ROBOTICS AND MECHATRONICS

USING MACHINE LEARNING TO DETECT VOIDS IN AN UNDERGROUND PIPE USING IN-PIPE GROUND PENETRATING RADAR

Antoinette Abhinaya

MSC ASSIGNMENT

November, 2021

072RaM2021
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Summary

The maintenance of underground pipelines and sewage systems is a significant challenge since they are underground. A sewer pipe installed underground is surrounded by compact sand. Due to water leakage, increase in groundwater level, or natural phenomenon, the concrete structure of the sewer might develop cracks. Due to the deterioration of the sewer surface, voids are formed in the soil around the pipe. When left unnoticed, these voids lead to a loss in the soil's structural integrity, leading to the formation of a sinkhole. This eventually leads to the ground surface caving in, which causes a significant disturbance to the surroundings and needs time and workforce to fix. In order to avoid the formation of a sinkhole, regular inspection of void formation by non-invasive methods is required.

This research is part of the Technology Innovation for Sewer Condition Assessment - Long-distance Information-system (TISCALI) project. A part of TISCALI researches the use of non-invasive methods from within the underground utilities. The most commonly used non-invasive method of the sewer inspection is the Ground Penetrating Radar (GPR). In this thesis, in-pipe GPR is used wherein the system is sent inside the pipe. This helps in detecting voids around the sewer pipeline. A high frequency of operation can be used due to the reduction in distance between the voids and the antenna. The voids and objects underground create hyperbolic reflections in the GPR radargrams. The objective of the thesis is to analyse and choose an object detection method that can be useful in void detection. The second objective is to create a model that can detect hyperbolic reflections and thereby detect the different compositions of voids. In addition to this, Image processing methods are applied to the Machine-Learning algorithms to check for improved accuracy.

The GPR radargrams are simulated using the gprMax software. Voids of differing sizes and compositions are simulated around a sewer pipeline. Faster-RCNN, YOLOV2, and YOLOV3 are chosen as the three CNN methods for void detection. In comparing the detector performance of the three methods, YOLOV3 has the highest accuracy of 99.6%. In void detection, it was inferred that Water voids have the highest accuracy in detection in all three methods due to many reflections in the radargrams. In addition to this, image processing was applied to the YOLOV3 method and was deemed useful when detecting hyperbolic reflections that were disturbed or in a heterogeneous sand environment.

In conclusion, YOLOV3 is the best fit for void identification, and in addition to feature extraction by image processing, it is notable in detection. Therefore, a symbiotic relationship between the operator and the algorithm can be created by applying machine learning to current in-pipe GPR practices.

Acknowledgement

Foremost, I would like to express my sincere gratitude to my daily supervisor Hengameh for guiding me throughout the course of my thesis. I want to specially thank her for all the motivation she gave me when I was facing difficulty with the results. Thank you for being a very understanding and kind supervisor and more than that, a really good friend. I would specially like to thank my supervisors dr.ir.E.C.Dertien and dr.ir.J.F.Broenink for their valuable feedback with the work and the report. Their brilliant comments and suggestions in the last few months of the thesis was very helpful in the writing stages.

I would like to thank Geert Jan Laanstra for his help when I was trying to figure out the working of the HPC Cluster. Special thanks to the members of the staff working on the Cluster because without the availability of this resource, I would not have been able to execute my tests. I also want to thank Jolanda for being my biggest strength when I was running around trying to schedule meetings and my thesis defense date. Thank you for also listening to me rant when we were not able to fix the date and being a moral support.

Without the love and support from my parents, none of this would have been possible - I take this moment to thank them for everything they have done for me and for being my number one supporters. I also want to thank my brother for motivating me by asking me when I'll be back home. I thank my friends at the RaM department for distracting me during the coffee breaks and keeping me motivated. I also wish them the best of luck with their thesis.

Lastly, I am so blessed to have made some good friends in Enschede. I thank my bestfriends for the meals we shared, for the trips we went on and for being my source of happiness.

Antoinette Abhinaya
Enschede, November 4, 2021

Contents

Acronyms

ANN Artificial Neural Networks. 13

ASCII American Standard Code For Information Interchange. 8

C3 Column Connection Clustering. 15, 16, 17

CCTV Closed-Circuit Television. 2

CLAHE Contrast Limited Adaptive Histogram Equalization. 22

CMP Common Midpoint. 25

CNN Convolutional Neural Networks. iii, 3, 4, 9, 10, 11, 12, 15, 16, 17, 18, 28, 29, 31, 32

CPU Central Processing Unit. 8

CTFP Column-based Transverse Filter Points. 15

CUDA Compute Unified Device Architecture. 21

DCSE Double Cluster Seeking Estimate. 15

DNN Deep Neural Networks. 14

FDTD Finite-difference time-domain. 7, 14

FN False Negative. 17

FP False Positive. 17

GPR Ground Penetrating Radar. iii, ix, 2, 3, 4, 5, 6, 7, 8, 13, 14, 15, 16, 21, 23, 24, 49, 50

GPU Graphics Processing Unit. 8, 21, 28

GUI Graphical User Interface. 7

HDF5 Hierarchical Data Format version 5. 21

HOG Histogram of Oriented Gradients. 15

HPC High Performance Computing. 7, 21, 28

IoU Intersection Over Union. 11

ML Machine-Learning. iii, 3, 4, 12, 13, 14, 15, 16, 50

MPI Message Passing Interface. 8

PNG Portable Network Graphics. 21, 24

RCNN Region-based Convolutional Neural Networks. ix, x, xii, 9, 10, 15, 16, 17, 18, 28, 29, 30, 31, 37, 38, 39, 40, 45, 47, 48

ReLU Rectified Linear Unit. 11, 28, 29, 46

ROI Region of Interest. 16

RPN Region Proposal Network. 9, 10, 28, 29

Rx Receiver. 5, 8

SGDM Stochastic Gradient Descent with Momentum. 12, 30, 32

SVM Support Vector Machines. 15, 16, 17, 18

TISCALI Technology Innovation for Sewer Condition Assessment - Long-distance Information-system. iii, 1, 3

TN True Negative. 17

TP True Positive. 17

tR Two way travel time. 7, 26, 36

Tx Transmitter. 5

YOLO You Only Look Once. ix, x, xii, 10, 11, 12, 15, 16, 17, 18, 28, 31, 32, 33, 34, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50

List of Figures

List of Tables

1 Introduction

The TISCALI (Technology Innovation for Sewer Condition Assessment - Long-distance Information-system) project at the University of Twente aims to create a non-invasive method of inspecting the sewer lines that will thereby decrease any possibility of damage to the infrastructure. "The project aims at utilizing, integrating, and further development of relatively low-cost, off-the-shelf, techniques to arrive at an objective detection and quantification of defects in sewers and to determine the constructive strength and stability of sewers" TISCALI (2021). This Graduation project is a part of the TISCALI project.

1.1 Problem Context

With the rapid rise in urbanisation, buried pipelines are required to supply water to and provide drainage from several homes. The collection and transport of wastewater happen through the public sewer pipelines that are underground. In the Netherlands alone, the sewer lines run for more than 90,000 km. With the growing and continuous usage of sewer lines over the years, they are prone to structural defects which need to be identified before it becomes an issue. Due to being underground, maintenance of such structures is ignored until an accident occurs. The maintenance of pipelines results in digging up the ground to inspect and fix the issue, which is inconvenient to the public, expensive, and requires quite some workforce.

The installation process of the pipeline involves placing the pipe over a layer of bedding and closing the trench with the backfill soil. The backfill is filled compactly to avoid gaps or voids. The lifetime of a pipe depends on the interaction between the pipe and the surrounding soil (McGrath et al., 1999). After installation, over time, due to erroneous placement of the pipes or natural degradation, cracks may start to form on the surface of these pipelines. For example: when the groundwater level increases, the water can infiltrate the pipeline through the cracks created on the surface. During this infiltration, the surrounding soil particles are also washed into the sewer creating a void in the soil. These voids may be filled with air or water and cause the pipe to lose the support of the surrounding soil, resulting in further structural damage. If left unnoticed, this finally results in the collapse of the ground surface over the pipe. This can be summarised into a three step collapse process (Davies et al., 2001) as shown below in Figure 1.1:

1. An initial defect - caused by an external or natural disturbance.
2. Deterioration - Infiltration of groundwater through the cracks or ex-filtration of sewer water into the groundwater table also washing away soil particles around the sewer line with it.
3. Collapse - Development of voids or loose ground caused by the erosion of soil particles around the sewer. Voids left unnoticed lead to a loss in the structural integrity of the soil, eventually resulting in a sinkhole.

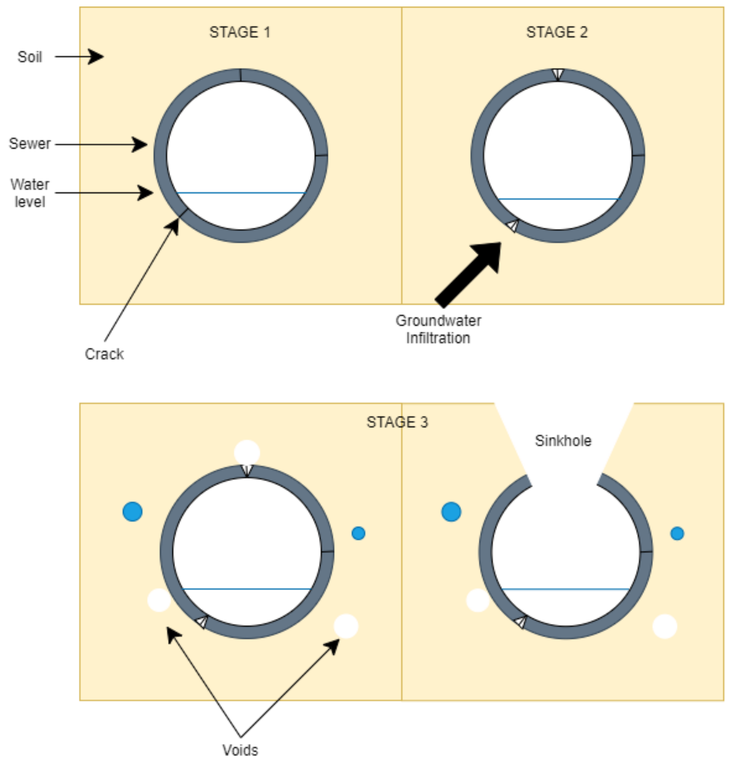


Figure 1.1: Collapse Process

The importance of the need for sewer inspection can be understood via the following example. In two separate events in 2007 and 2010, two giant sinkholes nearly 20 m wide and 100 m deep were formed in the city of Guatemala. Further investigation showed that it was caused due to a heavy storm followed by soil erosion around the sewer pipeline due to ex-filtration of water from the sewer. The government was later demanded to periodically inspect the sewer lines since it was found that the city's sewer system was in a bad state (National Geographic, 2021).

Hence, it is essential to prevent the formation of a sinkhole by regular inspection of void formation and maintenance when required.

1.2 Inspection methods

Current sewer assessment methods include manual inspection, photographic inspection, and CCTV inspection. The mentioned methods are commonly used, involve human intervention, and hence depend on the operator's skill. These methods mainly provide visual data about the inside of the pipe and do not provide insight into the structural and environmental defects around the pipe. In (Wirahadikusumah et al., 1998), non-invasive methods like Infrared thermography, Sonic Measurement techniques, and Ground Penetrating Radar (GPR) are explained.

The GPR is the most efficient non-destructive evaluation method due to the easier portability of the system, low initial cost, ability to survey large areas in less time, and ability to give faster results (Travassos and Pantoja, 2019). Therefore, the GPR is the focal point of this research. By using GPR, the radar will detect the voids around the surface of the sewer.

GPR is most commonly used by running the equipment on the ground surface and obtaining the data. A low frequency of operation is used for a greater depth in penetration, but this results in a low resolution in the corresponding radargram. To get a better resolution, a higher frequency of operation can be used by sending the GPR inside the pipe. This reduces the dis-

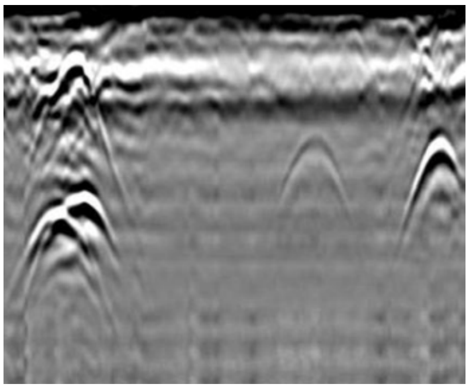


Figure 1.2: Example Radargram

tance between the voids to be identified and the antenna, giving a better resolution in radargram data. Most of the research and development in the field of GPR radargram interpretation focuses on the ground surface GPR and the identification and detection of subsurface objects. The GPR system can be sent inside the pipe to find the voids directly behind the sewer lines since this research primarily focuses on void identification. The voids formed around the sewer pipeline are filled with air or water. They can be distinguished based on their detection by the in-pipe GPR.

In the radargram produced by the GPR, the objects found underground or voids, in this case, are represented in the form of hyperbola-shaped signatures. This requires highly skilled personnel since the object reflections need to be read from the radargram. It might be challenging at times to read hyperbolic signatures due to the presence of noise or overlapping of hyperbolas as shown in Figure 1.2.

Machine-Learning (ML) algorithms and Convolutional Neural Networks (CNN) can detect voids behind a sewer pipeline. Hence to solve manual errors, automatic detection methods to localize hyperbolas in a radargram are employed. With considerable research in the field of Machine-learning, several algorithms could be used for object recognition or hyperbola fitting. Previous work done as a part of the TISCALI project focus on the accuracies between a planar (ground surface) and cylindrical (in-pipe) topology using different feature sets (Delft, 2019). A dummy classification model is used to classify the hyperbolas. It was concluded that the classification results for a planar and a cylindrical topology are identical. As an extension of his graduation assignment, the next step is to select a Machine-Learning model suitable for multiple void detections to improve the classification accuracy. This gives rise to the following Design criteria that is addressed in the course of this thesis project.

1.3 Research Objective

The following objectives are the main focus of the assignment.

1. Object detection using Automatic detection methods.
   - To analyse the existing approaches to object detection in subsurface scenarios.
   - To compare between the different object detection methods in hyperbola recognition and fitting.
2. Differentiation between different void compositions.
   - To simulate subsurface radargrams with voids and train the model to differentiate between voids filled with water or air.

- To determine how reflections due to different void compositions can be trained using CNN.

3. Inclusion of Pre-processing with Machine Learning.

   - Compare and check if there is an added benefit to contrast enhancement and image processing in a synthetic radargram.

1.4 Approach

Since the research on void identification is scarce, there is no dataset with ground truth indicating different voids' presence. It is not easy to find a relevant dataset that has different void compositions. Hence, the radargrams are synthesised as produced from an in-pipe GPR system, detecting multiple voids of different materials. Since the field of ML and CNN is vast, the approaches used for object detection in subsurface conditions are analysed. A suitable method that can be used for hyperbola recognition is chosen.

With the synthesised dataset, pre-processing is applied to enhance the features, and the dataset is labeled to train using the selected supervised machine learning model. The trained detectors are further compared with each other to find a fitting model for void detection. The selection is done based on the accuracy of hyperbola detection and void classification. The different types of voids are classified based on their reflections and appropriately trained using CNN. Depending on the accuracy of the detectors, it is further discussed if there is a need to add image processing methods for feature extraction.

The research aims to compare different state-of-the-art Machine-Learning methods applied to void detection and conclude a model that can serve as an automated void detection method.

1.5 Outline

Chapter 2 provides more insight into the theory behind GPR, ML, and the software used to obtain the results. Chapter 3 of this report explains related work to the automatic detection of hyperbolas in a radargram and the different algorithms used. An analysis of the Machine-Learning models chosen for this assignment is explained. The chosen algorithms are tested on the datasets, and the approach is explained in Chapter 4. The results of the detection algorithms and a detailed discussion of the performance are contained in Chapter 5. Chapter 6 consists of the concluding remarks, recommendations, and future scope of the project.

2 Background

The background chapter provides a theoretical framework on the different tools and concepts used to solve the problem statement. Section 2.1 focus on the theory and working of the Ground Penetrating Radar (GPR). The theory of the software used to simulate the experimental radargrams is explained in Section 2.2, the basic concept of Machine learning is briefed in Appendix B and the theoretical background of three methods to be used for void detection are explained in this chapter.

2.1 Ground Penetrating Radar (GPR)

GPR is a method of Non-destructive evaluation of subsurface objects. It is used widely due to its penetrative power and its ability to detect shallow or deep objects based on the frequency of the electromagnetic wave. GPR is a time-dependant technique that can provide a 3-D image of the subsurface and accurate depth indication of the objects underground. GPR works on the principle of Electromagnetic scattering. The GPR device consists of a pulse transmitter, a receiver antenna, a control unit, a power supply, and the survey cart.

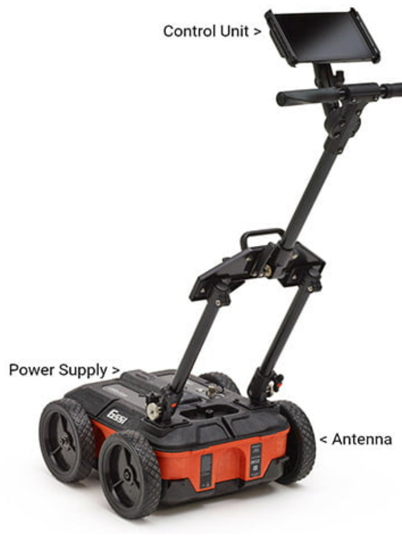


Figure 2.1: Commercial GPR System (GSSI, 2021)

GPR produces pulsed electromagnetic waves with a varying frequency that depend on the application that travels through the ground until these waves meet geological targets or different objects then they reflect to the surface (Kuo et al., 2005). The GPR transmits pulses of electromagnetic waves into the subsurface at a velocity that depends on the medium's permittivity(ϵ). The wave travels downwards, and due to the different material that might be present in the subsurface, a proportion of the wave is reflected when it hits an object with a different permittivity(ϵ). The remaining signal is refracted further into the ground and reflected when another difference in the dielectric medium is met. This process is repeated until the signal becomes weak to be received by the antenna. The higher the difference in permittivity(ϵ), the stronger the reflections are. Hence, the presence of voids or subsurface objects will have strong reflections due to the high contrast in permittivity(ϵ) with the surrounding soil. The received waves are recorded in the data storage system for further interpretation.

The basic working principle of the electromagnetic wave scattering and antenna Transmitter/Receiver (Tx/Rx) of the GPR device is as illustrated in Figure 2.2.

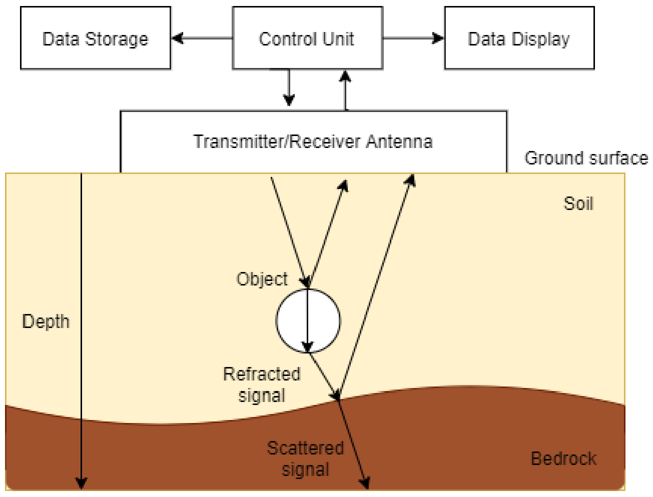


Figure 2.2: GPR Working principle

The velocity of the propagated wave can be calculated using the following formula :

$$v = \frac{c}{\sqrt{\epsilon_r}} \tag{2.1}$$

where v is the velocity of the propagated wave, c is the velocity of light, and ϵ_r is the relative permittivity of the medium.

Since the velocity depends on the permittivity (ϵ_r) of the medium of propagation, the signal transmitted through two different materials at the same distance would be received by the antenna at different times.

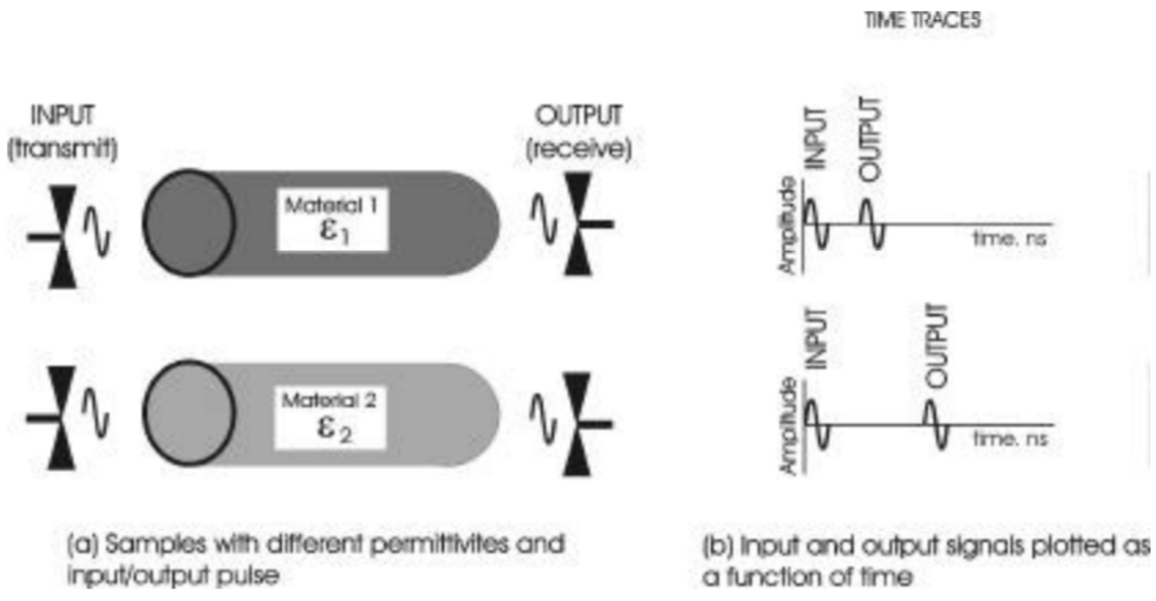

TIME TRACES
INPUT (transmit)
OUTPUT (receive)
Material 1 ε_1
Material 2 ε_2
Amplitude
INPUT
OUTPUT
time, ns
(a) Samples with different permittivites and input/output pulse
(b) Input and output signals plotted as a function of time

Figure 2.3: Time versus Amplitude plots showing time difference between the input and output plots of the corresponding samples (Daniels, 2000)

As shown in Figure 2.3, the recording of both pulses over a period of time by the receiver antenna system is called a "trace". A trace is the primary measurement of the GPR survey. It can be thought of as a time history of the travel of a single pulse from the transmit antenna to the receiver antenna and includes all of its different travel paths (Daniels, 2000).

The time taken for the propagation of the transmitted wave through the medium and to be received by the antenna is defined as the Two way travel time (tR). The time of arrival of the reflected wave can be obtained from the recording of the trace. The depth of penetration can hence be calculated using the following formula:

$$D = \frac{tR \times v}{2} \tag{2.2}$$

where D is the depth of penetration in metres and tR is the two-way travel time in seconds.

As shown in Figure 2.4, the GPR antenna is moved along the x-axis on the ground surface from point X_{-N} to X_N. z refers to the depth of the object from the ground surface. The distance between the object and the antenna varies as the antenna is moved horizontally. The reflection signals r_{-N}, r_0 and r_N can be projected orthogonally to the movement axis as shown in Figure 2.4(b). By sequentially connecting the ends of the obtained reflection signals, a hyperbola is formed (Ristic and Petrovacki, 2007). All the points seen on the scan are the amplitude of the reflections from the object. The reflection seen at r_0 or the apex of the hyperbola gives the accurate depth location of the object in the subsurface.

