

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Object Detection in GPR Data Using Classical Vision and YOLOv5

Manuel Maria Rocha

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Armando Jorge Sousa

Co-Supervisor: Filipe Neves dos Santos, PhD

July 29, 2022

Supervising Team

Rui Manuel Pereira Coutinho, M.Sc., collaborated with the supervising team throughout the development of this dissertation project.

Resumo

Ground Penetrating Radar (GPR) (Radar Penetrador de Solo) é uma técnica de imagem geofísica não destrutiva e não invasiva, reconhecida pela sua capacidade de preservação ecológica. É uma ferramenta de sensorização remota, utilizada maioritariamente para a caracterização das propriedades elétricas e magnéticas do subsolo, permitindo assim a localização de diversos artefactos enterrados.

Este trabalho apresenta três metodologias, cujo alvo comum é a detecção automática de objetos em *scans* provenientes de GPRs. A primeira metodologia é baseada em técnicas clássicas de visão por computador, tais como o *thresholding* automático de imagem; a transformada de *Hough*; e técnicas de *clustering*, todas concorrentes para a detecção do vértice das reflexões hiperbólicas. A partir do vértice é possível estimar a posição e a profundidade do artefacto enterrado. A segunda metodologia emprega os mais recentes modelos de *deep learning*, nomeadamente as várias arquiteturas da YOLOv5, para detecção e classificação de artefactos hiperbólicos em *scans* de GPR. O cálculo da posição e profundidade dos objectos foi feito através das coordenadas das *bounding boxes*. A terceira metodologia utiliza uma versão adaptada da metodologia clássica para pós-processar as previsões da YOLOv5, desta feita permitindo a eliminação de falsos positivos e uma estimativa da posição e profundidade do objeto baseado no vértice das reflexões hiperbólicas.

Capturamos um pequeno *dataset* em ambiente controlado, usando o GPR *Concretto*, fabricado pela OERAD; o *dataset* (com informação de *ground truth*) foi anotado e disponibilizado à comunidade científica. Devido às limitações de dados, e para gerar reflexões hiperbólicas artificiais, foi treinada uma *Wasserstein Generative Adversarial Network* com *Gradient Penalty* (WSGAN-GP). Foram construídos diversos *datasets* com *scans* GPR sintéticos, utilizando assinaturas hiperbólicas, reais e artificiais, alteradas com técnicas de aumento de *datasets* ao nível da imagem, nomeadamente *flips* e rotações aleatórias, injeção de ruído do tipo *speckle* e alterações do espaço de cores. Juntamente com técnicas de *transfer learning*, os *datasets* sintéticos foram utilizados para treinar os modelos da YOLOv5.

Este projeto de investigação concluiu que a automatização da detecção de objetos em dados GPR é viável, utilizando qualquer uma das metodologias selecionadas. Devido à falta de dados reais, o método baseado em visão clássica destaca-se na *performance* face à metodologia de *deep learning*. Seria expectável que os modelos de inteligência artificial conseguissem atingir *performances* superiores, se uma maior coletânea de *scans* reais fosse usada no treino das redes. Independentemente destes resultados, a combinação de ambas as metodologias foi a melhor das arquiteturas, alcançando uma *Precision* e um *Recall* de 94.4%. Ademais, nos objetos corretamente detetados, os erros de estimação de profundidade e posição foram 0.82 ± 0.54 cm, e 4.46 ± 4.6 cm, respetivamente.

Abstract

Ground Penetrating Radar is a non-destructive and non-invasive geophysical imaging technique, acknowledged for its ecology preservation abilities. It is a remote sensing tool, mostly employed to the characterization of subsurface's electrical and magnetic properties, thus enabling the analysis of the soil and the localization of buried artifacts.

This work proposes three methodologies for the automation of object detection in GPR scans. The first methodology is classic computer vision-based as it employs techniques such as Automatic Global Thresholding, *Hough* Transform and clustering, to detect the apex of hyperbolic reflections. From the apex it is possible to estimate the depth and position of the buried artifact. The second methodology employs state-of-the-art deep learning models of the YOLOv5 architecture to detect hyperbolic artifacts in GPR scans. Their depths and positions are estimated from the bounding boxes' coordinates. The third methodology post-processed YOLOv5's predictions using an adapted version of the classic vision algorithm, that allowed the elimination of false positive bounding box predictions, whilst estimating the depth and position of the objects from the hyperbolas' apex.

A small dataset was gathered in a controlled environment using OERAD's *Concretto*; the dataset (with ground truth information) was labelled and made available to the scientific community. Due to the limited amount of data, a *Wasserstein Generative Adversarial Network with Gradient Penalty* (WSGAN-GP) was used to generate artificial hyperbolic reflections. Datasets of synthetic GPR scans were assembled using real and fake hyperbolic signatures, augmented at image level using random flips and rotations, speckle noise injection and color space transformations. Alongside transfer learning, the synthetic datasets were used to successfully train the YOLOv5 models.

This research project concluded that automating object detection in GPR data is viable using either methodological option. The classic vision methodology slightly outperforms the state-of-the-art deep learning approach, due to the lack of real data. If a larger collection of real GPR scans was available, it was expected a better performance of the AI models. Regardless, the combination of both methodologies is the best of all architectures achieving *Precision* and *Recall* values of 94.4%. Moreover, for the correctly detected objects, the absolute depth and position estimation errors were 0.82 ± 0.54 cm, and 4.46 ± 4.6 cm, respectively.

Agradecimentos

À Faculdade de Engenharia da Universidade do Porto, na figura plural e diversa do seu corpo docente, pelas ferramentas intelectuais disponibilizadas nos cinco anos do curso de Mestrado Integrado em Engenharia Electrotécnica e de Computadores, que com esta dissertação termina, agradeço.

Ao Doutor Filipe Neves dos Santos e ao Mestre Rui Coutinho, aqui no papel de coorientadores, deixo o meu cumprimento.

Não esqueço todos os que, de uma ou outra forma, estiveram presentes neste caminho propedêutico da vida que estou prestes a iniciar, e recordo-os com gratidão sincera.

Chamo os meus avós. Dos que partiram e/ou ainda estão, lembro as aprendizagens e os modelos que consegui absorver, na expectativa de não os defraudar na honra e verticalidade.

À Tânia, pessoa com quem partilho sonhos futuros, criatura que me acompanha nas aventuras e nos desafios, estou devedor pela paciência de alguma ausência a que o foco deste trabalho me obrigou.

Ainda no que à paciência e tolerância toca, os principais lesados foram os meus pais. A ambos reconheço exigência e resiliência, padrões que tento imitar, nem sempre seguro de alcançar sucesso. Motivação, honestidade e empenho são-lhes também característicos. Faço votos de vir a saber dignificá-los no presente, mas sobretudo, na vida!

Ao Prof. Doutor Armando Jorge Sousa, Professor que se tem mostrado atento e disponível, acompanhando o meu crescimento intelectual neste percurso académico desde o 2º ano, aconselhando e ultrapassando largamente o papel de orientador científico desta tese que agora apresento, devo, mas sobretudo, quero, sublinhar sentidamente o meu penhorado Muito Obrigado!

Manuel Maria Rocha

Funding and Financing

This work is financed by the ERDF - European Regional Development Fund, through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme under the Portugal 2020 Partnership Agreement within project Replant, with reference POCI-01-0247-FEDER-046081.

*“Sejamos no íntimo absolutamente diferentes,
embora na aparência vivamos como os demais. ”*

Sêneca

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	2
1.3	Objectives	2
1.4	Contributions	3
1.5	Document Structure	3
2	State of the art & Fundamental Issues	5
2.1	GPR Fundamentals	5
2.1.1	Remote Sensing Principles and Radars	5
2.1.2	Electromagnetism	6
2.1.3	Electromagnetic Wave Properties	7
2.1.3.1	Velocity	7
2.1.3.2	Attenuation	9
2.1.3.3	Non-idealities	9
2.1.4	Hardware	10
2.1.4.1	Display and Interface unit	11
2.1.4.2	Control unit	12
2.1.4.3	Antennas	12
2.1.5	GPR Characteristics	13
2.1.5.1	Central Frequency	13
2.1.5.2	Bandwidth	14
2.1.5.3	Resolution	14
2.1.5.4	Penetration Depth	16
2.2	Object Detection in GPR Data	17
2.2.1	Preprocessing GPR Data	17
2.2.1.1	Filtering	17
2.2.1.1.1	<i>Dewow</i> Filters	17
2.2.1.1.2	Time-zero Correction	17
2.2.1.1.3	Time, Space and Time-space Filtering (Effective Filtering)	18
2.2.1.2	Gain management	19
2.2.1.3	Migration	19
2.2.1.3.1	<i>Hyperbolic Diffraction Summation</i>	20
2.2.1.3.2	<i>Phase-shift Migration</i>	20
2.2.1.4	<i>Kirchoff's Migration</i>	21
2.2.1.5	<i>Frequency-wavenumber Migration</i>	21
2.2.1.6	EM Propagation Velocity Estimation	21

2.2.1.6.1	Common-offset Estimation	22
2.2.1.6.2	Multi-offset Estimation	23
2.2.2	Classic Vision Applied to GPR Data	24
2.2.2.1	Thresholding	24
2.2.2.2	Edge Detection	25
2.2.2.3	<i>Hough</i> Transform	26
2.2.3	Neural Network-based GPR Data Processing	27
2.2.3.1	CNNs Applied to Object Detection	27
2.2.3.1.1	R-CNN	28
2.2.3.1.2	Fast R-CNN	28
2.2.3.1.3	Faster R-CNN	28
2.2.3.1.4	YOLO	28
2.3	Datasets, Transfer Learning and Data Augmentation	29
2.3.1	Cross-Folding	29
2.3.2	Transfer Learning	29
2.3.3	Augmentation at Image Level	29
2.3.4	Generative Adversarial Networks in data augmentation	30
2.4	Summary	30
3	Automatic Object Detection in GPR Data	33
3.1	Experimental Procedure, Equipment and GPR Surveys Dataset	33
3.2	Architecture	36
3.3	Background Removal and Normalization	38
3.4	Classic Vision Techniques to Detect Hyperbolic Shapes in GPR Data	39
3.4.1	Hyperbola Isolation Algorithm	39
3.4.1.1	Automatic Global Thresholding	39
3.4.1.2	Connected-Components Filtering	41
3.4.2	Hyperbola Apex Detection	42
3.4.2.1	<i>Hough</i> Transform	42
3.4.2.2	Line Filtering	45
3.4.2.3	Intersection Point Detection	46
3.4.2.4	Intersection Point Filtering	47
3.5	Data Augmentation	48
3.5.1	Generation of Synthetic Hyperbolic Signatures using a <i>Generative Adversarial Network</i>	50
3.5.2	Generation of Synthetic GPR B-scans	53
3.5.2.1	Background Assembly	53
3.5.2.2	Hyperbola Placing	54
3.6	YOLOv5 architectures applied in Detection of Hyperbolic Shapes in GPR Data	57
3.6.1	Datasets	57
3.6.2	Models	58
3.7	Integration of YOLOv5 Models with Classic Vision Techniques to Detect Hyperbolic Shapes in GPR Data	62
3.8	Conversion from pixel to real-world coordinates	63

4	Results and Discussion	65
4.1	Results	65
4.1.1	Expected Results	65
4.1.2	Metrics	65
4.1.3	Classic Image Processing Techniques to Detect Hyperbolic Shapes in GPR Data	67
4.1.4	YOLOv5 architectures applied in Detection of Hyperbolic Shapes in GPR Data	72
4.1.4.1	Training and Validation	72
4.1.4.1.1	Metrics	72
4.1.4.1.2	Training evaluation	72
4.1.4.1.3	Post-training validation	73
4.1.4.2	Testing	74
4.1.5	Integration of YOLOv5 Models with Classic Vision Techniques to Detect Hyperbolic Shapes in GPR Data	81
4.2	Discussion	87
5	Conclusions and Future Work	91
5.1	Final conclusions	91
5.2	Future Work	92
	References	95

List of Figures

2.1	Adapted image from [1]. Relationship between the wave's propagation velocity and the wave's frequency.	8
2.2	Adapted image from [1]. Relationship between the wave's attenuation and the wave's frequency.	9
2.3	Common GPR hardware architecture.	10
2.4	Comparison between A-scan, B-scan and C-scan (from [2]).	11
2.5	Electromagnetic signals present on GPR applications. Antenna shielding minimizes EM propagation to the air (air losses), blocks most interference signals and minimizes the direct air waves from transmitter to receiver (direct path). Shielding attempts to maximize the electromagnetic energy that travels on the main path.	13
2.6	Electromagnetic interactions between antennas and shields. Dotted lines represent shield generated signals, straight lines represent EM waves generated by the transmitting antenna.	13
2.7	Adapted image from [1]. Visual interpretation of pulse width at half peak amplitude, W	14
2.8	Adapted image from [3]. Understanding vertical resolution in GPR systems.	15
2.9	Adapted image from [3]. Understanding lateral resolution of a GPR systems.	16
2.10	Edge detector masks. a) <i>Sobel's</i> operator [4, 5]. b) <i>Prewitt's</i> operator [4, 5]. c) <i>Robert's</i> operator [4, 5].	26
3.1	Diagram of the conducted GPR surveys. The distance between the GPR and the dry pinewood's surface was negligible (< 1 cm).	34
3.2	Original B-scans, captured on Configurations 1 to 8.	36
3.3	Overview of the system's architecture.	37
3.4	Sequence diagram of the subsystems' intercommunication.	37
3.5	B-scan after the pre-processing steps of background removal and normalization.	39
3.6	Automatic Global Thresholding applied to the example B-scan.	41
3.7	Connected-Components filtering applied to the example B-scan. Different colors represent different labels.	42
3.8	Comparison between <i>Sobel</i> operator and skelotonization, in generating the optimal input for the <i>Hough</i> transform.	43
3.9	Horizontal sliding window visualization.	44
3.10	Candidate lines after <i>Hough</i> Transform and criteria evaluation. Blue lines represent positive-slope lines, red lines represent negative-slope lines.	44
3.11	Candidate lines after the line filtering procedure. Green lines are positive-slope line aggregations, orange lines are negative-slope line aggregations.	46
3.12	Interception of all candidate lines with opposite directions (violet dots).	47
3.13	Intersection point filtering steps.	48

3.14	Hyperbolic artifact types.	50
3.15	Result of point-wise multiplication of binary mask after the Automatic Global Thresholding (Figure 3.6d) with enhanced B-scan after background removal and normalization (Figure 3.5b).	51
3.16	Artifact generation evolution over the training iterations.	53
3.17	Background block examples.	54
3.18	Synthetic background with 190 traces and a maximum two-way travel time of 3.85 ns.	54
3.19	Example of hyperbolic artifact after data augmentation procedures.	56
3.20	Example synthesized B-scan, with and without annotations.	57
3.21	Performance analysis of YOLOv5 models (from [6]).	59
3.22	DenseNet architecture (from [7]). The colored squares represent convolutional layers.	59
3.23	PANet architecture (from [7]). Each P_i block represents an feature map outputted by a dense block.	60
3.24	YOLOv5's basic architecture (from [6]).	60
3.25	Example synthesized B-scan after resize and re-annotation steps. The green boxes with white text represent the original bounding boxes. The red and blue crosses represent the YOLO-format labelling.	61
4.1	Results of the application of the classic image processing algorithm, on all surveyed configurations.	70
4.2	Training and validation results of YOLOv5 Nano trained on dataset 6. The horizontal axis of all plots represents the training epochs.	73
4.3	Post-training validation metrics.	74
4.4	Visualization of the <i>IoU merging</i> technique. Green boxes are the ground-truth and red boxes are predicted bounding boxes (before and after the merging).	75
4.5	Post-processed bounding box predictions with a confidence threshold of 0.228 and a IoU threshold of 0.45. The green boxes represent the ground-truth and the red boxes represent the model's predictions.	77
4.6	Post-processed bounding box predictions with a confidence threshold of 0.6 and an IoU threshold of 0.45. The green boxes represent the ground-truth and the red boxes represent the model's predictions.	80
4.7	Binary masks and cropped B-scans (in pairs) of the bounding boxes, predicted by the YOLOv5 Nano. The blue dots represent the apex predictions.	84
4.8	Apex predictions of the combined methodologies. The green boxes represent the ground-truth, the red boxes represent YOLOv5's predictions and the blue dots represent classic vision's apex predictions.	86
4.9	Example of 'over-perfectionist' bounding box predictions. The model was a YOLOv5 Nano, trained on dataset 2 (6 classes).	89

List of Tables

3.1	Train/Validation dataset split at artifact level.	58
3.2	Characterization of the datasets used in the training YOLOv5 Nano and YOLOv5 Small. Artifacts indicates the used artifacts. R and S define the artifact's origin: real or synthetic. # classes represents the number of classes of the datasets. Train/Val set size is the number of generated images used for training and validating the networks, respectively.	58
3.3	Hyperparameters used to train YOLOv5 networks (Nano and Small).	61
4.1	ground-truth number of objects (N_{GT}) per configuration (Conf.), their depth (d_{GT}) and their absolute position (p_{GT}).	66
4.2	Summarized results of the classic image processing algorithm: false negative (FN), false positive (FP) and true positive (TP) apex predictions, and depth (d_o) and position (p_o) estimates of the true positive predictions of each configuration (Conf.).	71
4.3	Absolute depth error ($\Delta\epsilon_d$) and absolute position error ($\Delta\epsilon_p$) of true positive apex predictions (TP). Conf. is the surveyed configuration.	71
4.4	Summarized results of the YOLOv5 Nano, trained on dataset 6, using a confidence threshold of 0.228 and an IoU threshold of 0.45: false negative (FN), false positive (FP) and true positive (TP) bounding box predictions, and depth (d_o) and position (p_o) estimates of the true positive predictions of each configuration (Conf.). . . .	77
4.5	Summarized results of the YOLOv5 Nano, trained on dataset 6, using a confidence threshold of 0.6 and an IoU threshold of 0.45: false negative (FN), false positive (FP) and true positive (TP) bounding box predictions, and depth (d_o) and position (p_o) estimates of the true positive predictions of each configuration (Conf.). . . .	78
4.6	Absolute depth error ($\Delta\epsilon_d$) and absolute position error ($\Delta\epsilon_p$) of true positive bounding box predictions (TP), of the inferences with a confidence threshold of 0.228. Conf. is the surveyed configuration.	80
4.7	Absolute depth error ($\Delta\epsilon_d$) and absolute position error ($\Delta\epsilon_p$) of true positive bounding box predictions (TP), of the inferences with a confidence threshold of 0.6. Conf. is the surveyed configuration.	81
4.8	Summarized results of the combined methodologies: false negative (FN), false positive (FP) and true positive (TP) apex predictions, and depth (d_o) and position (p_o) estimates of the true positive predictions of each configuration (Conf.). . . .	86
4.9	Absolute depth error ($\Delta\epsilon_d$) and absolute position error ($\Delta\epsilon_p$) of the true positive apex predictions (TP), of the combined methodologies' predictions. Conf. is the surveyed configuration.	87

- 4.10 Performance assessment of all different training rounds and YOLOv5 models. **Dataset** is the training and validation dataset. **Model** is the YOLOv5 architecture used. **FN**, **FP** and **TP** are the total number of false negatives, false positives and true positives in the testing set. **Pr.**, **Rec.** and **F1** are the dataset-model pair overall *Precision*, *Recall* and *F1-score*. 88

Abbreviations and Symbols

ADPHE	Adaptive Double Plateaus Histogram Equalization
AGT	Automatic Global Thresholding
AI	Artificial Intelligence
AP	Average Precision
AUC	Area Under the Curve
CMP	Common Midpoint
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CSP	Cross Stage Partial
EM waves	Electromagnetic Waves
FLOPS	Floating Point Operations per Second
FN	False Negatives
FP	False Positives
GAN	Generative Adversarial Network
GPR	Ground Penetrating Radar
IoU	Intersection over Union
mAP	Mean Average Precision
MVA	Migration Velocity Analysis
R-CNN	Region-based CNN
ROS	Robot Operating System
TP	True Positives
UDP	User Datagram Protocol
WSGAN	Wasserstein Generative Adversarial Network
WSGAN-GP	Wasserstein Generative Adversarial Network with Gradient Penalty
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Context

State-of-the-art technologies are focusing on making the systems more efficient and greener by minimizing their ecological footprint. Robotic systems are being employed in agriculture-related scientific fields in an attempt to improve productivity and management, while avoiding harming the environment.

