COGNITIVE GPR FOR SUBSURFACE SENSING BASED ON EDGE COMPUTING AND DEEP REINFORCEMENT LEARNING

By

Maxwell Momanyi Omwenga

Dalei Wu
Associate Professor of Computer Science
(Chair)

Yu Liang
Professor of Computer Science
(Co-Chair)

Mina Sartipi
Professor of Computer Science
(Committee Member)

Lan Gao
Associate Professor of Statistics
(Committee Member)

Dryver Huston
Professor of Mechanical Engineering
(Committee Member)

COGNITIVE GPR FOR SUBSURFACE SENSING BASED ON EDGE COMPUTING AND DEEP REINFORCEMENT LEARNING

By

Maxwell Momanyi Omwenga

A Thesis Submitted to the Faculty of the
University of Tennessee at Chattanooga
in Partial Fulfillment of the Requirements of the
Doctor of Philosophy in Computer Science

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

August 2021

ABSTRACT

Ground penetrating radars (GPRs) have been extensively used in many industrial applications, such as coal mining, structural health monitoring, subsurface utilities detection and localization, and autonomous driving. Most of the existing GPR systems are human-operated due to the need for experience in operation configurations based on the interpretation of collected GPR data. To achieve the best subsurface sensing performance, it is desired to design an autonomous GPR system that can operate adaptively under varying sensing conditions. In this research, first, a generic architecture for cognitive GPRs based on edge computing is studied. The operation of cognitive GPRs under this architecture is formulated as a sequential decision process. Then a cognitive GPR based on 2D B-Scan image analysis and deep Q-learning network (DQN) is investigated. A novel entropy-based reward function is designed for the DQN model by using the results of subsurface object detection (via the region of interest identification) and recognition (via classification). Furthermore, to acquire a global view of subsurface objects with complex shape configurations, 2D B-Scan image analysis is extended to 3D GPR data analysis termed "Scan Cloud." A scan cloud enabled cognitive GPR is studied based on an advanced deep reinforcement learning method called deep deterministic policy gradient (DDPG) with a new reward function derived from 3D GPR data. The proposed methods are evaluated using GPR modeling and simulation software called GprMax. Simulation results show that our proposed

cognitive GPRs outperform other GPR systems in terms of detection accuracy, operating time, and object reconstruction.

DEDICATION

This work is dedicated to my loving wife, Evelyn, and our adorable children, Israel, Priscilla, and Brooklyn.

ACKNOWLEDGMENTS

To begin, I would like to acknowledge my beautiful wife, Evelyn Omwenga and our adorable children Israel K. Omwenga, Priscilla P. Nyaboke, and Brooklyn G. Gesare. You were here for all of the ups and downs. I could not have made it this far without my sweethearts by my side. Furthermore, I would like to thank my advisor Dr. Wu, co-advisor Dr. Liang, and everyone in the networked intelligence lab for giving me the opportunity to conduct research in an academic environment and facilitate my growth as a researcher. In addition, I would like to thank Dakila Ledesma, Grace Nansamba, Evelyn Namugwanya, Dr. Amani Altarawneh, Mehran Gafhari, Sai Medury, and the rest of the Ph.D. students. From the very beginning, they challenged me, which enabled me to grow in the field of machine learning. Without them, I would not have been able to undertake a research project as challenging as this. I also would like to include a special thank you to my thesis committee. The road to defense has been rocky, but we finally made it.

To my dad, Meshack N. Omwenga, thank you for speaking a blessing on me 21 years ago. You prayed that I become a doctor by 32 years old. I am excited at 39 years young I made it.

Dr. Joseph Kizza and Dr. Immaculate Kizza, thank you for the continuous encouragement and sound wisdom. Finally, to my family and friends, thank you for the prayers, the earnest presence, and for loving me through it all. I love you all.

TABLE OF CONTENTS

ABSTRACT iv

DEDICATION vi

ACKNOWLEDGMENTS vii

LIST OF TABLES xi

LIST OF FIGURES xii

LIST OF SYMBOLS xiv

LIST OF ABBREVIATIONS xvii

CHAPTER

1. INTRODUCTION 1

    1.1 Problem Statement 2
    1.2 Significance of the study 2
    1.3 Objectives of the study 2
    1.4 Contributions 3
    1.5 Organization of thesis 3

2. BACKGROUND 5

    2.1 GPR and GPR Data 5
    2.2 Deep Learning Based GPR Analysis 7
    2.3 Edge Computing 7
    2.4 Deep Reinforcement Learning 9

3. SYSTEM ARCHITECTURE 11

    3.1 Cognitive GPR System 11
    3.2 The Proposed System Architecture 14

3.3 Dynamic Computation Task Scheduling . . . 15
3.3.1 The Communication Models of the GPR Front End . . . 15
3.3.2 The Communication Models of the GPR Back End . . . 17
3.3.3 The Proposed Offloading Policy . . . 17
3.4 Service Migration . . . 19
3.5 Performance evaluation . . . 20
3.6 Summary . . . 23
3.7 Future Work . . . 25

4. GPR DATA ACQUISITION AND AUGMENTATION . . . 26

4.1 GPR Data Acquisition Through Modeling Simulation . . . 26
4.1.1 Introduction . . . 26
4.1.2 GprMax Configuration . . . 26
4.2 Preprocessing of Real-World GPR Sensory Data . . . 29
4.3 GPR Data Augmentation Based on Generative Adversarial Network (GAN) . . . 29

5. COGNITIVE GPR BASED ON B-SCAN DATA . . . 33

5.1 The Proposed System Model and Architecture . . . 34
5.1.1 The Original Concept of Cognitive GPR . . . 34
5.1.2 MDP Formulation of Cognitive GPR Operation . . . 35
5.1.3 The Architecture of the Proposed AC-GPR . . . 36
5.2 The Proposed DRL Approach . . . 37
5.2.1 Reward Function . . . 37
5.2.1.1 Reward Function Based on RoI Detection . . . 38
5.2.1.2 Reward Function Based on Subsurface Object Classification . . . 42
5.2.2 DRL Algorithm . . . 43
5.3 Performance Evaluation and Discussion . . . 47
5.3.1 Experiment Settings . . . 47
5.3.1.1 GprMax Simulator . . . 47
5.3.1.2 RoI Detection . . . 48
5.3.1.3 Object Classification . . . 48
5.3.1.4 DQN . . . 49
5.3.2 Performance Results . . . 50
5.3.2.1 Detection Accuracy vs. Noisy B-Scan Data . . . 50
5.3.2.2 Detection Accuracy vs. Object Diameters and Burial Depths . . . 51
5.3.2.3 Classification Accuracy vs. RoI Dimension . . . 54
5.3.3 Convergence Analysis . . . 55
5.3.3.1 Comparison Between Reward Functions . . . 55
5.3.3.2 Time Steps to Reach Convergence . . . 56
5.4 Summary . . . 56
5.5 Future Work . . . 57

6. COGNITIVE GPR BASED ON SCAN CLOUD DATA 58

6.1 The Proposed System Model and Architecture 59
6.1.1 MDP Formulation of Subsurface Detection 59
6.1.2 The Architecture of the Scan Cloud based Cognitive GPR 60
6.2 The Proposed Scan Cloud Approach 61
6.2.1 Reward Function 61
6.2.1.1 Reward Based on Amplitude 62
6.2.1.2 Reward Based on Scan Cloud 63
6.2.1.3 Reward Based on Subsurface Object Classification 66
6.2.2 DDPG Algorithm 68
6.3 Performance Evaluation and Discussion 70
6.3.1 Experiment Settings 70
6.3.1.1 GprMax Simulator 70
6.3.1.2 Amplitude analysis 70
6.3.1.3 RoI Detection 70
6.3.1.4 Object Shape Classification 71
6.3.1.5 DDPG 71
6.3.2 Performance Results 71
6.3.2.1 Object Reconstruction 71
6.3.2.2 Detection Accuracy vs. Heterogeneous Medium 72
6.3.2.3 Detection Accuracy vs. Object Size and Burial Depths 74
6.3.2.4 Classification Accuracy vs. Scan Cloud Points 74
6.3.3 Convergence Analysis 74
6.3.3.1 Comparison Between Reward Functions 74
6.3.3.2 Time Steps to Reach Convergence 76
6.4 Summary 76
6.5 Future Work 77

7. CONCLUSIONS 81

REFERENCES 83

VITA 93

LIST OF TABLES

4.1 Dielectric constant [58] 29
5.1 Architecture parameters 49
5.2 Deep Q network settings 50
5.3 Time steps convergence comparison 56
6.1 Architecture parameters 79
6.2 DDPG network settings 79
6.3 Classification accuracy of reward function variations 80
6.4 Time steps comparison 80

LIST OF FIGURES

1.1 1D A-Scan, 2D B-Scan and 3D C-Scan 1
3.1 Successive action choice procedure of dynamic subsurface object detection 12
3.2 The operational flows of a traditional GPR 13
3.3 A cognitive GPR 13
3.4 The architecture of the proposed cognitive GPR 14
3.5 Service migration architecture 21
3.6 GPR data collection on MLK 23
3.7 Edge computing and GPR data pre-processing performance 24
3.8 Service migration network performance 24
4.1 Homogeneous soil model 28
4.2 Heterogeneous soil model 28
4.3 B-Scan pre-processing steps 30
4.4 Proposed GAN Architecture 31
4.5 GAN generated B-Scan images 32
5.1 The iterative operational process of the proposed DRL-enabled AC-GPR 35
5.2 Region of interest detection process 39
5.3 Sample B-Scan images of the dataset 51
5.4 AC-GPR performance with noisy B-Scan images with varying clay factions 52
5.5 AC-GPR performance with noisy B-Scan images with varying sand fractions 52
5.6 AC-GPR performance with varying object diameter 53
5.7 AC-GPR performance with varying object burial depth 53
5.8 Classification accuracy vs. RoI dimension 54
5.9 Cumulative reward vs. time steps of different ε-greedy policies and rewards types . . . 55

6.1 Proposed architecture 60
6.2 Region of interest from various underground object shapes 64
6.3 The scan cloud formulated using tetrahedral elements 66
6.4 X-shaped scan cloud (left) with its corresponding signature (right) 67
6.5 The comparison of reconstruction results between other methods and our network . . . 72
6.6 Scan cloud performance with varying bulk density 73
6.7 Scan cloud performance with varying volumetric water 73
6.8 Scan cloud performance with varying object sizes 75
6.9 Scan cloud performance with varying burial depth 75
6.10 The scan cloud recognition accuracy based on various grid size resolutions 76

LIST OF SYMBOLS

$s_t \equiv \langle t, \Psi_t \rangle$, The state of ACGPR agent at time step t $_t$ is the state of the environment Ψ_t is the state of the agent such as position and battery energy status

s'_t, The succeeding state s_t of the agent after time step t

$a_t \equiv \langle \xi_t, \vec{v}_t, \vec{p}_t \rangle$, The action ACGPR agent will take, which involve moving direction ($\xi_t \in [0, 2\pi]$), velocity, and the configuration (e.g., frequency and orientation) of GPR

$\varepsilon \in [0, 1]$, The probability that a random action is taken

$r_t \in \Re$, Reward at time step t

$r_t^i \in \Re$, Rényi entropy-based reward

$r_t^m \in \Re$, Shannon entropy-based reward

$\gamma \in \Re$, Discount factor, of the target Q-value

D and N, Replay memory and its dimension

$Q(\theta)$, Evaluation Q-network that's updated every episode, where θ is the network parameter

$\hat{Q}(\hat{\theta})$, Target Q-network which is updated every E episodes, where $\hat{\theta}$ is the network parameter

π^*, Optimum policy

τ, Time-interval of the reflection signal from the object

$\langle \Delta\tau, \Delta\varpi \rangle$, Dimension of the region-of-interest in a B-Scan

ε, Dielectric constant

$z_i(\tau)$, $z_j(\varpi)$, Power normalization wrt time and scan axis

$E_\alpha(\tau)$, $E_\alpha(\varpi)$, GPR data singularity wrt time and scan axis

$\Im 1^*$, $\Im 2^*$, Optimum OTSU thresholds

$Z(\tau)$, Reflection signal

E_{min}, E_{max}, Minimum and maximum Rényi entropy

x_t, y_t, z_t, The $x, y,$ and z coordinates of the received pulse signal

φ, Amplitude

$|m_a(i)|$, Signal Enveloper

r_t^{amp}, Amplitude reward

r_t^{roi}, Region of interest (RoI) reward

r_t^{sc}, Scan cloud object classification reward

θ^Q, Q network

θ^μ, Deterministic policy function

$\theta^{Q'}$, Target Q network

$\theta^{\mu'}$, Target policy network

$r_t \in \Re$, Reward at time step t

$\mathscr{G}1^*, \mathscr{G}2^*$, 3D optimum OTSU thresholds

$E_\alpha(x)$ Entropy quantification with respect to the x axis

$E_\alpha(y)$ Entropy quantification with respect to the y axis

$E_\alpha(z)$, Entropy quantification with respect to the z axis

$\mathscr{X}^*, \mathscr{Y}^*$, and $\mathscr{Z}^*$, Sets of intermediate RoIs scan cloud points with respect to $x, y,$ and z axis respectively

V^{roi}, Volume of the region-of-interest formulated by tetrahedral elements

LIST OF ABBREVIATIONS

AC-GPR, Autonomous Cognitive GPR

ANN, Artificial Neural Network

CAPES, Computingaccess Points Edge Server

CNN, Convolution Neural Network

CPU, Central Processing Unit

DDPG, Deep Deterministic Policy Gradient

DRL, Deep Reinforcement Learning

DQN, Deep Q-Network

EC, Edge Computing

EM, Electromagnetic

FDTD, Finite Difference Time Domain

GAN, Generative Adversarial Network

GANs, Generative Adversarial Networks

GIS, Geographic Information System

GMM, Gaussian mixture model

GPR, Ground Penetrating Radar

MDP, Markov Decision Process

PVC, Polyvinylchloride

R-CNN, Region-based Convolutional Network

RL, Reinforcement Learning

QoS, Quality of Service

SDN, Software-defined Networking

TCP, Transmission Control Protocol

CHAPTER 1

INTRODUCTION

Ground penetrating radars (GPRs) have been extensively used in many industrial applications, such as coal mining, structural health monitoring, subsurface utilities detection and localization, and autonomous driving [1, 2]. In subsurface detection applications, a GPR system transmits an electromagnetic wave into the ground at several spatial positions and receives the reflected signal to form GPR data, called A-Scans, B-Scans, and C-Scans with a different number of dimensions [1, 3]. As shown in Fig. 1.1 a single radar trace, or waveform, is called A-Scan, which is one dimensional signal. A set of consecutive radar waveforms along a particular direction (for example, x-axis in Fig. 1.1) can be assembled into a two-dimensional image called B-Scan. Multiple B-Scan images along a particular direction (for example, y-axis in Fig. 1.1) can be composed into a C-Scan. In this thesis, we consider both A-Scan and B-Scan because they are the most commonly used GPR data modalities for subsurface object detection.

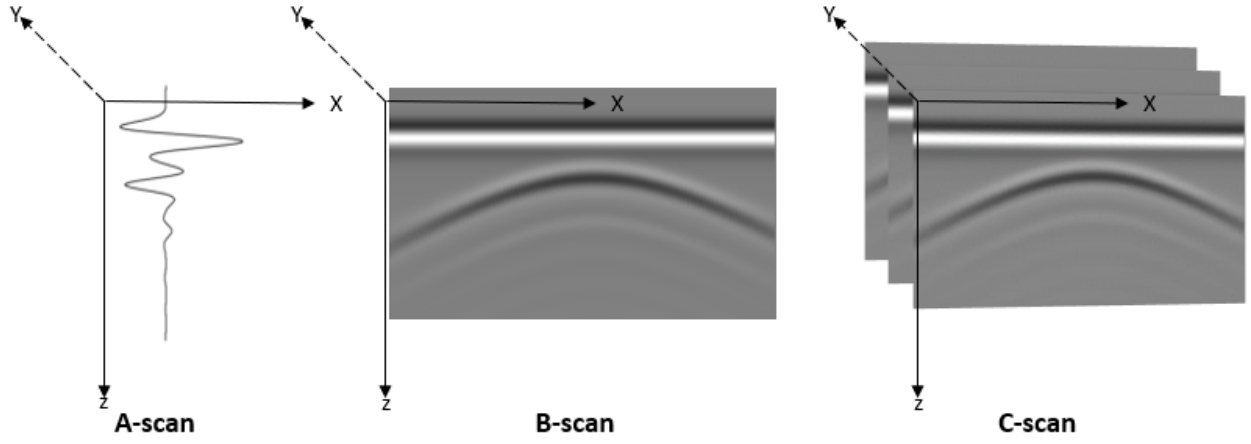


Figure 1.1 1D A-Scan, 2D B-Scan and 3D C-Scan

1.1 Problem Statement

Although GPRs are effective in many nondestructive applications, most of the existing GPR systems are human-operated due to the need for experience in operation configurations based on the interpretation of collected GPR data. GPR-based subsurface survey is complicated as various sensing environments and subsurface objects have dissimilar features. In nearly all existing GPR systems, GPR data processing is performed off-line where data from the field are collected and stored, and then post-processed on a computer after the scanning. Such a processing approach is time-consuming and lacks adaptivity. To achieve optimal sensing performance, it is desired to design an autonomous cognitive GPR system based on reinforcement learning, that can operate adaptively under varying sensing conditions. Specifically, the system is able to adaptively move with a robotic platform and adjust its operational parameters through real-time interaction with the sensing environment.

1.2 Significance of the study

The significance of this study is to demonstrate that reinforcement learning models can be successfully applied to GPR data collection and interpretation. Furthermore, this method can be used to classify and reconstruct underground objects.

1.3 Objectives of the study

The objectives of this study are three fold. The first is to establish reinforcement learning architectures for intelligent cognitive GPR sensing. The second is to develop an edge computing framework, capable of supporting service migration. The final is to extend the local approach for the detection and recognition of subsurface objects to a global approach.

These are the research questions that we want to answer:

- Can deep reinforcement learning (DRL) improve subsurface object detection performance of self-driving robotic GPR across different unknown geographical environment?
- What are the environmental and GPR calibration factors that affect whether GPR sensing data is good or bad? and how can these factors be used to aid DRL agent to accurately detect subsurface object and its material composition?

- What are the different features of the A-Scans and B-Scans collected from varying dielectric properties and how can these differences be used to aid the GPR users collect clean GPR data?

- Do cognitive GPRs based on edge computing and DRL achieve competitive performance when compared to other GPR sensing techniques?

1.4 Contributions

In this thesis, a deep reinforcement learning (DRL) method will be investigated for cognitive GPR sensing for underground object detection and recognition. Simulated data that mimic the real world data will be considered as input to the DRL to generate adaptive commands. Based on the feature analysis of GPR data, a novel reward function and a 3D scan cloud are proposed. Algorithms for cognitive GPR based on 2D B-Scan data and 3D scan cloud data are developed. To the best of our knowledge, this is the first work based on DRL for the development of autonomous cognitive GPR. The main contribution of this thesis can be summarized as follows.

- By formulating cognitive subsurface object detection as a Markov decision problem, a deep reinforcement learning framework is established to resolve the problem.

- A deep Q-network (DQN) and a deep deterministic policy gradient (DDPG) algorithms with a novel reward functions that combines rewards from amplitude analysis, region of interest (RoI) identification, and object classification are proposed.

- An edge computing framework for delay sensitive applications is proposed.

- To show the efficacy of the proposed frameworks, simulation based validations are performed on real-time GPR data from GprMax simulator by combining DRL with GPR operation modeling.

1.5 Organization of thesis

The remainder of this thesis is laid out as follows: We study the origin and applications of cognitive GPR in Chapter 2. The system model and architecture and algorithms are presented in

Chapter 3. Chapter 4 discusses the GPR data acquisition and augmentation processes. A cognitive GPR based on B-Scan image approach for subsurface object detection is presented in Chapter 5. In Chapter 6, an investigation into continuous subsurface object detection to generate object shape and orientation is presented. Finally, Chapter 7 concludes the thesis.

CHAPTER 2

BACKGROUND

2.1 GPR and GPR Data

Portable GPRs, such as handheld or drone-borne GPRs, have been extensively used in many industrial applications, such as coal mining, structural health monitoring, subsurface utility detection, and localization [4, 5]. GPR is a non-destructive evaluation technique for effective assessment of subsurface conditions in large dielectric bodies, such as city streets, by launching and receiving electromagnetic waves from the same side of a structure. The location and nature of subsurface objects can be characterized by collecting and analyzing reflected and scattered waves [1, 3].

GPR-based subsurface survey is complicated as various sensing environments and subsurface targets have dissimilar features. In an actual GPR survey, GPR sensing quality could be affected by many factors, including environmental factors, such as soil dielectric properties, environment noise, clutter, multi-path effects, combined near and far field effects, and GPR operational system parameters, such as wavelength (or frequency), waveform, polarization, wave timing, and transmitter and receiving antennas direct coupling, etc. In addition, the subsurface objects have different structural features and electromagnetic properties that affect GPR EM wave propagation differently. Hence processing GPR images and extracting information of interest are challenging and involve a series of sophisticated steps. In nearly all existing GPR systems, the GPR data processing is performed off-line where the data are collected on field and stored, and then post-processed on a central computer after the scanning. Such a processing approach is a dummy process as it does not adaptively adjust GPR operations in the survey.

The Internet of Things (IoT) and robotics fields are linking up to forge The Internet of Robotic Things (IoRT) [6] where smart gadgets can screen the events occurring around them,

intertwine their sensor information, utilize local and/or distributed intelligence to autonomously plan the course of action(s) to control gadgets in the physical world. For example, the complexity of autonomous driving on urban roadways is addressed by applying RL method [7] and DRL [8].

Autonomous GPRs have been extensively studied in field robotics, remote sensing and intelligent transportation systems [9, 10, 11, 12, 13, 14]. Cornick et al. [9] describe a localizing GPR system fused with GPS, LiDAR and camera hooked at the bottom of an autonomous vehicle for autonomous ground vehicle localization. The system allows real-time creation of single-track maps with online data processing, as well as real-time localization of the vehicle to a prior map. In [10], the authors developed an autonomous robotic system employing GPR probing of glacier surfaces for void detection in ice. Supervised machine learning with pre-trained models was applied to automatically classify data into crevasse and crevasse-free classes.

Various machine learning techniques have been applied to analyze B-Scans for object detection using Faster R-CNN [11], and incorporating frequency domain features in classification with augmented GPR data synthesized by the Generative Adversarial Network (GAN) [12]. Foessel et al. [13] describes a sled-mounted GPR integrated with position and latitude instrument for autonomous search for Antarctic meteorites. Using non-destructive evaluation (NDE) sensors, histogram of gradient (HOG) and Naive Bayes classifier, [14] developed an autonomous robotic system for real-time bridge deck inspection that generates condition map. Although the aforementioned systems have used robotic systems to move GPR scanners, GPR movement and its operational parameters were not adaptively adjusted on the fly.

To achieve optimum sensing performance, it is desired to design a GPR system that can operate adaptively under varying sensing conditions. For instance, to detect a shallowly buried object of small size, GPR radiating high-frequency EM waves can result in a fine sensing resolution on account of high frequency signals' short wavelengths; while to detect deep buried objects, radiating lower-frequency EM waves have advantages as signals of longer wavelengths have better ground penetrating capabilities. Based on such observations, cognitive GPRs have been proposed and investigated by dynamically tuning of GPR operational system parameters to improve sensing performance [15].

2.2 Deep Learning Based GPR Analysis

Subsurface objects can be recognized through different GPR data classification tasks, for example, determining the material type, structural defect, burial depth, and diameter depending on specific applications.

In this thesis, we use a 2D convolutional neural network (2D CNN) and a 3D convolutional neural network (3D CNN). The use of a convolutional neural network (CNN) is motivated by the fact that CNNs outperform other artificial neural networks on conventional computer vision tasks such as object detection [16], facial expression recognition [17], and medical imaging segmentation [18], through feature learning.

A 2D CNN architecture is developed, which is used to classify subsurface objects for material type, for example, metallic, concrete, and polyvinyl chloride (PVC). A 64×64 grayscale B-Scan image, serves as input to the pre-trained classifier, as discussed in detail in Chapter 5.

A 3D CNN approach is developed for complex subsurface object shape classification, for example, L-shape, T-shape, X-shape, and sphere shape. The model takes a 4-dimensional data as input and predicts the object shape. A detailed discussion of this approach is explained in Chapter 6.

A 3D CNN approach is developed for complex subsurface object shape classification, for example, L-shape, T-shape, X-shape, and sphere shape. The model takes a 4-dimensional data as input, and predicts the object shape. A detailed discussion of this approach is explained in Chapter 6.

2.3 Edge Computing

Edge computing is a computing paradigm that uses one or more collaborative end-user clients or near-user edge IoT devices to carry out a substantial amount of storage, communication, control, configuration, measurement and management. With distributed communication, computation, and storage resources and services on or close to devices and systems in the control of end-users, edge computing may enable real-time autonomous configurations and operations of those devices and systems [19, 20, 21, 22, 23].

Edge computing provides ideal support for implementing and operating cognitive portable GPRs. The functions of GPR signal processing and intelligence generation require significant computation and storage capabilities, which could pose significant challenges to portable GPRs that have limited energy, computing, and storage resources. With edge computing, a promising solution is to offload some or all of the computation and storage tasks to an edge server. A cognitive GPR requires contiguous low-latency communications for real-time transmission of data and control feedback. The proximity of edge servers to end users/devices may satisfy such communication requirements. In contrast, traditional remote cloud computing services have difficulty providing uninterrupted services to the cognitive GPR due to the intermittent network connectivity and long communication latency.

Although edge computing makes it very promising to develop a cognitive portable GPR, there are still several significant research challenges that need to be addressed. Online intelligence generation requires continuous and real-time transmission and analysis of vast volumes of GPR data. A roadway GPR inspection can produce 100 or more gigabytes of data per hour. To reduce the amount of data to be transmitted from the GPR to the edge servers, some local data-processing functions could be performed at the transceiver side. However, the local computation time will increase the overall latency of the feedback loop. Therefore, the trade-off between communication and computation of the perception-action cycle needs to be studied by considering resource constraints and performance requirements. As the GPR moves, service might need to migrate from one edge server to another, posing challenges on continuous operation and control of GPRs.