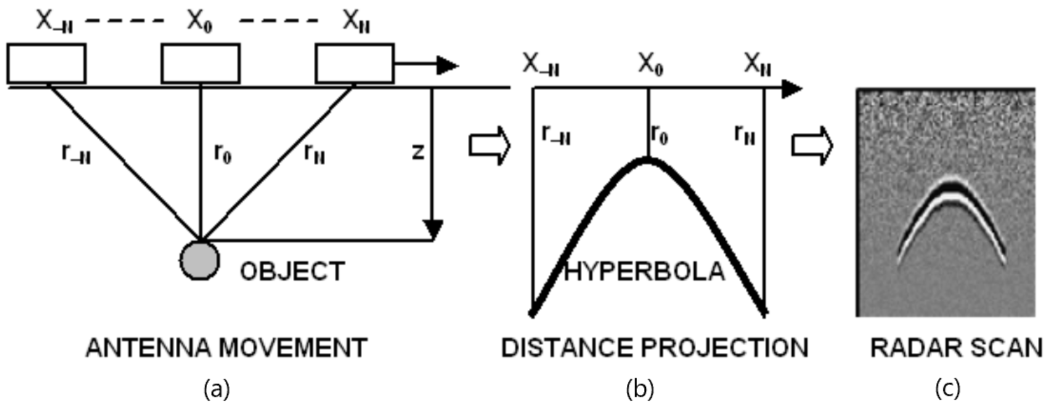


Figure 2.4: Radar scan generation (Ristic and Petrovacki, 2007)

2.2 gprMax

All electromagnetic phenomena can be described by Maxwell's equations. These partial differential equations are solved using the Finite-Difference Time-Domain (FDTD) method (Warren et al., 2015). GprMax is an open-source software that simulates electromagnetic wave propagation. The software works based on discretisation of the space and time domain. The discretised form of Maxwell's equations is then represented using a Yee cell. By assigning the appropriate parameters to the locations of the electromagnetic field components, the equations can be solved.

The software is designed for simulating the Ground Penetrating Radar (GPR) and can be used to model electromagnetic wave propagation in different fields of study. The software is command-line driven and hence does not have a Graphical User Interface (GUI). It is primarily written in Python and Cython and can run in High-Performance Computing (HPC) environments. It runs on Central Processing Unit (CPU) or Graphics Processing Unit (GPU), features the Messaging Passing Interface (MPI) task farm, and can hence run on multiple CPUs/GPUs.

The gprMax software has several features (Warren and Giannopoulos, 2021), among which the following are used for this research.

1. Modelling of soils with realistic dielectric
2. Building of heterogeneous objects
3. Built-in libraries of antenna models

An ASCII file is used to define the model parameters and environment. The input file consists of commands used to model the objects, environment, location of the antenna, etc. The common commands that are used to create an input file are as follows:

Command	Function
#domain	Specifies the physical size of the model
#dx_dy_dz	Defines the discretization steps
#time_window	Defines the total required simulated time window for the GPR trace
#material	Introduces different materials with their parameters
#box	Introduce a rectangle of specific properties into the model
#cylinder	Introduces a cylinder into the model
#hertzian_dipole	Introduces a current source/excitation signal at an electric field location
#src_steps & #rx_steps	Defines the number of steps to move the location of the source
#rx	Specifies the details of a receiver (Rx)

Table 2.1: Command commands used to create the gprMax input file (Warren and Giannopoulos, 2021)

For example, Figure 2.5 depicts the geometry view of a void placed in a homogeneous soil environment. A Hertzian dipole source fed with a Ricker waveform with a frequency of 1 GHz is used to excite the GPR. The current source is provided along the z-axis, and a time period of around 10 ns is provided.

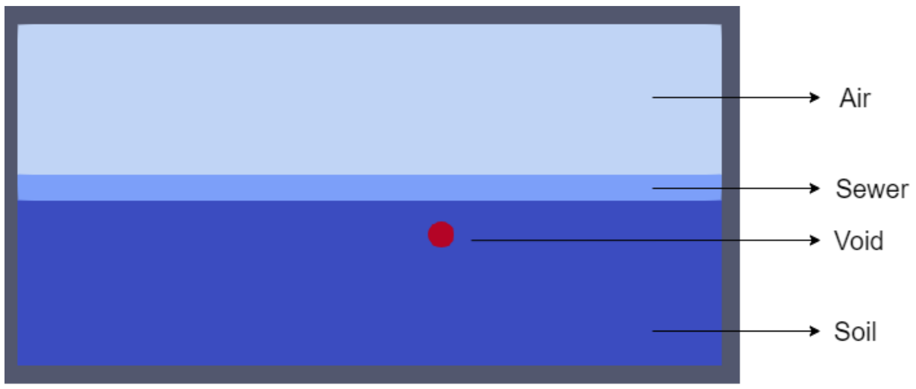


Figure 2.5: Example of a Geometry view of the simulated model

On running the input file created, a time history (Trace) of the electric and magnetic fields along the three-axis is obtained at the receiver. Since the excitation was along the z-axis, the electric field component along the z-axis (i.e.) E_z is elaborated further as shown in Figure 2.6. The initial part of the signal from 0.01 ns to 0.03 ns represents the direct signal from the transmitter to the receiver antenna. The second part of the signal from 0.03 ns to 0.07 ns represents the reflected wave from the void in the subsurface. This can be further explained by a B-Scan, which is a 2-D representation of the model. The B-scan is made up of several traces/A-scans as the antenna is moved along the surface.

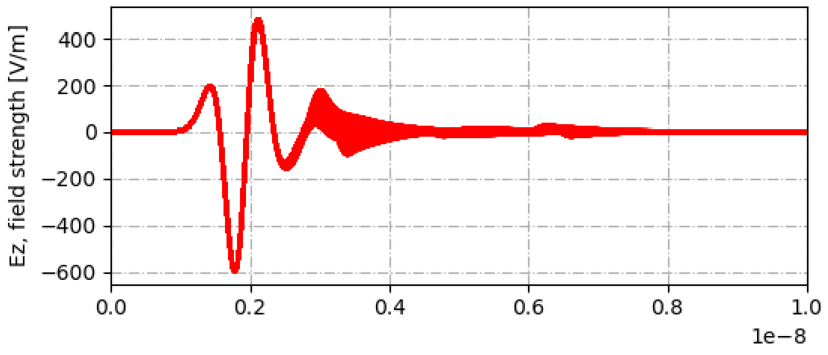


Figure 2.6: A-scan generated by gprMax

Figure 2.7 shows the B-scan of the model along the E_z direction. Compared with the A-scan as seen earlier, the initial direct signal from the transmitter to the receiver antenna is seen from 0.01 ns to 0.03 ns. The reflection from the void is then depicted in the shape of a hyperbola from 0.03 ns onwards.

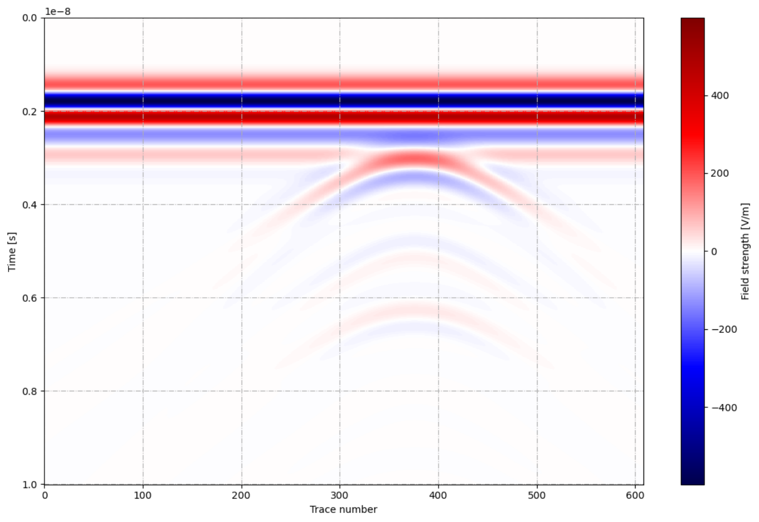


Figure 2.7: B-scan generated by gprMax

2.3 Machine-Learning

2.3.1 Faster RCNN Architecture

The Faster RCNN (Region-based Convolutional Neural Network) combines a Region Proposal Network (RPN) and a CNN. It has a two-stage detection process. The first stage identifies regions of the image where the object might be present. Once the regions of interest are identified, the second stage includes classifying the objects in the region. The architecture of Faster RCNN is as follows:

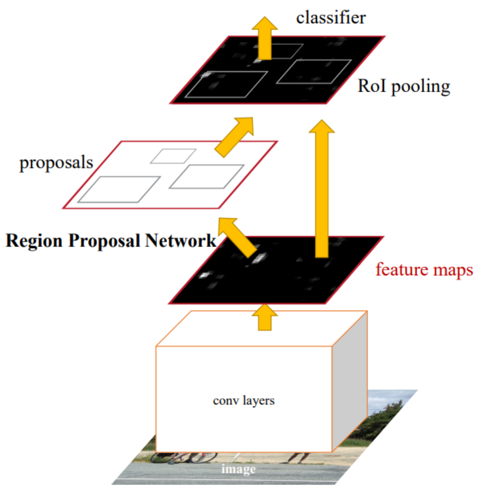


Figure 2.8: Faster RCNN Architecture (Ren et al., 2015)

The input image is fed to the convolution layers of a pre-trained CNN. The output obtained is a feature map that contains information on the object features. The feature maps are then fed to an RPN layer to obtain the object locations to classify the objects as the final step. The input parameters to the RPN layer are the Anchor boxes that are of the form [height, width]. The anchor boxes estimate the bounding boxes in an image based on the size of the objects in the training dataset. With the anchor boxes, the RPN layer produces region proposals with bounding boxes and object score (Ren et al., 2015).

2.3.2 YOLO V2

The second method used for training is the YOLOV2 (You Only Look Once) method. It uses a single-stage detection method and hence does not have any region proposal networks associated with it.

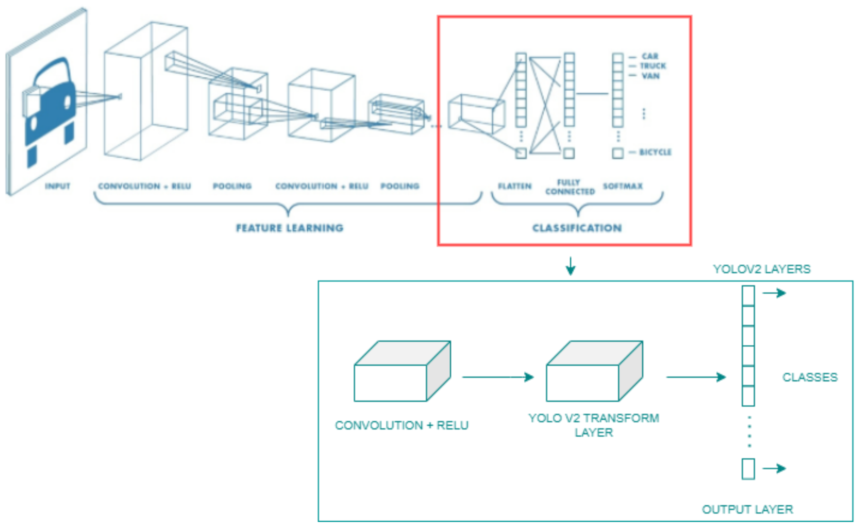


Figure 2.9: YOLO V2 Architecture

The YOLO V2 method replaces the fully connected layers of a CNN with a convolution block and YOLO V2 transform and output layers as seen in Figure 2.9. The transform and output layers use the anchor boxes to predict bounding boxes (Redmon and Farhadi, 2017). At the end of feature extraction by the CNN, the YOLO network applies the anchor boxes on the image to propose possible objects. The architecture predicts three attributes for each anchor box :

1. Intersection over union (IoU) - predicts the amount of overlap between anchor boxes, meaning a possible object location.
2. Anchor box offsets - refines the position of the anchor boxes
3. Class probability - Predicts the class label associated with every anchor box.

Figure 2.10 shows the predefined anchor boxes of classes air and water on the radargram, the overlap of anchor boxes on the possible location of the void, and the final classification after the three attributes are applied.

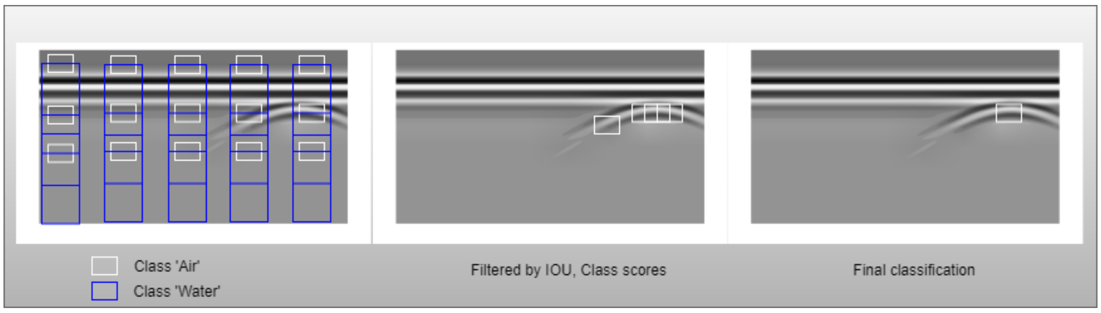


Figure 2.10: Illustration of voids predicted in a radargram using anchor boxes

2.3.3 YOLO V3

The final method is the YOLOV3 that is also a single-stage detector. The YOLOV3 works by making detections at different scales of the network. Similar to the YOLOV2 architecture, the YOLOV3 layers replace the fully connected layers of a pre-trained CNN. The YOLOV3 layers consist of detection heads that are serially connected convolution layer, Batch normalisation, and ReLU layer. Two or more detection heads can be used based on the size of the objects to be detected. The feature extraction network is chosen from one of the CNN layers and is concatenated with the output of the first detection head. This is fed as input to the second detection head. This procedure is repeated, and each concatenated feature map is fed to the next detection head (Redmon and Farhadi, 2018).

The salient feature of YOLOV3 architecture is the concatenation of feature maps from high resolution, partly processed activations, and the low resolution fully processed features at the end of the extractor.

Each detection head uses a set of anchor boxes that work on the same principle as described in Figure 2.10 and the output detector is obtained at the end of the last detection head.

2.3.4 K-means Clustering

Image segmentation is the method of classifying an image into different groups. K-means clustering is a type of unsupervised machine learning that can extract the area of interest from the background. A target number, ’K,’ is defined, referring to the number of centroids needed in the dataset. The number of centroids will produce the same number of clusters. A centroid is the imaginary or real location representing the center of the cluster. When the algorithm is applied to the image, every pixel is allocated to each cluster while keeping the centroid as small as possible. The ’means’ in the K-means refers to averaging of the data; that is, finding the centroid (Michael J Garbade, 2021).

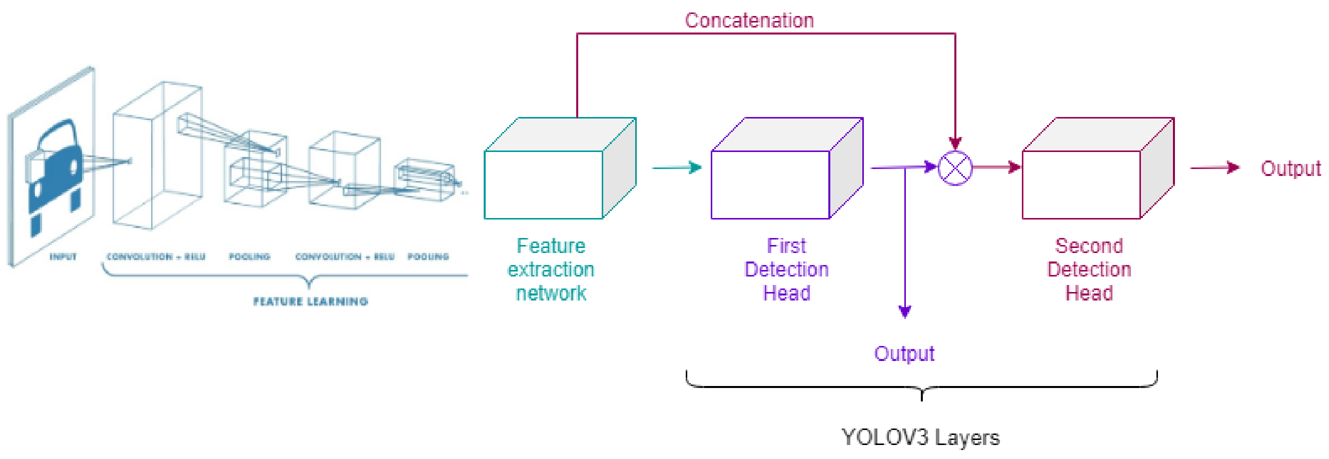


Figure 2.11: YOLO V3 Architecture

2.4 Training Options

The following options are used to train the detectors:

1. Solvername - Optimizer algorithms used to train the network. Stochastic Gradient Descent with Momentum (SGDM) optimizer helps accelerate gradients during network training for faster convergence.
2. Epoch - corresponds to the number of times the complete data is passed through the network.
3. MinibatchSize - For faster training, the dataset is divided into subsets called Mini batches. The Mini batch size specifies the number of images in one batch.
4. Iteration - Number corresponding to the number of times a minibatch is passed through the network.
5. Initial learning rate - Small positive value usually between 0.0 and 1.0. The smaller the learning rate, the slower the training progress. If the learning rate is too high, the training might reach a suboptimal result.

2.5 Summary

This chapter focuses on the essential tools that are used for this project. Firstly, the basics of Ground Penetrating Radar and generation of radargrams are explained. Then, simulation of the radargrams using the gprMax software is discussed. The background of the three different ML algorithms is briefly described, followed by the parameters used for training the data. The basis of Machine-learning and deep learning is briefed in Appendix B, followed by the working of a CNN architecture.

The following chapter concentrates on analyzing the different ML algorithms giving a motive for the choice of ML algorithms in this research.

3 Object-detection method Analysis

There has been extensive research in ground surface GPR surveying and automatic detection of objects which is explained briefly in the following section. Different types of ML and Artificial Neural Networks (ANN) that can be used to detect the objects underground are summarised and obtained as shown in Figure 3.1 (Travassos et al., 2018). The different state-of-the-art methods used over the years for detection of underground objects using GPR are as shown in Figure 3.1. The graph is plotted with the number of papers published in the ML topic versus the year of publication.

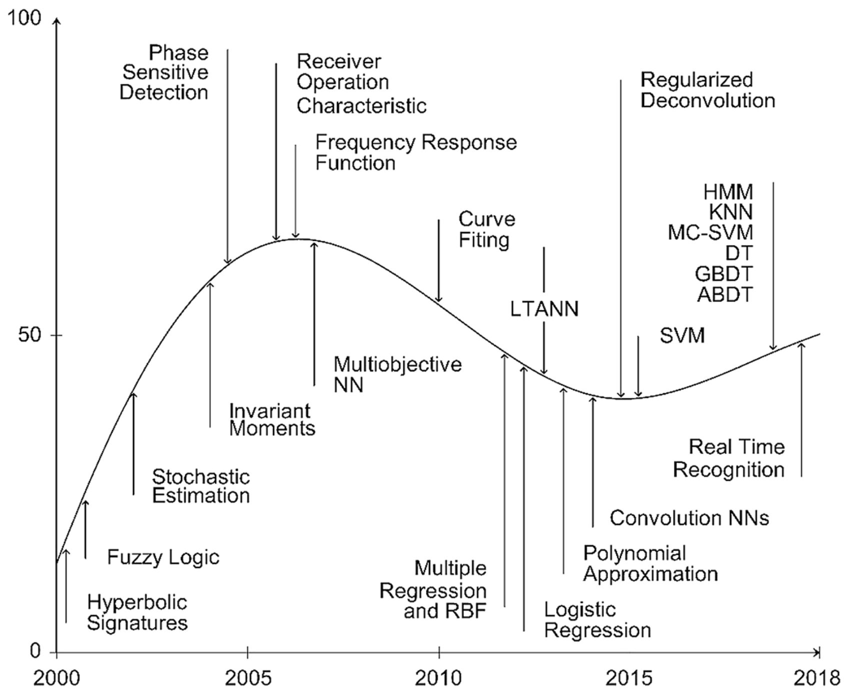


Figure 3.1: State of the art techniques for object detection (Travassos et al., 2018)

GPR is commonly used in Civil engineering practices, and hence there are three main types of detection that most research focused on:

1. Detection of buried objects, including landmines, pipelines, and archaeological objects.
2. Investigating the presence of reinforced concrete or steel structures in bridges, dams, and other civil engineering structures.
3. Inspection of soil firmness and checking for hollow cavities or voids.

In the analysis of the different methods available, the criteria for selection are defined, and suitable models are chosen.

3.1 Introduction

The objective of this research is to choose a model that can be used to detect subsurface voids. The voids can be composed of either Air or water, and the reflections produced due to the material varies as seen in Figure 3.2. Water with a higher permittivity value than Air and sand have more reflections in a radargram when compared to Air. This significant difference is used to identify voids of different compositions.

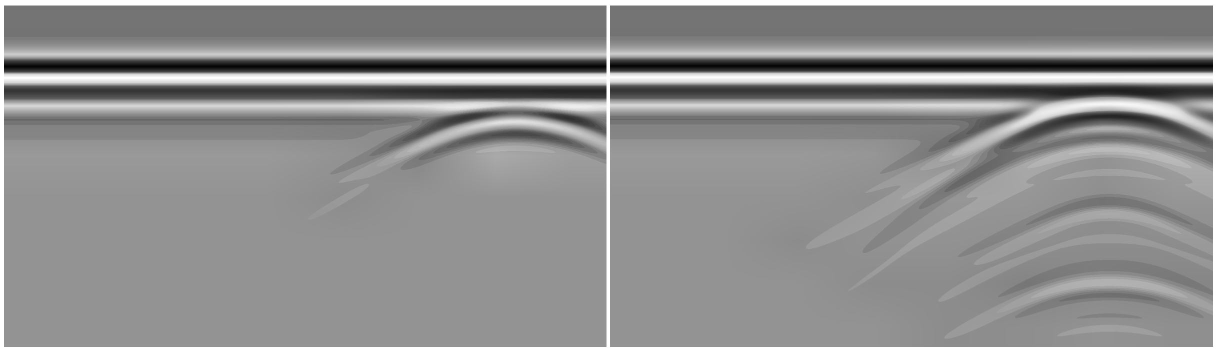

(a) Air void **(b)** Water void

Figure 3.2: Subsurface voids of different compositions

Hence, in addition to hyperbola recognition, a Machine-Learning model capable of detecting the difference in reflections should be chosen. To choose a respective machine learning model, the following criteria are followed:

1. Knowledge of the dataset: The data that has to be identified and objects to be classified are observed.
2. Accuracy and Speed: High Accuracy of the detector in identifying the void is required. Speed of detection is not considered in this case since real-time detection is not followed.
3. Parameters: The number of parameters required to train the algorithm should not be high to not have a very complex model.

As mentioned earlier, the vital visualisation factor in the dataset is the difference in reflections. Most of the available research focuses on ground surface GPR and the detection of buried objects. The possible methods are analysed, and the priority is to choose a model that has been used in underground object detection with good results. This model is then applied to void detection to check for accuracy.

The following literature is the current research available. The criteria for selection is applied to the following methods to choose the best among them.

In (Núñez-Nieto et al., 2014), the aim is to detect landmines. Hence, the dataset comprises radargrams of buried landmines and other non-explosive materials to have false targets to train the Neural network model. They use a combination of logistic regression and a machine-learning algorithm to train the model. The results only give a zero or a one as an output since logistic regression is used to find the actual targets. It was concluded that the model also detects the false targets and could be improved using a larger dataset.

Singh and Nene (2013) also uses GPR to detect Improvised explosive devices. The aim of the research is to detect the devices and to estimate the location and depth of the device. They use a combination of Image processing techniques and neural networks in MATLAB, and the designed model took 42 minutes and 8 seconds to detect an object. A dummy classifier was used in this case and can be improved by implementing ANN techniques and parallel computing.

In (Al-Nuaimy et al., 2000), the radargram interpretation is based on a combination of Neural network, signal, and image processing techniques to obtain a high-resolution image of the radargram. The Hough transform method is used to locate and identify the hyperbolas, which refer to the buried objects. The Finite-difference time-domain (FDTD) method is used to simulate multiple radargrams, and a nine-layer Deep Neural Network (DNN) model is implemented to extract the feature maps containing hyperbolas (Sonoda and Kimoto, 2018). They resulted in only 80 % accuracy due to the limited number of DNN layers.

The disadvantage of classic object detection methods like the Hough Transform, Viola-Jones detector, and template matching is that they need several parameters for discretization or setup of templates and hence involve a high computational complexity. Therefore, the below-mentioned family of Machine-Learning algorithms that are used explicitly for object detection is explored.

3.2 Machine Learning models

A Support Vector Machine (SVM) classifier trained on Histogram of Oriented Gradients (HOG) is used to detect underground objects from a radargram (Noreen and Khan, 2017). A detection rate of 75% is obtained, which can be improved with a more extensive dataset of over 1000 images, by pre-processing the data and by applying image processing before training the algorithm. A linear SVM classifier is used to distinguish between hyperbolic and non-hyperbolic features in the radargram (Skartados et al., 2018). A 72% accuracy is obtained in detecting the hyperbolic segments that were extracted using the HOG method.