Every year, in Portugal, many acres of agricultural lands and forests are completely burnt down, as a result of forest fires. Catastrophic events in natural areas are increasing due to bad management and risk-taking policies.

The national project "rePLANt"¹, aims to develop and implement collaborative strategies, i.e., systems that work alongside humans, to add value and minimize hazardous activities in agricultural sites, more specifically, in forest areas. The proposed strategies are mainly composed of digital systems that aid companies and other forest-related entities to make decisions that positively impact the forest's ecosystem. The strategies will help reduce the implementation of risk-taking policies, support and improve forestry area management, and, consequently contribute to decreasing the frequency and impact of forest fires.

To (re)populate healthy forest areas, rePLANt proposes an autonomous tree sapling plantation system. The plantation sites need specific attributes to contribute to healthy and fast tree growth. Portuguese forests are mainly composed of large-scale trees (e.g. pines, eucalyptus, etc.). These types of trees require strong foundations, i.e., a root system of a large wingspan. Root systems require an obstacle-free, nutrient-rich medium to grow in a way that provides strong radial support.

To ensure the system's maximum success rate, prior knowledge of the plantation site is required, i.e., subsurface characterization and mapping. Of all the different techniques and sensors that could fulfill this task, the *Ground Penetrating Radar* (GPR) is likely to be the most promising sensor. Its viability for the proposed task will be assessed throughout this dissertation.

¹<http://replant.pt>, visited July 1, 2022

1.2 Motivation

As previously mentioned, characterization and mapping of the soil's attributes is advisable when developing an autonomous tree sapling plantation system. To achieve optimal results, i.e., the plantations result in healthy full-grown trees, the subsurface's mapping must be adequate. Achieving high precision levels in this type of task is difficult, due to the frequent lack of ground truth verification. In subsurface characterization, ground truth measurements are nearly impossible, as it is impracticable for large areas and would adulterate the survey site and harm the ecosystem.

In forest areas, the different types of hidden artifacts that can pose obstacles to root development and strength are rocks, old roots, and air pockets. Furthermore, the soil's water content is of great importance, as nutrient-rich soils often have high moisture levels.

Such reasons make non-intrusive / remote sensing, that likely use Radio Frequency waves, such as the GPR equipment, one of the most interesting solutions for applications in forest and agriculture.

1.3 Objectives

This dissertation aims to qualify the GPR device as an adequate tool for agriculture and forest automated / robotic work. As such, GPR-based systems are candidates for non-intrusive object detection in subsurface data.

To evaluate the feasibility of object detection in GPR data, the initial datasets should be simple and with easy ground truth availability. Thus, GPR captures were conducted in a controlled environment, where medium and reflective objects were materials with antagonistic characteristics (to induce great signal changes), the object's locations were fully known, and reflective noise sources could be mitigated.

Two different lines of thought were developed, tested, and evaluated on their suitability to process GPR data. The first methodology employed classic image processing techniques, namely morphological operations, filters, *Hough* transforms, etc., to detect hyperbolic reflections (characteristic of GPR data) originating from hidden objects. The second methodology trained and tested a convolutional neural network (CNN), to detect the objects' signatures in GPR data. As the system must operate in real-time, the chosen CNN architecture was the state-of-the-art, YOLOv5.

Neural networks require large quantities of data to be successfully trained, however, only a few real world surveys could be performed in a controlled environment within the time constraints of this work. To contour the data limitations, data augmentation techniques were implemented. A *Wasserstein Generative Adversarial Network* (WSGAN) was trained to generate synthetic data. All synthetic data signatures (real and generated) suffered augmentation steps, before being used to assemble synthetic GPR data.

As a summary, the two main objectives proposed for this dissertation were:

1. **Develop image processing algorithms** to detect hyperbolic reflections in GPR data;

- (a) Using a classic vision approach
- (b) Using state-of-the-art Artificial Intelligence / Neural Networks

2. **Discuss, evaluate and compare** the performance of the developed methodologies.

1.4 Contributions

This dissertation provides several contributions in the field of GPR data processing:

- Extensive literature review on:
 - GPR systems' elementary operation and principles;
 - Standard data processing techniques of GPR data;
- Brief literature review on:
 - Computer vision-based processing techniques applied to GPR data;
 - Object detector CNNs applied to GPR data;
 - Data augmentation techniques
- Development of a classic vision algorithm to detect hyperbolic reflections in GPR data (object detection);
- Training and testing of YOLOv5 models to detect hyperbolic reflections in GPR data (object detection);
- Integration of YOLOv5 models with classic vision methodologies to detect hyperbolic reflections in GPR data (object detection);
- Automatic generation of artificial GPR hyperbolic signatures and GPR B-scans;
- The dataset built during this work was made available to the scientific community. It includes real and synthetic data. Real data includes Ground Truth information and labelling. Synthetic data is also labelled. The dataset is available at http://fe.up.pt/asousa/gpr_datasets.
- Performance assessment and comparison between classic vision and state-of-the-art CNNs in detecting hyperbolic signatures in GPR data.

1.5 Document Structure

This dissertation is structured in the following manner: Chapter 2 presents the literature review on the fundamentals of GPR operation and state-of-the-art methodologies applied to GPR data; Chapter 3 explains, in detail, the developed algorithms applied in object detection in GPR data;

Chapter 4 presents the results of the developed methodologies, and assesses the performance of the developed/implemented architectures to detect hyperbolic signatures in GPR data; Chapter 5 assesses the fulfillment of the dissertation's objectives and proposes some improvements for future developments.

Chapter 2

State of the art & Fundamental Issues

2.1 GPR Fundamentals

The starting point of this dissertation project is the usage of the GPR sensing system. This Section will summarize the fundamental concepts that enable the correct reading and understanding of this dissertation's text.

2.1.1 Remote Sensing Principles and Radars

Ground Penetrating Radar is a non-destructive and non-invasive geophysical imaging technique, acknowledged for its ecology preservation abilities [8], that is mostly utilized to characterize subsurface's electrical and magnetic properties, as well as for the localization and mapping of different buried/hidden artifacts [8, 9, 10, 11].

To gather information, the GPR emits electromagnetic waves (EM waves) towards the surface of the structure to be surveyed [9, 10, 11]. As the EM waves encounter discontinuities on the medium's electromagnetic properties they are reflected/scattered and refracted [9, 10, 11] according to Snell's Law [1]. The reflected/scattered waves return to the surface to be picked up by the GPR, whilst the refracted waves continue the propagation further down into the medium, until all energy is dissipated or all waves are reflected back to the surface [8, 9, 10, 11]. The time interval between emission and reception of an electromagnetic wave gives information on the depth and location of the discontinuity [10, 11].

Most GPR applications employ time domain radars, giving particular emphasis to the *Pulsed Radar* [10, 1]. Often, the control unit of these radars sends short electric pulses to the transmitting antenna, allowing the generation of electromagnetic waves [9, 10, 12], whose base frequency is defined by the antenna model [12]. The reflected/scattered EM waves are sampled by the receiver [10, 12] and are represented in an amplitude versus time graph [12].

Examples of frequency domain radars are the *Swept Frequency-Modulated Continuous-Wave Radar* and the *Stepped Frequency-Modulated Continuous-Wave Radar*. Both radar types emit sequences of electromagnetic waves of different frequencies over a defined bandwidth [10, 12] e.g. in a sequence of 10 EM waves, the first wave has a base-frequency of f_c , whilst the tenth

wave has a base frequency of $f_c + BW$, where BW is the bandwidth. Despite having a better signal-to-noise ratio due to the integration between the multiple echo signals [10], providing controlled operation frequencies, better power efficiency and the collection of complex data that allows the implementation of *Synthetic Aperture Radar* algorithms [12], frequency domain radars are much slower as they require frequency-time conversions to get the EM wave transit time [10]. Furthermore, the frequency-time conversion introduces sidelobes on strong signals which prevent weak reflection signals from being identified [12].

GPR is well known for being a high resolution tool [10, 11], that provides real-time information [11] about the depth, conductivity, permittivity and density of the medium's strata, as well as the accurate location of hidden artifacts [10]. Ground penetrating radar surveys are conducted by moving the transmitter and receiver modules, usually at a fixed distance i.e. fixed offset [1], along the to-be measured surface [10, 1].

To perform depth estimation from GPR data an accurate estimation of the propagation velocity of the EM waves in the medium is required [9, 10, 11]. Medium heterogeneity and stratification makes propagation velocity estimation in GPR surveys one of the most complex tasks [10].

As in all real systems, ground penetrating radar presents non-ideal behaviour. Electrical components, cables and mechanical parts induce delays and transition states¹ that affect the system's response [1]. As a result, some self-generated signals may prevail longer than the transmission period, inducing an effect called ring-down [1]. The ring-down, ideally, should decrease as quickly as possible after the pulse, and always be below the ambient noise level [1]. Ultimately, if these residual signals are of higher value than the EM reflections induced by the medium's discontinuities, then the latter won't be captured by the receiving antenna, producing an effect called extended transmitter blanking [1].

2.1.2 Electromagnetism

Electromagnetism is tightly connected to *Ground Penetrating Radar* systems. Maxwell's relationships describe the physics of electromagnetic waves, based on the interactions between electrical and magnetic concepts (e.g. electric field, magnetic field, charge displacement, etc.) [8, 1, 13, 14]. From these, it is possible to predict/explain the transport, dissipation, storage and generation of electromagnetic waves in any medium [14].

The object/medium's response to electromagnetic excitation are described by the constitutive relationships [8, 1]. These relationships provide a description of the average behaviour of electrons, atoms and molecules when influenced by any type of EM field [1]. For GPR, some important concepts present in the constitutive relationships are:

- electrical conductivity (σ): describes the free charge movement when influenced by an electric field [1]. As conductivity is inversely proportional to resistivity [15], it represents the material's energy dissipation [1];

¹a state that is neither True nor False

- dielectric permittivity (ϵ): quantifies the charge displacement on the material i.e. how easily the material polarizes when under the influence of an electric field [1, 15]. It is a representative measure of the energy storage index [1]. In GPR systems, the permittivity is referred to in relative terms i.e. relative dielectric permittivity² (ϵ_r) [15];
- magnetic permeability (μ): represents the reaction of atoms and molecules to magnetic fields [1]. In simple materials it serves as an energy storage indication, as those materials store energy under the influence of magnetic fields [1].

GPR applications are usually more sensitive to fluctuations in the permittivity (ϵ) and conductivity (σ) and less sensitive to permeability (μ) [1, 15]. GPR applications have better outcomes if the medium is low-loss i.e. if the electrical resistivity is close to zero ($\frac{1}{\sigma} \rightarrow 0$) [1], as the wave-form propagation³ of electromagnetic fields relies on the ratio between the medium's energy loss and energy storage indexes to be small i.e. $\frac{\sigma}{\mu + \epsilon_r} \rightarrow 0$ [1]. In cases where the aforementioned condition is not met, the electromagnetic fields will diffuse, leading to poor measurements [1].

GPR systems work outstandingly well in applications where there is a strong difference between the medium's and the target's electromagnetic properties e.g. detection of metal rebars in concrete walls [11]. In environments with a lot of clay, the GPR's penetrating capability is abruptly downgraded, as the conductivity is substantially higher and where the energy dissipation is much higher [11, 1].

The presence or absence of water in the medium is also determinant on the system's response [11, 1, 15]. Water conductivity varies a lot with salinity: it either can have very high conductivity (salted water), as it can have very low conductivity (distilled water) [15]. Moreover, water has a much higher relative dielectric permittivity than all other materials (~ 80) [1]. In addition, water greatly affects mediums composed of rocks and debris, as it fills the empty spaces (pore spaces), contributing to a greater heterogeneity of the medium's properties [1]. As the odds of finding distilled water in Nature are very low, it is commonly accepted that water-rich medium's are not ideal for GPR systems, due to the high energy dissipation of such mediums [11, 1, 15].

2.1.3 Electromagnetic Wave Properties

As previously stated, GPR signals are electromagnetic waves that propagate through a medium. As any EM wave (in a non-void environment), they have a characteristic propagation velocity and suffer from attenuation as they propagate further in the material [1]. These characteristics will be further explain in the following subsections.

2.1.3.1 Velocity

The electromagnetic wave's propagation velocity is determined by the medium's permittivity (ϵ) and permeability (μ) [9, 1, 16] and by the wave's frequency [1].

²The ratio between the material's dielectric permittivity and the free-space permittivity [15].

³Wave-form propagation of an EM wave guarantees finite propagation velocity and, even with the usual attenuation of the signal as it propels further into the medium, the signal's original shape is kept almost unchanged [1].

Assuming, for simplicity, that the signals emitted by the GPR are strictly sinusoidal, the relation between wave's properties (velocity and attenuation) and frequency is similar to a high-pass filter [1], where diffusive behaviour is expected for low frequencies and wave-form behaviour for high frequencies [1] (see Figure 2.1).

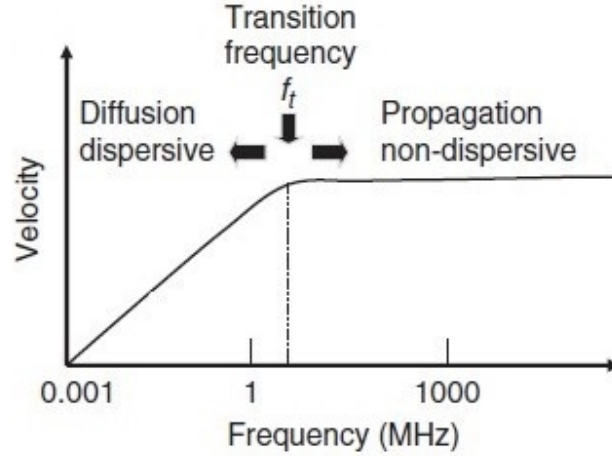


Figure 2.1: Adapted image from [1]. Relationship between the wave's propagation velocity and the wave's frequency.

With that assumption, the transient frequency, separating diffusive- from wave-form propagation, is proportional to the ratio between the medium's conductivity and permittivity [1]. In diffusive state (below the transient frequency), the properties vary proportionally to the wave's frequency [1] (cf. Figure 2.1). In wave-form propagation (above the transient frequency), the properties become frequency independent hence becoming (theoretically) constant⁴ [1] (cf. Figure 2.1).

As previously stated, for GPR systems to be effective, the EM wave propagation must be in wave-form [1]. When under those conditions, the propagation velocity can be estimated by Equation 2.1. If the material is non-magnetic, then $\mu \sim \mu_0$, and the propagation velocity is given by Equation 2.2, where μ_0 is the free-space magnetic permeability, ϵ_0 is the free-space dielectric permittivity, μ_r and ϵ_r are the medium's relative magnetic permeability and relative dielectric permittivity, respectively, and c is the free-space speed of light⁵.

$$v = \frac{1}{\sqrt{\epsilon \cdot \mu}} \quad [1, 15, 16] \quad (2.1)$$

$$v = \frac{1}{\sqrt{\epsilon_0 \epsilon_r \cdot \mu_0}} = \frac{c}{\sqrt{\epsilon_r}} \quad [1, 9, 15, 16, 13] \quad (2.2)$$

⁴Assuming that σ , ϵ and μ are frequency independent as well [1]

⁵The free-space speed of light is given by $c = \frac{1}{\sqrt{\epsilon_0 \cdot \mu_0}}$

The differences in the medium's electromagnetic properties are the key to understanding why two EM waves that travel the same distance present different travel times [9].

2.1.3.2 Attenuation

Electromagnetic signal attenuation dictates how deep into the medium the signal can penetrate until its amplitude drops below values that cannot be interpreted [15]. The attenuation is usually described in function of the medium's "skin depth" i.e. the depth at which the EM wave's amplitude drops below $\frac{1}{e}$ of its original value [15]. The bigger the "skin depth", the smaller the medium's attenuation is, and vice-versa [15]. Conductivity is the material characteristic that most affects the attenuation [1, 15] (see Equation 2.3).

$$\alpha = \sqrt{\frac{\mu}{\epsilon}} \cdot \frac{\sigma}{2} [1] \quad (2.3)$$

Attenuation's behaviour with respect to frequency is very similar to velocity's one (see Figure 2.2). At low frequencies (below the transient frequency), attenuation alongside the wave's frequency [1]. At high frequencies (above the transient frequency), the signal attenuation (theoretically) stabilizes [1].

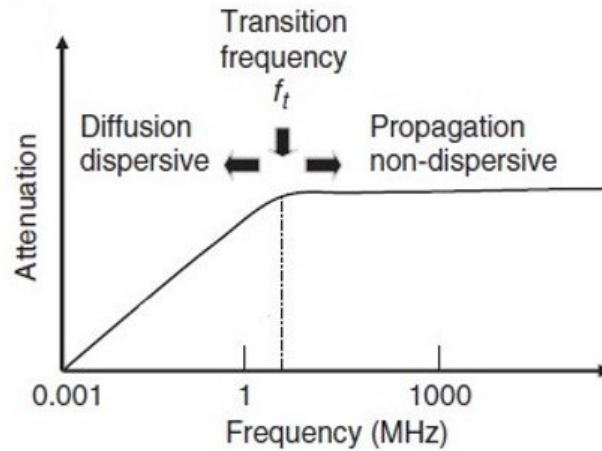


Figure 2.2: Adapted image from [1]. Relationship between the wave's attenuation and the wave's frequency.

2.1.3.3 Non-idealities

Even though on paper the "GPR Plateau"⁶ [1] has constant velocity and attenuation, when analysing real systems these properties vanish [1]. In fact, attenuation and velocity tend to slowly increase with frequency [1].

⁶Illustrative name, applied in GPR systems, to the constant values of velocity and attenuation when the signal's frequency is above the transient frequency [1].

As aforementioned, water plays a very important role in the performance of GPR systems, as it has a huge impact on the material's properties [1, 11, 15]. As frequency increases, water tends to absorb more energy and more strongly [1], as the wave's frequency gets closer to water's relaxation frequency [1]. As the medium absorbs more energy, the signal gets more attenuated.