There has been some research conducted on computation offloading for edge computing [24, 25, 26]. In most existing research on offloading, delay performance is evaluated either by using delay bound values or average delay values based on statistical information of the involved stochastic processes, such as the computation task arrival and the wireless channels, which may make them inapplicable in practical applications.

2.4 Deep Reinforcement Learning

Deep reinforcement learning techniques in autonomous robots and sequential decision making process have been broadly studied [27, 28, 29]. The current studies incorporate the versatile operation control of robots, mechanical control, and the administration in multi-robot frameworks locally, at the edge or the cloud. In [30], the authors used a DQN to develop an end-to-end autonomous robotic system that incorporates path planning. Robots and humans safely coexist through socially compliant interaction with inverse reinforcement learning (RL) [31]. The solution for mission-driven robotics with visual navigation problems has been developed through DRL and AI2-THOR framework in [32].

Recent research such as automatic exploration for navigation in an unknown environment using DRL-based decision algorithm with classical robotic methods [33] is gaining attention. Vehicle classification with a deep reinforcement learning algorithm that selects key areas from an image automatically, through integrating multi-glimpse and visual attention mechanism which highlights one part of an image and weaken the others [34].

By formulating active object detection as a sequential action decision process, the work in [35] implemented a hybrid of deep Q-learning network (DQN) with dueling. Deep reinforcement learning was also studied in a sequential decision making processes for imbalance classification [36] by formulating a sensitivity reward function for minority class data sets. An artificial agent is trained to act in a human-like manner to reduce over-segmentation errors through a joint surgical gesture segmentation and classification method [37].

Among different sensor modalities, GPRs have been widely used in subsurface target detection. However, mapping the underground targets from GPR signals is nontrivial, because different from a laser scanner, a GPR cannot provide 3D positions by any other means but C-Scans. Feng *et al*. [38] designed a GPR-based model reconstruction system for underground utilities using an auto-encoder. Yang *et al.* [39] also designed a GPR-based subsurface object detection and reconstruction using random motion and depthnet. Chen *et al.* [40] proposed an automatic concrete crack-detection method fusing point clouds and images based on improved Otsu's algorithm.

Recent research such as combining visual exploration of 2D ground data and 3D point cloud data for roads environment [41, 42, 43] is gaining attention. This method [41] is designed to inspect road surfaces, manholes covers and gullies. Weilin *et al.* [42] proposed a method that detects tree trunks with irregular contours, using LiDAR data and 2D images from the GPR scanner. The authors in [43] designed a two-step process that involves a grid data creation, where through interpolation, the point cloud data is changed into grid data suitable for filling the empty data and then enhancing the homogeneous points.

Point cloud has become one of the most significant data formats for 3D representation. Its gaining increased popularity as a result of increased availability of acquisition devices, such as LiDAR, as well as increased application in areas such as robotics, autonomous driving, augmented and virtual reality. Jürgen describes the relevance of 3D point clouds for a large number of geospatial applications through machine learning and deep learning [44]. Other sophisticated applications of point clouds include but not limited to connected autonomous vehicles [45], ground points segmentation for challenging terrains for autonomous vehicles [46], and modeling and predicting vehicle accident occurrence in smart cities [47].

CHAPTER 3

SYSTEM ARCHITECTURE

3.1 Cognitive GPR System

A typical operational scenario is that a GPR moves around in a predetermined geographical area (environment) to detect a subsurface object through transmitting EM waves into the ground, and receiving reflected EM waves whenever there's a contrast in material dielectric properties, as shown in Figure 3.1. Adaptive tuning of operational parameters of the AC-GPR, such as frequency, waveform, polarization, and wave timing, may lead to considerable improvement in the quality and efficiency of subsurface sensing. Figure 3.2 shows the conventional mode of GPR operation where an expert sets the operational parameters into an optimal configuration based on previous experience with similar past situations. This is an iterative time-consuming process. The conventional mode is not suitable for continuous long-time operations, especially in a complex environment inaccessible to humans.

The concept of cognitive GPR was proposed in [15] where intelligence was expected to be generated on the fly to adaptively adjust the operational parameters based on data analysis and feedback control. As shown in Figure 3.3, the cognitive GPR consists of an adaptive GPR transceiver, a perceptor module, a memory module, and a cognitive analyzer. The operation of the cognitive GPR follows a perception-action cycle: first, the GPR transceiver collects the reflected wave data about subsurface objects and sends them to the preceptor. Then, the preceptor processes and analyzes the data to extract signature patterns and format a perception of subsurface conditions. The memory module has a GIS database containing urban subsurface condition attributes and spatial locations. The cognitive analyzer carries out machine learning based on both the processing results from the perceptor and the prior knowledge about GPR measurement from the memory

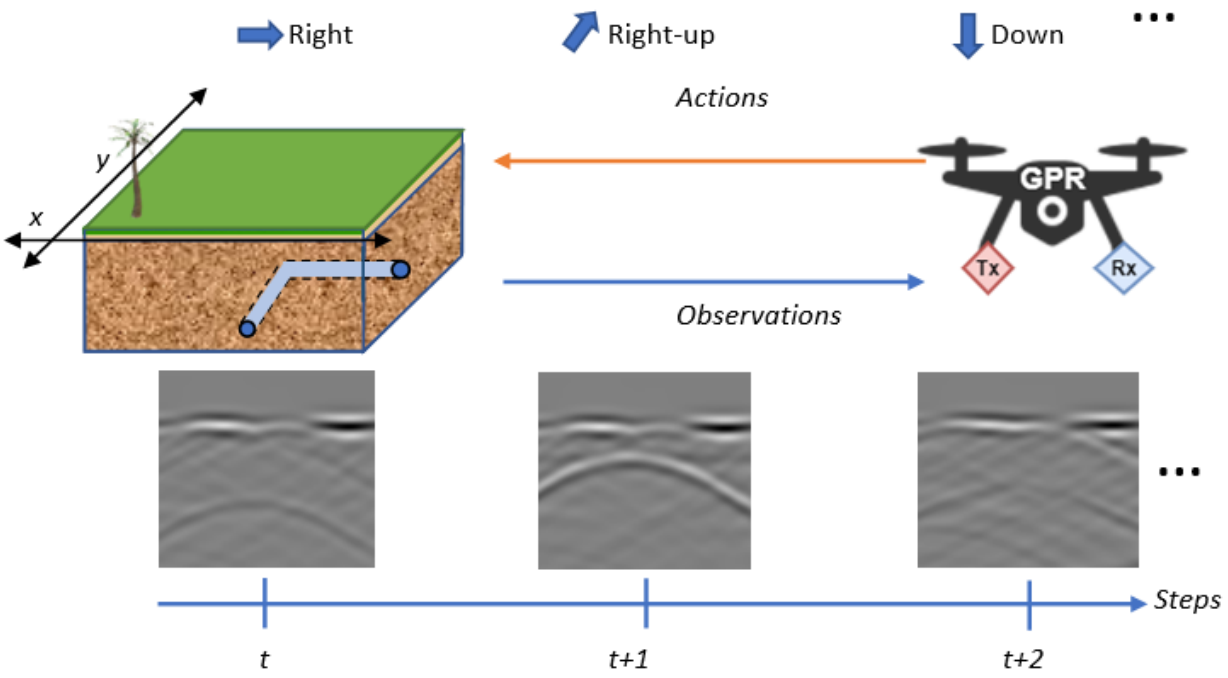


Figure 3.1 Successive action choice procedure of dynamic subsurface object detection

module to produce intelligent responses, locally or at the edge [48] for the control of radar transceiver reconfigurations.

Although the concept of cognitive GPR in [15] pointed to promising direction in the system architecture development, no unambiguous definition or approach for a cognitive GPR was provided to build an adaptive and smart GPR that generates intelligence to adaptively adjust its operational parameters in an uncertain and dynamic sensing environment.

Adaptive tuning of operational parameters of the AC-GPR, such as frequency, waveform, polarization and wave timing, may lead to considerable improvement in the quality and efficiency of subsurface sensing. Fig. 3.2 shows the conventional mode of GPR operation where an expert sets the operational parameters into an optimal configuration based on previous experience with similar past situations. This is an iterative time-consuming process. The conventional mode is not suitable for continuous long-time operations, especially in a complex environment inaccessible to humans.

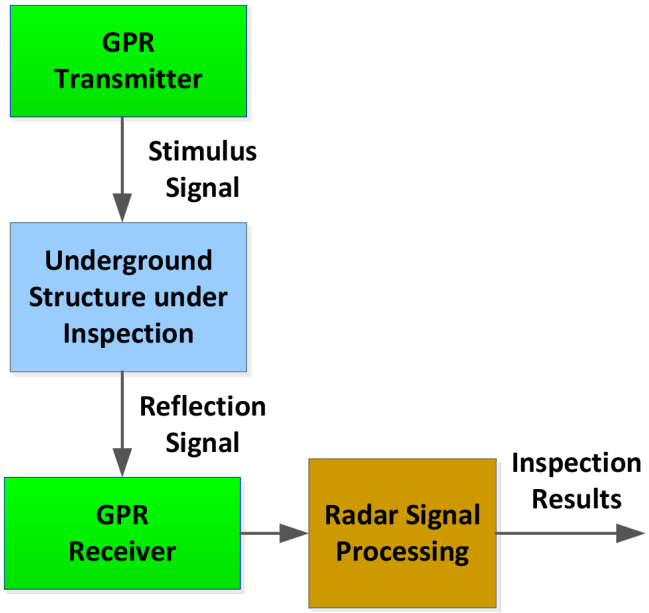


Figure 3.2 The operational flows of a traditional GPR

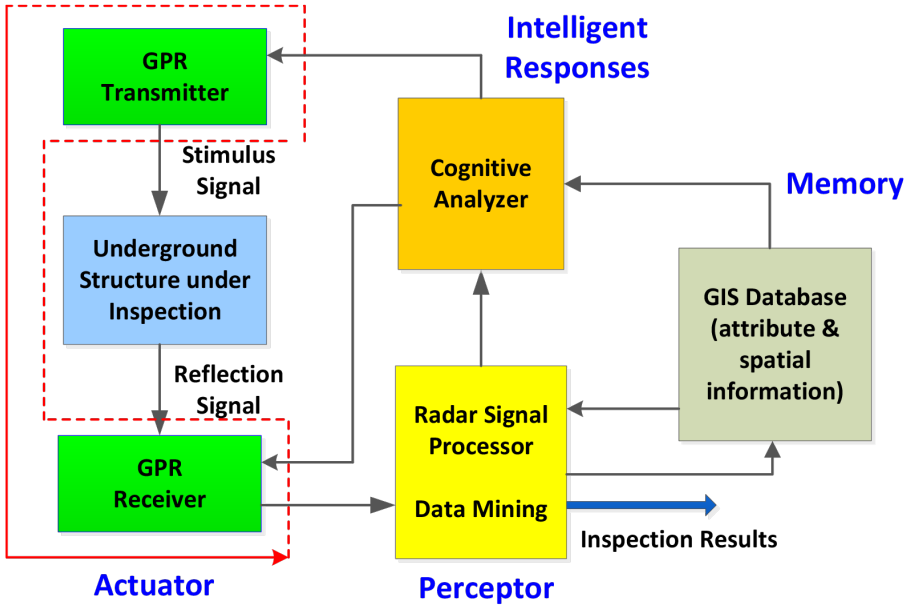


Figure 3.3 A cognitive GPR

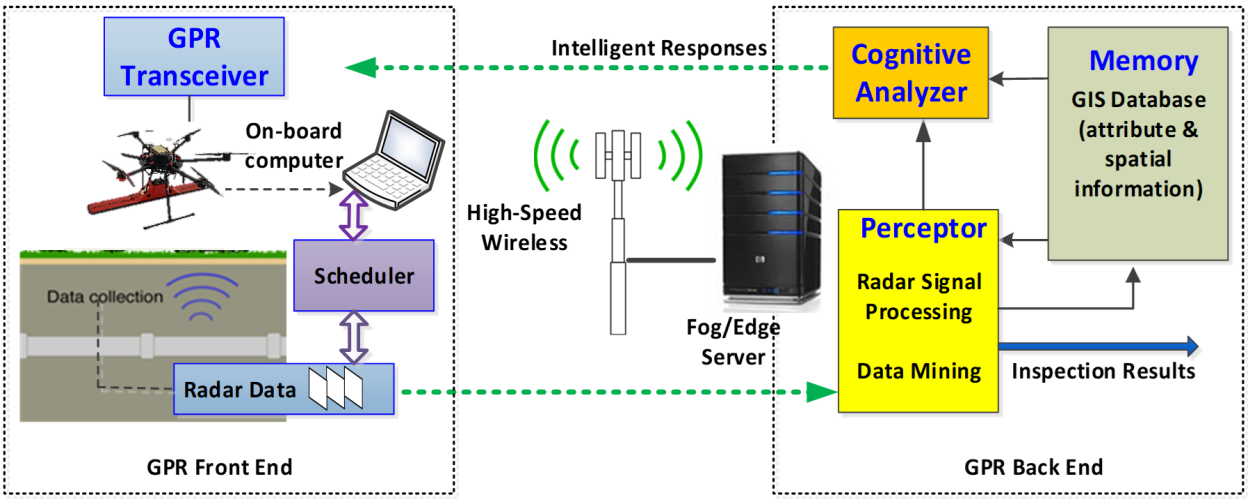


Figure 3.4 The architecture of the proposed cognitive GPR

3.2 The Proposed System Architecture

A conventional mode of GPR operation is that an expert sets the operational parameters into a proper configuration based on previous experience, which is an iterative time-consuming process, as shown in Figure 3.2. An alternative is to use a cognitive GPR where intelligence is generated on the fly to adaptively adjust the operational parameters based on data analysis and feedback control [15].

As shown in Figure 3.3, a cognitive GPR consists of an adaptive GPR transceiver, a perceptor module, a memory module, and a cognitive analyzer. The operation of the cognitive GPR follows a perception-action cycle: first, the GPR transceiver collects the reflected wave data about subsurface objects and sends them to the preceptor. Then, the preceptor performs signal processing and data mining on the data to extract signature patterns and format the perception of subsurface conditions. The memory module has a GIS database containing urban subsurface condition attributes and spatial locations. The cognitive analyzer carries out machine learning based on both the processing results from the perceptor and the prior knowledge about GPR measurement from the memory module to produce intelligent response for the control of radar transceiver reconfigurations. Once receiving the intelligent feedback from the cognitive analyzer, the adaptive GPR transceiver changes its operational parameters based on the feedback. During

this process, collected GPR data can also be integrated with other data acquired by IoT devices such as positioning sensors under network coordination.

In this section, we present a system architecture for our proposed EC-enabled cognitive portable GPR. As shown in Figure 3.4, the GPR mainly includes two parts: the front end and the back end. The front end is portable and includes a GPR transceiver module for launching and receiving electromagnetic waves, a microcomputer for local computation, and a pair of wireless transmitter and receiver for communicating with the edge server. Within the microcomputer runs a task scheduler which makes a decision on whether a computation task should be offloaded to the edge server. If it is decided that the task should not be offloaded, the microcomputer will execute the task. The back end includes the perceptor module, the memory module, and the cognitive analyzer of the GPR. In view of the requirement on extensive computation and storage associated with those back-end modules, we propose that those back-end modules reside at the edge server.

In the context of edge computing, the perception-action cycle of the cognitive GPR can be described as follows. The GPR transceiver collects the reflected wave data about subsurface objects. Based on the types of computation tasks, the delay performance requirement, and the resource constraints, the scheduler makes decisions about which computation tasks should be performed locally and which computation tasks should be offloaded to the edge server. Following the decisions, the microcomputer either offloads a task or executes the task locally and then wirelessly sends the results and associated data to the edge server where the preceptor and memory modules and the cognitive analyzer work together and generate control commands for the radar transceiver reconfigurations. The control command will be wirelessly sent back to the GPR front-end transceiver. The operational parameters of the transceiver changes based on the control feedback.

3.3 Dynamic Computation Task Scheduling

3.3.1 The Communication Models of the GPR Front End

As discussed previously, GPR data pre-processing, analysis, and intelligence generation involves different computation tasks. A computation task could either be executed by the local microcomputer equipped at the GPR front end or be offloaded to the edge server. Let D_i^{in} and L_i^{in}

denotes the input data and data size of computation task i, respectively. If computation task i is executed by the local microcomputer, output data D_i^{out} with size L_i^{out} will be produced. Let C_i be the CPU processing cycles of computation task i. Assume that the CPU at the local microcomputer is operating at frequency f^l with power consumption P^l. Then, the computation time needed to execute computation task i at the GPR front end is $T_i^{l\_cp} = C_i/f^l$; and the local energy consumption on computation is $E_i^{l\_cp} = T_i^{l\_cp} \cdot P^l$.

Assume that if the computation task i is executed locally at the GPR front end, output data D_i^{out} needs to be transmitted to the edge server. Let P^{tx} be the transmit power of the wireless transmitter at the GPR front end to communicate with the edge server and B the system bandwidth. The achievable throughput for transmitting D_i^{out} is $r_i = B\log_2\left(1+\frac{\gamma_i \cdot P^{tx}}{N_0 \cdot B}\right)$ where γ_i is the channel power gain which is assumed to be constant during transmitting the data of computation task i; and N_0 is the noise power spectral density at the receiver of the edge server. The communication delay and energy consumption of transmitting D_i^{out} can be calculated as $T_i^{l\_cm} = L_i^{\text{out}}/r_i$, and $E_i^{l\_cm} = P^{tx} \cdot T_i^{\text{out}}$, respectively. Since the computation delay on input data D_i^{in} and the transmission delay on output data D_i^{out} of task i could overlap in time, the overall delay of processing task i at the GPR front end, $t_i^{l\_pr}$, satisfies

$$\max\{T_i^{l\_cp}, T_i^{l\_cm}\} \leq t_i^{l\_pr} \leq T_i^{l\_cp} + T_i^{l\_cm} \tag{3.1}$$

where $\max\{T_i^{l\_cp}, T_i^{l\_cm}\}$ corresponds to the case where the maximum time overlap between computation and transmission takes place; $T_i^{l\_cp} + T_i^{l\_cm}$ is the case where the transmission of D_i^{out} starts right after the computation on input data D_i^{in} ends without time overlap.

The total energy consumption of processing computation task i locally, defined as the sum of energy consumption on both computation and communication, is

$$E_i^l = E_i^{l\_cp} + E_i^{l\_cm}. \tag{3.2}$$

3.3.2 The Communication Models of the GPR Back End

Assume that if the computation task i is offloaded to the edge server, input data D_i^{in} needs to be transmitted to the edge server for task execution. With the throughput r_i, the delay of transmitting input data D_i^{in} to the edge server is $T_i^{f\_cm} = L_i^{\text{in}}/r_i$. The corresponding energy consumption on data transmission is $E_i^{f\_cm} = P^{tx} \cdot T_i^{f\_cm}$. We assume that the edge server has powerful computation capability through parallel computing. Thus, the delay of executing a computation task at the edge server is negligible.

Algorithm 1: The proposed computation task offloading policy.

Calculate $t_i^{l\_pr}$, $T_i^{f\_cm}$, and $t_{1:i-1}^{q}$;
if $T_i^{f\_cm} + t_{1:i-1}^{q} \leq T_i^{\max}$ **then**
  if $E_i^{f\_cm} \leq E_i^{l}$ **then**
    Offload computation task i to the edge server by transmitting its input data D_i^{in};
  else
    Execute computation task i at the GPR front end, and transmit the resulting output data D_i^{out};
  end
else if $t_i^{l\_pr} + t_{1:i-1}^{q} \leq T_i^{\max}$ **then**
  Execute computation task i at the GPR front end, and transmit the resulting output data D_i^{out};
else
  Drop computation task i.
end

3.3.3 The Proposed Offloading Policy

Next, an offloading policy will be presented for the scheduler at the GPR front end to dynamically determine whether a computation task should be executed locally or offloaded to the edge server

The scheduler maintains a computation task buffer. We assume that different tasks in the buffer are scheduled on a first-come, first-served basis. Let $\mathbf{B} = \{1, 2, \cdots, i-1, i\}$ denotes the task buffer having i tasks at the present time with the ith task being the latest one to be processed. It is also assumed that both the local computation resources and channel side information for the already scheduled tasks are available to the scheduler so that based on these information the

scheduler could estimate a timeline of the completion of the scheduled tasks. Let $t^q_{1:i-1}$ be the overall time needed to complete the processing of the $i-1$ tasks existing in the buffer when the ith task enters the buffer. $t^q_{1:i-1}$ can be considered as the queuing delay of task i before it is processed. Note that among these $i-1$ tasks, some tasks may be locally executed at the GPR front end, and others may be offloaded to the edge server. Let $d^l_i, d^f_i \in \{0,1\}$ denotes the offloading decision indicator for computation task i, i.e., if the task is decided to be executed by the local microcomputer at the GPR front end, $d^l_i = 1$ and $d^f_i = 0$; otherwise $d^l_i = 0$ and $d^f_i = 1$. Thus, the overall time needed to complete all of the i tasks present in the buffer can be estimated as

$$t^q_{1:i} = t^q_{1:i-1} + d^l_i \cdot t^{l\_pr}_i + d^f_i \cdot T^{f\_cm}_i. \tag{3.3}$$

The proposed computation task offloading policy for scheduling task i is shown in Algorithm 1. The offloading policy takes into account the energy limitation of the mobile GPR front end. Assume that each computation task i is associated with a deadline $T^{\max}_i$. At the beginning, the scheduler calculates the estimated delay $T^{f\_cm}_i$ of transmitting data D^{in}_i, the estimated overall delay $t^{l\_pr}_i$, and the overall time $t^q_{1:i-1}$ needed to complete the $i-1$ task present in the buffer. If computation task i can be executed at the edge server before its deadline $T^{\max}_i$ (line 2), and at the same time, the energy consumption $E^{f\_cm}_i$ on transmitting input data D^{in}_i is less than the energy consumption E^l_i on local processing of task i (line 3), the computation task will be offloaded to the edge server (line 4). If computation task i can be executed at the edge server before its deadline $T^{\max}_i$ (line 2), and at the same time, the energy consumption $E^{f\_cm}_i$ on transmitting input data D^{in}_i is larger than the energy consumption E^l_i on local processing of task i (line 5), computation task i will be locally executed at the GPR front end; after the execution, the resulting output data D^{out}_i will be sent to the edge server (line 6). This way, the energy consumption at the GPR front end can be reduced as much as possible. If computation task i can not be executed at the edge server before its deadline $T^{\max}_i$ but it can be processed locally before the deadline (line 8), computation task i will be locally executed at the GPR front end; after the execution, the resulting output data D^{out}_i will be sent to the edge server. If computation task i can not be completed either at the GPR front end or at the edge server (line 10), it will be dropped from the buffer (line 11).

3.4 Service Migration

One key issue that comes with proximity is how to ensure that GPR scanners always receive good connectivity and service continuity as it moves across the field of interest. This can be achieved through GPR container migration from one wireless edge server to another using IP tables and Rsync via Transmission Control Protocol (TCP) sockets. Rsync is a utility for efficiently transferring and synchronizing files across computer systems, by checking the timestamp and size of files.

GPR scanner application may face challenges of intermittent network connection caused by signal disturbance, spurious disconnection during handoff, bandwidth limitation, voluntary disconnection, and unpredictable transmission errors. Since it is through the network connection that edge services are reached, software-defined networking enabled edge servers and service migration can increase the robustness of the network and quality of service (QoS).

Other latency sensitive applications include but not limited to active augmented reality, safety warning, mobile multimedia, and mobile online gaming [49]. These applications are catalyzed by emerging technologies such as, Internet of Things (IoT), cyber security, smart cities/grid/health, 5G LTE, deep learning, quantum computing, just to mention a few.

The proposed system model for service migration consists of a GPR scanner and four computing access points edge servers (CAPES) and controller as shown in 3.5. The test bed is made up of three open-source softwares, SDN Mininet-Wifi, gprMax simulator, and Docker. The simulation was implemented on Ubuntu 16.04 LTS with an Nvidia GeForce GTX 1080 Ti Graphics Card.

Application Plane: The top layer of SDN architecture resides the application plane which consists of application softwares such as offloading manager, GPR load balance manager, docker containers and GPR cognitive analyzer. As shown in Figure 3.5 the northbound application programming interfaces (APIs) (e.g., REST API, FML, Frentic) enable network functions such as path computation, loop avoidance, routing and security [50]. Furthermore, applications can construct an abstracted outlook of the network by gathering data from the controller for decision-making [51].

Control Plane: The control plane is an intelligent layer that transfers guidelines from the application layer to the data layer utilizing the northbound and southbound APIs, respectively. The controller manages the entire network in terms of security/firewall, traffic routing/re-routing, and bandwidth allocation. Southbound application programming interfaces (APIs) e.g. OpenFlow and NetConf are a set of routines, protocols, and tools fused to communicate between the SDN controller and the switches/routers to make adjustments to the network, so as to better adapt to changing geographical environments [52, 53].

Data Plane: The data plane layer consists of devices such as GPR scanner, switches, routers, and wireless access points. Through the southbound APIs, the networking devices forward the incoming and outgoing packets and frames [54, 55] as configured by the controller.

The proposed service migration logic described in Algorithm 2, shows a service migration technique that is based on the connectivity of the GPR scanner to the edge server. Assuming that R^{gpr} and R_j^{max} (line 4) is the wireless range of the GPR scanner and wireless range of the CAPES respectively. If the GPR scanner is not in the range of the jth CAPES, it migrates GPR container to the next hop (CAPES) closest to the GPR scanner (line 5), else carry on with the computation the current CAPES (line 10).

3.5 Performance evaluation

For design evaluation, an underground pipeline inspection experiment was carried out on a street close to an institutional campus. A picture of the test site view is shown in Figure 3.6. In the experiment, the GPR front end is a dual-band system with GSSI SIR-30 control unit which contains all radar electronics to control GPR signal generation and receiving signal acquisition and transmission. It has two sets of ground coupled ultra-wide bandwidth antennas for radiating EM waves and receiving subsurface reflection signals. The GPR front end hardware can be configured to be operable in two frequency bands, 400 MHz and 1.6 GHz. By making the frequency band selectable, it facilitates to achieve optimum sensing resolution and sensing depth. The GPR back end resides at an edge server which is located in a building in the vicinity of the test site and is connected to the GPR front end through a street WiFi network.

