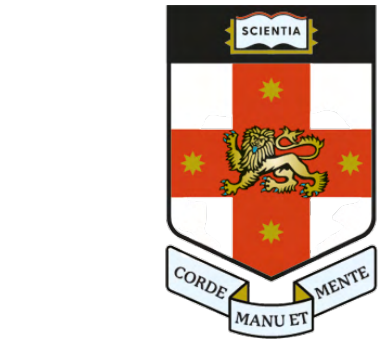


UNSW
SYDNEY

School of Mechanical and Manufacturing Engineering
Faculty of Engineering
UNSW Sydney

BY

Liam Viney

Underwater Object Classification Using Multibeam Sonar

Thesis submitted as a requirement for the degree of Bachelor of Engineering in Mechanical Engineering

Submitted: November 15, 2024	Student zID: z5257559
Supervisor: Will Midgley (UNSW)	

ORIGINALITY STATEMENT

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed: Liam Viney
Date: November 15, 2024

Abstract

In recent years, machine learning has greatly advanced in computer vision, primarily using optical images. However, similar techniques should be capable of achieving state-of-the-art results utilizing sonar data. Most prior sonar research has been limited to test tanks, which do not capture the complexity of open-water environments. This paper demonstrates the feasibility of applying machine learning to detect man-made objects in open-water sonar images. The paper presents a dataset collected and labelled from open water, model architectures, and training methods. Results indicate promising potential for adapting computer vision techniques to underwater object detection, suggesting future avenues for research.

Acknowledgments

I want to sincerely thank Dr. Will Midgley, my thesis supervisor, for all of his help, encouragement, and support during this research project. His knowledge and perceptions were really helpful to me in overcoming the obstacles this project presented. Additionally, I would like to express my gratitude to the University of New South Wales' Mechatronics Department for offering a first-rate research setting and the tools I needed to finish this project. I also appreciate Reach Robotics' assistance in supplying the sonar tools and materials I needed for my project. Sam Hall's help in configuring the sonar devices and his insightful comments during the study process are greatly appreciated. Lastly, I want to express my gratitude to my family and friends for their support and encouragement, which motivated me to persevere and complete this thesis.

Contents

List of Figures

List of Tables

Nomenclature

AUV Autonomous Underwater Vehicle

BCE Binary Cross-Entropy

CNN Convolutional Neural Network

CPU Central Processing Unit

CSV Comma Separated Value

CUDA Computer Unified Device Architecture

FFS Forward Facing Sonar

GAN Generative Adversarial Networks

GPS Global Positioning System

GPU Graphics Processing Unit

K-SVD K-singular value decomposition

KLT Kanade-Lucas-Tomas

LiDAR Light Detection and Ranging

LSTM Long Short-Term Memory

MSE Mean Squared Error

PSNR Peak Signal-to-Noise Ratio

R-CNN region-based convolutional neural network

RADAR Radio Detection and Ranging

ReLU rectified linear unit

RGB Red, Green, Blue

ROV Remotely Operated Underwater Vehicle

RRT Rapidly Exploring Random Trees

SONAR Sound Navigation and Ranging

SSS Side Scanning Sonar

UUV Unscrewed Underwater Vehicle

YOLO You Only Look Once

1 Introduction

Imaging sonar is a key sensor for situational awareness in underwater environments, however, interpreting the data requires significant experience and training. The images produced contain a distance and horizontal angle, as opposed to the horizontal and vertical angles of optical images, complicating their interpretation. Sonar images also only have a single channel for signal intensity providing less information than Red, Green, and Blue (RGB) images. Despite the complexities of interpreting the images, sonars are often used for object detection and classification tasks. A recent example is during the clean-up of the Francis Scott Key Bridge site [14], where they were used alongside commercial divers. The systems allow for the detection of objects at ranges far exceeding optical sensors in the underwater environment. Unfortunately, the requirement for a trained operator restricts the use of the systems. In an attempt to reduce the cognitive load on operators, several traditional object detection techniques have been applied to sonars, however, they have produced limited benefits. Machine learning techniques promise to outperform them.

Previous machine learning studies using sonar data have concentrated on side-scanning sonar (SSS), with very few papers using forward-facing sonar (FFS). This is despite forward-facing systems being the preferred sensor type for site inspections like the Francis Scott Key Bridge site. FFS provides detailed close-range images and a more advantageous view of objects in cluttered environments.

Existing works using FFS are confined to test tanks and fail to capture the complexities of real-world environments. This paper builds on these works by demonstrating the use of machine learning to identify man-made objects in open-water conditions. Due to the lack of existing public open water sonar data, a dataset was collected and labelled for use during the project.

Reach Robotics in Sydney, provided invaluable support throughout the project, allowing for the use of their sonar equipment, data collection sites, and computational resources. The data set was collected on an Oculus M750d Dual-Frequency Multibeam Sonar provided by Reach, enabling high-resolution sonar data collection. Data was gathered in Sydney Harbor using a Strategic Robotic Systems FUSION remotely operated underwater vehicle (ROV), allowing simultaneous optical and sonar data capture, from a variety of locations. The project focused on man-made objects commonly encountered in inspection scenarios, as these structures typically have distinctive shapes and edges that are more easily discernible in sonar images.

Various model architectures were trained and tested, producing variable results. Despite failing to produce a model with cutting-edge performance, the ability to use machine learning techniques on sonar data was successfully demonstrated. This proof of concept paves the way for practical systems to be developed and deployed.

2 Literature Review

2.1 Submerged Sensing Methods

2.1.1 Optical Sensors

Optical Sensors have been extensively developed and employed in a wide range of applications for obstacle detection. Optical video capture devices bring many inherent benefits, including comparably

low cost, low physical footprint, and the ability to perform object classification and detection.

Most systems use a field of artificial intelligence, known as computer vision, to analyse the incoming video streams. Modern systems are capable of identifying and classifying obstacles with high levels of accuracy [15]. An example of this is YOLO (You Only Look Once) [16], an open-source, deep-learning algorithm capable of tracking and identifying objects using video input.

However, finding an object's range using an optical sensor is a more complex problem. A common solution is a stereo camera system, as seen in the paper, Implementation of a Real-Time Stereo Vision System [17]. The authors of the paper use a second camera to triangulate the distances to objects. These systems require calibration to function properly and are adversely affected by changing light conditions, making them ill-suited to applications outside of controlled environments [18].

Despite many successful implementations of computer vision for obstacle detection, optical sensors have fundamental issues when used in submerged environments. When light passes through water it collides with water molecules, transferring energy to heat and preventing light from travelling long distances. Water also causes high levels of light reflection, refraction and diffusion. The short distance travelled by light results in its availability becoming progressively worse the deeper a sensor is operating [19]. Submerged environments are often turbid, further reducing the effectiveness of computer vision techniques. This can be seen in Figure 1

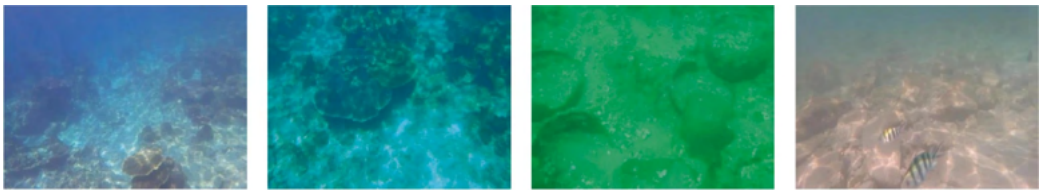


Figure 1: Examples of degraded underwater images [1]

The lack of natural light in deeper waters can be addressed with an artificial light source, however, this still suffers from light's inability to travel long distances through water. Artificial light sources create an additional problem known as back-scattering, caused by particles suspended in the water reflecting light back at the source. The reflections from these particles reduce visibility and cause images to be noisy [19].

2.1.2 Acoustic Sensors

Due to the high attenuation of electromagnetic waves in water, many sensing methods such as Light Detection and Ranging (LiDAR), Global Positioning System (GPS), and Radio Detecting and Ranging (RADAR) cannot be used in submerged environments. Other than optical sensors the only other commonly used sensor type is acoustic. Systems that use acoustics for sensing the environment are known as Sonar Systems (Sound Navigation and Ranging).

Passive Sonar One method of sensing using sound waves is known as passive sonar. It involves the use of hydrophones to detect sound emitted by objects in the surrounding environment. The lack of detectable emissions makes the method popular in military applications, allowing systems to operate without revealing their presence. A paper from the Polish Naval Academy [20] details a stereo

hydrophone system that can detect and triangulate the position of vessels with low-speed propellers. These systems have several shortcomings, being susceptible to interference and noise, and they are unable to detect objects without a sound signature. If zero emissions is not a system requirement it is generally more practical to use active sonar.

Active Sonar Active sonar is a time-of-flight system that measures the time taken for a sound wave to reflect off an object and return to the system. Sonars send out a short pulse of sound known as a "ping". The system then stays silent to detect any returning echoes. If the speed of sound is known, the distance to the object (R) that caused the echo can be calculated using the following equation:

$$R = \frac{c\,t}{2} \tag{1}$$

Where

$$c = \text{speed of sound}$$
$$t = \text{time elapsed from ping till echo return}$$

Water is significantly denser than air, with the ocean having an average density of $1028.13\,\mathrm{kg/m^3}$ [21], while air has an average density of $1.293\,\mathrm{kg/m^3}$ [22]. This results in sound waves travelling much faster and further in water, making active sonar systems particularly useful for detecting objects underwater. Due to sonar's lower frequencies compared to optical systems, they have much lower resolutions. The Ping360 sonar [2] is a high-frequency sonar designed for use on ROVs. It has an operating frequency of 750 kHz and it has a resolution of 0.08% of its range. This equates to a 1.6mm resolution at 2m, which is far less than optical systems. Higher frequency systems do exist, however, high-frequency sound waves lose more energy to the water resulting in lower sonar ranges [23]. For example, there are systems used to profile the ocean floor with ranges of up to 1000m, and a frequency of 4 Hz to 50 Hz, however, their resolution can be worse than 2 m [23].

Active Sonar is also very susceptible to noise, complicating obstacle-detection methods. The noise can be mainly attributed to sonar pings reverberating off the bottom, environment noise, and self-generated noise [24]. This noise results in sonar data being speckled with false signals. This requires signal processing to remove noise before sonar data is usable. Possible methods for this are discussed in the next section.

2.1.3 Active Sonar Types

Mechanically Scanned Sonar Mechanically scanned sonars use a mechanical system to steer a sonar array, allowing the system to stitch together a wider image. They typically consist of a single transceiver and receiver, and a stepper motor to steer the array. The Ping360 sonar is an example of a mechanical sonar system [2], seen in figure 2. The stepper motor rotates the array in increments, pausing at each step to obtain a sonar reading. Each reading is then stitched together to generate an image. Mechanically scanned sonars can achieve 360-degree scans at significantly lower prices than multibeam sonars [4], however, the time taken to make a complete scan is a major drawback. The time taken to rotate the array causes the Ping360 to take anywhere from 9 to 35 seconds to complete a full

rotation. This can become a significant issue as the velocities of the vehicle or objects it is tracking increase.

Figure 2: Ping360 Mechanically Scanned Sonar [2].

Multibeam Sonar Multibeam Sonars (Figure 5) typically have one to three transducers that emit a ping, and the resulting reflections are detected by an array of smaller transducers. The array is designed to detect echoes coming from a range of bearings, and beam-forming techniques are used to increase the resolution. Most are capable of forming between 64 and 256 listening transducers and are typically arrayed to detect echos in a 120-degree arc [25]. Since these systems are often mounted on the front of a vehicle they are often referred to as forward-facing sonars (FFS). Multibeam Sonar's main benefit over mechanical sonars is its significantly faster refresh rates and resolutions. There are several factors that enable FFS to achieve higher resolutions [26]. The absence of mechanical error allows for smaller and higher frequency transducers to be used, increasing potential resolution. The array of receivers allows for the use of beamforming techniques, Doppler analysis, and beam overlapping, increasing the resolution of the final image. The high refresh rate greatly reduces the effects of movement on the sonar image. This is a major issue for mechanical sonars, limiting their use to mostly static environments. FFS are also referred to as imaging sonars due to their high resolution and high refresh rates. The comparison in Figure 14a and Figure 14b demonstrates the greater resolutions offered by FFS systems.

Despite these benefits, Mechanical sonars remain in use in some fields due to their wider field of view and significantly lower unit costs [4]. The Blue Robotics Ping360 mechanical scanning sonar retails for $2650 US [2], while a low-end multibeam sonar, the Sonoptix ECHO, retails for $9000 US [27].

Side Scanning Sonar Side scanning sonars (SSS) (Figure 6) are typically used for scanning large areas of the sea bed. They emit a fan-shaped ping, allowing them to scan a slice of the bottom. When

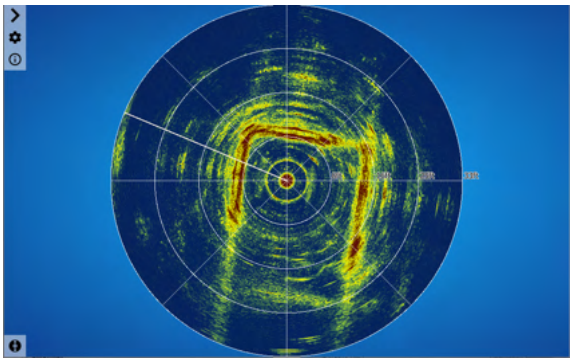


Figure 3: Ping360 mechanically scanned sonar return [2].

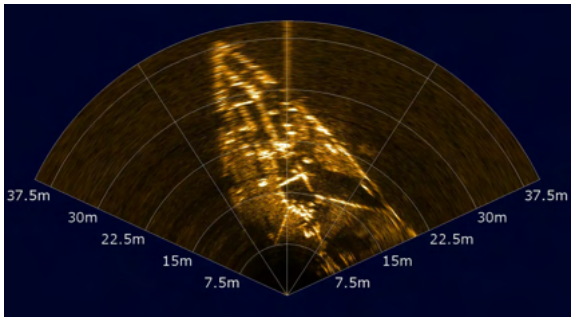


Figure 4: Oculus multibeam imaging sonar return [3].

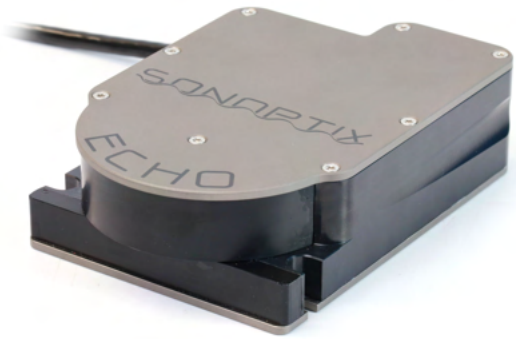


Figure 5: Sonoptix ECHO Multibeam Imaging Sonar [4].

they are moved along a path each scan can be stitched together to form a larger image [4]. SSS can be mounted on a vehicle but are often towed behind, allowing the sonar to operate deeper and away from vehicle noise [19].

Other Sensor Types

There are several other novel approaches to sensing objects underwater, such as haptic sensors. the authors of [28] used haptic sensors on a hydraulic gripper to explore underwater objects. While the system was successfully tested, it is impractical for obstacle detection systems used for route planning, and situational awareness due to the need to come into contact with obstacles to detect them.

Researchers are also exploring the potential of laser imaging underwater, as it promises better resolution and response times than acoustic sensors. The authors of [29] use lasers within the blue-green spectrum to reflect off of objects. The blue-green spectrum is used due to its minimal attenuation

Figure 6: Cerulean Omniscan 450 Side Scan Sonar [5].

in water, increasing their range. The reflections are then gathered by a CCD camera, and processed to create an image of the environment. The system is still inferior in range to acoustic systems, and its resolution advantage is often unnecessary for obstacle-detection systems. The system would be better suited to object classification tasks or point cloud generation for robotic arm control.

The fusion of optical and acoustic sensors is a growing field of research, allowing for the shortcomings of each sensor type to be mitigated by the other [30]. It is particularly useful for object detection and classification systems and has been used to better visualise sonar data for users [31]. However, the technology is still being developed and there are minimal commercial examples. Sensor fusion also requires more computing power, which is not always available on uncrewed underwater vehicles (UUV) and ROVs.

2.2 Noise Reduction Techniques

Sonar returns are inherently noisy signals, due to the transmitted pulses of sound reverberating in the environment. Sound can reverberate off the bottom, the water surface, and suspended particles in the water [32]. Noise from the environment generated by the sensor platform can also cause noisy sonar returns [24]. Reducing the level of noise in the data is an important step before the data is used for its intended purpose. Many techniques for achieving this are covered in the literature, and some of the more prominent methods are detailed below.

2.2.1 Filtering and Thresholding

Filtering and thresholding are commonly used techniques in image processing, and they can also be adapted to sonar images. A 2016 paper on sonar image matching [13], presents an analysis of filtering techniques. The paper tested four filtering techniques including; the mean method; the median method;

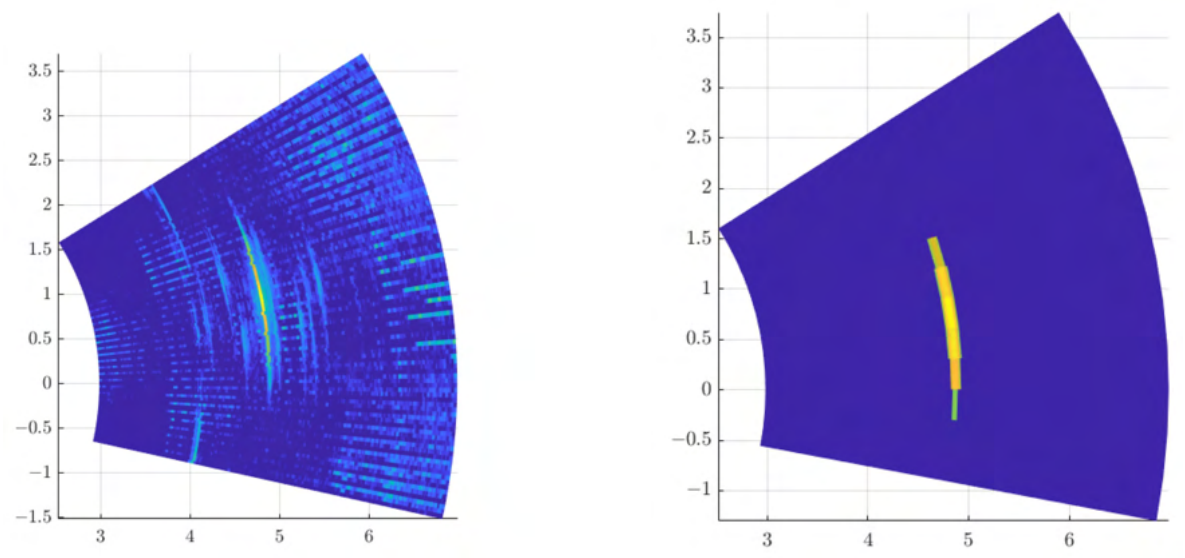
the morphological smoothing method; and, technology based on wavelet transforms. Each method was tested on the same sonar with Gaussian noise applied to it. The time taken to apply each filter and their performance was compared, using the Peak signal-to-noise ratio (PSNR), and the mean squared error (MSE). The results of this comparison can be seen in Table 1. If the filtering method is to be used in real-time, high speed becomes an important requirement. The paper chose to use the Median Method for this reason, and its better performance when compared to the Mean Method.

Table 1: Comparison of Filtering Methods from [13]

Method	**Time (s)**	**PSNR**	**MSE**
Mean Method	0.002032	23.7308	0.0041
Median Method	0.045952	23.8318	0.0040
Wavelet Transforms	0.368323	23.0682	0.0048
Morphological Smoothing	0.189010	20.1384	0.0096

The Median method uses a small window, commonly 5 by 5 pixels, to scan the sonar image. The median value of each pixel in the window is found and then applied to the centre pixel. The window is then shifted by a one-pixel row or column and applied again until the entire image has been filtered.

Rafal Kot implemented a mean filter that showed a significant reduction in noise on an FFS sonar image [6], although larger reverberations remained in the sonar image. By applying a threshold filter the image was dramatically improved. The combination of a mean or median filter with thresholding appears to be very effective and is not overly computationally expensive when compared to other denoising techniques. The results of this method can be seen in figure 7. However, its simplicity may result in an inability to remove larger reverberations, such as echoes off pool walls. It is however an attractive solution and should be considered if noise becomes an issue.



(a) Before processing

(b) After processing

Figure 7: Sonar data denoising by Rafal Kot [6]

2.2.2 Image Denoising by Bilateral Filtering

Bilateral filtering is another classical image-denoising technique, that promises to better preserve edges inside of an image. Each pixel in an image is replaced by a weighted average of the neighbouring pixel intensities. The weights are calculated using both the spatial and intensity domains of the neighbouring pixels and then multiplied by the intensity of the current pixel. The sum of these calculations is then divided by the sum of the original weights [33].

The authors of [33] implemented a bilateral filter together with a dual-domain filtering technique. This involved splitting the image into high and low-contrast layers. A bilateral filter was used on the high-contrast layer, and a wavelet filter was used on the low-contrast layer. The two layers were then added together to produce the denoised image. It showed a high level of effectiveness on optical images and has the potential to be used with sonar images. However, due to the sonar system's poor resolution, preserving edges is less important, and the method has a higher computational cost than the mean and threshold filtering used by [6]. The method does show promise for implementation with sonar images and should be further studied as an alternative to median and threshold filtering.

2.2.3 Dictionary Learning Methods

Dictionary learning [34] involves representing an image as a linear combination of simple elements, called atoms. The atoms are stored in a matrix called the dictionary, and when multiplied by a sparse vector they can be added together to create a patch of an image. Each image patch has a unique sparse vector. Denoising is commonly done by thresholding the small coefficients in the sparse vector of each patch. The dictionary must be learned using a training set of images. This is typically achieved through optimization techniques, such as K-singular value decomposition (K-SVD). The authors of [35] implemented a dictionary learning method for an SSS with good results. They highlight the method's advantages in computation time and required training set size over deep learning systems. While promising, dictionary learning methods are significantly more complicated than traditional filtering methods, and the increased performance may not be required for a sonar-based object detection method.

2.2.4 Mean Image Subtraction

Shi Zhao wrote a paper on obstacle detection and tracking using imaging sonar [36], and used a novel method for eliminating noise and ghost echos from the test tank walls. The sonar system was fixed in position in the tank and a series of sonar scans were recorded. This data was then averaged and subtracted from data collected from subsequent tests. Opening and closing operations were then performed on the resulting image. This method appears to have worked remarkably well, producing clean scans of obstacles in the tank. There is minimal noise and no ghost echoes from the tank walls. However, this method also removes the tank walls and relies on the sonar remaining stationary. Despite its obvious limitations the method's simplicity and apparent success at removing noise make it an attractive option. Its issues may be able to be solved using image segmentation to ignore known objects while taking a moving average of the rest of the scan.

2.3 Obstacle Detection Methods

Obstacle detection has been widely studied in the literature, with methods growing in complexity as available computation power has increased. This has allowed for greater accuracy, speed of updates, resolution and information about the object in question. In many review papers object detection methods have been split up into methods for known environments with static obstacles, and methods for unknown environments [37]. Efficient methods for known environments have been well studied but have limited practicality, and as such will not be covered in this report. There have also been many studies that use simulated sonar data for testing, however, simulated sonar data has been shown to have significant limitations. The data fails to incorporate the variability that occurs in open water and methods that are only tested in simulated environments may not be robust or reliable in open water [38]. These papers will also be omitted from the report. Many obstacle-detection methods incorporate collision avoidance or path-finding methods, as this is often the primary goal for the detection of obstacles. When applicable this report will briefly detail these aspects.

2.3.1 Point Cloud

Point clouds [39] are used in many obstacle detection and avoidance systems. Returns from a sensor such as a sonar are analysed and groups of points that are close together are clustered to form a new single point in space. If data points do not form a cluster they are discarded. Methods for clustering points vary, although the radially based nearest neighbour clustering method is common [39]. It uses radial distances from each sensor return to determine if they should form a cluster with the surrounding points. Point clouds offer several benefits, such as noise reduction and reduced data storage requirements. It is also easily implemented in real-time systems and is often used as part of other systems such as artificial potential fields [40]. The necessity for points to form clusters restricts the applicability of this method to sonars with sufficiently high resolutions.

2.3.2 Occupancy Grid

The occupancy grid method involves the partition of a scanned area into cells. Each cell is then either assigned a binary value or a probability, indicating if an object is present in the cell. The author of [41] implements two occupancy grid systems for a mechanical array. The first is the vehicle attached grid, representing the area currently in view by the sonar. The grid is initialised with a probability 0.5 in each cell. Cells are updated using a dynamic thresholding function on the intensity of the sonar return. The threshold reduces with the distance of the return and has a user-defined maximum range if signals are approaching the threshold value they are assigned a probability between 0 and 1. The cells are larger than the sonar resolution, allowing for multiple returns to contribute to the value of a cell. This helps mitigate the effects of noise on sonar returns. The second grid is the global occupancy grid which stays stationary. The global grid receives updates from the vehicle-attached grid every time there is a vehicle position update, and stores information about objects no longer in view. The author of [42], also implements an occupancy grid system, but with a single grid system that shifts the objects in the grid with an affine transformation, each time there is an update of the vehicle position.

The authors of [43] further improve the occupancy grid method by using a Bayesian update to

update each cell's probability. Using relatively large cells the authors were able to effectively detect obstacles with a low-cost and resolution sonar. This method also has relatively low computational and data requirements, however, it has a low resolution and is not designed to track moving objects. The Bayesian update also limits the operating speed of the vehicle, as multiple scans are required to build confidence about object locations.