Pham and Lefèvre (2018) proposed using the Faster-RCNN method to detect the hyperbola reflections from the radargrams. The results show that the method has significant improvements over other detectors like the HOG and Haar-like. A training dataset is first created using both actual and simulated radargrams, the CNN is pre-trained, and the trained model is fine-tuned by applying it to the test radargrams. Lei et al. (2019) used the Faster-RCNN method to detect hyperbolas and used image processing methods like the Double Cluster Seeking Estimate (DCSE) algorithm and Column-based Transverse Filter Points (CTFP) to extract the hyperbolic signatures further. An accuracy of 92% is obtained in hyperbola recognition.

Dou et al. (2017) applies a thresholding method to separate the regions of interest from the background. The Column Connection Clustering (C3) algorithm is then applied to separate the regions of interest from each other, and the model can be trained to detect the hyperbolas. This method can be used for real-time images since by using the thresholding method, all the noise in the background can be removed. It takes 0.73 s to fit one hyperbola, and though it can be used for real-time data, in the case of many anomalies, the algorithm will take much time to fit all the hyperbolas.

Li et al. (2020) proposes the YOLO V3 recognition method where the recognition speed of 12 frames per second per image is obtained. YOLO V3 is as accurate as faster RCNN and also uses the K-means algorithm to improve the accuracy of the bounding boxes. Non Maximum Suppression (NMS) method is also used to select the most appropriate bounding box in the case where there are several bounding boxes for one parabolic reflection. Pham et al. (2020) works with object detection from aerial and satellite remote sensing images. A YOLO-fine algorithm is developed based on the YOLOV3 architecture and is used to detect smaller objects. An accuracy of only 48.71% was achieved, which was still more than the accuracy rates for detecting smaller objects by YOLO, YOLOV2, and YOLOV3.

With the available literature, it is clear that current research only focuses on identifying hyperbolas to detect subsurface objects. Even though the reflection of a void on the radargram will be a hyperbola, there will be more or fewer reflections depending on the void material. The method of void detection used in this research should classify voids based on their material properties.

The second difference that could be noted is the number of traces required to detect a void vs. a more significant object. Núñez-Nieto et al. (2014) for example, in real-time detection, needed a window of 15 traces which is big enough to fit one landmine and also small enough not to have an overlap of different landmines. To detect smaller voids, a smaller number of traces might be sufficient in real-time detection by the in-pipe GPR. The objective of this research should be to choose an algorithm for void detection and classification to create a symbiotic relationship between the in-pipe GPR and automatic void detection.

3.2.1 Analysis

As seen in the available literature, several ML algorithms give good accuracy in the detection of hyperbolas. Standard object detection practices involve i) selecting the Region of Interest (ROI) where the object is present, ii) extracting the object features, and iii) feeding the features to an ML algorithm. Though supervised ML algorithms like the SVM have shown good detection accuracy, they always require feature extraction and additional pre-processing before classification. This results in further computational complexity. SVM works by finding an optimal separation line called the hyperplane that classifies the data into two or more classes. Though SVM used to be the state of the art classification solution, the introduction of Neural networks made classification problems easier and faster to compute.

Criteria/Decision	SVM	C3 Algorithm	RCNN	YOLO
Criteria				
Feature extraction required	x	x		
Dataset directly fed to model			x	x
Accuracy > 90%		x	x	x
No. of Parameters	Low	High	High	High
Decision				
Accept method			x	x
Decline method	x	x		

Table 3.1: Decision Table

When determining the ML algorithm, computational complexity and time taken to detect objects need to be analysed to compare the different detectors. The following points are considered for choosing an appropriate ML method:

1. Number of Parameters - This factor is crucial in choosing a model since the tuning of hyper-parameters improves the system's overall accuracy. More parameters to tune give rise to more combinations of parameters that could be obtained to improve the accuracy.
   - SVM has only two parameters C and Gamma.
   - CNN algorithms depend on the number of hidden layers, epoch, learning rate and Batch size.
2. Size of Input data
   - SVM requires a lesser amount of data in comparison to CNN.
   - CNN requires a large amount of data for training. The more data fed into the layers, the more accurate the system is since it improves the detection rate and reduces errors.
3. Preprocessing Data
   - An additional step to extract features from the dataset is required before applying the SVM algorithm.
   - Feature extraction is automatically done in the CNN algorithm. The convolution layers, along with Pooling, perform feature extraction on the input data.

As seen in Table 3.1, the neural network methods RCNN and YOLO are more suitable than the other methods. Though the number of parameters is low in the SVM method, the accuracy is only around 75% (Noreen and Khan, 2017). Since accuracy is the main criteria in detecting hyperbolic reflections, the SVM and C3 methods are not used. The thesis hence focuses on the RCNN and YOLO methods.

3.3 Transfer Learning

As seen in Chapter 2: Faster-RCNN and YOLO network, a CNN is required in the architecture. The network can either be trained from scratch or a pre-trained network can be used. Training a CNN from scratch takes up days or even weeks to be trained. It also requires a very large dataset in the range of 10000 images at least. Therefore, by using transfer learning, the knowledge of an already trained machine learning model can be applied to other models. This reduces the complexity of choosing and fine tuning the number of hidden layers and pooling. For training the models with the simulated radargram datasets, resnet-50 is chosen as the pre-trained network. Resnet-50 is a 50 layer network architecture that is commonly used for object detection problems. It is chosen to not complicate the model with a lot of additional layers and to reduce complexity in computation.

3.4 Evaluation metric

After the Machine learning models chosen are trained, the evaluation metrics commonly used to test the performance of the detectors are employed.

Confusion matrix is a performance measurement that contains a table with four different combinations of the predicted and the actual classes.

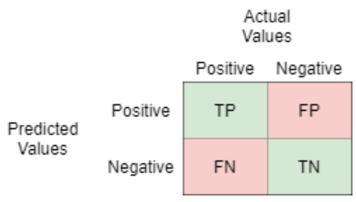


Figure 3.3: Confusion Matrix

As seen in the matrix, the True Positives (TP) are the correctly predicted object classes. The True Negatives (TN) are the correctly predicted negative values, meaning that the object was not detected since there was no object in the image. The False Positives (FP) are cases where the object class was detected when the object was not present in the image. False negatives (FN) are cases where the object class was not detected when the object was present in the image.

Precision gives the positive predictive value among all the values. It can be represented mathematically as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

Recall denotes the sensitivity of the detector in detecting true positives.

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

F1 Score is the weighted average of Precision and Recall.

$$F1Score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{3.3}$$

3.5 Summary

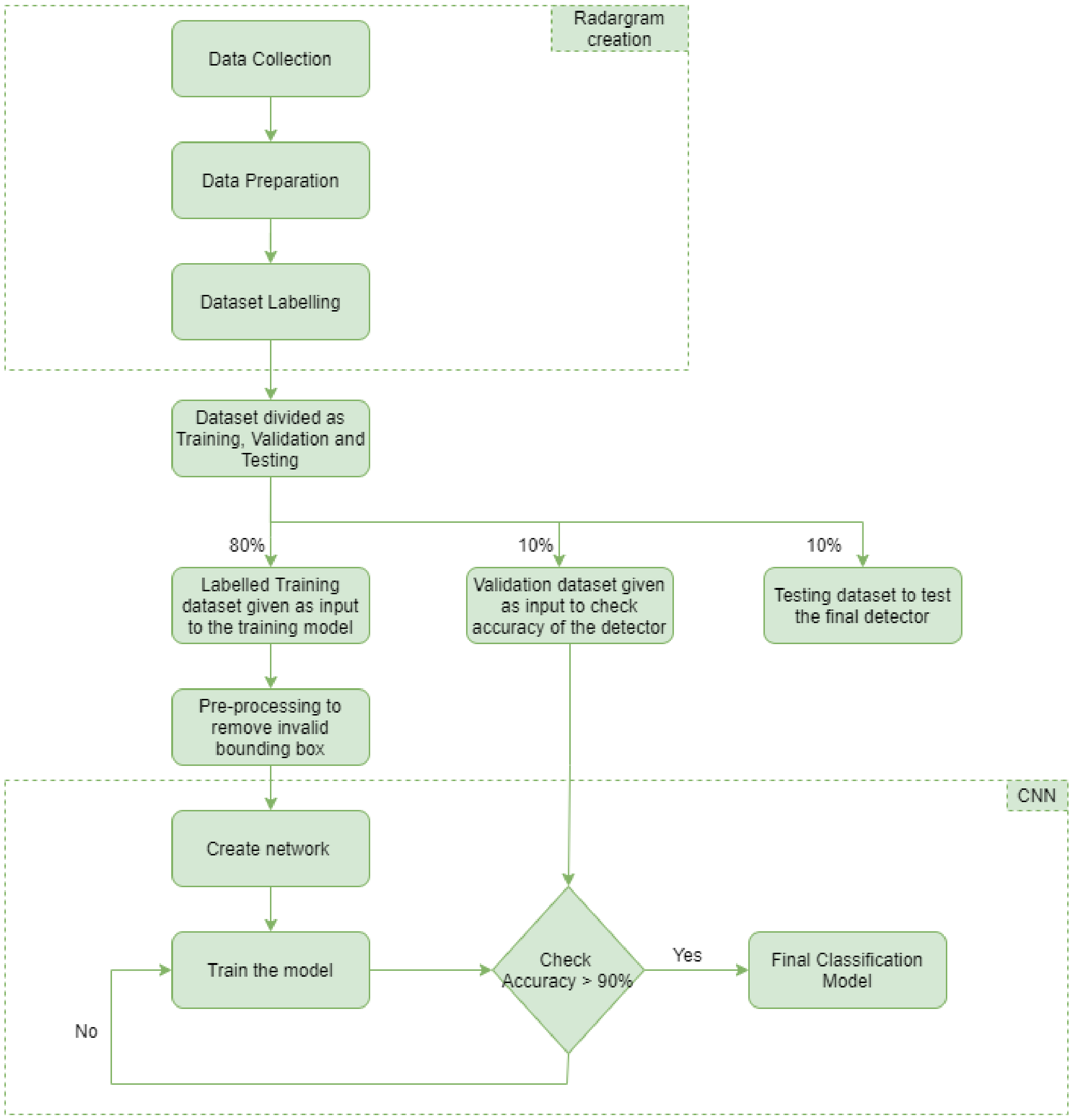
In summary, when it comes to limited data, SVM is the best choice, but neural networks perform better in object detection when opting for accuracy or performance. Though the training time is longer for CNN, it results in significant improvement in accuracy, which is the deciding factor in this case.

With improved CNN and Deep learning methods, this thesis focuses more on the Two-stage detection framework (Faster-RCNN) and the one-stage detectors (YOLO). In this assignment, gprMax software is used to simulate multiple synthetic radargrams based on the requirement.

4 Configuration and Calibration

4.1 Introduction

By using Machine Learning, the object detection algorithm can be trained to classify the voids based on the material Air, Water, or Other. This can be done by creating a dataset containing different material voids and training the algorithm accurately. As shown in Figure 4.1, the main steps for void detection involve creating the radargram dataset, pre-processing, and using neural network algorithms to detect and classify the voids. These parts are elaborated further in the following sections.



Radargram creation
Data Collection
Data Preparation
Dataset Labelling
Dataset divided as Training, Validation and Testing
80%
10%
10%
Labelled Training dataset given as input to the training model
Validation dataset given as input to check accuracy of the detector
Testing dataset to test the final detector
Pre-processing to remove invalid bounding box
CNN
Create network
Train the model
Check Accuracy > 90%
Yes
No
Final Classification Model

Figure 4.1: Methodology followed for void detection

4.2 Data Collection

The first step to a Machine learning algorithm is to prepare the dataset. Due to the lack of real-time radargrams that have a ground truth depicting the different void compositions, synthetic data is created using the gprMax software. All the voids in the radagram are simulated in different depths and locations throughout the dataset. The voids simulated for this research are primarily spherical and differ in size in the dataset. Different types of datasets are created to give more range to the training.

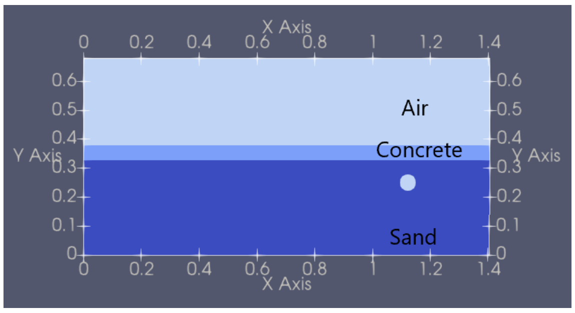
1. Voids of composition Air, Water, and other random material are simulated.
2. Dataset containing a single void that can be either Air, Water, or Other material in a homogeneous soil environment. This is further varied by introducing different soil compositions.
3. During infiltration/exfiltration of water, the dry sand around the sewer pipeline has areas of saturated sand mixed with it. The sand environment is no longer homogeneous in this case. A dataset containing a single void in a heterogeneous soil environment where the soil layers around the pipeline are disturbed is simulated.
4. Dataset containing multiple voids with random materials and sizes is simulated.

The objects simulated using the gprMax software are defined by their properties like relative permittivity (ϵ_r), conductivity (S/m) (σ) , relative permeability (μ_r), and magnetic loss (Ω/m) (σ_*). The electromagnetic wave reflections from the objects depend on the mentioned properties. Therefore, for the preparation of the dataset, the following materials as summarised in Table 4.1 are used. The relative permeability and magnetic loss are considered as 1 and 0 for all the materials, respectively.

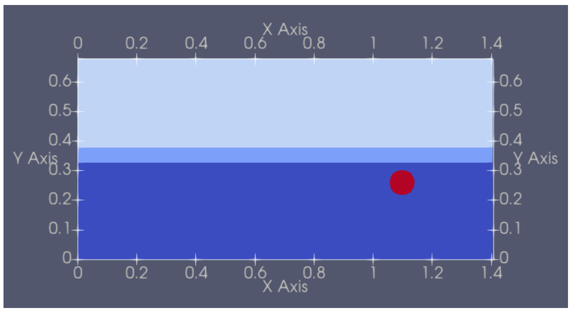
Material	Permittivity (ϵ_r)	Conductivity (S/m) (σ)
Dry sand	3	0.01
Saturated sand	19	0.01
Concrete	4.5	0.01
Air	1	0
Water	81.0	0.5
Wood	2.0	0
Brick	3.26	0
Clay	15	0.01

Table 4.1: List of Materials used to simulate the radargrams and their properties

As seen in Figure 4.2, voids of different materials and sizes are simulated in varying depths in the environment. The geometry of the environment is kept consistent in all the datasets. The domain size is simulated as 1407 x 679 x 2 mm. The environment consists of Air depicting the free space, Concrete for the sewer pipeline, and Soil around the sewer where the voids are present. The sewer pipe thickness is kept constant at 50 mm in all the radargrams, and the depth of the sand environment is around 329 mm. Different void compositions are simulated. As shown in the figure below, Air and Water voids shown as blue and red circular objects are simulated in the environment.



(a) Air void

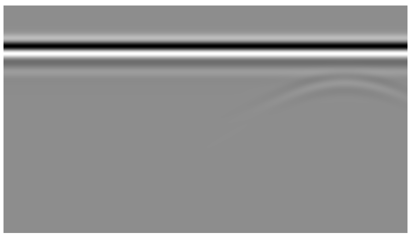


(b) Water void

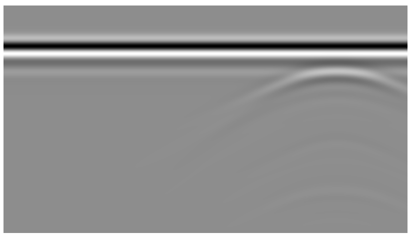
Figure 4.2: Simulated Sewer environment as visualised in the Paraview software (Ahrens et al., 2005)

GPR Antenna can be operated in varying frequencies. Low frequencies are used in places where a higher depth in penetration is required, whereas High frequencies are opted for objects of lower depth and better resolution. For the simulation, a Hertzian dipole fed with a source of 1 GHz is used to simulate the antenna. A time window of 10 ns is chosen to allow enough time for the wave propagation through the medium to be reflected from the void and received by the antenna. The files are generated using the commands as given in Table 2.1.

The gprMax simulations are performed using the High-Performance Computing (HPC) cluster of the University of Twente (HPC, 2021). Since the simulation of radargrams is high in computation, the NVIDIA CUDA programming environment is used to offload the workload across multiple GPU'S. The output files are obtained in the form of HDF5 files that are converted to 1920x1080 Portable Network Graphics (PNG) using matlab (Warren, 2018). The corresponding Radargrams of the environment in Figure 4.2 are as shown below:



(a) Air void



(b) Water void

Figure 4.3: Radargrams of the corresponding environment as visualised in Figure 4.2

4.3 Data Preparation

The obtained radargrams are prepared for the Machine-Learning algorithms. Image enhancement and time zero correction are applied to the radargrams to process the GPR data. The radargrams are then labelled and visualised. The dataset is split into three - 80% of the data for training the model, 10% of the data for evaluation of the trained model, and 10% of the data for testing the detection accuracy.

4.3.1 Contrast Enhancement

Machine-Learning algorithms focus on feature extraction, and hence the features essential to distinguish between the different voids need to be enhanced. Zooming in on the radargrams in Figure 4.3 show slightly more ripples in the water void than in the air void. Electromagnetic waves travel faster in the material of low permittivity and vice versa. Since the relative permittivity of water is 81 and sand is 3, the difference in permittivity is higher enabling the waves to travel the slowest when detecting water voids. This produces more reflections in the radargram.

These additional reflections produced by the water void is used as the main feature difference that helps in differentiating between Air and Water for the machine learning algorithms.

A histogram is a graphical representation of the number of pixels in an intensity scale that ranges from 0 to 255. By equalising the histogram of an image, areas of lower contrast gain a higher contrast, thereby improving the visibility of the pixels. Hence, to improve the visibility of the reflections, the Contrastive Limited Adaptive Histogram Equalization (CLAHE) method is used.

CLAHE (Zuiderveld, 1994) operates on small regions of the image, called tiles. It calculates the contrast transform function for each tile individually. The contrast in each tile is enhanced to a given distribution value, and the neighbouring tiles are combined to form the entire image.

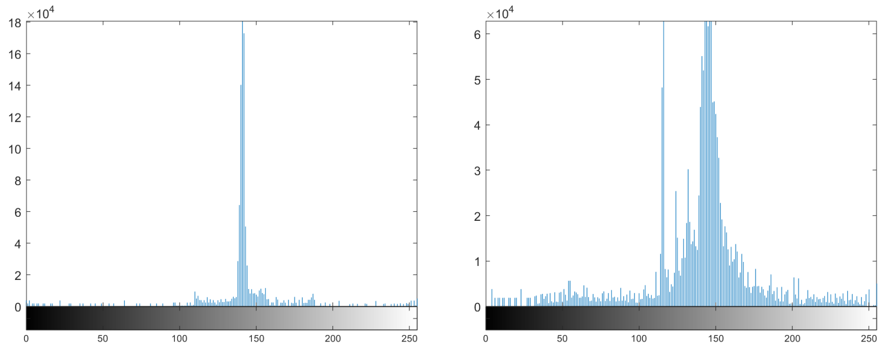


Figure 4.4: Before and After Histogram normalisation of the radargram containing the water void

CLAHE method is applied to the dataset using the MATLAB function $'adapthiseq'$. Figure 4.4 shows the histogram of the radargram containing the water void before and after the CLAHE method is applied. The figure shows that the histogram peaks are clipped, and the contrast is enhanced in the other areas. The peak in the centre of the histogram represent the reflections due to the sewer pipe. By using the CLAHE method, the contrast is enhanced for the void reflections which are represented by the additional high peaks.

The parameters of the CLAHE method that are tuned for this application are the cliplimit and the distribution function. The cliplimit is the factor that prevents oversaturation of the pixels in the homogenous areas. It refers to the histogram peaks where there is more number of pixels in the same gray range. A 'uniform' distribution function is chosen, which creates a flat histogram. The CLAHE method divides the image into 8x8 tiles to perform the contrast enhancement. Figure 4.5 and Figure 4.6 below shows the radargrams in Figure 4.3 after contrast enhancement.

The reflections are more visible than before contrast enhancement, and it is comparatively easier to now distinguish between the two voids.

4.3.2 Dataset labelling

Data labelling is an essential step in supervised Machine-Learning in which a ground truth is given to the dataset to facilitate detection between the different voids. The voids are classified into 'Air', 'Water', and 'Other' in the void detection problem. The 'Other' label consists of all other materials of objects that are not Air or Water. For simulation purposes, materials like wet sand and brick are used to create 'Other' objects.

The .in file used to simulate the radargrams consist of antenna location, the geometry of the simulated environment, void location, and material. With knowledge of the exact location of the voids and the time zero value, the data from the input file can be interpolated into labels for the radargram (Delft, 2019).

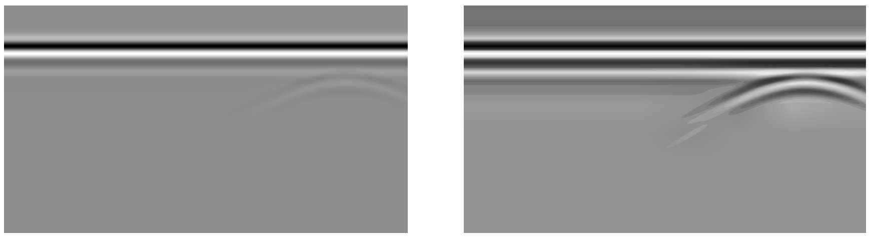


Figure 4.5: Before and After Contrast enhancement of Air void

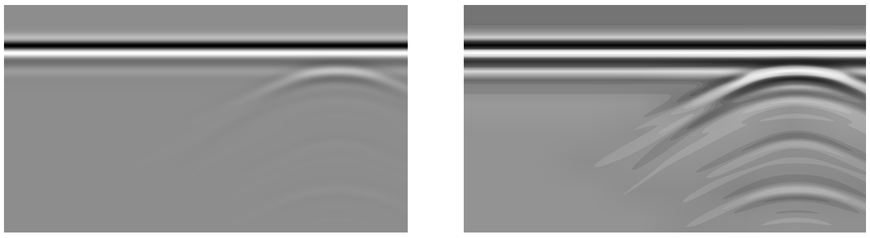


Figure 4.6: Before and After Contrast enhancement of Water void

Time Zero Correction

A Hertzian dipole antenna fed with a ricker waveform of 1 GHz is simulated to transmit pulses of electromagnetic waves. The direct wave from the transmitter to the receiver antenna is first recorded. After several tenths of a nanosecond, the second wave that corresponds to the reflected wave from the object is recorded as seen in Figure 4.7. In processing GPR data, the first thing to do is return the signal to the actual position (i.e.) the sewer line due to the in-pipe GPR. Since the B-scan of the GPR data is plotted as Time (y-axis) vs. trace number (x-axis) as shown in Figure 2.7, the time interval between the direct wave and the reflected wave has to be reduced. The Time-zero correction process is performed to eliminate the mentioned time intervals (Maruddani and Sandi, 2019).

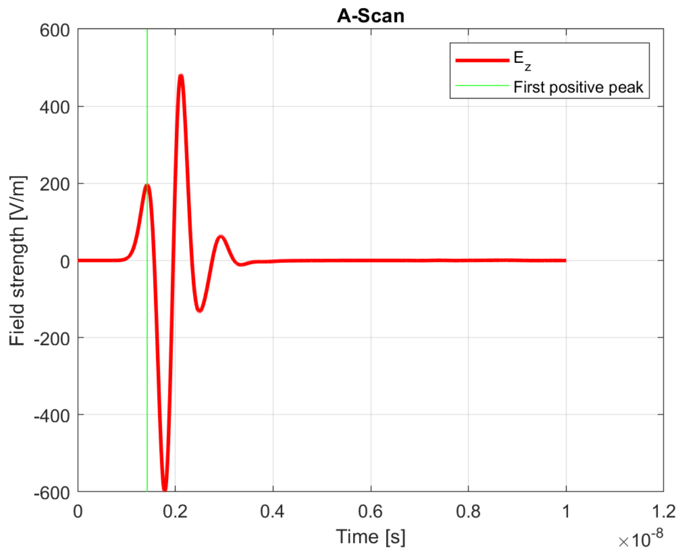


Figure 4.7: A-Scan of the radargram with the green line denoting the first positive peak

This step is essential for conducting accurate shallow depth measurements since the free space between the sewer surface and the antenna can be significantly higher than the distance from the surface to the void. The Time zero is set at the first positive peak of the direct wave as shown in Figure 4.7. This point is the first positive amplitude of the direct wave and is hence easy to identify.

The radargram is made up of several traces, and the antenna location varies. Hence the time location of the first peak of the direct wave will vary in every trace. Therefore, the Time zero value is calculated for every trace, and the mean of the values gives the time zero of the radargram.