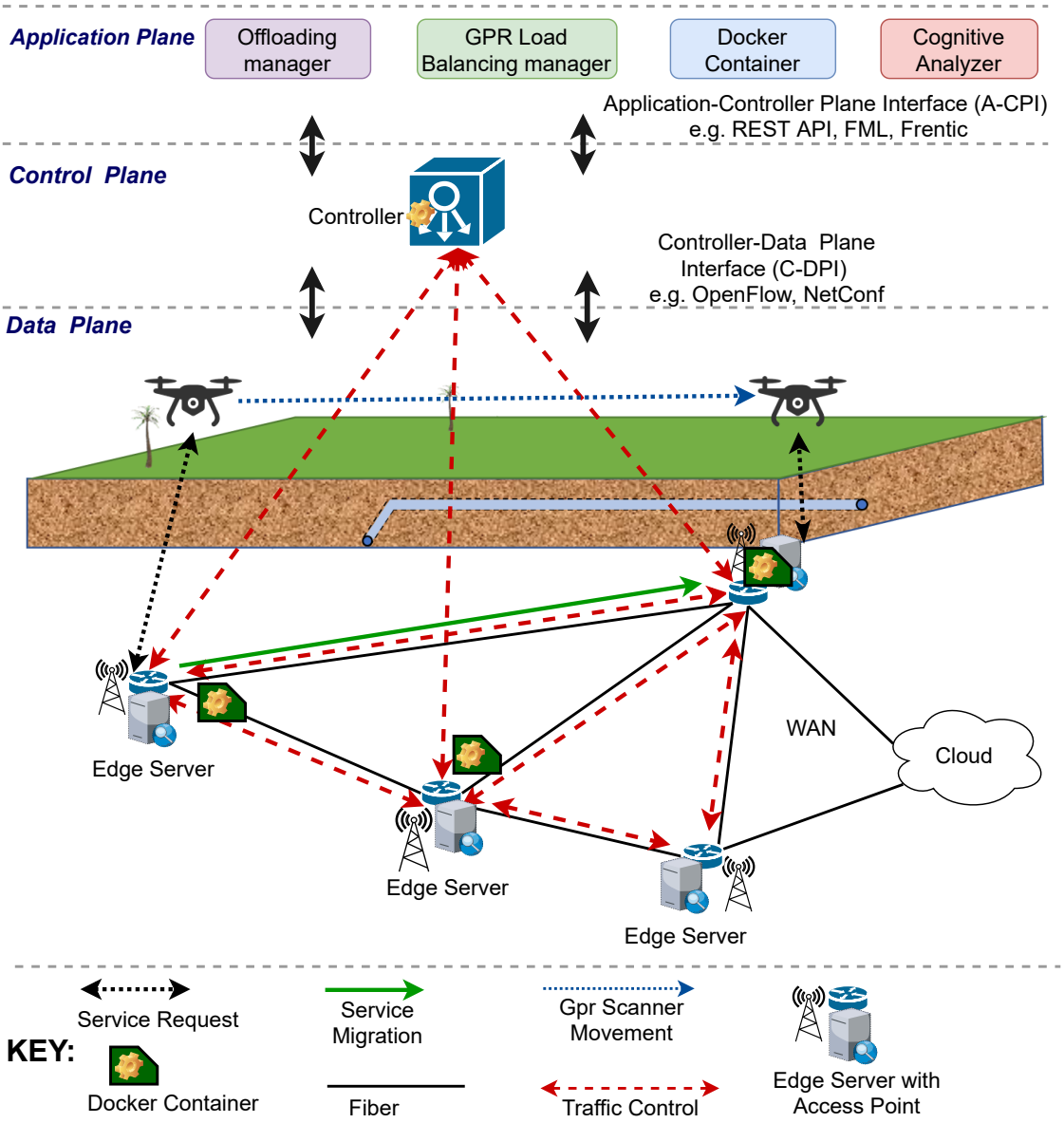


Figure 3.5 Service migration architecture

Algorithm 2: Service migration.

Calculate $t_i^{l\_pr}$, $T_i^{f\_cm}$, and $t_{1:i-1}^{q}$;
if $T_i^{f\_cm} + t_{1:i-1}^{q} \leq T_i^{\max}$ **then**
  if $E_i^{f\_cm} \leq E_i^{l}$ **then**
    if $R^{gpr} \neq R_j^{max}$ **then**
      Migrate to the next hop (edge server)
    else
      Offload computation task i to the edge server by transmitting its input data D_i^{in};
    end
  else
    Execute computation task i at the GPR front end, and transmit the resulting output data D_i^{out};
  end
else if $t_i^{l\_pr} + t_{1:i-1}^{q} \leq T_i^{\max}$ **then**
  Execute computation task i at the GPR front end, and transmit the resulting output data D_i^{out};
else
  Drop computation task i.
end

In the operation, the GPR front end radiates short pulses toward the ground. The reflection signal at each location produces an A-scan waveform whose amplitude and phase parameters record the features of subsurface objects that the pulse encounters during its propagation. By moving GPR antennas to scan the survey area, numerous A-scan waveforms are collected. By assembling all A-scan waves together, B-scan images are obtained to produce more comprehensive views of the subsurface objects. One of the B-Scan images obtained onsite is also shown in Figure 3.6.

In the context of edge computing, signal processing and intelligence generation tasks are partitioned and processed based on the task scheduling policy. In the experiment, multiple GPR scans are run over different spaces. Due to different scanning lengths, different scans generate different amounts of GPR raw data (binary file format with extension GZT). For each scan, the GPR raw data are sent from the GPR front end to the edge server. Figure 3.7 shows the raw data size of each GPR scan and the GPR raw data transmission time. The pre-processing time for

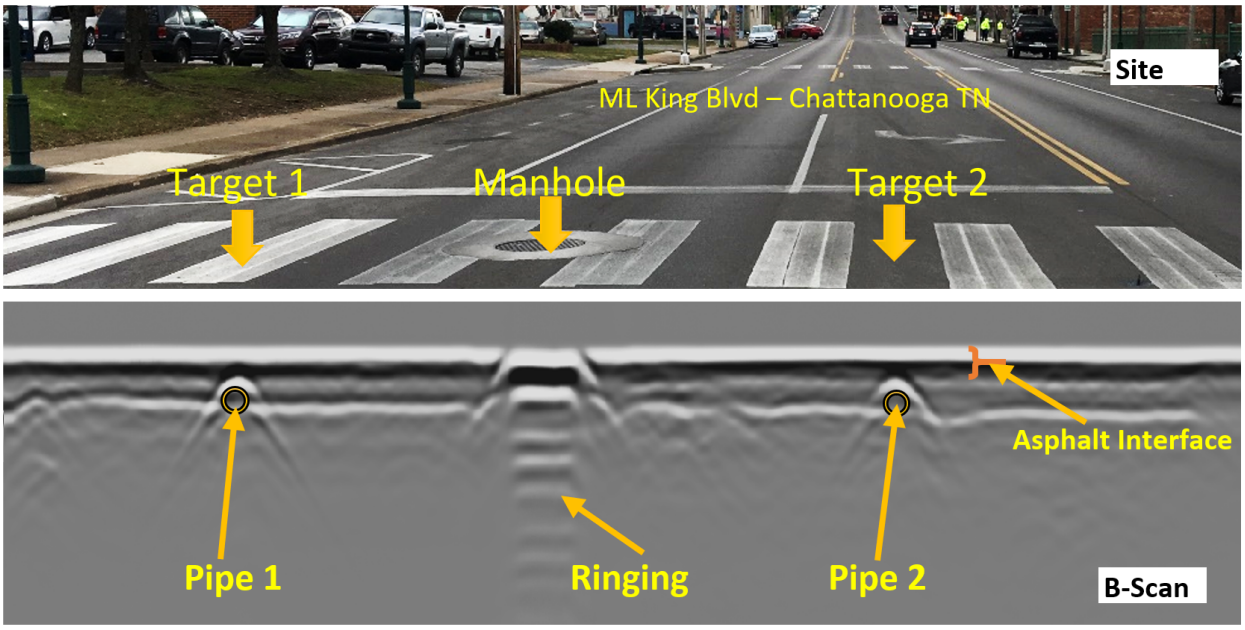


Figure 3.6 GPR data collection on MLK

preprocessing the GPR raw data and the B-Scan rendering time for assembling the corresponding A-Scans into a B-Scan image at the edge server is also shown in the figure. It can be observed that the transmission time is not proportionate to the data size due to the time-varying data rate of the WiFi network. In addition, for some scans, the transmission time is much longer than the B-Scan rendering time and the pre-processing time.

The simulation performance of the proposed service migration framework demonstrates a stable transmission delay, averaging at about 11 seconds across various Docker file sizes as shown in Figure 3.8.

3.6 Summary

An edge computing framework for the development of cognitive portable GPRs was developed. The system architecture of the proposed EC-enable cognitive portable GPR was designed with different computation tasks where a typical perception-action cycle of cognitive GPRs was identified and explained.

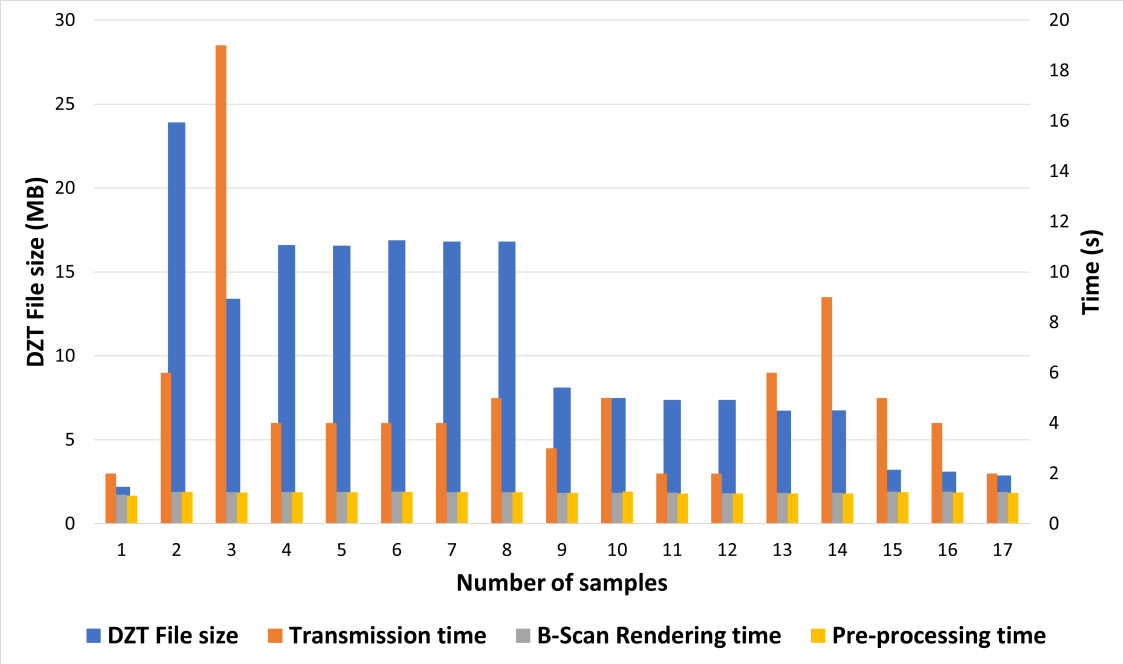


Figure 3.7 Edge computing and GPR data pre-processing performance

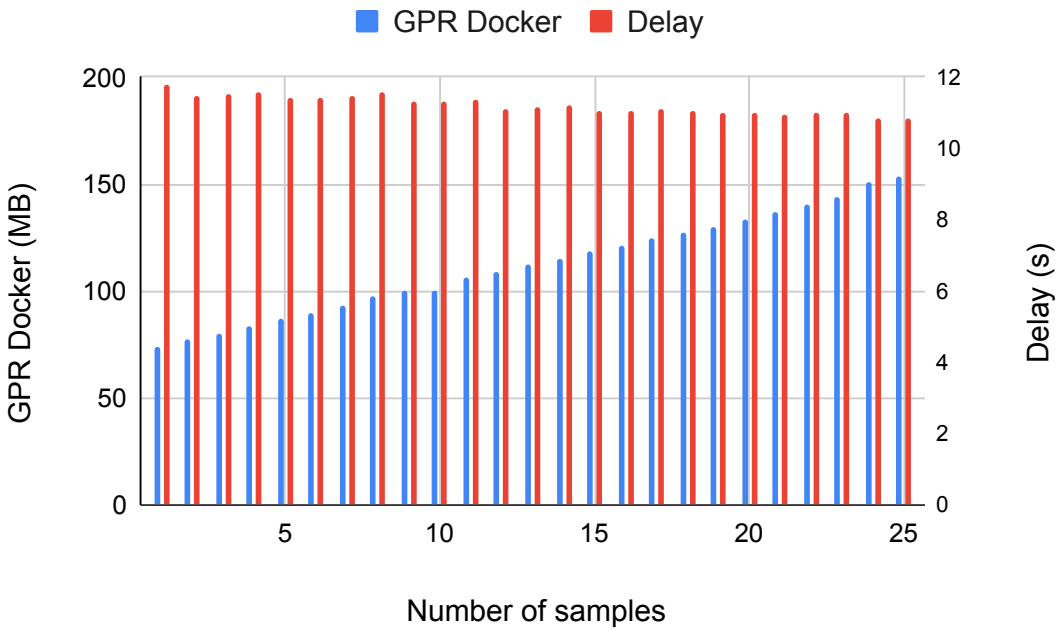


Figure 3.8 Service migration network performance

A computation task offloading policy was designed to determine whether a computation task should be executed at the local GPR computer or at the remote edge server. Furthermore, a service migration network based on software-defined networking (SDN) was developed. Service migration is very important for GPR sensing operations, where inferences and intelligence generated from the edge server are guaranteed for portable GPRs during sensing.

The proposed method was tested along M.L.K. Blvd Chattanooga Tennessee, where several GPR data were collected and transmitted to the edge server for pre-processing, and inference results were displayed onto the engineer's tablet. Experiments were conducted to demonstrate the efficacy of the proposed system. From the results the transmission time from GPR to the edge server was very unpredictable, in some instances, a small file will take longer to be transmitted and vice versa, however, in most cases transmission time averages at about 3s for up to 23 MB of GPR data.

3.7 Future Work

Future implementation of this project would be to further streamline offloading policy to address time sensitive service migration requests. This can be accomplished by implementing a dynamic TCP buffer memory, and a dynamic socket buffer size. However, since over buffering can cause some applications to behave poorly (typically causing sluggish interactive response) and risk running the system out of memory, large default socket buffers have to be considered carefully on multi-user systems. Furthermore, a Kubernetes ecosystem will be developed which is meant to coordinate clusters of edge servers at scale in an efficient manner.

CHAPTER 4

GPR DATA ACQUISITION AND AUGMENTATION

4.1 GPR Data Acquisition Through Modeling Simulation

4.1.1 Introduction

Machine learning model performance is heavily reliant upon the availability of training data. Furthermore, in situ applications of machine learning require data to be indicative of the real world environment the applications plan to infer in. For buried object detection, this would be a B-Scan image that bears a close resemblance to B-Scans collected in the field. However, these data are not widely available, and when available, the number of samples is few. In situations like this, data augmentation can produce many samples to enhance the classification of buried objects [29].

In the world of Ground Penetrating Radar, an open source software named *gprMax* is used to simulate the presence of underground objects [56]. This software is based on FDTD [57], a numerical method to solve Maxwell's equations that govern wave propagation within a specific medium. The problem with this type of simulated data is that it bears little resemblance to a B-Scan that would be obtained in the real world. Furthermore, due to the complexity of FDTD, time to complete a single simulation can take several hours. Thus making the synthesis of a large set of images for training data, virtually impossible. To solve this problem, we propose a generative model architecture to synthesize realistic B-Scans in real time. However, before that, in the next subsection, we will discuss the gprMax configurations.

4.1.2 GprMax Configuration

To simulate close to real-world GPR data, one has to take into consideration the following material properties i.e. relative permittivity, conductivity, relative permeability, and magnetic loss. Table 4.1 shows a sample of materials and their respective dielectric constant values [58]. For

example, air has a dielectric constant of 1, ice is 3, saturate sand has a dielectric constant of 15, fresh water is 80 and metal is infinity.

The study of dielectric constant is an important principle because GPR works by sending a signal from a receiver and into the surface. The signal is bounced off whenever there's a change in contrast of the material it encounters within the surface, hence creating a reading. Below is a gprMax code sample on Figure 4.1 that models an environment made of concrete medium, and within it is a metallic cylinder pipe configured as `#cylinder: 0.77 0.1 0 0.77 0.9 0.0025 0.1 pec y` with its axis in the y direction, a length of 0.8 m, and a radius of 100 mm.

It is desired to see the reflection from the cylinder, therefore the time window must be long enough to allow the electromagnetic waves to propagate from the source through the medium to the cylinder and be reflected back to the receiver. To allow for the entire source waveform to be reflected back to the receiver an initial time window of 12 ns is applied.

The Ricker waveform is created with the `#waveform` command, specifying an amplitude of one, center frequency of 400 MHz and picking an arbitrary identifier of `MyLineSource`. The Hertzian dipole source is created using the `#hertzian_dipole` command, specifying a `z` direction polarisation (perpendicular to the survey direction if a B-scan were being created), location on the surface of the slab, and using the Ricker waveform already created.

Real world mediums are mostly heterogeneous in nature. With gprMax heterogeneous soil can be modeled using a stochastic distribution of dielectric properties. A mixing model for soils proposed by Peplinski [59] is used to define a series of dispersive material properties for the soil.

Line 5 on Figure 4.2 defines a series of dispersive materials to represent a soil with sand fraction 0.1, clay fraction 0.1, bulk density $2g/cm^3$, sand particle density of $2.66g/cm^3$, and a volumetric water fraction range of 0.001 - 0.25. The volumetric water fraction is given as a range which is what defines a series of dispersive materials.

To make the gprMax models more distinctive surface roughness can be added. For example, to add surface water we use the command `#add_surface_water`, or grass `#add_grass`. Experimental testing and results of these simple and complex soils are discussed in Chapter 5 and Chapter 6.

```
#material: 6.0 0.1 1.0 0.0 concrete

#domain: 2.5 0.2 0.00375
#dx_dy_dz: 0.00375 0.00375 0.00375
#time_window: 12e-9

#box: 0.0 0.0 0 2.5 0.15 0.0025 concrete
#cylinder: 0.77 0.1 0 0.77 0.9 0.0025 0.1 pec y

#rx: 0.1125 0.1525 0-
#src_steps: 0.002 0.0 0
#rx_steps: 0.002 0.0 0

#waveform: gaussiandotnorm 1.0 400e6 MyLineSource
#hertzian_dipole: z 0.0875 0.1525 0 MyLineSource
```

Figure 4.1 Homogeneous soil model

```
#domain: 0.5 0.5 0.002
#dx_dy_dz: 0.002 0.002 0.002
#time_window: 5e-9

#soil_peplinski: 0.1 0.1 2.0 2.66 0.001 0.25 my_soil
#fractal_box: 0 0 0 0.15 0.15 0.070 1.5 1 1 1 50 my_soil my_soil_box
#add_grass: 0 0 0.1 0.1 0.1 0.1 1.5 0.2 0.25 100 my_soil_box

#cylinder: 0.15 0.08 0 0.15 0.08 0.002 0.08 free_space y

#rx: 0.08 0.17 0
#src_steps: 0.002 0.0 0
#rx_steps: 0.002 0.0 0

#waveform: ricker 1 1.5e9 my_ricker
#hertzian_dipole: z 0.04 0.17 0 my_ricker
#messages: n
```

Figure 4.2 Heterogeneous soil model

Table 4.1 Dielectric constant [58]

Material	**K**
Air	1
Gasoline	2
Ice	3
Dry Sand	5
Granite	6
Dry Salt	6
Limestone	8
Shale	15
Saturated Sand	25
Silts	30
Clays	40
Distilled Water	80
Fresh Water	80
Sea Water	80
Metal	∞

4.2 Preprocessing of Real-World GPR Sensory Data

GPR data preprocessing steps are a critical procedure for extracting accurate interpretation of underground objects. The following preprocessing steps are implemented. A B-Scan image first goes through the following preprocessing steps: signal denoising by removing the DC component (arithmetic mean) from each trace of the GPR image, time-zero correction which adjusts all traces to a common time-zero position where the first break of air-wave is observed, and signal enhancement/amplification [3] to enhance the hyperbolic signatures of buried targets.

As shown in Figure 4.3, after the B-Scan image has undergone the first three steps, the B-Scan image is clean enough for other feature extraction procedures such as region of interest identification, as shown in step four with the red box which indicates where the object target is located.

4.3 GPR Data Augmentation Based on Generative Adversarial Network (GAN)

Generative adversarial networks (GANs) are used widely in image generation, video generation, and voice generation. GANs are algorithmic architectures that use two neural

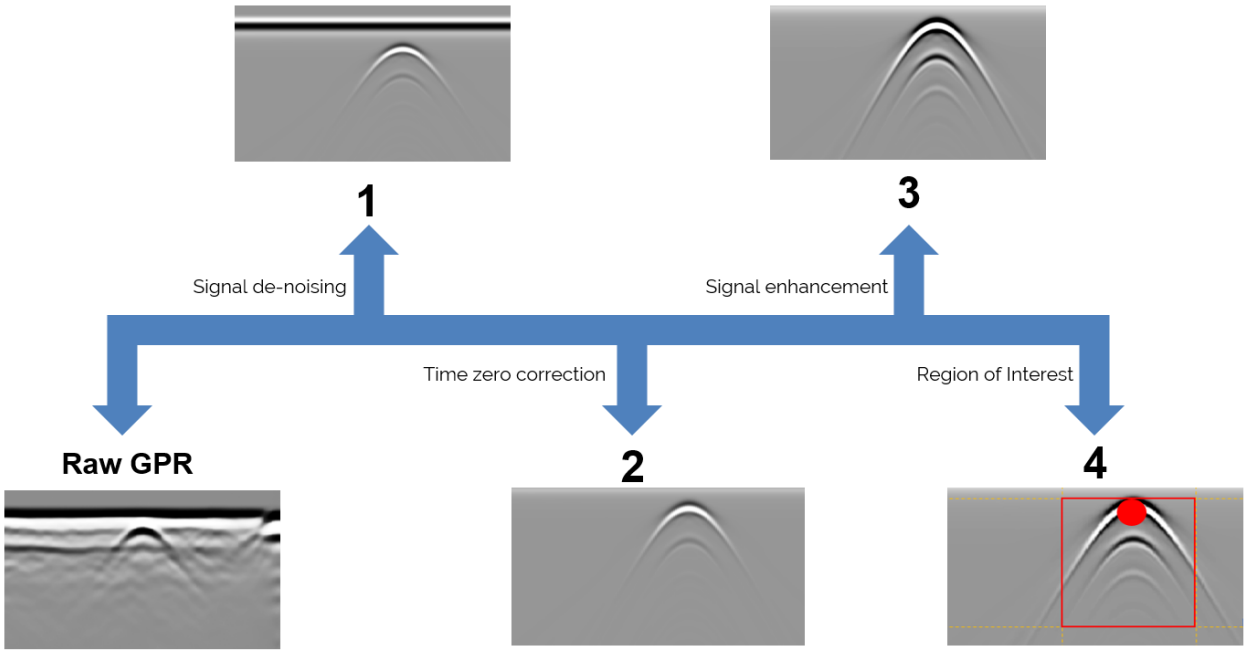


Figure 4.3 B-Scan pre-processing steps

networks, pitting one against the other (thus the “adversarial”) in order to generate new, synthetic instances of data that can pass for real data.

In this thesis, a GAN architecture [12] was implemented to generate close to real world synthetic B-Scan data suitable for machine learning purposes, as shown in Figure 4.4. The results of the proposed GPR data augmentation method are displayed in Figure 4.5.