2.3.3 Edge Detection

Edge detection is capable of extracting more information about objects, such as dimensions and objects' centre of mass. This enables larger objects to be identified and tracked as one object rather than a grouping of individual sonar returns. A detected object can be stored with a single location coordinate and its dimensions measured from that point, or its shape can be stored as a template. The edges can be detected using traditional image processing techniques. The authors [44] used an autonomous ground vehicle to test different edge-detection techniques and found that a Canny edge detector worked best.

The author of [45] uses a very similar method to that detailed above, using a corner detection algorithm instead of an edge detector. The corners are then used to create templates of each object which can be compared between frames. The author assumes that all the objects are stationary, allowing them to be used to calculate the vehicle velocity. If templates match between frames they are assumed to be the same object and the relative distance travelled by the object is calculated. Optimisation functions are then used to find the vehicle velocity. This method could be used to estimate the velocity of moving objects if the vehicle velocity is already known. Alternatively, a Kalman filter could be used to combine the sonar measurements with an inertial measurement unit to more accurately estimate the vehicle state.

2.4 Neural Networks and Deep Learning with Sonar Data

Since the release of the Alex Net convolutional neural network [46], neural networks and deep learning have become the state of the art for computer vision systems. The literature is rich with work on neural networks for optical sensors, however, work with sonar systems has been limited. Several contributing factors have limited the work on deep learning with sonar data, mainly the lack of publicly available data sets and the reduced resolution of sonar systems compared to optical [47]. Despite these issues, the number of papers using deep learning for sonar image analysis has been steadily increasing in recent years, and in some cases have outperformed classical approaches. Review papers predict neural networks and deep learning will replace classical approaches and recommend researchers focus their attention on the field [47], [48], [49].

Sonar research has been historically dominated by the defence industry, resulting in much of the work not being publicly available. Sonar data is also very expensive to collect, requiring significant resources and time. The data that does exist comes in a variety of forms, with most collected by passive sonar, and the rest split between side scanning and FFS sonars. This further limits the useful data for specific applications. Training neural networks require large amounts of varied data to be successful, prompting researchers to use standard augmentation [50] to enlarge data sets or create synthetic data [51]. These methods struggle to simulate real-world data, and if there is insufficient real-world data

in the training set, the model will be prone to overfitting. Overfitting can also occur when models are trained on data collected in pools, resulting in poor performance in open water.

The poor resolution of sonar systems limits the use of neural networks trained on optical images. Current high-resolution sonar systems have resolutions measured in centimetres, using a single data channel instead of the three channels in RGB (red, green, blue) images. Additionally, sonar is affected by unique types of noise such as reverberations. Simple applications of the current models for RGB images fail when provided with sonar data [47]. Neural Networks designed for optical images can be used with sonar data, however, they must be adapted. The authors of [52], adapt the YOLO (you only look once) model to work with FFS sonar. They introduced an attention mechanism to the model, which improved the detection accuracy, however, the model was trained on data collected in a test tank.

Due to the large amounts of noise present in sonar data, many object detection papers first use some form of image prepossessing on the data. Classical image processing techniques, such as filtering and thresholding are used by some papers [6]. There are examples of neural networks being trained to remove noise in sonar images [47], and many authors argue that networks should be trained on raw noisy data. This allows networks to learn to deal with the noise, becoming more robust. Regardless of whether denoising techniques are used on training data, small ROV class sonars, such as the Oculus M series [8], apply their own denoising algorithms.

2.4.1 Enlarging Data Sets

There are currently two main methods used in the literature to enlarge sonar data sets, standard augmentation and generative adversarial networks (GAN) [47]. Standard augmentation involves applying changes to an existing data set to enlarge it. This can include adding white Gaussian noise, image rotations, and flipping. GANs are used to create synthetic data, similar to the original data set. Both methods have been shown to increase the generalisation of data sets, improving the performance of machine learning models. Most object classification papers implement some form of data set enlargement. It is particularly useful for training with sonar data due to the high cost of collection, however, certain techniques should be avoided. In particular, vertical flipping should be avoided as it changes the location of sonar shadows relative to objects. This increase in variability increases the required training set size without increasing a model's potential functionality.

2.4.2 Transfer Learning

Transfer learning involves the use of a preexisting model or data set to initialise the weights, and can help with training times and model performance. If a model solves a similar task to the desired task it can be taken and retrained with the new data set. This leverages the knowledge already in the original model and may enable the model to recognise patterns that are difficult to learn with the available training set. Models can have every layer trained with the new data, or layers can be selectively trained. It is common practice to keep all but the final layer of an existing model the same. This allows for the feature extraction of the original model to be kept. The final layer and any additional layers are trained to look for features from the classes of interest [53]. An example of transfer learning with the

YOLO model can be found in [52], and the authors state that their transferred model outperformed state-of-the-art region-based convolutional neural network (R-CNN).

If a suitable model does not exist or a model is not similar enough, pre-training can achieve similar results. Pre-training involves training neural networks with a similar data set before training with the desired training set. The ImageNet [54] database has been commonly used to pre-train sonar image networks [47]. The paper [55] is interesting because of its use of a synthetic aperture radar data set for pre-training. Radar and sonar data have high-level similarity, and the authors show that models pre-trained on this data have improved performance.

Transfer learning and pre-training have been particularly useful for sonar-based systems, reducing the required sonar data set size. It should be noted that transfer learning is not always successful. The authors of [56] compared few-shot learning methods (See section 2.4.3) with pre-trained weights from the ImageNet database. The models trained from scratch with the few-shot techniques had better results.

There are two forward-facing sonar data sets that have the potential for use in pre-training models. The first is the Shipwreck-Dataset [12], which contains sonar images from an Oculus M1200d sonar of four different shipwrecks. It is the only publicly available data set containing forward-facing sonar images in open water. The other is a data set of ten types of marine debris, recorded in a small water tank [57]. The data in the tank has been labelled with bounding boxes.

2.4.3 Few-Shot Learning

Few-shot learning is a field in machine learning that helps models train more effectively when there is a limited amount of training data. One example is Prototypical Networks [58]. The training set is broken up into classes, which contain similar features and are then used to train the model. Each class is placed in a non-linear embedding space. When the model makes a prediction, it is placed in the embedding space and assigned to the class within the shortest distance to it within the space. A test image will be assigned to the class with the lowest distance measurement. Relation Networks, Soft K-means, and Consistent Prototypical Networks are other few-shot learning types. Unfortunately, few-shot learning significantly underperforms compared to larger machine learning models, with cutting-edge models achieving around 70% accuracy [59]. These papers achieve this with semantically very different classes, making the feature extraction process easier. Unfortunately, most real-world applications of machine learning involve semantically similar classes, significantly reducing the performance of few-shot models. This challenge is likely exacerbated when using sonar data, due to its limited information and the high degree of visual similarity between objects. Nonetheless, the high cost and scarcity of sonar data make the potential success of these methods particularly desirable.

2.4.4 Classification Models

Research on classification models for sonar has mostly been focused on SSS and synthetic aperture sonars. These sonar types produce images with a similar view to optical images, allowing neural networks developed for optical images to be more easily used. They have been successful at detecting and classifying a wide variety of objects, from sediment types [60], to man-made objects such as shipwrecks [61], and fishing nets [62]. Figure 8 shows an example of a synthetic aperture sonar image,

which has many similarities to a traditional optical image. Despite the larger quantity of papers using SSS and synthetic aperture sonar, the systems do not provide the real-time imaging capability of FFS.

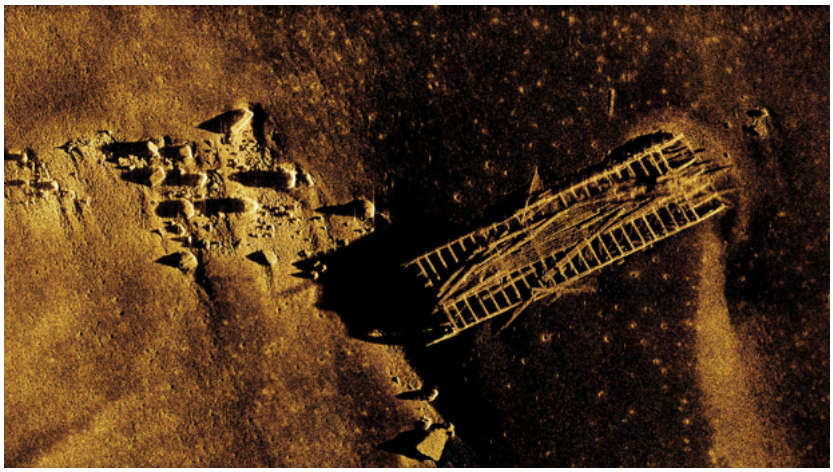


Figure 8: Example of synthetic aperture sonar image [7].

In [60] the authors successfully identified a range of different sediment types using a small training data set. They were able to pre-train their models using the greyscale CIFAR-10 image data set, using a Generative adversarial network to augment the data set and create synthetic sonar data snippets. They found that deeper models benefited more from the pre-training and had the best performance with DenseNet-151 and ResNet-4-2 models.

The authors of [61] were interested in using machine learning to identify underwater archaeological sites. Traditional image processing techniques were used to extract features before being input into the neural network. One set of inputs for the neural network used traditional feature extraction methods to detect areas of interest in the image, while an edge detector was used to detect another set of areas of interest. The areas of interest were segmented into sub-images, which were used to train the network. The paper states that the model had low precision, possibly caused by the small size of the training set. The paper indicates that feature extraction does not significantly help with classification models.

The paper [62] is interesting as it uses an FFS sonar to classify fishing nets, plastic bags and cloth. Papers using neural networks with FFS sonars are less common than side scanning and synthetic aperture sonars, having less similarity to classic optical images. This complicates transfer learning with open-source classification models, with the pre-trained feature extraction not guaranteed to be effective. The authors create a multiple receptive field network based on CenterNet, with good classification performance. This paper demonstrates that FFS sonars can be successfully used for classification tasks. Figure 9 demonstrates the different ways information is displayed in an FFS image. The image contains information about distance, horizontal angle, and signal intensity. This is different to RGB images, which contain horizontal and vertical angles, and three channels of signal intensity.

Matias Alejandro published a PhD paper titled, Deep Neural Networks For Marine Debris Detection In Sonar Images [57]. He achieved an impressive 99.7% accuracy on a classification task, using deep neural networks with custom convolutional blocks. Although the paper provided an impressive proof of concept, aspects of the testing methodology resulted in the work not being directly appli-

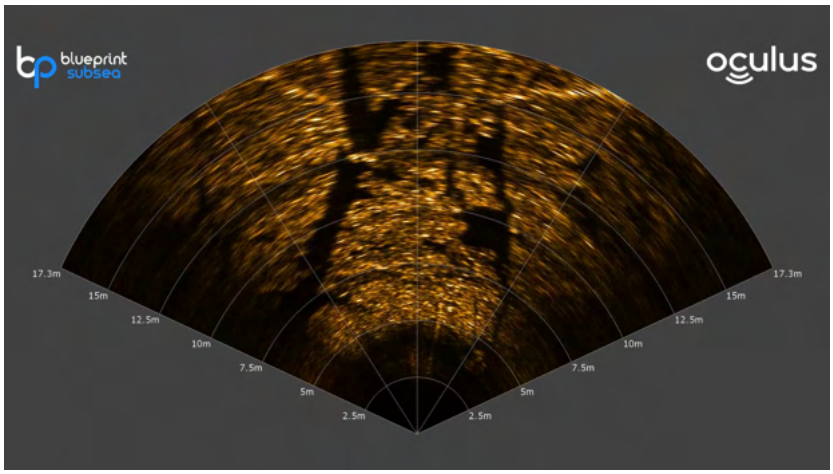


Figure 9: Example of forward facing sonar image [8].

cable to possible real-world applications. Ten classes were chosen and placed in a small water tank. This removes much of the irregularities and noise present in open water sonar images and introduces reverberations from the walls of the tank. The sonar images were also cropped around the object of interest before being fed to the model. Cropping can dramatically improve learning rates, however, it makes the use of the model in real-time impossible. At the time of writing no papers could be found that train classification models exclusively on uncropped forward-facing sonar data collected in open water.

2.5 Promising Machine Learning Models

The literature is full of work on classification models for optical images, with new model architectures being regularly released and adopted. Despite the differences between sonar and optical images, the cutting-edge model architectures designed for optical images would likely perform strongly when trained on sonar data. While not a direct example, the authors of RainNet [63] used a U-Net [64] to predict future rainfall. U-Net was originally designed for biomedical image segmentation, demonstrating the versatility of cutting-edge neural network architectures.

2.5.1 Convolutional Neural Networks (CNN)

Convolutional layers are a fundamental tool in machine learning for vision tasks, such as classification. They apply a small filter or kernel to a larger input matrix. The filter is moved across the input in steps, outputting the dot product of the overlapping matrices at each step, which produces a new matrix, known as a feature map. The values of the filter are learned during model training through backpropagation. Multiple filters can be used to increase the number of channels in the model, as seen in Figure 10. Networks built using convolutional layers often include repeating blocks, which typically consist of a normalization layer, a convolution layer, a rectified linear unit (ReLU) activation function, and a pooling layer [65] [9]. Pooling layers reduce dimensionality, allowing the model to focus on important features. In classification models, the output feature maps are flattened after

passing through the convolutional blocks. These flattened outputs are then fully connected to nodes representing the classes, where each node contains the logit values for each class.

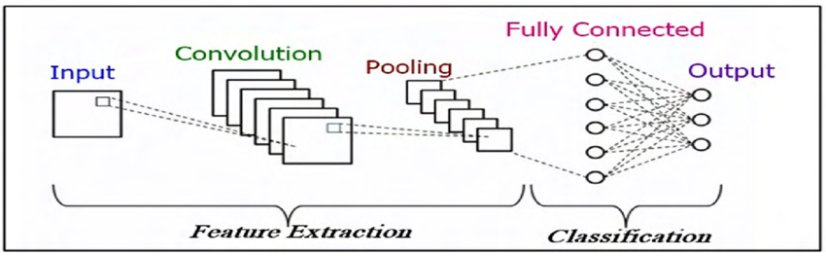


Figure 10: Block diagram of CNN architecture [9].

2.5.2 Res-Net

In 2015 a team from Microsoft released a paper titled Deep Residual Learning for Image Recognition [66]. It produced very successful results in several benchmarks and is still considered to be a cutting-edge architecture. The model uses CNN blocks with skip connections bypassing some CNN blocks. This is depicted in Figure 11. The Res-Net architecture solved the problem of exploding and vanishing gradients during training. These issues would stop a model from training if they were made too deep [67].

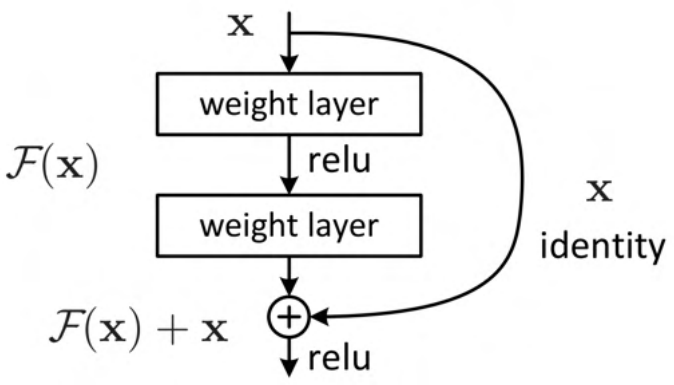


Figure 11: Block diagram of Res-Net architecture [9].

2.5.3 Models with Time Dimension

Forward-facing sonar data is considered not view-invariant, meaning that objects can not always be classified from any viewing angle. As a result machine learning models could struggle to classify individual images, and it may be beneficial to design models that have access to historical images from a video, or have memory.

Three Dimensional Convolutional Layers One method is feeding a model several time stamps at once. Sam Black attempts this technique using three-dimensional convolutional layers, in the paper Deep Convolutional Networks for Monocular Velocity Estimation. [68]. It is the simplest method of providing a classification model with historical information.

Recurrent Neural Networks Another method of adding temporal memory to a model is the Recurrent neural network [69]. The model uses its output as one of the input channels, providing information about the past with every new input. Figure 12 depicts the feedback loop of a recurrent neural network

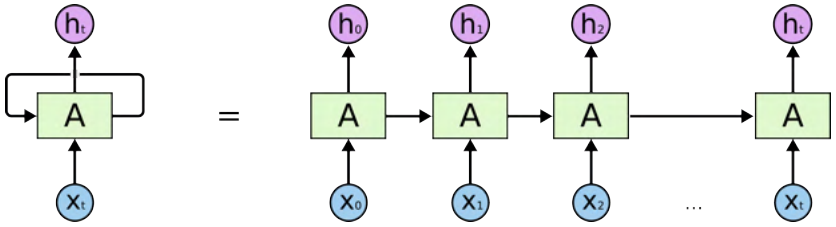


Figure 12: Block diagram of Recurrent Neural Network [10].

Long Short Term Memory (LSTM)

2.6 Literature Review Conclusion

Future research into obstacle detection for ROVs and AUVs (Autonomous Underwater vehicles) should focus on machine learning because of its potential to bring better performance and new capabilities when compared to classical methods. While optical and passive sonar sensors have use cases for which they are best suited, active sonar has the widest range of applications, making it an important field of research. Although numerous papers already use machine learning for sonar data most focus on SSS, with comparably few papers using FFS. As ROVs and AUVs become more prevalent the use of FFS is going to grow, as it provides the optimal field of view for several tasks. This includes route finding, object detection, and physical manipulation tasks. Due to the superior performance of multibeam sonars, there are no papers training machine learning models with mechanical sonars. Mechanical sonar has system size, weight, and cost advantages. The Blue Robotics Ping360 mechanical scanning sonar retails for $2650 US [2], while the Sonoptix ECHO Multibeam Imaging Sonar from Blue Robotics retails for $9000 US [27]. Despite this, commercial operations have almost completely adopted multibeam systems, as the cost savings are generally insignificant. Following this conclusion, it is the opinion of this review that further research on underwater object detection should focus on training neural networks on multibeam sonar data.

3 Methodology

3.1 Identifying Target Sector

The literature review reveals a gap in the literature that exists on the application of machine learning techniques to multibeam sonar data. There has been a limited amount of work done in controlled environments, however, these works fail to prove the ability of these systems to operate in the real world. This project aims to address this by gathering open-water sonar data for the training of neural networks, providing a practical example of success. To provide further direction to the project design and goals, it was decided to frame the work around a target sector that has a practical use for an automated sonar system.

Recreational and Competitive Fishing Recreational and competitive fishing presents a potentially large market for an automated sonar system. Multibeam sonars are already in widespread use in competitions, allowing anglers to precisely locate and identify fish, and monitor how fish react to different techniques. Competitive anglers report using multibeam sonar to identify individual fish, and changing their techniques based on the fish's reaction [70]. Unfortunately using the sonar in this fashion requires experience and constant attention. Many fishermen enjoy the sport because of the time spent in the natural environment, and the requirement to monitor a display could dissuade from the adoption of existing systems. A neural network that could autonomously detect fish and alert the anglers would solve this issue.

Unfortunately, the author's lack of fishing knowledge and experience creates significant barriers to the creation of such a system. The collection and labelling of the required data would require extensive research and is prohibitively difficult for the scope of the project. While an automated fish classification system has the potential for commercial success it was decided to pursue a different target sector.

Site Inspection An alternative use case was inspired by the collapse of the Francis Scott Key Bridge in Baltimore [71]. The accident created an unknown underwater environment, which had to be surveyed as quickly and safely as possible. Due to the unknown stability of the wreckage, sending divers or manned vessels to work around the site carried significant risk. According to USNI News, divers working in the wreckage had visibility of 1 to 2 feet [14], significantly complicating their work. A sonar system capable of detecting man-made objects would be valuable in this situation.

The Scott Key Bridge incident pushed the project towards the detection of man-made objects, for site inspections. A small forward-facing imaging sonar mounted on an inspection class ROV would allow the system to be quickly deployed, with minimal cost, supporting infrastructure, and operator training. At the start of 2024, the School of Mechanical Engineering at UNSW was in the process of acquiring the equipment required, however, the timeline was uncertain. After searching for alternative options, Reach Robotics in Sydney expressed an interest in the project. The company offered to assist with the project by providing their FUSION ROV, and computer resources.

3.2 Available Resources

The available resources constrained the project's methodology. The support offered by Reach Robotics provided high-quality equipment, allowing the project to increase in scope. Before securing their assistance the project would have been trained on a data set collected with a Ping360 mechanical sonar, with the majority of the data sourced from the UNSW pool. As discussed in sections 2.1.3 and 2.4, both the use of the mechanical sonar and a pool are undesirable compromises. Mechanical sonars provide inferior performance to multibeam sonars, and data gathered in pools or test tanks inhibits the generalization of models to open water environments. As such, without Reach Robotics, the project's goal of an open water system would not have been realized.

3.2.1 Oculus M750d Dual-Frequency Multibeam Sonar

The Oculus M series sonars [72] are a family of forward-facing multi-beam imaging sonars, widely used in industry on inspection class ROVs. The M750d version owned by Reach Robotics, can operate at 750 KHz or 1.2 MHz, providing detailed sonar images at ranges up to 120 m or 40 m for the respective operating frequencies. As is evident in figure 13, operating at 1.2 MHz, captures significantly higher detail. This is likely to make convergence during model training easier than 750 KHz data. All the data captured in this project was at 1.2 MHz.

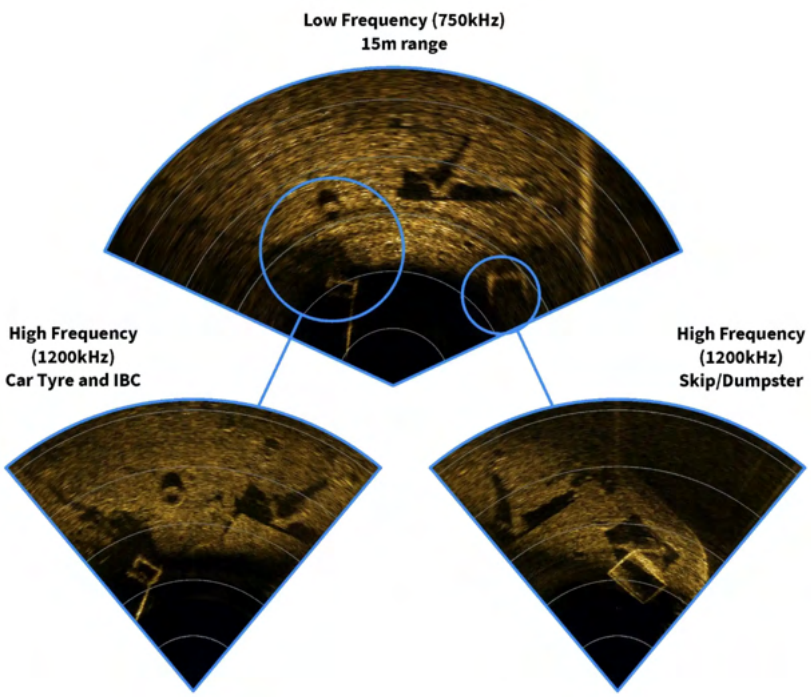


Figure 13: M750d operating frequency comparison [11].

3.2.2 Strategic Robotic Systems FUSION ROV

The Strategic Robotic Systems FUSION ROV is an inspection class ROV provided by Reach Robotics, capable of operating without an external power supply. The unit at Reach Robotics came with a 500 m optical tether, however, tethers of up to 2000 m can be purchased. The ROV's sensors included a HD camera, which can collect optical video alongside the sonar. The simple operation of the ROV has made it a popular system with many organizations, including the US Coast Guard, and is a perfect platform for the Oculus M750d sonar. If the project developed into a commercial product it would likely be deployed on a FUSION or a similarly sized ROV.

3.2.3 Nvidia GeForce RTX 3080

To train neural networks, a graphics processing unit (GPU) that can run CUDA (Compute Unified Device Architecture) is required, as training on CPUs is prohibitively slow. While the University of New South Wales possesses CUDA-compatible machines, they are shared by several students. Training models for multiple days would have been unfair to other students and risky, as memory requests from other students' work could consume all the GPU's resources and crash the training script. Fortunately, Reach Robotics had a machine with a Nvidia GeForce RTX 3080 graphics card, which they were willing to use for training the project's models.

3.2.4 Access to Sydney Harbor

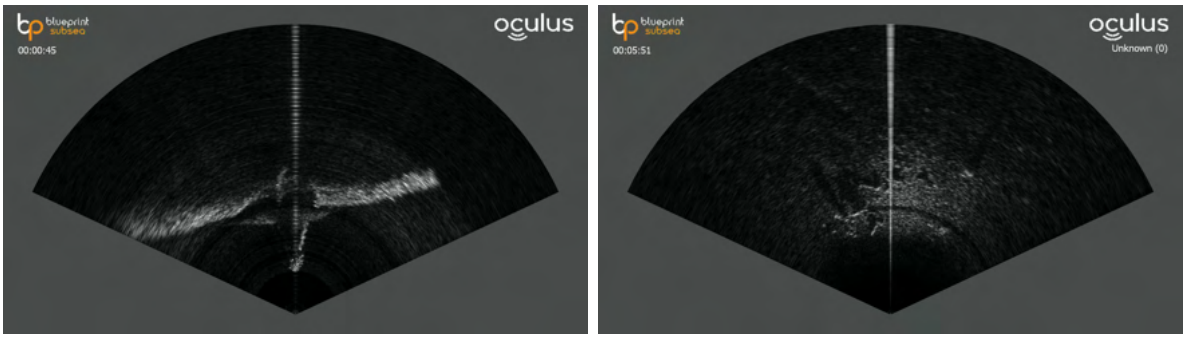
Reach Robotics has access to several areas in Sydney Harbor where it tests its products. The company allowed sonar data to be collected at these sites whenever it was conducting its testing and was another vital contribution to the project.

