine-grained features works well

[3]https://github.com/pmorerio/minimal-entropy-correlation-alignment
[4]https://github.com/canqin001/PointDAN

for object classification, however we show it does not scale as well as adversarial training to large scenes.

- *Adversarial architecture* (AT) is our proposed approach which uses a lightweight and powerful 3D adversarial discriminator to encourage similar feature extraction from real and synthetic data in a fully 3D manner.

- *Adversarial architecture with adaptive sampling* (AT-AS) uses the same architecture as AT but leverages our adaptive sampling module in the discriminator to avoid removing important object information before completing classification.

Given the small amount of real training data available, it is relevant to explore if non-deep learning methods could outperform deep learned methods. For example, Wang et al. propose a method which extracts shape-factor features from point clouds and uses an SVM to classify these features as an object of interest or not [96]. While their code is proprietary, we implemented this method, ran it on our SubT data, and found it was unable to achieve higher than 0.01 average precision on the test set survivor. We hypothesize this is due to the shape-factor features not being powerful enough to detect objects with high appearance change, as is the case in our application. Specifically, the objects for SubT are significantly smaller than other applications such as the KITTI dataset and therefore common challenges with point clouds such as occlusions and varying point densities can be much more degrading to our objects of interest.

Table 5.3: Point-based 3D object detection results

	Average precision on real test data			
Train method	Backpack	Survivor	Fire Ext.	mAP
OnlySyn	0.29±0.14	0.04±0.07	0.00±0.01	0.11±0.03
OnlyReal	0.56±0.02	0.49±0.09	0.05±0.01	0.37±0.04
SynAndReal	0.56±0.03	0.61±0.13	0.03±0.03	0.40±0.02
GeoLoss [108]	0.61±0.02	0.54±0.05	**0.07**±0.02	0.41±0.03
PointDAN [76]	0.57±0.03	0.45±0.10	0.04±0.03	0.35±0.03
AT (Ours)	0.62±0.04*	0.64±0.06*	0.04±0.01	0.43±0.04
AT-AS (Ours)	**0.63**±0.02*	**0.71**±0.04*	**0.07**±0.04	**0.47**±0.01

* = Statistically significant with p-value <0.05
when compared with next-best approach

5.3.2.3 Point-Based Object Detection

To train our adversarial discriminator for point-based object detection, we used a learning rate of 0.0005, weight decay of 0.1, batch size of 8, and class sampling weightings of 1, 2, and 2 for the backpack, fire extinguisher and survivor respectively. We used an n_{critic} of 2 because the discriminator trained quickly and a λ value of 10. To emphasize useful feature extraction early on, we varied the α balancing term in a linear manner from 0 to 1 starting at epoch 0 until epoch 300.

All methods were trained to convergence based on validation mAP (defined as the average across object classes). Therefore training usually lasted up to 200 epochs and takes 13 hours on a single GTX 1080 GPU. For the point-based object detection experiment, all methods used a VoteNet backbone pretrained on the ScanNet dataset[5]. For data augmentation we rotated/translated the clouds, scaled the clouds by $\pm$ 10%, applied $\pm$ 2 cm of jitter, and moved objects between clouds.

[5]https://github.com/facebookresearch/votenet

Methods are evaluated using the VoteNet evaluation pipeline and average precision metrics are computed at 0.25 3D IOU. We note that the average precision is a single metric that captures both the precision and recall of the detector across all detection thresholds [70]. The mean and standard deviation results averaged over three training runs are shown in Table 5.3. To compute statistical significance, for each object we run a t-test with N = 691 and compare both the AT-AS and AT methods with the next-highest-performing method. At a high-level, the results are intuitive: many methods do the best on the large survivor and perform relatively poorly on the fire extinguisher. Given the rough walls and overhangs that often accompanied the fire extinguisher, this was a particularly challenging environment for such detections.

Investigating the results in detail, the performance difference between OnlySyn and OnlyReal demonstrates the sim-to-real-gap. The difference between OnlyReal and SynAndReal demonstrates the utility of using the synthetic data for increasing the amount of data we have. Due to the use of attention on low-level features, PointDAN struggles to scale to larger 3D scenes. The GeodesicLoss provides improvement over SynAndReal, but due to the inflexible nature of its loss function, it is not able to beat our adversarial approach in many cases. Our adversarial architecture (AT) improves performance on the backpack, survivor, and on the Mean Average Precision (mAP) metric by not compressing information and learning the discriminator while training. Finally, our adversarial architecture with adaptive sampling (AT-AS) achieves the best performance on all three objects by maintaining important information throughout the discriminator.