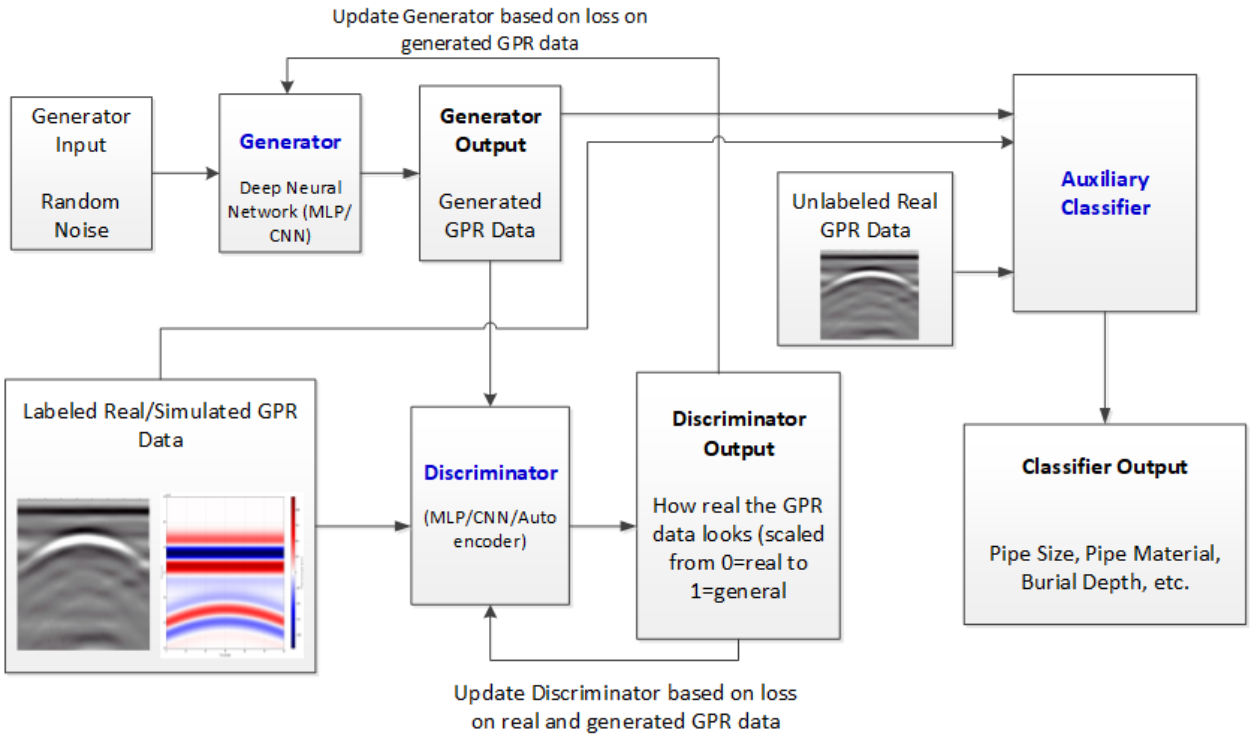


Figure 4.4 Proposed GAN Architecture

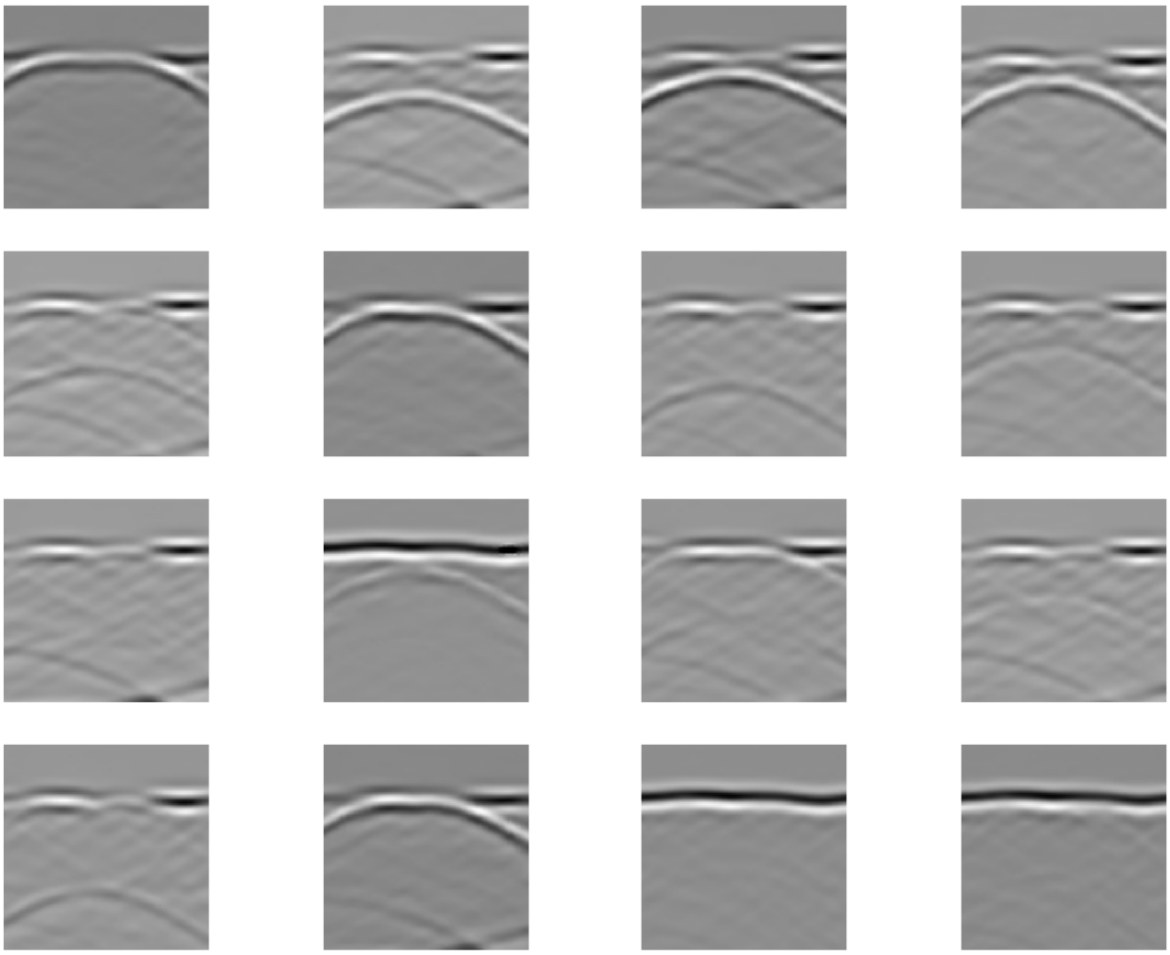


Figure 4.5 GAN generated B-Scan images

CHAPTER 5

COGNITIVE GPR BASED ON B-SCAN DATA

This chapter is focused on the development of a deep reinforcement learning (DRL) framework that enables autonomous cognitive GPR (AC-GPR). To this end, the 2D B-Scan images (observations) and a proper reward function are needed to effectively reflect the value of different actions of the AC-GPR agent at different states. Also, a DRL algorithm with the reward function needs to be developed to learn a policy that directs the AC-GPR's actions. To the best of our knowledge, this is the first work based on DRL for the development of AC-GPR. The main contribution of this method can be summarized as follows.

1. By formulating cognitive subsurface object detection as a Markov decision problem, a deep reinforcement learning framework is established to resolve the problem.

2. A deep Q-network (DQN) algorithm with a novel reward function that combines rewards from both Region of Interest (RoI) identification and object classification is proposed.

3. To show the efficacy of the proposed framework, simulation based validations are performed on real-time GPR data from gprMax simulator by combining DRL with GPR operation modeling [60, 56].

To achieve optimal sensing performance, it is desired to design an autonomous GPR system that can operate adaptively under varying sensing conditions. Specifically, the system is able to adaptively move with a robotic platform and adjust its operational parameters through real-time interaction with the sensing environment. Although the decision-making process of operating an autonomous GPR can be modeled as a finite-horizon Markov decision process (MDP) with finite state and action spaces, the curse of extremely high dimensionality of state space makes

it computationally infeasible to derive optimal action using the standard infinite-horizon DP algorithm [61]. Deep reinforcement learning (DRL) is suitable for this problem since it can reduce the dimensionality of the large state space while learning the optimal policy at the same time.

5.1 The Proposed System Model and Architecture

5.1.1 The Original Concept of Cognitive GPR

A typical operational scenario is that a GPR moves around in a predetermined geographical area (environment) to detect a subsurface object through transmitting EM waves into the ground, and receiving reflected EM waves whenever there's a contrast in material dielectric properties, as shown in Figure 3.1. Adaptive tuning of operational parameters of the AC-GPR, such as frequency, waveform, polarization and wave timing, may lead to considerable improvement in the quality and efficiency of subsurface sensing. Figure 3.2 shows the conventional mode of GPR operation where an expert sets the operational parameters into an optimal configuration based on previous experience with similar past situations. This is an iterative time-consuming process. The conventional mode is not suitable for continuous long-time operations, especially in a complex environment inaccessible to humans.

The concept of cognitive GPR was proposed in [15] where intelligence was expected to be generated on the fly to adaptively adjust the operational parameters based on data analysis and feedback control. As shown in Figure 3.3, the cognitive GPR consists of an adaptive GPR transceiver, a perceptor module, a memory module, and a cognitive analyzer. The operation of the cognitive GPR follows a perception-action cycle: first, the GPR transceiver collects the reflected wave data about subsurface objects and sends them to the preceptor. Then, the preceptor processes and analyzes the data to extract signature patterns and format a perception of subsurface conditions. The memory module has a GIS database containing urban subsurface condition attributes and spatial locations. The cognitive analyzer carries out machine learning based on both the processing results from the perceptor and the prior knowledge about GPR measurement from the memory module to produce intelligent responses, locally or at the edge [48] for the control of radar transceiver reconfigurations.

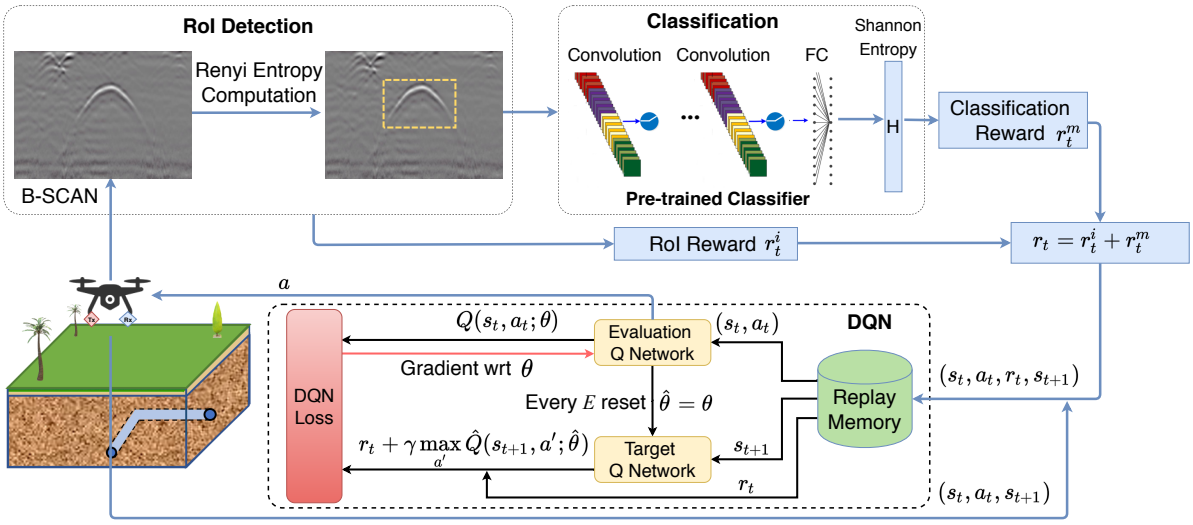


Figure 5.1 The iterative operational process of the proposed DRL-enabled AC-GPR

Although the concept of cognitive GPR in [15] pointed to promising direction in the system architecture development, no unambiguous definition or approach for a cognitive GPR was provided to build an adaptive and smart GPR that generates intelligence to adaptively adjust its operational parameters in an uncertain and dynamic sensing environment.

5.1.2 MDP Formulation of Cognitive GPR Operation

The cognitive control of the positioning and operational parameters of a GPR can be formulated as a sequential decision-making problem which can be further modeled as a finite-horizon Markov decision process (MDP) with finite state and action spaces.

Without loss of generality, we consider a discrete-time system in which time is divided into slots of unit length ΔT such that each slot t corresponds to the time duration $[(t-1)\cdot\Delta T, t\cdot\Delta T)$.

The MDP model is described as follows:

- $\mathscr{S}$: a set of environment and system operational states. Let $s_t = (_t, \Psi_t) \in \mathscr{S}$ denotes the state of the GPR sensing system and the environment in each discrete time slot t. $_t$ is the newly updated observation about the environment, in the form of a captured B-Scan image. Ψ_t is the operating state vector of the GPR, such as the remaining battery energy of the

mobile GPR platform and the agent's position $X_t \in \mathbb{C}$ (a complex number), i.e., $X_t = x_t + jy_t$, representing the GPR location with coordinates (x_t, y_t).

- $\mathscr{A}$: a set of actions of the GPR. Let $a_t = (\xi_t, \vec{v}_t, \vec{p}_t) \in \mathscr{A}$ denotes the action vector to be performed at time step t where $\vec{p}_t$ is the operational parameter values of the GPR; $(\xi_t, \vec{v}_t)$ denotes the moving direction and velocity of the GPR platform, respectively. Thus, the position of the GPR at time step t can be derived as $X_t = X_{t-1} + \vec{v}_t \cdot \Delta T \cdot e^{j\xi_t}$.

- $P_t(s, a, s^{'}) = Pr(s_{t+1} = s^{'} | s_t = s, a_t = a)$: the probability of transition from state s to state $s^{'}$ under action a.

- K : horizon over which the GPR will act.

In the proposed research, the core problem of the MDP is to find a "policy" for the GPR: a function π that specifies the action $a_t = \pi(s_t)$ that the GPR will choose in state s_t to maximize its accumulative knowledge about the subsurface object over horizon K : $\mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t(s_t, a_t)]$ where $\mathbb{E}[.]$ is the expectation taken over $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$ and $\gamma (0 \leq \gamma \leq 1)$ is the discount factor of the reward at different time steps. Due to the extreme curse of dimensionality in the state space $\mathscr{S}$ and the immense challenge of identifying transition probability $P(s_{t+1}|s_t, a_t)$, it is impractical to use exact methods such as linear programming and dynamic programming to solve the MDP problem.

To overcome this challenge, we will investigate a DRL framework where an AC-GPR agent is reinforced to learn a policy. As a computational methodology for automated decision-making of intelligent agents in uncertain environments, DRL has progressed tremendously in the past decade [35, 62]. DRL is concerned with how a decision-making agent ought to take actions from a given state of an environment so as to maximize some notion of cumulative reward. The full potential of DRL requires the agent to directly interact with the environment to attain a flow of real-world experience, as shown in Figure 3.1.

5.1.3 The Architecture of the Proposed AC-GPR

In this section, we present an overview of the proposed AC-GPR architecture, as shown in Figure 5.1. The architecture has an iterative operational process involving environment

observation, reward identification, DQN-based policy learning, and action execution. The observations (B-Scan images) from the AC-GPR agent are feed into an RoI detection module where an RoI, for example, an image area including a hyperbolic signature resulting from a subsurface rebar, is identified and extracted through the image segmentation technique using Rényi entropy and Otsu method [63, 64]. The pre-trained classifier receives the RoI image as input for classification. The classification probability output is used to characterize the classification confidence. The output results from the RoI module and classification module are used to form the reward for the AC-GPR agent, which will be described in Section 5.2.1.

The DQN module takes a tuple of state, action, reward, and the future state as experience, and guides the AC-GPR agent to learn an optimum policy that maximizes the future discounted reward. The algorithm of the DQN module will be described in Section 5.2.2. As one of value-based DRL methods, DQN is considered because it provides a better sample efficiency and more stable performance compared with policy gradient methods that have the drawback of high variance in estimating the gradient.

5.2 The Proposed DRL Approach

The cognitive analyzer in Figure 3.3 is a critical component of the proposed AC-GPR. It produces intelligence to direct the GPR movement and its operational configurations based on the collected GPR data and prior knowledge about GPR measurement. This section presents a DRL approach to the implementation of the cognitive analyzer with a novel rewarding mechanism.

5.2.1 Reward Function

The AC-GPR agent is rewarded through the outcome of RoI detection and object classification while interacting with the environment. The reward function $r : \mathscr{S} \times \mathscr{A} \rightarrow \mathbb{R}$ is derived by combining two types of rewards that are computed based on Rényi entropy and Shannon entropy, respectively, that effectively characterize the AC-GPR agent's newly acquired subsurface knowledge about the subsurface object from the sensory data.

The rationale behind this combination of the two sub-rewards is that the AC-GPR agent would receive reward r_t^i when it identifies an RoI in the B-scan image and receives reward r_t^m when

it recognizes some object properties, such as the diameter and material of a subsurface pipeline, through GPR data classification.

Thus the overall reward function is

$$r_t(s_t, a_t) = \eta r_t^i + \rho r_t^m, \tag{5.1}$$

where r_t^i denotes the region of interest (RoI) detection reward; r_t^m denotes object recognition reward; and η and ρ denotes the weight coefficients whose values are determined based on the relative importance of the RoI detection reward and the object recognition reward.

In this subsection, we present in detail the concept of RoI detection reward and subsurface object classification reward.

5.2.1.1 Reward Function Based on RoI Detection

Prior to the RoI detection, a B-Scan image first goes through the following preprocessing steps: signal denoising by removing the DC component (arithmetic mean) from each trace of the GPR image, time-zero correction which adjusts all traces to a common time-zero position where the first break of air-wave is observed, and signal enhancement/amplification [3]. The pre-processed B-Scan image is input to the RoI module in Figure 5.1 to identify a peculiar area of the image, such as a hyperbola signature, through computing Rényi entropy and Otsu threshold, generating an extracted RoI image. Rényi entropy is preferred because of its high level of accuracy on signal processing tasks compared to Tsallis [65, 66, 67]. Rényi entropy has been considered in vast domains such as structure health monitoring clutter rejection for intrawall diagnostics [65], tracking electroencephalographic signals changes [66], and cardiac autonomic neuropathy in diabetic patients [67].

In this work Rényi entropy is calculated to recognize the singular region on a B-scan image. In particular, a high Rényi entropy value demonstrates a high level of information similarity while a low Rényi entropy value features a high level of information peculiarity [63]. Let $Z(\tau)$ denotes the collected GPR reflection signal which can be depicted as

$$Z(\tau) = D(\tau) + \zeta(\tau), \tag{5.2}$$

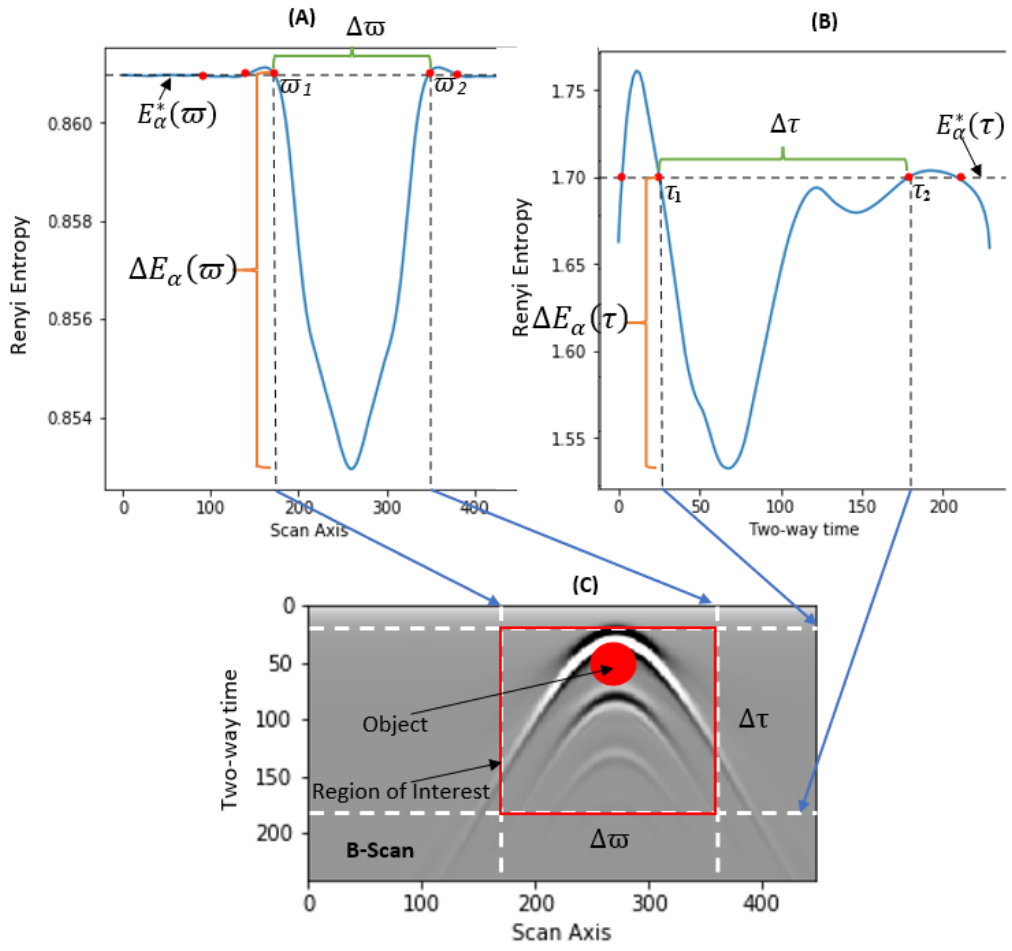


Figure 5.2 Region of interest detection process

where $D(\tau)$ represents the reflection signal from the object of interest, and $\zeta(\tau)$ models remaining interference and noise upon preprocessing. In calculation, power normalization is first performed with the summation of the power of the same time index data points on different traces, which can be expressed as

$$z_i(\tau) = \frac{||Z_i(\tau)||^2}{\sum_{i=1}^{I} ||Z_i(\tau)||^2}, \tag{5.3}$$

where $z_i(\tau)$ is the normalized signal, i is the trace index, I is the total number of traces included; τ is the time index of pulse data on each reflection trace waveform. Upon power normalization, to assess data singularity over the wave travel time axis (that is, y-axis) of the B-scan, a generalized

Rényi entropy is calculated as

$$E_\alpha(\tau) = \frac{1}{1-\alpha} \log_e \sum_{i=1}^{I} [z_i(\tau)]^\alpha, \tag{5.4}$$

where $E_\alpha(\tau)$ is the entropy quantification, and α denotes the entropy order. Eq. (5.4) is equivalent to the basic Shannon entropy limiting value as $\alpha \to 1$.

Similarly, Rényi entropy calculation is applied to the scanning traces axis (that is, x-axis)

$$z_j(\varpi) = \frac{||z_j(\varpi)||^2}{\sum_{j=1}^{J} ||z_j(\varpi)||^2}, \tag{5.5}$$

where $z_j(\varpi)$ is the normalized signal, j is the time index of pulse and J is the total number of time indexes; $\varpi = \delta t \cdot \vec{v}$ is the displacement along the trace axis of the pulse data. Then as shown in Figure 5.2 (A) the Rényi entropy to assess data singularity over the scanning position along the x-axis of the B-scan is

$$E_\alpha(\varpi) = \frac{1}{1-\alpha} \log_e \sum_{j=1}^{J} [z_j(\varpi)]^\alpha, \tag{5.6}$$

Figure 5.2 (A) and (B) show the entropy plots with respect to the trace axis and two-way travel time axis of a B-Scan image, respectively.

Through OTSU method, a classic image segmentation technique for extracting an object from its background, optimum OTSU thresholds are computed from Rényi entropy distribution. Let $\Im 1$ and $\Im 2$ be the OTSU thresholds where $\Im 1 < \Im 2$, and the within-class variance is represented as

$$\sigma^2 = \sum_{E_i \in [E_{min}, \Im 1, \Im 2, E_{max}]} (E_i - \mu)^2 p_i \tag{5.7}$$

where E_i denotes Rényi entropy points, μ the mean and p_i specifies E_i value occurrence frequency or the normalized probability. The optimum thresholds $\Im 1^*$ and $\Im 2^*$ can be determined by maximizing σ^2

$$\sigma^2(\Im 1^*, \Im 2^*) = \max_{\Im 1, \Im 2} \sigma^2(\Im 1, \Im 2) \tag{5.8}$$

where $\Im 1^*$ and $\Im 2^*$ denotes the lower bound and upper bound thresholds respectively, that minimizes the entropy within-class variance, this turns out to be the same as maximizing the between-class variance.

With two selected entropy thresholds $\Im 1$ and $\Im 2$, the B-scan image can be segmented into three classes of non-overlapping regions: singular region, stationary background region, and the transition region in-between. The singular region entropy values are lower than threshold $\Im 1$, the stationary background region entropy values are higher than $\Im 2$. While for the transitioning region, its entropy values are between these two thresholds.

The optimum thresholds $\Im 1^*$ and $\Im 2^*$ are determined through the Otsu method [64], a classic image segmentation technique for extracting an object from its background. Specifically, by initializing both $\Im 1$ and $\Im 2$ at zero, the Otsu method performs statistical analysis to identify appropriate thresholds so as to segment images into different regions based on the criteria: the intensity values variances of the same region is minimized while the variances of different regions are maximized. In this work when applying the Otsu method, the entropy is chosen as the intensity value [63].

Figure 5.2 (A) and (B) show the calculated upper bound Otsu thresholds: $\Im 2^* = 0.8601$ over the scan axis and $\Im 2^* = 1.70$ over the travel time axis, respectively. By identifying Rényi entropy values that intersect with the thresholds, boundary trace values (i.e. ϖ_1 = 182 and ϖ_2 = 363) and travel time values (i.e., τ_1 = 26, τ_2 = 178) can be determined with low entropy values contained in the resulting intervals. Then these four boundary values are superimposed on Figure 5.2 (C), demarcating the RoI, as shown by the red box. As an example, Figure 5.2 (C) shows the resulting RoI including a section of hyperbola where the highest value marked by the red ball is the position of the object, such as rebar or a pipe.

From the computed Rényi's probability distributions $E_\alpha(\tau)$ and $E_\alpha(\varpi)$, coupled with the optimum Otsu thresholds $\Im 2_t^* = E_\alpha^*(\tau)$ and $\Im 2_s^* = E_\alpha^*(\varpi)$, the reward for detecting the RoI is computed as

$$r_t^i = a(\Delta E_\alpha(\tau) \cdot \Delta E_\alpha(\varpi)) + b(\Delta\tau \cdot \Delta\varpi), \tag{5.9}$$

$$\Delta E_\alpha(\tau) = E_\alpha^*(\tau) - \min\{E_\alpha(\tau)\}, \tag{5.10}$$

$$\Delta E_\alpha(\varpi) = E^*_\alpha(\varpi) - \min\{E_\alpha(\varpi)\}, \tag{5.11}$$

$$\Delta\tau = \tau_2 - \tau_1, \tag{5.12}$$

$$\Delta\varpi = \varpi_2 - \varpi_1, \tag{5.13}$$

where $\Delta E_\alpha(\tau)$ and $\Delta E_\alpha(\varpi)$ denotes the Rényi entropy variation with respect to the travel time axis and the scanning position axis, respectively; $\Delta\tau$ denotes the related two-way travel time interval and $\Delta\varpi$ the related scanning position interval, indicating the dimension of the detected RoI; and a and b are weight coefficients whose values are determined based on the relative importance of the Rényi entropy variation and the RoI dimension; As higher data singularity corresponds to lower Rényi entropy, higher $\Delta E_\alpha(\tau)$ and $\Delta E_\alpha(\varpi)$ indicate a higher chance of detecting the subsurface object [63]. The proposed reward function in Equation (5.9) combines the Rényi entropy change with the RoI dimension, mitigating the false positive detection resulting from outliers.