3.3 Identifying Desired Classes

In order to train an object detection model using a supervised learning technique the dataset must contain a set of defined classes. The choice of classes significantly affects the convergence of future models. To achieve faster convergence, the classes should be as distinct as possible. This makes it easier for the model to extract important features, as discussed in Section 2.4.3. In practical applications, this is not always possible due to the constraints of the task.

For this project, it was decided to focus on recording man-made objects, which could be easily sourced in Sydney Harbor. Reach Robotics has access to several industrialized areas for testing, providing environments full of objects commonly found at incidents, such as the Scott Key Bridge [14]. Man-made objects were also hypothesized to offer greater visual differences between classes compared to natural objects. They tend to have distinct sides, curves, and corners, distinguishable in sonar images. Figure 14 shows an example of the distinct sides and corners in sonar images of man-made objects.

Man-made objects are easier to label due to their distinct shapes and can be manually classified without extensive research. In contrast, labelling natural objects would have required additional research due to a lack of subject matter expertise.



(a) Ships stern, including propeller and rudder. (b) Seabed containing no man-made objects

Figure 14: Comparison of man-made and natural objects in sonar images.

Although the class list was unknown before obtaining data, a general idea of what was desired helped prioritise time during recording sessions on the harbour. The testing requirements of Reach Robotics limited the use of the ROV, and efficient use of time was important. Large and obvious objects were prioritized, minimizing decision-making time during field testing.

3.4 Data Collection

3.4.1 Public Data Sets

The first data acquired was a publicly available Sonar and Optical image data set, named The Shipwreck-Dataset [12]. Produced for a paper titled Self-Calibration of a Sonar–Vision System for Underwater Vehicles, it was ideal for pre-training the project models or increasing the project's data set. The dataset's sonar images were taken alongside the optical images, allowing for easy classification of the sonar data. The images were from three different shipwrecks and occasionally contained other objects such as tyres and ROVs. The Shipwreck Dataset was the only public sonar data recorded in open water. A couple of other datasets were found, however, they contained data from test tanks, [57], [73]. Information gathered in the literature review suggested that test tank data increased the risk of overfitting, leading to their omission from the project. A data set produced by Xie, K., Yang, J. and Qiu, K [74] in a lake appeared promising, however, difficulties parsing the data caused the data to be discarded.

3.4.2 Producing New Data

The primary data was collected with Reach Robotics at Glebe Point Rowing Club and the Cape Don lighthouse tender at Balls Point. When a team from Reach Robotics was testing products in the harbour the sonar feed was recorded alongside their testing. Due to the company's testing requirements, there was minimal flexibility to record specific objects, however a wide variate was captured. Drawing on experience with The Shipwreck Dataset, the optical camera on the ROV was used to record data with the sonar, simplifying the classification of objects in the sonar data. Figure 15 shows the FUSION ROV used to capture the data.

Figure 15: FUSION ROV used to collect primary data.

3.5 Labeling Process

The process of labelling data for model training was the most time-consuming. Various processes were attempted each with different issues, before settling on the final labelling workflow. Once a method was chosen the process still consumed significant time, constraining the project's scope. By the completion of the project, less than an eighth of the collected data was labeled, due to the need to spend time on other tasks.

3.5.1 Label Type

The labelling workflow required a decision on the type of label to be used. There were three main labelling types applicable for a site inspection system including whole frame labelling, bounding box labels, and segmentation labelling. The initial aim was to implement a segmentation mask, which involves assigning a label to each pixel. It provides the most information about objects and offers flexibility when using the labels. should they be needed, bounding boxes or whole frame labels can be created using the existing semantic labels. Despite the versatility of segmentation masks, the time required is prohibitively long. Open source tools such as Label Studio [75] provide tools for manually drawing segmentation masks over images, however, this was not a realistic option for the number of labels required. Several paid services claim to provide automated labelling tools, however, their effectiveness on sonar data was doubtful and was not experimented with.

The second option was bounding box labels, providing information on an object's class, location, and approximate size. They also allow for whole frame labels to be easily created. Initially, rotated bounding boxes were used for their ability to conform to the shape of an object. This was soon changed to standard bounding boxes, as they are quicker to produce, and rotated boxes are rarely used in the machine learning community. Many machine learning and labelling tools do not support them, and there appears to be a consensus that rotated bounding boxes provide minimal benefit and high computational cost [76].

3.5.2 Labeling Software

The open-source tool Label Studio [75] was experimented with, providing a relatively straightforward process for creating bounding box labels, however, each label had to be individually drawn. The collected data contained multiple labels in each video frame, and there was a clear need to find a tool for accelerating the process. MATLAB's ground truth labelling apps appeared to provide a potential solution, containing several tools for automating the labelling process. Tutorials available on MathWorks showed the tools to be highly effective, and the decision was made to transition the labelling process to MATLAB.

There were a few teething issues with the transition. One difficult issue was MATLAB's alphabetical ordering of data. The Shipwereck-Dataset came as a series of images instead of a video file, and Matlab reorganized the images into alphabetical order. This caused an image named '100' to come before an image named '9'. The issue was fixed by padding image names with zeros, however, diagnosing the issue took longer than expected. Another issue was converting MATLAB's file types to comma-separated value (CSV) files, which could be used with PyTorch. A script was found online, solving the issue.

3.5.3 ACF Detector and YOLO Labeling

The main attraction of MATLAB was its aggregate channel features (ACF) object detector tool. It could be trained using a sample of ground truth labels and promised to automatically draw bounding boxes on the remaining instances of an object. The online tutorials demonstrated a high effectiveness, with minimal incorrect labels. An attempt was made to train an ACF object detector for the detection of a shipwreck, however the results were disappointing. As seen in figure 16, The detector would place bounding boxes over any bright area of noise in the image. This experience highlights the issues of using sonar data with traditional computer vision tools.

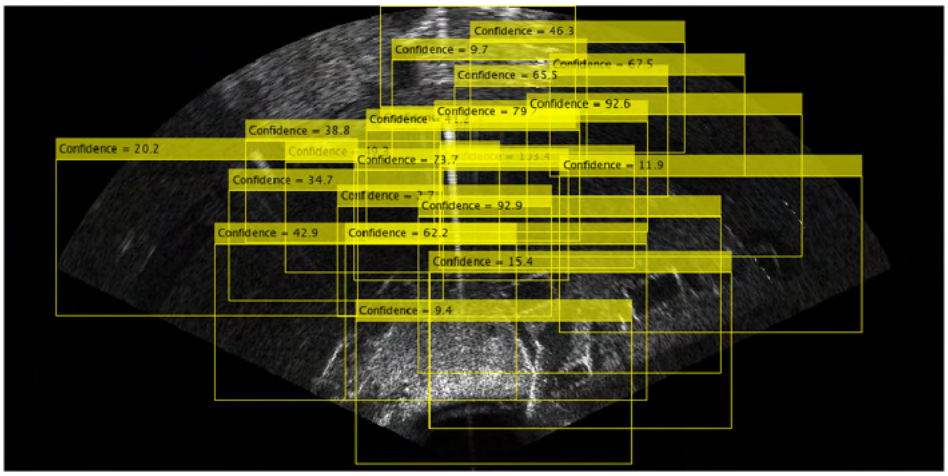


Figure 16: Attempt to use ACF object detector in MATLAB.

This was followed with an attempt to retrain a YOLO model in MATLAB, however, converting the ground truth labels into the correct format proved overly difficult and the decision was made to try the less sophisticated tools available. If a future project attempts to use MATLAB's labelling tools it should be noted that the MathWorks tutorials demonstrate best-case scenarios, and are unlikely to represent true performance. Online support for these tools was also difficult to find.

3.5.4 Temporal Interpolator

Two other promising labelling tools on MATLAB included the point tracker and the temporal interpolator. The point tracker tool used a Kanade-Lucas-Tomasi (KLT), feature-tracking algorithm, and was capable of labeling a few consecutive frames. unfortunately, it quickly produced errors, and the time required for corrections negated any time saved by the tool.

The final tool experimented with was the temporal interpolator. It averaged the change between two ground truth labels across the intervening frames. This proved to be the most effective option, dramatically reducing the number of labels that had to be manually drawn. The tool allowed for labels to be drawn at five-second intervals, with the remainder drawn automatically. Another benefit of MATLAB, was the ability to pair sonar recordings with optical videos, displaying them side by side. This made the manual classification of objects in the sonar data far easier. Interpreting sonar data requires experience and is not trivial, and many objects in the data set would have remained unclassified

without the assistance of the optical videos. In videos with a high rate of change, bounding boxes would be drawn every second to ensure accuracy. The labelling process on MATLAB can be seen in figure 17

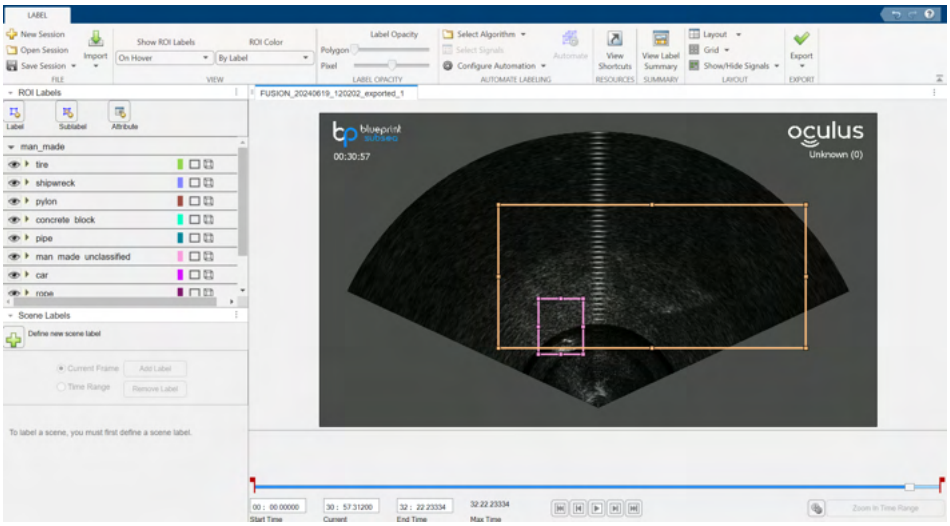


Figure 17: MATLAB labeling process.

Unfortunately, there were still several issues with the labelling workflow, including MATLAB's frequent freezing and crashing when working with video files longer than 10 minutes. Despite this, the need to make progress with the project made searching for an alternative method an undesirable option. Should a future project require the labeling of data it is recommended to search for an alternative technique. One possibility is adapting the 2012 paper, Efficient video annotation with visual interpolation and frame selection guidance [77], for use with Python and OpenCV. The paper details a method for visual interpolation, using feature extraction, and a method for selecting the optimal frames for manual labeling. The Authors reported a 50% reduction in labelling times compared to commonly used linear interpolation methods. If this paper could be implemented with OpenCV, many of the issues encountered in this project may be resolved.

3.5.5 Observations on Sonar Data

During the labelling process, 91,832 sonar images were labelled, and a number of observations on the peculiarities of sonar were made.

Automated Gain There are several variables that affect the strength of a sonar return, including distance, temperature, and the material of an object. If the returns were not normalized sonar images would have considerable variations in intensities. As a result sonar systems, including the Oculus M750d, apply a variable gain to returns which increases with distance. The functions used for this are not infallible, and sonar images occasionally become over or under-saturated. This often occurs over a distance range in the image, resulting in an arc of pixels that are too dark or bright. The issue generally persists for a few seconds, however it could cause issues with machine learning.

Lack of Height Information Forward-facing sonar has no height information about objects. This can lead to incorrect classification of objects when they are positioned above and below each other, or when viewing an object from underneath. An example from this project was the miss classification of a dock as the sea bed. The Oculus sonar spreads vertically at 20 degrees, and in this situation returns from the floating dock and the sea bed were overlapping. Due to marine growth, the dock appeared to be part of the sea bed and was initially left unlabeled.

Operating Distance and Object Scale When collecting data for machine learning care should be taken to keep the sonar operating range constant, as it affects the scale of objects. When operating at longer ranges the same object occupies a smaller space in the sonar image. Objects that do not have a constant scale negatively affect a neural network's ability to converge.

Lack of View Invariance Sonar is not view-invariant, meaning it is not possible to classify an object from every angle. This is in contrast to optical images of common objects such as bikes or dogs, which can be classified from every viewing angle. This issue was encountered multiple times during labelling. The solution was to classify the object in a separate frame, and then visually follow the object across frames. Despite the object no longer being recognizable, the previous label could be assigned to it.

3.6 Model Training

PyTorch was chosen as the machine learning library for the project. It is widely considered to be user-friendly, highly flexible, and is commonly used in the machine learning community. The PyTorch forum is active, with users regularly providing support. Before training on sonar data, the PyTorch tutorials were completed, providing an understanding of how PyTorch operates and its coding best practices. The process of designing and training models improved considerably during the project, with the PyTorch Forum proving invaluable for solving specific issues. Figure 18 is a representation of the model training process which is further discussed below.

3.6.1 DataLoader Design

The PyTorch-provided Dataset class is used to fetch data and labels, which are then passed to a DataLoader class. The method for achieving this is written in the __getitem__ method of the data set. PyTorch provides the Dataset class for increased modularity, readability, and ease of use. It creates an iterable that is used to access data together with its corresponding labels, simplifying the possession of data.

The DataLoader class contains methods for randomly iterating through Dataset classes. It creates batches of data for the training loop and uses multiprocessing to speed up the data-fetching process.

Since the available data consists of both images and videos, two custom Dataset classes were required to manage the different data types.

The first class, SonarImages, handles still images from the Shipwreck Dataset. The images are organized into folders, with each folder containing a CSV file holding labels for every image in the folder. A SonarImages object is created for each folder and appended to a list. When the __getitem__

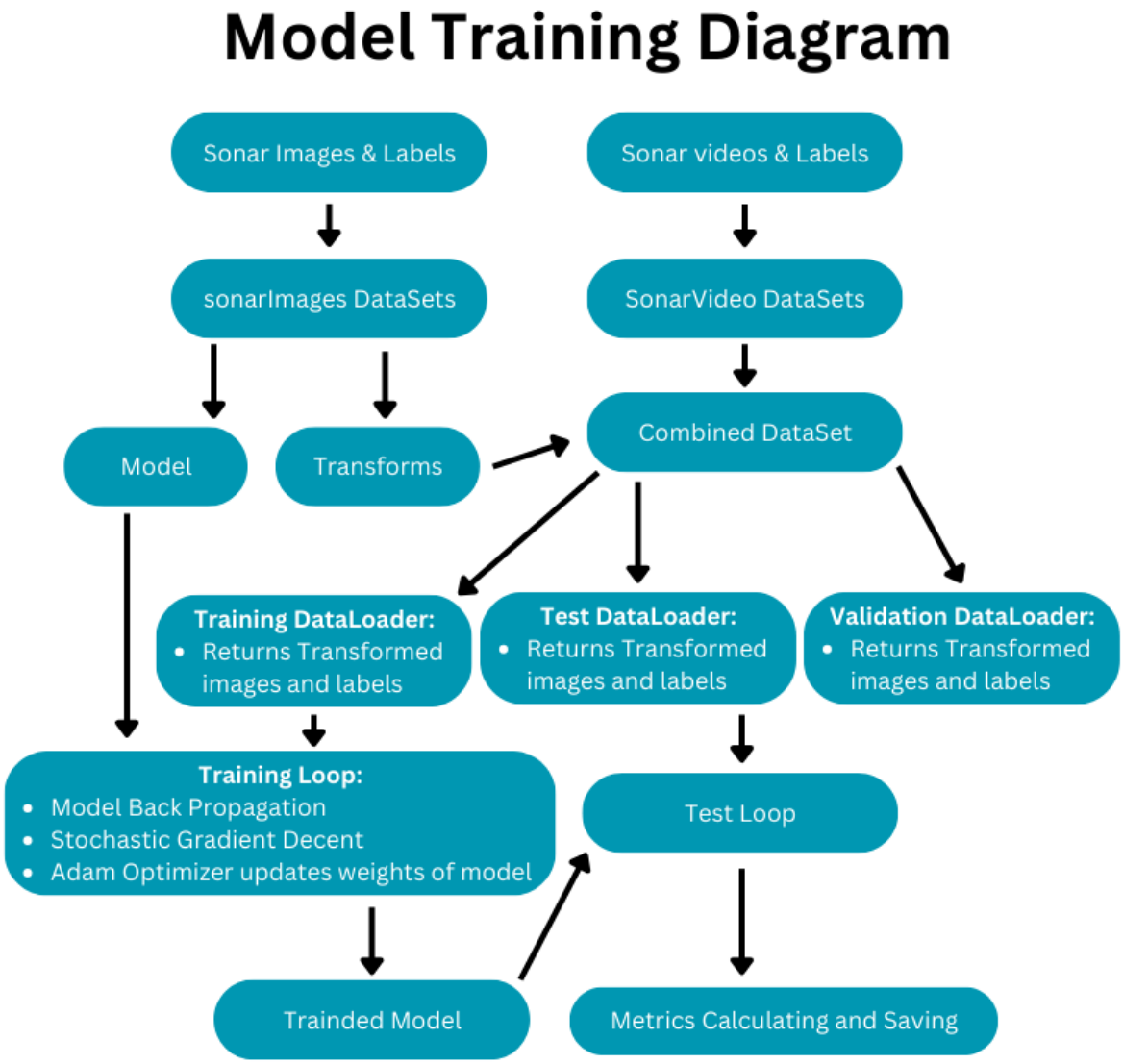


Figure 18: Flow chart of model training process.

method of a SonarImages object is called, it returns the desired image along with a list of flags representing each class in the dataset. If a flag is TRUE, the corresponding class is present in the image.

The second class, SonarVideo, inherits from the PyTorch Dataset class and fetches video data along with the associated labels. Each video is stored in a separate folder with a corresponding CSV file containing labels for each video frame. The __getitem__ method pairs the desired frames with the correct rows in the CSV file and returns a series of flags and images in the same format as SonarImages. A separate SonarVideo object is created for each video folder and added to a list.

SonarImages and SonarVideo apply specific transformations to images before returning them for model inference. These include ensuring the images are grayscale, applying downscaling, and normalizing. Additionally, images are randomly horizontally flipped, and Gaussian noise is applied to increase dataset variability. As discussed in Section 2.4.1, increased data variability helps models generalize. The Dataset objects also pass the data to the GPU for efficient training.

A third class, CombinedSonarDataset, which also inherits from the PyTorch Dataset class, takes the lists of SonarImages and SonarVideo objects to create a single dataset for model training. Originally, the datasets were accessed independently using for-loops, however, this did not enable random data selection or PyTorch's multi-threading capabilities. Merging these datasets into a single CombinedSonarDataset resolved these issues. The dataset is split into a training set (70%), a test set (25%), and a validation set (5%), which are all provided as parameters to DataLoader objects, for use in training, testing, and validation.

Due to the complexity of machine learning and uncertainty in outcomes, simplifications were made where possible to increase the likelihood of training a successful model. This included using whole-frame classification, instead of bounding boxes, using the flag system outlined above.

3.6.2 Training Loop

The training loop uses the Training DataLoader to fetch batches of data, sampling randomly from the entire data set. For models requiring sequences of consecutive images, such as those with a long short-term memory (LSTM) layer, each random frame would be returned with a specified number of previous frames. A binary cross-entropy (BCE) loss function and an Adam optimizer are used to update model weights, with backpropagation performed by PyTorch. The training loop processes every available image before exiting.

3.6.3 Testing

The test loop runs at the end of every epoch, using the data in the test DataLoader. This data has not been used in the model's training process, allowing for an indication of potential underfitting or overfitting. If the model is underfitting, the loss values for the test set will be similar to those of the training set. However, if the test loss is fifteen per cent higher than the training loss the model could be overfitting. The test loop's performance is used to adjust hyperparameters and assess the impact of any changes.

3.6.4 Validation

Since the test set is used to tune hyperparameters, the model may over-fit to it. This is despite the test data never being used in backpropagation. Decisions such as learning rate, batch size, and model architecture are influenced by test set performance. Therefore, once a model achieves the desired performance on the test set, it should be evaluated on new unseen data. This ensures it has generalized to the task effectively, instead of being optimized for the test set. The validation dataset is reserved for this step, and should not be used for any training or hyperparameter decisions.

3.6.5 Metrics

At the end of each epoch, a series of metrics are recorded to evaluate and compare the effects of model modifications or to assess the relative performance of different models. The following metrics are tracked during training:

Loss The loss measures how well a model has converged, with lower values indicating better performance. The value is calculated using the same loss function as in the training loop, which in this project is BCE loss. This loss function measures the difference between the probabilities predicted by the model and the binary ground truth labels. Equation 2 shows the general formula for calculating BCE loss:

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)] \tag{2}$$

- N: The total number of samples in the dataset.
- y_i: The TRUE label for the i-th sample, where $y_i \in \{0, 1\}$.
- $\hat{y}_i$: The probability that the i-th sample belongs to class 1.

Accuracy Accuracy is used to evaluate how well the model correctly predicts the target classes, calculated as the ratio of correct predictions to the total number of predictions. Higher accuracy values indicate better model performance on the dataset. The formula for accuracy is given in Equation 3:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(\hat{y}_i = y_i) \tag{3}$$

- N: The total number of samples in the dataset.
- y_i: The TRUE label for the i-th sample.
- $\hat{y}_i$: The predicted label for the i-th sample, based on a threshold (e.g., 0.5 for binary classification).
- $\mathbb{1}(\hat{y}_i = y_i)$: The indicator function, equal to 1 if the predicted label $\hat{y}_i$ matches the TRUE label y_i, and 0 otherwise.

Hamming Loss Hamming Loss is a metric that calculates the fraction of incorrect labels, useful in multi-label classification tasks. Lower values indicate better performance. Equation 4 provides the formula for Hamming Loss:

$$\text{Hamming Loss} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{L}\sum_{j=1}^{L}\mathbb{1}(\hat{y}_{ij} \neq y_{ij}) \tag{4}$$

- N: The total number of samples.
- L: The number of labels per sample.
- y_{ij}: The TRUE label for the j-th label of the i-th sample.
- $\hat{y}_{ij}$: The predicted label for the j-th label of the i-th sample.
- $\mathbb{1}(\hat{y}_{ij} \neq y_{ij})$: The indicator function, equal to 1 if $\hat{y}_{ij} \neq y_{ij}$, and 0 otherwise.

Precision Precision measures the accuracy of positive predictions, indicating how many predicted positives were correct. Higher values indicate fewer false positives. Equation 5 defines precision:

$$\text{Precision} = \frac{\text{TRUE Positives}}{\text{TRUE Positives} + \text{FALSE Positives}} \tag{5}$$

Recall Recall measures the proportion of true positives identified by the model. Higher values indicate fewer false negatives. Equation 6 provides the formula for recall:

$$\text{Recall} = \frac{\text{TRUE Positives}}{\text{TRUE Positives} + \text{FALSE Negatives}} \tag{6}$$

F1-Score The F1-Score is the harmonic mean of precision and recall. It is useful when the data set is imbalanced. Equation 7 shows the formula for F1-Score:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

3.6.6 CNN Models

The first attempts at training models were designed as binary classifiers, aiming to detect the presence of man-made objects in each frame. A frame containing any object class was classified as "man-made". Otherwise, the frame was classified as "not man-made". The models used for the binary man-made predictions consisted of convolutional blocks. These blocks were created by calling a function named CNNBlock, which took input channels, output channels, kernel size, stride, and padding as parameters. Each block created a convolutional layer, followed by a batch normalization layer, a ReLU activation function, and a dropout layer. Different combinations of the CNNBlock and max pooling layers were tested, including;

- 1 Layer CNN model
- 4 Layer CNN model

- 5 Layer CNN model
- 18 Layer CNN model

Batch normalization has been shown to increase the learning rate and stability of networks [78], and was included in this model for these benefits. The ReLU activation function [79] was chosen due to its widespread adoption and success across various applications. It helps models learn complex relationships by introducing non-linearity, reduces the chances of vanishing gradients, and requires minimal computational resources. Dropout [80] is employed to prevent overfitting by randomly thinning the model during training, effectively zeroing out a subset of weights. A Diagram of the CNNBlock can be seen in Figure 19

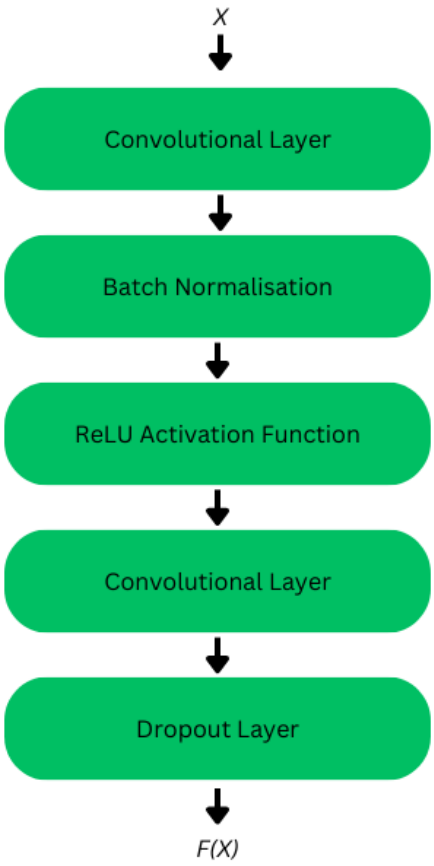


Figure 19: Diagram of the CNN block used to create the CNN models.

3.6.7 ResNet

Pre-trained ResNet Models are available to download from PyTorch [81], and can be retrained for a new purpose. The models have been trained on the ImageNet database [54], and the feature extraction

layers can be repurposed by retraining the final layer of the model. Both the eighteen and thirty-four-layer models were retrained with the sonar data.

3.6.8 ResNet with time information

From experience labelling sonar data and sonar's non-view invariance, a hypothesis that models would perform better with a memory of prior frames was formed. It was thought that if an object could be classified in an earlier frame the model should be able to track the object and continue to classify it. To test the hypothesis three different models were designed. These models used a modified DataSet classe, which would return a series of consecutive frames each time the __getitem__ method was called.