Qualitative results for our point-based adversarial architecture on real test data from Edgar Mine, CO are shown in Fig. 5.6. Overall, we are able to detect objects well in this

Table 5.4: Voxel-based 3D object detection results

	Average precision on real test data			
Train method	Backpack	Survivor	Fire Ext.	mAP
OnlySyn	0.31±0.06	0.0±0.01	0.07±0.05	0.13±0.01
OnlyReal	0.39±0.06	0.13±0.06	0.18±0.08	0.23±0.01
SynAndReal	**0.57**±0.05*	0.09±0.03	0.13±0.08	0.26±0.02
GeoLoss [108]	0.52±0.08	**0.16**±0.07*	0.12±0.01	0.27±0.04
PointDAN [76]	0.44±0.07	0.15±0.02	0.15±0.08	0.25±0.01
AT (Ours)	0.46±0.12	0.15±0.03	**0.24**±0.03	**0.28**±0.05*

* = Statistically significant with p-value <0.05
when compared with next-best approach

challenging real environment. However, one area for future improvement is the angular estimation of the bounding boxes. In Fig. 5.6c it is interesting to note that on the right side of the scene we have a number of false positives which were caused by the DARPA organizers covering up another fire extinguisher with a large form-fitting covering. This was not labeled as a ground truth object location in our dataset, and is demonstrative of the difficulty of our task. We also found thin objects such as support poles to be a source of false positives from our detector.

5.3.2.4 Voxel-Based Object Detection

For the voxel-based architecture experiments, data pre-processing and data augmentation techniques were all similar to the point-based architecture. All methods use a PartA2 anchor-free backbone[6] pretrained on the KITTI dataset and are trained to convergence based on validation set mAP.

[6]https://github.com/open-mmlab/OpenPCDet

We use a learning rate of 0.0005, weight decay of 0.1, and class sampling weightings of 1, 2, and 2 for the backpack, fire extinguisher and survivor respectively. The same n_{critic} and linear increase to α from the point-based experiment are used. Due to the dense cluttered environments in our dataset we set a very small voxel size of 1.8 cm which allowed clouds to fit onto the GPU. Due to the computational expense of these small voxels, the model took 25 hours to train on 4 Tesla K80 GPUs. We only changed two significant model parameters. To reduce pooling in these dense environments we increased the output pool size from 12 to 20. To encourage the detection of the sparse objects, we decreased the foreground threshold from 0.65 to 0.25.

Voxel-based methods are evaluated in the same manner as the point-based experiment. The results averaged over 3 training runs are shown in Table 5.4. To compute statistical significance, for each object we run a t-test with N = 691 and compare each of the best performing methods (in bold) with the the next-highest-performing method. Overall, the results are intuitive and follow the same trend as the point-based methods: SynAndReal is able to leverage the quantity of the synthetic data and the small amount of real data to perform better than OnlySyn and OnlyReal. PointDAN is unable to provide much improvement due to its focus on small fine-grained features. Different than for the point-based methods, our adversarial approach performs similarly to the GeodesicLoss approach. This could be because a structured voxelized representation does not require the increased flexibility of the adversarial discriminator. Additionally, due to difficulties implementing the WGAN-GP loss function for our voxel-based architecture, the use of a less-stable WGAN loss function could be to blame. In spite of this, our method still

Table 5.5: Effect of feature encoding level on detection performance

	Mean Average Precision (mAP)		
Architecture	Level2	Level3	Level4
Point-based	0.38	0.41	0.47
Voxel-based	0.25	0.31	0.21

does the best at detecting the fire extinguisher, and in terms of Mean Average Precision, if only by a small margin.

While the trends between the voxel-based and point-based architectures are similar, the performance of the voxel-based architecture is not as high overall. This could be because the dense environments VoteNet was designed for, more closely resemble the tight underground tunnels in our dataset.

Qualitative results for our voxel-based architecture can be found in Fig. 5.7. Overall, the detector once again is adept at detecting objects in this challenging environment. It is interesting to note that in Fig. 5.7a that there are few points on the survivor's legs and even though the label does not extend over the legs, the detector is able to use contextual information to extend outward in that direction.

5.3.3 Utility of Leveraging Feature Vector Across Encoding Levels

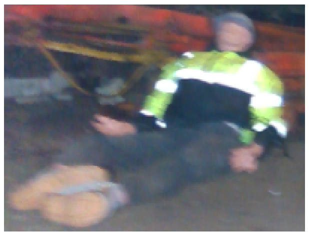
Fig. 5.2 shows the adversarial discriminator taking input from the fourth level of feature encoding (called "Level4"). In this experiment, for both the point and voxel-based architectures, we examine the effect of taking the feature vector input from different levels of the encoding pipeline.