5.2.1.2 Reward Function Based on Subsurface Object Classification

Subsurface objects can be recognized through different GPR data classification tasks, for example, determining the material type, burial depth, and diameter depending on specific applications. In GPR data processing and analysis, as shown in the top part of Figure 5.1, the RoI identified through the Rényi entropy computation is passed to a pre-trained convolutional neural network (CNN) classifier. The use of CNN is motivated by the fact that CNNs outperform other artificial neural networks on conventional computer vision tasks such as object detection [16], facial expression recognition [17], and medical imaging segmentation [18], through feature learning. Let $P = \{p(1), p(2), \cdots, P(N)\}$ denotes the classification probability output from the classifier where $p(n)$ is the class probability that the processed B-scan belongs to class n, and N is the total number of classes. The possible classes depend on the specific classification task. For example, if the classification task is to determine the material type of the subsurface object, the possible classes could be different material types, namely concrete, metallic, polyvinyl chloride (PVC), etc. As entropy is a measure of uncertainty [68, 69], in this work Shannon entropy is considered to quantify the confidence in the classification. The Shannon entropy of

the classification probability distribution P can be computed as

$$r_t^m = -\sum_{n=1}^{N} p(n)\log(p(n)), \tag{5.14}$$

It is inferred from Equation (5.14) that a balanced classification probability distribution results in high entropy indicating high uncertainty and low classification confidence while a skewed classification probability distribution has low entropy indicating low uncertainty and high classification confidence.

5.2.2 DRL Algorithm

In reinforcement learning, an agent learns to better perform tasks by learning from its experiences interacting with the environment. Our proposed deep reinforcement learning method is based on the Deep Q-Network (DQN) algorithm taking four inputs (s, a, r, s') i.e. state observation, action, reward, next state observation, via epsilon-greedy strategy.

The use of deep reinforcement learning techniques in autonomous robots has been broadly studied. The current studies incorporate the versatile operation control of robots, mechanical control, and the administration in multi-robot frameworks locally, at the edge or the cloud. According to [30], the authors used Deep Q-Network (DQN) to develop an end-to-end autonomous robotic system that incorporates path planning. Robots and humans safely coexist through socially compliant interaction with inverse reinforcement learning (RL) [31]. The solution for mission-driven robotics with visual navigation problems has been developed through DRL and AI2-THOR framework in [32].

As shown in Figure 5.1 the DQN algorithm has three main components, the evaluation Q-network $(Q(s,a;\theta))$, the target Q-network $(\hat{Q}(s,a;\hat{\theta}))$, and the replay memory. The evaluation Q-network and the target Q-network have the same network structure but different weights and biases. However, the evaluation Q-network is updated instantly for every episode, whereas the target Q-network is updated periodically after every H episodes as shown in Table 5.2, by replacing the values of the target Q-network with the evaluation Q-network values. The target network is updated only infrequently in order to mitigate the risk of non-stationarity of the target values in

the loss function in Eq. (5.17) caused by the feedback loops between the target and estimated Q-values. The generated Q-values, through a replay memory with random batch size of B as shown in Table 5.2, are used to compute the DQN loss in Eq. (5.17).

The AC-GPR is inclined to execute the action with the highest Q-value derived from each episode. The Q-value corresponding to the pair of state and action represents an expected discounted accumulated future reward.

As discussed in Section 5.2.1, the AC-GPR agent received reward $r_t = r(s_t, a_t)$ at each time instance t. The accumulated reward is defined as $R_t = \sum_{t=0}^{T} \gamma^t r_t$ with a discounting factor $\gamma \in [0,1]$. The action-value function under action policy π is defined as $Q^{\pi}(s_t) = \mathbb{E}[R_t|s_t, \pi]$, and the optimal policy is determined by estimating the action-value function Q^{π}. The estimation can be obtained by Bellman equation recursively [70]. An episode is terminal on one of the following two conditions: 1) when AC-GPR detects a subsurface object, or 2) when AC-GPR observes a B-Scan with a predefined number of traces (A-Scans).

To improve the convergence of DQN, the agent's experience at each time step t is stored at the replay memory, where a batch B of experiences are randomly sampled, as shown in Table 5.2. This process is called experience replay that provides diverse and decorrelated training data and solves the issue of correlated inputs/output [71]. Let e_t represent the agent's experience at time t which is defined as

$$e_t = (s_t, a_t, r_t, s_{t+1}). \tag{5.15}$$

This tuple contains the state s_t, the action a_t taken at state s_t, the reward r_t given to the agent at time t as a result of the previous state-action pair (s_t, a_t), and the next state s_{t+1}. The replay memory is set to a finite size limit L, and therefore, it will only store the latest L experiences.

The DQN utilizes a CNN as a nonlinear approximation to the optimal action-value function $Q(\theta) \rightarrow Q^*$. All parameters of the network are denoted as θ, and the parameters are estimated iteratively by minimizing the temporal difference error as

Algorithm 3: The cognitive subsurface object detection algorithm based on Deep Q-network

Initialize replay memory D to capacity L
Initialize Q with random parameter θ
Initialize $\hat{Q}$ with weights $\hat{\theta} = \theta$
for *episode = 1, M* **do**
  Initialize state s_1
  for *t=1, T* **do**
    Select a random action a_t with probability ε;
    Otherwise select $a_t = arg\max_a Q(s_t, a; \theta)$;
    Execute action a_t, capture B-Scan image, and compute reward $r_t = \eta r_t^i + \rho r_t^m$;
    Set $s_{t+1} = (x_{t+1}, y_{t+1})$,
    Store transition (s_t, a_t, r_t, s_{t+1}) in D;
    Sample random batch (s_j, a_j, r_j, s_{j+1}) from D;
    Calculate target y_j

$$y_j = \begin{cases} r_j & \text{if terminal } s_{j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \hat{\theta}) & \text{else;} \end{cases}$$

    Perform a gradient descent step in Eq. (5.17) with respect to network parameters θ;
    Every H steps reset $\hat{Q} = Q$ i.e. set $\hat{\theta} = \theta$.
  end
end

$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}\left[\left(r_t + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; \hat{\theta}) - Q(s_t, a_t; \theta)\right)^2\right]. \tag{5.16}$$

The optimization is converted into a regression problem in DQN by regarding temporal difference error as loss

$$L(\theta) = \Big[\underbrace{r_t + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; \hat{\theta})}_{\text{target y}} - Q(s_t, a_t; \theta)\Big]^2. \tag{5.17}$$

In the training process, y is regarded as the target of the regression function. The loss function is used to train the neural network to adjust parameters θ.

After training, parameters θ^* are obtained, and the algorithm stops after a finite number of steps M with the optimal policy $V^{\pi_M} = V^*$. With approximation, the optimum policy maximizes the future discounted reward as

$$\forall s \;\; \pi^*(s_t) = \arg\max_a \sum_{s_{t+1}} P_{s_t,a_t}(s_{t+1}) V^*(s_{t+1}), \tag{5.18}$$

where $P_{s_t,a_t}(s_{t+1})$ is the state transition probability as discussed in Section 5.1.2, and V^* the optimum value function with convergence $\pi_{t+1}(s_{t+1}) = \pi_t(s_t)$. Note that the computation of both $P_{s_t,a_t}(s_{t+1})$ and the sum in Eq. (5.18) is avoided due to the action-value function approximation through $Q(\theta)$.

After learning the policy, the agent needs to perform action selection and execution. The agent faces the well-known exploration-exploitation dilemma of whether to exploit the current knowledge by following the learned policy or to continue to explore the uncertain environment to acquire more knowledge. To resolve the dilemma, the agent adopts an ε-greedy-based policy for selecting actions where $\varepsilon \in [0,1]$ denotes the exploration probability. The configuration that $\varepsilon = 0$ would result in the pure greedy policy that always selects the action corresponding to the highest Q-value. The pure greedy method may cause the policy learning to get stuck at a local optima. In contrast, the configuration that $\varepsilon = 1$ would result in the pure random policy that selects an action randomly without considering the Q-value.

As described in Algorithm 3, at first (Lines 1-3), the network parameters are initialized randomly. To enhance the learning stability, the target network is introduced which has the same structure as the evaluation network. Then, the exploration process is conducted, and the action is derived from the DQN. The DQN employs the RoI and classifier modules to update the proposed reward function (line 9), by combining the RoI detection reward and object classification reward, which intuitively measures the amount of the acquired information about the subsurface object from the observed B-Scan image. Next (Lines 11-14), the experiences are stored into the replay memory. Then the minibatch method is used to randomly collect examples from the replay

memory. The weights and biases of the network are updated by training the DQN according to the loss function (5.17). The training process will terminate once it reaches a predefined number of episodes. During each episode, the AC-GPR agent stops performing actions after a predefined number of time steps, or could terminate the episode early if it detected the subsurface object. The computational complexity of AC-GPR algorithm is expressed as $\mathcal{O}(MT)$, where M denotes the total number of episodes and T the number of time steps.

5.3 Performance Evaluation and Discussion

In this section, we systematically evaluate the performance of the proposed AC-GPR by using a GPR simulator called gprMax [60, 56] designed for modeling GPR operations. B-Scan images are generated by gprMax on the fly in mimicing the GPR data acquisition in real environment.

5.3.1 Experiment Settings

5.3.1.1 GprMax Simulator

The gprMax simulator used in this study solves two dimensional (2D) Maxwell equations using the Finite-Difference Time-Domain (FDTD) method [72]. The simulation in this work considers small diameter pipes or rebars made of three types of materials, namely, concrete, metallic and PVC, as subsurface objects. The gprMax characterizes the impact of common pipe materials on resulting GPR data based on the dielectric constant also called relative permittivity which indicates how easily a material can become polarized. Relative permittivity, defined as the ratio of the permittivity of a substance to the permittivity of space of vacuum, is expressed as $\varepsilon = \frac{C}{C_o}$, where C denotes the capacitance of the material as the dielectric capacitor, $C_o = \frac{\varepsilon_o A}{d}$ represents the capacitance using vacuum as the dielectric, ε_o the permittivity of free space (8.85 x 10-12 F/m i.e. Farad per metre), A the area of the sample cross section area, and d the thickness of the sample. The dielectric constant for PVC, concrete and metal are 4.0, 4.94 and infinity, respectively.

GprMax uses a mixing model for modeling radio propagation in soil [59]. The soil composition involves sand fraction 0.3, clay fraction 0.1, bulk density $2g/cm^3$, sand particle density of $2.66g/cm^3$, and a volumetric water fraction range of 0.001 - 0.25.

The GPR dataset contains B-Scan images from the gprMax simulator, which was used to train the classifier, as shown in Figure 5.1. Figure 5.3 shows some example B-Scan images which are corresponding to concrete, metallic and PVC objects. Some images have sharp, dim or no hyperbola due to different factors including, but not limited to, object material, burial depth, soil dielectric properties, GPR antenna configuration and orientation.

In this work, we generated a total of 40320 images with each material type having 13440 B-Scan images, derived from varying soil dielectric properties and object diameter. All the 40320 images are unique, each exhibiting a different level of contrast, shape and size on hyperbola signature. The B-Scans from concrete objects exhibit weak or no hyperbola at all due to the fact that the signals are attenuated as they propagate through the soil and through the object hence weakening the signal reflection. The PVC objects have a slightly higher dielectric constant than concrete objects; accordingly, they produce B-Scans with a slightly high hyperbola contrast. B-Scans from metallic objects have strong and high resolution hyperbola contrast due to the high relative permittivity of metals which allows for strong reflected signals.

5.3.1.2 RoI Detection

The RoI detection module in Figure 5.1 receives a preprocessed B-Scan image and identifies a possible RoI with the method detailed in Section 5.2.1. The RoI is resized into a 64 x 64 grayscale images which will be feed into the classification module, as illustrated in Figure 5.2.

5.3.1.3 Object Classification

The pre-trained classifier in Figure 5.1 is configured with two hidden convolutional layers. The first layer has 32 filters, kernel size 3 applied with stride 2, while the second one has 64 filters, kernel size 3 with stride 2. All two layers are followed by a LeakyReLU non-linearity. This is followed by a fully connected layer with 256 units, and three-class output layer and softmax, learning rate of 10^{-3} and batch size of 64. Based on the classification outcome, i.e., the probability values of different classes, Shannon entropy is calculated. During training a high class prediction probability means low uncertainty hence a high confidence of a particular class. Table 5.1 summarizes the hyperparameters of the CNN-based object classifier.

Table 5.1 Architecture parameters

Layer (type)	**CNN Object Classifier**	**DQN Evaluation Network & Target Network**
Conv2D	Filter: 32 Kernel size: 3 Stride: 2 Padding: Same Activation: LeakyReLU	Filter: 256 Kernel size: 3 Activation: ReLU
MaxPooling	Pool size: 2	Pool size: 2
Dropout	Rate: 0.3	Rate: 0.5
Conv2D	Filter: 64 Kernel size: 3 Stride: 2 Padding: Same Activation: LeakyReLU	Filter: 256 Kernel size: 3 Activation: ReLU
MaxPooling	Pool size: 2	Pool size: 2
Flatten	Output shape: 1024	Output shape: 50176
Dense	Classes: 3	Classes: $\mathscr{A}$

5.3.1.4 DQN

Simulations are conducted on a core i7 computer with four cores, 2.2 GHz Intel Xeon CPU, and 16GB RAM. The training process is run with Python 3.6 and TensorFlow 1.10.0. The size of the replay memory is 5×10^4, and the sample mini batch is $B = 32$. During the training process, the GPR agent interacts with the environment and receives tuples of state, action, reward and next state. A total of 5×10^4 such tuples are stored in the replay memory as experiences which are then sampled and used during the learning. The agent starts by exploring the environment to build knowledge about transitions and action rewards. Then through decaying the exploration probability ε, the agent gradually exploits the gathered information to detect subsurface objects. Tables 5.1 and 5.2 summarize the architecture parameters and network hyperparameters of DQN, respectively.

Table 5.2 Deep Q network settings

Hyperparameter	**Value**	**Description**
Learning rate (α)	0.01	The learning rate used in ADAM to update the DQN
Discount factor (γ)	0.99	The discount factor used in the reinforcement update
Epsilon (ε)	$[0, 1]$	Exploration probability
Replay memory size (L)	5×10^4	Number of the most recent experiences stored in the replay memory
Mini batch size (B)	32	Number of experiences used for training
Epsilon decay	0.99975	Exploration probability decay factor
Episodes (M)	15000	Number of Episodes
Frequency of target network update (H)	5	Number of time steps before the target network is updated

5.3.2 Performance Results

5.3.2.1 Detection Accuracy vs. Noisy B-Scan Data

In this thesis, we evaluate the performance of the proposed AC-GPR under different levels of clutter noise caused by clutter from heterogeneous soil.

Based on the model of radio propagation proposed by Peplinski [59], heterogeneous soils are modeled by considering sand fraction, clay fraction, bulk density, sand particles density, and the range for volumetric water fraction. The noise levels are modeled by adjusting both sand and clay fraction from 0.1 to 0.9. The clutter noise level increases as the fraction value increases. To make the clutter noise levels more distinct, different types of soil surfaces including smooth, rough, water, and grass surfaces, are added to the fractal-box (a box that houses Peplinski heterogeneous soil).

Interpreting noisy B-scan data caused by clutter noise is challenging, and sometimes it is impossible to extract some knowledge about the subsurface object from the data. In this work, the likelihood of successful object detection and recognition is used to evaluate the performance of the proposed AC-GPR on this aspect. Specifically, a performance metric termed detection

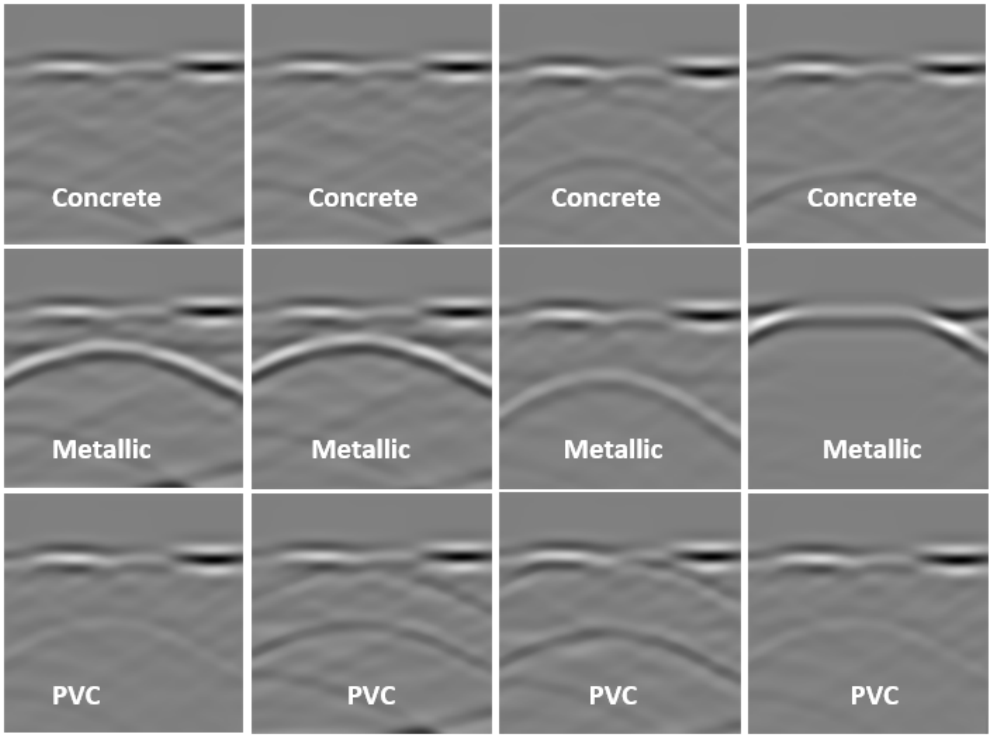


Figure 5.3 Sample B-Scan images of the dataset

accuracy is defined as $\frac{w}{(w+l)}$ where w denotes the number of episodes with successful object detection, and l the number of episodes with failed detection. Successful object detection and recognition are characterized by high confidence results of object detection and material type classification, respectively, from a B-Scan image. Figure 5.5 shows that the resulting detection accuracy decreases along with the increase of the level of the clutter noise caused by the increased clay or sand fraction.

5.3.2.2 Detection Accuracy vs. Object Diameters and Burial Depths

To simulate real-world scenarios, the performance of the proposed AC-GPR was also tested through modeling various object diameters and burial depths in dry sand, pavement and heterogeneous soil. Figure 5.6 shows that as the diameter of the object increases the detection accuracy increases. This is because with a larger object diameter more signals are reflected back to the GPR receiver, hence generating B-Scans with higher resolution hyperbolas. As shown in Figure

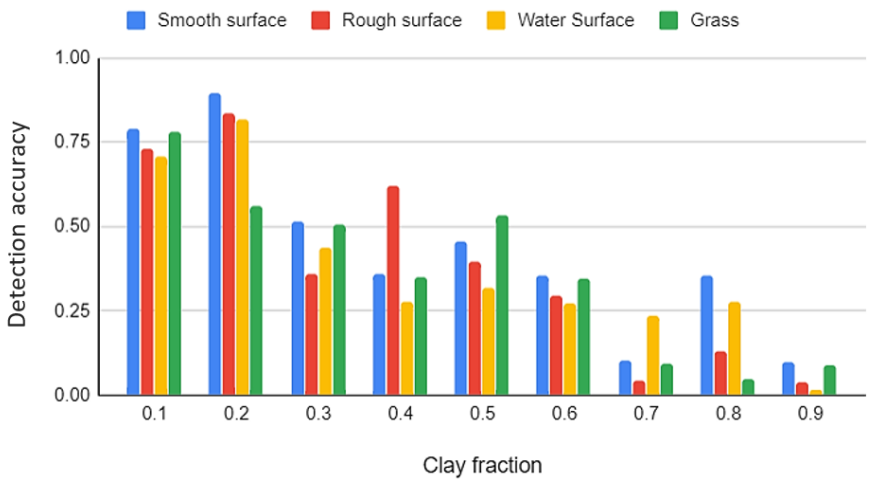


Figure 5.4 AC-GPR performance with noisy B-Scan images with varying clay factions

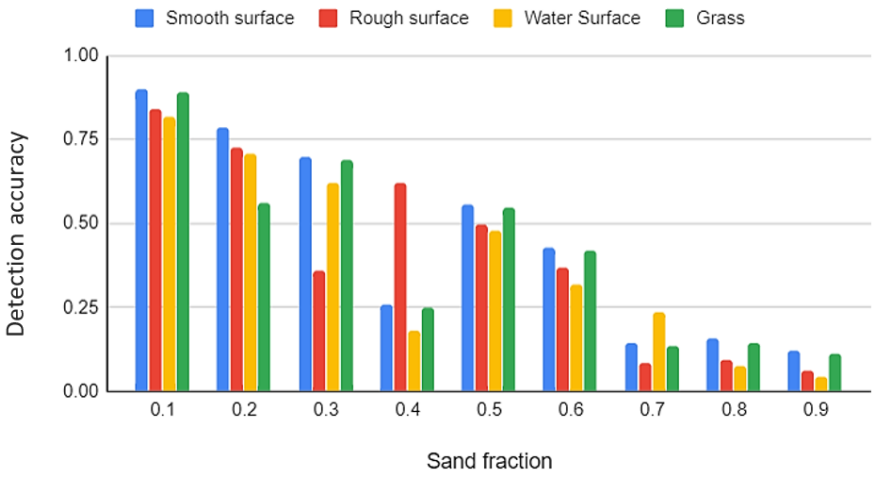


Figure 5.5 AC-GPR performance with noisy B-Scan images with varying sand fractions

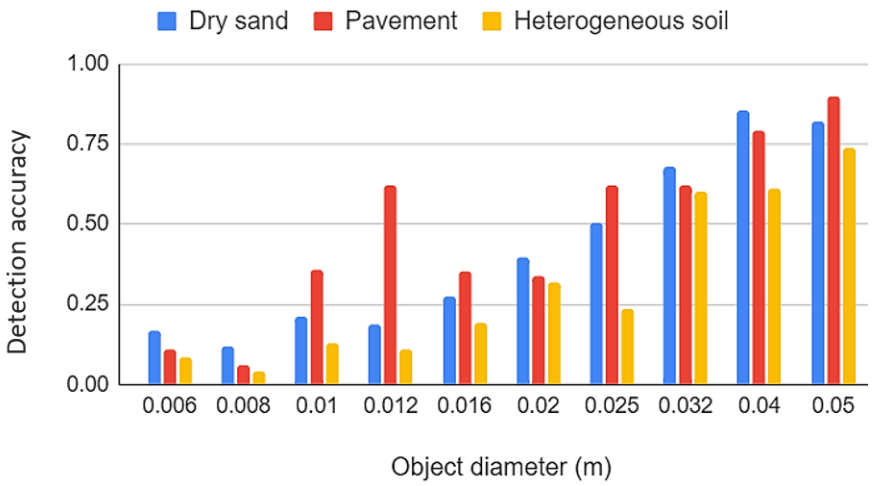


Figure 5.6 AC-GPR performance with varying object diameter

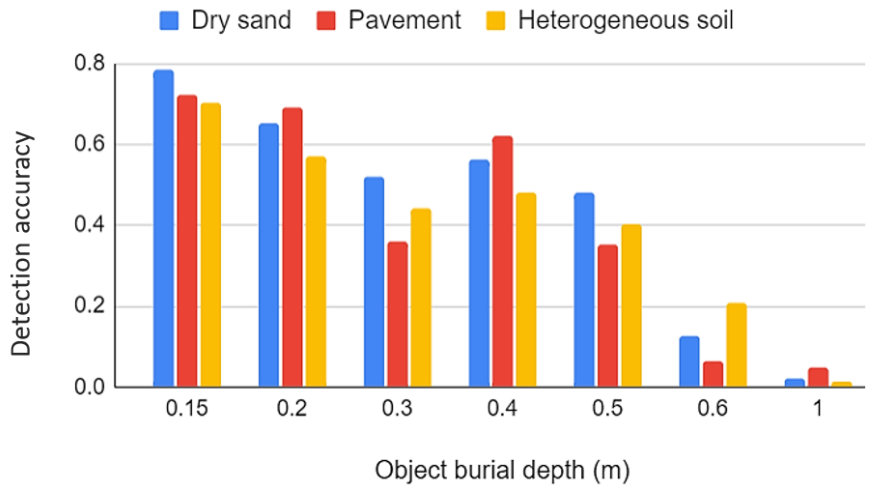


Figure 5.7 AC-GPR performance with varying object burial depth

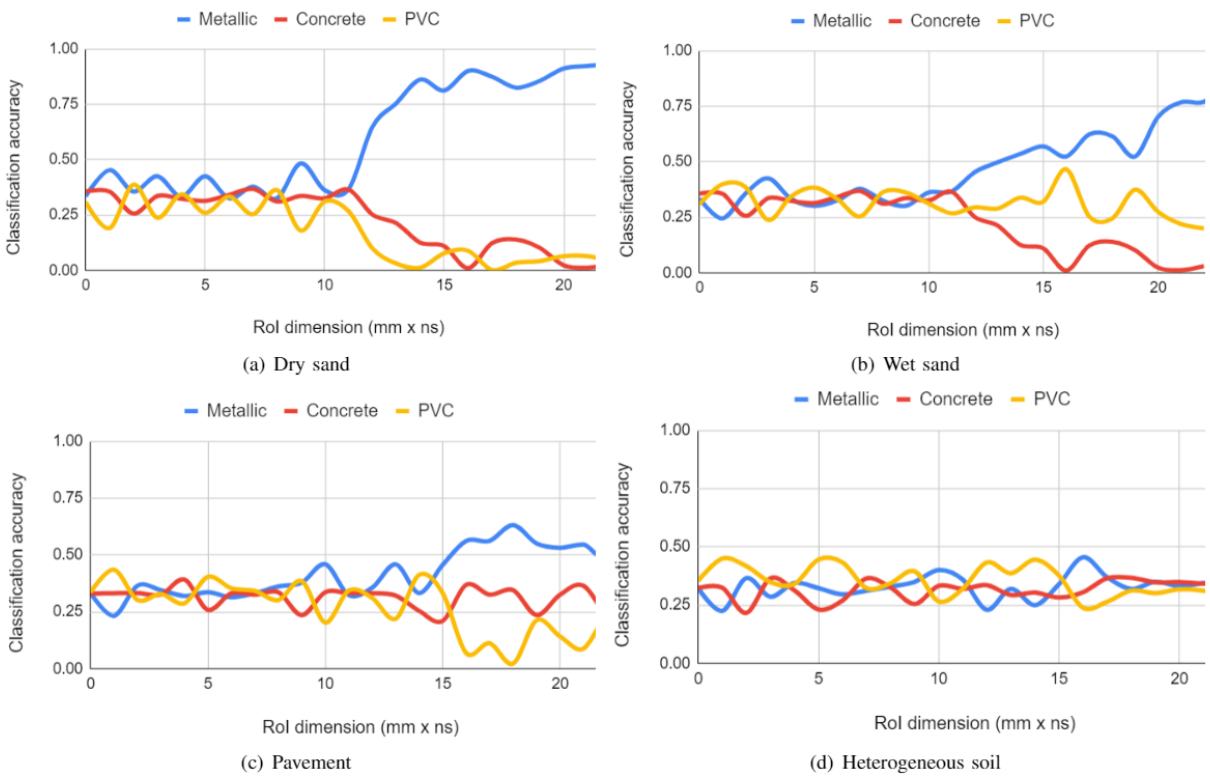


Figure 5.8 Classification accuracy vs. RoI dimension

5.7, the detection accuracy decreases as the burial depth of the object increases. This is because the deeper the object is buried the more attenuation the signals incur as they propagate through the soil, resulting in weaker reflection and fainter hyperbolas that makes it more challenging for the AC-GPR to detect and recognize the subsurface object.