3D CNN This method feeds a set of consecutive frames to the model for each inference. The number of frames was decided before training, along with the rest of the hyperparameters. The model uses the same structure as the CNN blocks described above, however, the layers have an additional dimension to handle multiple frames.

Recurrent ResNet This model uses the pre-trained ResNet models from PyTorch and adds the PyTorch Recurrent layer to the end of the feature extraction layers.

LSTM ResNet Similar to the Recurrent ResNet, the LSTM layer provided by PyTorch was attached to the end of the feature extraction layers of a ResNet model.

3.7 Training Optimization

After training the first model on sonar data, the importance of training optimization became clear. Simple models take up to four days to complete fifty training epochs. To successfully train an effective model many different model architectures and hyperparameter settings would have to be tested, and any techniques to shorten training time would be helpful. Further research on the topic revealed several techniques that could be implemented.

Asynchronous Data Loading By default, the PyTorch DataLoader loads data into GPU memory synchronously, causing inefficiencies when the process is forced to wait for data. The 'num_workers' parameter can be modified to specify how many CPU cores the DataLoader can use, enabling asynchronous data loading. To further speed up data transfer to the GPU device the 'pin_memory' flag was set to 'TRUE', reducing the number of data transfers within the CPU.

Disable Bias in Convolutional Layers PyTorch convolutional layers apply a bias after the convolution has been calculated, however, this is also applied by the batch normalization layer. Since the CNN blocks used in the project have a batch normalization layer, the convolution bias flag can be set to FALSE. This reduces the number of computations made during training and inference.

Mixed Precision Enabling mixed precision in the GPU can cause calculations to be made up to three times faster than other GPU architectures [82]. It allows variables to be stored using fewer bits, reducing computational costs during backpropagation calculations.

Batch Size Transferring data from CPU to GPU has significant overheads, and reducing the number of transfers reduces the training time. Larger batch sizes achieve this, however, available GPU memory limits the maximum batch size. Training larger and more complex models reduces the maximum batch size. The GPU used during the project had 8Gb of available memory, and during the training of the LSTM model, the maximum batch size was one. Simpler models could train with batch sizes of 40 to 50.

3.8 Creating Public Dataset

After obtaining permission from Reach Robotics, the data collected during the project was uploaded to the Zenodo repository, under the title Paired Sonar / Optical Video Dataset [1]. Zenodo is run by CERN, with funding provided by the European Commission. Its servers are located in Europed, with uploads stored for the life of the organization. This dataset is licensed under the Creative Commons CC BY-NC-SA license. Reach Robotics retains ownership and rights to the data, and can accept or deny access requests on Zenodo. The data set is hoped to be used for further research on computer vision on sonar images.

The code used during the project was made public on GitHub in case a future project finds it useful [2].

4 Results and discussion

4.1 Gathering and Representing Results

Throughout the project many lessons were learned and implemented, resulting in initial work not being completed to the same quality as work completed later on. As a result early models only recorded, accuracy, precision, recall, and loss. The metrics were measured with the test set to mitigate the effects of overfitting, and stored in a CSV file to enable the graphical representation of the data. The model weights at the end of every epoch were saved, allowing for older models to be loaded if the newest models began to overfit.

4.1.1 Inferencing Models

To use a trained model for inference, the model class which contains its architecture, and the model's saved weights are loaded into the GPU. The model can then be fed data, with the predicted classes and the models' confidence returned. Inferencing was conducted using the validation data set, and an example can be seen in Figures 20, and 21. Figure 20 contains the sonar image fed to the model overlaid with the results. In this case, the model correctly identified the pylon in the top right of the

[1] DOI: 10.5281/zenodo.14087658
[2] https://github.com/theunistudent/Sonar-Object-Detection-Using-Machine-Learning

image, and the dock appeared as a weaker noisy return in the centre. However, it fails to detect the object closest to the sonar, which was part of the "Man-Made Unclassified" class. Figure 21 shows the scene recorded from the optical camera onboard the ROV. The Pylon can be seen on the right of the image, however, the dock is not in the frame. Due to the vertical and horizontal spread of the sonar, objects present in the sonar return were often out of frame in the optical images. There is also a large amount of noise present in the sonar image.

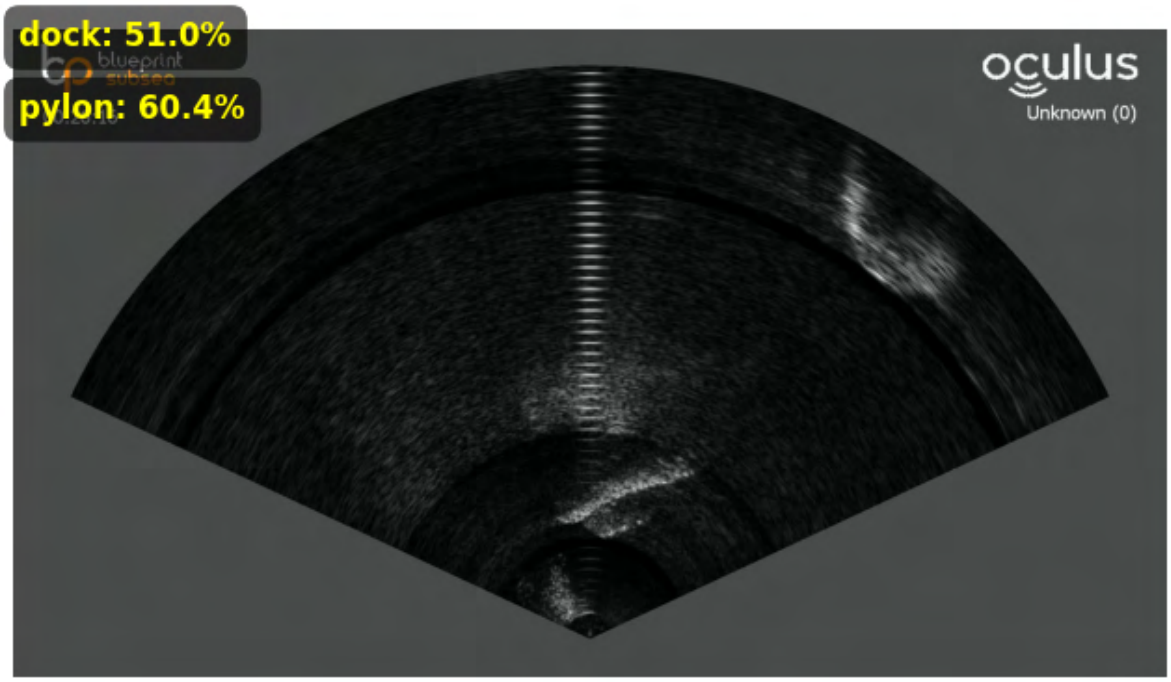


Figure 20: Model inference example.

4.2 Low Confidence Values During Inference

The class flags would be returned as TRUE if the model had a confidence value greater than 50%, however, as shown in figure 20, confidence values for TRUE positives were consistently low across models. Positive class predictions had average confidence around 60%, with confidence rarely 90%. There are several probable causes, however a known issue is the imbalance in the data set. Table 2 contains the total number of labels for each class in the data set, and the imbalance between different classes is quite pronounced. Class imbalances complicate the ability of a model to converge and its ability to learn generalized feature extraction layers. This leads to low confidence in predictions.

The models may also be underfitting due to difficulties extracting features from raw sonar data. If this is a major factor causing low confidence values, increasing the data set size and model depth would improve performance. Deeper models improve the ability to learn complex features, however, they require larger training data sets and are more difficult to train. Pre-processing, such as cropping, could also extract specific features. The models would be fed the cropped images during training, simplifying the data complexity. This method has been used with success in the literature [83], however, it can

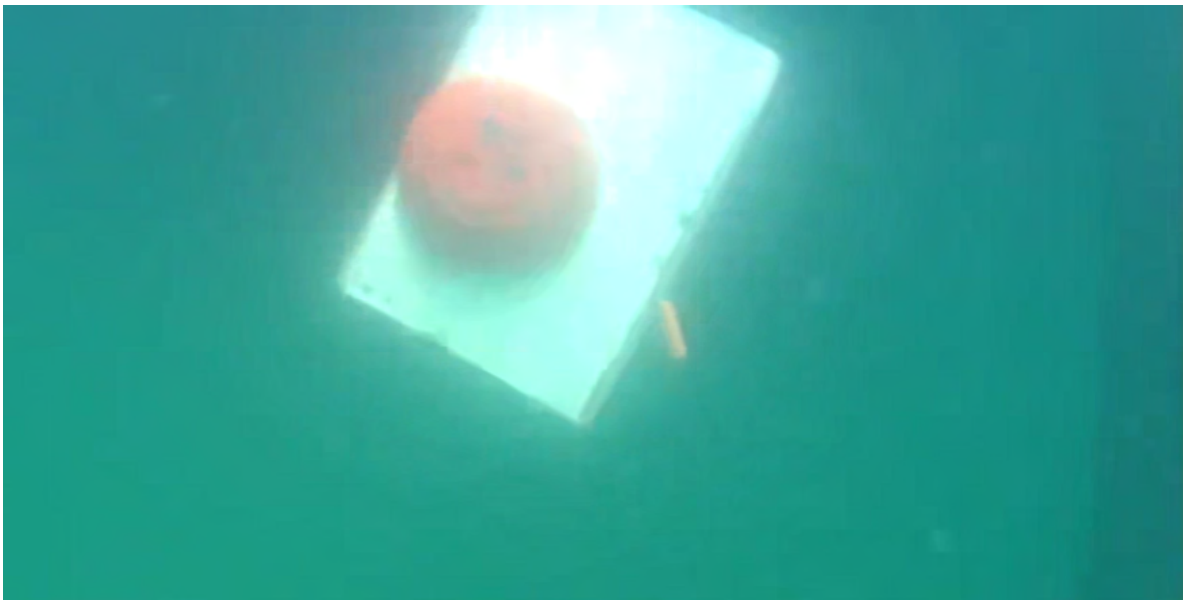


Figure 21: RGB image of the scenario in figure 20.

make inference in real-time challenging. The visual similarity of sonar data may make underfitting a difficult issue to overcome (See Appendix: Section 5.2).

4.3 Man-Made Binary Classification

The initial models were trained as binary classifiers, detecting man-made objects. All the classes were treated as a single man-made class, represented by a single binary flag. These models were created using the CNN blocks described in section 19, with a variety of configurations.

4.3.1 Single-Block Model

The first model trained on sonar data used a single CNN block and was created to verify the training process. Its results are displayed in figure 22, with loss displayed in red and accuracy, precision, and recall displayed in blue, green and purple, respectively. By the fourth epoch, the model had completely converged and the loss oscillated for the remainder of the training. Behaviour like this was expected due to the single-layers' inability to learn complex feature extraction. The figure reveals the lack of data diversity in the single-class binary classifier approach. Since only 12% of the images belong to the not man-made class, the model is incentivized to default to predicting the existence of man-made objects. This results in a lack of false negative predictions and the recall metric reaching 100%. Precision and accuracy become equal once this occurs.

4.3.2 Three-Block Model

Moving to an architecture with three CNN blocks produced a dramatic improvement over the single block. The models' metrics show consistent convergence, as seen in figure 23. The single-block and three-block models shared the same hyperparameters, indicating the improvements were due to the

Table 2: Number of class labels in dataset

Class Name	Instances
Tire	5 687
Shipwreck	7 540
Pylon	58 267
ROV	159
Rope	65
Concrete Block	58 923
Pipe	762
Man Made Unclassified	59 218
Car	234
Propeller	19 003
Ship Hull	25 025
Rudder	25 910
Dock	58 267
Total Labels	319 060
Images With No Labels	10 570
Total Sonar Images	91 832

architecture change. The greater number of learnable parameters in the three-block model likely allowed for more complex features to be extracted, improving model performance. The effects of the class imbalance remain evident in the behaviour of the recall metric. It starts close to 100% with the models' outputs defaulting to man-made. Over time the model begins to correct this behavior, resulting in a fall in the recall metric as more false negatives are produced.

4.3.3 Four-Block Model

The addition of a Fourth CNN block to the model architecture produced very similar results to the three-block model. When comparing figure 24 with figure 23, it can be seen that the four-block model has slower convergence. After 87 epochs it reaches similar accuracy to the three-block model after 70 epochs. This behaviour was expected due to the increased number of learnable parameters. The model continues to suffer from the class imbalance with the recall metric displaying similar behavior to the single-block and three-block models.

The comparison of figure 23, and figure 25 demonstrates the ability of the four-block model to converge further with a greater number of training epochs. The simpler architecture of the single-block and three-block models inhibits their ability to learn the more complex features. Although it is evident that the four-block model is approaching an asymptote, as all metrics have slowed their rate of change by the 127th epoch.

4.3.4 Eighteen-Block Model

The eighteen-block model failed to learn, with figure 26 showing no convergence trend in the metrics. It is difficult to diagnose the reason for the failure, however it may be caused by the exploding and vanishing gradient problems. When traditional CNN models become deeper, problems can arise when calculating backpropagation during training. The gradients can become overly large, overly amplifying

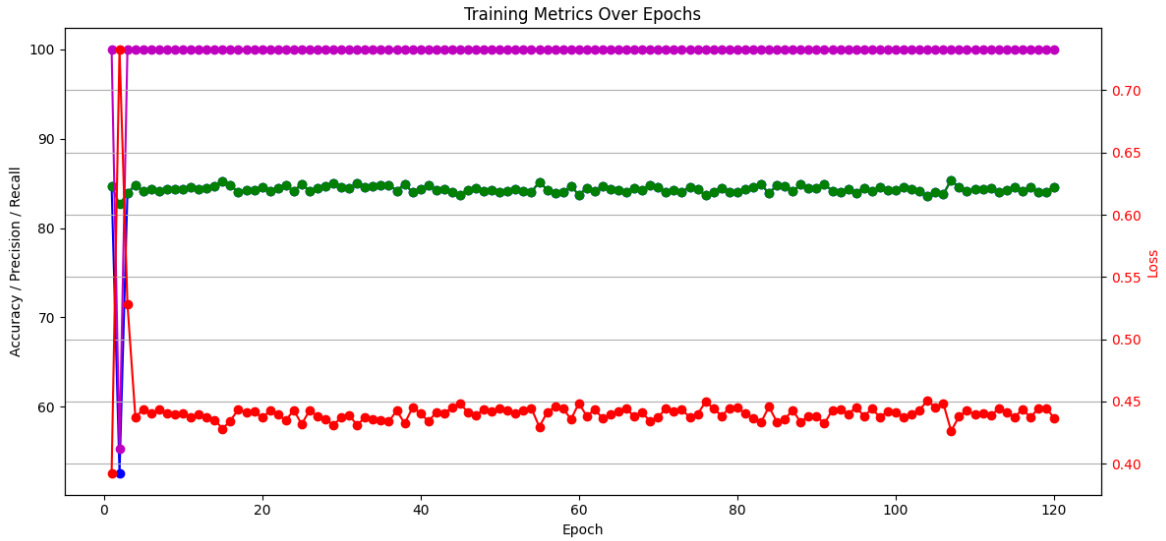


Figure 22: Single CNN Block binary classifier, for detection of man-made objects.

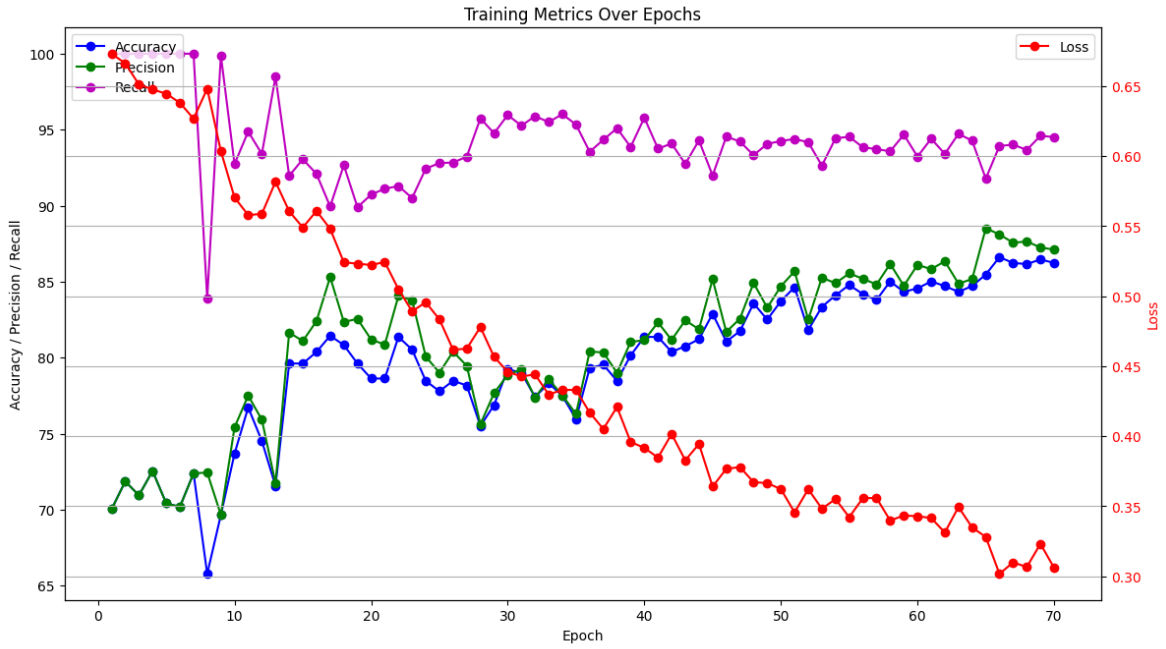


Figure 23: Three CNN Block binary classifier, for detection of man-made objects.

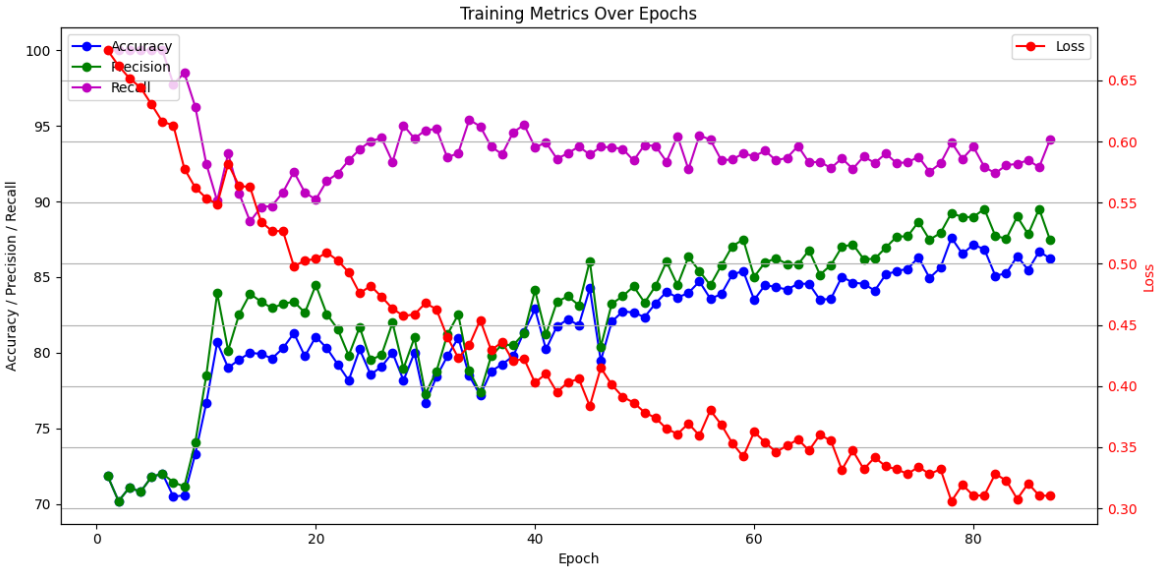


Figure 24: Four CNN Block binary classifier, for detection of man-made objects (87 Epochs).

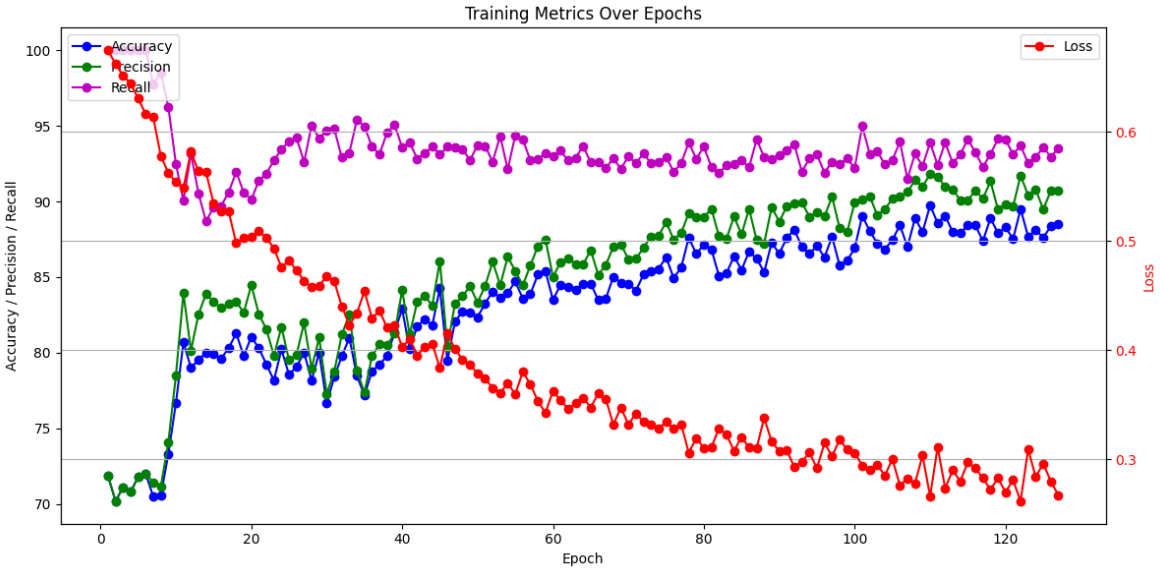


Figure 25: Four CNN Block binary classifier, for detection of man-made objects (127 Epochs).

a node, or go to zero and stop learning [67]. Both issues cause models to fail to converge. Another potential issue may be the choice of hyperparameters, such as the learning rate. If the learning rate is set too high a model will change its weights to quickly, failing to learn and causing the loss function to oscillate. Ideally, a grid search technique would be implemented to optimist the hyperparameters, however project time constraints prevented this. Training deeper models took significant time to optimise, with the eighteen-block model training for ten days.

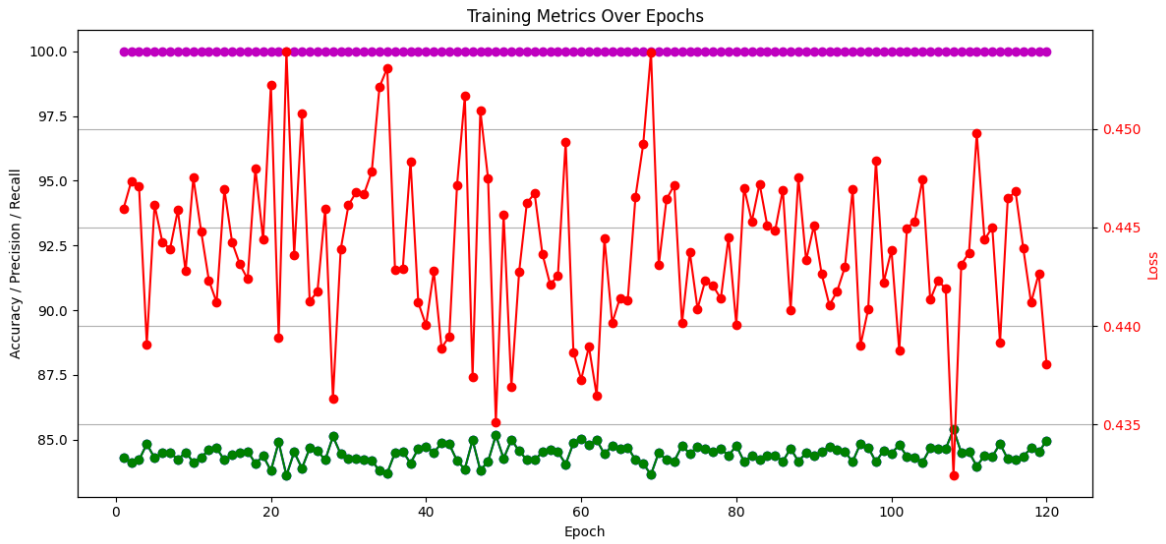


Figure 26: Eighteen CNN Block binary classifier, for detection of man-made objects (120 Epochs).

4.4 Multi-Class Binary Classification using CNNs

Unfortunately, the single-class binary classifiers never reached the desired performance levels. The relatively simple classification task was expected to reach an accuracy of above 90%, however several factors prevented this. The most significant issue was likely the class imbalance and a new approach was needed to address this issue. It was decided to switch to a multi-class binary classifier, using the flag system discussed in Section 3.6.1. The model would be trained with a subset of classes to more evenly balance the data set, and attempt to detect which classes were present in a given sonar image. The first attempt used a five CNN block design and was trained to detect pylons, rudders, and tyres. The subset of classes was used to simplify the feature learning process, hopefully producing a model with better performance.

Another method for addressing the class imbalance involved applying a gain to the loss function when false negative predictions occurred. Since the data was prominently of images without any of the three classes, the gain was used to punish the model when the existence of a class was missed. Initially, a gain of three for each class was used, however these could be changed as required.

Figure 27 shows the five CNN block architecture failed to converge when training on the multi-class problem. The Hamming loss and the training loss function show a general trend of increasing

instability, and the model shows no signs of successful training. The increased complexity of the task is the most likely issue, and the architecture was insufficient for learning the required features.