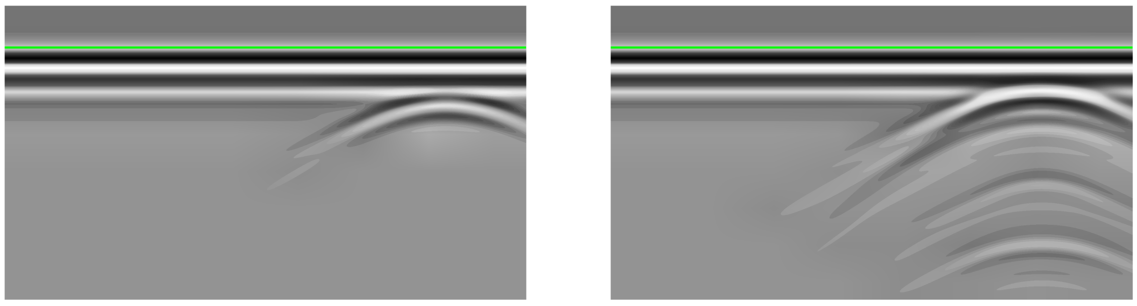

(a) Air void **(b)** Water void

Figure 4.8: Green line depicting Time zero in the radargram

The horizontal direction of the radargram represents the movement of the antenna over the surface in every trace. In contrast, the vertical direction represents the time taken for the reflected wave to reach the antenna. The horizontal displacement and the time taken vertically are calculated to obtain the labels based on the input file. The radargram obtained is a 1920x1080 two-dimensional PNG image, and hence the labels are also described in two dimensions. The image resolution is hence 1920 and 1080 labelled as x_{res} and y_{res} respectively. From the input file, the position and size of the void in cartesian coordinates can be obtained in the form (x, y, r) representing the centre (x,y) of the cylinder and radius r.

Figure 4.9 shows the cross section of the cylindrical sewer pipeline. The sewer is of thickness w_2. The GPR transmitter (red circle) and receiver (green circle) are placed inside the pipe. Due to the GPR system carrying the transmitter and receiver unit, there is a vertical free space between the system and the sewer wall denoted as w_1. A void is simulated at a distance w_3 from the sewer line. By rolling out the cylindrical topology for a better idea of the coordinates, Figure 4.10 is obtained.

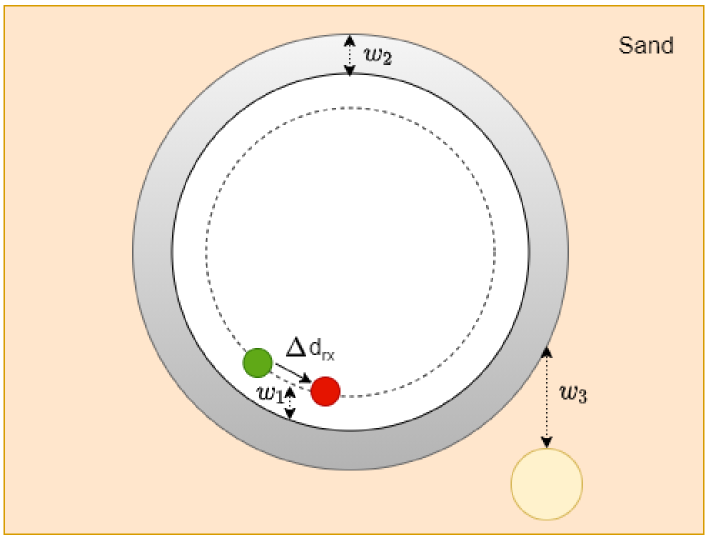


Figure 4.9: Cross section of the sewer

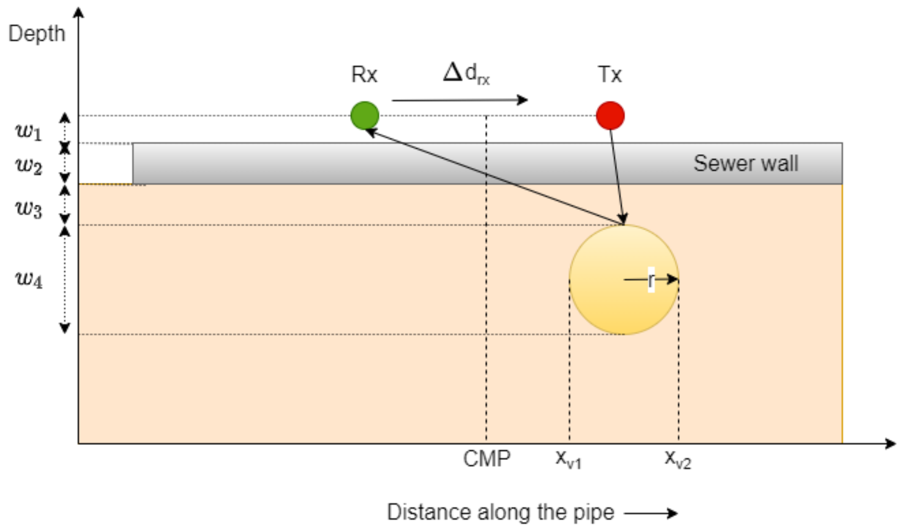


Figure 4.10: Geometry of the void and sewer line in cartesian coordinates

As shown in Figure 4.10, to calculate the two horizontal points of the label, the endpoints of the diameter of the void x_{v1}, x_{V2} are calculated from the cylinder coordinates. In addition to this, the common midpoint (CMP) is also calculated. In Geophysics, the CMP is defined as the point on the surface halfway between the source and receiver that is shared by numerous source-receiver pairs (Schlumberger, 2021). The receiver antenna is moved along the ground surface in steps of 2 μm; hence the change in position of the receiver is calculated as Δ_{rx}. Hence, the horizontal coordinates of the label (l_{x1}, l_{x2}) as seen in Figure 4.11 are calculated by scaling the location of the void to the measured distance and the horizontal resolution of the radargram.

$$l_{x1} = x_{res} * (x_{v1} - CMP)/\Delta_{rx} \tag{4.1}$$

$$l_{x2} = x_{res} * (x_{v2} - CMP)/\Delta_{rx} \tag{4.2}$$

The vertical coordinates are calculated using the Time-zero correction (TZC) value, the two-way travel time (tR), and the time window for simulation. tR can be calculated from the general formula for calculating the depth of penetration.

$$D = \frac{tR \times v}{2} \tag{4.3}$$

The input file contains the dielectric values of all the materials in the radargram as mentioned in Table 4.1. With the dielectric value of the material, the velocity of the wave through the material can be calculated using the following formula.

$$v_m = \frac{c}{\sqrt{\epsilon_r}} \tag{4.4}$$

w in Figure 4.10 indicates the vertical distance between the different materials. The distance between the antenna and the sewer pipe, sewer line thickness, sewer pipe to the void, and height of the void are represented as w_1, w_2, w_3 and w_4 respectively. The depth of penetration in each material can then be calculated by dividing the thickness/vertical distance between the material and the velocity of propagation (v_m). The sum of propagation of the wave in all layers multiplied by two will give the two-way travel time (tR).

$$D_{tm} = \frac{\omega_m}{v_m} \tag{4.5}$$

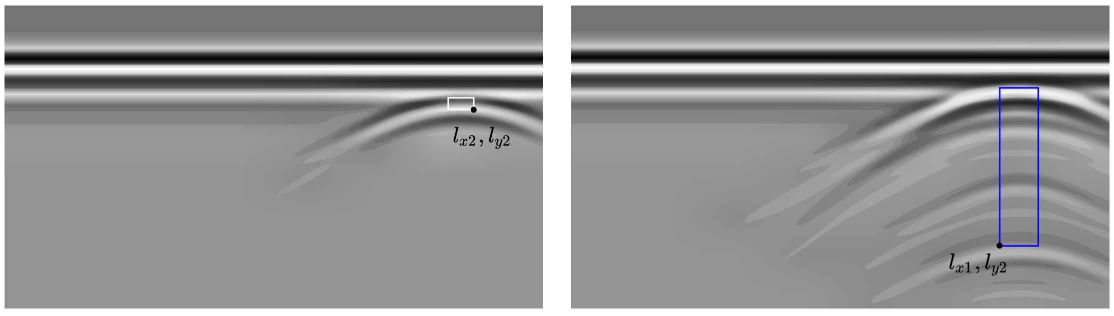
$$tR = 2 * \sum_{i=1}^{n} D_{tm}(i) \tag{4.6}$$

Knowing the values of the TZC, tR, the time window of simulation (Δt) and the image resolution in the vertical axis (y_{res}), the vertical coordinates of the label (l_{y1}, l_{y2}) can be obtained.

$$l_{y1} = y_{res} * (tR_{n-1}/\Delta t) + TZC \tag{4.7}$$

$$l_{y2} = y_{res} * (tR_n/\Delta t) + TZC \tag{4.8}$$

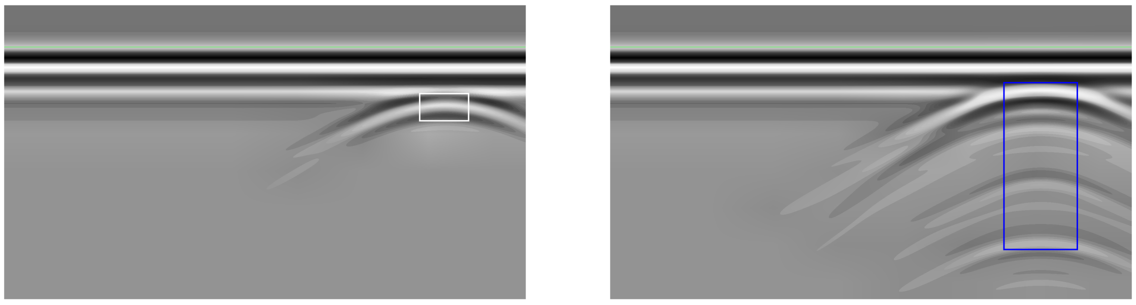
The vertical and horizontal coordinates give an estimate of the location of the hyperbolic reflection in the radargram. As seen in Figure 4.11, the reflections in the vertical direction vary depending on the contrast in permittivity of the material with the environment.



(a) Air void **(b)** Water void

Figure 4.11: Radargrams with bounding box before padding

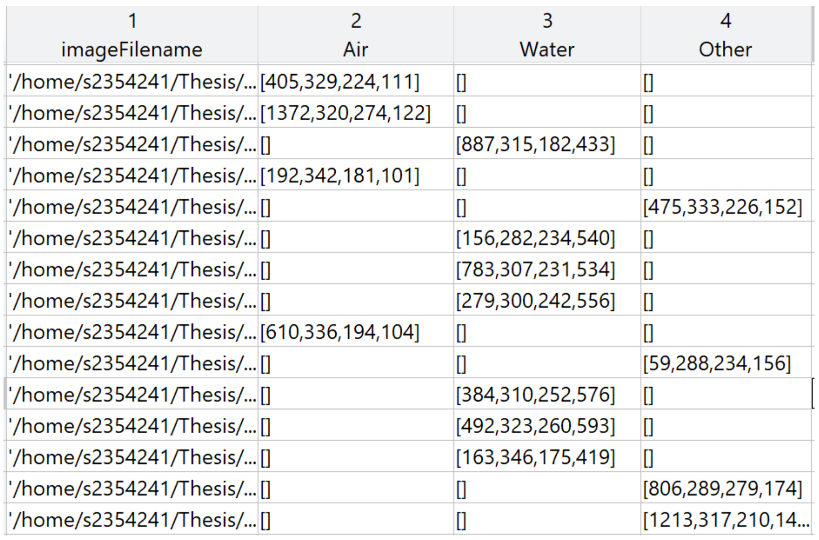
The current position of the label gives a reasonable estimation of the hyperbolic reflection, but few reflections still fall outside the box. Hence, padding is added to the bounding box to account for the reflections along the vertical axis. Horizontal padding is added to increase the width of the bounding box since the size of the hyperbolic reflection is bigger than the actual void size as shown in Figure 4.12. In addition to this, based on the type of void material, the bounding box is labelled as 'Air', 'Water', and 'Other'. This helps in better object recognition during training since the machine learning algorithms can differentiate between the different void compositions.



(a) Air void **(b)** Water void

Figure 4.12: Radargrams with bounding box after padding

For training the dataset with the algorithms, the radargrams are labelled and specified as a table with columns representing the different classes to be detected. The first column of the table contains the file path of the radargram. The remaining columns are cell vectors that contain n-by-4 matrices representing a single object class. The term 'n' represents the number of voids in an image. The columns contain 4-element double arrays representing the bounding boxes in the format [x,y, width, height]. The coordinates specify the upper-left corner location and size of the bounding box in the corresponding image.



1 imageFilename	2 Air	3 Water	4 Other
'/home/s2354241/Thesis/...	[405,329,224,111]	[]	[]
'/home/s2354241/Thesis/...	[1372,320,274,122]	[]	[]
'/home/s2354241/Thesis/...	[]	[887,315,182,433]	[]
'/home/s2354241/Thesis/...	[192,342,181,101]	[]	[]
'/home/s2354241/Thesis/...	[]	[]	[475,333,226,152]
'/home/s2354241/Thesis/...	[]	[156,282,234,540]	[]
'/home/s2354241/Thesis/...	[]	[783,307,231,534]	[]
'/home/s2354241/Thesis/...	[]	[279,300,242,556]	[]
'/home/s2354241/Thesis/...	[610,336,194,104]	[]	[]
'/home/s2354241/Thesis/...	[]	[]	[59,288,234,156]
'/home/s2354241/Thesis/...	[]	[384,310,252,576]	[]
'/home/s2354241/Thesis/...	[]	[492,323,260,593]	[]
'/home/s2354241/Thesis/...	[]	[163,346,175,419]	[]
'/home/s2354241/Thesis/...	[]	[]	[806,289,279,174]
'/home/s2354241/Thesis/...	[]	[]	[1213,317,210,14...

Figure 4.13: Table variable containing the Ground Truth

4.4 Machine-Learning Algorithms

This Section describes the model architecture that is followed along with the training criteria that were used. As mentioned in Chapter 3, the two-stage detection framework (Faster-RCNN) and the one-stage detectors (YOLOV2 and YOLOV3) are chosen to train the model. The training process for the algorithms is performed using the HPC cluster. Parallel computing was enabled, and the training was run on one GPU.

4.4.1 Faster RCNN

Training

To build the Faster-RCNN network, a pre-trained CNN model as mentioned in Section 3.3 is chosen. For this project, the resnet-50 model, which has 50 convolution layers is chosen. As shown in Figure 4.14, the architecture is made up of repeating convolution blocks that contain the Convolution layer, Batch normalization to normalise convolution layer outputs, ReLU activation layer, and pooling layer. A brief description of each layer is given in the Table 4.2. Different branches of the convolution block are added together in the 'add_1' layer. The convolution blocks are 50 layers deep. Hence, before adding the RPN network, a suitable convolution block has to be chosen as the feature extraction layer.

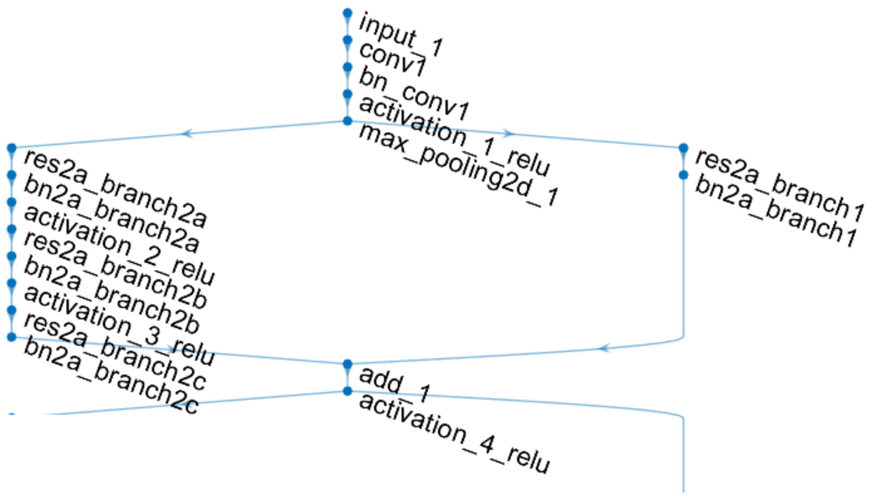


Figure 4.14: Resnet-50 Initial 4 convolution layers

Name	Type
input_1	Image input
conv1	Convolution
bn_conv1	Batch Normalization
activation_1_relu	ReLU
max_pooling2d_1	Max pooling
res2a_branch2a	Convolution
bn2a_branch2a	Batch Normalization
activation_2_relu	ReLU
add_1	Addition

Table 4.2: Initial Layers of Resnet-50

The parameters required to create the training network are

1. Input image size
2. Anchor boxes
3. Feature extraction network

For Resnet-50, the minimum input network size is 224*224, and hence the radargrams in the dataset are scaled down to the network size. This helps in reducing the computational complexity during training. The anchor boxes are chosen using the matlab function '$estimateAnchorBoxes$' and the number of anchor boxes is entered as 3. As the number of convolution layers increases, the complexity of the features extracted will increase as mentioned in Section B.1.2. Hence, the RPN is added after the 40th convolution layer. The RPN network architecture is a CNN with three convolution layers, one ReLU layer, and a softmax layer at the end. The Box regression layer output predicts four box offsets for each anchor box and the classification output layer classifies each anchor as "object" or "background". The RPN output proposals are given to the remaining 10 convolution layers. The final layer is the fully connected RCNN layer with the box regression outputs and the classification output that calculates the position and class of the void in the radargram, respectively. Figure 4.15 and Figure 4.16 show the network architecture as explained above.

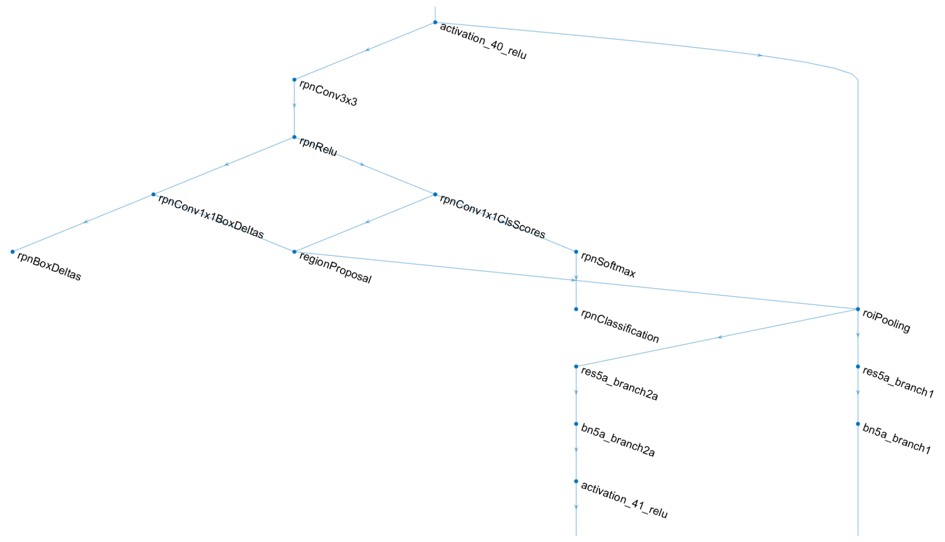


Figure 4.15: Introduction of the RPN layers after 40th convolution layer

Name	Type
rpnConv3x3	Convolution
rpnRelu	ReLU
rpnConv1x1BoxDeltas	Convolution
rpnConv1x1ClsScores	Convolution
regionProposal	Region Proposal
roiPooling	ROI Max Pooling
rpnSoftmax	Softmax Layer
rpnBoxDeltas	Box regression output
rpnClassification	Classification output

Table 4.3: RPN architecture layers

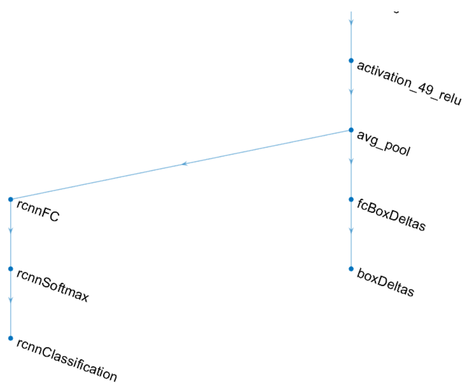


Figure 4.16: Final convolution block with the Fully connected block of the RCNN

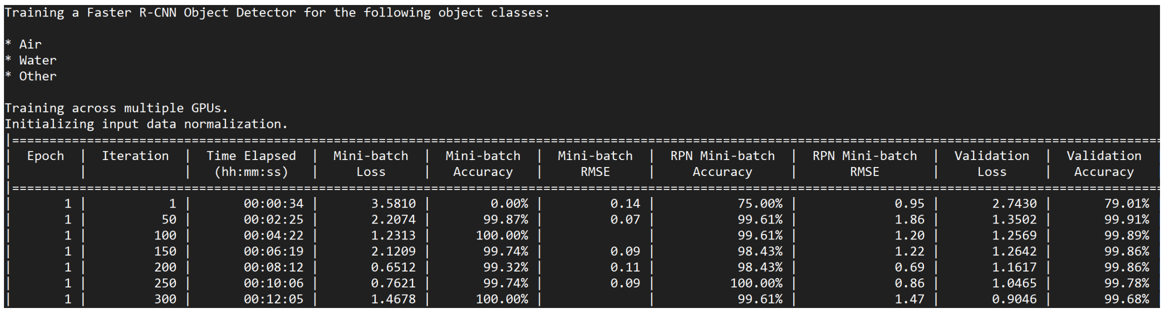
Once the network is built, the next step is to pre-process the dataset to ensure that all the bounding boxes are valid. Pre-processing involves scaling the images to the network input size and checking for negative coordinates in the bounding box or boxes outside the image resolution of 224*224.

To train the neural network, the following parameters as briefly explained in Section 2.4 are set:

Training Option	Value
Solver Name	SGDM
Maximum Epochs	20
Minimum Batch size	2
Initial Learning Rate	1e-3
Validation Data	Validation data of the dataset

Table 4.4: Training parameters for Faster-RCNN network

The pre-processed training dataset, the Faster RCNN network created, and the training options are given as input to the detector for training using the MATLAB function '*trainFasterRCNNObjectDetector*'. At the end of the training, the output obtained is a detector that is used on the testing dataset to check for the performance.


```
Training a Faster R-CNN Object Detector for the following object classes:

* Air
* Water
* Other

Training across multiple GPUs.
Initializing input data normalization.
|==========================================================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Mini-batch  |  RPN Mini-batch  |  RPN Mini-batch  |  Validation  |  Validation
|         |             |   (hh:mm:ss)   |     Loss     |   Accuracy   |     RMSE     |     Accuracy     |       RMSE       |     Loss     |   Accuracy
|==========================================================================================================================================================
|       1 |           1 |       00:00:34 |       3.5810 |        0.00% |         0.14 |           75.00% |             0.95 |       2.7430 |       79.01%
|       1 |          50 |       00:02:25 |       2.2074 |       99.87% |         0.07 |           99.61% |             1.86 |       1.3502 |       99.91%
|       1 |         100 |       00:04:22 |       1.2313 |      100.00% |              |           99.61% |             1.20 |       1.2569 |       99.89%
|       1 |         150 |       00:06:19 |       2.1209 |       99.74% |         0.09 |           98.43% |             1.22 |       1.2642 |       99.86%
|       1 |         200 |       00:08:12 |       0.6512 |       99.32% |         0.11 |           98.43% |             0.69 |       1.1617 |       99.86%
|       1 |         250 |       00:10:06 |       0.7621 |       99.74% |         0.09 |          100.00% |             0.86 |       1.0465 |       99.78%
|       1 |         300 |       00:12:05 |       1.4678 |      100.00% |              |           99.61% |             1.47 |       0.9046 |       99.68%
```

Figure 4.17: Faster RCNN Training status at the first Epoch

At the end of 20 epochs, the validation loss is reduced to 0.12, and the validation accuracy is 99.94%.

4.4.2 YOLO V2

Training

To build the YOLOV2 network, Resnet-50, as shown in Figure 4.14, is chosen as the pre-trained CNN model. The training process is similar to the Faster RCNN method and involves choosing an appropriate network input size, anchor boxes, and feature extraction layer. The minimum network input size is 224*224, three anchor boxes are obtained, and as chosen in the Faster RCNN network, the 40th convolution block is the feature extraction layer.

The convolution blocks and the fully connected layers of resnet-50 are removed and replaced by the YOLOV2 layers. The layers have two serially connected convolution blocks followed by a transformation layer and an output layer. The transformation layer transforms the raw CNN output into a form suitable for object detection. The output layer defines the anchor box parameters like the object class probabilities and location offsets of the bounding box. The final layer implements the function used to train the detector. The architecture is visualised in Figure 4.18.