The results are shown in Table 5.5. For the point-based architecture, the results are fairly intuitive: as the hierarchy of features are built up, the encoded feature vector is more useful to the adversarial discriminator. For the voxel-based architecture it is interesting to note that the performance goes down by using the most-encoded feature vector, *Level4*. This could be due to the more aggressive pooling that occurs with the PartA2 feature extraction pipeline.

5.4 Conclusion

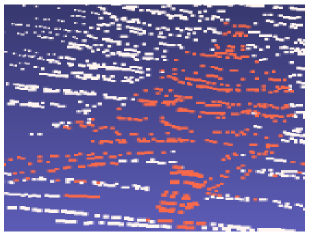
In this contribution we developed a 3D adversarial training architecture for point cloud object detection in robotic environments where real data is extremely limited. To our knowledge this is the first 3D adversarial approach targeting 3D robotic scenes. Our approach does not compress information into 2D and is lightweight enough to scale to 3D robotics scenes. Additionally, we developed an adaptive sampling module to maintain important 3D information throughout the discriminator. These characteristics offered large performance benefits for unstructured point-based representations.

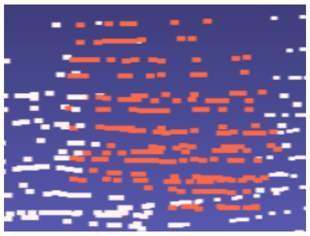
In this chapter we improved on the data mixing sim-to-real-process from our underwater sonar work by explicitly encoding the extraction of similar features from real and synthetic data into the training process. This brings the resulting model closer to a preferable scenario where the synthetic training is directly applicable to models deployed in the real-world.

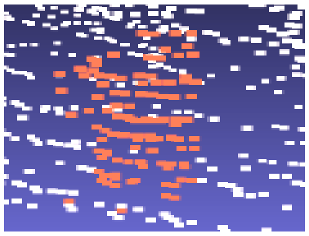

(a) Real survivor

(b) Real backpack

(c) Real fire ext.


(d) Real survivor point cloud (pink)

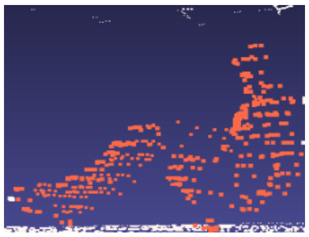

(e) Real backpack point cloud (pink)

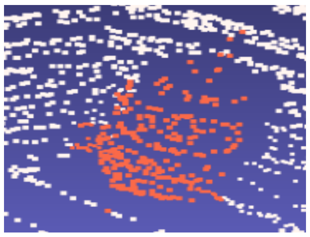

(f) Real fire ext. point cloud (pink)

(g) Synthetic survivor

(h) Synthetic backpack

(i) Synthetic fire extinguisher


(j) Synthetic survivor point cloud (pink)


(k) Synthetic backpack point cloud (pink)

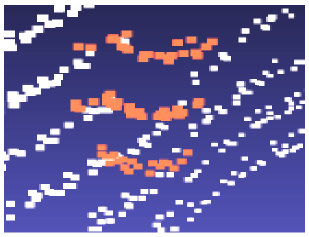
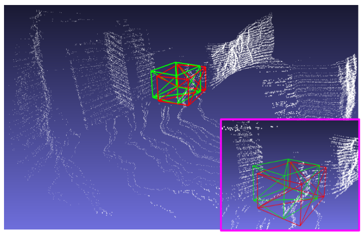
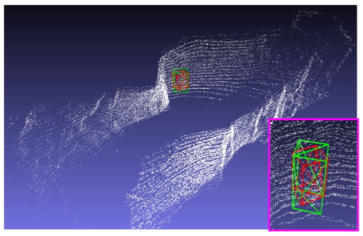

(l) Synthetic fire ext. point cloud (pink)

Figure 5.5: Dataset overview. (a)-(c) are the objects in camera images taken from the robot. Notice the motion blur that can occur in camera imagery underground. (d)-(f) are the objects in point cloud form. (g)-(i) and (j)-(l) show the camera images and point clouds from AirSim. Note that while the real and synthetic point cloud representations are similar, the simulated clouds are often cleaner and more regularly sampled.