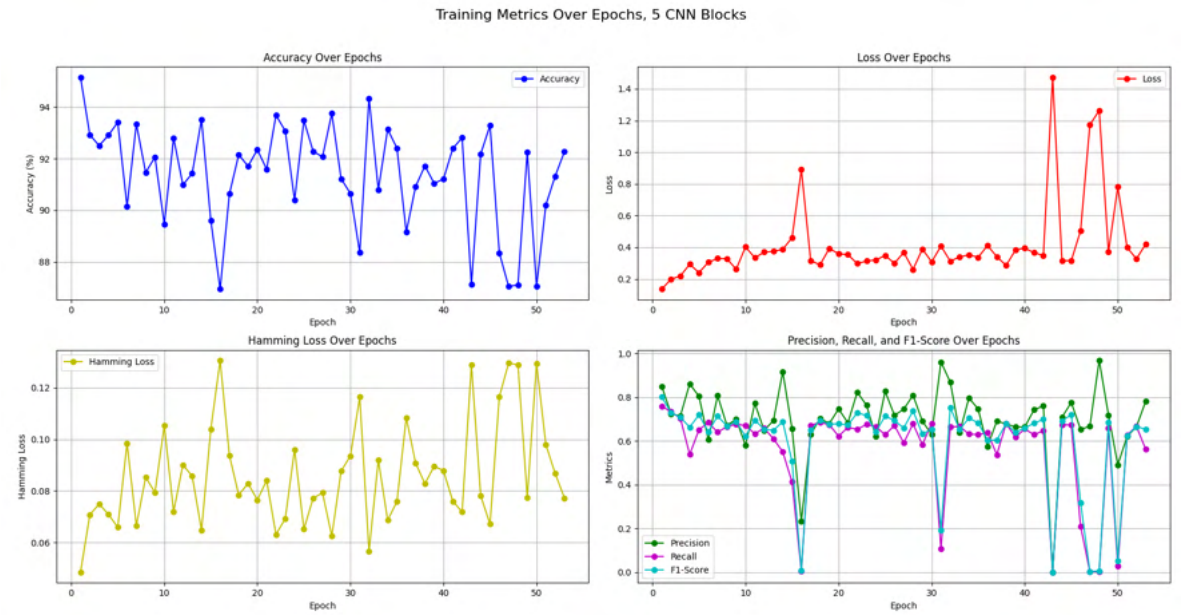


Figure 27: Five CNN Block, multi-Class binary classifier.

4.5 Multi-Class Binary Classification using ResNet

The failure of the five-block model to converge on the multi-class task indicated a need to use a deeper architecture, however, the previous failure of the eighteen-block model (Figure 4.3.4) had created a scepticism of deep CNN models. The ResNet [66] architecture provided a solution, with the ability to train deeper networks while avoiding the exploding and vanishing gradient problems. A ResNet-18 and a ResNet-34 were retrained using the method described in section 3.6.7. Both models produced better metrics than the five CNN block models, with figure 28, and figure 29 showing both models converging until the 40th epoch. At this point both models appear to experience overfitting, resulting in increased instability of the metrics.

Although the training process was more promising than the CNN block architecture, analysis of the metrics left much to be desired. The models experienced large fluctuations in precision, recall and F1-score, and the loss and accuracy metrics did not produce clean converging curves as seen in figure 25. It is highly likely that a grid search to optimise hyperparameters would improve these issues, However, time constraints make this impractical. The eighteen-layer model took seven days to train while the deeper thirty-four-layer model took 10 days.

4.6 Five-Class Binary Classification using ResNet

Following the relative levels of success in retraining the ResNet architecture it was decided to increase the quantity of classes. Figure 30 shows the results of a 34 ResNet trained to detect pylons, rudders,

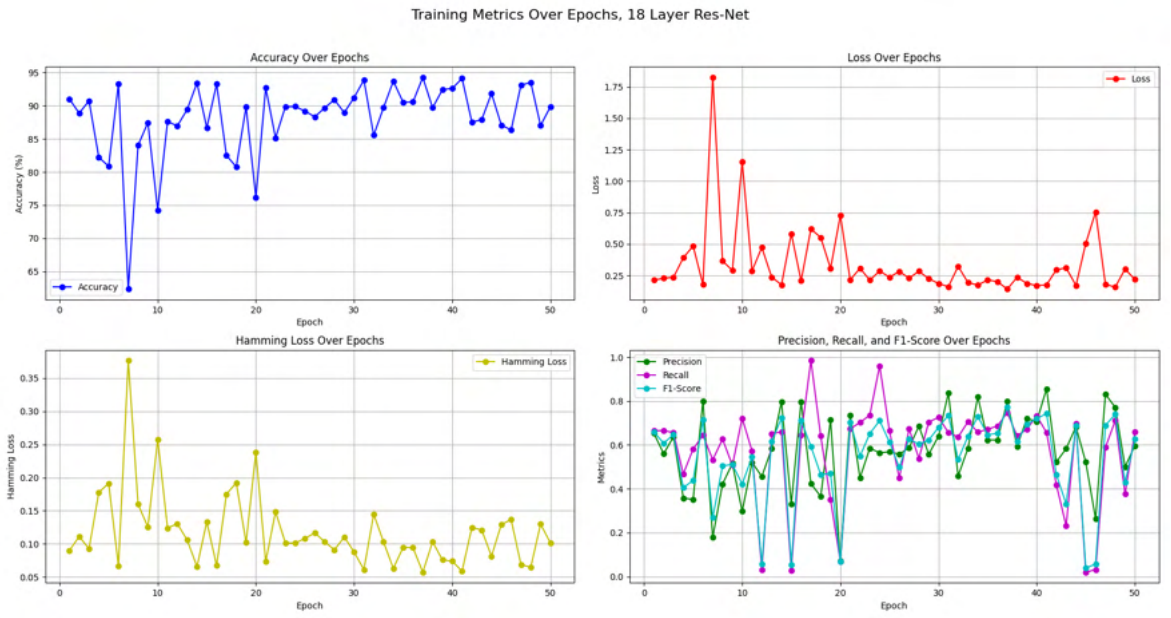


Figure 28: Eighteen layer ResNet, multi-Class binary classifier.

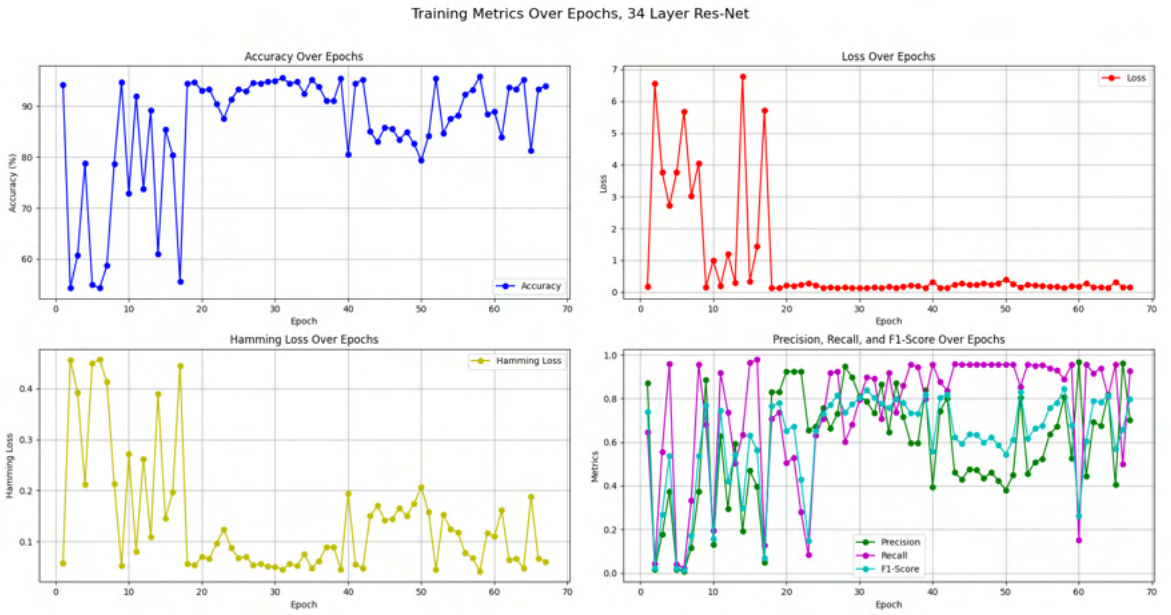


Figure 29: Thirty-four layer ResNet, multi-Class binary classifier.

tyres, concrete blocks, and docks. The model performed well after 25 epochs of training data, however, the metrics show signs of overfitting soon after. By the 35th epoch, the metrics have become increasingly unstable and show strong signs of overfitting. The instability is again most likely caused by data set imbalances, although a grid search for hyperparameter tuning would improve the stability of the metrics. While there are issues with the results, the model converges on the more complex task and proves the viability of machine learning techniques for sonar object detection.

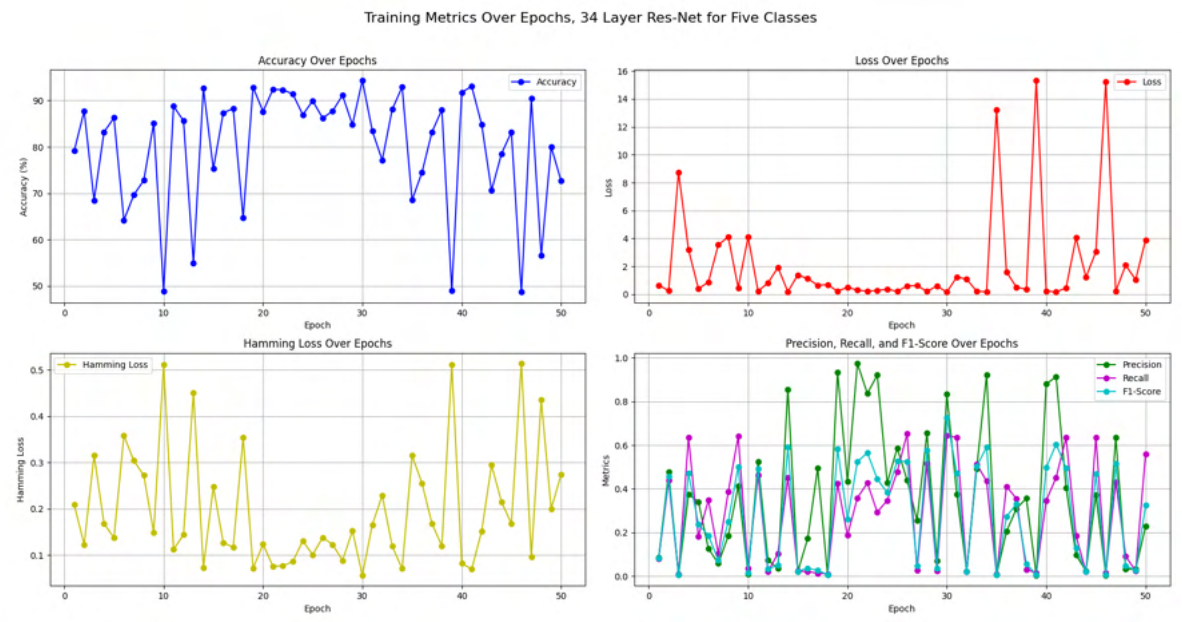


Figure 30: Thirty-four layer ResNet, multi-class binary classifier, trained on five classes.

4.7 Incorporating Historical Information

The attempt to provide historical information to the models was inspired by the non-view invariant nature of sonar images, as discussed in section 3.6.8. Three different architectures were attempted however the results were unsuccessful. Due to time constraints, each model could only be trained once, and there were no opportunities for attempting different training techniques. Previous models were able to undergo several iterations to improve training loops and solve bugs, however, the architectures with historical information were significantly more complex and time-consuming.

4.7.1 Multi-Class Binary Classification using Recurrent ResNet

The recurrent ResNet required significantly more GPU memory and could only take a batch size of one. It also increased the computational cost of back propitiation and the training of 46 epochs took 17 days. As shown in figure 31, there was no period during the training process where the model was converging. The occasional spikes in accuracy appear to be inversely related to the recall metric. Both the spikes and the relationship are unexpected, and it is unclear what the cause could be. The training data set is likely too small to train a recurrent model, however, this is not the only issue affecting the

results.

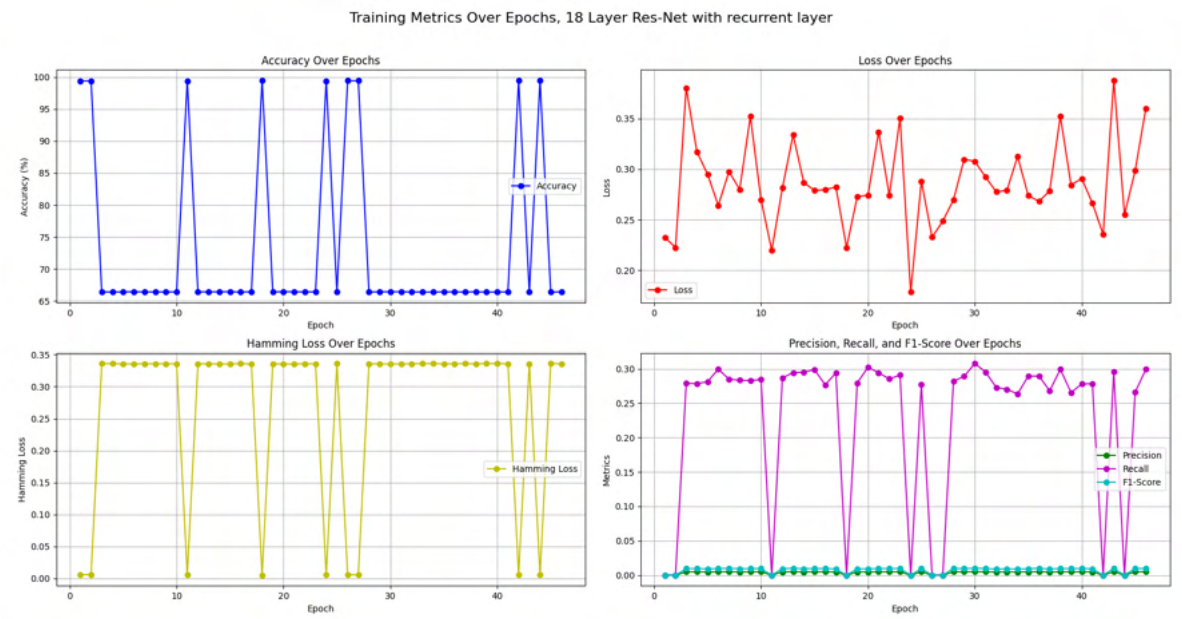


Figure 31: Recurrent, Eighteen Layer ResNet, Multi-Class Binary Classifier.

4.7.2 Multi-Class Binary Classification using ResNet with LSTM

The results of the ResNet with an LSTM layer, shown in figure 32, are very similar to the recurrent ResNet model. It produced the same spikes in accuracy, which are inversely related to the recall metric, and the metrics failed to converge. The LSTM model took 14 days of training to finish 32 epochs, using a batch size of one.

4.7.3 Multi-Class binary Classification using 3D-CNN

An attempt was made to train a 3D-CNN model, however, during the training process, an issue occurred with the computer and the data was lost. Reach Robotics runs several simulations on the machine, and a reboot caused a loss of training data.

4.8 Model Inference Examples

The following section contains inference examples from the 34-layer ResNet described in Section 4.5 and Section 4.6. Each model output is displayed with the corresponding optical image of the scene.

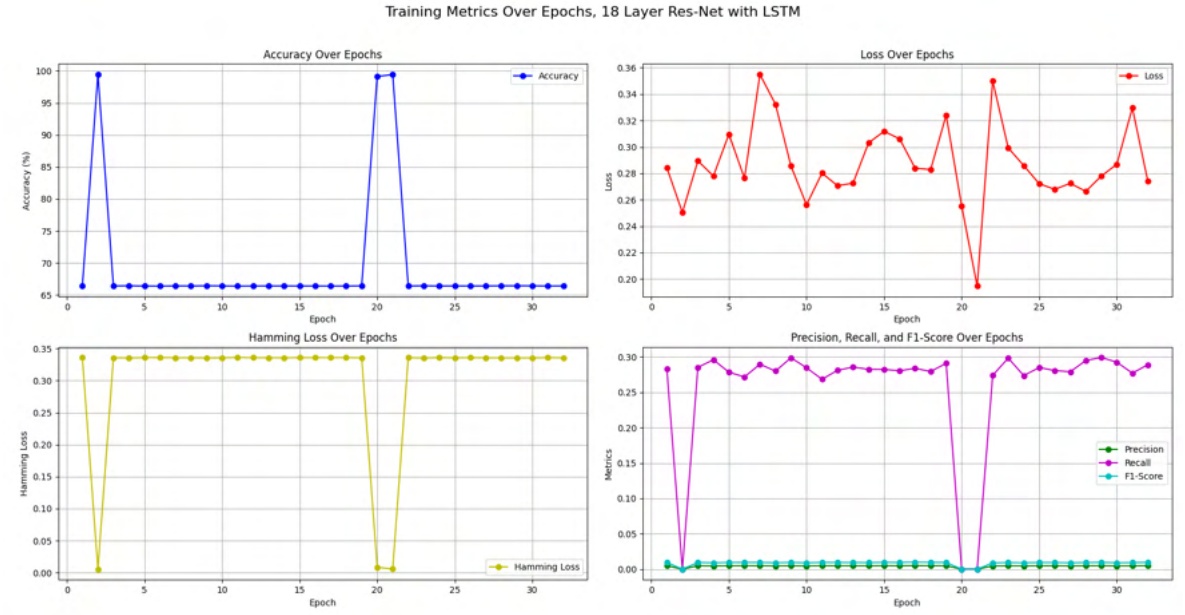


Figure 32: Eighteen layer ResNet with LSTM layer, Multi-Class binary classifier.

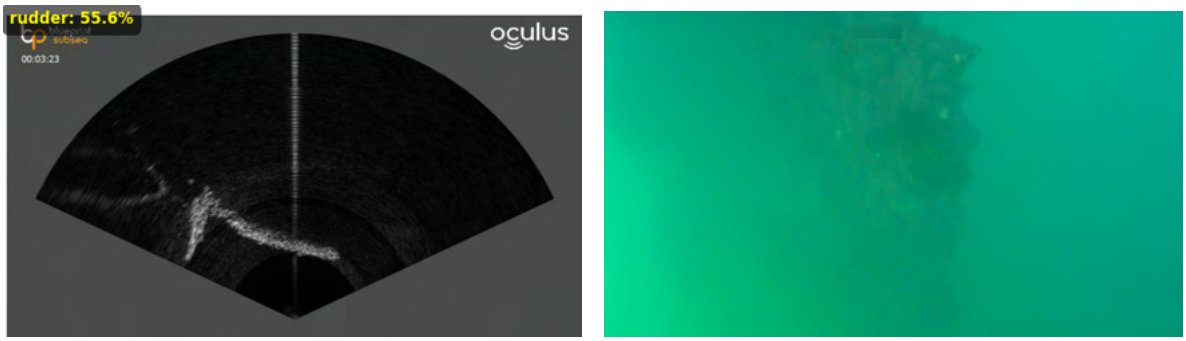


Figure 33: Rudder of a ship in Sydney Harbor

Figure 34: Pylon and dock, recorded in Sydney Harbor

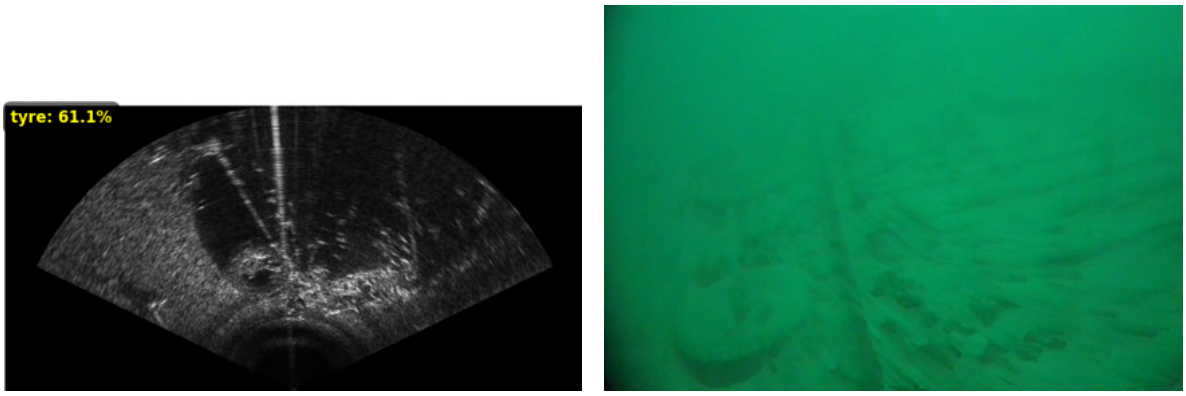


Figure 35: Tyre from the Shipwreck-Dataset [12]

5 Conclusion

Imaging sonars play an important role in underwater site inspections due to their ability to operate in low visibility. The systems are often used to gain situational awareness before deploying commercial divers, reducing risk and increasing efficiency. To effectively use current systems constant attention from a skilled operator is required. This paper attempts to demonstrate that machine-learning techniques for computer vision can be successfully applied to sonar data, reducing the cognitive load on sonar operators. The method for labelling sonar data using MATLAB dramatically increased the speed of the labelling process, however, several issues led to the recommendation for future projects to seek alternatives.

All data collected for the project was released on Zenodo and will provide resources for future computer vision projects for underwater environments [3]. The data set only contains data gathered in Sydney Harbour, providing one of the largest open-water, forward-facing sonar data sets available for research.

Several architectures for object detection were tested, the most successful being the 34-layer retrained ResNet. The model was shown to converge on a binary multi-class classification problem before showing signs of overfitting. Due to time and resource constraints on the project, the model could not be developed to a deployment-ready level of performance, however, the results demonstrate potential for this. The 34-layer ResNet in Figure 29, achieved an accuracy of 94% after 31 epochs on the test set, demonstrating good convergence to the multi-class binary classification problem.

5.1 Future Work

For the training of a model with performance levels high enough for deployment in real-world situations, several improvements can be made. A larger more diverse data set together with a hyperparameter optimization technique, such as a grid search, should significantly improve performance. Testing with deeper versions of the ResNet architecture is also recommended, however, the use of a more capable GPU may be required to train models within a reasonable time.

[3] DOI: 10.5281/zenodo.14087658

The attempt to incorporate historical information in the models was unsuccessful, however, the reason for the models' failure to converge is unknown. The author still sees promise in these techniques and would encourage further research.

Training of unsupervised models is another area of interest for future work. The author recommends training an anomaly detector with data containing no man-made objects. This could be based on the U-Net architecture [64], with the network trained to reproduce the provided images. When the model is provided with a man-made object the loss function should dramatically increase, allowing for the object's detection.

The author sees significant potential in the fusion of optical and sonar data for object detection due to their differing strengths and weaknesses. This paper strongly recommends future research focus on this area.

References

[1] Zhaorui Gu, Xiuhan Liu, Zhiqiang Hu, Guoyu Wang, Bing Zheng, John Watson, and Haiyong Zheng. Underwater computational imaging: a survey. *Intelligent Marine Technology and Systems*, 1(1):2, 2023.

[2] Blue Robotics. Ping360 scanning imaging sonar.

[3] Bay Dynamics. Blueprint subsea oculus sonar, 2024. Product page.

[4] Rusty and Nicolette and Megan. A smooth operator's guide to underwater sonars and acoustic devices.

[5] Blue Robotics. Cerulean sidescan sonar. Available online. Accessed: 17/04/2024.

[6] Rafał Kot. Review of obstacle detection systems for collision avoidance of autonomous underwater vehicles tested in a real environment. *Electronics*, 11(21), 2022.

[7] Kraken Robotics. Synthetic aperture sonar (sas). `https://www.krakenrobotics.com/products/synthetic-aperture-sonar/`. Accessed: 2024-10-15.

[8] Deep Trekker. Blueprint oculus multibeam sonars - m750d. `https://www.deeptrekker.com/shop/products/blueprint-oculus-multibeam-sonars-m750d`. Accessed: 2024-10-16.

[9] Soumadip Ghosh, Suharta Banerjee, Supantha Das, Arnab Hazra, Saurav Mallik, Zhongming Zhao, and Ayan Mukherji. Evaluation and optimization of biomedical image-based deep convolutional neural network model for covid-19 status classification. *Applied Sciences*, 12:10787, 10 2022.

[10] Christopher Olah. Understanding lstms. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. Accessed: 2024-10-16.

[11] Blueye Robotics. Oculus m750d multibeam sonar, 2024. Accessed: 2024-10-21.

[12] Nicolas Pecheux, Vincent Creuze, Frédéric Comby, and Olivier Tempier. Self calibration of a sonar–vision system for underwater vehicles: A new method and a dataset. *Sensors*, 23(3), 2023.

[13] Ke Tan, Xiaonan Xu, and Hongyu Bian. The application of ndt algorithm in sonar image processing. In *2016 IEEE/OES China Ocean Acoustics (COA)*, pages 1–4, 2016.

[14] Heather Mongilio. Murky water, twisted steel: Inside the operation to clear the key bridge wreckage. April 2024. Updated: April 14, 2024, Accessed: 2024-10-21.

[15] Haidi Zhu, Haoran Wei, Baoqing Li, Xiaobing Yuan, and Nasser Kehtarnavaz. A review of video object detection: Datasets, metrics and methods. *Applied Sciences*, 10(21), 2020.

[16] Ultralytics. Ultralytics documentation, 2024. Accessed on March 26, 2024.

[17] Seunghun Jin, Junguk Cho, Xuan Dai Pham, Kyoung Mu Lee, Sung-Kee Park, Munsang Kim, and Jae Wook Jeon. Fpga design and implementation of a real-time stereo vision system. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(1):15–26, 2010.