5.3.2.3 Classification Accuracy vs. RoI Dimension

As GPR data interpretation is affected by the size of the detected RoI, the impact of RoI dimension on the classification accuracy of the classifier was also evaluated. Figure 5.8 shows the classification accuracy vs. RoI dimension for metallic, concrete, and PVC objects buried in four different media: dry sand, wet sand, pavement, and heterogeneous soil. As shown in Figure 5.8(a), the classification for metallic objects in dry sand has the highest accuracy. The level of accuracy increases as the RoI dimension increases. Figure 5.8(b) shows a slight decline in classification accuracy in wet sand with we compared with the result of dry sand. That is because the more water

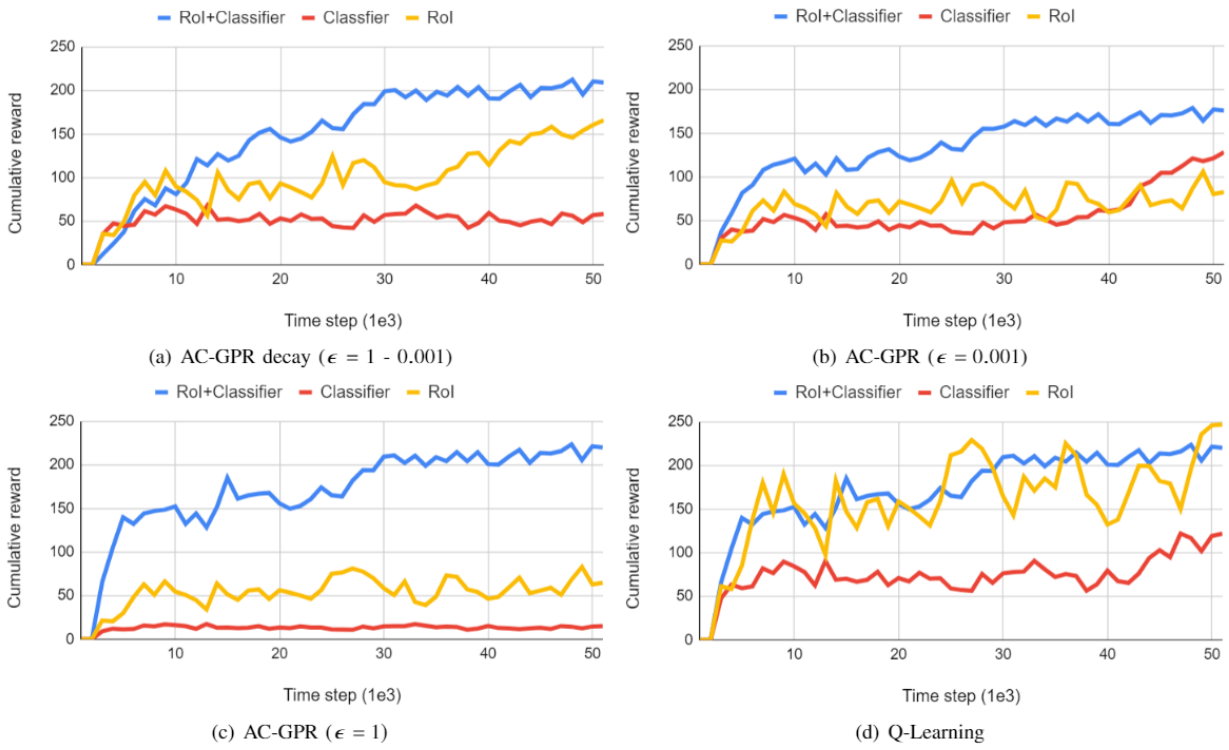


Figure 5.9 Cumulative reward vs. time steps of different ε-greedy policies and rewards types

content in the wet sand caused more signal attenuation. Figure 5.8(c) and Figure 5.8(d) shows the low classification accuracy in Pavement and heterogeneous soil, respectively. This is because the severe signal attenuation in the pavement and heterogeneous soil resulted in weak signature of object in the RoI, such as a hyperbola. Even though the RoI dimension increases, the classifier fails to produce a high level of confidence about the classification of the object material.

5.3.3 Convergence Analysis

5.3.3.1 Comparison Between Reward Functions

In order to evaluate the proposed reward function, we study the impact of different reward function variations on performance convergence. In the evaluation, three types of rewards were considered, that is, the reward only from RoI detection, the reward only from object classification, and the combined rewards from both RoI detection and object classification. The rewards were computed from a heterogeneous soil setup containing varying clay and sand fractions (0.1-0.9)

with an object having a diameter of 0.05 meters at a burial depth of 0.3 meters. For comparison, different AC-GPRs with different ε-greedy policies were investigated. In addition, a Q-Learning based method was also evaluated by considering a fairly small state space. Figure 5.9 shows the cumulative reward vs. time step in different cases. It is observed that the AC-GPR using the proposed combined rewards outperforms the systems using other types of rewards.

5.3.3.2 Time Steps to Reach Convergence

The time needed for the proposed AC-GPR to reach converged performance was also evaluated. The time steps to reach convergence for different AC-GPR implementations adopting different ε-greedy policies are displayed in Table 5.3. It is shown that the proposed AC-GPR configured with decaying-epsilon-greedy policy (ε decay) outperforms the rest. Additionally, the proposed AC-GPR using the combined reward demonstrates better convergence performance.

Table 5.3 Time steps convergence comparison

Model	**RoI Detection**	**Object Classification**	**RoI Detection + Object Classification**
AC-GPR (ε decay)	1.53×10^3	1.34×10^3	1.28×10^3
Q-Learning	3.56×10^3	4.07×10^3	3.39×10^3
AC-GPR ($\varepsilon = 0.001$)	4.23×10^3	4.19×10^3	4.04×10^3
AC-GPR ($\varepsilon = 1$)	7.27×10^3	7.15×10^3	7.12×10^3

5.4 Summary

In this chapter, an autonomous cognitive GPR (AC-GPR) based on deep reinforcement learning (DRL) was developed. A novel reward function was developed such that the AC-GPR agent is rewarded from both region of interest (RoI) detection and object classification. With the proposed reward function a DQN-based model was developed to enable the AC-GPR to learn to take optimal actions that maximize the long term discounted reward, hence detecting and identifying subsurface objects from its experiences of interacting with the environment. The proposed AC-GPR was evaluated by adopting the GPR operation simulator, gprMax, and

simulating real-world environment and GPR operations. Simulation results show the proposed AC-GPR has superior performance over other GPR systems in terms of object detection detection accuracy, object classification accuracy, and convergence.

In the next chapter, an extension into 3D scan cloud approach is studied. Scan cloud applies advanced machine learning algorithms and models for an AC-GPR that incorporate effective continuous GPR signal processing methods and GPR data processing approaches suitable for detecting and modeling objects in complicated environments.

5.5 Future Work

It is worth noting that the proposed approach for AC-GPR is suitable for the detection of a single subsurface object with simple structural configurations, such as a pipe or rebar, and under a relatively homogeneous environment. However, in a real-world environment, detection of subsurface objects involve multiple challenges including inherent uncertainties and complexities of the environment for electromagnetic wave propagation and GPR data acquisition, scarcity of ground-truth GPR dataset for model training, and structural heterogeneity of different subsurface objects. Therefore, as part of our future work, real-world field evaluation and testing of the proposed framework and algorithms will be conducted to examine its capabilities and advantages.

CHAPTER 6

COGNITIVE GPR BASED ON SCAN CLOUD DATA

Accurate 3D maps for underground infrastructures, like gas, water, and sewage pipes, are significant for governments, service organizations, and structural architects. Nonetheless, the exact locations and conditions of underground infrastructure in old cities are generally unknown. A ground-penetrating radar (GPR) is a significant device for locating and identifying underground objects [73]. However, there are some limitations with conventional GPR techniques: piecemeal analysis of different formats of GPR data (A-Scan, B-Scan, and C-Scan) and offline interpretation of GPR data with the need of domain expert experience. These limitations result in inaccurate and incomplete knowledge of underground objects and time-consuming operation of GPR systems.

To address these limitations, we propose a new approach termed *scan cloud* that considers all the GPR A-Scans in the field as a point cloud. The rationale behind scan cloud is that holistically analyzing all the in-situ available GPR data may provide more knowledge than focusing on partial A-Scan data or individual B-Scan images. Specifically, we model the GPR sensing process by integrating the 1D A-Scan signals and analyzing them to extract (x, y, z) coordinates of the signal where the amplitude pulse from the object is detected. This information is stored in a database to form a novel 3D scan cloud dataset. This allows us to develop new data point each time an A-Scan is received.

We also integrate scan cloud analysis and deep reinforcement learning (DRL) to adapt the operation of a GPR system. We first apply Rényi entropy and modified 3D OTSU methods to detect the region of interest (RoI) in the scan cloud. The RoI is transformed into a signature that serves as input into an object classifier. All the previous procedures result in a reward function for the deep deterministic policy gradient (DDPG) [74, 75], a typical DRL method dealing with large and continuous state and/or action spaces. This chapter is focused on the development of a DDPG

framework that enables adaptive GPRs for subsurface object detection. To this end, a proper reward function is needed to effectively reflect the value of different actions of the GPR agent at different states. To the best of our knowledge, this is the first work on subsurface object detection based on scan cloud and DDPG. The main contribution of this method can be summarized as follows.

1. By formulating GPR-based subsurface object detection as a Markov decision problem, a 3D data modality and a deep reinforcement learning framework is established to resolve the problem.

2. A deep deterministic policy gradient (DDPG) algorithm with a novel reward function that combines rewards from amplitude analysis, Region of Interest (RoI) identification and object classification is proposed.

3. To show the efficacy of the proposed framework, simulation based validations are performed on real-time GPR data from gprMax simulator by combining DDPG with GPR operation modeling [60, 56].

6.1 The Proposed System Model and Architecture

6.1.1 MDP Formulation of Subsurface Detection

The cognitive control of the positioning and operational parameters of a GPR can be formulated as a sequential decision-making problem which can be further modeled as a finite-horizon Markov decision process (MDP) with finite state and action spaces.

Without loss of generality, we consider a discrete-time system in which time is divided into slots of unit length ΔT such that each slot t corresponds to the time duration $[(t-1)\cdot\Delta T, t\cdot\Delta T)$.

The MDP model is described as follows:

- $\mathscr{S}$: a set of environment and system operational states. Let $s_t = (_t, \Psi_t) \in \mathscr{S}$ denotes the state of the GPR sensing system and the environment in each discrete time slot t. $_t$ is the newly updated observation about the environment, in the form of captured A-Scan signal. Ψ_t is the operating state vector of the GPR, such as the remaining battery energy of the mobile GPR platform and the agent's position $X_t \in \mathbb{C}$ (a complex number), i.e., $X_t = x_t + jy_t$, representing the GPR location with coordinates (x_t, y_t).

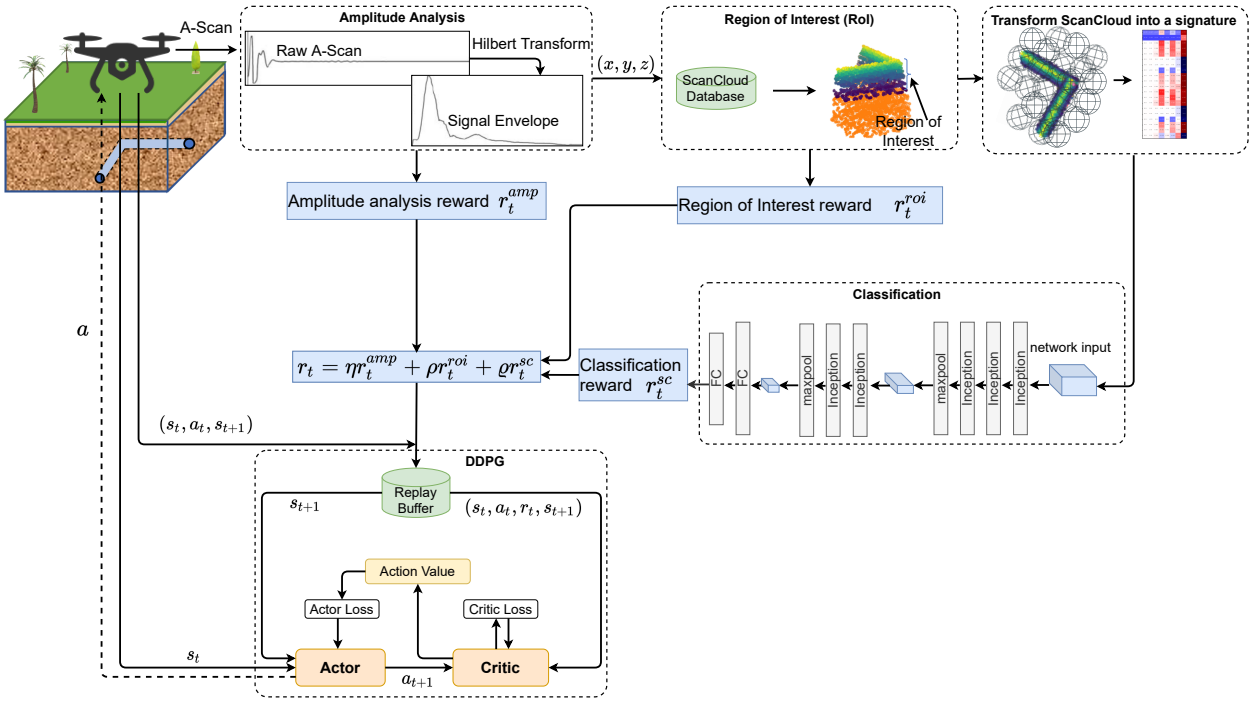


Figure 6.1 Proposed architecture

- $\mathscr{A}$: a set of actions of the GPR. Let $a_t = (\xi_t, \vec{v}_t, \vec{p}_t) \in \mathscr{A}$ denotes the action vector to be performed at time step t where $\vec{p}_t$ is the operational parameter values of the GPR; $(\xi_t, \vec{v}_t)$ denotes the moving direction and velocity of the GPR platform, respectively. Thus, the position of the GPR at time step t can be derived as $X_t = X_{t-1} + \vec{v}_t \cdot \Delta T \cdot e^{j\xi_t}$.

In reinforcement learning for discrete action spaces, exploration is done via probabilistically selecting a random action such as ε-greedy. For continuous action spaces, exploration is done via adding noise to the action itself. In this thesis, we will investigate a DDPG framework where a GPR agent is reinforced to learn both a Q-function and a policy.

6.1.2 The Architecture of the Scan Cloud based Cognitive GPR

In this section, we present an overview of the proposed scan cloud architecture, as shown in Figure 6.1. The architecture has an iterative operational process involving environment observation, reward identification, DDPG-based policy learning, and action execution. The observations (A-Scan signals) from the agent are feed into the amplitude analyzer for x, y, z

coordinates extraction, which are stored in the scan cloud database. From the accumulated scan cloud points a region of interest (RoI) is detected in the object formation module where the RoI is made of points with high amplitude values signifying the presence of a subsurface object. These RoI points are identified and extracted through the modified 3D OTSU technique and Rényi entropy [64, 63]. The pre-trained classifier receives the RoI signature representation as input for classification. The classification probability output is used to characterize the classification confidence. The output results from the amplitude analyze module, RoI module, and classification module are combined to form the reward for the GPR agent, which will be described in Section 6.2.1.

The DDPG module takes a tuple of state, action, reward, and future state as experience, and guides the GPR agent to learn a Q-function and the optimum policy that maximizes the future discounted reward for continuous action spaces. The algorithm of the DDPG module will be described in Section 6.2.2. DDPG uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy.

6.2 The Proposed Scan Cloud Approach

The region of interest (RoI) module in Figure 6.1 is a critical component of the proposed method. It produces intuitive information about the underground object, which is used to direct the GPR movement and its operational configurations based on the collected scan cloud data and prior knowledge about GPR measurement. This section presents a DDPG approach to the implementation of the 3D subsurface object detection method with a novel rewarding mechanism.

6.2.1 Reward Function

The GPR agent is rewarded through the combined outcomes from amplitude analysis, RoI detection and object classification while interacting with the environment. The reward function $r : \mathscr{S} \times \mathscr{A} \rightarrow \mathbb{R}$ is derived by combining three types of rewards that are computed based on Hilbert Transform, alpha shapes and Shannon entropy, respectively, that effectively characterize the GPR agent's newly acquired subsurface knowledge about the subsurface object from the sensory data.

The rationale behind this combination of the three sub-rewards is that the GPR agent would receive reward r_t^{amp} from analysing the A-Scan's amplitude, r_t^{roi} when it identifies an RoI in the A-Scan image and receives reward r_t^{sc} when it recognizes some object properties, such as the diameter and material of a subsurface pipeline, through GPR data classification.

Thus the overall reward function is

$$r_t(s_t, a_t) = \sigma r_t^{amp} + \zeta r_t^{roi} + \vartheta r_t^{sc}, \tag{6.1}$$

where r_t^{amp} denotes the amplitude analysis reward, r_t^{amp} the region of interest (RoI) detection reward, r_t^{sc} object recognition reward and σ, ζ, ϑ denotes the weight coefficients whose values are determined based on the relative importance of the amplitude analysis reward, RoI detection reward and the object recognition reward.

In this subsection, we briefly present the concept of amplitude analysis reward in Subsection 6.2.1.1, RoI detection reward in Subsection 6.2.1.2, and then describe the subsurface object classification reward in Subsection 6.2.1.3.

6.2.1.1 Reward Based on Amplitude

GPR works by sending a signal from a receiver into a surface. The signal is reflected by any materials it encounters within the surface, and creates a reading. Each reflected A-Scan pulse from the reflected signal off the subsurface object is examined to produce a tuple (x, y, z) that forms the coordinates of the scan cloud points, where x, y denotes the x and y coordinates of the GPR scanner, and z denotes A-Scan's two way travel time. The z of the reflected electromagnetic pulse signal is analyzed for amplitude strength. These signals are pre-processed through the following steps. 1.) We stack every β A-scan traces then calculate the average to boost the signal-to-noise ratio (SNR). The selection of β traces for calculation considers the balance between the obtainable signal resolution and noise reduction performance. 2.) A signal pulse envelope is extracted through Hilbert Transform which measures the signal power. The Hilbert Transform of the ith A-Scan trace $m(i)$ can be considered as the convolution of $m(i)$ with the function $h(i) = \frac{1}{\pi i}$ which can be expressed as

$$\hat{m}(i) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{m(\tau)}{i-\tau} d\tau, \tag{6.2}$$

where $\hat{m}(i)$ is the direct output of the Hilbert Transform of $m(i)$. The magnitude of the analytical signal $m_a(i)$ equals

$$|m_a(i)| = \sqrt{m(i) + \hat{m}(i)}, \tag{6.3}$$

where $|m_a(i)|$ denotes the envelope of $m(i)$, which facilitates the signal power characterization. 3.) Search for the amplitude pulse φ_t of the reflected A-Scan signal that corresponds to the subsurface object in the envelope signal $|m_a(i)|$ and record the amplitude reward as

$$r_t^{amp} = |m_a(i)|[\text{"}Lookup\text{"}, \varphi_t]. \tag{6.4}$$

6.2.1.2 Reward Based on Scan Cloud

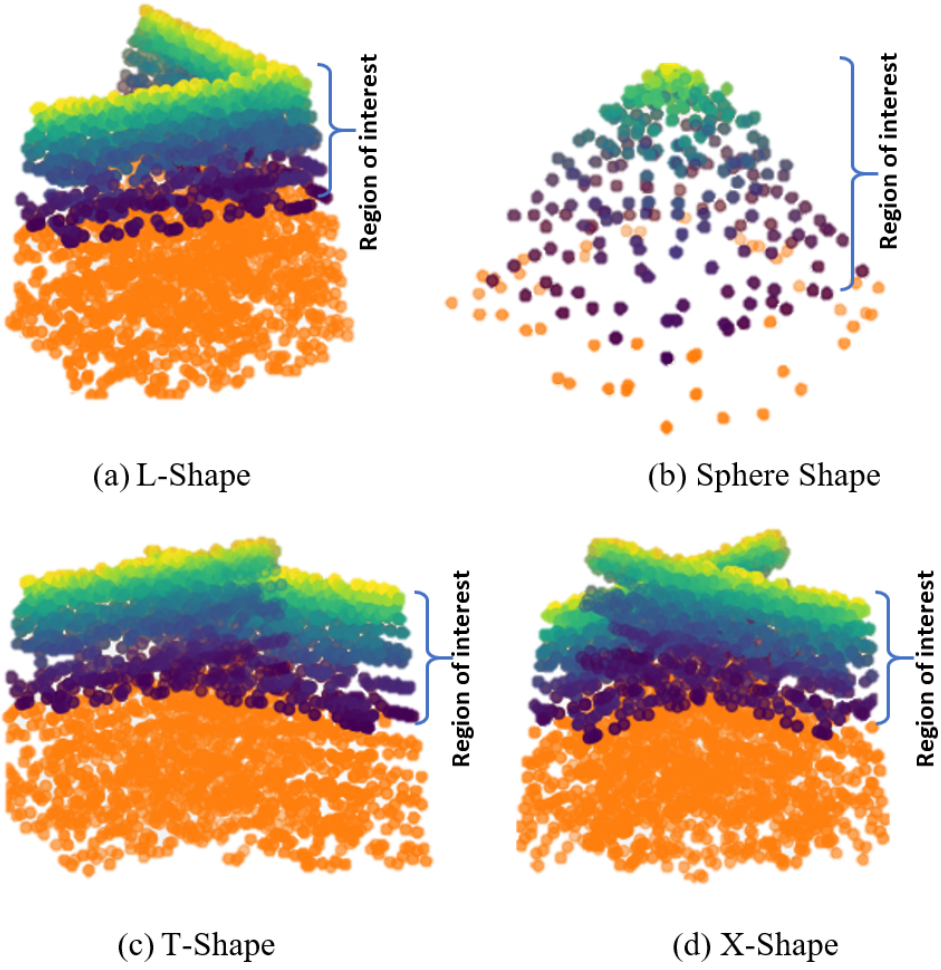
Through the Rényi entropy method, separate computations were conducted with respect to the x, y, z coordinates of the scan cloud points. Rényi entropy is preferred because of its high level of accuracy on signal processing tasks compared to Tsallis [65, 66, 67].

In this work, Rényi entropy is calculated to recognize the singular region on a scan cloud. In particular, a high Rényi entropy value demonstrates a high level of information similarity while a low Rényi entropy value features a high level of information peculiarity [63]. The Rényi entropy computation upon the scan cloud is calculated as

$$E_\alpha(x) = \frac{1}{1-\alpha} \log_e \sum_{f=1}^{F} [z_f(x)]^\alpha, \tag{6.5}$$

where $E_\alpha(x)$ is the entropy quantification with respect to the x axis, $z_f(x)$ is the normalized signal, f is the coordinate index of pulse and F is the total number of x indexes; $\hat{x} = \delta t \cdot \vec{v}$ is the displacement along the trace x axis.

$$E_\alpha(y) = \frac{1}{1-\alpha} \log_e \sum_{g=1}^{G} [z_g(y)]^\alpha, \tag{6.6}$$



(a) L-Shape (b) Sphere Shape

(c) T-Shape (d) X-Shape

Figure 6.2 Region of interest from various underground object shapes

where $E_\alpha(y)$ is the entropy quantification with respect to the y axis, $z_g(y)$ is the normalized signal, g is the coordinate index of pulse and G is the total number of y indexes; $\hat{y} = \delta t \cdot \vec{v}$ is the displacement along the trace y axis.

$$E_\alpha(z) = \frac{1}{1-\alpha} \log_e \sum_{h=1}^{H} [z_h(z)]^\alpha, \tag{6.7}$$

where $E_\alpha(z)$ is the entropy quantification with respect to the z axis, $z_h(z)$ is the normalized signal, h is the trace index, H is the total number of traces included, and z is the time index of pulse data on each reflection A-Scan trace waveform.