(a) Survivor detection


(b) Backpack detection

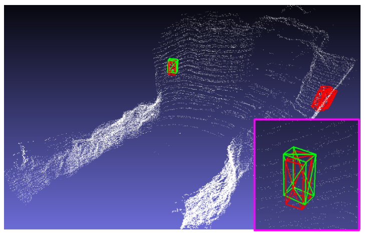
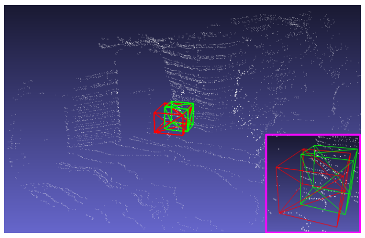
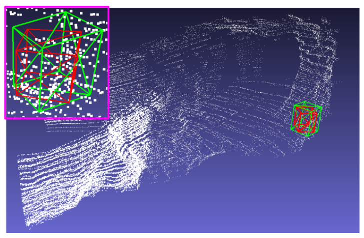

(c) Fire extinguisher detection

Figure 5.6: Qualitative results for our point-based adversarial architecture. Green boxes are ground truth, red boxes are predicted bounding boxes. The pink inset images are zoomed in on the detection shown in the larger scene. All clouds were captured in the Edgar Experimental Mine, CO as part of the DARPA SubT STIX Integration Exercise. Additional results with 3D manipulation of detections can be found at: https://youtu.be/oXk88gmsx8g


(a) Survivor detection


(b) Backpack detection

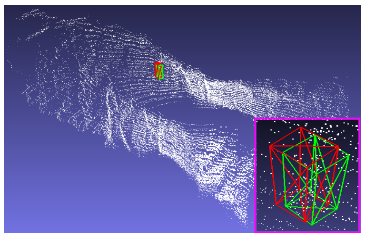

(c) Fire extinguisher detection

Figure 5.7: Qualitative results for our voxel-based adversarial architecture. Green boxes are ground truth, red boxes are predicted bounding boxes. All clouds were captured in the Edgar Experimental Mine, CO as part of the DARPA SubT STIX Integration Exercise. Additional results with 3D manipulation of detections can be found at: https://youtu.be/oXk88gmsx8g

Chapter 6: Conclusion

6.1 Conclusion and Future Work

In this thesis we make progress towards leveraging the power of deep learning in target environments that lack sufficient training data. We also demonstrate that by developing architectures which handle noise more explicitly, deep networks can be deployed that are less affected by the noise which often corrupts sensor data in harsh environments. Specifically:

- Our first contribution was a deep-learned method leveraging atrous convolution for generating real-time 3D reconstructions underwater. By using atrous convolution and the natural hierarchical structure of deep learning, we gain a larger context for sonar image analysis than previous works. We demonstrated that this larger context results in an architecture that is more noise-robust and overfits less to limited training data than competing methods. We then showed that using our image selection approach enabled human-in-the-loop real-time 3D reconstruction capabilities. This work resulted in a workshop and conference paper [19, 20] as well as code released for processing Tritech Gemini 720i sonar data [1].

- Our second contribution relieved the human from the reconstruction process by promoting a fully automatic method for generating 3D underwater reconstruction

[1]https://github.com/osurdml/GeminiECD_Decoder

methods without environmental priors. We achieved this by intelligently incorporating synthetic data into our training set and proposing a self-supervised approach for training a deep network on real 2D underwater sonar images where ground truth information is not available. To our knowledge, this is the first time such a network has been trained using real 2D underwater sonar imagery. We extended this into an approach which could take as input multiple images collected naturally as the robot moves around the world. We demonstrated that by leveraging the information from multiple images, even more accurate 3D reconstructions could be generated. This contribution lead to a conference paper [18].

- For our third contribution we took the incorporation of synthetic data a step further by developing a method for enforcing similar features extraction from real and synthetic data. Specifically, we developed an adversarial training approach to encourage similar feature extraction from real and synthetic point clouds on the 3D object detection in subterranean environments task. We demonstrated that by using our adversarial training architecture and incorporating an adaptive sampling module in the adversarial discriminator we could achieve a higher mean average precision on the 3D object detection task. This contribution lead to a journal paper [17] and the open-sourcing of the code/dataset[2] developed for the paper. This work was inspired by and developed in conjunction with a journal paper [81].

Overall, in this thesis we developed models and training approaches for leveraging the power of deep learning for perception tasks in harsh environments. Underwater

[2]https://github.com/debortoli/AdvNet3D

this enabled accurate 3D reconstructions to be generated without a human in the loop or without explicitly requiring environmental priors to hold true. Underground this enabled the more accurate detection of objects of interest, even in scenarios with very limited real-world data. These contributions should provide a basis for deploying powerful perception models even in scenarios where cameras have very limited visibility.