[18] Muskan Khan, Danny Johnson, and Jane Smith. Analysis on computer vision range finding techniques. 05 2023.

[19] Dinh Quang Huy, Nicholas Sadjoli, Abu Bakr Azam, Basman Elhadidi, Yiyu Cai, and Gerald Seet. Object perception in underwater environments: a survey on sensors and sensing methodologies. *Ocean Engineering*, 267:113202, 2023.

[20] Pawel Piskur, Piotr Szymak, Krzysztof Jaskólski, Leszek Flis, and Marek Gasiorowski. Hydroacoustic system in a biomimetic underwater vehicle to avoid collision with vessels with low-speed propellers in a controlled environment. *Sensors*, 20(4), 2020.

[21] Encyclopedia Britannica. Density of seawater and pressure. Accessed on March 27, 2024.

[22] NASA Earthdata. Air mass density, 2017. Accessed on March 27, 2024.

[23] Geoscience Australia. Sonar, 2023.

[24] Fei Yuan, Fengqi Xiao, Kaihan Zhang, Yifan Huang, and En Cheng. Noise reduction for sonar images by statistical analysis and fields of experts. *Journal of Visual Communication and Image Representation*, 74:102995, 2021.

[25] NOAA Ocean Exploration. Multibeam sonar fact sheet. Accessed: 2024-10-14.

[26] University of Connecticut. *SeaBeam Multibeam Sonar: Theory of Operation*, 2018. Accessed: 2024-10-09.

[27] Blue Robotics. Sonoptix echo. Accessed: 16/04/2024.

[28] A. Aggarwal, P. Kampmann, J. Lemburg, and F. Kirchner. Haptic object recognition in underwater and deep-sea environments. *Journal of Field Robotics*, pages 1–19, 2014.

[29] Walid Gomaa, Ashraf F. El-Sherif, and Yasser H. El-Sharkawy. Underwater laser detection system. In W. Andrew Clarkson and Ramesh K. Shori, editors, *Solid State Lasers XXIV: Technology and Devices*, volume 9342, page 934221. International Society for Optics and Photonics, SPIE, 2015.

[30] Fausto Ferreira, Diogo Machado, Gabriele Ferri, Samantha Dugelay, and John Potter. Underwater optical and acoustic imaging: A time for fusion? a brief overview of the state-of-the-art. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–6, 2016.

[31] Hong-Gi Kim, Jungmin Seo, and Soo Mee Kim. Underwater optical-sonar image fusion systems. *Sensors*, 22(21), 2022.

[32] University of Rhode Island. Reverberation.

[33] Jinsheng Xiao, Wentao Zou, Shangyue Zhang, Junfeng Lei, Wen Wang, and Yuan-Fang Wang. Video denoising algorithm based on improved dual-domain filtering and 3d block matching. *IET Image Process.*, 12(12):2250–2257, 2018.

[34] Qiegen Liu, Shanshan Wang, Leslie Ying, Xi Peng, Yanjie Zhu, and Dong Liang. Adaptive dictionary learning in sparse gradient domain for image recovery. *IEEE Transactions on Image Processing*, 22(12):4652–4663, 2013.

[35] Yifan Huang, Weixiang Li, and Fei Yuan. Speckle noise reduction in sonar image based on adaptive redundant dictionary. *Journal of Marine Science and Engineering*, 8(10), 2020.

[36] Shi Zhao. Automatic underwater multiple objects detection. Master's thesis, School of Mechanical Engineering, The University of Adelaide, 2008.

[37] Chunxi Cheng, Qixin Sha, Bo He, and Guangliang Li. Path planning and obstacle avoidance for auv: A review. *Ocean Engineering*, 235:109355, 2021.

[38] Johnny L. Chen and Jason E. Summers. Deep neural networks for learning classification features and generative models from synthetic aperture sonar big data. *Proceedings of Meetings on Acoustics*, 29(1):032001, 05 2017.

[39] Seong Ju Lee, Yong Seon Moon, Nak Yong Ko, Hyun-Taek Choi, and Jong-Moo Lee. A method for object detection using point cloud measurement in the sea environment. In *2017 IEEE Underwater Technology (UT)*, pages 1–4, 2017.

[40] Ding Fu-guang, Jiao Peng, Bian Xin-qian, and Wang Hong-jian. Auv local path planning based on virtual potential field. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 4, pages 1711–1716 Vol. 4, 2005.

[41] Mingxi Zhou, Ralf Bachmayer, and Brad Deyoung. Mapping for control in an underwater environment using a dynamic inverse-sonar model. pages 1–8, 09 2016.

[42] Ørjan Grefstad and Ingrid Schjølberg. Navigation and collision avoidance of underwater vehicles using sonar data. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–6, 2018.

[43] Christopher Morencya, Daniel J. Stilwella, and Stephen T. Krauss. Use of a low-cost forward-looking sonar for collision avoidance in small auvs, analysis and experimental results. *Elsevier*, 2023.

[44] Boris Crnokić, Snježana Rezić, and Slaven Pehar. *Comparision of Edge Detection Methods for Obstacles Detection in a Mobile Robot Environment*, pages 0235–0244. 01 2016.

[45] Michael R. Dolbec. Velocity estimation using forward looking sonar. Master's thesis, Naval Postgraduate School Monterey, California, 2007.

[46] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[47] Yannik Steiniger, Dieter Kraus, and Tobias Meisen. Survey on deep learning based computer vision for sonar imagery. *Engineering Applications of Artificial Intelligence*, 114:105157, 2022.

[48] Dhiraj Neupane and Jongwon Seok. A review on deep learning-based approaches for automatic sonar target recognition. *Electronics*, 9(11), 2020.

[49] Bowen Teng and Hongjian Zhao. Underwater target recognition methods based on the framework of deep learning: A survey. *International Journal of Advanced Robotic Systems*, 17(6):1729881420976307, 2020.

[50] A. Galusha, J. Dale, J. M. Keller, and A. Zare. Deep convolutional neural network target classification for underwater synthetic aperture sonar imagery. In Steven S. Bishop and Jason C. Isaacs, editors, *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIV*, volume 11012, page 1101205. International Society for Optics and Photonics, SPIE, 2019.

[51] Yannik Steiniger, Dieter Kraus, and Tobias Meisen. Generating synthetic sidescan sonar snippets using transfer-learning in generative adversarial networks. *Journal of Marine Science and Engineering*, 9(3), 2021.

[52] Fang Wang, Huitao Li, Kai Wang, Lichen Su, Jing Li, and Lili Zhang. An improved object detection method for underwater sonar image based on pp-yolov2. *Journal of Sensors*, 2022:Article ID 5827499, 12 pages, 2022.

[53] PyTorch. Transfer learning for computer vision tutorial. `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`, 2023. Accessed: 2024-10-14.

[54] Imagenet. `https://www.image-net.org/`, 2021. Accessed: April 13, 2024.

[55] Zhen Cheng, Guanying Huo, and Haisen Li. A multi-domain collaborative transfer learning method with multi-scale repeated attention mechanism for underwater side-scan sonar image classification. *Remote Sensing*, 14(2), 2022.

[56] Mateusz Ochal, Jose Vazquez, Yvan Petillot, and Sen Wang. A comparison of few-shot learning methods for underwater optical and sonar image classification. In *Global Oceans 2020: Singapore – U.S. Gulf Coast*, pages 1–10, 2020.

[57] Matias Valdenegro-Toro. Deep neural networks for marine debris detection in sonar images. *CoRR*, abs/1905.05241, 2019.

[58] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning, 2017.

[59] Victor Cecchi. Few-shot learning benchmarks. `https://data-ai.theodo.com/blog-technique/few-shot-learning-benchmarks`, 2022. Accessed: 2024-10-15.

[60] Xiaoming Qin, Xiaowen Luo, Ziyin Wu, and Jihong Shang. Optimizing the sediment classification of small side-scan sonar images based on deep learning. *IEEE Access*, 9:29416–29428, 2021.

[61] Nandeeka Nayak, Makoto Nara, Timmy Gambin, Zoë Wood, and Christopher M. Clark. Machine learning techniques for auv side-scan sonar data feature extraction as applied to intelligent search for underwater archaeological sites. In Genya Ishigami and Kazuya Yoshida, editors, *Field and Service Robotics*, pages 219–233, Singapore, 2021. Springer Singapore.

[62] Rixia Qin, Xiaohong Zhao, Wenbo Zhu, Qianqian Yang, Bo He, Guangliang Li, and Tianhong Yan. Multiple receptive field network (mrf-net) for autonomous underwater vehicle fishing net detection using forward-looking sonar images. *Sensors*, 21(6), 2021.

[63] Georgy V. Ayzel, Tobias Scheffer, and Maik Heistermann. Rainnet v1.0: a convolutional neural network for radar-based precipitation nowcasting. 2020.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[65] Jason Brownlee. Convolutional layers for deep learning neural networks. `https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/`, 2019. Accessed: 2024-10-16.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[67] Vardhan Vishal. The challenge of vanishing/exploding gradients in deep neural networks. `https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/`, 2021. Accessed: 2024-10-16.

[68] Anil Tilbe. Deep convolutional networks for monocular velocity estimation. `https://towardsdatascience.com/deep-convolutional-networks-for-monocular-velocity\linebreak-estimation-a0081d6bc7a9`, 2019. Accessed: 2024-10-16.

[69] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *CoRR*, abs/1912.05911, 2019.

[70] Mercury Marine. How forward-facing sonar is changing fishing. `https://www.mercurymarine.com/us/en/lifestyle/dockline/how-forward-facing-sonar-is-changing-fishing#:~:text=%E2%80%9CIt%20helps%20me%20determine%20the,few%20ways%20it%20helps%20me.`, 2023. Accessed: 2024-10-16.

[71] National Transportation Safety Board. Preliminary report dca24mm031: Marine investigation - 12 docket items. Accessed: 2024-10-21, September 2024. Date of Accident: 03/26/2024, City: Baltimore, MD, Country: United States, Project Type: Investigation.

[72] Blueprint Subsea. Oculus m-series multibeam sonar, 2024. Accessed: 2024-10-21.

[73] Terry Sejnowski and R. Gorman. Connectionist Bench (Sonar, Mines vs. Rocks). UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C5T01Q.

[74] Kaibing Xie, Jian Yang, and Kang Qiu. A dataset with multibeam forward-looking sonar for underwater object detection. *Scientific Data*, 9(1):739, December 2022.

[75] Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. Label Studio: Data labeling software, 2020-2024. Open source software available from https://github.com/HumanSignal/label-studio.

[76] User on AI Stack Exchange. Is it difficult to learn the rotated bounding box for a rotated object?, 2018. Accessed: 2024-10-23.

[77] A. Kuznetsova, A. Talati, Y. Luo, K. Simmons, and V. Ferrari. Efficient video annotation with visual interpolation and frame selection guidance, 2020.

[78] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[79] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019.

[80] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *Proceedings of the 27th International Conference on Machine Learning (ICML-14)*, pages 192–200. PMLR, 2014.

[81] PyTorch. Pytorch vision resnet, 2021. Accessed: 2024-10-25.

[82] PyTorch Team. Performance tuning guide, 2023. Accessed: 2024-10-26.

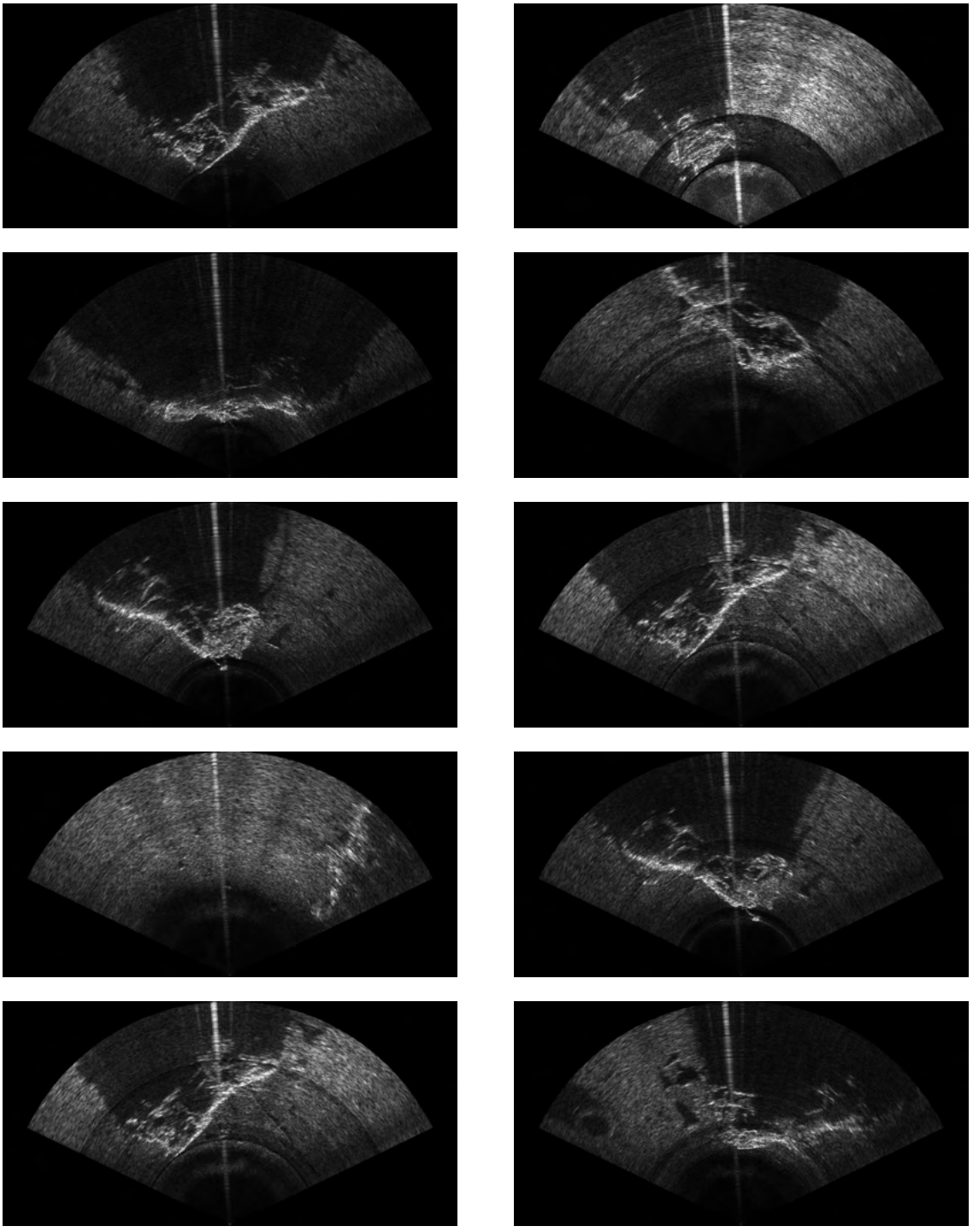
[83] Pycad. Fast medical imaging cropping, 2021. Accessed: [Your access date, e.g., October 29, 2024].

Appendix

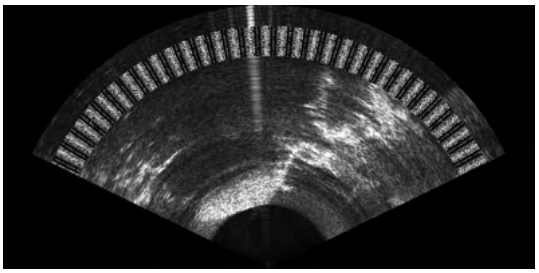
5.2 Example Sonar Images From Each Class

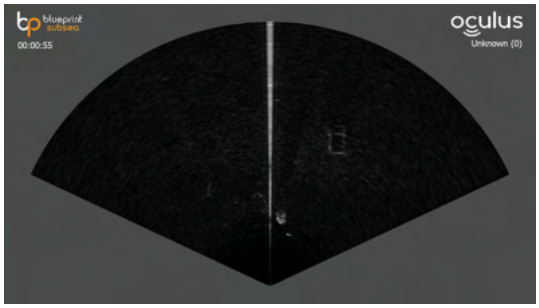
The following section contains ten images for each object class. The images may contain multiple classes. The images are sourced from the primary data collected in Sydney Harbor, and the Shipwreck-Dataset [12]

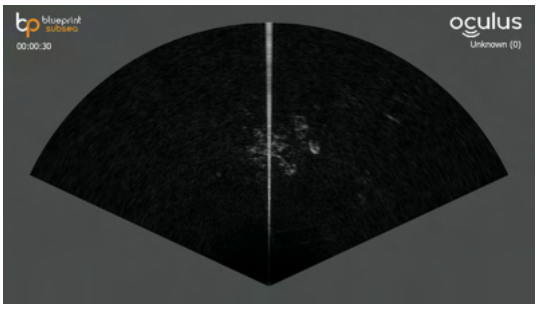
Car

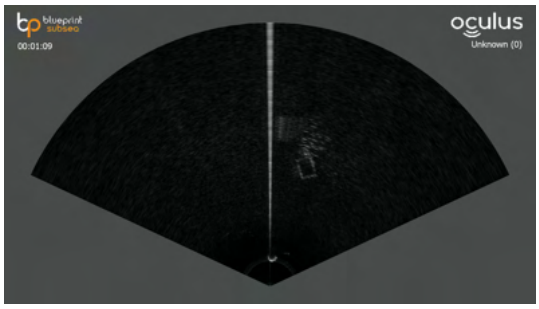


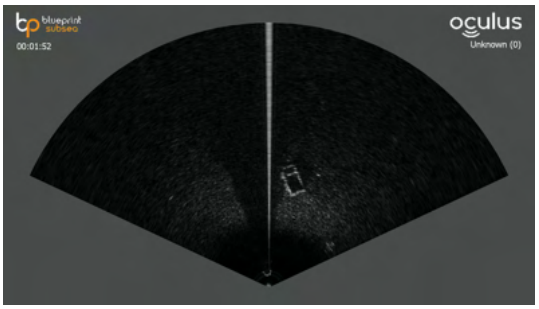
Concrete Block

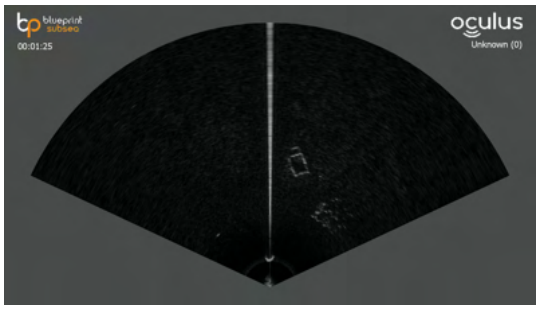


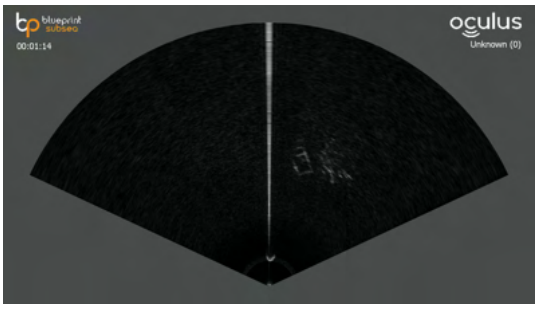


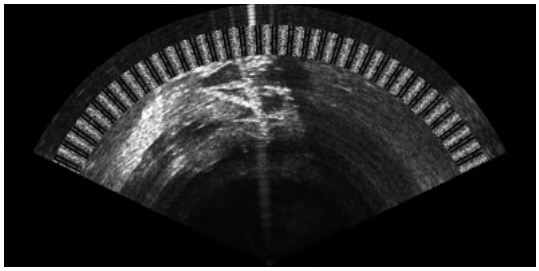


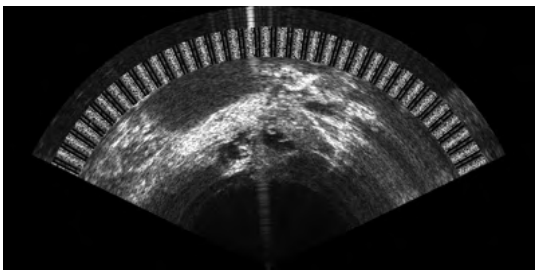


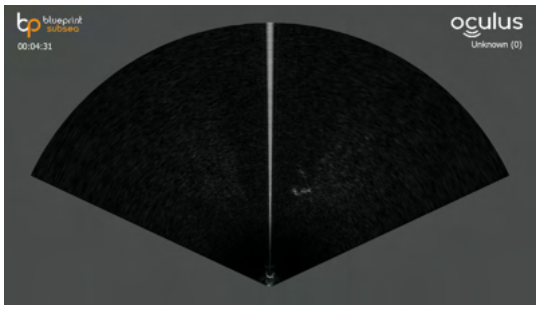




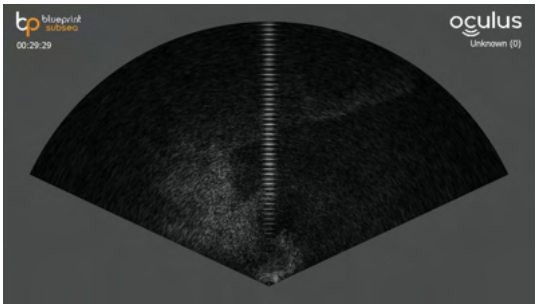


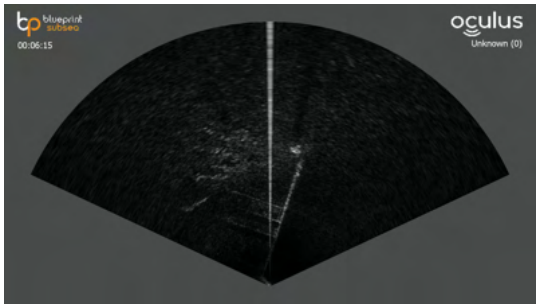


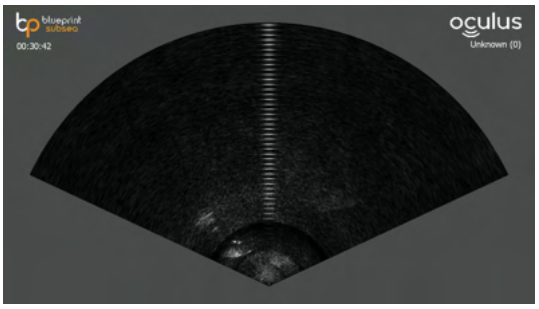


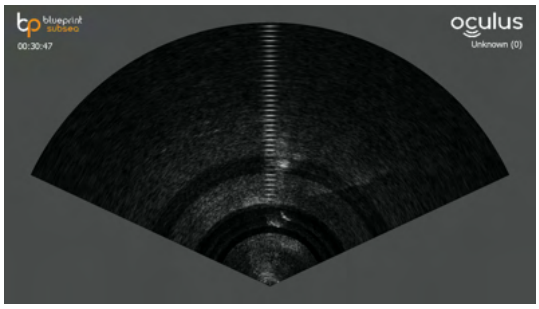


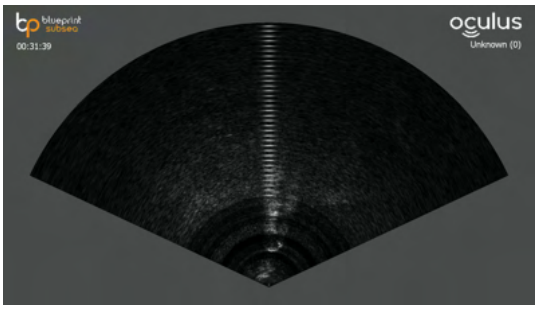
Dock

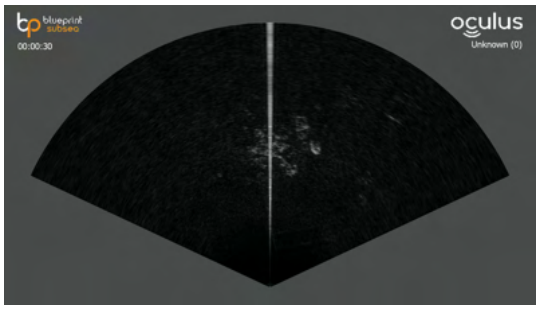

blueprint subsea
00:29:29
oculus
Unknown (0)

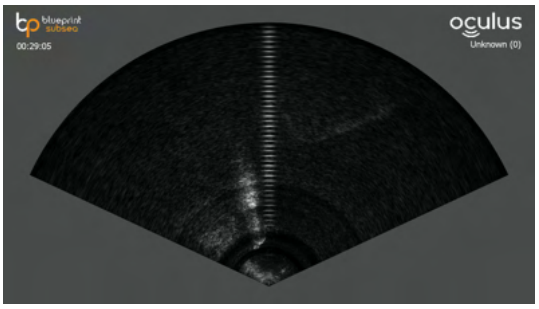

blueprint subsea
00:06:15
oculus
Unknown (0)

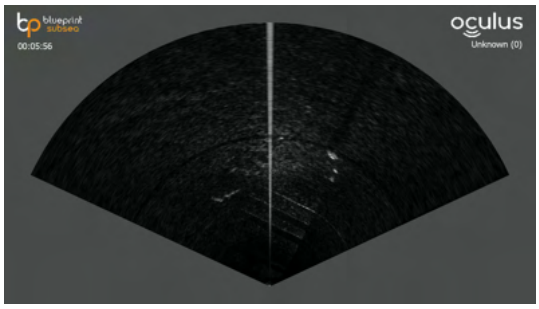

blueprint subsea
00:30:42
oculus
Unknown (0)

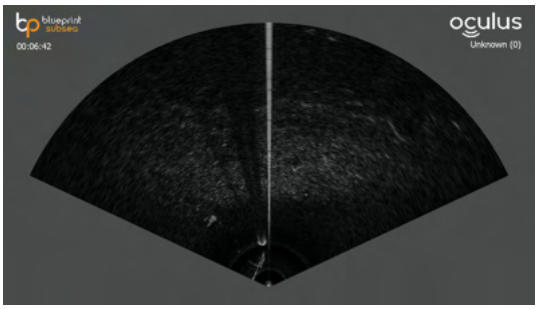

blueprint subsea
00:30:47
oculus
Unknown (0)