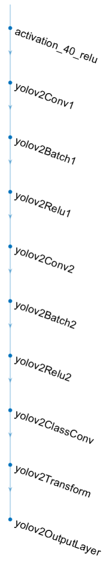


Figure 4.18: YOLOV2 final layers

Name	Type
yolov2Conv1	Convolution
yolov2Batch1	Batch Normalization
yolov2Relu1	ReLU
yolov2Conv2	Convolution
yolov2Batch2	Batch Normalization
yolov2Relu2	ReLU
yolov2ClassConv	Convolution
yolov2Transform	YOLOV2 Transform layer
yolov2OutputLayer	YOLOV2 Output layer

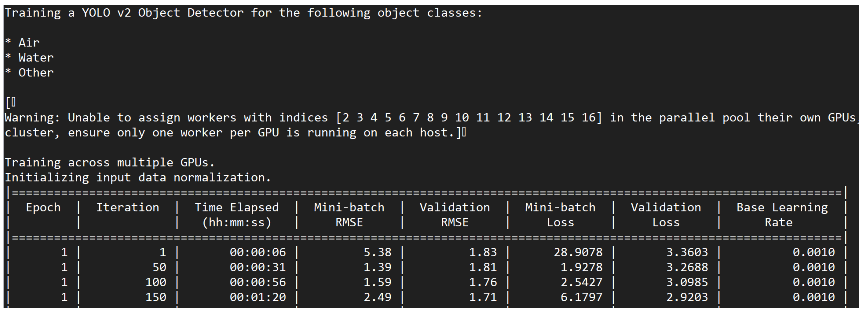
Table 4.5: YOLOV2 architecture layers

The radargrams are resized to the network input size, pre-processed to remove/modify inaccurate bounding boxes, and converted to 224*224*3 since the YOLOV2 algorithm only trains three-channel images. To train the neural network, the following parameters are set:

Training Option	Value
Solver Name	SGDM
Maximum Epochs	20
Minimum Batch size	2
Initial Learning Rate	1e-3
Validation Data	Validation data of the dataset

Table 4.6: Training parameters for YOLOV2 network

The pre-processed training dataset, the YOLOV2 network, and the training options are given as input to the detector for training using the MATLAB function $'trainYOLOv2ObjectDetector'$. At the end of the training, the output obtained is a detector that can be used on the testing dataset to check for performance.



```
Training a YOLO v2 Object Detector for the following object classes:

* Air
* Water
* Other

[
Warning: Unable to assign workers with indices [2 3 4 5 6 7 8 9 10 11 12 13 14 15 16] in the parallel pool their own GPUs,
cluster, ensure only one worker per GPU is running on each host.]

Training across multiple GPUs.
Initializing input data normalization.
|==============================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     RMSE     |     RMSE     |     Loss     |     Loss     |      Rate       |
|==============================================================================================================|
|       1 |           1 |       00:00:06 |         5.38 |         1.83 |      28.9078 |       3.3603 |          0.0010 |
|       1 |          50 |       00:00:31 |         1.39 |         1.81 |       1.9278 |       3.2688 |          0.0010 |
|       1 |         100 |       00:00:56 |         1.59 |         1.76 |       2.5427 |       3.0985 |          0.0010 |
|       1 |         150 |       00:01:20 |         2.49 |         1.71 |       6.1797 |       2.9203 |          0.0010 |
```

Figure 4.19: YOLOV2 Training status at the first Epoch

At the end of 20 epochs, the validation loss is reduced to 1.82, and the training loss is reduced from 28.9 to 1.73.

4.4.3 YOLO V3

Training

To train the YOLOV3 method, a pre-trained CNN needs to be chosen. Initially, resnet50 as seen in the previous algorithms, is selected. Two feature extraction layers need to be chosen for the concatenation and the first detection head. Convolution blocks 30 and 40 are chosen, network input size set to 224*224 and built. It was inferred that since the resnet50 architecture has 50 layers, the computation was high when combined with the YOLOV3 layers and could not be run on a single GPU with parallel computing enabled. Hence, squeezenet is chosen as the pre-trained CNN, and the network is built. Squeezenet is 18 layers deep, and the convolution block is made of a standalone convolution layer and fire modules. The fire modules are the vital block for the Squeezenet architecture since it is made up of 'Squeeze' and 'Expand' convolution layers (Iandola et al., 2016). The number of filters in the squeeze convolution layer is reduced, reducing the inputs to the expand convolution layer. By reducing the number of connections, the number of parameters in the CNN is reduced. The squeezenet CNN is hence used for the training of the YOLOV3 algorithm due to the minimum number of layers and the lower computation required.

The minimum input size for the squeezenet network is 227*227*3, and 6 anchor boxes are generated. The two detection heads use three anchor boxes each. The anchor boxes are represented by [width height] of a bounding box and are arranged in descending order. The first three bigger boxes are applied to the first detection head since the image resolution is high, and the remaining small anchor boxes are applied to the second detection head. The next step

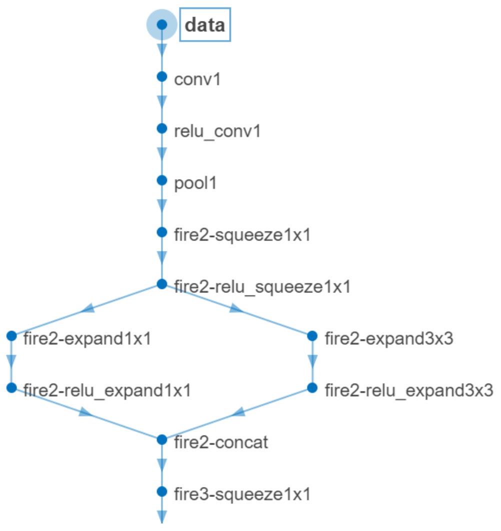


Figure 4.20: Initial two fire modules of the Pre-trained Squeezenet

Name	Type
data	Image input
conv1	First Convolution
relu_conv1	ReLU
pool1	Max Pooling
fire2-squeeze 1x1	'Squeeze' Convolution
fire2-relu_squeezl 1x1	ReLU
fire2-expand 1x1/3x3	'Expand' Convolution with 1x1 and 3x3 filters
fire2-relu_expand 1x1/3x3	ReLU
fire2-concat	Depth Concatenation

Table 4.7: Initial layers of Squeezenet

is to choose the two feature extraction layers for the concatenation and the first detection head. The firemodule 5 and 9 are chosen respectively, as shown in Figure 4.21.

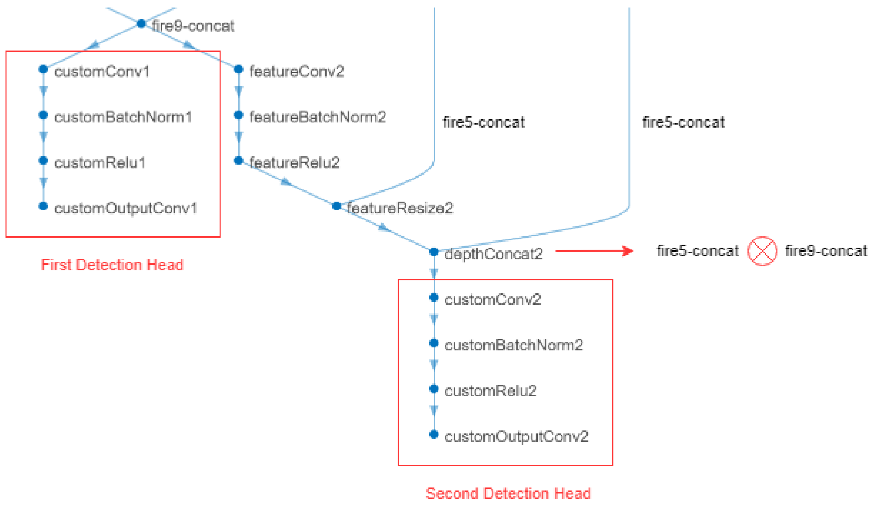


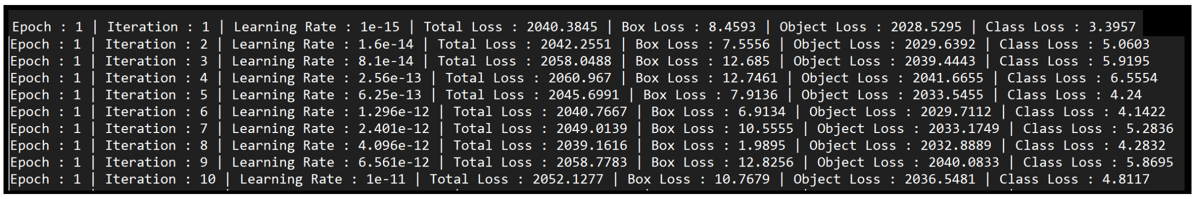
Figure 4.21: YOLOV3 layers of the squeezenet network

Once the network for training is created, the dataset is pre-processed as in earlier methods to resize to 227*227*3. Since the number of layers in the squeeznet is lower than resnet-50, the number of epochs required for training is increased. The following training parameters are set:

Training Option	Value
Maximum Epochs	80
Minimum Batch size	8
Initial Learning Rate	1e-3
Warm-up Period	1000

Table 4.8: Training parameters for YOLOV3 network

The training of the YOLOV3 network differs from the other methods since the pre-processed data is split into mini-batches for training. The detector is created by building the network using the '$yolov3ObjectDetector$' command in MATLAB. The detector parameters are then updated during training based on the learning rate and the training loss.



```
Epoch : 1 | Iteration : 1 | Learning Rate : 1e-15 | Total Loss : 2040.3845 | Box Loss : 8.4593 | Object Loss : 2028.5295 | Class Loss : 3.3957
Epoch : 1 | Iteration : 2 | Learning Rate : 1.6e-14 | Total Loss : 2042.2551 | Box Loss : 7.5556 | Object Loss : 2029.6392 | Class Loss : 5.0603
Epoch : 1 | Iteration : 3 | Learning Rate : 8.1e-14 | Total Loss : 2058.0488 | Box Loss : 12.685 | Object Loss : 2039.4443 | Class Loss : 5.9195
Epoch : 1 | Iteration : 4 | Learning Rate : 2.56e-13 | Total Loss : 2060.967 | Box Loss : 12.7461 | Object Loss : 2041.6655 | Class Loss : 6.5554
Epoch : 1 | Iteration : 5 | Learning Rate : 6.25e-13 | Total Loss : 2045.6991 | Box Loss : 7.9136 | Object Loss : 2033.5455 | Class Loss : 4.24
Epoch : 1 | Iteration : 6 | Learning Rate : 1.296e-12 | Total Loss : 2040.7667 | Box Loss : 6.9134 | Object Loss : 2029.7112 | Class Loss : 4.1422
Epoch : 1 | Iteration : 7 | Learning Rate : 2.401e-12 | Total Loss : 2049.0139 | Box Loss : 10.5555 | Object Loss : 2033.1749 | Class Loss : 5.2836
Epoch : 1 | Iteration : 8 | Learning Rate : 4.096e-12 | Total Loss : 2039.1616 | Box Loss : 1.9895 | Object Loss : 2032.8889 | Class Loss : 4.2832
Epoch : 1 | Iteration : 9 | Learning Rate : 6.561e-12 | Total Loss : 2058.7783 | Box Loss : 12.8256 | Object Loss : 2040.0833 | Class Loss : 5.8695
Epoch : 1 | Iteration : 10 | Learning Rate : 1e-11 | Total Loss : 2052.1277 | Box Loss : 10.7679 | Object Loss : 2036.5481 | Class Loss : 4.8117
```

Figure 4.22: YOLOV3 Training status at the first 10 iterations of the first Epoch

Epoch	Total Loss	Box Loss	Object Loss	Class Loss
20	0.47151	0.26405	0.19854	0.0089262
80	0.016954	0.0072697	0.0090093	0.0006751

Table 4.9: Comparison of YOLOV3 training results after 20th and 80th Epoch

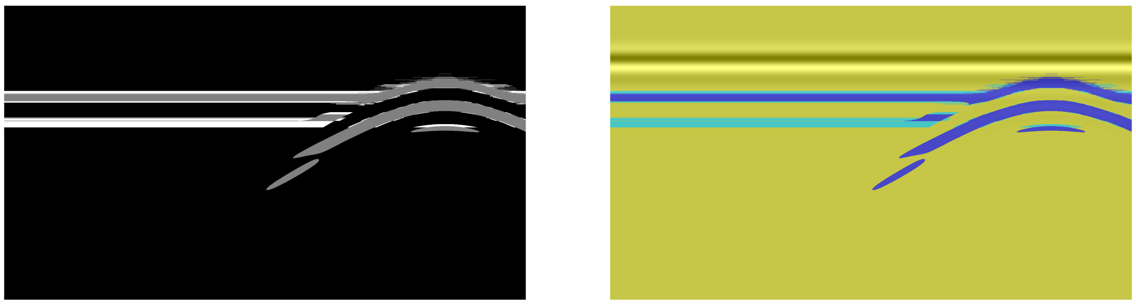
4.4.4 Image processing

In addition to the Machine learning algorithm, image processing is added to the methodology to extract the hyperbolic features before training. This results in more efficient training since the required features have been extracted. The following steps are taking for image processing.

1. Removing the sewer line
2. Clustering and Segmentation
3. Overlay of filter on the original image

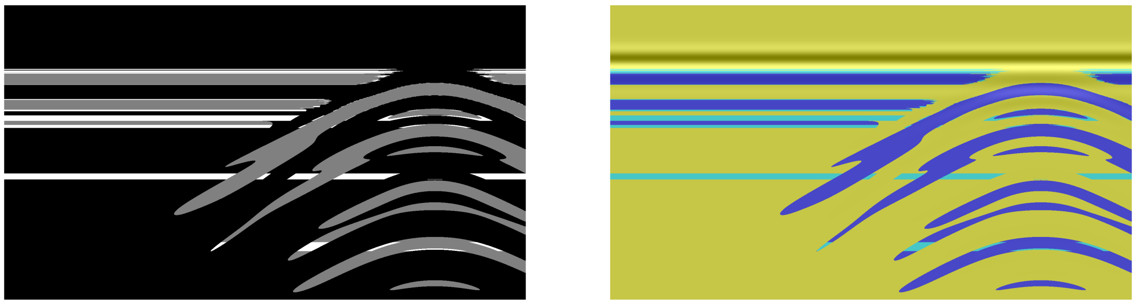
Due to the reflections of the electromagnetic waves from the sewer structure, a horizontal line can be seen on the radargrams as seen in Figure 4.3. The horizontal line is removed due to the false detection of voids close to the sewer line. The apex of the hyperbolic reflection of these voids is interrupted due to the line's presence and is hence not detected by Machine-Learning detectors. The radargram is converted to the frequency domain using Fourier transform, and a convolution operation is performed with a filter mask to remove the horizontal sewer line.

K-means clustering is performed on the radargrams to segment the image into 3 clusters. The segmentation results are fused with the original image to obtain three-channel images of dimensions 1920*1080*3 for training purposes. The results of segmentation and image overlay are as seen in Figure 4.23 and Figure 4.24. It can be observed that the reflections of the voids are extracted and, therefore can be used for training detectors. During the removal of the horizontal line, additional horizontal lines are generated in the image. During training, these lines are ignored since the feature of interest is the hyperbolic signature with its apex intact.



(a) K-means segmentation **(b)** Overlay of filter

Figure 4.23: Radargram containing an Air void



(a) K-means segmentation **(b)** Overlay of filter

Figure 4.24: Radargram containing a Water void

5 Results and Discussion

The detection outputs of each ML model is elaborated in this chapter. The performance metric commonly used for classification algorithms is applied, and the results are discussed. The voids are labelled as 'Air'(white bounding box), 'Water'(blue bounding box), and 'other'(red bounding box) depending on the detection accuracy. The radargrams are plotted with the two way travel time (tR) along the y-axis and the number of traces along the x-axis. The axes of the radargrams are not displayed in the images to reduce cluttering.

The detection results can be subdivided into True Positives, False Positives, and False Negatives. Since all the datasets have at least one void simulated, there are no True Negative detection.

This Chapter contains an overview of the results obtained for each detector, followed by the comparison of the progress during training of the detectors. A final summary gives the results obtained as per the research objectives. Appendix A includes the results of the detector on the datasets.

5.1 Datasets

Three different kinds of dataset are created as mentioned in Section 4.2. The first dataset consists of a homogeneous sand environment with single voids and is hereafter termed 'Dataset A'.

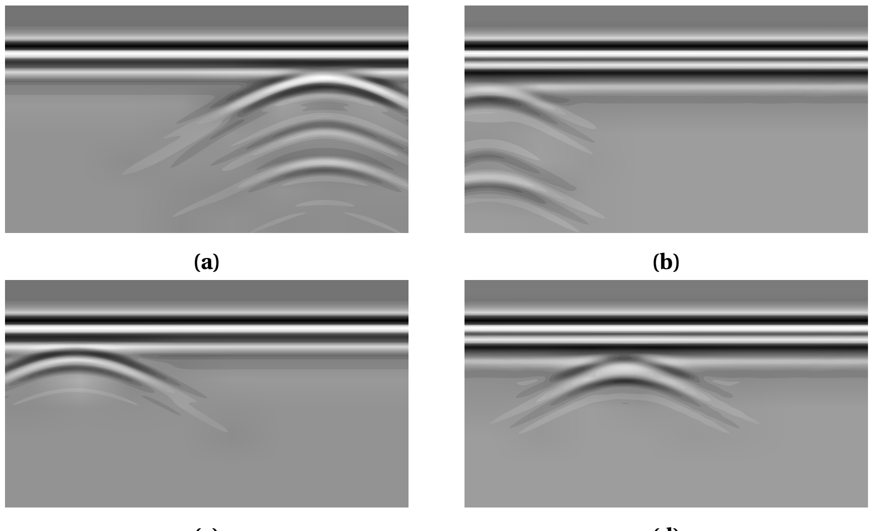


Figure 5.1: Dataset A example radargrams: (a), (b) Water void in dry and saturated sand (c),(d) Air void in dry and saturated sand

The second dataset consists of a heterogeneous sand environment with different sand layers. The geometry of the layers can be visualised as shown in the Figure 5.2. 500 radargrams are created in a heterogeneous environment and is termed as 'Dataset B'.

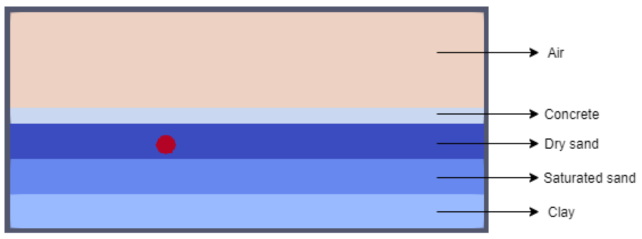
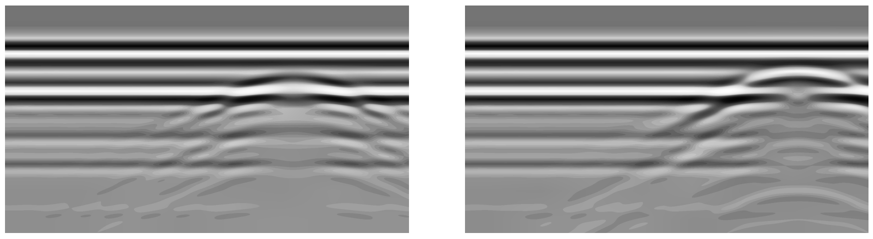


Figure 5.2: Example of the geometry of the heterogeneous sand environment



(a) Air void **(b)** Water void

Figure 5.3: Dataset B : Single void in a heterogeneous soil environment

The third dataset used for training consists of multiple voids of randomized materials that are generated. The dataset is termed as 'Dataset C'. An example of a radargram with multiple voids is as shown in Figure 5.4.

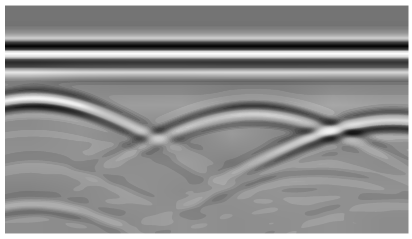
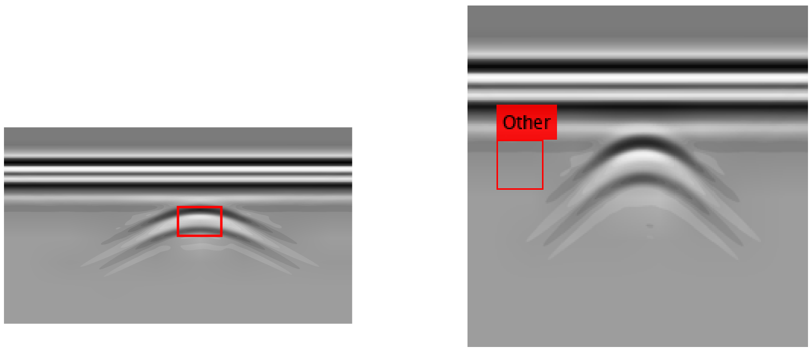


Figure 5.4: Dataset C - Radargram with multiple voids

5.2 Faster RCNN

Faster-RCNN detector is first applied on dataset A. 1000 images were trained and the elaborate results can be found in Appendix A. Before applying the detector on the testing dataset, the original radargrams had to be resized to the training input size of 224*224 for detection. The detector did not work on the original size of the images and the detection took an average time of 4.6 seconds for every image.

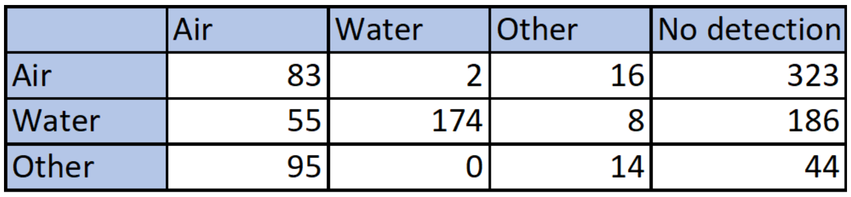
In terms of classification of the 'Other' objects, the RCNN detector performs poorly as seen in Figure 5.5. No voids were detected in this class and the only detection that was made were false positives in saturated sand.



(a) Ground truth **(b)** Detection result

Figure 5.5: Faster-RCNN detection results for Other void in Saturated sand

With the available data, the confusion matrix of the classification was generated as shown in Figure 5.6



	Air	Water	Other	No detection
Air	83	2	16	323
Water	55	174	8	186
Other	95	0	14	44

Figure 5.6: Confusion Matrix for Faster-RCNN method (Dataset A)

Inference

From the performance of the Faster-RCNN algorithm, the following points are obtained.

1. 55.3% of the voids in the 1000 radargrams were False negatives.
2. 19.16% of Air voids were true positives.
3. 41.10% of Water voids were true positives.
4. The algorithm has an overall detection accuracy of only 27.10%

The performance metrics to measure the performance of the detection are summarised in Table 5.1. As seen in the table, the detection of water voids is better than the other classes. Though the accuracy for 'other' objects is at 83.7%, the classification was misplaced as seen in Figure 5.5 and did not detect any voids.

Class	Number of actual cases	Number of Predicted cases	Accuracy	Precision	Recall	F1 Score
Air	424	233	50.9%	0.20	0.36	0.25
Water	423	176	74.9%	0.41	0.99	0.58
Other	153	38	83.7%	0.092	0.37	0.15

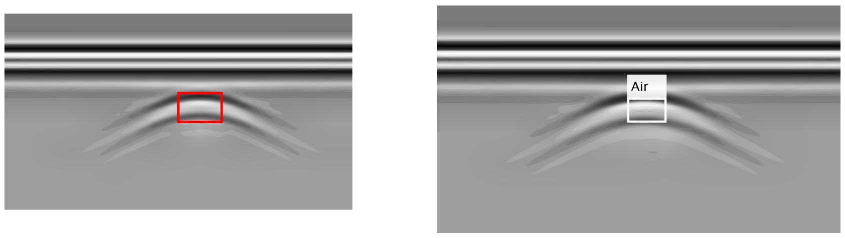
Table 5.1: Performance metrics for the Faster-RCNN method (Dataset A)

Due to the low overall accuracy and a high number of False positives, it can be concluded that the Faster-RCNN method is not a viable method for void detection. The detector is hence not applied on the other datasets.

5.3 YOLOV2

The YOLOV2 detector is used on the same dataset that has been used for the Faster-RCNN. The YOLOV2 detector could be used on the original resolution of the radargram, but they had to be converted to three-channel radargrams. The detection speed is an average of 0.2 s per radargram.

The material of the objects used for class 'Other' was either wood or brick with relative permittivity (ϵ_r) 2 and 3.26, respectively. Since the ϵ_r of Air is 3 and is close to the values of wood and brick, the hyperbolic reflection of the 'Other' objects is similar to 'Air' voids. Hence, as seen in Figure 5.7, all the classes of 'other' objects were detected as Air. Though there is a void detection, the classification is wrong, and they are all detected as False positives.