6.1.1 Future Work

There are a number of interesting directions for future work, particularly in the areas of leveraging multi-image inputs for underwater reconstruction, underwater sonar image dataset generation, and investigating low-level data representations for 3D point cloud analysis.

6.1.1.1 Explicitly Leveraging Additional Information in Multi-Sonar-Frame CNNs

One interesting future research direction from this thesis is more explicitly encoding the different types of information for our multi-frame sonar image CNNs. For example, to more explicitly encode pixel correspondences across images, a cost-volume approach could be taken where the cost for pixels at different elevation angles could be built into a 3D volume which is then fed into the network for further analysis [98]. This could reduce the amount that a multi-frame CNN is required to learn, thus enabling better training convergence.

It would also be interesting to investigate the use of Convolutional LSTMs as a framework that is less-affected by the noise in a single image. By utilizing the different gates (e.g. forget gates) particularly poor images could be ignored or down-weighted to reduce its effect on the final elevation angle estimation.

6.1.1.2 Generation of a Large-Scale 3D Ground-Truthed Sonar Dataset

Fueled by large corporations and great research interest, terrestrial domains have seen a recent explosion of large-scale labeled datasets for the purpose of training deep learning networks. Examples of this include the KITTI [26], Audi [27] and Waymo [91] datasets all for the autonomous driving application. However, the same explosion of labeled datasets has yet to take place underwater.

While there does exist some datasets for RGB imagery underwater [57], the sonar 3D reconstruction task is particularly lacking large ground-truthed datasets. The generation of such a dataset is non-trivial and would involve determining the correspondence between a point in sonar image space with a 3D point in the world. Using the lessons learned from underwater RGB dataset generation such as [28] would be great place to start.

6.1.1.3 Investigating Point Cloud Data At a Lower-Level

In this thesis we explored the use of purely point-based or purely voxel-based data representations for 3D point cloud object detection. We found that each representation had its advantages: with point-based methods better at detecting backpacks and survivors and voxel-based approaches better at detecting the fire extinguisher object. In the future, it would be interesting to explore different data representations such as point-voxel based approaches which can potentially leverage the benefits of both representations [84].

It would also be interesting to explore leveraging the intensity information from the lidar. In this thesis we used a custom SLAM solution [115] which did not natively output this information in the generated point clouds, however intensity information has been shown to be useful in a variety of scenarios including object detection [12]. Finally, it would be interesting to more explicitly address challenges in lidar data integrity, such as saturation or blooming, which can occur with objects very close or very far away from the sensor respectively [46].

Bibliography

[1] DARPA subterranean challenge [online]. `www.subtchallenge.com`.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proc. International Conference on Machine Learning*, pages 214–223, 2017.

[3] Murat D Aykin. *3D reconstruction and motion estimation using forward looking sonar*. PhD thesis, University of Miami, 2015.

[4] Murat D Aykin and Shahriar Negahdaripour. Forward-look 2D sonar image formation and 3D reconstruction. In *Proc. IEEE/MTS OCEANS Conference, San Diego, CA*, 2013.

[5] Murat D Aykin and Shahriar Negahdaripour. Three-dimensional target reconstruction from multiple 2D forward-scan sonar views by space carving. *IEEE Journal of Oceanic Engineering*, 42(3):574–589, 2017.

[6] Murat D Aykin and Shahriar S Negahdaripour. Modeling 2D lens-based forward-scan sonar imagery for targets with diffuse reflectance. *IEEE Journal of Oceanic Engineering*, 41(3):569–582, 2016.

[7] Lee J Baumgartner, Nathan Reynoldson, Leo Cameron, Justin Stanger, et al. Assessment of a dual-frequency identification sonar (DIDSON) for application in fish migration studies. *NSW Department of Primary Industries-Fisheries Final Report Series*, 84:1–33, 2006.

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[9] Heiko Bülow and Andreas Birk. Spectral registration of noisy sonar data for underwater 3D mapping. *Autonomous Robots*, 30(3):307–331, 2011.

[10] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180, 2017.

[11] Rômulo Cerqueira, Tiago Trocoli, Gustavo Neves, Sylvain Joyeux, Jan Albiez, and Luciano Oliveira. A novel GPU-based sonar simulator for real-time applications. *Computers & Graphics*, 68:66–76, 2017.

[12] Erzhuo Che, Jaehoon Jung, and Michael J Olsen. Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review. *Sensors*, 19(4):810, 2019.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 40(4):834–848, 2017.