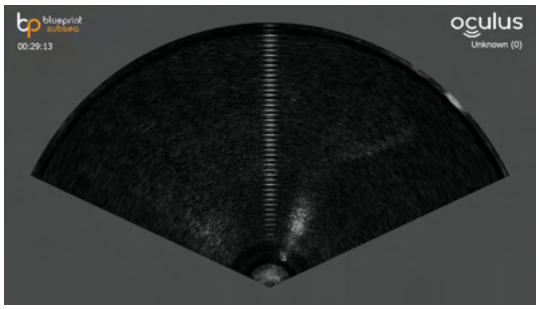

blueprint subsea
00:31:39
oculus
Unknown (0)


blueprint subsea
00:00:30
oculus
Unknown (0)

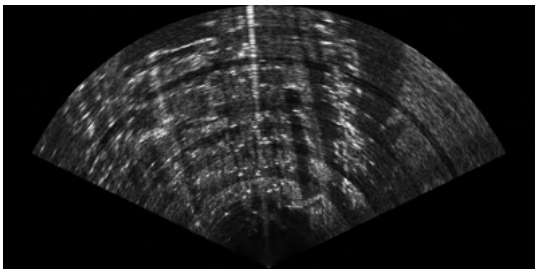

blueprint subsea
00:29:05
oculus
Unknown (0)

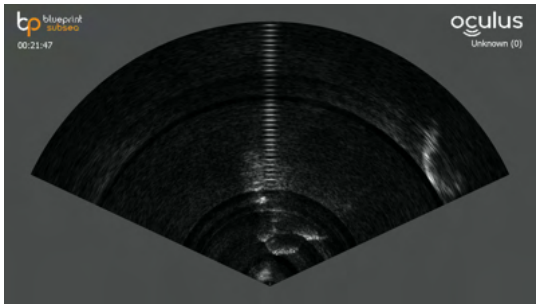

blueprint subsea
00:05:56
oculus
Unknown (0)

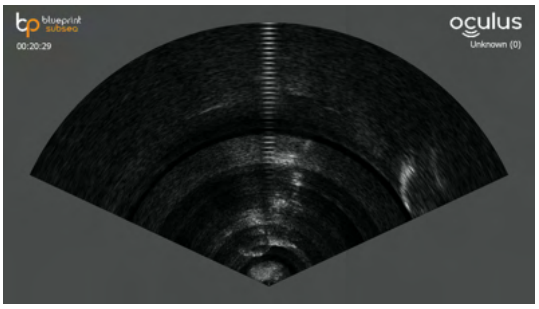

blueprint subsea
00:06:42
oculus
Unknown (0)

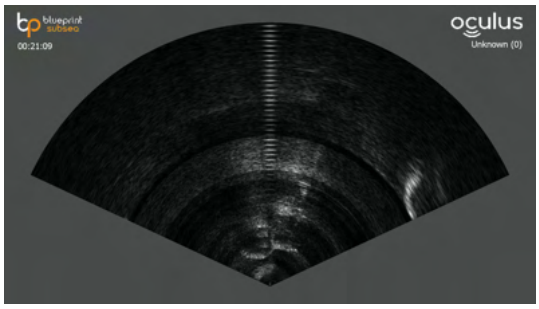

blueprint subsea
00:29:13
oculus
Unknown (0)

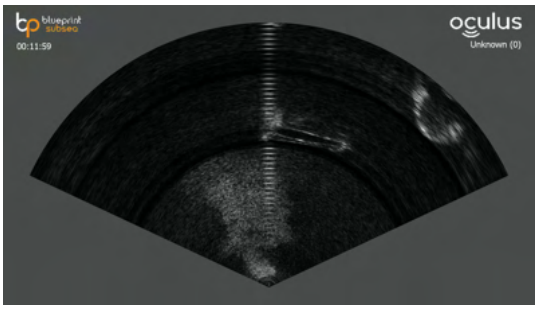
Man Made Unclassified

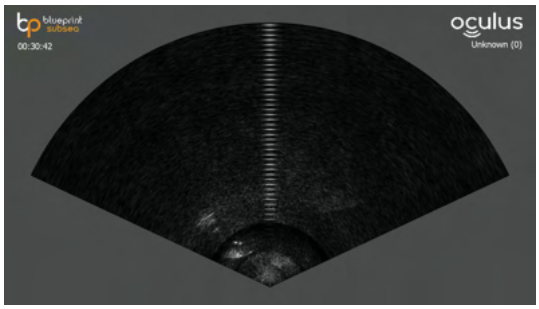


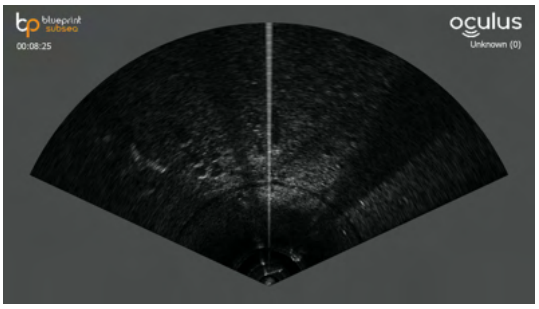


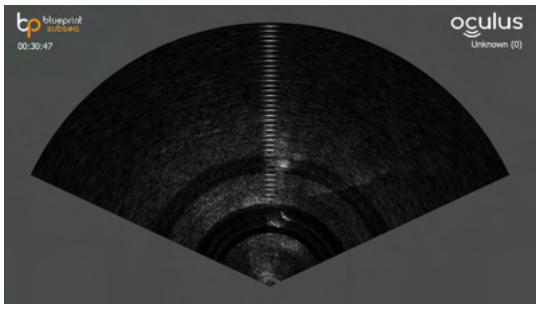


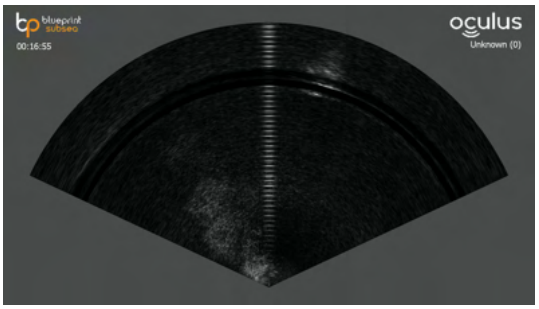


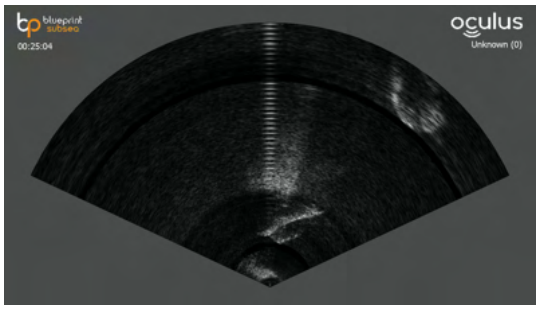




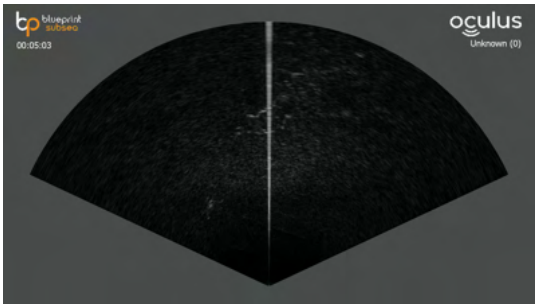


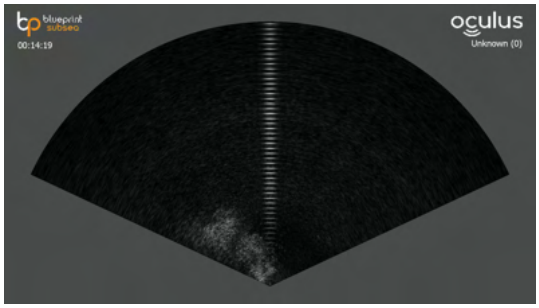


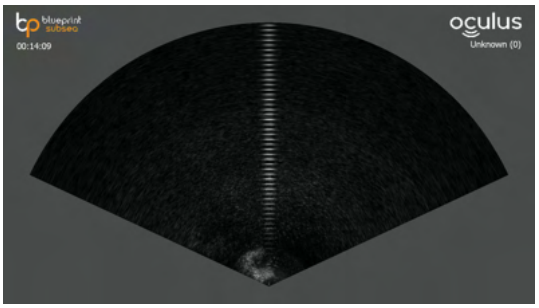


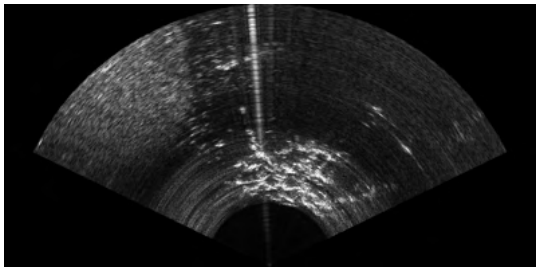


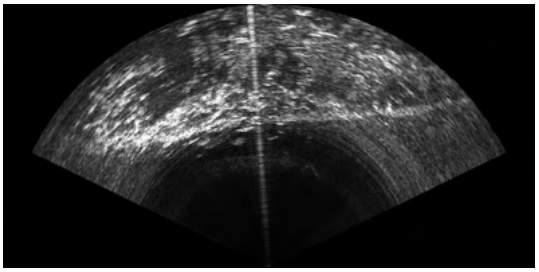
No Class

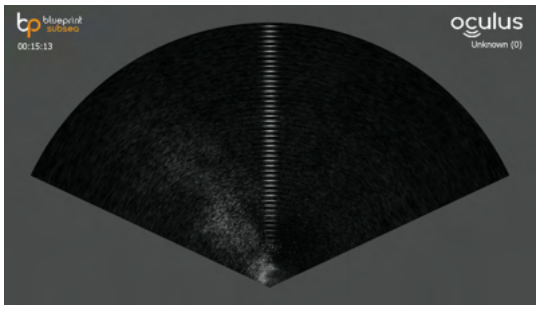

blueprint subsea
00:05:03
oculus
Unknown (0)

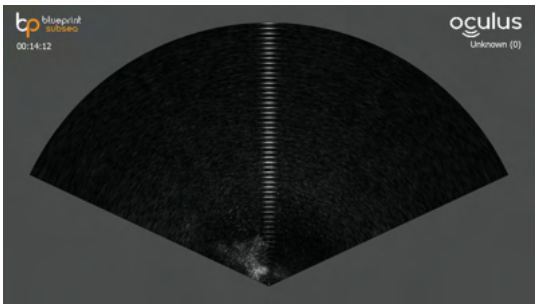

blueprint subsea
00:14:19
oculus
Unknown (0)

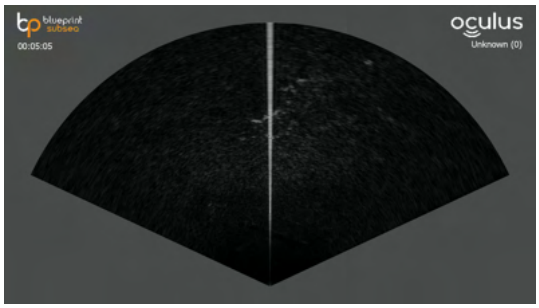

blueprint subsea
00:14:09
oculus
Unknown (0)

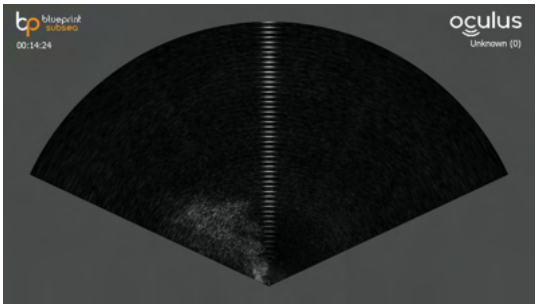


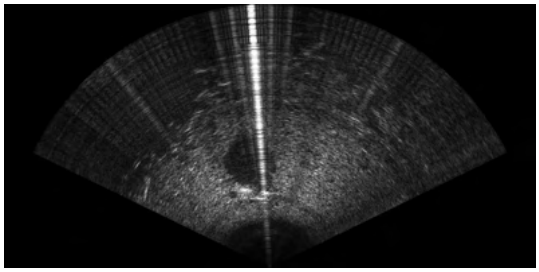



blueprint subsea
00:15:13
oculus
Unknown (0)

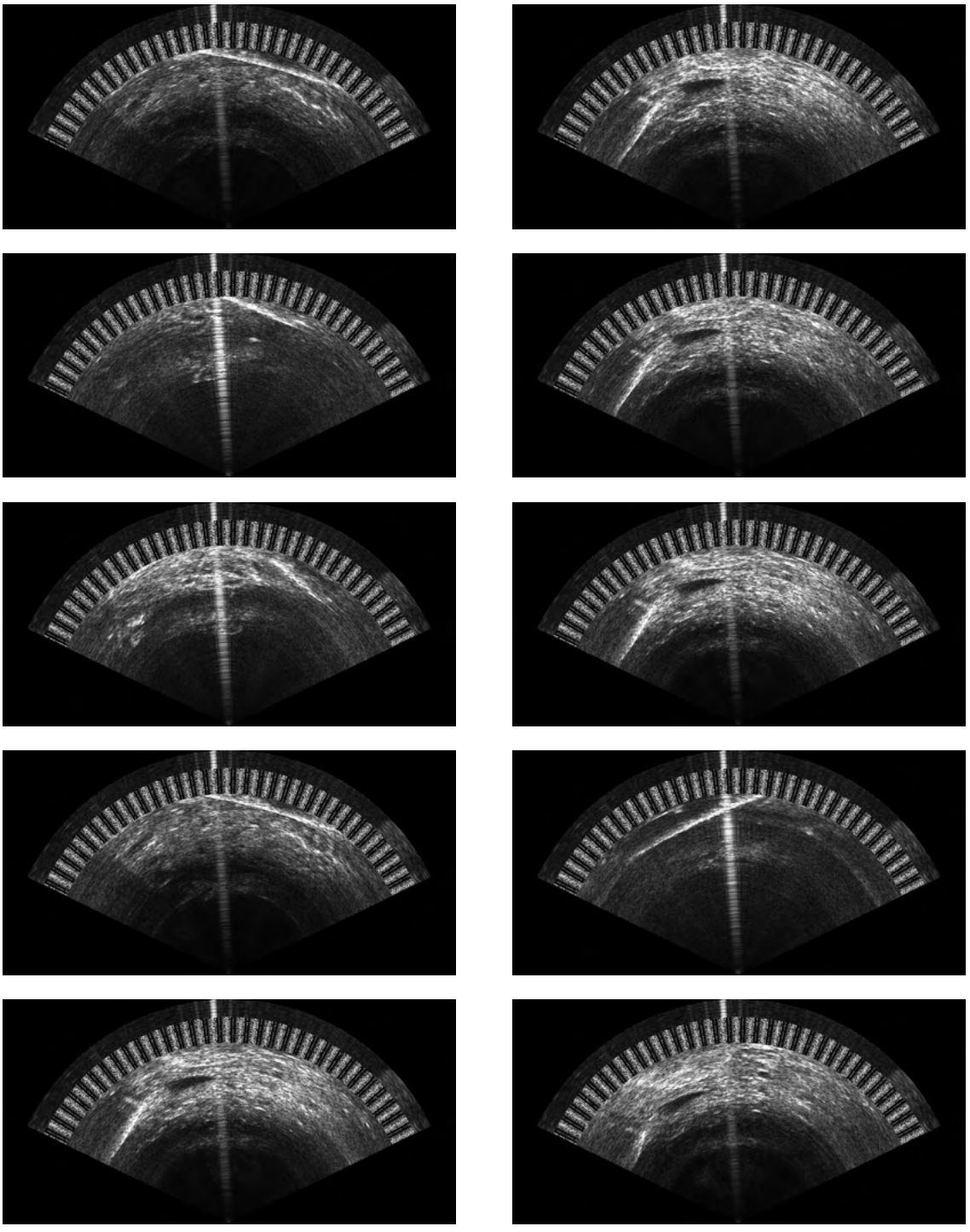

blueprint subsea
00:14:12
oculus
Unknown (0)


blueprint subsea
00:05:05
oculus
Unknown (0)

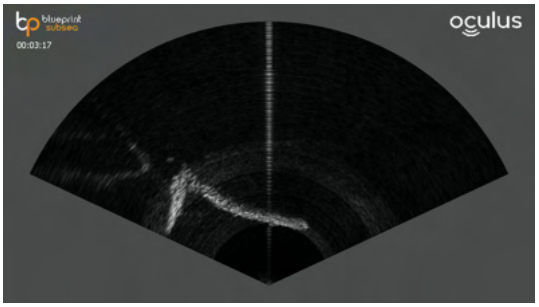

blueprint subsea
00:14:24
oculus
Unknown (0)

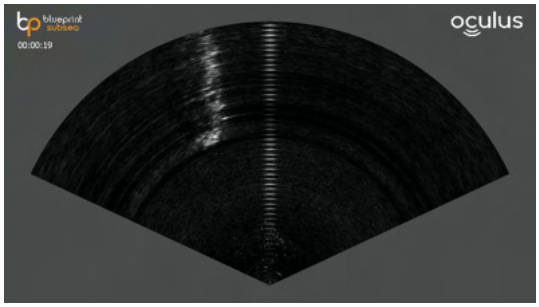


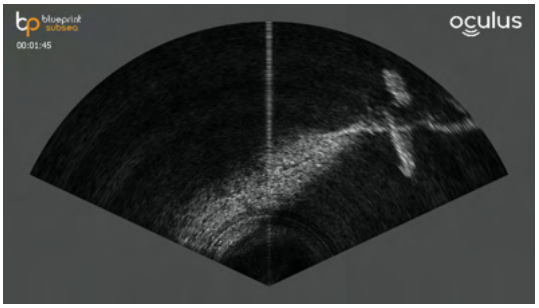
Pipe

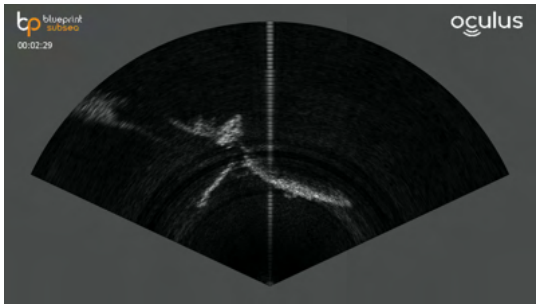


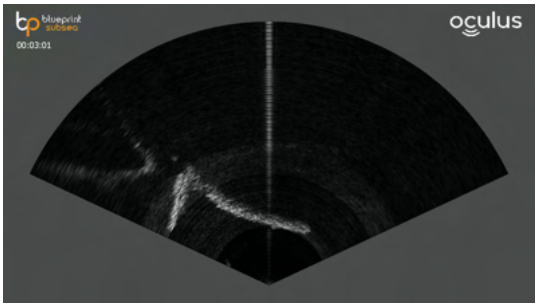
Propeller

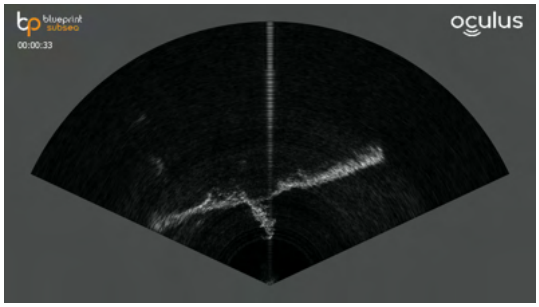

blueprint subsea
00:03:17
oculus

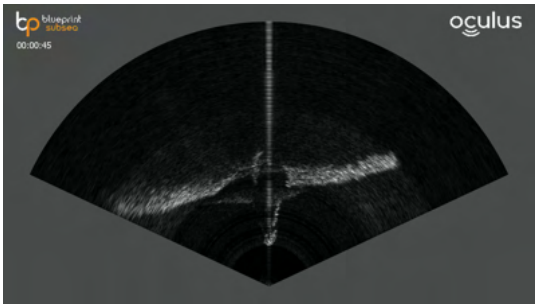

blueprint subsea
00:00:19
oculus

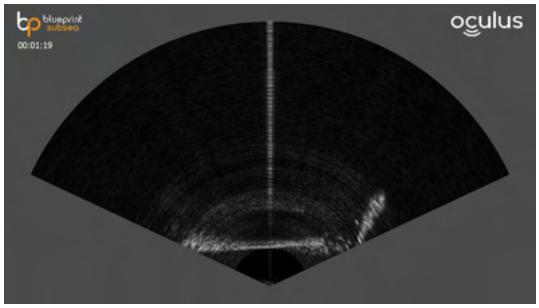

blueprint subsea
00:01:45
oculus

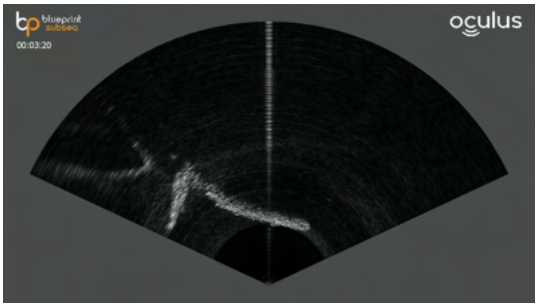

blueprint subsea
00:02:29
oculus

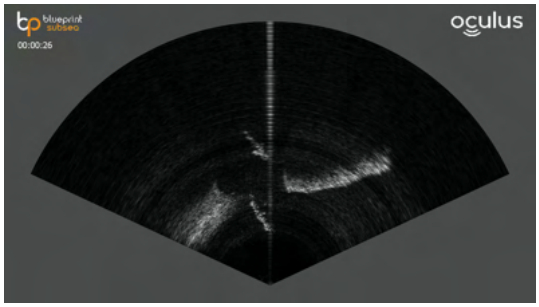

blueprint subsea
00:03:01
oculus


blueprint subsea
00:00:33
oculus

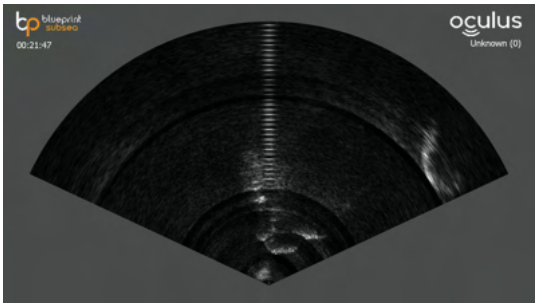

blueprint subsea
00:00:45
oculus

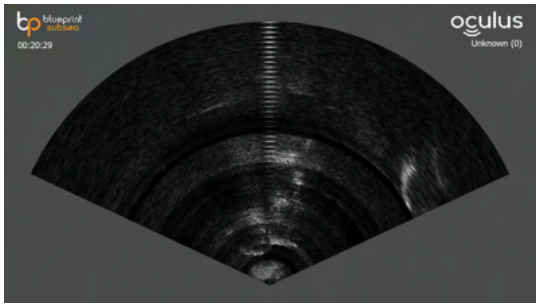

blueprint subsea
00:01:19
oculus

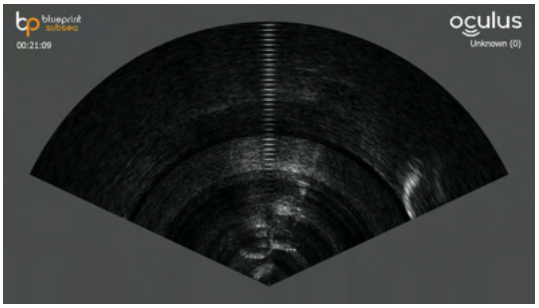

blueprint subsea
00:03:20
oculus

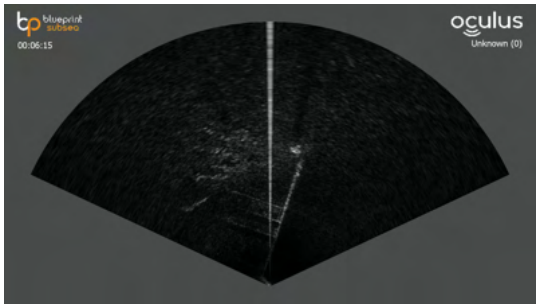

blueprint subsea
00:00:26
oculus

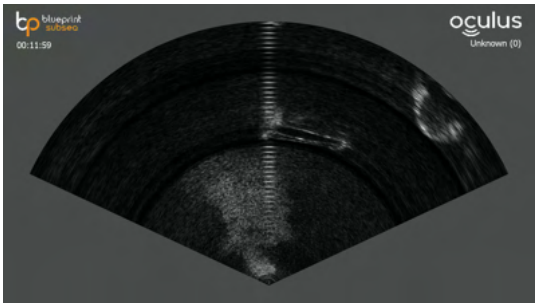
Pylon

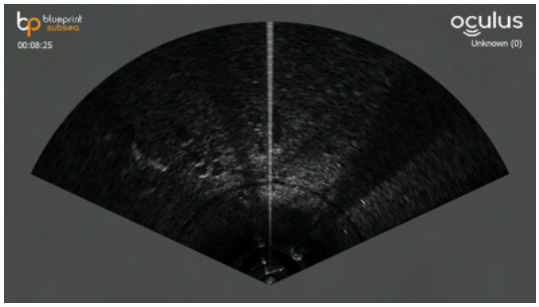

blueprint subsea
00:21:47
oculus
Unknown (0)