Attenuation is characterized by two terms: ohmic attenuation (material loss attenuation) and scattering⁷ attenuation [1]. Usually, scattered waves are so small, that their effect is negligible [1]. However, as frequency rises these signals get much stronger (see Equation 2.4), drawing more and more energy from the incident EM wave, resulting in the increase of the signal's attenuation [1].

$$\alpha_s \propto f^4 [1] \quad (2.4)$$

2.1.4 Hardware

To further understand the working principle of GPRs, it is important to list and analyse its main components. A GPR is composed of two antennas, a transmitter module, a receiver module, a control unit and a display/interface/data storage unit [9, 10, 11, 15]. The diagram of Figure 2.3 shows how the components are connected to each other. Electrical connections to the transmitter and receiver modules are done with optical cables to guarantee insulation from electrical/electromagnetic-related problems [15]. Moreover, the antennas are often embedded in their respective module, to further reduce the aforementioned problems [15, 1].

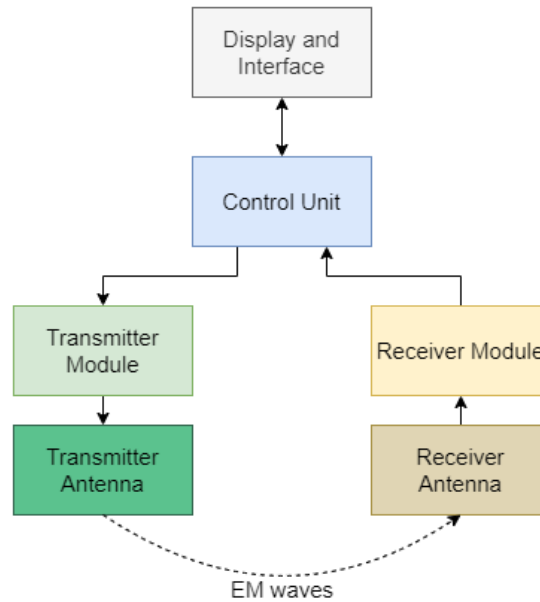


Figure 2.3: Common GPR hardware architecture.

⁷Scattering is the reflection of EM waves by the mediums smaller heterogeneities [1, 15].

2.1.4.1 Display and Interface unit

Data display is of uttermost importance as it is the key to data interpretation [9]. In GPR systems, the display of information can be done in three different ways: A-scans, B-scans or C-scans (see Figure 2.4).

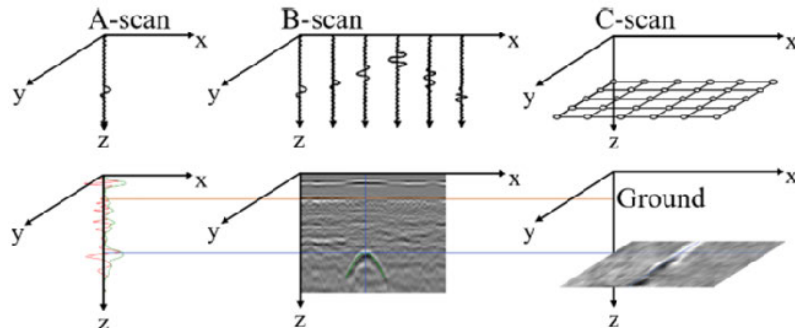


Figure 2.4: Comparison between A-scan, B-scan and C-scan (from [2]).

The A-scan is the representation of a single GPR measurement on an time-amplitude graph [10]. Majorly, the GPR scientific community refers to the individual GPR measurements as traces. A trace is the result of a ninety degree rotation of an A-scan [10].

During GPR surveys, multiple sets of traces are taken. When the survey is a simple line, the traces can be aggregated in a position-time graph known as a B-scan [10, 17]. On such graphs, the signals are highlighted based on their amplitude, using colors or brightness modulation [10]. Finally, by correlating several B-scans (grid survey, parallel survey, etc.) one can obtain a C-scan i.e. a 2D-position-time graph, that presents a three-dimensional view of the subsurface [10]. The user interface allows the operator to configure all survey settings, e.g. hardware/software filter strength and color modulation [9, 15].

Due to the lack of directivity of the transmitting antenna, the electromagnetic waves propagate as ellipses, meaning that the coverage area increases as the waves propagate deeper [8]. As a direct result of this effect, all detected discontinuities, e.g. objects, reflect before and after the antenna is directly above them [8]. The reflection's origin will appear to be directly below the scanning device, whose estimated depth depends on the scanning position with respect to the object's position [8, 10]. As the GPR is moved along the survey site, successive traces are collected and plotted in the B-scan graph [8, 10]. If the reflected object has a symmetric shape and its dimensions are bounded, then an hyperbolic shape will appear on the graph, whose apex corresponds to the real depth of the object [8, 10, 13, 16, 18, 19, 20]. Furthermore, the aperture of said hyperbola depends on the object's dimensions, e.g. a spherical object with diameter D_1 shows a bigger hyperbola aperture than a spherical object with diameter D_2 if $D_1 > D_2$ [13]. However, if the object does not have a symmetric shape or if its shape is too elongated then its reflections are different from an hyperbolic signature [13, 16, 19, 20]. Xei, F. et al. [19] and Forte, E. et al. [16] state that the survey angle poses a critical part on the objects reflections. They concluded that the B-scans of perpendicularly scanned cylindrical objects resulted in hyperbolic shapes but

oblique survey lines resulted in wider, flatter and assymetrical hyperbolas. Moreover, they added that near-parallel or parallel survey lines, provided continuous reflections with similar travel times i.e. a line. Tanikawa, T. et al. [21] added that near parallel survey lines over non-elongated objects resulted in very weak hyperbolic reflections, if any.

2.1.4.2 Control unit

The GPR's control unit is responsible for receiving, recording, sampling and filtering the electrical signals converted by the receiving antenna [8]. Moreover, it generates the short pulses needed for the correct emission of EM waves and sampling times of the receiver antenna [9]. To oversee the measurement process, store data and to establish the connection to the display, a micro-processor with a standard operating system is often integrated in the control unit's electronics [9]. When not present, the GPR system must be connected to a laptop or some other form of storage unit [15]. Filtering the electrical signals is a task that can be performed on hardware (using electronic components) and/or on software [9]. Anyhow the filtering delays should be minimized such that the data is interpretable in real-time [9].

2.1.4.3 Antennas

Antennas are transducers that convert electrical currents into electromagnetic waves and vice-versa. Changes in the current's direction, induce electromagnetic radiation, whose direction depends on the antenna's design [9, 1].

One main characteristics of antenna design is shielding [15, 1]. Shielding's main goal is to enhance/maximize the GPR signals travel to/from the medium and to suppress/minimize all other electromagnetic signals [1, 22], i.e., the direct EM waves between transmitter and receiver, the dispersion of EM waves into the air and the interference signals e.g. from cell-towers [9, 1, 22]. A diagram representing the EM signals present on GPR applications is visible in Figure 2.5.

Despite its multiple advantages, shielding makes the GPR systems bulkier, heavier and more expensive [1]. Moreover, for shielding to be effective against the signals it aims to minimize, the shield must have an electromagnetic response [1]. Consequently, as the transmitter antenna emits EM waves, the shields will partially absorb electromagnetic energy and radiate it [1]. Usually the signals generated by the shields are strong and long, contributing for the increase of the system's residual response, i.e., ring-down [1]. These signals are depicted in Figure 2.6. Often, shielded antennas have the transmitter and the receiver in the same container, reducing the survey versatility of the GPR system they embed [15].

Unshielded antennas are not enclosed, so, in result, there is an aptitude for interference signals, as well as for scanning unwanted data from the environment [15], i.e., the number of captured air reflections is bigger [22]. They are mostly used for low frequency operation [1, 22] where the appropriate antennas are bigger and heavier [22] and the size and weight of the system are of the uttermost importance [1].

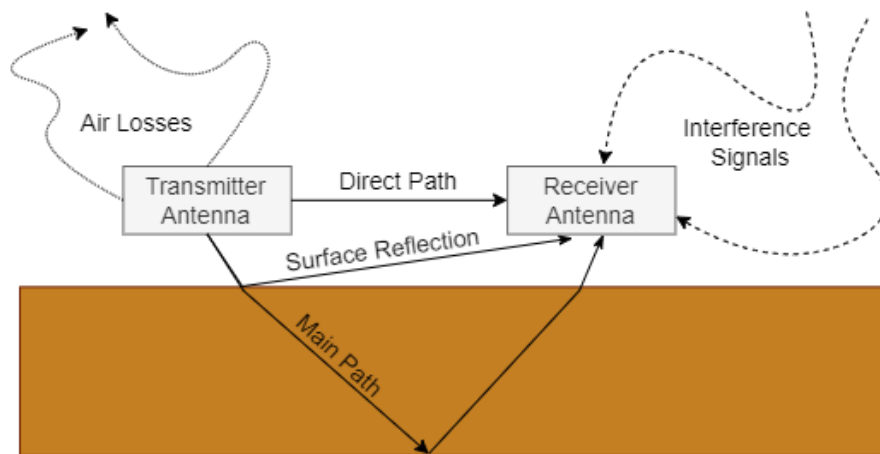


Figure 2.5: Electromagnetic signals present on GPR applications. Antenna shielding minimizes EM propagation to the air (air losses), blocks most interference signals and minimizes the direct air waves from transmitter to receiver (direct path). Shielding attempts to maximize the electromagnetic energy that travels on the main path.

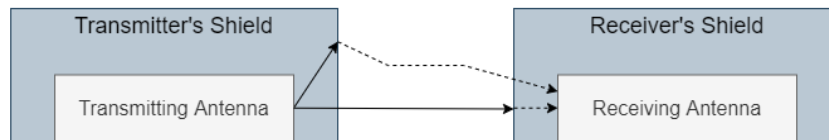


Figure 2.6: Electromagnetic interactions between antennas and shields. Dotted lines represent shield generated signals, straight lines represent EM waves generated by the transmitting antenna.

2.1.5 GPR Characteristics

Ground penetrating radars have specific characteristics, that are determinant on the system's effectiveness. Of those, the most important are: center frequency, bandwidth, resolution and penetration depth.

2.1.5.1 Central Frequency

To guarantee the GPR system's effectiveness, the central frequency should be as close as possible to the transient frequency [1]. The reasoning behind this is the fact that both electrical and scattering losses rapidly increase with frequency, resulting in a smaller penetration depth [1], as it will explained in the following Sections. Generally, GPR systems are characterized by their bandwidth-to-central frequency ratio [1], where the bandwidth should be maximized (resulting in a better resolution, as will be explained in the following Sections) and the center frequency minimized (ensuring an optimal penetration depth) [1].

2.1.5.2 Bandwidth

As stated before, the most common type of GPR signals are short pulses of EM energy [9, 15]. As a result, most GPR signals have large bandwidth [15]. Usually, the bandwidth is set close to the central frequency of the antennas [1, 14], commonly at a scale of 1.5 [15].

Bandwidth is application dependent and is tightly related to GPR's resolution [1]. In normal GPR operations, the antennas are closely positioned, hence when the transmitter emits a pulse, a direct wave travels from transmitter to receiver [1] (see Figure 2.5). If the transmitted pulse is too long, it may overload the receiver antenna's electronics, making it unable to detect the target's reflected EM waves [1]. This effect is called transmitter blanking [1].

In order to ensure optimal resolution, the generated pulses should be as small as possible [10], i.e., the bandwidth should be maximized.

2.1.5.3 Resolution

Resolution indicates the level of certainty when estimating the position, shape, size, thickness, etc. of the buried artifact [1]. It can also be expressed in terms of the smallest detectable subsurface feature [8]. Resolution is split into two separate measures: longitudinal/vertical resolution and lateral/angular resolution [10, 1].

Vertical resolution dictates the minimum distance in depth between two targets, that allows them to be distinguishable [1, 3]. It is widely accepted that for two targets to be distinguishable, their reflected pulses must be separated by at least half of pulse width at half peak amplitude [1, 3] (cf. Figure 2.7).

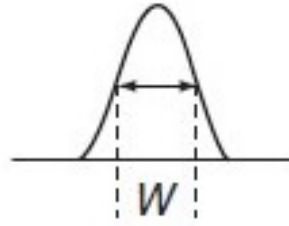


Figure 2.7: Adapted image from [1]. Visual interpretation of pulse width at half peak amplitude, W .

The example from Figure 2.8 (adapted from [3]) assumes that the distance between receiver and transmitter is much smaller than the depth of the target objects. In the example, $T1$ and $T2$ are two similar objects, $d1$ is the distance to target object $T1$ and Δr is the vertical resolution. The medium's propagation velocity is v , therefore the transit times of the EM waves when hitting targets $T1$ and $T2$ are given by $t_1 = \frac{2d1}{v}$ and $t_2 = \frac{d1+2\Delta r}{v}$ [3], respectively. Hence, the time difference between t_1 and t_2 is given by Equation 2.5. The minimum time difference between $T1$ and $T2$ is given Equation 2.6, where W is the pulse width at half peak amplitude. By solving

Equations 2.5 and 2.6, one finds that the vertical resolution is given by Equation 2.9, where B is the pulse bandwidth at half peak amplitude⁸.

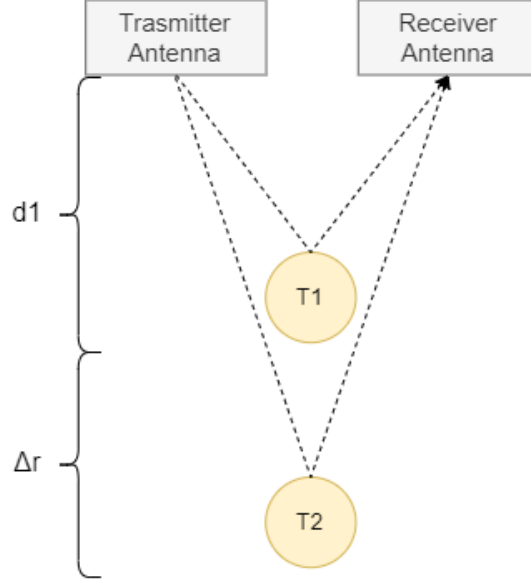


Figure 2.8: Adapted image from [3]. Understanding vertical resolution in GPR systems.

$$\Delta t = t_2 - t_1 = \frac{2\Delta r}{v} [3] \quad (2.5)$$

$$\frac{W}{2} \leq \Delta t [1, 3] \quad (2.6)$$

$$\Delta r \geq \frac{v \cdot W}{4} \Leftrightarrow \Delta r \geq \frac{v}{4B} [1, 3] \quad (2.7)$$

In ideal conditions (Equation 2.9), vertical resolution does not depend on the target's depth, however, in a real scenario that's not the case due to the increase of EM wave attenuation and dispersion as EM waves propagate deeper [1].

Lateral resolution dictates the minimum lateral distance between two objects that allows them to be perfectly distinguishable [3]. The example from Figure 2.9 (adapted from [3]) assumes two similar targets, $T1$ and $T2$, at the same depth, d , in a medium with propagation velocity v . The lateral resolution is represented by Δl . As the proximity between the transmitter and receiver antennas is much smaller than the target's depth, it is assumed that the EM wave that hits target $T1$ has a vertical propagation. The transit time of the EM waves that arrive at $T1$ and $T2$ are $t_1 = \frac{2d}{v}$ and $t_2 = \frac{2\sqrt{d^2 + \Delta l^2} - d}{v}$ [3]. The time difference between $T1$ and $T2$ can be approximated by Equation 2.8.

⁸ $B = \frac{1}{W}$ [1]

$$\Delta t = t_2 - t_1 = \frac{\Delta l^2}{v \cdot d} [3] \quad (2.8)$$

Since the minimum time difference for the lateral velocity must also be half the pulse width at half peak amplitude, it follows Equation 2.6. As such, by solving Equations 2.6 and 2.8, one concludes that the lateral resolution is given by be

$$\Delta l \geq \sqrt{\frac{v \cdot d \cdot W}{2}} \Leftrightarrow \Delta l \geq \sqrt{\frac{v \cdot d}{2B}} [1, 3] \quad (2.9)$$

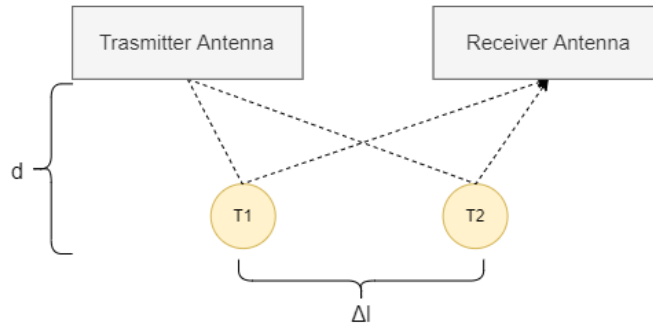


Figure 2.9: Adapted image from [3]. Understanding lateral resolution of a GPR systems.

It is clearly visible, that, unlike vertical resolution, the lateral resolution depends on the target's depth (even in ideal scenarios) [8, 1]. In fact, the increase in coverage area as the waves propagate deeper, results in a bigger lateral resolution value, i.e., a deterioration in the ability to distinguish two laterally close targets [1, 8].

To sum up, the resolution of the GPR system is highly dependant on the medium's electromagnetic properties, as they have a great impact on the EM waves propagation velocity [8, 23]. Moreover, the increase of the system's bandwidth increases the resolution, i.e., the minimum depths and distances that allow the distinction between two targets decrease, making antennas a critical component in GPR systems [9, 10, 11, 23, 24].

2.1.5.4 Penetration Depth

Penetration depth refers to the maximum depth a GPR emitted electromagnetic wave can reach. This characteristic is mostly determined by signal attenuation [9, 23]. As aforementioned, attenuation heavily depends on the medium's electrectromagnetic characteristics, mainly conductivity. As so, in clay-rich or salt-water-rich environments the penetration depth is lower, than in less conductive environments [10, 11, 15, 23]. Moreover, GPR signal's frequency also poses a great influence on the system's penetration depth, as scatter-induced attenuation rapidly increases with frequency (cf. Equation 2.4) [9, 11, 23, 24].

2.2 Object Detection in GPR Data

Quite often, human intervention is required for the correct interpretation of GPR data [25]. The automatization of this process still poses a great challenge. In this Section, different approaches to process and to perform object detection in GPR data are introduced.

Standard GPR data processing is composed of several steps that ensure optimal visualization and interpretation of the subsurface features. The most important procedures are filtering, amplitude gaining, propagation velocity estimation and data migration [26]. These procedures are applied in almost every GPR application [26].

After these initial steps, classic vision methods may be applied to automatically detect hidden objects. AI-based algorithms can also be applied after the aforementioned preprocessing steps. AI algorithms mainly refer to the family of neural networks for image processing, specifically Convolutional Neural Networks. A subsection is dedicated to each of these issues.

2.2.1 Preprocessing GPR Data

2.2.1.1 Filtering

The main objective of data filtering is to remove unwanted signals from the GPR scans, leading to an increase of subsurface features interpretability. Filtering can be summed up into three distinct steps: *dewowing*, time-zero correction and effective filtering [25].