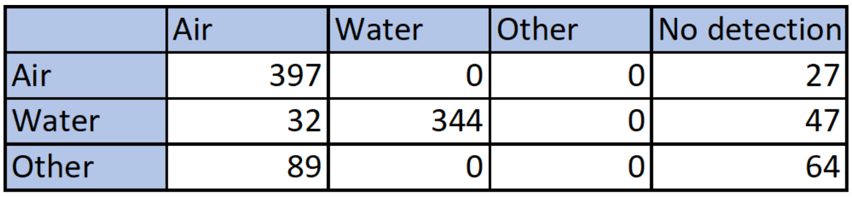


(a) Ground truth **(b)** Detection result

Figure 5.7: YOLOV2 detection results applied to Dataset A for Other void in Saturated sand

In addition to the True positives, there was a 13.8% of no detection (False negatives). In comparison with the previous detection method, there is a decrease of 75% in false positives.

The confusion matrix of the predicted class against the actual class is as shown in Figure 5.8.



	Air	Water	Other	No detection
Air	397	0	0	27
Water	32	344	0	47
Other	89	0	0	64

Figure 5.8: Confusion Matrix for YOLOV2 method (Dataset A)

Inference

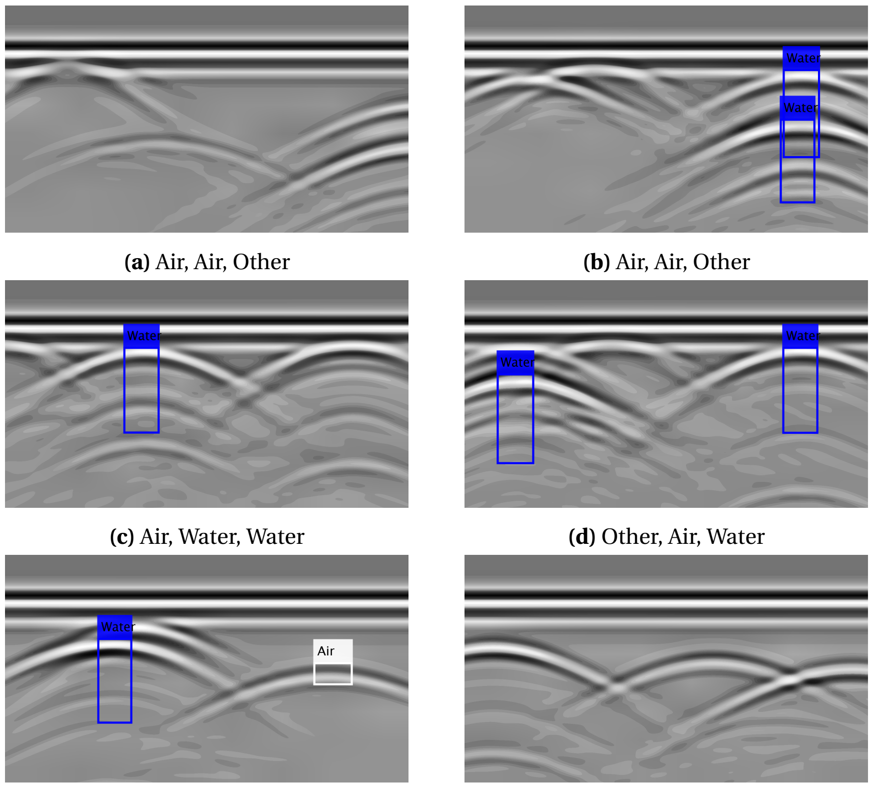
1. The overall accuracy of the detection is 74.10%, which is more than three times the classification accuracy of the Faster-RCNN method.
2. There was a higher classification accuracy for Air and water voids at 93.60% and 81.30% respectively.
3. All the objects belonging to class 'Other' were detected as 'Air'.

Class	Number of actual cases	Number of Predicted cases	Accuracy	Precision	Recall	F1 Score
Air	424	518	85.2%	0.94	0.77	0.84
Water	423	344	92.1%	0.81	1.0	0.90
Other	153	0	84.7%	0.0	0.0	0.0

Table 5.2: Performance metrics for the YOLO-V2 method (Dataset A)

The classification of Air and Water voids in the radargram by the YOLOV2 detector on dataset A gave an accuracy of around 87%.

Since the YOLOV2 has a high detection accuracy compared to Faster-RCNN, the detector is then checked on dataset C. The detection of voids in this dataset is as seen in Figure 5.9.



(a) Air, Air, Other **(b)** Air, Air, Other

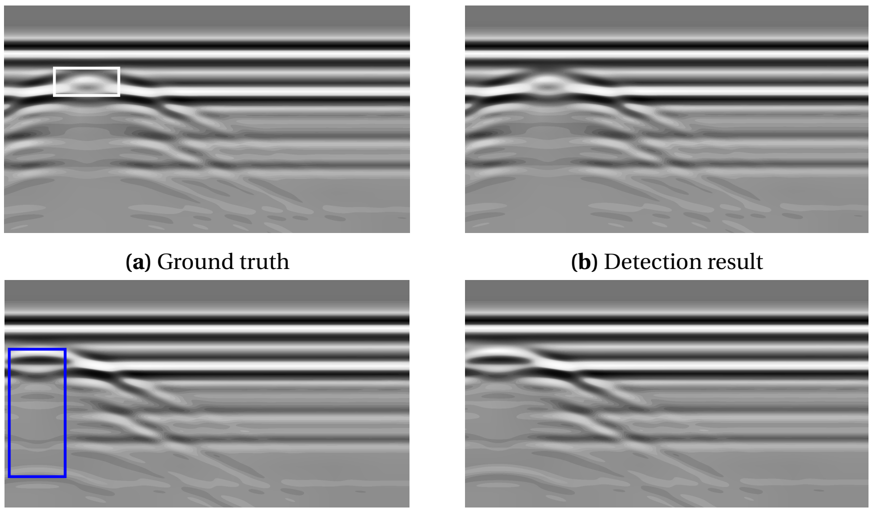
(c) Air, Water, Water **(d)** Other, Air, Water

(e) Water, Air, Air **(f)** Water, Air, Water

Figure 5.9: YOLOV2 detection results on Dataset C

The YOLOV2 detector works better on single voids, but the detector fails to produce accurate results when there are multiple voids. As seen in Figure 5.9(c)(d)(e), one of the voids in the radargram is detected accurately even though the false positives are more. The inaccuracy may be due to the overlap of the parabolic reflections in the radargram.

The final training was done on dataset B, and the performance was measured. As seen in the Figure 5.10, none of the voids could be detected due to the presence of the horizontal line representing the change in sand layers. This results in the interruption of the hyperbola apex, and the reflections due to the change in permittivity of sand overlap the hyperbolic reflections. Since the feature to be extracted are the hyperbolic reflections, the detector fails in detecting them.



(a) Ground truth **(b)** Detection result

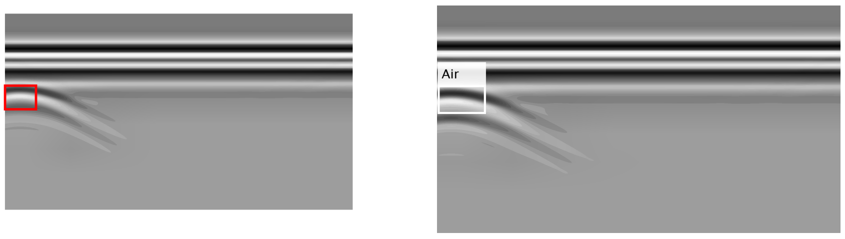
(c) Ground truth **(d)** Detection result

Figure 5.10: YOLO V2 detection result on Dataset B

5.4 YOLOV3

The YOLOV3 method is performed next to compare the accuracy of detection with the YOLOV2 method.

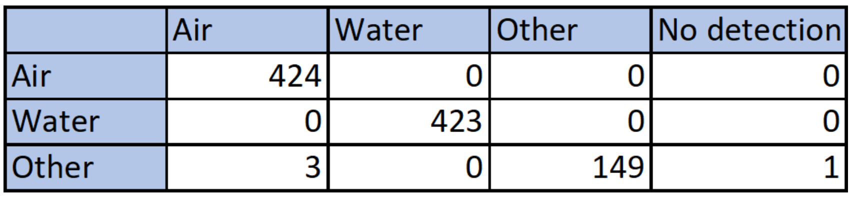
In a total of 1000 images in the dataset, there were only three False positives and one False negative. The false positive detections were objects of class ’Other’ wrongly detected as ’Air’ voids as seen in Figure 5.11(b). This occurred due to the placement of the hyperbola at the edge of the image.



(a) Ground truth **(b)** Detection result

Figure 5.11: YOLOV3 detection results applied to Dataset A (b) False positive

A confusion matrix is generated from the results obtained and the performance metrics of the detector are evaluated as shown in Figure 5.12 and Table 5.3 respectively.



	Air	Water	Other	No detection
Air	424	0	0	0
Water	0	423	0	0
Other	3	0	149	1

Figure 5.12: Confusion Matrix for YOLOV3 method (Dataset A)

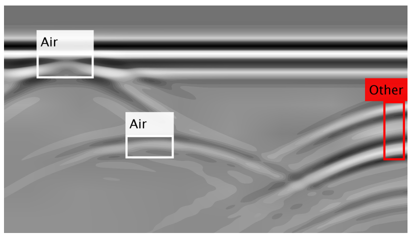
Class	Number of actual cases	Number of Predicted cases	Accuracy	Precision	Recall	F1 Score
Air	424	427	99.7%	1.0	0.99	1.0
Water	423	423	100%	1.0	1.0	1.0
Other	153	149	99.6%	1.0	1.0	0.99

Table 5.3: Performance metrics for the YOLO-V3 method (Dataset A)

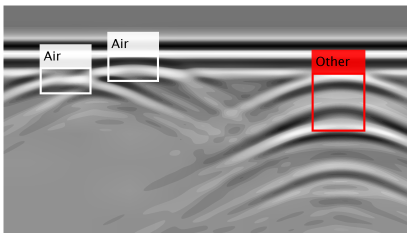
5.4.1 Inference

1. The overall detection accuracy of the YOLOV3 detector is 99.6%.
2. The performance of the detector for every class is above 99%.

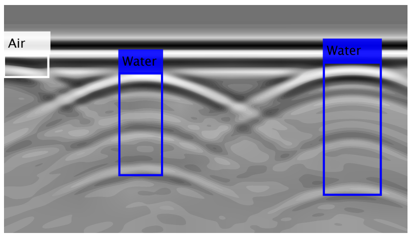
Due to near-perfect accuracy in detection by the YOLOV3 detector, it becomes promising to deal with void detection. The detector is hence trained on Dataset C. The detection results are as seen in Figure 5.13.



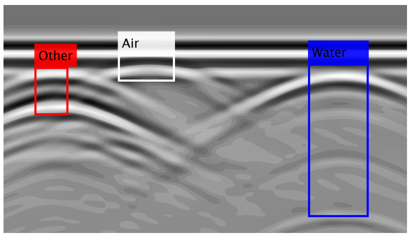
(a) Air, Air, Other



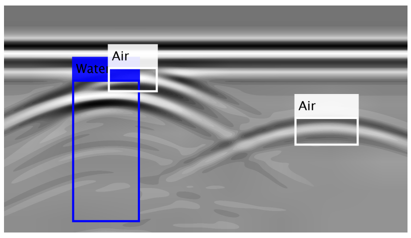
(b) Air, Air, Other



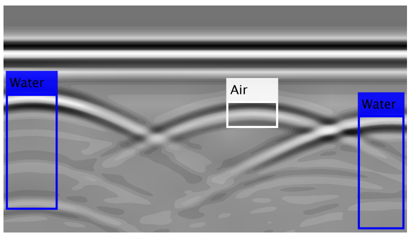
(c) Air, Water, Water



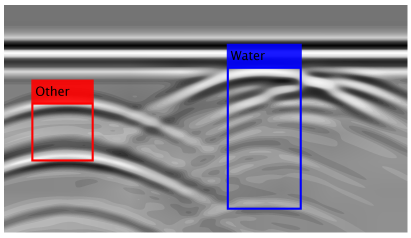
(d) Other, Air, Water



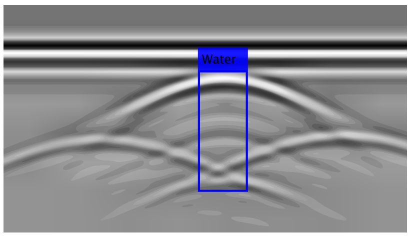
(e) Water, Air, Air



(f) Water, Air, Water



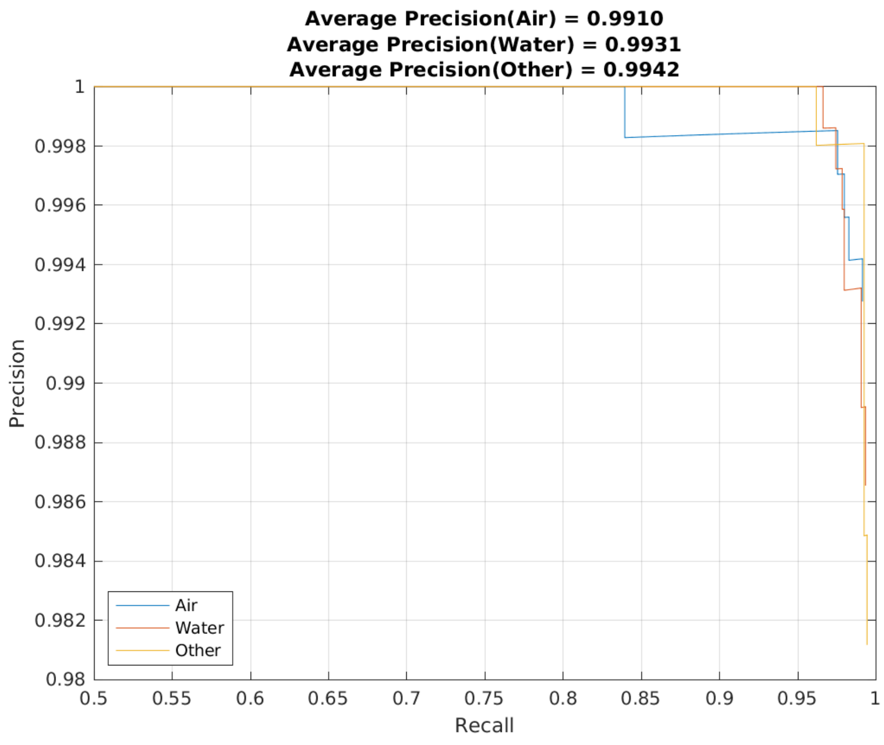
(g) Other, Water, Air



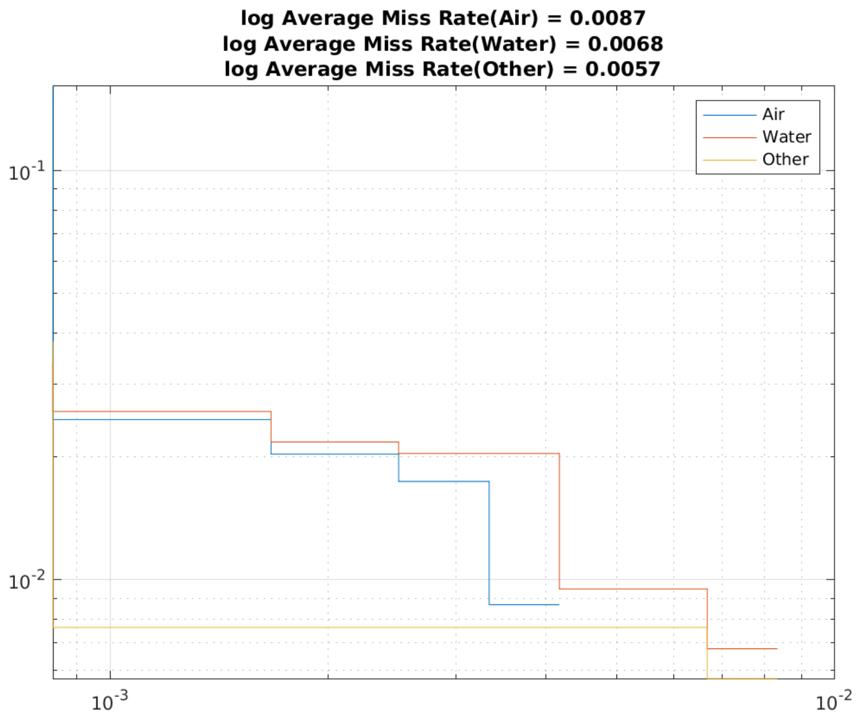
(h) Other, Water, Other

Figure 5.13: YOLOV3 detection results on datasets containing multiple voids

It is inferred from the detection results that void detection in cases of multiple voids performs better than the YOLOV2 method. As seen in Figure 5.13(g) and (h), the voids that have their hyperbola apex disturbed are not detected.



(**a**) Precision vs Recall



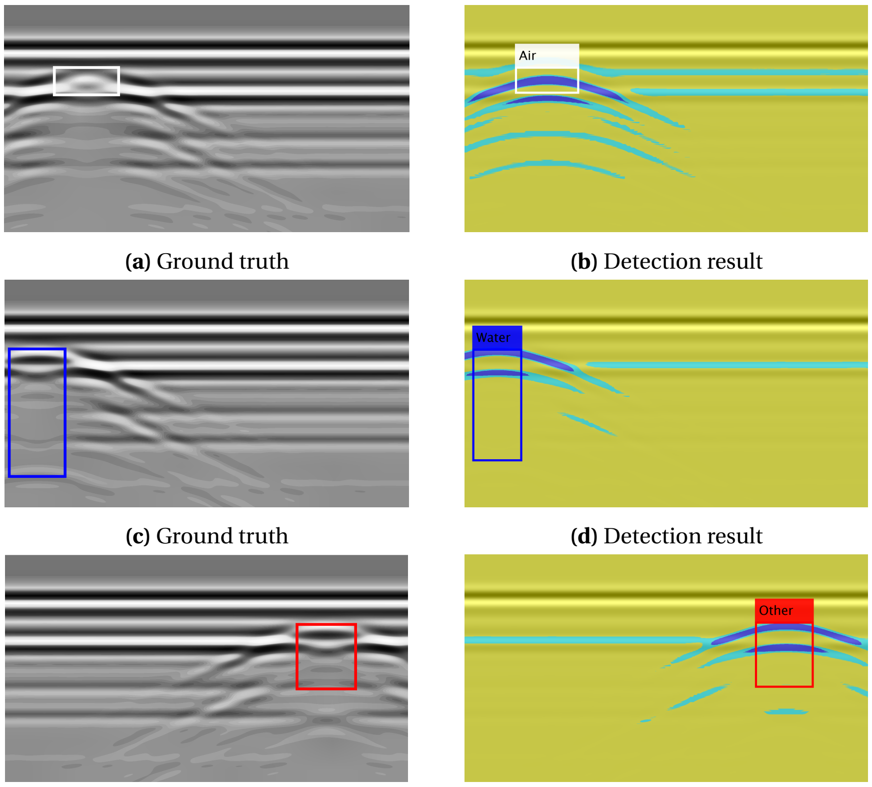
(**b**) Detection Miss rate

Figure 5.14: Performance metric of the YOLOV3 detector on Dataset C

The performance of the detector can be visualised in Figure 5.14a and Figure 5.14b. The first figure shows the precision vs. recall plots of the three classes. The average value of each class remains at above 99%, denoting the number of true positives detected. The average miss rate of the detector is as seen in Figure 5.14b and is very low, indicating good detection accuracy.

In order to further improve detection accuracy in cases where the hyperbolas are not detected due to a disturbance in their apex, image processing is employed. The image processing method as explained in section 4.4.4 is used to extract the hyperbolic reflections to retain their shape before training. To check the feature extraction, image processing is first applied to Dataset B. The hyperbolic features irrespective of the other reflections are extracted and the dataset is further trained using the YOLOV3 method.

The detection results as seen in Figure 5.15 show more number of reflections in the Air void. This is due to the difference in sand permittivity across the layers.



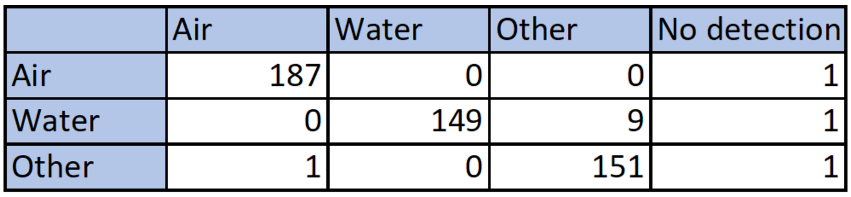
(a) Ground truth **(b)** Detection result

(c) Ground truth **(d)** Detection result

(e) Ground truth **(f)** Detection result

Figure 5.15: YOLOV3 detection results after applying Image processing on Dataset B

With the available data, the confusion matrix of the classification was generated as shown in Figure 5.16



	Air	Water	Other	No detection
Air	187	0	0	1
Water	0	149	9	1
Other	1	0	151	1

Figure 5.16: Confusion Matrix for Image processing + YOLOV3 method applied to Dataset B

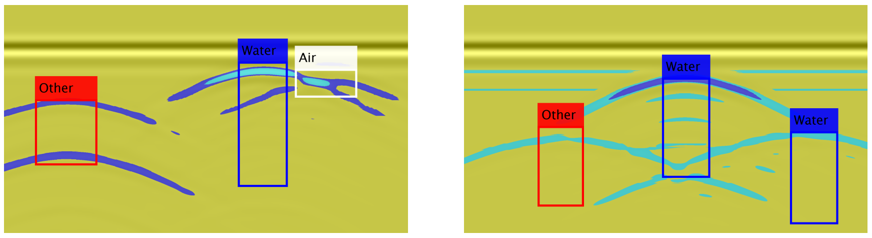
Out of a training dataset of 500 images, three images were False negatives, and one image was a false positive. The performance metric for this method of detection gives an overall accuracy of 97.4%.

Class	Number of actual cases	Number of Predicted cases	Accuracy	Precision	Recall	F1 Score
Air	188	188	99.6%	0.99	0.99	0.99
Water	159	149	98%	0.94	1.0	0.97
Other	153	160	97.8%	0.99	0.94	0.96

Table 5.4: Performance metric for YOLOV3 + Image processing (Dataset B)

The image processing + YOLOV3 method is then applied to the dataset of multiple voids. Figure 5.17 display the radargrams that were not detected using the YOLOV3 method as seen in Figure 5.13.

Using this method, the hyperbola detection accuracy is improved since the features are extracted using image processing. In Figure 5.17(b), the third void is detected as class 'Water' when it should have been detected as 'Air'.



(a) Other, Water, Air **(b)** Other, Water, Other

Figure 5.17: Image processing + YOLOV3 detection method applied to Dataset C

5.5 Training evaluation

The overall training progress of the detectors is plotted using the number of iterations and the training loss obtained during training of dataset A. The training loss indicates the poor performance of the detector on a single detection. The goal of machine learning is to reduce the total loss to have a perfect model prediction. As seen in the figure Figure 5.18, the training loss for the Faster-RCNN detector reduces to less than 1 from the 400th iteration and does not fluctuate to a value of more than 2.

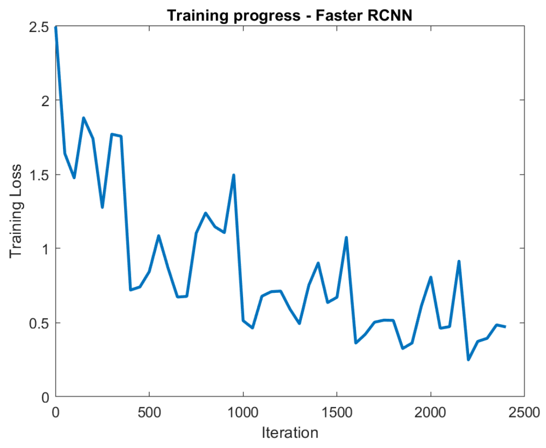


Figure 5.18: Iteration vs Training Loss of Faster RCNN detector

The training loss for the YOLOV2 detector, as seen in Figure 5.19 shows much fluctuation throughout the training process. The fluctuations remain until the end of the training, which shows the detector's imperfect prediction of voids.

The fluctuations in training loss can be caused due to several factors which need further tuning of a) the size of the network, b) Learning rate, c) Batch normalization, d) Activation function, or e) depth of the network. The tuning of the different parameters needs to be done on a trial and error basis till the appropriate parameters are chosen. The pre-trained resnet-50 model currently employed had batch normalization after every convolution layer followed by the ReLU activation function. Since the different machine learning methods are applied to choose the most accurate one for void detection, the tuning for the YOLOV2 detector was not performed.