The region of interest (RoI) signifying the area where the subsurface object is located, is computed through the 3D OTSU threshold method. The 3D OTSU is our modified version of OTSU, which is suitable for 3D scan cloud data. With two selected entropy thresholds $\mathscr{G}1$ and $\mathscr{G}2$, the scan cloud can be segmented into three classes of non-overlapping regions: singular region, stationary background region, and the transition region in-between. The singular region entropy values are lower than threshold $\mathscr{G}1$, the stationary background region entropy values are higher than $\mathscr{G}2$. While for the transitioning region, its entropy values are between these two thresholds. The 3D OTSU method was applied to all scan cloud points to generate three Rényi entropy plots corresponding to the x, y, z axes, where respective optimum thresholds were computed $\mathscr{G}1_x^*, \mathscr{G}1_y^*, \mathscr{G}1_z^*$. The Rényi entropy points which fall below the intersection between the Rényi entropy and 3D OTSU $\mathscr{G}1_x^*, \mathscr{G}1_y^*, \mathscr{G}1_z^*$ are extracted and the corresponding scan cloud points are selected to form the intermediate first, second and third, parts of the global RoI scan cloud points, which is calculated as

$$roi = \mathscr{X}^* \cap \mathscr{Y}^* \cap \mathscr{Z}^*, \tag{6.8}$$

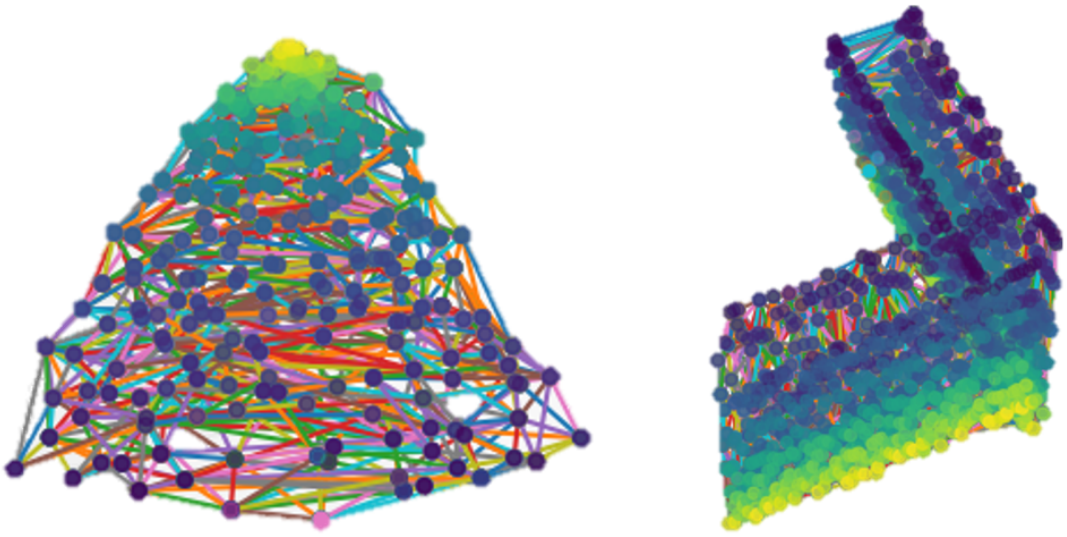
$$\mathscr{X}^* = \{x \mid E_\alpha(x) < \mathscr{G}1_x^*\}, \tag{6.9}$$

$$\mathscr{Y}^* = \{y \mid E_\alpha(y) < \mathscr{G}1_y^*\}, \tag{6.10}$$

$$\mathscr{Z}^* = \{z \mid E_\alpha(z) < \mathscr{G}1_z^*\}, \tag{6.11}$$

where $\mathscr{X}^*, \mathscr{Y}^*$, and $\mathscr{Z}^*$ are sets of intermediate RoIs scan cloud points with respect to x, y, and z axis respectively. All the three intermediate RoIs scan cloud points are finally combined to form the global RoI scan cloud points denoted as *roi*. The global RoI samples are shown in Figure 6.2, where the top part of the scan cloud (none orange scan cloud points) are detected as the RoI. This region is special because it explicitly indicates the region where the subsurface object is located.

From the global RoI scan cloud *roi*, let a, b, c, and d denotes the nodes of its tetrahedrons. We calculate the volume covered by the global RoI scan cloud which is considered as the RoI



(a) Sphere Shape (b) L-Shape

Figure 6.3 The scan cloud formulated using tetrahedral elements

reward, which can be expressed as

$$r_t^{roi} = \frac{\kappa}{1+e^{-V^{roi}}}, \tag{6.12}$$

where V^{roi} indicates the volume of the region-of-interest formulated by tetrahedral elements as shown in Figure 6.3, namely

$$V^{roi} = \sum_{p=1}^{P} \frac{|(a-d).((b-d)\times(c-d))|}{6}. \tag{6.13}$$

In Eq. 6.13, P is the total number of scan cloud tetrahedral elements.

6.2.1.3 Reward Based on Subsurface Object Classification

Region of interest scan cloud from the previous section is transformed into a signature through Gaussian mixture model (GMM) method [76]. For the underlying density model, we use a mixture of Gaussians with Gaussian centers (μk) on a uniform 3D $m \times m \times m$ grid. Such Gaussians induce a Fisher vector [77] that preserves the point set structure: the presence of points in a specific

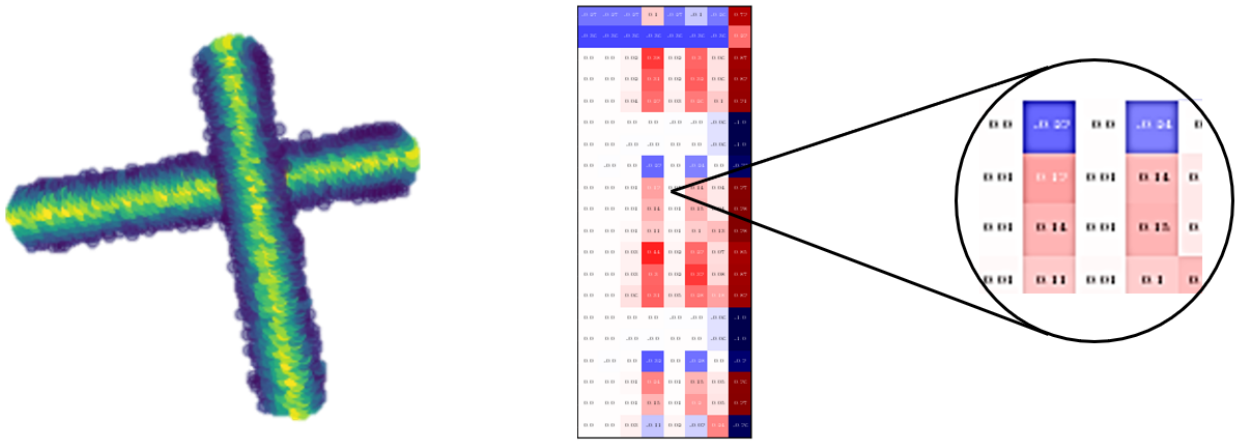


Figure 6.4 X-shaped scan cloud (left) with its corresponding signature (right)

3D location would significantly influence only some, pre-known, Fisher components. The other GMM parameters, weight and covariance, are common to all Gaussians. Figure 6.4 depicts a scan cloud (left) and its corresponding signature representation $(m = 8)$ as a color coded image (right). Each column of the image represents a single Gaussian in an $8 \times 8 \times 8$ Gaussian grid. Zero values are white whereas positive and negative values correspond respectively to the red and blue gradients. Note that the representation lends itself to intuitive interpretation. For example, many columns are white, except for the first two column entries. These correspond to Gaussians that do not have model points near them.

A 4-dimensional data representing the derived signature serves as input to the pre-trained 3D convolutional neural network (CNN) classifier. Subsurface objects can be recognized through different GPR data classification tasks, for example, determining the shape, material type, burial depth, and diameter depending on specific applications. Our scan cloud method is very descriptive in that it not only provides information on object shape, and depth as the work in GPRNet [38], GPRDepthNet [39] did, but also the orientation of the object.

Let $P = \{p(1), p(2), \cdots, P(N)\}$ denotes the classification probability output from the classifier where $p(n)$ is the class probability that the processed scan cloud belongs to class n, and N is the total number of classes. The possible classes depend on the specific classification task. For example, if the classification task is to determine the material type of the subsurface object,

the possible classes could be different object shapes, namely sphere, T shape, L shape, X shape. As entropy is a measure of uncertainty [68, 69], in this work Shannon entropy is considered to quantify the confidence in the classification. The Shannon entropy of the classification probability distribution P can be computed as

$$r_t^{sc} = -\sum_{n=1}^{N} p(n)\log(p(n)), \tag{6.14}$$

It is inferred from Equation (6.14) that a balanced classification probability distribution results in high entropy indicating high uncertainty and low classification confidence while a skewed classification probability distribution has low entropy indicating low uncertainty and high classification confidence.

6.2.2 DDPG Algorithm

In the previous chapter, deep Q-Network (DQN) method was leveraged for subsurface object detection. However, DQNs are meant for problems with a few possible actions, and are therefore not appropriate for continuous action space, like in this case. Nevertheless, a recently proposed Deep RL algorithm referred to as Deep Deterministic Policy Gradient (DDPG) [78] naturally accommodates this kind of problems. It combines the actor-critic classical RL approach [79] with Deterministic Policy Gradient [80].

As used in Deep Q learning, DDPG also uses a replay buffer to sample experience to update neural network parameters. During each trajectory roll-out, we save all the experience tuples (state, action, reward, next_state) and store them in a finite-sized cache a "replay buffer." Then, we sample random mini-batches B of experience from the replay buffer when we update the value and policy networks.

The value network is updated similarly as is done in Q-learning. The updated Q value is obtained by the Bellman equation as shown in Algorithm 4 (Line 27). However the next-state Q values are calculated with the target policy network and target value network. Then, we minimize

the mean-squared loss between the updated Q value and the original Q value:

$$L = \frac{1}{N}\sum_{j}(y_j - Q(s_j, a_j|\theta^Q))^2. \tag{6.15}$$

To calculate the policy loss, we take the derivative of the objective function with respect to the policy parameter. Keep in mind that the actor (policy) function is differentiable, so we have to apply the chain rule. But since we are updating the policy in an off-policy way with batches of experience, we take the mean of the sum of gradients calculated from the mini-batch:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_{j} \nabla_a Q(s, a|\theta^Q)|_{s=s_j, a=\mu(s_j)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|s_j. \tag{6.16}$$

The GPR agent is inclined to execute the action with the highest Q-value derived from each episode. The Q-value corresponding to the pair of state and action represents an expected discounted accumulated future reward.

During learning the Q-function and the policy, the agent needs to perform action selection and execution. The agent faces the well-known exploration-exploitation dilemma of whether to exploit the current knowledge by following the learned policy or to continue to explore the uncertain environment to acquire more knowledge. To resolve the dilemma, exploration is done by adding mean-zero Gaussian noise to the actions at training time. At test time, to see how well the policy exploits what it has learned, we do not add noise to the actions.

As described in Algorithm 4, at first (Lines 1-4), the network parameters are initialized. To generate a more descriptive and interpretive region of interest (RoI) and object classification, an iterative process is implemented from (Lines 11 – 22), where periodical RoI computations and object classifications are conducted, after a substantial amount of scan cloud points have been created, which results in mitigating the computational overhead.

The DDPG employs the amplitude analyzer, RoI and object classifier modules to update the proposed reward function (Lines 16 and 20), by combining the amplitude reward, RoI detection reward and object classification reward, which intuitively measures the amount of the acquired information about the subsurface object from the observed A-Scan pulses. Next (Lines 25 - 14), a

defined amount of experiences are stored into the replay buffer in order for the algorithm to have a stable behavior. Then the minibatch method is used to randomly collect examples from the replay buffer. The weights and biases of the network are updated by training the DDPG according to the loss function Eq.(6.15). The training process will terminate once it reaches a predefined number of episodes. During each episode, the GPR agent stops performing actions after a predefined number of time steps, or could terminate the episode early if it detected the subsurface object. The computational complexity of the algorithm is expressed as $\mathcal{O}(MT)$, where M denotes the total number of episodes and T the number of time steps.

6.3 Performance Evaluation and Discussion

In this section, we systematically evaluate the performance of the proposed method by using gprMax [60, 56]. A-Scan signals are generated by gprMax on the fly in mimicking the GPR data acquisition in a real environment. From these A-Scan signals, a novel scan cloud data format is formulated. Scan cloud is a 3D data format that incorporates derived intuitions, for example, object shape, signal amplitude, object depth, and object diameter.

6.3.1 Experiment Settings

6.3.1.1 GprMax Simulator

The gprMax simulator used in this study solves two dimensional (2D) Maxwell equations using the Finite-Difference Time-Domain (FDTD) method [72]. The simulation in this work considers small diameter objects of various shapes namely sphere, T-shape, L-shape and X-shape as shown in Figure 6.2. The gprMax characterizes the impact of common underground object shapes, which formulates the GPR scan cloud based on the reflected A-Scan signals.

6.3.1.2 Amplitude analysis

Section 6.2.1.1 describes a three step process of how a single A-Scan signal gets processed to produce the coordinates of the A-Scan that form the scan cloud dataset.

6.3.1.3 RoI Detection

The region of interest (RoI) detection module (2^{nd} top right) of Figure 6.1 receives scan cloud points stored in the database and identifies a possible RoI with the method detailed in Section

6.2.1.3. The RoI scan cloud points are extracted based on the optimum 3D OTSU thresholds, in preparation for the classification task.

6.3.1.4 Object Shape Classification

The pre-trained classifier receives the RoI scan cloud and converts it to a signature representation on a grid. The main parts of the network consist of an inception module, maxpooling layers, and finally four fully connected layers. Based on the classification outcome, i.e., the probability values of different classes, a Shannon entropy is calculated. During training, a high class prediction probability means low uncertainty hence high confidence of a particular class. Table 6.1 summarizes the hyperparameters of the 3D CNN-based object classifier.

6.3.1.5 DDPG

Simulations are conducted on a core i7 computer with four cores, 2.2 GHz Intel Xeon CPU, and 16GB RAM. The training process is run with Python 3.6 and tensorflow 1.10.0. The size of the replay buffer is 5×10^4, and the sample mini batch is $B = 32$. During the training process, the GPR agent interacts with the environment and receives tuples of state, action, reward and next state. A total of 5×10^4 such tuples are stored in the replay buffer as experiences which are then sampled and used during the learning. The agent starts by exploring the environment to build knowledge about transitions and action rewards.

6.3.2 Performance Results

6.3.2.1 Object Reconstruction

Based on the results shown in Figure 6.5, our method outperforms other methods on 3D object reconstruction. Depending on the view, other intuitive object features like shape, depth, and diameter can be observed. Our method provides a much better visual intuition of special shapes like spheres, boxes, plates, and other complex underground infrastructure containing multiple objects.

Using DDPG Algorithm 4 the GPR agent learns to take continuous sensing actions around the geographical area to acquire the global view of the underground infrastructure.

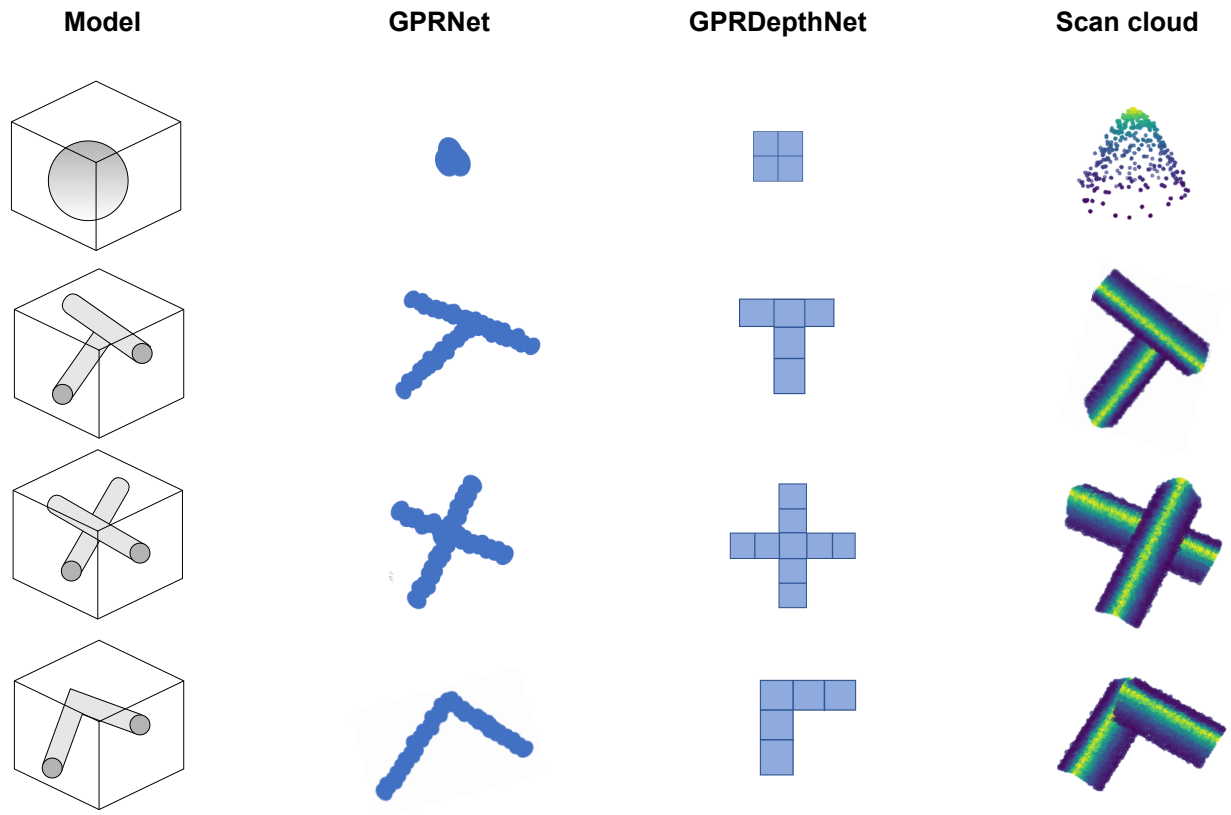


Figure 6.5 The comparison of reconstruction results between other methods and our network

6.3.2.2 Detection Accuracy vs. Heterogeneous Medium

In this thesis, we evaluate the performance of the proposed method under different levels of clutter noise caused by clutter from heterogeneous soil. We considered variations of bulk density and volumetric water of the medium. To make the clutter noise levels more distinct, different types of soil surfaces including smooth, rough, water, and grass surfaces, are added to the fractal-box (a box that houses Peplinski heterogeneous soil).

The visualization in Figure 6.6 detection accuracy vs. bulk density and Figure 6.7 detection accuracy vs. volumetric water, indicates that with a low bulk density and volumetric water a high object detection rate and reconstruction rate is recorded for both setups. This is so because, the higher the bulk density and volumetric water, the higher the dielectric constant, hence, the GPR signal experiences a higher attenuation as it propagates through the medium. The Pearson

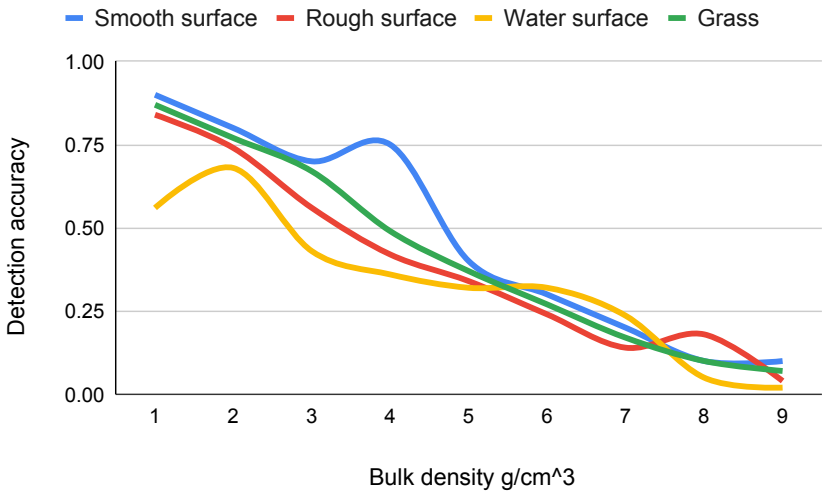


Figure 6.6 Scan cloud performance with varying bulk density

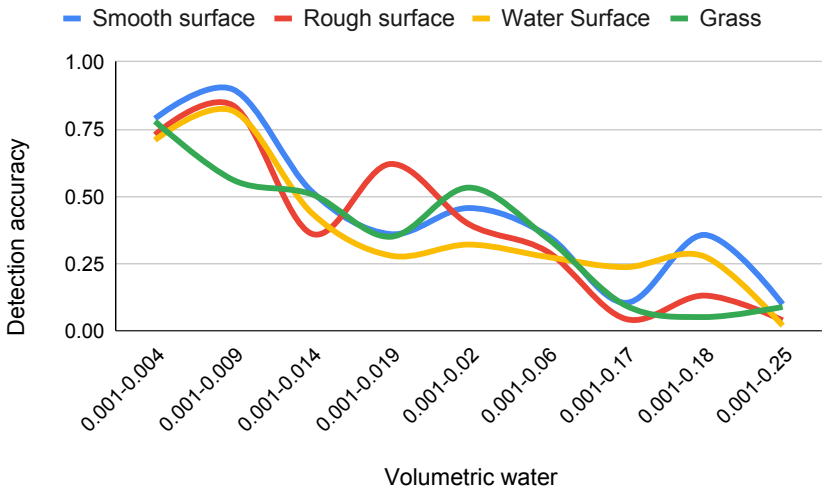


Figure 6.7 Scan cloud performance with varying volumetric water

correlation $r = \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2\sum(y_i-\bar{y})^2}}$ evaluation indicates a very strong negative correlation with an average score of -0.971 between bulk density and detection accuracy, and between volumetric water and detection accuracy, a negative correlation with an average score of -0.6803 .

6.3.2.3 Detection Accuracy vs. Object Size and Burial Depths

To simulate real-world scenarios, the performance of the proposed scan cloud method was also tested through modeling various object sizes and burial depths of sphere and pipes as shown in Figures 6.8 and 6.9, respectively. From these plots, using the Pearson correlation coefficient method, we observed a very strong positive correlation between object size and detection accuracy with an average score of 0.971, and a very strong negative correlation between object depth and detection accuracy with an average score of -0.964.

6.3.2.4 Classification Accuracy vs. Scan Cloud Points

As GPR data interpretation is affected by the density of the scan cloud points, the impact of scan cloud dimension on the classification accuracy of the classifier was also evaluated with various Gaussian grid sizes $m = 11, m = 8, m = 5, m = 4$, and $m = 3$. We found that accuracy increases with large grid size (m) and the number of points representing the scan cloud as shown in Figure 6.10. However, careful considerations must be taken when selecting a high grid size that could introduce some computational overhead.

6.3.3 Convergence Analysis

6.3.3.1 Comparison Between Reward Functions

In order to evaluate the proposed reward function, we study the impact of different reward function variations on performance convergence. In the evaluation, four types of rewards were considered, that is, the reward only from, amplitude analysis, RoI detection, object classification, and the combined rewards from amplitude analysis, RoI detection and Object classification. The rewards were computed from four types of mediums namely; dry sand, wet sand, pavement and heterogeneous soil. Table 6.3 displays the classification accuracy scores of the various reward functions.

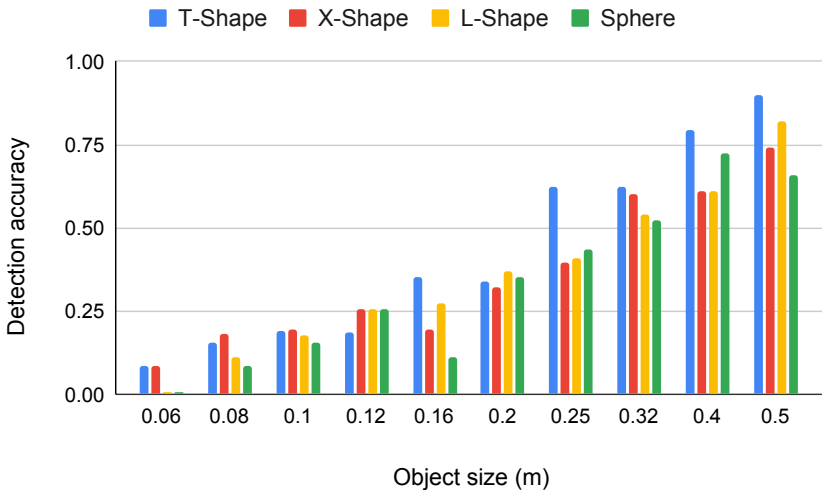


Figure 6.8 Scan cloud performance with varying object sizes

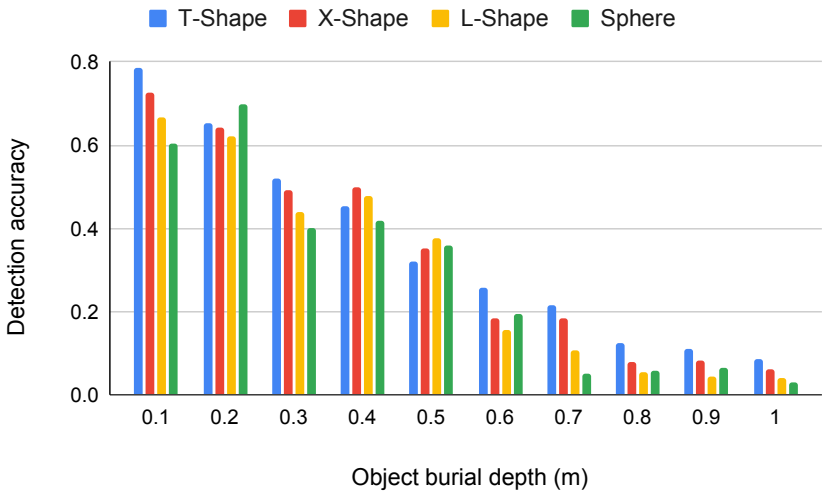


Figure 6.9 Scan cloud performance with varying burial depth

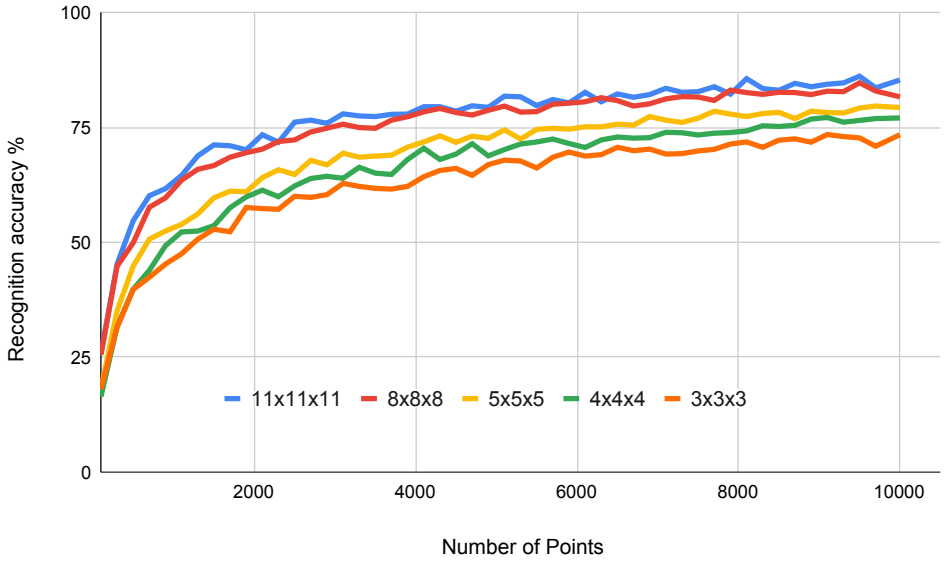


Figure 6.10 The scan cloud recognition accuracy based on various grid size resolutions

6.3.3.2 Time Steps to Reach Convergence

The time needed for the proposed method to reach converged performance was also evaluated. The time steps to reach convergence for different scan cloud architecture implementations adopting different grid sizes (m) are displayed in Table 6.4. From the table, the proposed method configured with a grid size of $(8 \times 8 \times 8)$ outperforms the one with grid size $(11 \times 11 \times 11)$, because a higher grid size introduces computational overhead, despite yielding higher classification accuracy results.