2.2.1.1.1 Dewow Filters

Dewow filtering is an A-scan filter [27] that attempts to remove a very low frequency signal, called *Wow*, that makes GPR traces drift from their real value [1, 25]. It is induced by antenna coupling effects, due to the proximity between transmitter and receiver antennas [1], and by early arrivals through air (e.g. direct waves) or surface reflections (e.g. air-ground reflections), that overload the receiving antenna [25, 27]. *Dewowing* makes the A-scan a zero-mean signal [25, 27] with a symmetric amplitude distribution [27]. These characteristics allow the correct application of spectral processing steps [25, 27], time gaining [13] and a proper positive-negative color filling on the data, without the risk of signal clipping [25]. Nowadays, most GPR systems apply *dewowing* automatically to each recorded trace in real-time or in post-processing [25, 1].

Dewowing has been applied to improve velocity estimation techniques [19].

2.2.1.1.2 Time-zero Correction

Time-zero correction is a technique that assures a single time reference point in the collected GPR data [27]. It is a required processing step as some discontinuities may appear in the collected data due thermal variations (e.g. air temperature fluctuations) [25, 27], electrical instabilities [25], different cable lengths [25, 27] and/or the variation of antenna height while surveying [25, 27]. The reference point can be set using several criteria [25, 13], e.g. using the first direct wave point [25, 13]. If a discontinuity is found along the sequence of reference signals i.e.

a step, an adjustment is performed on the erroneous data [25]. This type of error affects the whole trace, meaning that all information carried by that trace e.g. the depth of objects or stratum boundaries, will be shifted by the same amount, leading to incorrect interpretation of data [25, 27]. If no correction is applied the next processing steps (e.g. time gaining) may be jeopardized [27]. In most systems this correction is performed automatically by software [25]. However, human supervision is often required to ensure consistency among the corrected samples, as noisy data might make the reference points undetectable [25].

Time-zero correction is widely employed. From root-biomass estimation [28] and root quantification [29] to the optimization of EM propagation velocity [19], dune analysis [30], depth estimation of groundwater tables [31], snow water equivalent evaluation [32] and tunnel safety analysis [33].

2.2.1.1.3 Time, Space and Time-space Filtering (Effective Filtering)

Finally, effective filtering attempts to remove or enhance signals of different frequencies across a different number of GPR traces [25]. It is a main method to aid image quality and interpretability [25]. Filtering should occur before amplitude gaining or other more advanced processing techniques, to operate on the "rawest" form of GPR data [25]. Filters can be one-dimensional, spatial or multi-dimensional.

Temporal filters operate in the time domain or in the frequency domain and are applied to each trace individually, i.e., they are applied to a specific time/depth window of the trace [25]. Examples of 1D-time domain filters are simple mean and simple median [25]. Examples of 1D-frequency domain filters are low pass, high pass or band pass filters [25]. One-dimensional filters are the simplest filters and are usually employed to frequency components with higher or lower frequencies than the GPR's bandwidth [25]. Essentially, they are denoising filters [25].

Spatial filters are very similar to temporal filters except that they operate along a window of traces instead of time/depth window [25]. Examples spatial time domain filters are simple running average, average subtraction and background removal⁹ [25]. Examples of spatial wavenumber-domain¹⁰ filters are spacial high, low and band pass filters [25]. Spatial filters are mostly used to remove the direct waves, air-ground reflections and the effects of antenna ringing [25].

Two-dimensional filters operate both in time and space at the same time, i.e., they operate over a number of traces in a specific time/depth window [25]. They are very similar to their one-dimensional versions. Examples of time-space domain filters are time-space average and time-space median [25]. Examples of frequency-wavenumber domain filters can be high, low or bandpass [25]. Frequency-wavenumber filters are mostly used to suppress/enhance dipping

⁹Average subtraction and background removal are very similar to each other. Average subtraction takes the mean of a number of traces in a defined window and subtracts it from each individual trace of the window [25]. Background removal takes the mean of all traces in a window and subtracts it from each individual trace in that window [25].

¹⁰Wavenumber-domain is related to the space-domain, as the frequency domain is related to the time domain [25]

features¹¹ in a specific dip direction [25, 16]. They can also be used to reduce coherent noise patterns¹².

The filtering techniques are extremely application-dependant, as the suppression or enhancement of features depends on the survey's goal, medium's heterogeneity, etc. [25]. Background removal has been applied to the estimation of tree-root biomass [28], detection of roots [35, 21], root quantification [29] and to improve EM wave propagation velocity estimation [19], making it one of the most used filter-types. Running average filter was employed for archaeological surveys [36] and dune analysis [30]; band-pass filters were applied to root biomass estimation [28], root quantification [29], dune analysis [30], depth estimation of groundwater tables [31], snow water equivalent evaluation [32] and to tunnel safety analysis [33]; low-pass filtering was applied on archaeological evaluations [37]; spatial average filters has been applied to improve velocity estimation techniques [19]; frequency-wavenumber filters have been employed in tunnel safety analysis [33] and in groundwater level estimation [38].

2.2.1.2 Gain management

As described in Section 2.1.3.2, EM waves suffer from attenuation as they propagate deeper into the medium, reducing both the system's resolution and penetration depth. The set of algorithms attempt to overcome this effect are called 'gaining'. All gaining methodologies operate similarly, i.e., by multiplying a compensation factor by the traces [25]. The most common gain compensation algorithms are: energy decay compensation, constant gain, linear gain, exponential gain, automatic gain control and some form of user defined gain function [25]. Energy decay compensation automatically overcomes attenuation issues, as it takes advantage of the geometrical spreading effect that the EM waves present while travelling [25]. Constant, linear, exponential and user-defined function gains multiply the received signals by a function (straightforward from the name) [25]. Automatic gain control compensates attenuation trace-by-trace, using a compensation factor dependant on the difference between the mean of the trace, in a specific time/depth window, and its maximum amplitude. It mostly enhances deeper, i.e., weaker events, at the cost of noise increase [25].

Gaining is a widely used technique in GPR processing. Examples of these applications are archaeological prospects [36], root biomass estimation [28], abnormal rock identification [39], root quantification [29], dune analysis [30], groundwater table identification [31], snow water equivalent analysis [32] and tunnel structure evaluation [33].

2.2.1.3 Migration

Migration attempts to remove antenna directionality from the reflection data and reconstruct the real reflective structure of the subsurface, providing a more realistic insight of the underground attributes and their location [17, 1, 25, 18].

¹¹B-scan features that have a trend-slope, e.g. a stratum boundary that gets deeper as the survey line goes further [25].

¹²The opposite of random noise, e.g. hyperbolic features from surface-scattering sources as fences, trees, etc. [34].

As mentioned in Section 2.1.4.1 hyperbolas appear in B-scans due to the conic shape of the EM waves emitted by the GPR's antennas. For migration algorithms to provide optimal results they require:

- the real EM wave propagation velocity structure (velocity per stratum) [25];
- the attenuation should be zero (the medium and objects are lossless and have frequency independent properties [25]);
- that traces are collected with zero antenna offset [25];
- the EM waves to propagate spherically and uniformly in space [25].

The assumptions and/or simplifications performed by migration algorithms, make migration an estimation technique, that approximates the real target's location by re-positioning the reflection/diffraction wavelet [17, 25, 18] and by correcting some distortions induced by the reflector's characteristics [40]. In the ideal situation, the diffracted hyperbolas would be collapsed into a single point and dipping reflections re-positioned with their correct form (slope and position) [25].

Of all the non-idealities, the most important is the constant lateral velocity assumption¹³. Very accurate CMP surveys (with strict intervals and overlapping areas) could be a solution for this problem, however due to the amount of time and equipment such techniques require, they are not a viable option [25]. Hyperbola matching also provides a good velocity estimation, however the estimation is only valid in the area around the hyperbola [25]. One possible workaround for this situation is an iterative migration process, where the velocity estimation is performed on the neighbourhood of each reflected/diffracted feature [25].

The most common migration algorithms are *Hyperbolic Diffraction Summation*, *Phase-shift Migration*, *Kirchhoff's Migration* and *Frequency-Wavenumber Migration*. [17, 25, 18]

2.2.1.3.1 *Hyperbolic Diffraction Summation*

Hyperbolic Diffraction Summation [18, 17] or *Diffraction Stack Migration* [25] is a migration algorithm that finds the hyperbolic functions that best fit the hyperbolic reflections of each buried object [18]. By using an hyperbolic equation on all the pixels that represent the object, it is possible to compute and re-assign energy values to every pixel of the image, and estimate the real object's location [18].

2.2.1.3.2 *Phase-shift Migration*

Phase-shift Migration tries to correct the vertical velocity variations in heterogeneous mediums [17] by applying a phase-shift factor, based on depth, to every point in the B-scan [18]. All processing is performed in the frequency domain. After transforming back into the time domain,

¹³Assuming constant lateral velocity, is to say that the EM waves propagate with the same velocity in the same stratum, disregarding the stratum's heterogeneities [25, 16]

using an Inverse Fast Fourier Transform [17, 18], the energy values of the original samples appear redistributed, providing estimates of the real objects' depths [17, 18].

2.2.1.4 Kirchhoff's Migration

Kirchhoff's Migration is very similar to *Hyperbolic Summation*, but not only does it take into account antenna directivity to correct the reflected traces [18], as it also compensates the EM waves' phase and amplitude by factors proportional to the center frequency of the GPR system [17, 18].

In essence, *Kirchhoff's migration* uses a point-target assumption and, based on geometry, collapses the reflected curves or lines to the objects' real (estimated) location [17, 28].

Kirchhoff's Migration is not suited for real-time applications as it requires the computation of derivatives and integrals [39], making it very computationally demanding. This migration algorithm has been applied to multiple applications like root mapping [41], fine root detection of agricultural crops [35], tree-root biomass estimation [28] and groundwater level estimation [38].

2.2.1.5 Frequency-wavenumber Migration

Frequency-wavenumber Migration or *Stolt's migration* operates solely on the frequency domain [18, 42] and is the fastest of the previous algorithms [18]. On the down side, *Frequency-wavenumber Migration* can only be applied to near-homogeneous mediums [42].

This algorithm has been applied to the detection of wet bodies in controlled environment [43], snow water equivalent estimation [32] and archaeological applications [36].

2.2.1.6 EM Propagation Velocity Estimation

Correctly estimating the propagation velocity of the electromagnetic waves is crucial for any GPR application, as the conversion of transit time¹⁴ to depth depends on the propagation velocity [16, 25] (see Equation 2.10, where t is the transit time, v is the EM wave propagation velocity, and D is the depth). Moreover, velocity estimation greatly contributes to amplitude compensation algorithms [40] and for the effectiveness of some migration algorithms [40], thus contributing to the correct imaging and characterization of the subsurface [40].

$$D = v \cdot \frac{t}{2} \quad [44, 13] \quad (2.10)$$

Velocity estimation can be performed using common-offset or multi-offset techniques [1]. Common-offset estimation techniques gather information from the subsurface using a single transmitter-receiver pair in a fixed geometry (distance and orientation), that samples the subsurface at a constant rate [1]. Multi-offset methodologies survey the subsurface using different offsets between

¹⁴the two-way travel time i.e. time interval transmitter-object plus time interval object-receiver

the GPR's antennas [1]. Even though common-offset techniques are less accurate when estimating the velocity [16], they are the most used, as multi-offset methodologies are more time consuming and more complex [1]. Multi-offset velocity estimation performs substantially better than common-offset when the goal of the GPR survey is to detect soil water content [24].

2.2.1.6.1 Common-offset Estimation

With only common-offset data, velocity estimation can be performed by using different diffraction events recorded on GPR's B-scans [16]. Examples of such methodologies are *Diffraction Hyperbola Fitting* [16, 25, 20], *Migration Velocity Scan* [16] and *Depth-to-known Reflector/Ground Truthing* [25, 20].

Diffraction Hyperbola Fitting [16, 31], *Hyperbolic Matching* [25] or *Hyperbolic Geometry Method* [20] is a software technique that attempts to find the best fitting velocity-dependant hyperbolic function that adjusts to the scanned reflections, by means of simulating different EM propagation velocities [25]. The method's greatest advantage is the fact that it does not require information regarding the medium's dielectric constant nor the depth of the object [20]. The list of disadvantages is large. First, it is extremely human-biased, as evaluating the correctness of the hyperbola's matching depends on a human operator, hence is subject to his/hers professional opinion [20]. Furthermore, this methodology requires the B-scan to present hyperbolic signatures [20]. As stated in Section 2.1.4.1, for an hyperbolic signature to appear in the B-scan, the object must be symmetrical and not elongated w.r.t. the survey direction [19, 16]. If the objects do not follow this criteria, then the reflections may be deformed hyperbolas, making the task of hyperbola fitting exponentially harder [20]. Object size is also very important to hyperbolic fitting, as small and shallow targets often contribute to erroneous estimations [20]. As all estimation methods, when tested on real-world scenarios the results are most likely to be sub-optimal. Some techniques have been proposed to reduce the effects of irregular diffractions [19] that often appear when performing in-cite tests [16]. Basic *Diffraction Hyperbola Fitting* has been employed in archaeological applications [36] and to estimate the depth of underground water tables [31].

Migration Velocity Scan [16] or *Migration Velocity Analysis* [45, 46, 47] (MVA) is similar to *Diffraction Hyperbola Fitting* as both techniques attempt to estimate the EM propagation velocity by trial and error [45]. Their biggest difference is that MVA performs GPR data migration with different velocity values, in search for the best performing value [46, 47]. Due to the application of migration algorithms it is an image domain technique, as opposed to *Diffraction Hyperbola Fitting* which estimates the velocity in the data domain [47]. MVA can be applied with a validation step by means of equivalent synthetic diffractors¹⁵ [32] or by hyperbola energy analysis, i.e., by finding the velocity value that returns maximum trace energy on a specific time-space window [26]. MVA has been employed in snow-water equivalent estimation [32], and it seems to have been used in root detection and mapping [41], despite the authors not explicitly mentioning the MVA approach. An extension of the MVA method has been proposed, in which the velocity

¹⁵Diffractors created by simulation

estimation takes into account the medium's heterogeneities [48] to provide multiple velocity estimations, dependant on the depth of the artifact/boundary, allowing more complete and accurate results [48].

Depth-to-known reflector or *Ground-Truthing* is simple methodology to estimate the medium's velocity. It is one of the most direct and accurate velocity estimation methods [44]. The basic idea behind this method is to bury some reflective material at a known depth, measure the two-way travel time, i.e., the transit time, and from Equation 2.10 estimate the medium's velocity [44]. This experimental procedure requires prior knowledge of the dimensions of the testing area [43]. Often, velocity estimation through *Depth-to-known reflector* is followed by the estimation of the relative dielectric permittivity [44, 43, 35, 49]. Dielectric permittivity can be derived from Equation 2.2, leading to Equation 2.11. This methodology has been applied to several fields of expertise like fine root detection of agricultural crops [35], wet body detection in controlled environments [43], soil depth estimation in forest environments [44, 49] and on archaeological applications [37].

$$\epsilon_r = \left(\frac{c}{v}\right)^2 [43, 16, 49] \quad (2.11)$$

2.2.1.6.2 Multi-offset Estimation

The most common of Multi-offset methods is *Common Midpoint* (CMP). As the name suggests, CMP varies the antenna separation while keeping the same central point, ensuring that the EM waves reach the same reflection point via a different path [1, 38]. It is a widely used method given that once the all the collected profiles are collapsed (summed into a single profile) [50] the resulting signal-to-noise ratio is much higher than when using any common-offset technique [1, 38].

The different captures are highly correlated to each other, as the reflective object is the same [38]. After stacking the different captures, the energy values in the area around the objects are higher, thus distinguishable, from other surveyed areas [38]. On the other hand, stacking different CMP profiles, drastically reduces the signal's noise, as most noise events are produced by random events [38]. Although coherent noise gets enhanced by this type of capture, the application of, e.g., frequency-wavenumber filters before stacking the CMP profiles, mitigates its influence [33, 38].

Multi-offset techniques, in particular, *Common Midpoint*, help transform raw GPR data into more realistic images (because velocity plays a huge role in migration algorithms) [38, 16], and in estimating the medium's dielectric permittivity [38, 1].

CMP-based velocity estimation has been applied to estimate groundwater levels [38]. As water is very distinguishable for its dielectric constant [1, 38], the correct identification of this property, enables the correct estimation of the water-table depth [38]. CMP data has also been used to estimate root biomass [28], where the stacking was performed assuming different velocities, to find the velocity estimate that resulted in the maximum energy [28]. CMP has also been applied in archaeological applications [36] and tunnel stability evaluation [33].

2.2.2 Classic Vision Applied to GPR Data

The advances in technology have led to the automation of several mundane tasks. As seen from the previous Sections, *Ground Penetrating Radar* data processing requires skilled personnel, able to interpret B-scans (or C-scans), to characterize a survey site and/or to locate hidden artifacts. Although several algorithms were developed to make the operator's job easier, they were very flawed. As described in Section 2.2.1.3, migration algorithms help visualize the true location of the object, however they are computationally demanding and/or sensitive to the medium's heterogeneities. As explored in previous Sections, GPR data is characterized by patterns (hyperbolas, dips, etc.), which makes it a great candidate for the application of computer vision algorithms, such as classic image processing techniques.

Segmentation is the process of subdividing an image into its constituent parts [51, 4] and is one of the classic image processing techniques more frequently applied in GPR data. Segmentation can be either region- [51, 52], discontinuity-/edge- [51, 52], threshold- [51, 52] or feature clustering- based [51, 52].

Hough Transform is another technique that is frequently applied in the detection of objects in GPR data. *Hough* Transform provides a new parametric plane, that enables the detection of features like lines, circles or hyperbolas.

2.2.2.1 Thresholding

Thresholding is a very simple, yet powerful technique to transform a grayscale image into a binary image [51, 53, 52], by means of finding the appropriate threshold that separates the interest pixels from all others [51, 53, 52]. This separation can be performed manually (using *a priori* knowledge) or automatically (using image information) [52].

Thresholding can be separated into two different categories: global and local [51, 53, 52]. As the name suggests, global thresholding is when a single threshold is applied to all the pixels in the image [51, 53, 52]. Most global thresholding algorithms analyse the image's histogram to find the threshold that separates the foreground from background [53, 54]. As an example, if an image has an object that occupies a range of gray levels, clearly different from the background gray level range, the image histogram will present two peaks and a deep valley: the optimal threshold is the gray level at the bottom of the valley [53, 54]. Histogram-based algorithms have one major drawback: they do not take into account spatial information, which makes them fail if the valleys are not properly defined [53].

Local thresholding or adaptive thresholding is when a technique computes different threshold values depending on the image region [51, 53]. It usually requires the image to be split into a grid of cells, each with their unique threshold [53, 54]. Several local thresholding algorithms compute the threshold values based on local features and/or region similarities [53, 54].

In recent years, thresholding steps have been frequently employed in GPR data processing pipeline [55, 56, 57, 58, 59], mainly to detect regions of interest where objects are likely to exist.

2.2.2.2 Edge Detection

Edge detection algorithms, as the name suggests, are sets of methodologies that allow the detection of edges in graylevel images [51, 52, 53, 5]. By detecting the boundaries of the objects present in an image, it is possible to perform object segmentation [51, 52]. After the application of edge detection algorithms, the result is a binary image [51].

Edge detection methodologies are defined in two categories: sequential or parallel [53]. Sequential techniques decide whether a pixel belongs to an edge or not, based on the previously examined pixels [53]. Parallel techniques classify pixels as edges or not, based on the pixels' neighbourhood [53].