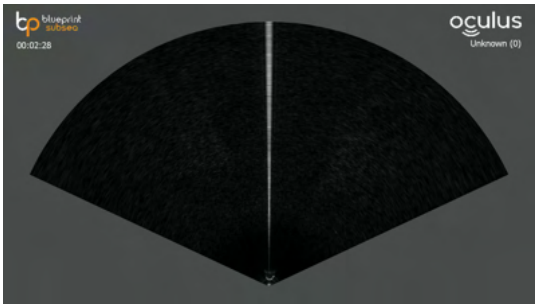

blueprint subsea
00:20:29
oculus
Unknown (0)

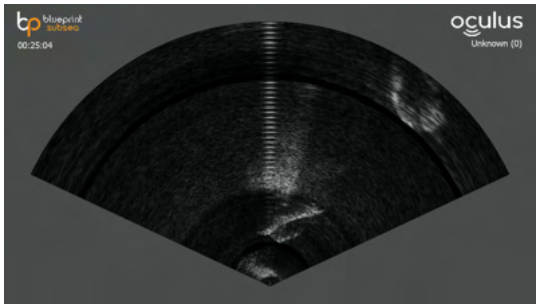

blueprint subsea
00:21:09
oculus
Unknown (0)

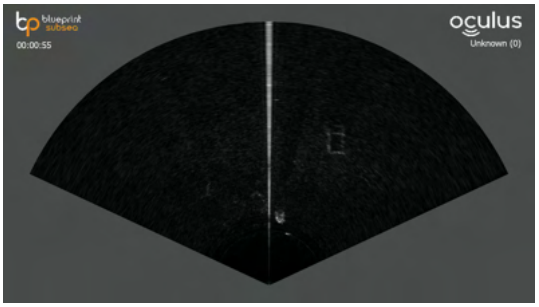

blueprint subsea
00:06:15
oculus
Unknown (0)

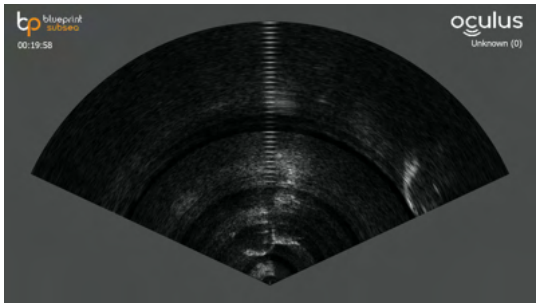

blueprint subsea
00:11:59
oculus
Unknown (0)


blueprint subsea
00:08:25
oculus
Unknown (0)

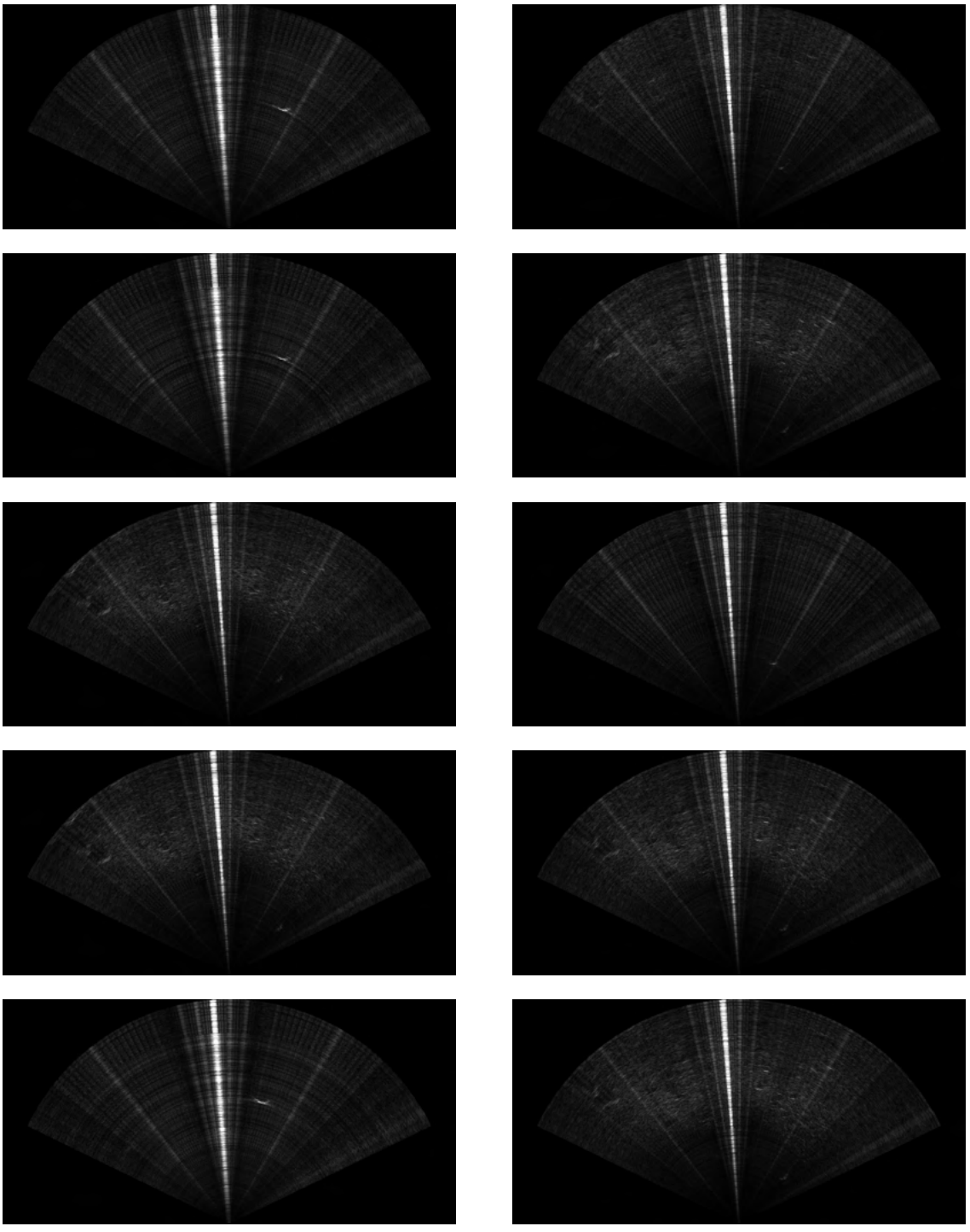

blueprint subsea
00:02:28
oculus
Unknown (0)


blueprint subsea
00:25:04
oculus
Unknown (0)

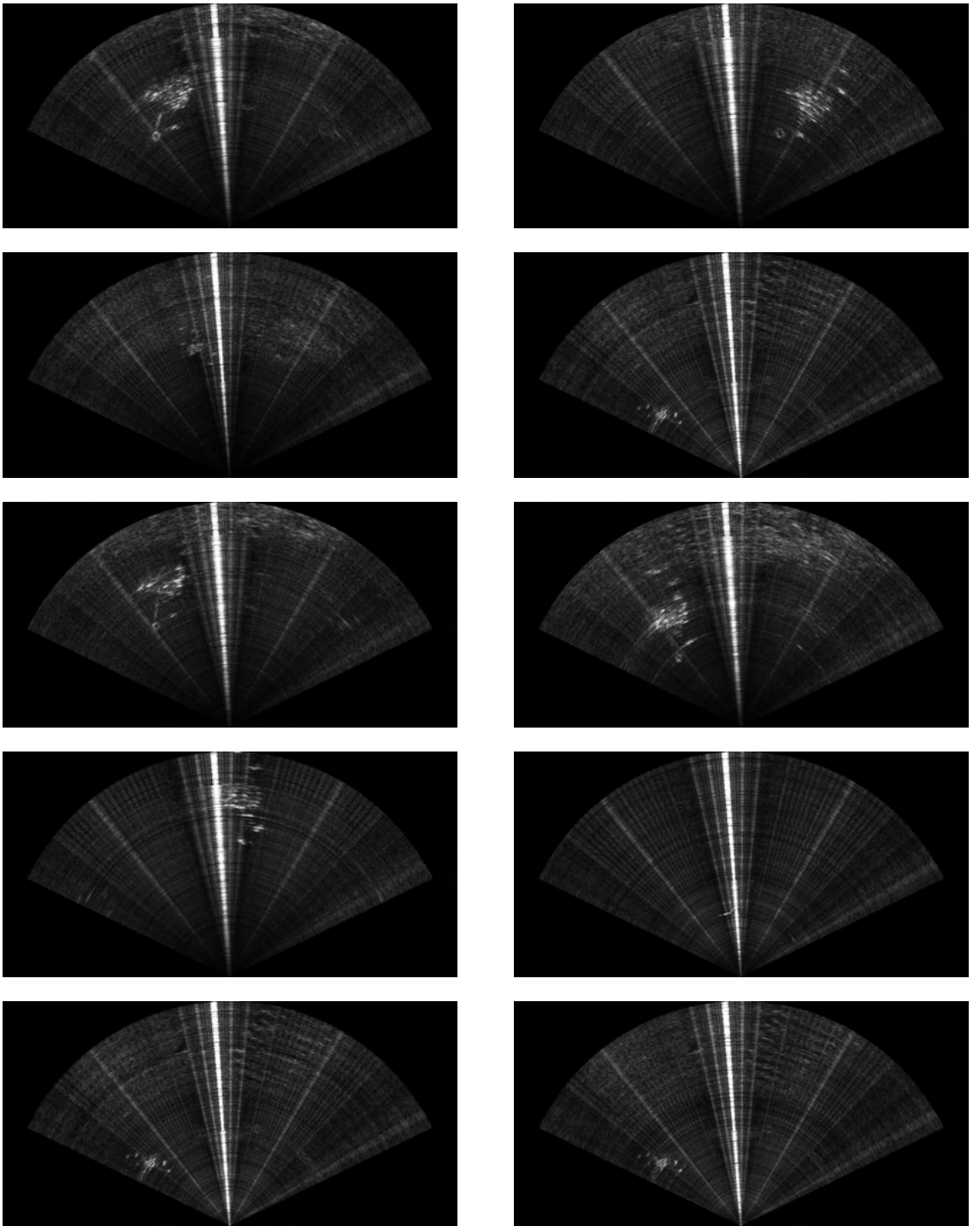

blueprint subsea
00:00:55
oculus
Unknown (0)


blueprint subsea
00:19:58
oculus
Unknown (0)

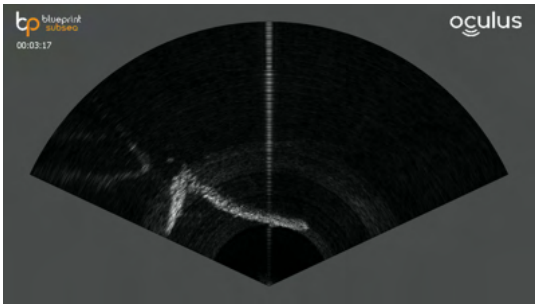
Rope

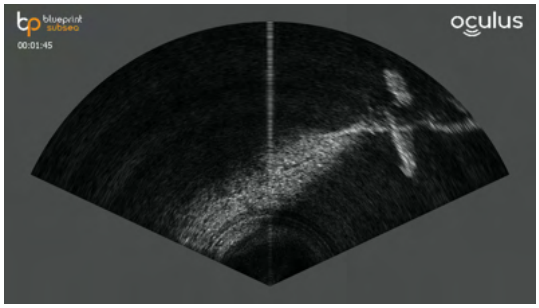


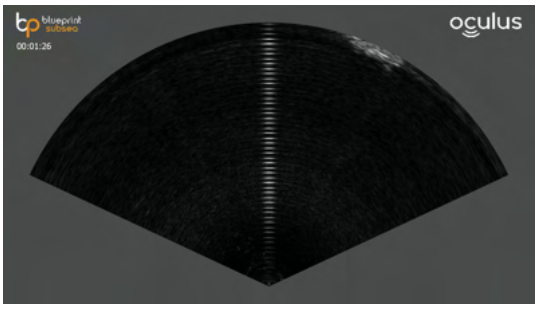
ROV

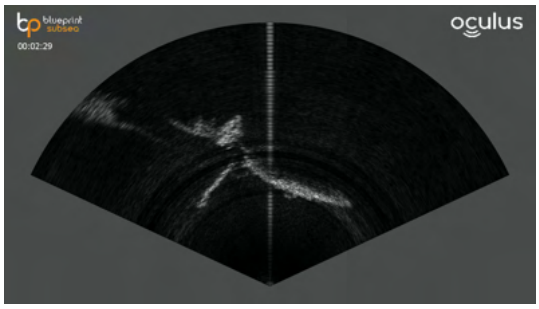


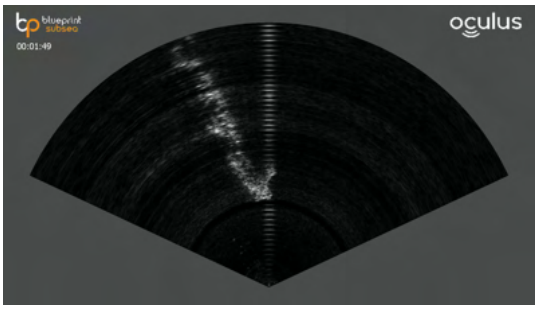
Rudder

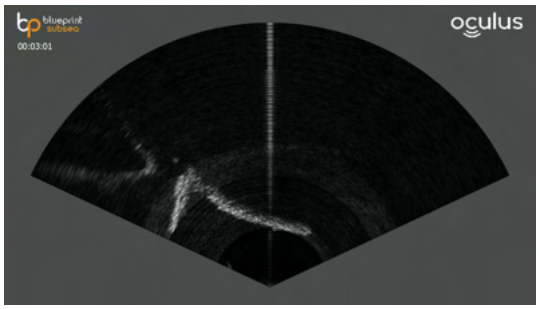

blueprint subsea
oculus
00:03:17

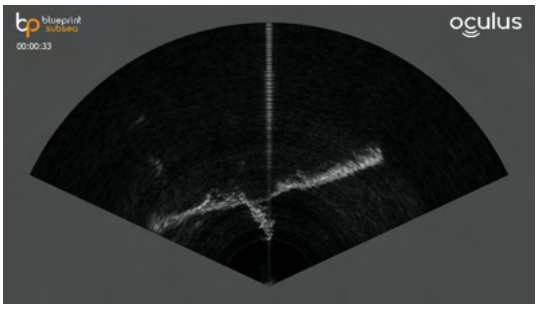

blueprint subsea
oculus
00:01:45

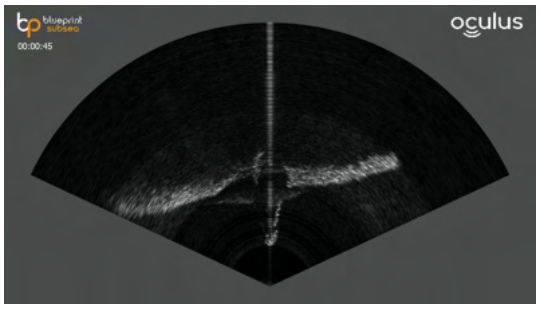

blueprint subsea
oculus
00:01:26

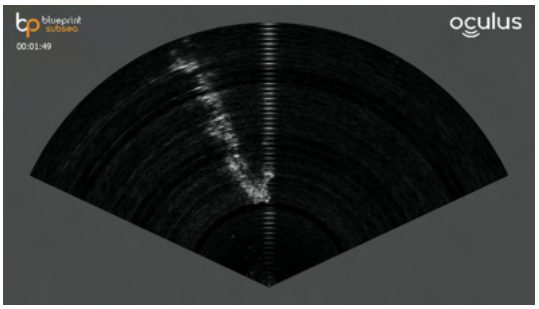

blueprint subsea
oculus
00:02:29

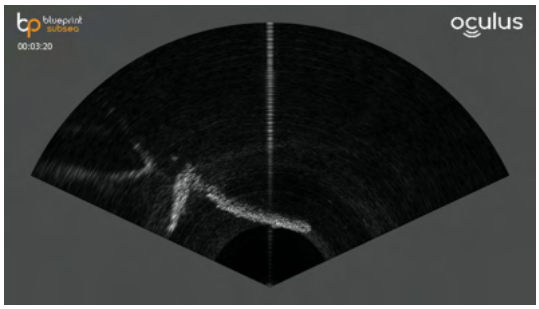

blueprint subsea
oculus
00:01:49


blueprint subsea
oculus
00:03:01

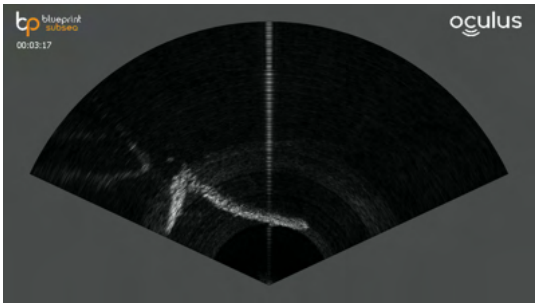

blueprint subsea
oculus
00:00:33

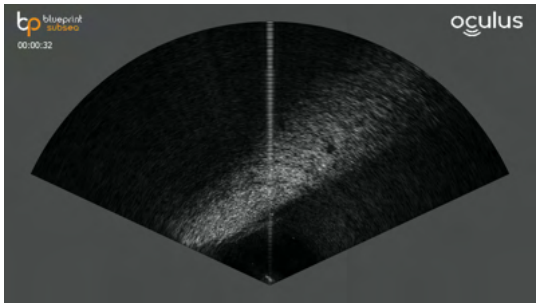

blueprint subsea
oculus
00:00:45

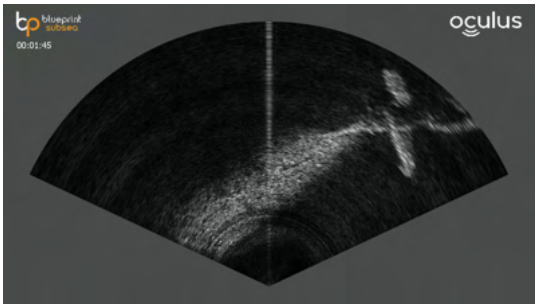

blueprint subsea
oculus
00:01:49

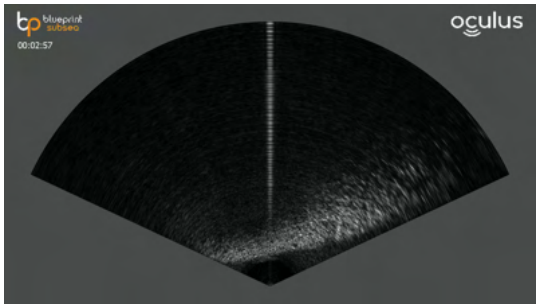

blueprint subsea
oculus
00:03:20

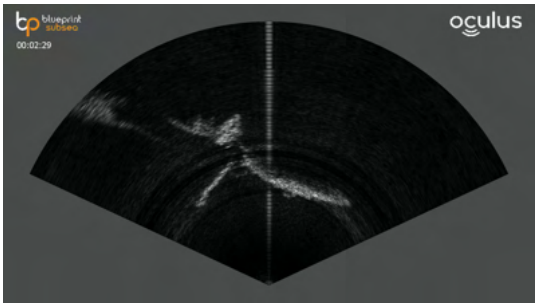
Ship Hull

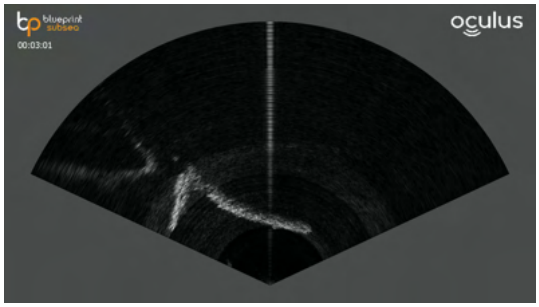

blueprint subsea
00:03:17
oculus

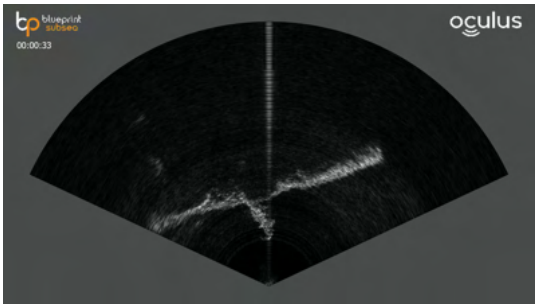

blueprint subsea
00:00:32
oculus

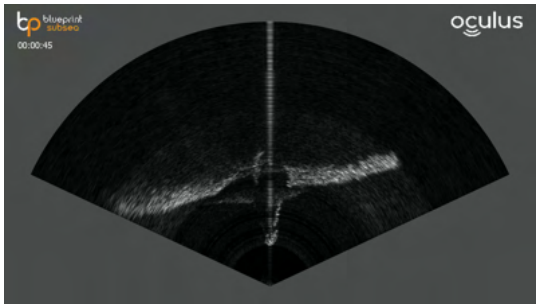

blueprint subsea
00:01:45
oculus

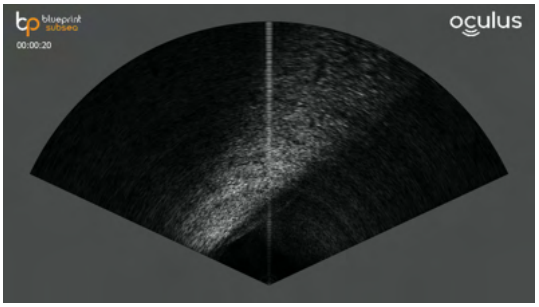

blueprint subsea
00:02:57
oculus

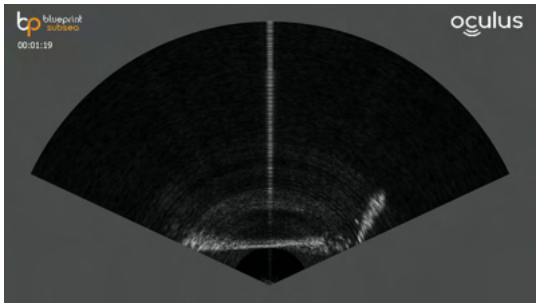

blueprint subsea
00:02:29
oculus


blueprint subsea
00:03:01
oculus

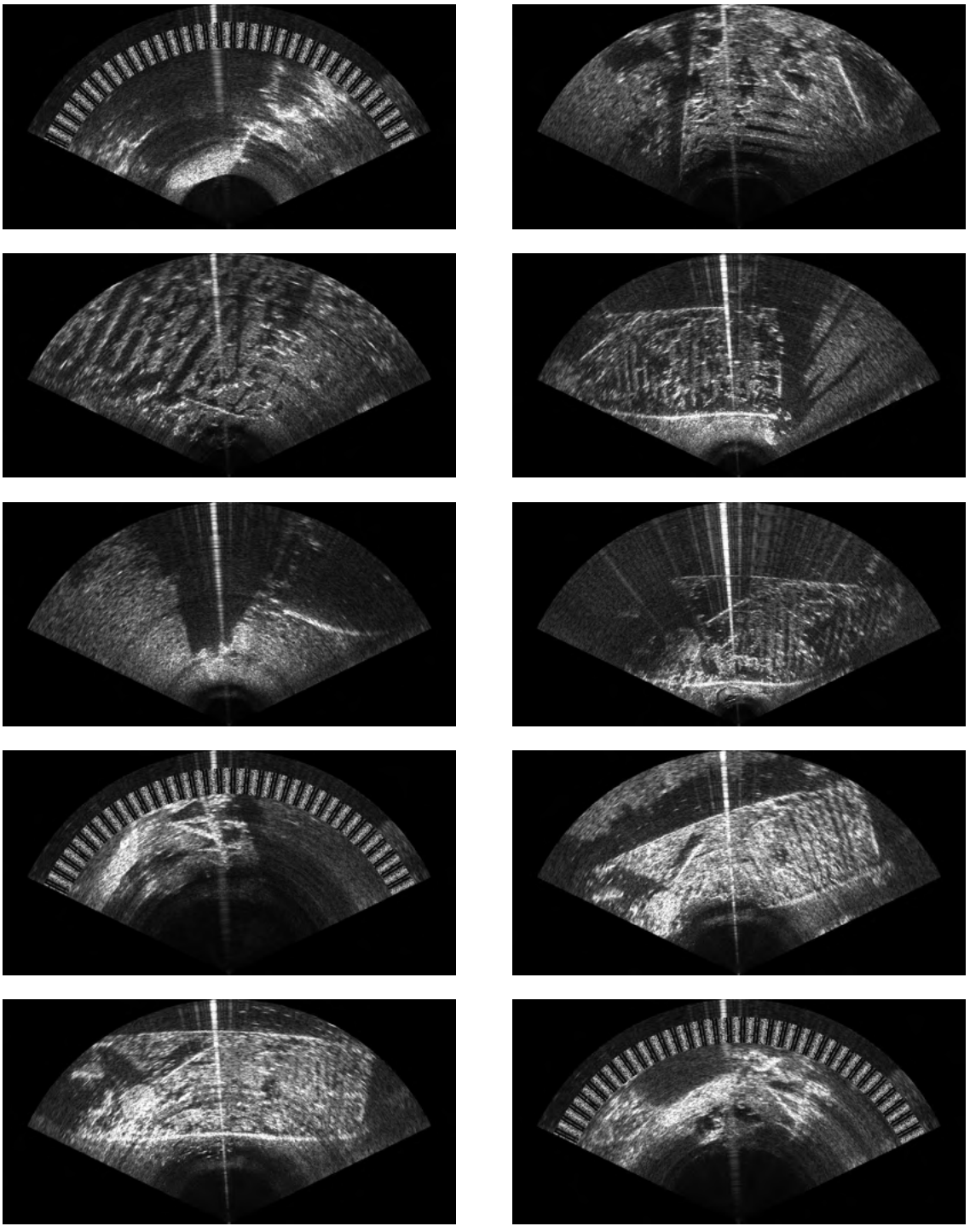

blueprint subsea
00:00:33
oculus


blueprint subsea
00:00:45
oculus


blueprint subsea
00:00:20
oculus


blueprint subsea
00:01:19
oculus

Shipwreck



Tire