6.4 Summary

In this chapter, a novel GPR data analysis approach termed scan cloud which is focused on the whole in situ GPR dataset rather than on individual A-Scans, B-Scans or C-Scans is developed. The scan cloud and a deep reinforcement learning method called deep deterministic policy gradient (DDPG) are integrated to adapt the operations of GPR system.

A reward function was developed by combining three intermediate rewards stemming from the amplitude analysis module, region of interest module, and shape classifier module. The reward

function is designed such that the GPR agent is highly rewarded by first, receiving an A-Scan with a high amplitude, thereafter form a 3D scan cloud dataset where a region of interest is detected, and lastly a high shape classification score.

The performance robustness of the 3D scan cloud method was explored with different GPR environment configurations namely; dry sand, wet sand, pavement, and heterogeneous soil. To make these configurations more distinct, medium surface conditions were adjusted to include smooth surface, rough surface, water surface, and grass. Also, the bulk density and volumetric water parameters were fine tuned.

To show the efficacy of this method, convergence analysis of different variations of the reward function was conducted. The scan cloud reward function demonstrated superior classification accuracy compared to the rest. Additionally, different grid sizes were tested for convergence speed. A careful trade-off must be considered while selecting a grid size. A bigger grid size produces a high classification accuracy but with a higher computing cost.

6.5 Future Work

Future implementation of this project would be to further streamline scan cloud data format which can represent unique structural conditions, for example, common concrete structure defects which include; air void, delamination and salty moisture. Early and accurate detection of these defects can prevent structural damage and minimize maintenance costs. Other structural anomalies such as water leaks and surface erosion will be considered. Another future work is to develop GPR swarm intelligent model based on advanced reinforcement learning algorithms.

It's no secret that machine-learning models tuned and tweaked to near-perfect performance in the lab often fail in real settings. However, as part of the future work, we will design an additional stage to the training and testing process, in which many machine learning models are produced at once instead of just one. These competing models can then be tested again on specific real-world GPR sensing tasks to select the best one for the job.

Algorithm 4: DDPG algorithm

Randomly initialize critic network $Q(s,a|\theta^{Q})$ and actor $\mu(s|\theta^{\mu})$ with weights θ^{Q} and θ^{μ}
Initialize target network $Q^{'}$ and $\mu^{'}$ with weights $\theta^{Q'} \leftarrow \theta^{Q}, \theta^{\mu'} \leftarrow \theta^{\mu}$
Initialize replay buffer D
Initialize scan cloud buffer Z
for *episode = 1, M* **do**
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial observation state s_1
    for *t=1, T* **do**
        Select action $a_t = \mu(s|\theta^{\mu}) + \mathcal{N}_t$ according to the current policy and exploration noise
        Execute action a_t, capture A-Scan signal
        if $i < I$ **then**
            Compute amplitude value φ_t and amplitude reward r_t^{amp}
            if $r_t^{amp} > h$ **then**
                store scan point data (x_t, y_t, z_t) in Z
            end
            $r_t = \eta r_t^{amp} + C$
        else
            Compute φ_t, r_t^{amp} in lines (12)
            Compute object shape & RoI rewards r_t^{sc}, r_t^{roi}
            $r_t = \eta r_t^{amp} + \rho r_t^{sc} + \beta r_t^{roi}$
            $C = r_t^{sc}$
            $i = 0$
        end
        Set $s_{t+1} = (x_{t+1}, y_{t+1})$
        Store transition (s_t, a_t, r_t, s_{t+1}) in D
        Sample random batch (s_j, a_j, r_j, s_{j+1}) from D
        Set $y_j = r_j + \gamma Q^{'}(s_{j+1}, \mu^{'}(s_{j+1}|\theta^{\mu'})|\theta^{Q'})$
        Update critic by minimizing the loss in Eq. (6.15)
        Update the actor policy using the sampled policy gradient in (Eq. 6.16)
        Update the target networks: $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$ $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
    end
end

Table 6.1 Architecture parameters

Layer (type)	**3D CNN Object Classifier**	**DDPG Actor Network & Critic Network**
Conv3D/2D	Filter: 32 Kernel size: 3 Stride: 1 Padding: Same Activation: LeakyReLU	Filter: 256 Kernel size: 3 Activation: ReLU
MaxPooling	3D Pool size: 2	Pool size: 2
Dropout	Rate: 0.3	Rate: 0.5
Conv3D/2D	Filter: 64 Kernel size: 3 Stride: 2 Padding: Same Activation: LeakyReLU	Filter: 256 Kernel size: 3 Activation: ReLU
MaxPooling	Pool size: 2	Pool size: 2
Fully Connected	2FC 512 and 256 nodes	1FC 1024 nodes
Dense	Classes: 4	Classes: $\mathscr{A}$

Table 6.2 DDPG network settings

Hyperparameter	**Value**	**Description**
Learning rate (α)	0.01	The learning rate used in ADAM to update the DDPG
Discount factor (γ)	0.99	The discount factor used in the reinforcement update
replay buffer size (L)	5×10^4	Number of the most recent experiences stored in the replay buffer
Mini batch size (B)	32	Number of experiences used for training
Epsilon decay	0.99975	Exploration probability decay factor
Episodes (M)	15000	Number of Episodes
Frequency of target network update (H)	5	Number of time steps before the target network is updated

Table 6.3 Classification accuracy of reward function variations

Model	**Dry sand**	**Wet sand**	**Pavement**	**Heterogeneous soil**
Scan cloud-r_t^{amp}	85.1	83.5	84.2	82.8
Scan cloud-r_t^{sc}	90.1	87.9	89.1	83.0
Scan cloud-r_t^{roi}	90.7	88.4	89.5	87.4
Scan cloud-r_t	91.3	89.1	90.6	88.0

Table 6.4 Time steps comparison

Grid Size	**Dry Sand**	**Wet Sand**	**Pavement**	**Heterogeneous Soil**
$3 \times 3 \times 3$	3.25×10^3	4.32×10^3	4.12×10^3	4.25×10^3
$4 \times 4 \times 4$	3.02×10^3	4.25×10^3	4.05×10^3	4.21×10^3
$5 \times 5 \times 5$	3.11×10^3	4.16×10^3	3.84×10^3	4.03×10^3
$8 \times 8 \times 8$	2.61×10^3	3.64×10^3	3.21×10^3	3.56×10^3
$11 \times 11 \times 11$	2.95×10^3	3.98×10^3	3.56×10^3	3.95×10^3

CHAPTER 7

CONCLUSIONS

In this thesis, cognitive GPR-based subsurface sensing based on edge computing and deep reinforcement learning was studied. Novel reward functions were developed such that the GPR agent is rewarded from amplitude analysis, region of interest, and object classification. With the proposed reward functions reinforcement learning (RL) models were developed to enable the GPR to learn to take optimal actions that maximize the long-term discounted reward, hence detecting and identifying subsurface objects from its experiences of interacting with the environment.

This work presented an edge computing (EC) framework for the development of cognitive GPR systems. First, the system architecture of the proposed EC-enable cognitive GPR was developed with different modules residing at either the GPR front end or at the edge server. Then, different computation tasks in a typical perception-action cycle of cognitive GPRs were identified and explained. A computation task scheduling algorithm was designed as the offloading policy to determine whether a computation task should be executed at the local GPR computer or at the remote edge server. Experiments were conducted to demonstrate the efficacy of the proposed system.

An autonomous cognitive GPR (AC-GPR) based on deep reinforcement learning (DRL) approach was proposed. A novel reward function was developed such that the AC-GPR agent is rewarded from both region of interest (RoI) detection and object classification. With the proposed reward function a DQN-based model was developed to enable the AC-GPR to learn to take optimal actions that maximize the long term discounted reward, hence detecting and identifying subsurface objects from its experiences of interacting with the environment. The proposed AC-GPR was evaluated by adopting the GPR operation simulator, gprMax, and simulating real-world environment and GPR operations. Simulation results show the proposed AC-GPR has superior

performance over other GPR systems in terms of object detection success rate, object classification accuracy, and convergence.

The conventional ground penetrating radar (GPR) data analysis methods, which use piecemeal approaches in processing the GPR data formulated in variant formats such as A-Scan, B-Scan, and C-Scan, fail to provide a global view of underground objects on the fly to adapt to the operations of GPR systems in the field. To bridge the gap, in this thesis, a novel GPR data analysis approach termed scan cloud is proposed which is focused on the whole in situ GPR dataset rather than on individual A-Scans, B-Scans or C-Scans. We also study the integration of scan cloud and a deep reinforcement learning method called deep deterministic policy gradient (DDPG) to adapt the operation of GPR system. Simulation results show that our proposed cognitive GPRs outperform other GPR systems in terms of detection accuracy, operating time, and object reconstruction.

REFERENCES

[1] H. M. Jol. *Ground Penetrating Radar Theory and Applications*. Elsevier, 2009. 1, 5

[2] WaveSense. https://wavesense.io, 2021. [Online; accessed 5-June-2021]. 1

[3] A. Turk, A. Hocaoglu, and A. Vertiy. *Subsurface Sensing*. Wiley, 2011. 1, 5, 29, 38

[4] X. Feng, M. Sato, M. Butler, and C. Liu. Subsurface Imaging Using a Handheld GPR MD System. *IEEE Geoscience and Remote Sensing Letter*, 9(4):659–662, July 2012. 5

[5] Madhu Chandra and Tullio Joseph Tanzi. Drone-borne GPR Design: Propagation Issues. *Comptes Rendus Physique*, 19(1):72 – 84, 2018. 5

[6] P. P. Ray. Internet of robotic things: Concept, technologies, and challenges. *IEEE Access*, 4:9489–9500, 2016. 5

[7] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016. 6

[8] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017. 6

[9] Matthew Cornick, Jeffrey Koechling, Byron Stanley, and Beijia Zhang. Localizing ground penetrating radar: A step toward robust autonomous ground vehicle localization. *Journal of Field Robotics*, 33(1):82–102, 2016. 6

[10] R. M. Williams, L. E. Ray, and J. Lever. An autonomous robotic platform for ground penetrating radar surveys. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 3174–3177, July 2012. 6

[11] V. Kafedziski, S. Pecov, and D. Tanevski. Detection and classification of land mines from ground penetrating radar data using faster r-cnn. In *2018 26th Telecommunications Forum (TELFOR)*, pages 1–4, 2018. 6

[12] W. Rice, M. M. Omwenga, D. Wu, and Y. Liang. Enhanced underground object detection with conditional adversarial networks. In *ISSAT International Conference on Data Science & Intelligent Systems*, 2019. 6, 30

[13] Alex Foessel-Bunting, Dimitrios Apostolopoulos, and William L. Whittaker. Radar sensor for an autonomous Antarctic explorer. In Howie M. Choset, Douglas W. Gage, Pushkin Kachroo, Mikhail A. Kourjanski, Marten J. de Vries, Pushkin Kachroo, Mikhail A. Kourjanski, and Marten J. de Vries, editors, *Mobile Robots XIII and Intelligent Transportation Systems*, volume 3525, pages 117 – 124. International Society for Optics and Photonics, SPIE, 1999. 6

[14] T. Le, S. Gibb, N. Pham, H. M. La, L. Falk, and T. Berendsen. Autonomous robotic system using non-destructive evaluation methods for bridge deck inspection. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3672–3677, May 2017. 6

[15] S. Haykin. Cognitive Radar: A Way of the Future. *IEEE Signal Processing Magazine*, pages 30–40, Jan. 2006. 6, 11, 12, 14, 34, 35

[16] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 7, 42

[17] Y. Li, J. Zeng, S. Shan, and X. Chen. Occlusion aware facial expression recognition using cnn with attention mechanism. *IEEE Transactions on Image Processing*, 28(5):2439–2450, 2019. 7, 42

[18] Baris Kayalibay, Grady Jensen, and Patrick van der Smagt. Cnn-based segmentation of medical imaging data. *CoRR*, abs/1701.03056, 2017. 7, 42

[19] M. Chiang and T. Zhang. Fog and IoT: An Overview of Research Opportunities. *IEEE Internet of Things Journal*, 3(6):854–864, Dec. 2016. 7

[20] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, and Q. Yang. Incorporating Intelligence in Fog Computing for Big Data Analysis in Smart Cities. *IEEE Transactions on Industrial Informatics*, 13(5):2140 – 2150, Oct. 2017. 7

[21] S. Mishra, D. Putha, J. Rodrigues, B. Sahoo, and E. Dutkiewicz. Sustainable Service Allocation using Metaheuristic Technique in Fog Server for Industrial Applications. *IEEE Transactions on Industrial Informatics*, PP(99), Jan. 2018. 7

[22] D. Miao, L. Liu, R. Xu, J. Panneerselvam, Y. Wu, and W. Xu. An Efficient Indexing Model for the Fog Layer of Industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, PP(99), Jan. 2018. 7

[23] F. Tseng, M. Tsai, C. Tseng, Y. Yang, C. Liu, and L. Chou. A Lightweight Auto-Scaling Mechanism for Fog Computing in Industrial Applications. *IEEE Transactions on Industrial Informatics*, PP(99), Jan. 2018. 7

[24] Y. Mao, J. Zhang, and K. Letaief. Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE Journal on Selected Areas in Communications*, 34(12), Dec. 2016. 8

[25] J. Liu, Y. Mao, J. Zhang, and K. Letaief. Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, Jul. 2016. 8

[26] M. Chen, B. Liang, and M. Dong. Joint Offloading and Resource Allocation for Computation and Communication in Mobile Cloud with Computing Access Point. In *Proc. IEEE INFOCOM*, May 2017. 8

[27] Takuya Sugimoto and Manabu Gouko. Acquisition of hovering by actual uav using reinforcement learning. In *Proc. The 3rd International Conference on Information Science and Control Engineering (ICISCE)*, July 2016. 9

[28] Wei Xia, Huiyun Li, and Baopu Li. A control strategy of autonomous vehicles based on deep reinforcement learning. In *Proc. The 9th International Symposium on Computational Intelligence and Design (ISCID)*, 2016. 9

[29] Abdulmajid Murad, Frank Alexander Kraemer, Kerstin Bach, and Gavin Taylor. Autonomous management of energy-harvesting iot nodes using deep reinforcement learning. *CoRR*, abs/1905.04181, 2019. 9, 26

[30] J. Xin, H. Zhao, D. Liu, and M. Li. Application of deep reinforcement learning in mobile robot path planning. In *2017 Chinese Automation Congress (CAC)*, pages 7112–7116, Oct 2017. 9, 43

[31] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016. 9, 43

[32] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364, May 2017. 9, 43

[33] H. Li, Q. Zhang, and D. Zhao. Deep reinforcement learning-based automatic exploration for navigation in unknown environment. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2019. 9

[34] D. Zhao, Y. Chen, and L. Lv. Deep reinforcement learning with visual attention for vehicle classification. *IEEE Transactions on Cognitive and Developmental Systems*, 9(4):356–367, 2017. 9

[35] X. Han, H. Liu, F. Sun, and X. Zhang. Active object detection with multistep action prediction using deep q-network. *IEEE Transactions on Industrial Informatics*, 15(6):3723–3731, June 2019. 9, 36

[36] Enlu Lin, Qiong Chen, and Xiaoming Qi. Deep reinforcement learning for imbalanced classification. *CoRR*, abs/1901.01379, 2019. 9

[37] Daochang Liu and Tingting Jiang. Deep reinforcement learning for surgical gesture segmentation and classification. *CoRR*, abs/1806.08089, 2018. 9

[38] Jinglun Feng, Liang Yang, Ejup Hoxha, Stanislav Sotnikov, Diar Sanakov, and Jizhong Xiao. Gpr-based model reconstruction system for underground utilities using gprnet. *CoRR*, abs/2011.02635, 2020. 9, 67

[39] Jinglun Feng, Liang Yang, Haiyan Wang, Yifeng Song, and Jizhong Xiao. Gpr-based subsurface object detection and reconstruction using random motion and depthnet. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7035–7041, 2020. 9, 67

[40] Xiaolong Chen, Jian Li, Shuowen Huang, Hao Cui, Peirong Liu, and Quan Sun. An automatic concrete crack-detection method fusing point clouds and images based on improved otsu's algorithm. *Sensors*, 21(5), 2021. 9

[41] Jb Wolf, S. Discher, L. Masopust, S. Schulz, R. Richter, and J' Döllner. Combined visual exploration of 2d ground radar and 3d point cloud data for road environments. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 231–236, 2018. 10

[42] Weilin Li, Jian Wen, Zhongliang Xiao, and Shengxia Xu. Application of ground-penetrating radar for detecting internal anomalies in tree trunks with irregular contours. *Sensors*, 18(2), 2018. 10

[43] Yunfeng Ge, Huiming Tang, Ding Xia, Liangqing Wang, Binbin Zhao, Jenkins Wholda Teaway, Hongzhi Chen, and Ting Zhou. Automated measurements of discontinuity geometric properties from a 3d-point cloud based on a modified region growing algorithm. *Engineering Geology*, 242:44–54, 2018. 10

[44] Jürgen Döllner. Geospatial artificial intelligence: Potentials of machine learning for 3d point clouds and geospatial digital twins. *Journal of Photogrammetry, Remote Sensing and Geoinformation Science volume*, page 15–24, 2020. 10

[45] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 514–524, 2019. 10

[46] Patiphon Narksri, Eijiro Takeuchi, Yoshiki Ninomiya, Yoichi Morales, Naoki Akai, and Nobuo Kawaguchi. A slope-robust cascaded ground segmentation in 3d point cloud for autonomous vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 497–504, 2018. 10

[47] Jeremiah Roland, Peter D. Way, Connor Firat, Thanh-Nam Doan, and Mina Sartipi. Modeling and predicting vehicle accident occurrence in chattanooga, tennessee. *Accident Analysis Prevention*, 149:105860, 2021. 10

[48] Dalei Wu, Maxwell M. Omwenga, Yu Liang, Li Yang, Dryver Huston, and Tian Xia. A fog computing framework for cognitive portable ground penetrating radars. In *Proc. IEEE ICC*, May 2019. 12, 34

[49] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis. Live service migration in mobile edge clouds. *IEEE Wireless Communications*, PP(99):2–9, 2017. 19

[50] W. Zhou, L. Li, M. Luo, and W. Chou. Rest api design patterns for sdn northbound api. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pages 358–365, May 2014. 19

[51] Lubos Mercl, David Sec, and Vladimir Sobeslav. Analysis of frameworks for building iaas cloud using by cloud computing providers. In Fatos Xhafa, Santi Caballé, and Leonard Barolli, editors, *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 655–663, Cham, 2018. Springer International Publishing. 19

[52] K. Dhamecha and B. Trivedi. SDN issues A survey. *Int. J. Comput. Appl.*, 73(18):30–35, 2013. 20

[53] N. Gude. NOX: Towards an operating system for networks. *ACM SIGCOMM Comput. Commun. Rev*, 38(3):105–110, 2008. 20

[54] S. S. Lee, K.-Y. Li, K. Y. Chan, G.-H. Lai, and Y. C. Chung. Software-based fast failure recovery for resilient openflow networks. In *Reliable Networks Design and Modeling (RNDM), 2015 7th International Workshop, IEEE*, pages 194–200, Munich,Germany, Oct. 2015. 20

[55] M. Goyal, K. K. Ramakrishnan, and Wu chi Feng. Achieving faster failure detection in ospf networks. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 1, pages 296–300 vol.1, May 2003. 20

[56] Craig Warren, Antonios Giannopoulos, and Iraklis Giannakis. gprmax: Open source software to simulate electromagnetic wave propagation for ground penetrating radar. *Computer Physics Communications*, 209:163 – 170, 2016. 26, 33, 47, 59, 70

[57] R. M. Joseph and A. Taflove. Fdtd maxwell's equations models for nonlinear electrodynamics and optics. *IEEE Transactions on Antennas and Propagation*, 45(3):364–374, 3 1997. 26

[58] Mark E. Orazem, Isabelle Frateur, Bernard Tribollet, Vincent Vivier, Sabrina Marcelin, Nadine Pébère, Annette L. Bunge, Erick A. White, Douglas P. Riemer, and Marco Musiani. Dielectric properties of materials showing constant-phase-element (CPE) impedance response. *Journal of The Electrochemical Society*, 160(6):C215–C225, 2013. 26

[59] N. Peplinski, F. Ulaby, and M. Dobson. Dielectric properties of soils in the 0.3-1.3 GHz range. *IEEE Trans. Geoscience and Remote Sensing*, 33(3):803–807, May 1995. 27, 47, 50

[60] Craig Warren, Antonios Giannopoulos, and Iraklis Giannakis. gprmax: Open source software to simulate electromagnetic wave propagation for ground penetrating radar. *Computer Physics Communications*, 209:163 – 170, 2016. 33, 47, 59, 70

[61] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007. 34

[62] Mohamed A. Abd-Elmagid, Aidin Ferdowsi, Harpreet S. Dhillon, and Walid Saad. Deep reinforcement learning for minimizing age-of-information in uav-assisted networks. *CoRR*, abs/1905.02993, 2019. 36

[63] Yu Zhang, Panglijen Candra, Guoan Wang, and Tian Xia. 2-D Entropy and Short-Time Fourier Transform to Leverage GPR Data Analysis Efficiency. *IEEE Transactions on Instrumentation and Measurement*, 64(1):103–111, 2015. 37, 38, 41, 42, 61, 63

[64] N. Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979. 37, 41, 61

[65] Soldovieri Francesco Solimene Raffaele, D'Alterio Antonietta. Entropy-based clutter rejection for intrawall diagnostics. *International Journal of Geophysics*, 2012, 2012. 38, 63

[66] Jonatan Lerga, Nicoletta Saulig, and Vladimir Mozetič. Algorithm based on the short-term rényi entropy and if estimation for noisy eeg signals analysis. *Computers in Biology and Medicine*, 80:1 – 13, 2017. 38, 63

[67] D. J. Cornforth, M. P. Tarvainen, and H. F. Jelinek. Using renyi entropy to detect early cardiac autonomic neuropathy. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5562–5565, 2013. 38, 63

[68] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948. 42, 68

[69] Alireza Namdari and Zhaojun Li. A Review of Entropy Measures for Uncertainty Quantification of Stochastic Processes. *Advances in Mechanical Engineering*, 11(6), 2019. 42, 68

[70] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An introduction*. Cambridge: MIT press, 2018. 44

[71] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396, May 2017. 44

[72] Karl S Kunz and Raymond J Luebbers. *The finite difference time domain method for electromagnetics*. CRC press, 1993. 47, 70

[73] M. M. Omwenga, D. Wu, Y. Liang, L. Yang, D. Huston, and T. Xia. Autonomous cognitive gpr based on edge computing and reinforcement learning. In *2019 IEEE International Conference on Industrial Internet (ICII)*, pages 348–354, 2019. 58

[74] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):4213–4220, Jul. 2019. 58

[75] Chengrun Qiu, Yang Hu, Yan Chen, and Bing Zeng. Deep deterministic policy gradient (ddpg)-based energy harvesting wireless communications. *IEEE Internet of Things Journal*, 6(5):8577–8588, 2019. 58

[76] Jinhua Zhang, Jie Yan, David Infield, Yongqian Liu, and Fue sang Lien. Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and gaussian mixture model. *Applied Energy*, 241:229–244, 2019. 66

[77] Huiping Lin, Hang Chen, Kan Jin, Liang Zeng, and Jian Yang. Ship detection with superpixel-level fisher vector in high-resolution sar images. *IEEE Geoscience and Remote Sensing Letters*, 17(2):247–251, 2020. 66

[78] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. 68

[79] S. P. Singh Y. Mansour et al. R. S. Sutton, D. A. McAllester. Policy gradient methods for reinforcement learning with function approximation, 1999. 68

[80] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China, 22–24 Jun 2014. PMLR. 68

VITA

Maxwell Omwenga was born in 1982 and raised in Nakuru, Kenya. He moved to Uganda in 2000 to pursue advanced secondary level studies, and that started the journey of living and loving Uganda, where he lived for the next 15 years. He graduated from Makerere University in 2006 with a Bachelor of Science in Computer Science and in 2012 with a Master of Science in Data Communication and Software Engineering. After his undergraduate studies, he worked as a web administrator at Makerere University for 7 years. During those years Maxwell doubled as a part time lecturer for 2 years at Bugema University. He traveled to America in 2016 to pursue a Ph.D. in Computer Science at the University of Tennessee at Chattanooga; His doctoral dissertation was conducted under the tutelage of Dr. Dalei Wu and Dr. Liang Yu, which examines the use of reinforcement learning models for autonomous cognitive ground penetrating radar (GPR), computer vision, 3D scan cloud, and edge computing. He won a Sigma Xi Scientific Research Honor (2019) and a research award with UTC Technology Symposium (2019). Maxwell loves jazz music, biking, volleyball, and traveling.