The three fundamental steps in edge detection algorithms are:

- reduction of noise without destroying the edges, a.k.a. image smoothing [4, 51];
- classifying pixels as edge candidates or noise [4, 51];
- determining the location of the edge [4, 51]. Usually edge thinning and linking are required [4, 51].

Edge detection techniques that rely on the first order derivative of the image are called *Gradient-based Edge Detectors* [4, 5] (which are much more resistant to noise [53]) and the ones that require second order derivatives are *Laplacian-based Edge Detectors* [4, 5] (very sensitive to noise [53]).

Examples of *Gradient-based Edge Detectors* are the *Sobel's* operator, the *Prewitt's* operator and the *Robert's* operator [4, 5]. All operators compute the first derivative of the image by running two different masks over it [4, 5]. These edge detectors have a parallel operation [53]. On *Sobel's* and *Prewitt's* operators, one mask detects the horizontal edges whilst the other detects vertical edges [4, 5]. On *Robert's* operator, one mask detects positive-slope edges whilst the other detects negative-slope [4, 5]. The masks of the aforementioned edge detectors are visible in Figure 2.10.

Finally, once the masks are runned over the image, the magnitude of the gradient in each pixel is computed following Equation 2.12 (where $M(i, j)$ is the gradient magnitude in pixel (i, j) , δ_x and δ_y are the horizontal and vertical derivatives). The resulting magnitude image is thresholded, to obtain the final edged image [5].

$$M(i, j) = \sqrt{\delta_x^2(i, j) + \delta_y^2(i, j)} \quad (2.12)$$

Gradient-based Edge Detectors work best when the objects' edges are rapid [4], i.e., transition grayscale levels fast. To accurately detect smoother edges, the use of *Laplacian-based Edge Detectors* is advisable [4]. This type of detector, not only computes the first derivative, to find values above a threshold [52], but also computes the second derivative of the image and analyses the zero-crossing points [4, 52].

a) Horizontal

-1	-2	-1
0	0	0
1	2	1

Vertical

-1	0	1
-2	0	2
-1	0	1

b) Horizontal

-1	-1	-1
0	0	0
1	1	1

Vertical

-1	0	1
-1	0	1
-1	0	1

c) Positive

-1	0
0	1

Negative

0	-1
1	0

Figure 2.10: Edge detector masks. a) *Sobel's operator* [4, 5]. b) *Prewitt's operator* [4, 5]. c) *Robert's operator* [4, 5].

The most common *Laplacian-based edge detector* is the *Laplacian of Gaussian* [4, 53]. The *Laplacian of Gaussian* is an edge detector that first performs a bi-dimensional Gaussian smoothing to the image (to reduce the noise), and then applies the *Laplacian operator*¹⁶ [53].

One of the most used edge detection algorithms in classical computer vision is the *Canny Edge Detector* [60]. The three main objectives of this detector are: good detection, i.e., low probability of failing to detect an edge point and wrongfully classify noise as an edge point [53] (achieved by maximizing the signal-to-noise ratio of the image [53, 60]); good edge localization [60], i.e., points marked as edges should be as close as possible to the real edge location [53] (achieved by evaluating the *root-mean squared* distance between candidate and true edge [53]); and single response, i.e., only one response to a single edge point [53, 60].

Edge detection is a common intermediate step in vision-based GPR applications [61, 55, 57, 58].

2.2.2.3 Hough Transform

Hough Transform is characterized by the robustness against noise and partial occlusion of features [62]. It has many formulations, the simplest being *Hough Lines* (line detection).

To tackle the fact that there is a infinite number of different lines, defined by slope (m) and y-intercept (b), that pass through a single point [63], *Hough Lines* maps the coordinates (x, y) of every pixel of a binary image from the Cartesian plane to a new plane (ρ, θ) [64, 62].

$$\rho = x \cos \theta + y \sin \theta \quad (2.13)$$

¹⁶Computes the second derivative w.r.t. the y-axis and the x-axis.

In the new plane, lines are represented by their distance to the origin, ρ_i , and by the slope of their normal, θ_i [63]. Each point of the Cartesian plane is represented by a sinusoidal curve in the (ρ, θ) plane [63]. If two points, of the original image, lay on the same line, their sinusoidal curves will intercept in a specific ρ_i, θ_i pair [63, 65]. The more points belong to the same line, the more intersections occur in the ρ_i, θ_i pair [65]. Lines are found by finding the ρ_i, θ_i pairs with more votes [63, 65].

Hough Transforms have some problems. Firstly, their computational cost is high, which may degrade the real-time performance of the systems where it is employed [64]. Secondly, it requires a binary input image, so thresholding or edge detection techniques are often a pre-requisite for its application [64]. Lastly, the classic *Hough* Transform is incapable of determining the start and end points of the detected lines [64].

Through the literature review, it is proven that *Hough* Transform and some variations that aim improve its performance are often employed in classic image processing algorithms to process GPR data [61, 66, 58, 67, 55].

2.2.3 Neural Network-based GPR Data Processing

Following History's course, classic computer vision algorithms were quickly overtaken by the rise of neural networks and their reliability to classify, regress and segment data. In that manner, neural network-based approaches to perform object detection in GPR data have appeared, and quickly became the state-of-the-art in the field.

Neural networks have already been extensively characterized and dissected in literature. A review of the evolution of neural networks, their basic principles and mathematical foundations can be found in [68]. For the purpose of this dissertation it is more important to understand the application of neural networks, more specifically, CNNs in object detection tasks.

Several types of neural networks have been applied to GPR data in attempts to automatize the detection of objects. Although object detectors would be the ideal candidate for that task, most GPR application employ other types of networks, mainly as intermediate steps in the global system. Examples of such applications can be found in [69, 70, 71, 72, 73, 74, 75]. Some object detectors have been employed to the detection of hyperbolic signatures in GPR's B-scans (generic) [55, 56], and specifically to the analysis of GPR data originated from tunnel lining [76].

In the following subsections a brief overview of the evolution of object detector CNNs is presented.

2.2.3.1 CNNs Applied to Object Detection

CNNs are the state-of-the-art in the deep learning field. They provide a hierarchical feature representation from pixel to high-semantical level, which can reveal hidden characteristics of the input data [77]. Moreover, CNNs are traditionally deep networks optimized to achieve great performances in image processing tasks [77].

2.2.3.1.1 R-CNN

One of the first CNNs to be used in object detection was the R-CNN. This model proposed a set of plausible candidate regions (around 2000), called region proposals [77, 78]. Each region was reshaped to a square and was fed into convolutional blocks to get their features extracted [77, 78]. Each set of features would be analysed by classifier network that would classify the regions as background or as an interest object of a specific class [77, 78]. Finally, the bounding box for that region would be regressed and filtered using non-maximum suppression [77, 78].

This model was found impossible to apply in real-time systems as the inference time was around 47 seconds per image [78, 77]. Moreover, the training time was humongous, as for every image, 2000 regions would have to be classified [78, 77]. Finally, the region proposal algorithm was fixed, i.e, it didn't learn to propose better over time [78, 77]. R-CNN is detailed in [79].

2.2.3.1.2 Fast R-CNN

The author of the R-CNN model fixed some of its issues, and came up with the Fast R-CNN. This model, instead of feeding each proposed region to the convolutional network, fed the input image. The region proposal algorithm was then run on the input's feature map [78]. This network was remarkably faster as the image was only convolved once, however it was still bottlenecked by the region proposal algorithm [78]. Its inference time was around 2.3 seconds [78]. Fast R-CNN is thoroughly explained in [80].

2.2.3.1.3 Faster R-CNN

The Faster R-CNN model solved the main issue of its predecessors: the use of selective search to perform the region proposals. This model fed the image through the convolutional layers [78]. The feature maps were then forwarded to a region proposal network, that had trainable parameters to improve the proposal accuracy [78]. Finally, the proposed regions of the features maps were fed to a classifier [78]. This model was so fast, that it could finally be used in real-time applications, achieving inference times around 200 ms [78]. This model is completely explained in [81].

2.2.3.1.4 YOLO

Contrarily, to the aforementioned models, YOLO-like networks look at the input image as a whole, instead of only inspecting regions [78]. YOLO partitions the image into a $N \times N$ grid [78]. Inside each grid cell it predicts m bounding boxes. Then, for each bounding box the class probability is computed [78]. The bounding boxes where the probability of containing a class were bigger would be selected. The YOLO network was much faster than all other models, achieving an inference speed around 22 ms per image [78]. The biggest drawback of YOLO-like networks was the detection of small objects [78]. The original YOLO algorithm can be found in [82].

YOLO algorithms have continued their developments and improvements over the years. In the present time, the most recent versions of YOLO are the version 4 (YOLOv4) and the version 5 (YOLOv5). As will be explained in the next Chapter, this dissertation exploits the capabilities of the YOLOv5 model to perform object detection in GPR data.

2.3 Datasets, Transfer Learning and Data Augmentation

With small amounts of data is impossible to successfully train a neural network, as the network would not be able to generalize, i.e., it would overfit.

A common approach is to use external datasets, but no suitable dataset with labelled data was found; additionally, the work team valued the application of a real GPR to perform local tests, as it would give a more complete insight on the equipment's working principles. As such, a small real-world dataset with the existing equipment was collected.

In general terms, to successfully train neural networks on small datasets, two approaches are adequate: transfer the weights from pre-trained networks and/or augment the real dataset (with automatic changes to the data).

2.3.1 Cross-Folding

Generally, neural networks should be trained with a portion of the dataset, then validated to ascertain overfitting and, finally, the resulting network should be tested to really assess the final performance [83]. Some authors state that the final test should be done with real (non augmented) data [83].

2.3.2 Transfer Learning

One way to avoid the model's overfitting is by using transfer learning. This technique consists of training the desired model on a different, bigger dataset, and then using those weights as the starting point for the new task [84]. Transfer learning is effective because most datasets share low-level spatial features, and the learning process of these features is facilitated on large datasets [84].

Regardless, there are situations where transfer learning is not enough. An example of this is when the available dataset is extremely short (e.g. 10 different images), where transfer learning would not induce much change. In these situations, the use of data augmentation techniques is the state-of-the-art.

2.3.3 Augmentation at Image Level

The easiest and most frequent type of data augmentation techniques are transformations applied directly to the dataset's images. Examples of these transformations are listed below:

- Flipping: horizontal or vertical axis flip of the image [84];
- Rotation: right or left rotation of the image. The angle of rotation must be bounded to ensure that the objects are still recognizable [84], e.g. if an hyperbolic signature collected from B-scans is over-rotated the aperture would become positive, making it an erroneous image;

- Translation: vertical or horizontal translation of the image. Used to avoid positional bias in the data, e.g. if all important details are centered during training, the testing required centered images as well [84];
- Noise injection: adding a matrix of random values (usually Gaussian) to the original image, resulting in a noisy image [84];
- Color space transformations: these consist of changing the image at pixel level, by changing the pixel's values to increase contrast, brightness, sharpness, etc. [84].

All of these techniques are very easy to implement and have a low cost-value ratio, hence their frequent application, even when training deep learning models in the large datasets (e.g. ImageNet [85], COCO [86], etc.).

2.3.4 Generative Adversarial Networks in data augmentation

Generative Adversarial Networks (GANs) are a type of CNN that attempt to create artificial instances of a dataset, such that the differences between artificial and real images are almost imperceptible [84]. These networks are being heavily employed in data augmentation due to their remarkable results in approximating real images and the enhanced performance of DL models trained on GAN-generated data [84].

GANs train based on two distinct networks: the generator and the discriminator. The generator learns to generate a real image from some form of input, e.g., a Gaussian noise distribution, such that the discriminator cannot tell the difference between real and fake images. It goes without saying that the discriminator's learns how to correctly distinguish the two types of images [84]. GANs are the state-of-the-art in data augmentation and dataset generation, and are in the focus of the research community [84].

The problem of data collection in GPR-based systems is very common. Many GPR applications currently employ different types of GAN architectures to create synthetic B-scans. For example, Yue, Y. et. al [87] used an improved *Least-Squares* GAN to generate high-precision GPR B-scans. Quin H. et. al [76] used a *Deep Convolutional* GAN to generate synthetic B-scans of tunnel surveys. Finally, *Wasserstein* GANs have been employed to augment the training datasets of networks to denoise GPR B-scan [70] and to perform automatic object recognition in B-scans [71].

2.4 Summary

This Chapter presented the fundamental issues and strategies for the successful employment of GPR-based systems and the associated data processing techniques, that will enable accurate object detection. It provided extensive information on the characterization of GPR's working principle and the electromagnetism and EM wave concepts associated with it. It also provided a detailed

description of the most important characteristics of GPR systems. Furthermore, the standard pre-processing steps of GPR data is also extensively approached.

Moreover, it summarized classic vision methodologies, traditionally used to perform object detection, such as thresholding, edge detection and *Hough* transforms. It also provided a summary of the evolution of neural networks used to replace the classic systems in the object detection task. Finally, it described different data augmentation techniques from classic image level transformations, to state-of-the-art *Generative Adversarial Networks*.

Chapter 3

Automatic Object Detection in GPR Data

In this Chapter, three different algorithms to perform object detection in GPR data are presented. The first algorithm implements classic computer vision techniques (based on Section 2.2.2) to detect the apex of hyperbolic signatures in B-scans. The second algorithm is based on deep learning, and implements a YOLO architecture (as in Section 2.2.3.1), particularly from YOLOv5's family, to perform object detection in GPR data. Finally, the third algorithm combines the YOLO architecture and an adapted version of the classic vision methodologies to detect hyperbolic signatures (object detection) and estimate their depth and position (apex estimation).

As the GPR data available was highly limited, some data augmentation techniques were implemented. Various classic (image level) data augmentation methods (based on Section 2.3.3) were combined to create synthetic B-scans. Furthermore, a *Wasserstein Generative Adversarial Network with Gradient Penalty* (WSGAN-GP) was trained, on real data, to generate synthetic hyperbolic reflections.

3.1 Experimental Procedure, Equipment and GPR Surveys Dataset

The *Ground Penetrating Radar* used to collect the data for this dissertation was OERAD's *Concretto*¹. This equipment operated at a center frequency of 2 GHz with a vertical and horizontal resolution of 0.001 meters (1 millimeter). Moreover, it presented a variable maximum capture time window² of 7.5 or 15 nanoseconds with 585 samples per trace.

The surveys were conducted in a controlled environment where the reflective objects' location was fully known. The propagation medium was a combination of dry pinewood and air (although, for calculus simplicity, it was considered only dry pinewood). The utilized objects were made of aluminium and iron. The divergence in the electromagnetic properties of medium and objects

¹All information was retrieved from OERAD's official website <https://www.oerad.eu/concretto>

²The maximum capture time window defines the maximum two-way travel time of the emitted EM waves.

made the reflections stand out from the background. The diagrams of the surveyed configurations are available in Figure 3.1. The survey direction, in all configurations, was from left to right.

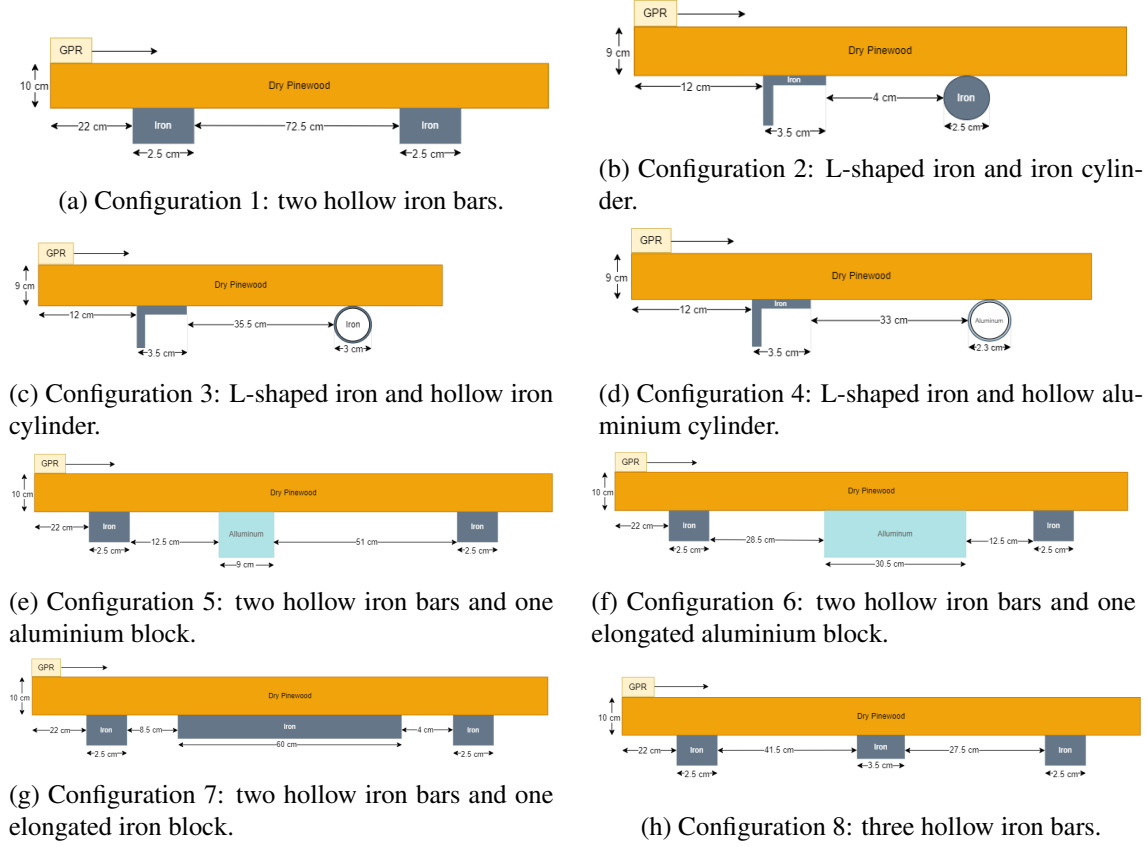


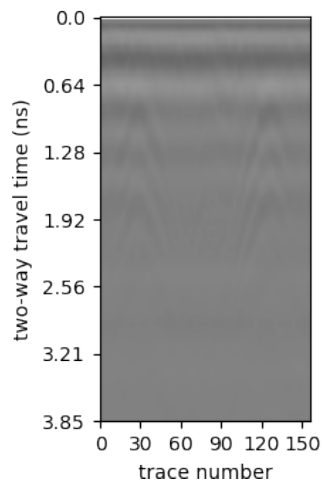
Figure 3.1: Diagram of the conducted GPR surveys. The distance between the GPR and the dry pinewood's surface was negligible (< 1 cm).

As mentioned in Section 2.1.2, the electromagnetic properties of a material heavily depend on its moisture level. The study from 2009, carried out by Kol, H. [88], evaluated the variation of the relative dielectric permittivity (ϵ_r) of pinewood with its moisture content. The estimate of the relative dielectric permittivity of pinewood at 0% moisture level was 1.6 [88]. From this estimate, it was possible to use Equation 2.2 to approximate the EM wave propagation velocity in the survey's medium (only considering dry pinewood). The result was ~ 23.72 cm/ns.