[14] Woen-Sug Choi, Derek R Olson, Duane Davis, Mabel Zhang, Andy Racson, Brian Bingham, Michael McCarrin, Carson Vogt, and Jessica Herman. Physics-based modelling and simulation of multibeam echosounder perception for autonomous underwater manipulation. *Frontiers in Robotics and AI*, 8, 2021.

[15] Enrique Coiras, Yvan Petillot, and David M Lane. Multiresolution 3D reconstruction from side-scan sonar images. *IEEE Transactions on Image Processing*, 16(2):382–390, 2007.

[16] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[17] Robert DeBortoli, Li Fuxin, Ashish Kapoor, and Geoffrey A Hollinger. Adversarial training on point clouds for sim-to-real 3D object detection. *IEEE Robotics and Automation Letters*, 6(4):6662–6669, 2021.

[18] Robert DeBortoli, Fuxin Li, and Geoffrey A Hollinger. Elevatenet: A convolutional neural network for estimating the missing dimension in 2D underwater sonar images. In *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 8040–8047, 2019.

[19] Robert DeBortoli, Austin Nicolai, Fuxin Li, and Geoffrey A. Hollinger. Assessing perception quality in sonar images using global context. In *Proc. IEEE Conference on Intelligent Robots and Systems Workshop on Introspective Methods for Reliable Autonomy, Vancouver, CA*, 2017.

[20] Robert DeBortoli, Austin Nicolai, Fuxin Li, and Geoffrey A Hollinger. Real-time underwater 3D reconstruction using global context and active labeling. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 6204–6211, 2018.

[21] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2366–2374, 2014.

[22] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. 3D mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014.

[23] Nathaniel Fairfield, George Kantor, and David Wettergreen. Real-time SLAM with octree evidence grids for exploration in underwater tunnels. *Journal of Field Robotics*, 24(1-2):03–21, 2007.

[24] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[25] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *Proc. European Conference on Computer Vision*, pages 740–756. Springer, 2016.

[26] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[27] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, et al. A2d2: Audi autonomous driving dataset. *arXiv preprint arXiv:2004.06320*, 2020.

[28] Arturo Gomez Chavez, Andrea Ranieri, Davide Chiarella, Enrica Zereik, Anja Babić, and Andreas Birk. Caddy underwater stereo-vision dataset for human–robot interaction (hri) in the context of diver activities. *Journal of Marine Science and Engineering*, 7(1):16, 2019.

[29] Daniel Gordon, Ali Farhadi, and Dieter Fox. Re3: Real-time recurrent regression networks for visual tracking of generic objects. *IEEE Robotics and Automation Letters*, 3(2):788–795, 2018.

[30] Thomas Guerneve and Yvan Petillot. Underwater 3D reconstruction using blue-view imaging sonar. In *Proc. IEEE/MTS OCEANS Conference, Genova, IT*, 2015.

[31] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In *Proc. Conference on Neural Information Processing Systems*, pages 5769–5779, 2017.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[33] Richard P Hodges. *Underwater Acoustics: Analysis, Design and Performance of Sonar*. John Wiley & Sons, 2011.

[34] Geoffrey A Hollinger, Brendan Englot, Franz Hover, Urbashi Mitra, and Gaurav S Sukhatme. Uncertainty-driven view planning for underwater inspection. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4884–4891, 2012.

[35] Berthold KP Horn and Michael J Brooks. *Shape from shading*. MIT press, 1989.

[36] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[37] Franz S Hover, Ryan M Eustice, Ayoung Kim, Brendan Englot, Hordur Johannsson, Michael Kaess, and John J Leonard. Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *The International Journal of Robotics Research*, 31(12):1445–1464, 2012.

[38] Tiffany A Huang and Michael Kaess. Towards acoustic structure from motion for imaging sonar. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 758–765, 2015.

[39] Tiffany A Huang and Michael Kaess. Incremental data association for acoustic structure from motion. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1334–1341, 2016.

[40] Natàlia Hurtós, David Ribas, Xavier Cufí, Yvan Petillot, and Joaquim Salvi. Fourier-based registration for robust forward-looking sonar mosaicing in low-visibility underwater environments. *Journal of Field Robotics*, 32(1):123–151, 2015.

[41] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[42] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[43] 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*, pages 224–230, 2016.

[44] Matthew Johnson-Roberson, Oscar Pizarro, Stefan B Williams, and Ian Mahon. Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys. *Journal of Field Robotics*, 27(1):21–51, 2010.

[45] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[46] Alireza G Kashani, Michael J Olsen, Christopher E Parrish, and Nicholas Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15(11):28099–28128, 2015.