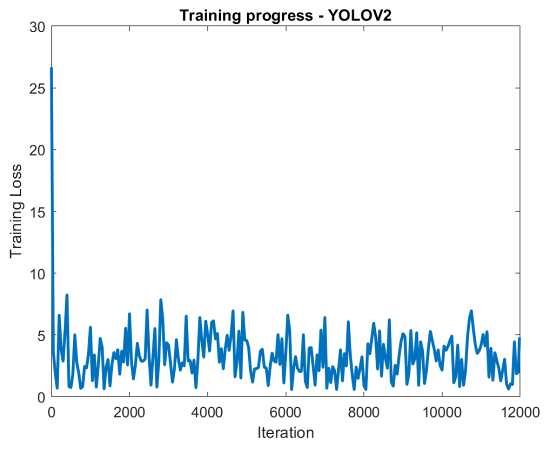


Figure 5.19: Iteration vs Training Loss of YOLOV2 detector

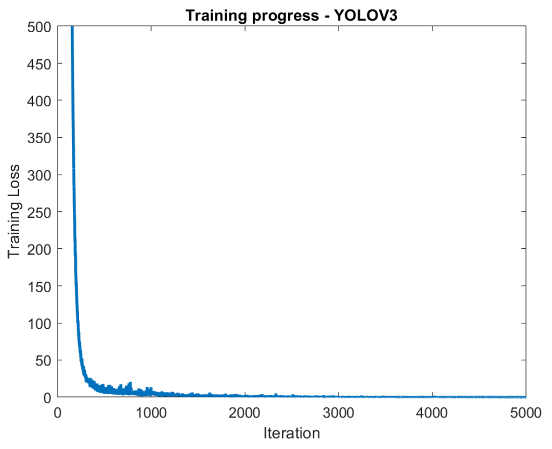


Figure 5.20: Iteration vs Training Loss of the YOLOV3 detector

The training progress of the YOLOV3 method, as seen in Figure 5.20, show no fluctuations in the training loss resulting in a total loss of less than two after the 1500th iteration. At the end of the training, the total loss is '0.021172'.

5.6 Summary

The datasets used for training and testing the detectors are as follows:

Dataset	No. of Radargrams	No. of voids	Dry Sand	Saturated sand	Heterogeneous sand
A	750	1	x		
A	250	1		x	
B	500	1			x
C	500	3	x		

Table 5.5: Datasets A,B and C in their respective environment

1. **Object detection using Automatic detection methods.**

   Three different machine learning methods are applied to the datasets and compared. Two-stage detectors like the Faster-RCNN model and single-stage detectors YOLOV2 and YOLOV3 were used. On comparing the three methods, the YOLOV3 detector performed very well by giving accurate results. The training loss was also lower during the training of the YOLOV3 detector. The accuracy of each method can be summarised as below in Table 5.6.

Detector	Overall Accuracy	Class Accuracy -Air	Class Accuracy - Water	Class Accuracy - Other
Faster- RCNN	27.10%	50.9%	74.9%	83.7%
YOLO V2	74.10%	85.2%	92.1%	84.7%
YOLO V3	99.6%	99.7%	100%	99.6%

Table 5.6: Comparison of performance of the three detection methods

   The faster RCNN method was the commonly used state-of-the-art model for hyperbola recognition. Though it performed well in object detection, it failed in the detection of voids of different materials. The total number of false negatives was also high in the Faster RCNN detector. The training progress of the detectors as shown in Figure 5.21 show more fluctuations during training of the YOLOV2 network when compared to the Faster RCNN and YOLOV3 model.

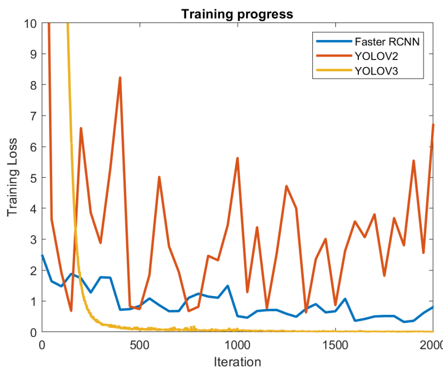


Figure 5.21: Iteration vs Training Loss of the three detectors

   A deep neural network is like a black box with weights trained over an extensive data set to extract features. Each layer learns at different speeds, and they each have varying gradients. The model works by making predictions, and when there are heavy fluctuations in the training loss, it means the model is making completely random predictions. This makes the current implementation of the YOLOV2 detector unreliable even though the accuracy is higher than the Faster RCNN method. Hence, the current model with 74.10% detection accuracy was suitable for single void detection. The detector's performance was inadequate when it came to multiple voids in a radargram or a heterogeneous soil environment.

The final method, the YOLOV3, had near-perfect results in void detection even though the network used was a pre-trained Squeezenet that is 18 layers deep. Even with the reduction in the number of layers, the YOLOV3 architecture was more suitable for void detection.

2. **Differentiation between different void compositions.**

   The objective of this research was to focus on the detection of voids primarily composed of Air or Water. The feature difference between the two voids is the number of hyperbolic reflections in the radargram. The difference in features between the voids resulted in successful detections by all three machine learning methods. All the detectors were able to detect true positives though the accuracy varied. Table 5.7 shows the number of true positives in every object class detected by the three methods.

Detector	True Positives Air	True Positives Water	True Positives Other
Faster- RCNN	19.6%	41.1%	9.15%
YOLO V2	93.6%	81.3%	0.00%
YOLO V3	100.0%	100.0%	97.39%

Table 5.7: Number of True positives detected in each class by the three detection methods in Dataset A

3. **Inclusion of Pre-processing with Machine Learning.**

   Image processing was only combined with the YOLOV3 method to check the improvement in accuracy. The Faster RCNN and YOLOV2 methods had their disadvantages, and the accuracy was comparatively lower than YOLOV3. Hence image processing was only applied with the YOLOV3 method to improve the accuracy in detection for false positives and false negatives.

   The dataset with heterogeneous sand (Dataset B) could not be trained since the features to be extracted were overlapped by the extra reflections. Few voids in Dataset B also did not give satisfactory results since the apex of the hyperbola was interrupted by the adjacent void. In cases like these, image processing was used to extract the hyperbolic features. This improved the accuracy in detection since the voids in the heterogeneous environment could be detected.

6 Conclusion and Recommendations

In this chapter, discussion if the research objectives are met is elaborated based on the results from Chapter 5. The goal of the research is answered concerning the design objectives formulated earlier. Furthermore, recommendations for future work are explained.

6.1 Conclusion

1. **Object detection using Automatic detection methods.**

   Three Machine-Learning models were chosen from the analysis made in Chapter 3. Based on the results obtained, the YOLOV3 detector performed the best by giving accurate results. An overall accuracy of 99.6% is obtained with low training loss making YOLOV3 the most fitting method for void detection.

2. **Differentiation between different void compositions.**

   The primary objective of this research is to focus on hyperbolic reflections to detect voids composed of Air or Water. The difference in reflections were captured in all the three methods though the precision accuracy varied.

   It can be inferred that the detection accuracy of class ’Water’ is higher in all three detectors due to the significant difference in features compared to the Air and Other class. The objects of composition ’Other’ have the lowest number of True Positives in every method. This was because of the similar permittivity value to Air. The hyperbolic reflections appear similar on the radargram, and the features were not clear enough to be detected accurately. The advantage of the YOLOV3 method is the architecture since it helps the model detect small objects. This helped accurately detect the minute differences in features between the ’Air’ voids and ’other’ objects.

3. **Inclusion of Pre-processing with Machine Learning.**

   Contrast enhancement was essential during the dataset’s preparation since it helped enhance the reflections that were the key features to detection. Image processing combined with the YOLOV3 method proved to give better results in cases where the hyperbola apex is interrupted. By extracting the hyperbolic shape, image processing is a valid addition to YOLOV3 in cases where the hyperbolic reflections are distorted.

   The similar performance of the YOLOV3 with or without image processing on single voids in a homogeneous soil environment strengthens the conclusion that the YOLOV3 method is suitable for void detection.

Summary

In summary, compared to the different methods chosen, the YOLOV3 method performed far better than the other detection methods. Currently, the GPR survey is done manually by trained specialists. A fully automated sewer inspection system could be created by developing in-pipe GPR and applying machine learning algorithms to the radargrams.

6.2 Recommendations

The current model chosen out of the three can successfully detect hyperbolas and classify the void compositions in a synthetic radargram. Several possible extensions and improvements can be considered to develop the system further.

Training Dataset

The current dataset used to train the models was synthetically created. The actual survey data has more noise and sharper resolution of the hyperbolas. A dataset containing ground truth depicting the different void compositions of the hyperbola would be a more realistic dataset for training. Due to the novelty of the in-pipe GPR survey, no datasets for void detection were available. This gives rise to future work that involves operating a GPR survey inside the pipe and detecting subsurface voids using machine learning methods.

Void shapes

The voids synthesized for this research were all cylindrical. Subsurface voids are generally irregular in shape, and hence different void shapes could be simulated to check the detection accuracy. In addition to simulating perfect voids, areas of lower density around the sewer pipeline can be simulated for future detection.

Hyper-parameter Training

As explained earlier in Chapter 5, the different machine learning model parameters can be tuned for further improvement in the model's training progress. Different pre-trained networks can be applied on a trial and error basis to check for the best network. Currently, the first two networks were trained with 20 epochs. The number of epochs could be increased in batches to avoid overfitting and check if the accuracy could be increased for YOLOV2.

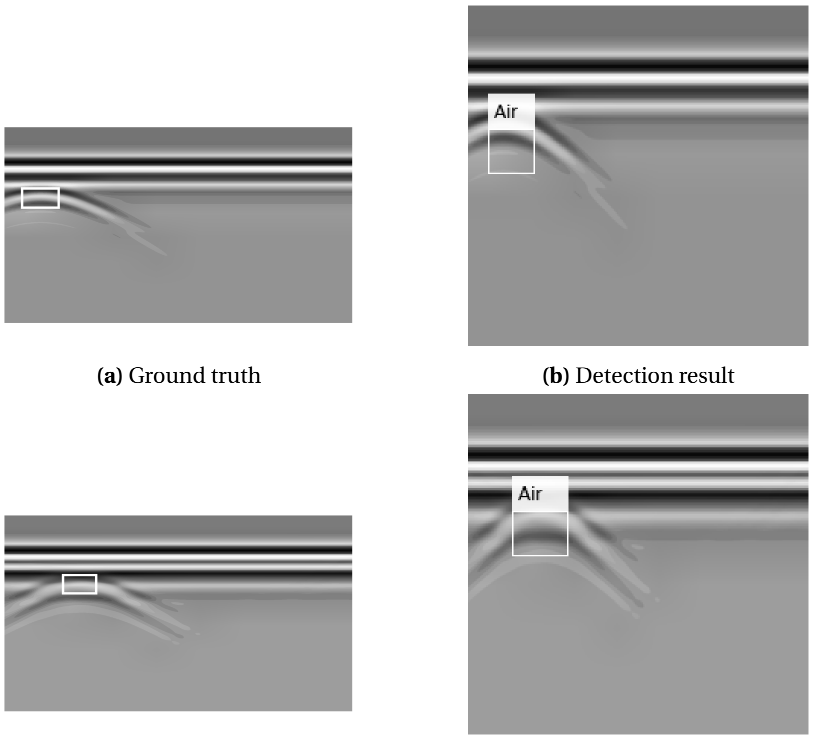
Future work

Future work involves sending a GPR survey robot inside the sewer pipeline. Currently, GPR survey is carried out by dragging the system on the ground surface. Since this cannot be possible inside the sewer. an autonomous model of the GPR system can be developed to move inside the sewer. By combining this with the ML algorithm for void detection, a fully automated sewer inspection system can be developed.

A Appendix 1 - RESULTS

A.1 Faster RCNN

Dataset A is used for testing and consists of single voids in a radargram composed of Air, Water, and Other material. 1000 images were trained and the results are obtained as seen in Figures A.1, A.2 and A.3.



(a) Ground truth **(b)** Detection result

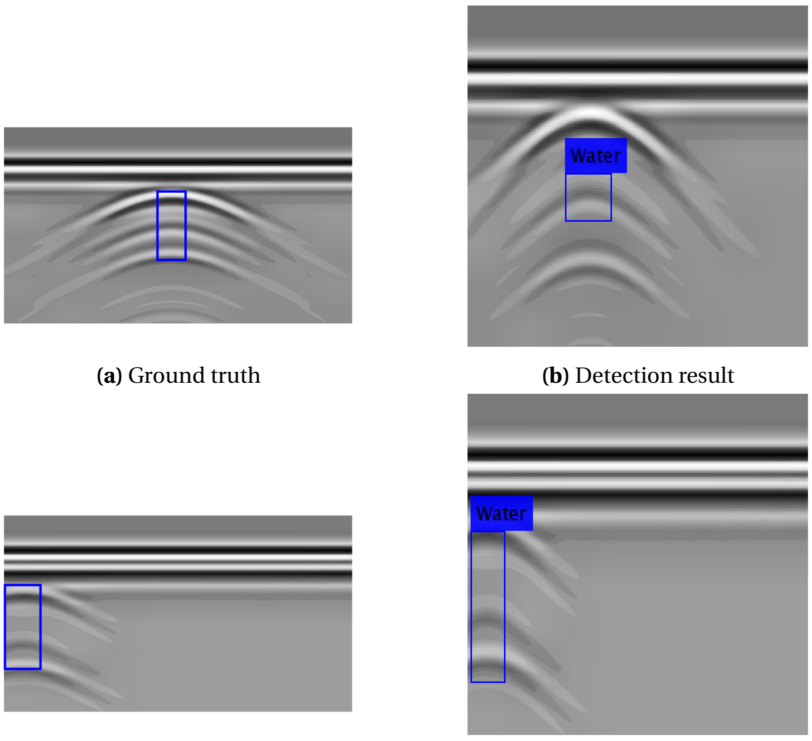
(c) Ground truth **(d)** Detection result

Figure A.1: Faster-RCNN True Positive results for air void: (a),(b) Dry sand (c),(d) Saturated sand

Figure A.1 shows the prediction of air voids in a homogeneous environment composed of dry sand and saturated sand. The bounding box prediction is accurate and as per the initial labels.

In the case of detection of water voids as seen in Figure A.2, the placement of the bounding boxes and the size of the box vary with each radargram detected. The position of the bounding box is not accurate since the apex of the hyperbola is not detected, and as seen in Figure A.2(b), one of the reflections of the void is detected instead.

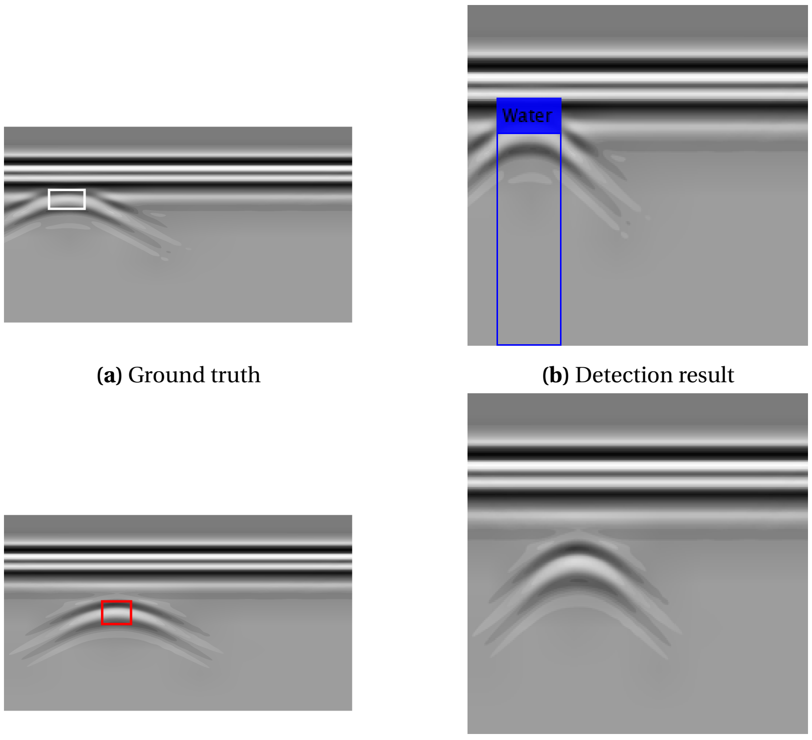
On applying the Faster-RCNN method, 55.3% of the voids in the dataset were not detected (i.e.) False negatives. As seen in Figure A.3, the example detected shows false negative and false positive due to incorrect detection.



(a) Ground truth **(b)** Detection result

(c) Ground truth **(d)** Detection result

Figure A.2: Faster-RCNN True Positive results for Water void: (a),(b) Dry sand (c),(d) Saturated sand



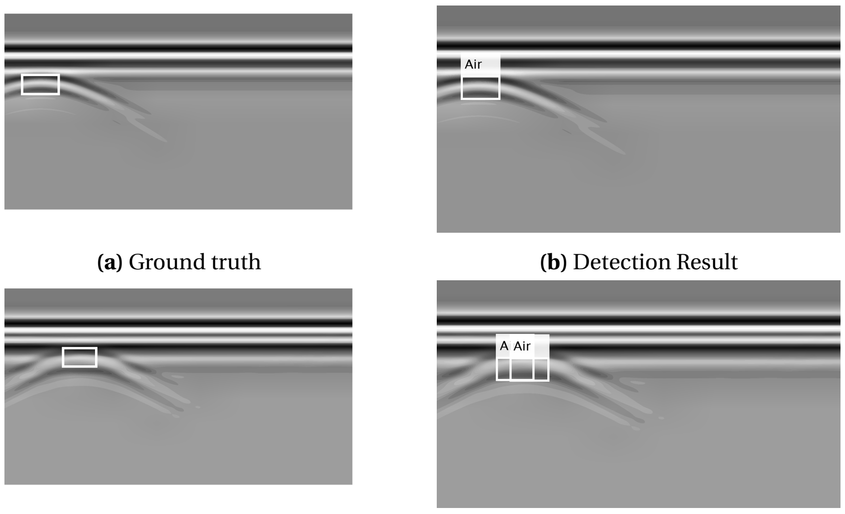
(a) Ground truth **(b)** Detection result

(c) **(d)**

Figure A.3: Faster-RCNN detection results: (c) False positive (d) False negative

A.2 YOLOV2

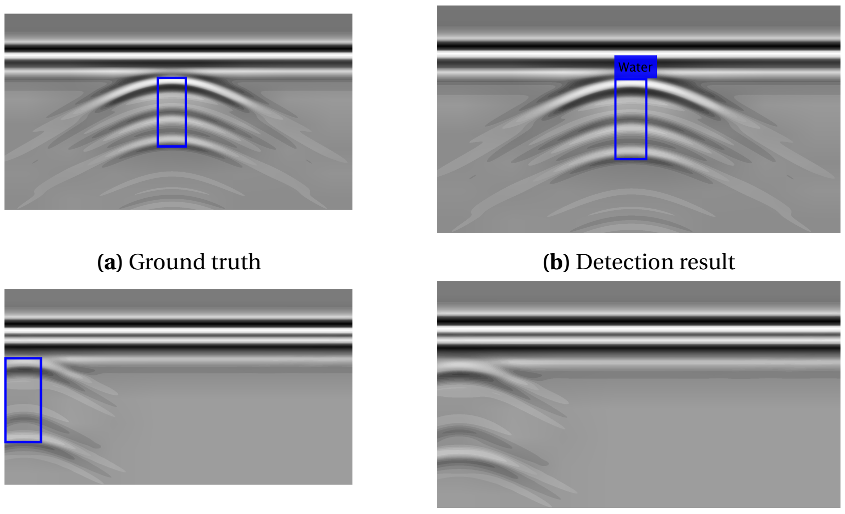
The results obtained for different classes are given in Figures A.4 and A.5. In a homogeneous soil environment filled with dry sand, the YOLOV2 detector works well for the ’Air’ and ’Water’ classes. In saturated sand, the detector detects two reflections of the ’Air’ class as seen in the example in Figure A.4(d). For class ’Water’ in saturated sand, there was no detection by the YOLOV2 detector in all of the radargrams.



(a) Ground truth **(b)** Detection Result

(c) Ground truth **(d)** Detection result

Figure A.4: YOLOV2 detection results applied in Dataset A for air void: (a),(b) Dry sand (c),(d) Saturated sand

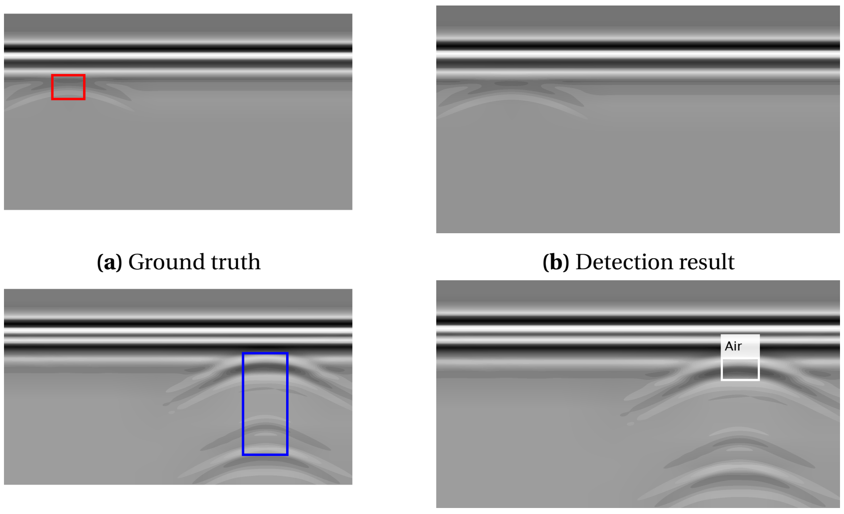


(a) Ground truth **(b)** Detection result

(c) Ground truth **(d)** Detection result

Figure A.5: YOLOV2 detection results applied in Dataset A for Water void: (a),(b) Dry sand (c),(d) Saturated sand

Figure A.6 (c) and (d) are water voids in saturated sand that are classified as ’Air’ since the other reflections were not detected. Image (b) was not detected since the apex of the hyperbola overlapped the horizontal line due to the reflections from the sewer pipe.

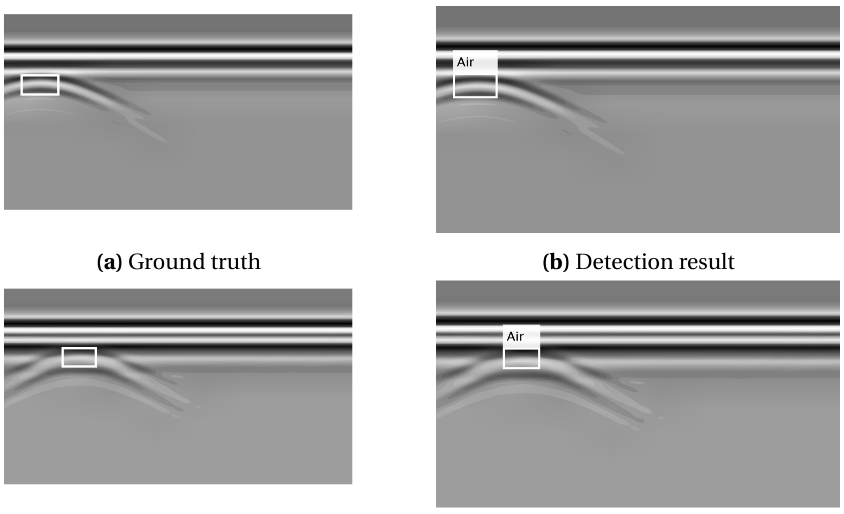


(a) Ground truth (b) Detection result (c) Ground truth (d) Detection result

Figure A.6: YOLOV2 detection results applied to Dataset A: (b) False negative (d) False positive

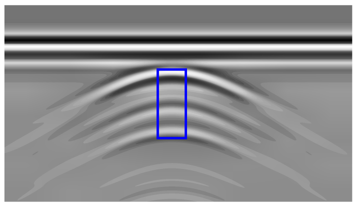
A.3 YOLOV3

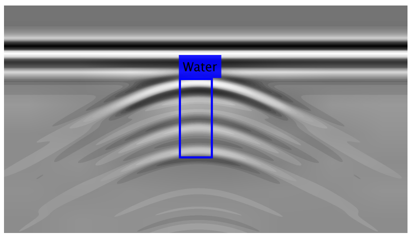
The detector is run through the images in Dataset A. The results of the three different classes are as seen in Figures A.7, A.8 and A.9. The detector performs very well for all the single voids, irrespective of the type of homogeneous sand.