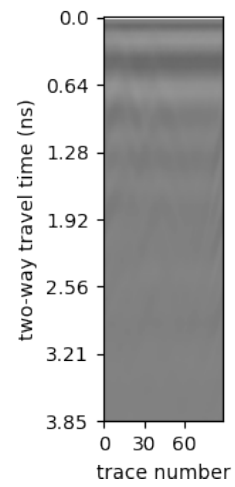
The maximum capture time window used on all surveys was 7.5 ns, which translates to a maximum capture depth of 88.94 cm (as per Equation 2.10). Moreover, as the number of samples per capture was 585, the sampling interval was $\frac{7.5}{585} = 0.0128$ ns or 0.152 cm (cf. Equation 2.10).

The captured B-scans are presented in Figure 3.2. All the scans were clipped after a two-way travel time of 3.85 nanoseconds, to minimize the processing time, as there were no hyperbolic shapes lower than that depth³. The different gray levels of the B-scans refer to the intensity of the reflections. Interest objects appear at lower gray levels, e.g., the presence of objects with different electromagnetic properties or changes in the propagation medium.

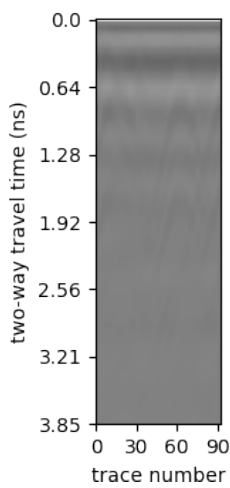
³Two-way travel time represents depth.



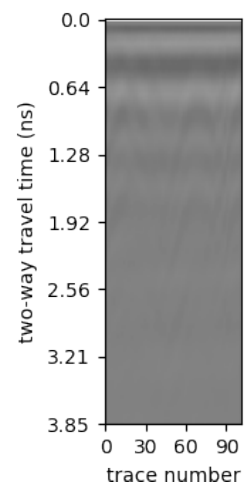
(a) Configuration 1.



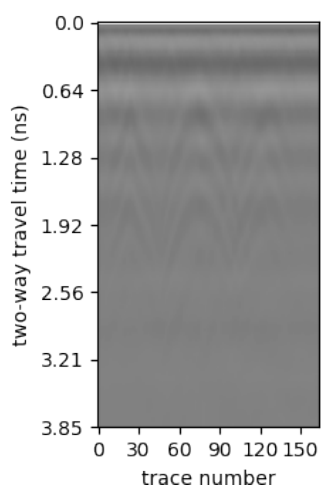
(b) Configuration 2.



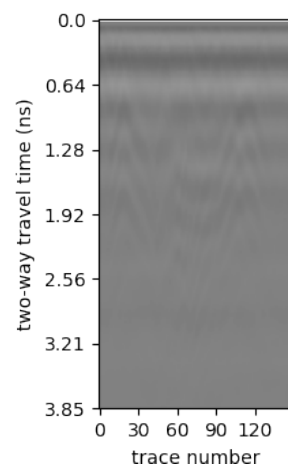
(c) Configuration 3.



(d) Configuration 4.



(e) Configuration 5.



(f) Configuration 6.

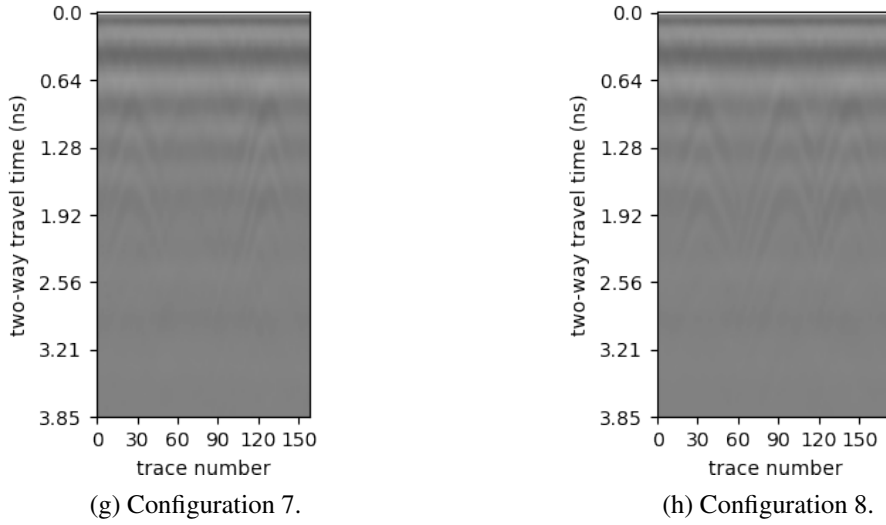


Figure 3.2: Original B-scans, captured on Configurations 1 to 8.

Finally, it is important to note that the GPR's release and capture of EM waves was determined by *Concretto*'s movement. Faster movement meant faster trace rate, and vice-versa. This was achievable due to the presence of an encoder on the GPR's left-back wheel. From the calibration interface provided by OERAD, it was estimated that the GPR captured an average of 130 traces per meter.

3.2 Architecture

The image processing techniques developed and applied to GPR data were highly influenced by other applications in the same field of expertise. Figure 3.3 presents an overview of the system's architecture.

Due to *Concretto*'s encoder, the GPR traces were collected at a fixed interval. As the GPR was moved along the survey line its firmware fed the captured traces (A-scans), as UDP datagrams, to a UDP Server. There, they were either stored in a database or forwarded to a *B-scan generator*. Before being fed to the object detector, the B-scans were pre-processed by a background removal filter and a normalization step. The system's output was the depth and position estimates of the detected objects.

The system's communication architecture is decomposed in four subsystems: *Ground Penetrating Radar*, *User Datagram Protocol (UDP) Server*, *User Datagram Protocol Client* and *B-scan Generator*. UDP was used considering it is a simple and fast network protocol, where error checks and recovery services are not a requirement [89]. Moreover, UDP services are widely preferred in real-time applications, as data is sent continuously without reception confirmations [89]. The subsystem's intercommunication is depicted in the sample sequence diagram of Figure 3.4.

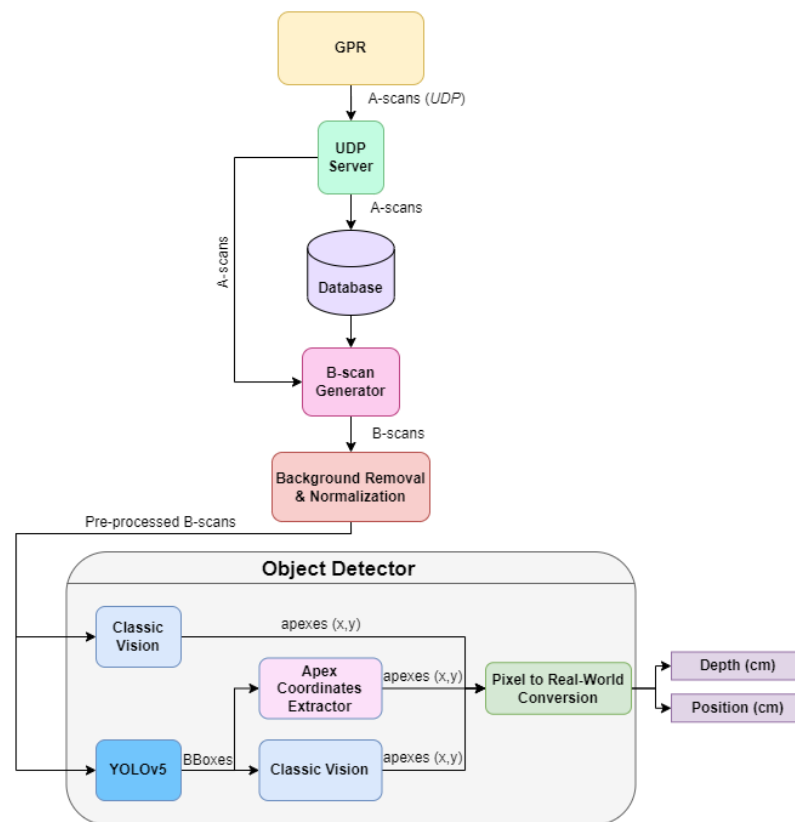


Figure 3.3: Overview of the system's architecture.

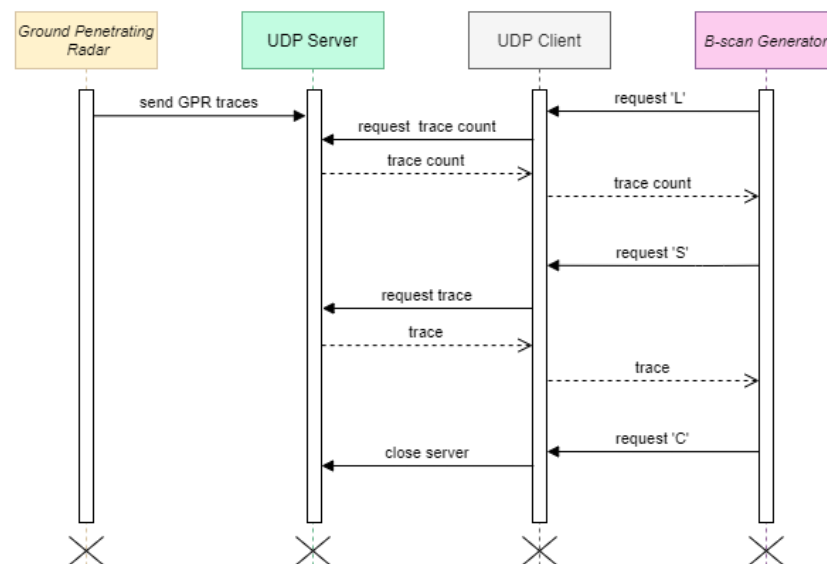


Figure 3.4: Sequence diagram of the subsystems' intercommunication.

As mentioned before, the GPR fed the captured traces, in real-time, to an UDP Server. When requested, the UDP Server sent information to the UDP Client. The UDP Client represented an interface between the *B-scan Generator* and the UDP Server, as it merely forwarded messages between the latter.

The *B-scan Generator* sent specific messages to the UDP Client, where they were translated before being forwarded to the UDP Server. The different message types and their translation were:

- 'L': request trace count i.e. how many traces the Server had on its buffer.
- 'S': request a trace from the Server's buffer.
- 'C': close the UDP Server.

After sending a trace to the Client, the Server erased it from the memory buffer, contributing to the system's memory efficiency, without hindering the hyperbolic detection. The UDP Client forwarded the trace to the *B-scan Generator*, where it was aggregated with the previously requested traces.

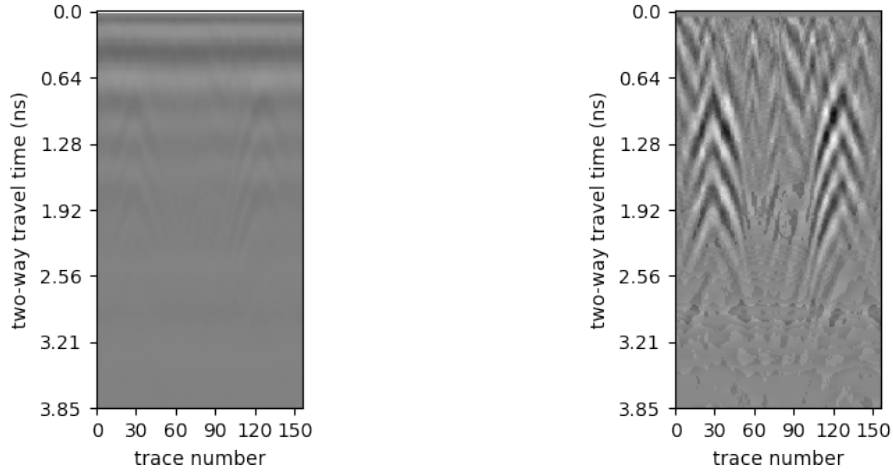
It is important to mention that the UDP protocol and the YOLOv5 model were implemented in *Python*, whilst the standard GPR pre-processing techniques and the classic vision algorithms were developed in *R*. Although *Python* has a very active and supportive community, is easy to learn and is more time-efficient than *R* [90], the latter was still chosen for the classic architecture because it provided a complete GPR-oriented library, with easy-to-follow hands-on tutorials, that allowed an easy understanding of the initial processing steps of GPR data (*RGPR* by Emanuel Hueber⁴). This contributed to a fast understanding of the practical implementations of standard GPR-related methodologies, such as the ones described in Section 2.2.1. Moreover, *R* is a language very easily integrated with *C++*, which is important when looking at the bigger picture: the integration of the GPR system in an autonomous robot. Robotics systems are usually run on the *Robot Operating System* (ROS) framework (which is mostly developed in *C++*).

3.3 Background Removal and Normalization

All collected B-scans presented direct waves between transmitter and receiver (between two-way travel times of 0 and 0.32 ns), medium's surface reflections (between two-way travel times of 0.32 and 0.64 ns), and low contrast around the hyperbolic signatures (cf. Figure 3.5a). To overcome these issues, a background removal filter (as explained in Section 2.2.1.1.3) was applied to all B-scans, before being submitted to the object detection methodologies.

Furthermore, to make the data values more interpretable and manageable, a normalization step was added after the background removal, converting all data points to floating-point values between 0 and 1. The comparison between the original B-scan (from Configuration 1) and its pre-processed version is visible in Figure 3.5.

⁴<https://github.com/emanuelhuber/RGPR>



(a) Original B-scan from Configuration 1 (used as example throughout this Section). (b) B-scan from Configuration 1 after pre-processing.

Figure 3.5: B-scan after the pre-processing steps of background removal and normalization.

Background removal and normalization were the global data pre-processing techniques applied to the original B-scans. All object detection algorithms ran on the pre-processed B-scans.

3.4 Classic Vision Techniques to Detect Hyperbolic Shapes in GPR Data

The classic vision object detector was split into two separate algorithms: hyperbola isolation and apex detection. The first was responsible for detecting and filtering valid hyperbolic candidates. The second detected the apex of the valid hyperbolic candidates. The overview of the classic vision object detector is available in Algorithm 1.

To better understand the result of the methodologies described in the following Sections, all transformations will be applied to the example B-scan of Configuration 1 (see Figure 3.5).

3.4.1 Hyperbola Isolation Algorithm

3.4.1.1 Automatic Global Thresholding

Automatic Global Thresholding (AGT) was a processing step that, as the name suggests, automatically found a global threshold (as per Section 2.2.2.1) that isolated the darker hyperbolic signatures⁵.

The algorithm iteratively searched for the best fitting threshold. The initial threshold, t_h , was the average pixel intensity of the B-scan (after background removal and normalization). After being applied to the B-scan, the average of the resulting binary image (a_{bin}) was compared against

⁵From a close inspection of the GPR B-scans, the darker hyperbolas provided a better match to the objects' real depth.

Algorithm 1 Overview of classic vision methodology to detect hyperbolic shapes in GPR data**Require:** x , pre-processed B-scan

-
- ```

1: ▷ Start of hyperbola isolation algorithm
2: $x_{th} \leftarrow \text{AGT}(x, m, s_t)$ ▷ Automatic Global Thresholding
3: $x_{CCF} \leftarrow \text{CCF}(x_{th}, c_p)$ ▷ Connected-components filtering
4: ▷ End of hyperbola isolation algorithm
5:
6: ▷ Start of hyperbola apex detection algorithm
7: for each x_{win} in $\text{SlidingWindow}(x_{CCF})$ do
8: $x_{ed} \leftarrow \text{skeleton}(x_{win})$
9: $lines \leftarrow \text{HoughLines}(x_{ed})$
10: $lines_F \leftarrow \text{FilterLines}(lines, dif)$
11: end for
12: $intersects \leftarrow \text{Intersection}(lines_F)$
13: $apexes_{xy} \leftarrow \text{FilterIntersections}(x_{th}, intersects, dif)$
14: ▷ End of hyperbola apex detection algorithm

```
- 

a margin value ( $m$ ): if  $a_{bin}$  was bigger than  $m$ , the threshold was reduced (to isolate darker signatures) by a step ( $s_t$ ), and the cycle would repeat; otherwise the cycle would end. The AGT procedure is found on Algorithm 2.

**Algorithm 2** Automatic Global Thresholding. The default values used in this dissertation were  $m = 0.18$ ,  $s_t = 0.01$ .**Require:**  $x$ , the image to be thresholded,  $m$ , the minimum value for the average of the resulting binary image,  $s_t$ , the threshold reduction step.

- 
- ```

1:  $t_h \leftarrow \text{avg}(x)$                                 ▷ Initial threshold
2:  $x_{bin} \leftarrow x < t_h$ 
3:  $a_{bin} \leftarrow \text{avg}(x_{bin})$ 
4: while  $a_{bin} > m$  do
5:    $t_h \leftarrow t_h - s_t$ 
6:    $x_{bin} \leftarrow x < t_h$ 
7:    $a_{bin} \leftarrow \text{avg}(x_{bin})$ 
8: end while

```
-

To increase the performance of the thresholding algorithm, an Adaptive Double Plateaus Histogram Equalization (ADPHE) was applied, prior to the computation of the first threshold. ADPHE is a classic image processing technique that automated the threshold selection of the Double Plateaus Histogram Equalization, to better enhance the details (low threshold) and further constraint the noise (high threshold) [91]. ADPHE updates the thresholds in real-time, to find the values that result in the best image enhancement [91].

A sequence of morphological operations and filters was used to clean-up some residual noise after the thresholding. The result of every step is visible in Figure 3.6.

It is important to mention that the AGT procedure was not tested on B-scans with reflective objects at different depths or with very distinctive electromagnetic behaviours, hence could lead to unsatisfactory results in those situations.