[47] Juhwan Kim, Hyeonwoo Cho, Juhyun Pyo, Byeongjin Kim, and Son-Cheol Yu. The convolution neural network based agent vehicle detection using forward-looking sonar image. In *Proc. IEEE/MTS OCEANS, Monterey, CA*, 2016.

[48] Juhwan Kim and Son-Cheol Yu. Convolutional neural network-based real-time rov detection using forward-looking sonar image. In *Proc. IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 396–400, 2016.

[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[50] Prabin Kumar Rath, Alejandro Ramirez-Serrano, and Dilip Kumar Pratihar. Real-time moving object detection and removal from 3D pointcloud data for humanoid navigation in dense gps-denied environments. *Engineering Reports*, 2(12):e12275, 2020.

[51] Kevin Lai and Dieter Fox. Object recognition in 3D point clouds using web data and domain adaptation. *The International Journal of Robotics Research*, 29(8):1019–1037, 2010.

[52] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *International Conference on 3D Vision (3DV)*, pages 239–248, 2016.

[53] Nicholas Lawrance, Robert DeBortoli, Dylan Jones, Seth McCammon, Lauren Milliken, Austin Nicolai, Thane Somers, and Geoffrey Hollinger. Shared autonomy for low-cost underwater vehicles. *Journal of Field Robotics*, 36(3):495–516, 2019.

[54] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[55] Eon-ho Lee and Sejin 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*, pages 497–500, 2016.

[56] Bo Li. 3D fully convolutional network for vehicle detection in point cloud. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518, 2017.

[57] Chongyi Li, Chunle Guo, Wenqi Ren, Runmin Cong, Junhui Hou, Sam Kwong, and Dacheng Tao. An underwater image enhancement benchmark dataset and beyond. *IEEE Transactions on Image Processing*, 29:4376–4389, 2019.

[58] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[59] John Makhoul. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):27–34, 1980.

[60] Daniel Maturana and Sebastian Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[61] John McKay, Isaac Gerg, Vishal Monga, and Raghu Raj. What's mine is yours: Pretrained CNNs for limited training sonar ATR. In *Proc. IEEE/MTS OCEANS Conference, Anchorage, AK*, 2017.

[62] Kaichun Mo, He Wang, Xinchen Yan, and Leonidas Guibas. PT2PC: Learning to generate 3D point cloud shapes from part tree conditions. In *European Conference on Computer Vision*, pages 683–701, 2020.

[63] Shahriar Negahdaripour, Pezhman Firoozfam, and Payam Sabzmeydani. On processing and registration of forward-scan acoustic video imagery. In *Proc. Canadian Conference on Computer and Robot Vision*, pages 452–459, 2005.

[64] José Neira and Juan D Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[65] Ehsan Nezhadarya, Ehsan Taghavi, Ryan Razani, Bingbing Liu, and Jun Luo. Adaptive hierarchical down-sampling for point cloud classification. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12956–12964, 2020.

[66] Paul Ozog, Giancarlo Troni, Michael Kaess, Ryan M Eustice, and Matthew Johnson-Roberson. Building 3D mosaics from an autonomous underwater vehicle, doppler velocity log, and 2D imaging sonar. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1137–1143, 2015.

[67] Yan Pailhas and Yvan Petillot. Large mimo sonar systems: a tool for underwater surveillance. In *Sensor Signal Processing for Defence (SSPD)*, pages 1–5, 2014.

[68] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. *Advances in Neural Information Processing Systems Autodiff Workshop*, 2017.

[69] Kaustubh Pathak, Andreas Birk, and Narunas Vaskevicius. Plane-based registration of sonar data for underwater 3D mapping. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4880–4885, 2010.

[70] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[71] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[72] Horia Porav, Tom Bruls, and Paul Newman. I can see clearly now: Image restoration via de-raining. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7093, 2019.

[73] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep Hough voting for 3D object detection in point clouds. In *Proc. IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019.

[74] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[75] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[76] Can Qin, Haoxuan You, Lichen Wang, C.-C. Jay Kuo, and Yun Fu. PointDAN: A multi-scale 3D domain adaption network for point cloud representation. In *Proc. Conference on Neural Information Processing Systems*, 2019.

[77] Scott Reed, Yvan Petillot, and J Bell. Automated approach to classification of mine-like objects in sidescan sonar using highlight and shadow information. *Proc. IEEE Radar, Sonar and Navigation*, 151(1):48–56, 2004.

[78] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on*

Medical Image Computing and Computer-Assisted Intervention, pages 234–241. Springer, 2015.

[79] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[80] Connor Schenck and Dieter Fox. Perceiving and reasoning about liquids using fully convolutional networks. *The International Journal of Robotics Research*, 37(4-5):452–471, 2018.