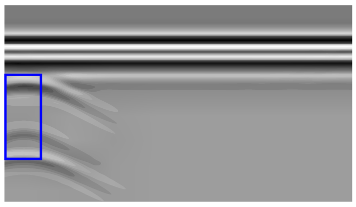
(a) Ground truth (b) Detection result (c) Ground truth (d) Detcetion result

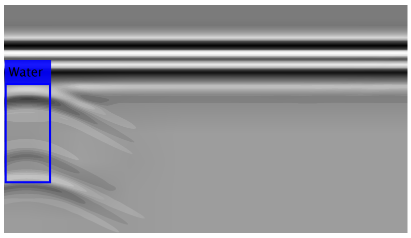
Figure A.7: YOLOV3 detection results applied to Dataset A for air void: (a),(b) Dry sand (c),(d) Saturated sand


(a) Ground truth



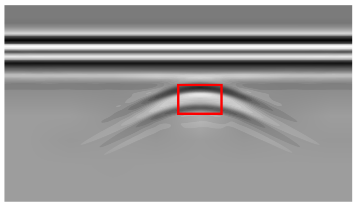
(b) Detection result

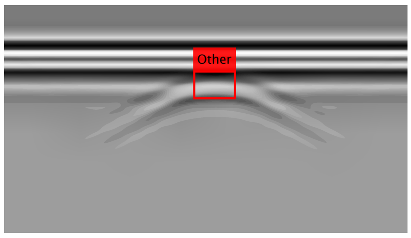

(c) Ground truth



(d) Detection result

Figure A.8: YOLOV3 detection results applied to Dataset A for Water void: (a),(b) Dry sand (c),(d) Saturated sand


(a) Ground truth



(b) Detection result

Figure A.9: YOLOV3 detection results applied to Dataset A for Other objects in saturated sand

B Appendix 2 - Background on Machine-Learning

B.1 Machine Learning

Machine Learning is a sub-field of Artificial Intelligence that makes machines imitate human intelligence by automatically training and improving data. It involves several statistical techniques that help the machines identify objects or make decisions without external human help. An initial dataset is provided to the model, trained over several iterations to finally discover patterns in the data, resulting in successful predictions. There are three main kinds of machine learning based on the type of input given to the model.

Supervised machine learning models are trained with datasets containing labels representing the ground truth of the input provided. This allows the models to learn and detect unlabelled inputs more accurately. For example, if a dataset labelled with cars and other objects is used for training, the model will detect cars independently from the inputs provided.

Unsupervised machine learning models are not given a labelled dataset. This type of machine learning detects the familiar patterns of the given data and can identify clusters or common structures. It is commonly used for clustering/grouping algorithms and cannot be used for object recognition or classification problems.

Reinforcement machine learning models train the algorithm through trial and error to make the right decision by granting an award. Based on the decision taken and feedback received, the model interacts with the environment and gets an award if the action is right. The model is trained by trying to increase the received award.

Every model has its advantages and applications. This research focuses more on the **Supervised Machine Learning** models in deep learning.

B.1.1 Deep Learning

Deep Learning is a subset of Machine Learning that predicts data by sending information through several layers. The following figure shows that the neural network architecture consists of three main parts: an input layer, a set of hidden layers, and an output layer. The term 'deep' refers to the number of hidden layers present in the architecture. They can range from 3 to around 150 layers deep.

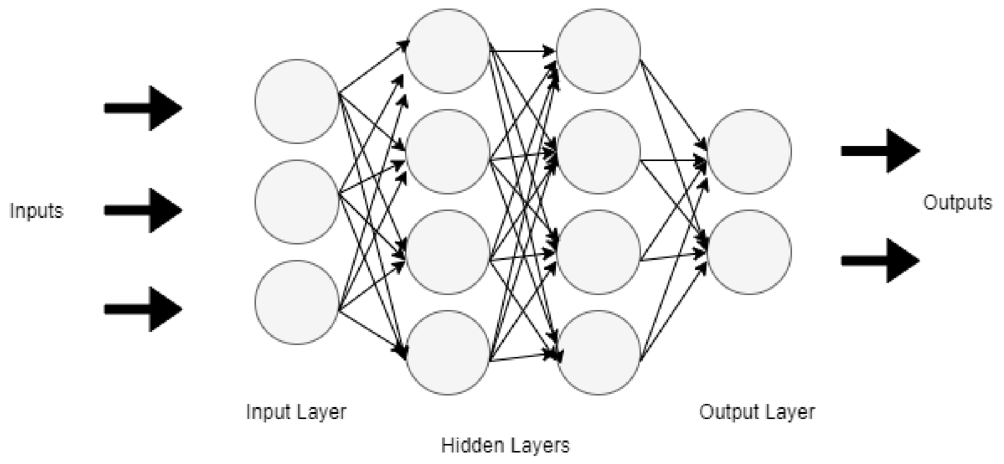


Figure B.1: Neural Networks

A neural network operates with several neurons or nodes in parallel, and each layer uses the previous layer as input, so all the nodes are interconnected. Each node has a weight associated with it, which is increased or decreased during the training process. The output layer has an activation function that maps the required outputs from the previous layer.

B.1.2 CNN

Convolutional Neural Networks (CNN) are one type of Neural network that is primarily used for object detection, pattern recognition, and natural language processing. The CNN architecture is a combination of two basic blocks:

1. The Convolution Block - consists of the Convolution layer and the Pooling layer. The feature extraction from the input images is done in this block.
2. The Fully Connected Block - consists of a fully connected neural network that performs classification based on the features extracted in the previous layers.

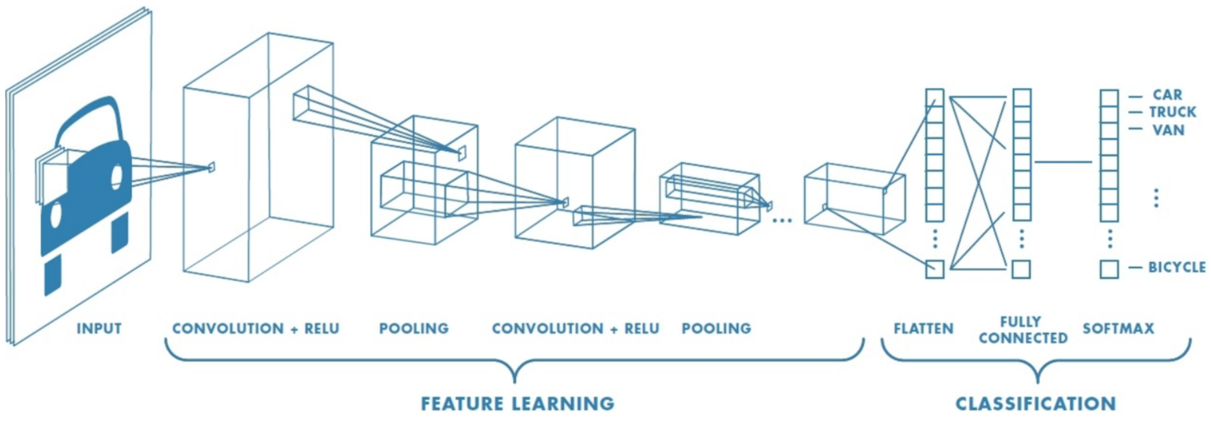


Figure B.2: Convolutional Neural Network architecture (Chatterjee, 2021)

Convolution Layer

The hidden layers in the network perform a convolution operation on the input to recognise patterns. A filter of dimensions 3x3 or 5x5 is introduced in the layers. These filters slide over the input image, multiplying the cells' values that correspond to the values of the filter matrix. The filter matrix slides over the entire image, and the resulting data are added together to form the output.

Padding operation is also done on the input image to avoid data loss due to the filter not detecting the corners and sides of the input matrix. Another row or column containing '0' is added to the input matrix at all the sides of the matrix. This results in equal weightage of the pixels during the convolution process. The result is a feature matrix with higher values when a feature is detected and lower values for areas of no interest.

As the number of convolution layers increases, the complexity of the feature extraction also increases. The first few layers perform edge detection on the input images. The middle few layers detect features/characteristics like eyes, nose, wheels, etc. The final few convolution layers learn to recognize complete shapes, which results in the object detection algorithms.

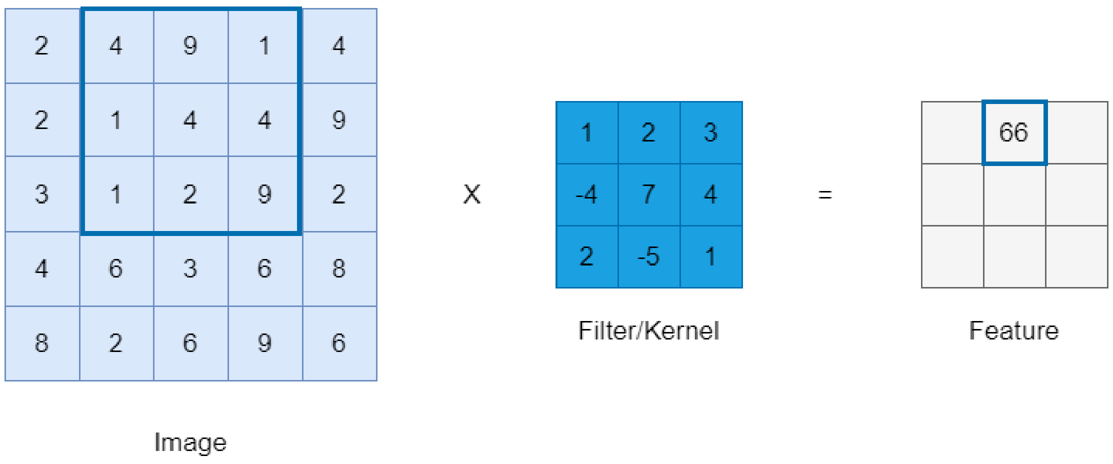


Figure B.3: Working principle of a Convolution Layer

Relu Ativation Layer

Rectified Linear Unit (RELU) is applied to the output of convolution. It's defined by the following function:

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases} \tag{B.1}$$

After the convolution is performed, pixels of values 0 or less than 0 imply that the feature of interest is not present there. Therefore the values are replaced by 0 and the original pixel value is retained for the other cases with values greater than 0.

Pooling Layer

The pooling layer is added between every convolution layer to reduce the size of the output matrix. This reduces the number of parameters and the spatial size, resulting in lower computation of the network. The size of the output matrix can then be reduced by reducing the size of the output matrix. The maximum or average value in a 2x2 matrix (for example) part of the output is chosen. Thus, the values that denote the presence of a feature are selected, thereby getting rid of other unwanted information.

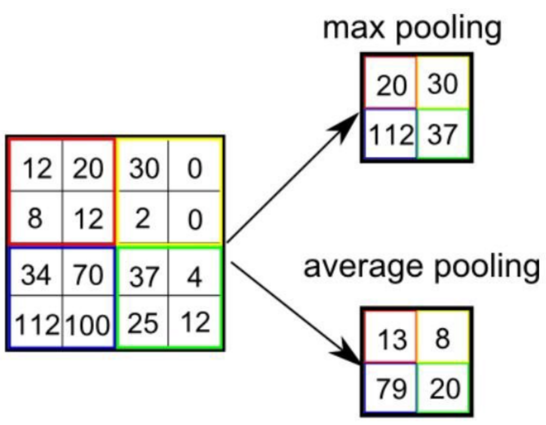


Figure B.4: Example of Max-pooling and Average-pooling (Chatterjee, 2021)

Fully Connected Layer

This layer forms the final part of the CNN architecture and consists of a simple neural network. It might consist of two or three hidden layers and an output layer with the softmax activation function, classifying the outputs. The output of the feature learning block is a multi-dimensional matrix converted to 1 x no. of classes matrix by the flatten layer.

C Appendix 3 - Code Repository

This appendix includes the description file of the code repository containing all the software created during the course of this research.

C.1 Pre-requisites

- Simulation of the radargrams using gprMax: http://www.gprmax.com/
- Creation of .in files for the radargrams and conversion of .out files to .png files were done using the python language
- All machine learning algorithms are developed in MATLAB: https://nl.mathworks.com/products/matlab.html

Additional Toolboxes required for the ML algorithms :

- MATLAB R2021a
- Computer Vision Toolbox
- Deep Learning Toolbox
- Image processing Toolbox
- Parallel Computing Toolbox
- Deep Learning toolbox model for ResNet-50 Network
- Computer Vision Toolbox™ Model for YOLO v3 Object Detection (Only compatible with R2021a)

C.2 GPR Radargrams

The radargrams are created using the gprMax software. Four different datasets are created for the simulation. Each containing 500 radargrams.

Dataset 1 contains single voids of Air or water (Dataset named as Singlevoids)

```
#sbatch run_gprmax_Single.sbatch 500 1
```

Dataset 2 contains single voids of Air, water or Other material (Dataset named as Dataset2)

```
#sbatch run_gprmax_Dataset2.sbatch 500 1
```

Dataset 3 contains multiple voids of Air, water or Other material (Dataset named as Dataset_multiplevoids)

```
#sbatch run_gprmax_Dataset2.sbatch 500 3
```

Dataset 4 contains single voids of Air, water or Other material in a heterogeneous sand environment (Dataset named as Dataset_het)

```
#sbatch run_gprmax_Dataset2.sbatch 500 1
```

C.3 MATLAB

The datasets are processed for machine learning algorithms, the data is divided into Training, Validation and Testing.

Faster - RCNN

```
#sbatch matlab_RCNN.sbatch
```

YOLOV2

```
#sbatch matlab_yolo.sbatch
```

YOLOV3

```
#sbatch matlab_yolo_v3.sbatch
```

C.4 Directory Tree

- Thesis (directory)
  - TESTING.m (matlab script to test detector on input images)
  - RCNN (directory)
    - RCNN_main.m (Contains the main matlab script for Faster-RCNN)
    - matlab_RCNN.sbatch
    - RCNNdetector_dataset2.mat (Faster-RCNN Detector)
    - Outputtable_1000.mat (Input dataset) **Supporting matlab scripts**
    - label_radargram.m
    - timezero_correction.m
    - validateInputData.m
  - YOLO (directory)
    - Yolo_main.m (Contains the main matlab script for YOLOV2)
    - Yolo_multiple.m (Contains the main matlab script for YOLOV2 detector for multiple voids)
    - matlab_yolo.sbatch
    - YOLOdetector_single.mat (YOLOV2 detector for both single and multiple voids)
    - YOLOdetector_multiple1.mat (YOLOV2 detector for multiple voids)
    - Outputtable_test1.mat (Input dataset of single voids for detection)
    - Outputtable_multiple_pro1.mat (Combined dataset of both single and multiple voids)
      Supporting matlab scripts
    - label_radargram.m
    - label_radargram_het.m
    - timezero_correction.m
    - validateInputData.m
    - preprocessData.m
    - helperSanitizeBoxes.m
      Directory containing the three channel images

    * Dataset_het (directory)
    * Dataset2 (directory)
    * Multiple (directory)
    * Singlevoid (directory)
- YOLOV3 (directory)
    * Yolo_v3.m (Contains the main matlab script for YOLOV3)
    * Yolo_v3_Mul.m (Contains the main matlab script for YOLOV3 for multiple voids)
    * Yolo_hyp_v3.m (Contains the main matlab script for YOLOV3 for machine learning + image processing)
    * horizremove.m (Code to remove the horizontal line due to sewer pipe)
    * matlab_yolo_v3.sbatch
    * YOLOdetector_v3.mat (YOLOV3 detector for both single voids)
    * YOLOdetector_mul_v3.mat (YOLOV3 detector for multiple voids)
    * YOLOdetector_v3_hyp.mat (YOLOV3 detector for dataset after image processing)
    * Outputtable_test1.mat (Input dataset of single voids for detection)
    * Outputtable_multiple_pro1.mat (Combined dataset of both single and multiple voids)
    * Outputtable_IP.mat (Dataset of radargrams containing single voids combined with image processing)

    Supporting matlab scripts
    * label_radargram.m
    * label_radargram_het.m
    * timezero_correction.m
    * validateInputData.m
    * preprocessData.m
    * helperSanitizeBoxes.m
    * configureTrainingProgressPlotter.m
    * createBatchData.m
    * displayLossInfo.m
    * generateTargets.m
    * modelGradients.m
    * piecewiseLearningRateWithWarmup.m
    * updatePlots.m

    Directory containing the three channel radargrams and radargrams after image processing.
    * Dataset_het (directory)
    * Dataset2 (directory)
    * Multiple (directory)
    * Singlevoid (directory)
- gprMax (directory)
    * contains .py files for creating the .in files .sbatch files to run the corresponding scripts
- Datasets (directory)
    * contains the datasets used for training, validation and testing.

C.5 Additional points

- The matlab scripts have the path as

  ```
  '/home/s2354241/Thesis/*Directory*',...
  ```

  Before proceeding, change the directory to your local folder.

  ```
     for i=1:height(HyperbolatrainingData)
     HyperbolatrainingData.imageFilename{i}
     = strrep(HyperbolatrainingData.imageFilename{i},...
         '/home/s2354241/Thesis/',...
         '/*Home directory*/Thesis');
  end
  ```

- Change the name of the main matlab file in the corresponding .sbatch files. example : in 'matlab_yolo_v3.sbatch', change the matlab script file to the required one.

- The number of CPU's and GPU's used in the course of the project varies. The following is used for the machine learning algorithms. It can be modified accordingly to accomodate more number of iterations or a larger dataset.

```
#SBATCH --cpus-per-task=16       # number of cores, 1
#SBATCH --gres=gpu:1          # number of gpus, 2
```

Bibliography

Ahrens, J. P., B. Geveci and C. C. Law (2005), ParaView: An End-User Tool for Large-Data Visualization, in *The Visualization Handbook*.

Al-Nuaimy, W., Y. Huang, M. Nakhkash, M. Fang, V. Nguyen and A. Eriksen (2000), Automatic detection of buried utilities and solid objects with GPR using neural networks and pattern recognition, **vol. 43**, no.2-4, pp. 157–165, doi:10.1016/S0926-9851(99)00055-5, cited By 185.

Chatterjee, H. S. (2021), A Basic Introduction to Convolutional Neural Network, [https://medium.com/@himadrisankarchatterjee/a-basic-introduction-to-convolutional-neural-network-8e39019b27c4] (URL accessed on 13-09-2021).

Daniels, J. (2000), Ground Penetrating Radar Fundamentals, doi:10.4133/1.2921864.

Davies, J. P., B. A. Clarke, J. T. Whiter and R. J. Cunningham (2001), Factors influencing the structural deterioration and collapse of rigid sewer pipes, *Urban Water*, **vol. 3**, pp. 73–89, ISSN 1462-0758, doi:10.1016/S1462-0758(01)00017-6.

Delft, M. V. (2019), Towards feature-based underground void detection with ground penetrating radar from within sewers using image processing. [http://essay.utwente.nl/79979/]

Dou, Q., L. Wei, D. Magee and A. Cohn (2017), Real-Time Hyperbola Recognition and Fitting in GPR Data, **vol. 55**, no.1, pp. 51–62, doi:10.1109/TGRS.2016.2592679, cited By 61.

GSSI (2021), What Is GPR?, [https://www.geophysical.com/whatisgpr] (URL accessed on 03-09-2021).

HPC (2021), High Performance Computing, [http://korenvliet.ewi.utwente.nl/wiki/doku.php?id=wiki:slurm:start] (URL accessed on 17-09-2021).

Iandola, F. N., M. Moskewicz, K. Ashraf, S. Han, W. Dally and K. Keutzer (2016), SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size, *ArXiv*, **vol. abs/1602.07360**.

Kuo, S., L. Zhao, H. Mahgoub and P. F. Suarez (2005), Investigation of Ground Penetrating Radar for Detection of Leaking Pipelines under Roadway Pavements and Development of Fiber-Wrapping Repair Technique.

Lei, W., F. Hou, J. Xi, Q. Tan, M. Xu, X. Jiang, G. Liu and Q. Gu (2019), Automatic hyperbola detection and fitting in GPR B-scan image, *Automation in Construction*, **vol. 106**, p. 102839, ISSN 0926-5805, doi:https://doi.org/10.1016/j.autcon.2019.102839. [https://www.sciencedirect.com/science/article/pii/S0926580519301347]

Li, Y., Z. Zhao, Y. Luo and Z. Qiu (2020), Real-Time Pattern-Recognition of GPR Images with YOLO v3 Implemented by Tensorflow, *Sensors (Basel, Switzerland)*, **vol. 20**.

Maruddani, B. and E. Sandi (2019), The Development of Ground Penetrating Radar (GPR) Data Processing, *International Journal of Machine Learning and Computing*, **vol. 9**, pp. 768–773, doi:10.18178/ijmlc.2019.9.6.871.

McGrath, T. J., E. T. Selig, M. C. Webb and G. V. Zoladz (1999), Pipe Interaction With the Backfill Envelope, tech Report. [https://rosap.ntl.bts.gov/view/dot/48814]

Michael J Garbade (2021), Understanding K-means Clustering in Machine Learning, [https://towardsdatascience.com/

understanding-k-means-clustering-in-machine-learning-6a6e67336aa1] (URL accessed on 29-09-2021).

National Geographic, K. T. (2021), Guatemala Sinkhole Created by Humans, Not Nature, [https://www.nationalgeographic.com/science/article/100603-science-guatemala-sinkhole-2010-humans-caused] (URL accessed on 02-09-2021).

Noreen, T. and U. S. Khan (2017), Using pattern recognition with HOG to automatically detect reflection hyperbolas in ground penetrating radar data, in *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pp. 1–6, doi:10.1109/ICECTA.2017.8252064.

Núñez-Nieto, X., M. Solla, P. Gómez-Pérez and H. Lorenzo (2014), GPR signal characterization for automated landmine and UXO detection based on machine learning techniques, **vol. 6**, no.10, pp. 9729–9748, doi:10.3390/rs6109729, cited By 45.

Pham, M.-T., L. Courtrai, C. Friguet, S. Lefèvre and A. Baussard (2020), YOLO-Fine: One-Stage Detector of Small Objects Under Various Backgrounds in Remote Sensing Images, **vol. 12**, no.15, ISSN 2072-4292, doi:10.3390/rs12152501. [https://www.mdpi.com/2072-4292/12/15/2501]

Pham, M.-T. and S. Lefèvre (2018), Buried Object Detection from B-Scan Ground Penetrating Radar Data Using Faster-RCNN, in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 6804–6807, doi:10.1109/IGARSS.2018.8517683.

Redmon, J. and A. Farhadi (2017), YOLO9000: Better, Faster, Stronger, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525.

Redmon, J. and A. Farhadi (2018), YOLOv3: An Incremental Improvement, *ArXiv*, **vol. abs/1804.02767**.

Ren, S., K. He, R. Girshick and J. Sun (2015), Faster r-cnn: Towards real-time object detection with region proposal networks, *Advances in neural information processing systems*, **vol. 28**, pp. 91–99.

Ristic, A. and D. Petrovacki (2007), An underground utility detection technology – our experiences.

Schlumberger (2021), Common Midpoint, [https://glossary.oilfield.slb.com/en/terms/c/common_midpoint] (URL accessed on 21-09-2021).

Singh, N. and M. Nene (2013), Buried object detection and analysis of GPR images: Using neural network and curve fitting, doi:10.1109/AICERA-ICMiCR.2013.6576024, cited By 26.

Skartados, E., I. Kostavelis, D. Giakoumis, A. Simi, G. Manacorda, D. Ioannidis and D. Tzovaras (2018), Ground Penetrating Radar Image Processing Towards Underground Utilities Detection for Robotic Applications, in *2018 International Conference on Control, Artificial Intelligence, Robotics Optimization (ICCAIRO)*, pp. 27–31, doi:10.1109/ICCAIRO.2018.00013.

Sonoda, J. and T. Kimoto (2018), Object Identification form GPR Images by Deep Learning, in *2018 Asia-Pacific Microwave Conference (APMC)*, pp. 1298–1300, doi:10.23919/APMC.2018.8617556.

TISCALI (2021), TISCALI, [https://www.ram.eemcs.utwente.nl/research/projects/tiscali] (URL accessed on 16-03-2021).

Travassos, X., S. Avila and N. Ida (2018), Artificial Neural Networks and Machine Learning techniques applied to Ground Penetrating Radar: A review, *Applied Computing and Informatics*, doi:10.1016/j.aci.2018.10.001, cited By 12.

Travassos, X. L. . and M. F. Pantoja (2019), *Ground Penetrating Radar*, Springer International Publishing, Cham, pp. 987–1023, ISBN 978-3-319-26553-7, doi:10.1007/978-3-319-26553-7_9.

Warren, C. (2018), gprMax, [https://github.com/gprMax/gprMax/tree/master/tools/Jupyter_notebooks].

Warren, C. and A. Giannopoulos (2021), gprMax Software, [https://www.gprmax.com/about.shtml] (URL accessed on 10-09-2021).

Warren, C., A. Giannopoulos and I. Giannakis (2015), An advanced GPR modelling framework: The next generation of gprMax, in *2015 8th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*, pp. 1–4, doi:10.1109/IWAGPR.2015.7292621.

Wirahadikusumah, R., D. Abraham, T. Iseley and R. Prasanth (1998), Assessment technologies for sewer system rehabilitation, *Automation in Construction*, **vol. 7**, pp. 259–270, doi:10.1016/S0926-5805(97)00071-X.

Zuiderveld, K. (1994), Contrast Limited Adaptive Histogram Equalization, in *Graphics Gems*.