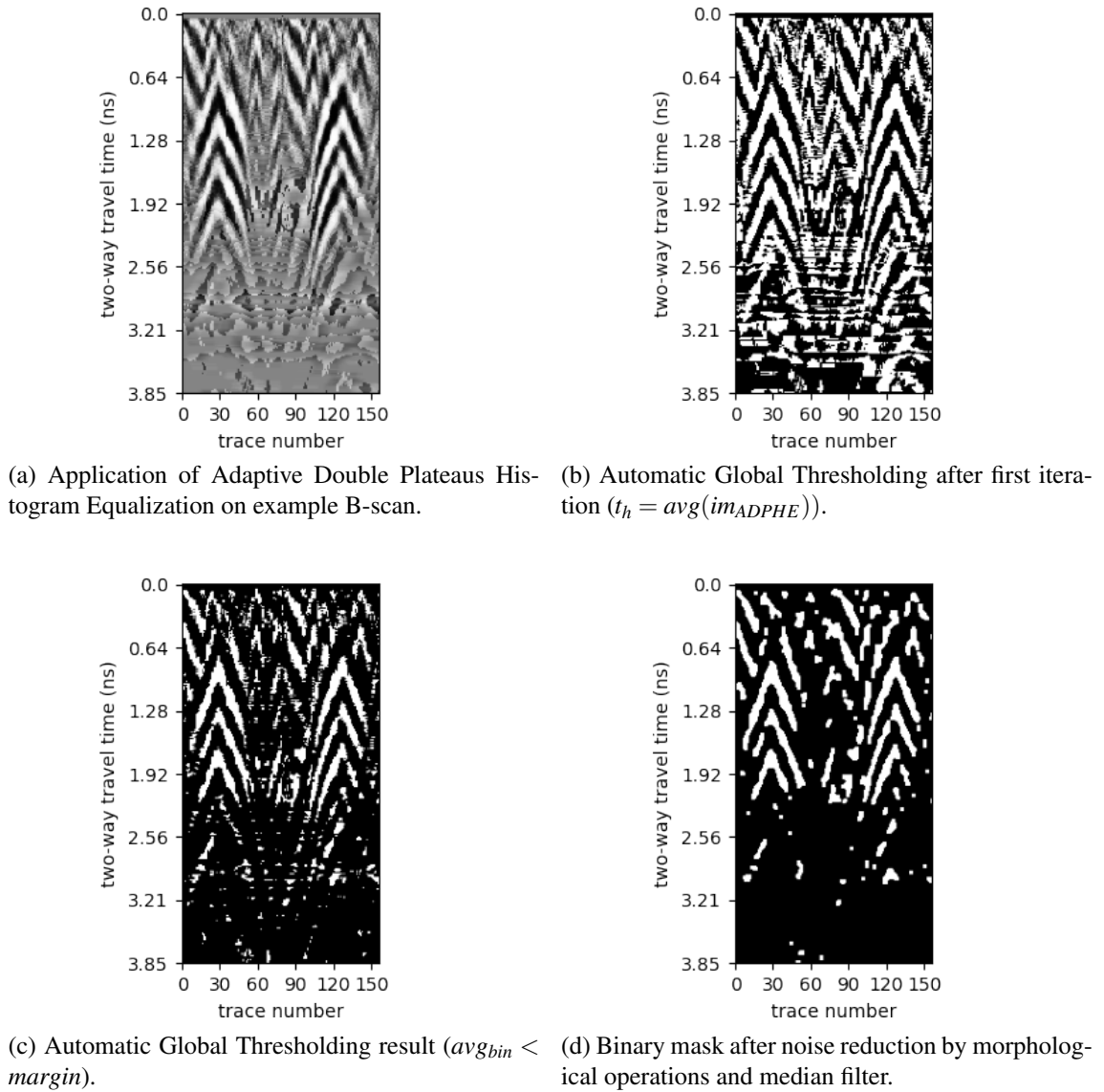


Figure 3.6: Automatic Global Thresholding applied to the example B-scan.

3.4.1.2 Connected-Components Filtering

As it is visible from Figure 3.6d, a lot of non-hyperbolic artifacts were still considered plausible candidates after the clean-up. To remove them, a connected-components-based filter was applied. The connected-components of a binary image are sets of pixels, connected to each other, that have an intensity value of 1 [92]. Each separated pixel group is labeled differently [92].

The connected-component filtering discarded all independent objects that had less pixels (n_{comp}) than a percentage (c_p) of the number of pixels of the component with maximum pixel count (see Equation 3.1). The result of this filtering step is visible in Figure 3.7.

$$label^i = \begin{cases} \text{Valid, if } n_{comp}^i \geq c_p \times \max(n_{comp}^j), \forall j \in labels \\ \text{Invalid, otherwise.} \end{cases} \quad (3.1)$$

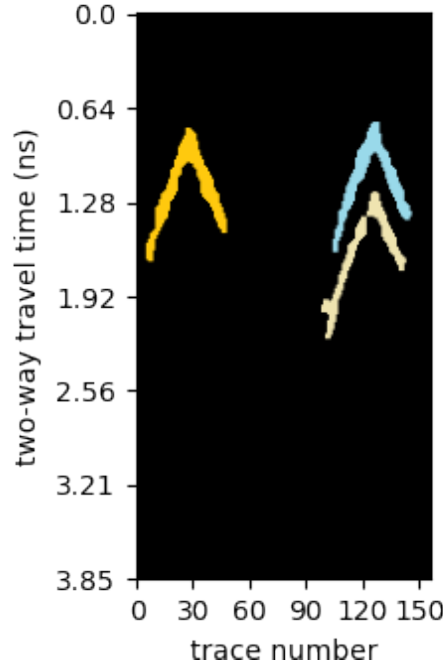


Figure 3.7: Connected-Components filtering applied to the example B-scan. Different colors represent different labels.

3.4.2 Hyperbola Apex Detection

The key idea of Hyperbola Apex Detection was to detect the lines that represented each arm of the candidate hyperbola and compute their intersection. The intersection point gave a fine approximation of the hyperbola's apex, and, consequently, of the object's depth and position.

3.4.2.1 Hough Transform

As described in Section 2.2.2.3, the *Hough* Transform detects lines by converting the images coordinates from Cartesian to polar format. Moreover, the *Hough* algorithm requires some form of an edge-enhanced image as input, so the line detection is correctly performed and the computational time is minimized.

To fulfil said requirements, a *Sobel* filter (cf. Section 2.2.2.2) was applied to the binary image. However, the fact that each hyperbolic arm is characterized by two edges (cf. Figure 3.8a) resulted in an under-performance of the *Hough* algorithm. To achieve optimal performance, the binary image was skeletonized, i.e., the foreground artifacts were reduced preserving the connectivity

of the original region [93], achieving a single "edge" that represented the skeleton of the object. Furthermore, the skeleton was thickened by a morphological dilation, to add more degrees of freedom to the line detection algorithm (see Figure 3.8b).

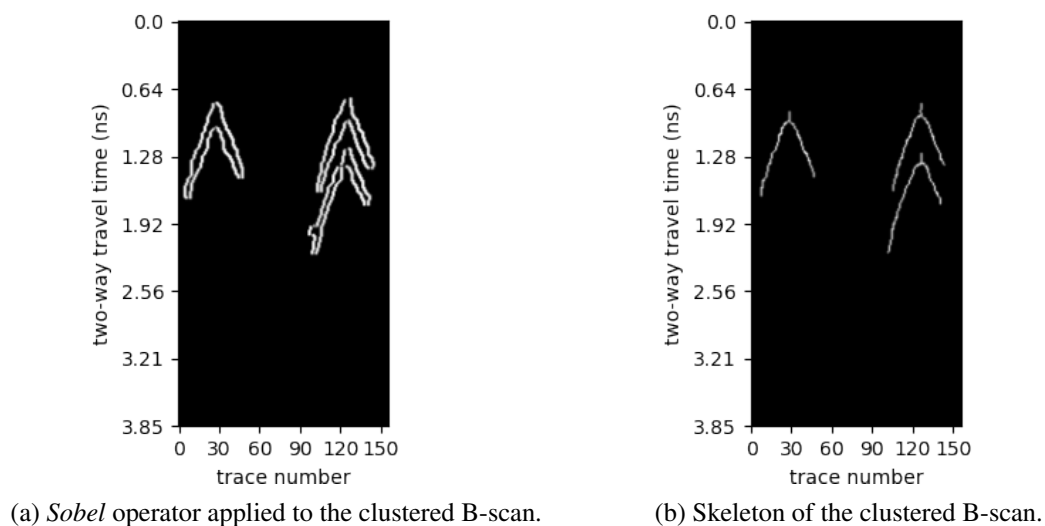


Figure 3.8: Comparison between *Sobel* operator and skeletonization, in generating the optimal input for the *Hough* transform.

A line was selected as a candidate if it obeyed three criteria:

1. The absolute angle (in the polar plane) was between 15° and 85° . These values ensured that vertical and horizontal lines were discarded;
2. The line's vote count was bigger than an absolute minimum value (defined as 10);
3. The line's vote count was bigger than a percentage of the maximum vote count of lines with the same direction. The percentage was defined as 90%, as it resulted in the best candidate selection.

The third criterion allowed a contextualized selection of lines with two degrees of freedom. As is visible from Figure 3.8, the left arms appear to be longer than the right arms, possibly due to the surveys' direction (left to right). This could lead to erroneous left-arm eliminations (if a direction-free measure was employed), or to an excess of candidates (if the percentage was lowered to overcome the size differences). By employing a directional measure, candidates of the same type were only compared against each other.

Nevertheless, this criterion was based on a global measure (or context), for each direction, which could lead to problems if different reflections resulted in very different arm lengths. To overcome this issue, an horizontal sliding window (see Figure 3.9) was applied to the skeletonized image, prior to the evaluation of the aforementioned criteria, creating a local context for each object, while maintaining data continuity. The window size used was 50 traces with an overlap of 20% (10 traces).

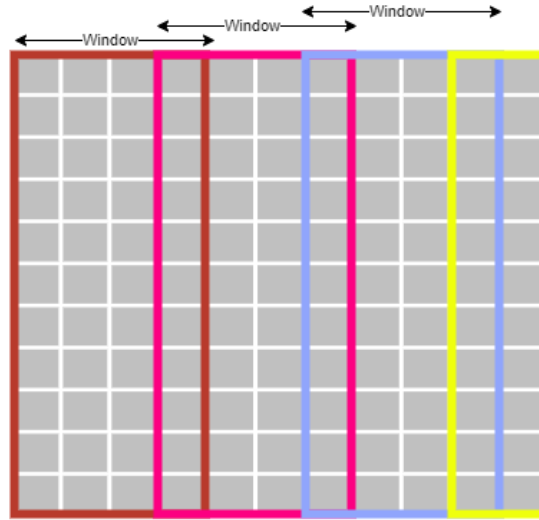


Figure 3.9: Horizontal sliding window visualization.

The application of the methodologies described in this Section resulted in multiple plausible candidates for each arm (see Figure 3.10) and zero meaningless candidates, e.g. candidate lines that jump from arm to arm.

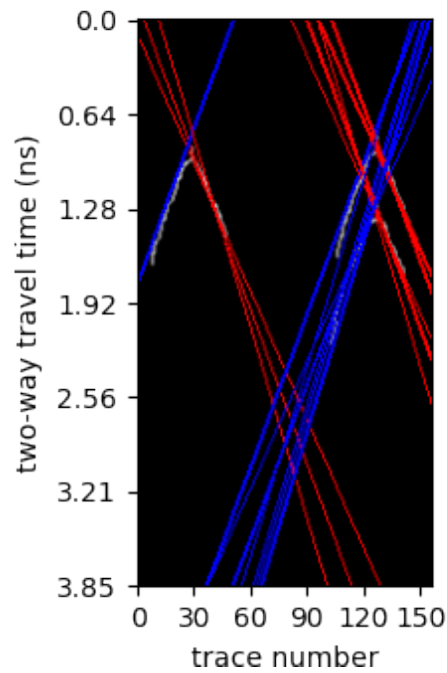


Figure 3.10: Candidate lines after *Hough* Transform and criteria evaluation. Blue lines represent positive-slope lines, red lines represent negative-slope lines.

3.4.2.2 Line Filtering

To fine-select the line(s) that best represented each arm, an additional line filtering step was implemented. The basic principle was to compute the average of candidate lines with the same direction that had a close y-intercept point. The y-intercept proximity threshold was 5 pixels.

As in the previous Section, the line filtering procedure covered candidates inside the sliding window. This technique guaranteed that each line could be a candidate to one, and only one, hyperbolic arm. Algorithm 3 presents the complete line filtering procedure.

Algorithm 3 Line filtering algorithm. The default value, used in this dissertation, for $dif = 5$.

Require: *lines*, the lines to be filtered in format (slope, y-intercept) (m , b), dif , maximum y-intercept difference that results in line aggregation.

```

1:  $gid \leftarrow 1$  ▷ Line group identifier
2:  $t_{lines} \leftarrow \dim(lines)$  ▷ Total number of lines
3:  $ids_{pos} \leftarrow \text{which}(lines.m > 0)$  ▷ Ids of lines with positive slope
4:  $ids_{neg} \leftarrow \text{which}(lines.m < 0)$  ▷ Ids of lines with negative slope
5: for each line in lines do
6:   if line.gid = 0 then
7:     next
8:   end if
9:   if line.m > 0 then ▷ Current line has a positive slope
10:     $aggids \leftarrow ids_{pos}[\text{which}(|line.b - lines.b| \geq dif)]$ 
11:  else
12:     $aggids \leftarrow ids_{neg}[\text{which}(|line.b - lines.b| \geq dif)]$ 
13:  end if
14:   $mSum \leftarrow 0$  ▷ Sum of the slopes of lines of the same group
15:   $bSum \leftarrow 0$  ▷ Sum of the y-intercepts of lines of the same group
16:  for each id in aggids do
17:     $line[id].gid \leftarrow gid$ 
18:     $mSum \leftarrow mSum + line[id].m$ 
19:     $bSum \leftarrow bSum + line[id].b$ 
20:  end for
21:   $t_{agg} \leftarrow \text{len}(aggids)$  ▷ Total number of aggregated lines in that group
22:   $filtered[gid].m \leftarrow \frac{mSum}{t_{agg}}$ 
23:   $filtered[gid].b \leftarrow \frac{bSum}{t_{agg}}$ 
24:   $gid \leftarrow gid + 1$ 
25: end for

```

Several candidates were still valid for each arm after the filtering step, however, the total candidate count was reduced (in the example B-scan from 23 to 15, see Figure 3.11), which validates the importance of this step for real-time applications (where the execution time is critical).

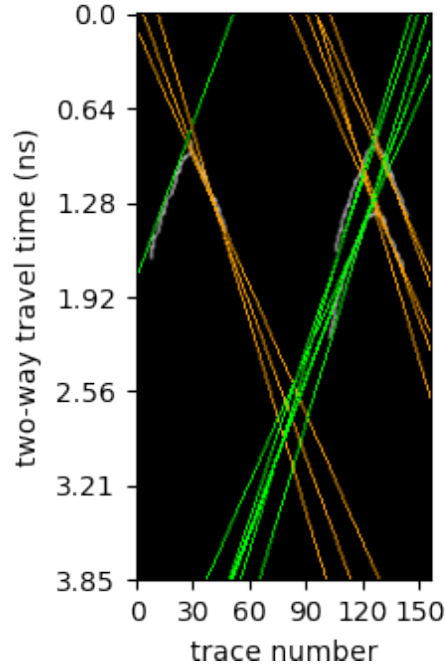


Figure 3.11: Candidate lines after the line filtering procedure. Green lines are positive-slope line aggregations, orange lines are negative-slope line aggregations.

3.4.2.3 Intersection Point Detection

The easiest way to compute the interception of two lines is to parameterize the lines in two-point form (see Equation 3.2). The computation of variables A , B , and C , requires two arbitrary points that belong to the line.

$$A \cdot x + B \cdot y = C \quad (3.2)$$

On previous processing steps, the candidate lines were parameterised by their slope, m , and y-intercept, b (see Equation 3.3). This parametrization allowed the computation of any point in the line.

$$y = m \cdot x + b \quad (3.3)$$

After computing two arbitrary points, defined by (x_1, y_1) and (x_2, y_2) , the system defined by Equation 3.4 could be solved, to parameterize the line in point-form.

$$\begin{cases} A = y_2 - y_1 \\ B = x_1 - x_2 \\ C = B \cdot y_1 + A \cdot x_1 \end{cases} \quad (3.4)$$

Assuming that two intersecting lines parameterized in two-point form were (A_1, B_1, C_1) and (A_2, B_2, C_2) , the coordinates of the intersection point were given by Equations 3.5 and 3.6. If the denominator of the aforementioned equations was zero, it meant that the lines were parallel, hence they didn't intersect.

$$x = \frac{B_2 \cdot C_1 - B_1 \cdot C_2}{A_1 \cdot B_2 - A_2 \cdot B_1} \quad (3.5)$$

$$y = \frac{A_1 \cdot C_2 - A_2 \cdot C_1}{A_1 \cdot B_2 - A_2 \cdot B_1} \quad (3.6)$$

The interception of candidate lines was only computed between lines with opposite directions, as intersecting two arms with the same direction does not provide useful information regarding the hyperbola's apex. The interceptions are visible in Figure 3.12.

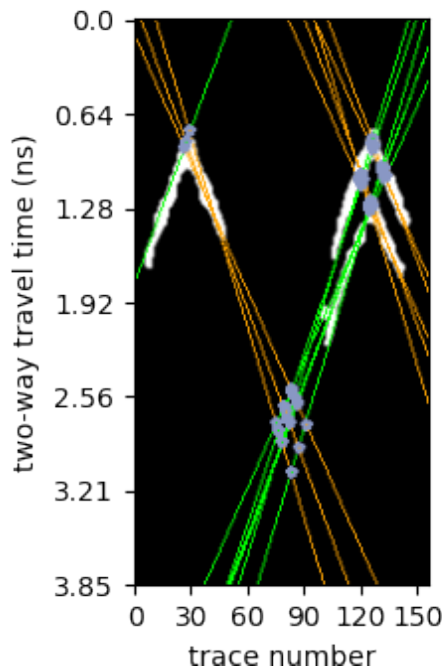


Figure 3.12: Interception of all candidate lines with opposite directions (violet dots).

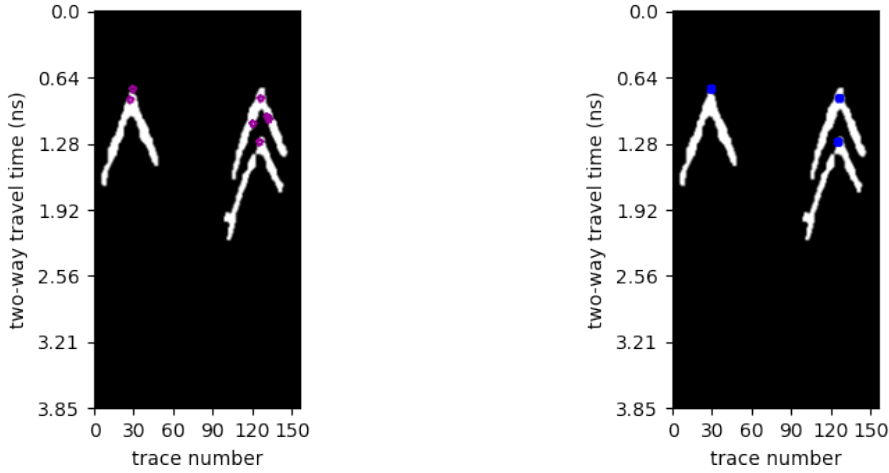
3.4.2.4 Intersection Point Filtering

To select the candidates that best represent the hyperbola apexes, three intersection point filtering steps were developed:

1. Intersection points close to each other were collapsed to a single point, represented by their average coordinates (see Figure 3.13a). The point proximity threshold was fine-tuned to 5 pixels.

For simplicity purposes, all candidate points after this step will be denominated 'master points'.

2. If the master point's coordinates did not match with any cluster point (cf. Section 3.4.1.2), i.e., if the binary image (Figure 3.7) on those coordinates was zero, the master point would take the label of the closest cluster in a 5×5 neighbourhood⁶. If no clusters were in the neighbourhood, the master point was discarded (cf. Figures 3.12 and 3.13a).
3. All master points belonging to the same cluster were collapsed into the point with the minimum two-way travel time, as it represented the best candidate for the cluster's hyperbola apex (see Figure 3.13b).



(a) Master intersection points, after intersection filtering steps 1 and 2.

(b) Hyperbola apex estimates, after intersection filtering step 3.

Figure 3.13: Intersection point filtering steps.

Algorithm 4 presents the intersection filtering procedures.

Intersection filtering is the final step the classic vision algorithm, hence it outputs the pixel coordinates (x, y) of all apex predictions.

3.5 Data Augmentation

Neural Networks require large quantities of data to be able to generalize their predictions to new situations/contexts. As mentioned in Section 3.1, in this dissertation only eight different GPR surveys were conducted, which is very little for training a neural network.