[81] Sebastian Scherer, Vasu Agrawal, Graeme Best, Chao Cao, Katarina Cujic, and Robert DeBortoli et al. Resilient and modular subterranean exploration with a team of roving and flying robots. *Field Robotics*, To appear.

[82] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[83] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, pages 621–635. Springer, 2018.

[84] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3D object detection. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[85] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-A2 net: 3D part-aware and aggregation neural network for object detection from point cloud. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2019.

[86] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[87] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-YOLO: An Euler-region-proposal for real-time 3D object detection on point clouds. In *European Conference on Computer Vision*, pages 197–209, 2018.

[88] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[89] Irvin Sobel. An isotropic 3× 3 image gradient operator. *Machine Vision for Three-Dimensional Scenes*, pages 376–379, 1990.

[90] Suraj Srinivas, Ravi Kiran Sarvadevabhatla, Konda Reddy Mopuri, Nikita Prabhu, Srinivas SS Kruthiventi, and R Venkatesh Babu. A taxonomy of deep convolutional neural nets for computer vision. *Frontiers in Robotics and AI*, 2:36, 2016.

[91] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. *arXiv*, pages arXiv–1912, 2019.

[92] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[93] Pedro V Teixeira, Michael Kaess, Franz S Hover, and John J Leonard. Underwater inspection using sonar-based volumetric submaps. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4288–4295, 2016.

[94] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[95] Abhinav Valada, Gabriel L Oliveira, Thomas Brox, and Wolfram Burgard. Deep multispectral semantic scene understanding of forested environments using multimodal fusion. In *International Symposium on Experimental Robotics*, pages 465–477. Springer, 2016.

[96] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, pages 10–15607. Rome, Italy, 2015.

[97] Jinkun Wang, Shi Bai, and Brendan Englot. Underwater localization and 3D mapping of submerged structures with a single-beam scanning sonar. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4898–4905, 2017.

[98] Kaixuan Wang and Shaojie Shen. Mvdepthnet: Real-time multiview depth estimation neural network. In *Proc. International Conference on 3D Vision (3DV)*, pages 248–257. IEEE, 2018.

[99] Weidong Wang, Wei Dong, Yanyu Su, Dongmei Wu, and Zhijiang Du. Development of search-and-rescue robots for underground coal mine applications. *Journal of Field Robotics*, 31(3):386–407, 2014.

[100] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in Germany, test in the USA: Making 3D object detectors generalize. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11713–11723, 2020.

[101] Yusheng Wang, Yonghoon Ji, Dingyu Liu, Hiroshi Tsuchiya, Atsushi Yamashita, and Hajime Asama. Elevation angle estimation in 2D acoustic images using pseudo front view. *IEEE Robotics and Automation Letters*, 6(2):1535–1542, 2021.

[102] Eric Westman, Akshay Hinduja, and Michael Kaess. Feature-based SLAM for imaging sonar with under-constrained landmarks. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3629–3636, 2018.

[103] Eric Westman and Michael Kaess. Wide aperture imaging sonar reconstruction using generative models. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8067–8074, 2019.

[104] Rob Weston, Sarah Cen, Paul Newman, and Ingmar Posner. Probably unknown: Deep inverse sensor modelling radar. In *Proc. International Conference on Robotics and Automation (ICRA)*, pages 5446–5452, 2019.

[105] David P Williams. On optimal AUV track-spacing for underwater mine detection. In *Proc. IEEE International Conference on Robotics and Automation*, pages 4755–4762, 2010.

[106] David P Williams. Underwater target classification in synthetic aperture sonar imagery using deep convolutional neural networks. In *International Conference on Pattern Recognition (ICPR)*, pages 2497–2502, 2016.

[107] David P Williams and Samantha Dugelay. Multi-view sas image classification using deep learning. In *Proc. IEEE/MTS OCEANS Conference, Monterey, CA*, 2016.

[108] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4376–4382, 2019.

[109] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3D point clouds. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

[110] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016.

[111] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5589–5598, 2020.

[112] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3D: Self-training for unsupervised domain adaptation on 3D object detection. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10368–10378, 2021.

[113] Dana R Yoerger, Albert M Bradley, Barrie B Walden, M-H Cormier, and William BF Ryan. Fine-scale seafloor survey in rugged deep-ocean terrain with an autonomous robot. In *Proc. IEEE International Conference on Robotics and Automation(ICRA)*, pages 1787–1792, 2000.

[114] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *Proc. of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico*, 2016.

[115] Ji Zhang and Sanjiv Singh. LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.