⁶ 5×5 neighbourhood gave the best results.

Algorithm 4 Intersection point filtering algorithm. The default value, used in this dissertation, for $dif = 5$.

Require: x , the binary image with the hyperbolic masks and their labels, $points$, the interception points to be filtered in format (x, y) , dif , maximum coordinates difference that results in point aggregation.

```

1:  $g_{id} \leftarrow 0$  ▷ Point group identifier
2:  $records \leftarrow 0$  ▷ Record array of processed point groups
3:  $window \leftarrow 2$  ▷ Neighbourhood size (in all directions)
4:  $w \leftarrow \text{width}(x)$ 
5:  $h \leftarrow \text{height}(x)$ 
6: for each  $point$  in  $points$  do
7:   if  $g_{id}$  in  $records$  then
8:     next
9:   end if
10:   $agg_{ids} \leftarrow \text{which}(|point.x - points.x| < dif \ \& \ |point.y - points.y| < dif)$ 
11:   $avg_x \leftarrow \text{avg}(points[agg_{ids}].x)$ 
12:   $avg_y \leftarrow \text{avg}(points[agg_{ids}].y)$ 
13:   $max_x \leftarrow \max(avg_x + window, w)$ 
14:   $min_x \leftarrow \min(avg_x - window, 0)$ 
15:   $max_y \leftarrow \max(avg_y + window, h)$ 
16:   $min_y \leftarrow \min(avg_y - window, 0)$ 
17:   $x_{neigh} \leftarrow x[min_x : max_x, min_y : max_y]$ 
18:  if  $\text{sum}(x_{neigh}) > 0$  then
19:    if  $x[avg_x, avg_y].label == 0$  then
20:       $c \leftarrow \text{floor}(window + 0.5)$ 
21:       $min_D \leftarrow 1000$ 
22:      for each  $position$  in  $x_{neigh}$  do
23:        if  $x_{neigh}[position].label \neq 0$  then
24:           $d \leftarrow |c - position|$ 
25:          if  $d < min_D$  then
26:             $min_D \leftarrow d$ 
27:             $min_L \leftarrow x_{neigh}[position].label$ 
28:          end if
29:        end if
30:      end for
31:       $filtered_p[g_{id}].label \leftarrow min_L$ 
32:    else
33:       $filtered_p[g_{id}].label \leftarrow x_{neigh}[avg_x, avg_y].label$ 
34:    end if
35:     $filtered_p[g_{id}].x \leftarrow avg_x$ 
36:     $filtered_p[g_{id}].y \leftarrow avg_y$ 
37:     $\text{append}(records, g_{id})$ 
38:     $g_{id} \leftarrow g_{id} + 1$ 
39:  end if
40: end for

```

To overcome this problem, some data augmentation techniques were developed in *Python*. These techniques included the generation of synthetic hyperbolic signatures, using a *Wasserstein Generative Adversarial Network*, and the generation of synthetic B-scans with randomly augmented hyperbolic signatures.

Both techniques required real hyperbolic artifacts, collected from the surveyed B-scans. As mentioned in Section 3.3, the original B-scans (Figure 3.2) had very low contrast around the hyperbolic reflections. To improve the quality of the synthetic data, the hyperbolic signature collection was retrieved from the B-scans after the background removal and normalization steps (example in Figure 3.5b). Consequently, the synthesized B-scans were B-scans after the background removal and normalization steps.

Figure 3.14 presents the different types of collected artifacts.

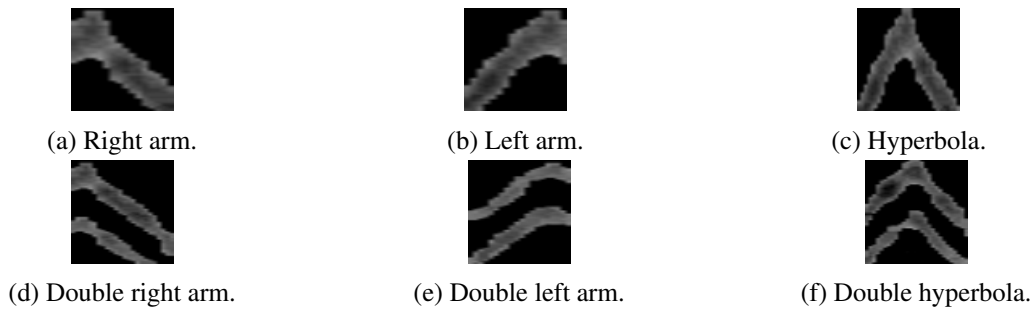


Figure 3.14: Hyperbolic artifact types.

To only collect the hyperbolic signatures (without background), the binary masks that resulted from the Automatic Global Thresholding (Figure 3.6d) were multiplied with the enhanced B-scans after the background removal and normalization (Figure 3.5b). The result was an image that isolated the darker hyperbolic signatures from the B-scan's background (see Figure 3.15). The collected images had the hyperbolic artifact over a black background (examples in Figure 3.14).

3.5.1 Generation of Synthetic Hyperbolic Signatures using a *Generative Adversarial Network*

A considerable problem of GANs are vanishing gradients. If the GAN architecture has a great discriminator and a bad generator [94], the loss function will most likely stabilize, and the generator will learn nothing [94]. To tackle this problem Arjovsky, M. et. al [95], proposed the usage of the *Wasserstein* distance as the GAN's loss function (hence the name *Wasserstein Generative Adversarial Network*). This function assured a smoother gradient everywhere, eliminating the problem of vanishing gradients [95]. Moreover, the WSGAN replaced the discriminator network by a critic. The critic is very similar to the discriminator but instead of a probability (defining if the generated image is real or not) it outputs a scalar score used to measure the *Wasserstein* distance [94].

The loss function of the critic network becomes

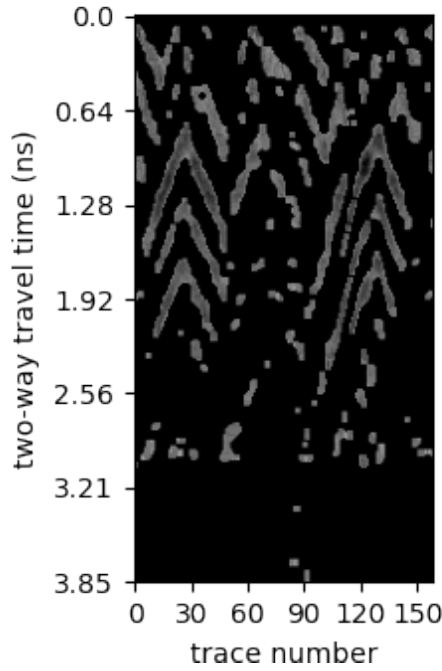


Figure 3.15: Result of point-wise multiplication of binary mask after the Automatic Global Thresholding (Figure 3.6d) with enhanced B-scan after background removal and normalization (Figure 3.5b).

$$\nabla_w \frac{1}{m} \sum_{i=1}^m [f(x^{(i)}) - f(G(z^{(i)}))], \quad (3.7)$$

and the loss function of the generator network becomes

$$\nabla_{\theta} \frac{1}{m} \sum_{i=1}^m f(G(z^{(i)})), \quad (3.8)$$

where f is a 1-Lipschitz function, i.e., a function that guarantees that

$$|f(x) - f(y)| \leq |x - y|, \quad (3.9)$$

for all x and y in the function's domain [96].

To enforce that the Lipschitz constraint is met, the standard WSGAN clips the critic's weights when these reach a certain value, c (defined as a hyperparameter) [94]. Arjovsky, M. et. al [95] admitted that weight clipping was a bad approach to enforce the Lipschitz constraint. They stated that for large values of c , the weights could never reach their limit, leading to poor critic performance [95]. On the other hand, if c was too small it could lead to gradient vanishing problems in deeper networks [95]. Regardless, the authors of the WSGAN, Arjovsky, M. et. al [95], kept the weight clipping step due to its implementation simplicity.

Gulrajani, I. et. al [97] concluded that weight clipping also affected the model's capability to estimate complex functions. Hence they proposed an improved version of the WSGAN: the WSGAN with *Gradient Penalty*. Instead of clipping the model's weights, they penalize the model if the L2-norm of the critic's gradient drifts from 1 [97].

To generate hyperbolic signatures the *Wasserstein Generative Adversarial Network* with *Gradient Penalty* was trained on a dataset of 160 real artifacts. The number of artifacts of each type was:

- Right arms: 36;
- Left arms: 35;
- Hyperbolas: 35;
- Double right arms: 22;
- Double left arms: 15;
- Double hyperbolas: 27.

The critic's architecture was composed of 5 two-dimensional convolutional layers, with a kernel size of 4, a stride of 2, and a padding of 1, activated by leaky ReLU units with a negative slope of 0.2. The generator's architecture was composed of 1 deconvolutional layer (transpose two-dimensional convolution) with a kernel size of 4, a stride of 1 and no padding, followed by 4 deconvolutional layers with a kernel size of 4, a stride of 2, and a padding of 1. All the deconvolutional layers were activated by ReLU units, except the last layer, which was activated by a hyperbolic tangent function (*tanh*).

The network was trained, according to the author's recommendations [98], for 100 000 iterations with a batch size of 8 images, which results in about 5000 epochs (see Equations 3.10 and 3.11, where n_Images is the total number of artifacts and it is the number of iterations).

$$batches = \frac{n_Images}{batch_size} \quad (3.10)$$

$$epochs = \frac{it}{batches} \quad (3.11)$$

The learning rate was 0.0002, the number of discriminator training steps per generator training step was 5 and the loss optimizer for both generator and discriminator was *Adam*.

As the WSGAN-GP framework [98] only allowed 64×64 input images, all artifacts were resized in advance. Furthermore, the artifacts were normalized with 0.5 mean and 0.5 standard deviation.

The evolution of a batch of synthetically generated hyperbolic artifacts is visible in Figure 3.16, where all samples had the same original noise distribution.

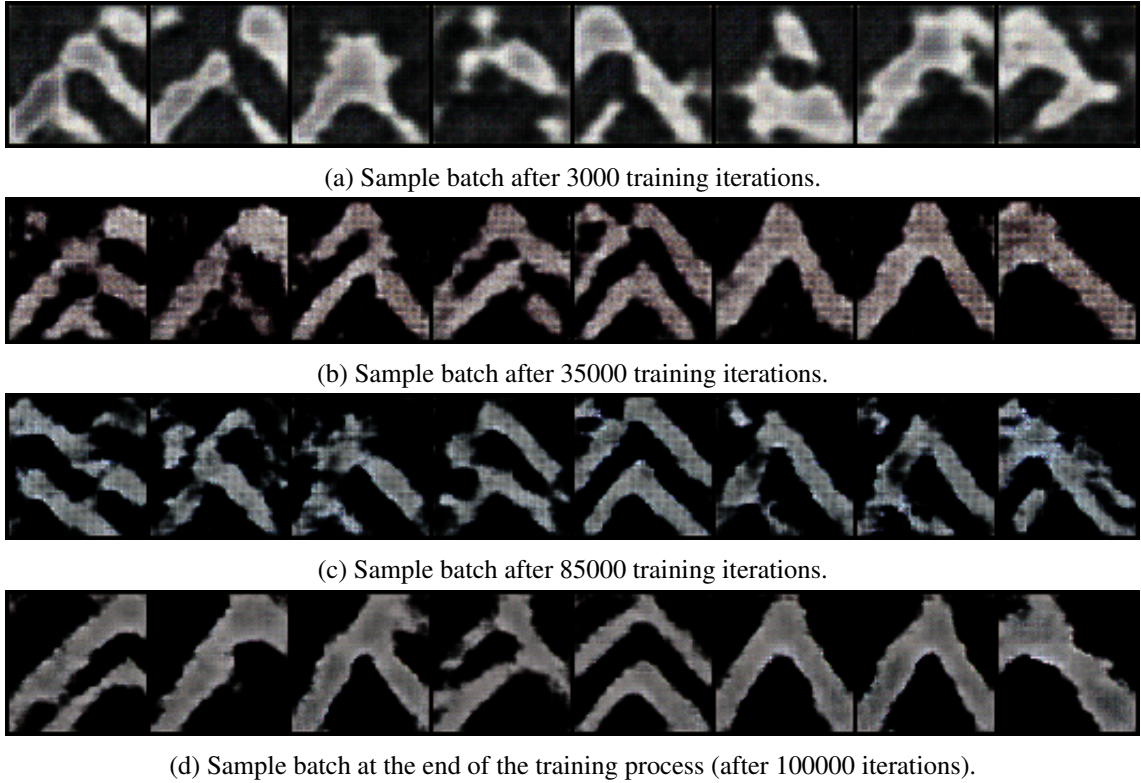


Figure 3.16: Artifact generation evolution over the training iterations.

At the end of the training round, the WSGAN-GP model was generating hyperbolic signatures almost indistinguishable from the originals: grayish hyperbolic artifacts over black backgrounds (cf. Figures 3.14 and 3.16d).

3.5.2 Generation of Synthetic GPR B-scans

The synthetic B-scan generation is separated into two different parts: background assembly and hyperbola placing.

3.5.2.1 Background Assembly

From the B-scan samples with a two-way travel time bigger than 3.85 nanoseconds, background blocks of 75×60 pixels, were gathered and stored. Similar to the collection of hyperbolic signatures, the background blocks were cropped from the B-scans after background removal and normalization (as the example of Figure 3.5b). In total, 45 different background blocks were gathered and two examples are visible in Figure 3.17.



(a) Background block example from survey of Configuration 7.

(b) Background block example from survey of Configuration 5.

Figure 3.17: Background block examples.

A synthetic background is a composition of random background blocks. These blocks were aggregated until the desired image size was achieved, even if it meant clipping the blocks to fulfil the size requirements. A 190×300 (190 traces with a depth of 3.85 ns) synthetic background example is visible in Figure 3.18.

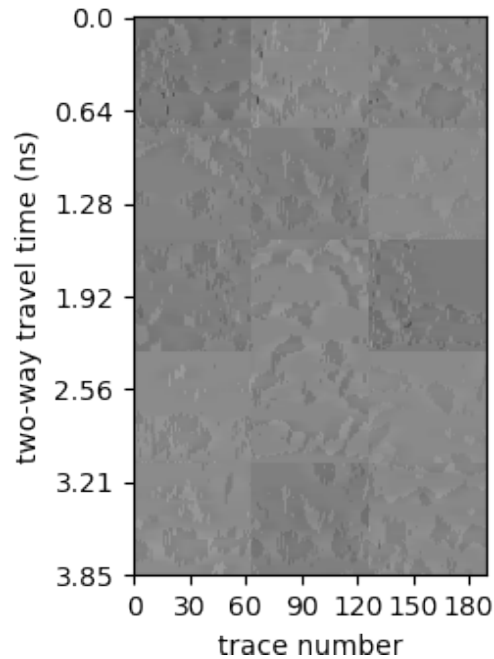


Figure 3.18: Synthetic background with 190 traces and a maximum two-way travel time of 3.85 ns.

3.5.2.2 Hyperbola Placing

The artifacts were randomly drawn and altered by a sequence of random transformations, to increase their variability.

To effectively get a transparent background, all black pixels from the artifact's images were set with an alpha-channel⁷ value of zero, i.e., they became completely invisible.

The artifact augmentation sequence was composed of the following steps:

1. Brightness adjustment: the artifacts had their brightness decreased by a random factor between 50 and 60%;
2. Random rotation (only applied to single/double right/left arms): the artifacts suffered a random absolute rotation between 0° and 15°;
3. Random artifact compression (only applied to single/double hyperbolas): the artifacts had a random compression, in width, between 0 and 20% of their original width. This resulted in narrower artifacts;
4. Random vertical flip: the artifacts were randomly flipped over their vertical (y) axis (probability of 50%). This was mainly important to add variability to non-symmetrical artifacts;
5. Random clipping: the artifacts were randomly clipped in width and height. The clipping was dependent on the type of object, to ensure that no key feature was disregarded (e.g. in double arms, erasing one of the arms):
 - Right/left arms and hyperbolas: horizontal reduction between 0 and 25% of their width; vertical reduction between 0 and 25% of their height;
 - Double right/left arms and double hyperbolas: horizontal reduction between 0 and 25% of their width; vertical reduction between 0 and 15% of their height;

The horizontal clipping direction differed depending on the object type, to preserve the type's individual characteristics. The directions were:

- Single/double right arms: reduction from right to left;
- Single/double left arms: reduction from left to right;
- Single/double hyperbolas: reduction from both sides.

The hyperbolic shapes of the B-scans after background removal and normalization were characterized by a black and white pattern (cf. Figure 3.5b), however, the Automatic Global Thresholding binary mask only isolated the darker areas, thus the collected artifacts were all dark. When generating synthetic B-scans it was desired that the degree of similarity with respect to a real B-scan was maximized. To achieve this, the grayscale levels of the augmented artifacts were reversed⁸, i.e., the darker areas became lighter and vice-versa. The reverse-colored artifacts were placed below the original ones with an offset of 7 pixels, resulting in a final artifact 7 pixels taller than the originals. To induce a smooth transition from the darker to lighter areas, the final artifacts

⁷The alpha-channel controls the transparency of the pixel/image [99]

⁸To reverse a grayscale image, *im*, with 256 gray values, is to do $255 - im$.

were convolved with a *Gaussian* filter. Figure 3.19 show an augmented double hyperbola and its original form, where augmentation steps 1, 3, and 5 are clearly noticeable.



(a) Original double hyperbola.



(b) Augmented double hyperbola.

Figure 3.19: Example of hyperbolic artifact after data augmentation procedures.

The final artifact was placed in a random location on the synthetic background. The maximum number of artifacts per synthetic image was a parameter required by the *Python* script. The total number of artifacts of a synthetic B-scan was randomly chosen (limited by the maximum). After the total amount of objects was placed on the synthetic background, random speckle noise was added. Speckle noise is a multiplicative noise type that commonly appears in radar-like applications due to fluctuations in the captured signal [100]. This noise type was added to the original image using Equation 3.12, where im was the original image, $N(0, 1)$ was a random Gaussian distribution with zero mean and standard deviation of one and p was a random percentage between 10 and 15%.

$$im_n = im + N(0, 1) \cdot im \cdot p \quad (3.12)$$

In total, 160 real and 200 synthetic artifacts were used to generate the synthetic B-scans. Figure 3.20a presents an example of a synthetic B-scan with 190 traces and a maximum two-way travel time of 3.85 nanoseconds.

The annotations of generated images, i.e., bounding box dimensions and classes, were completely automatized. The artifact's bounding box and its placement location, i.e., where the bounding box's top-left corner should be placed, were fully known, at all times. The class of the object was also, evidently, known. The annotated example synthesized B-scan image is visible in Figure 3.20b.

The bounding box annotations had the form $[tl_x, tl_y, br_x, br_y]$ with (tl_x, tl_y) and (br_x, br_y) being the relative coordinates of the top left and bottom right corners, respectively. Relative coordinates express the absolute coordinates as a proportion of the image's dimensions. The x coordinates are a proportion of the image's width (Equation 3.13) and the y coordinates a proportion of the image's height (Equation 3.14).

$$r_x = \frac{or_x}{im_w} \quad (3.13)$$

$$r_y = \frac{or_y}{im_h} \quad (3.14)$$

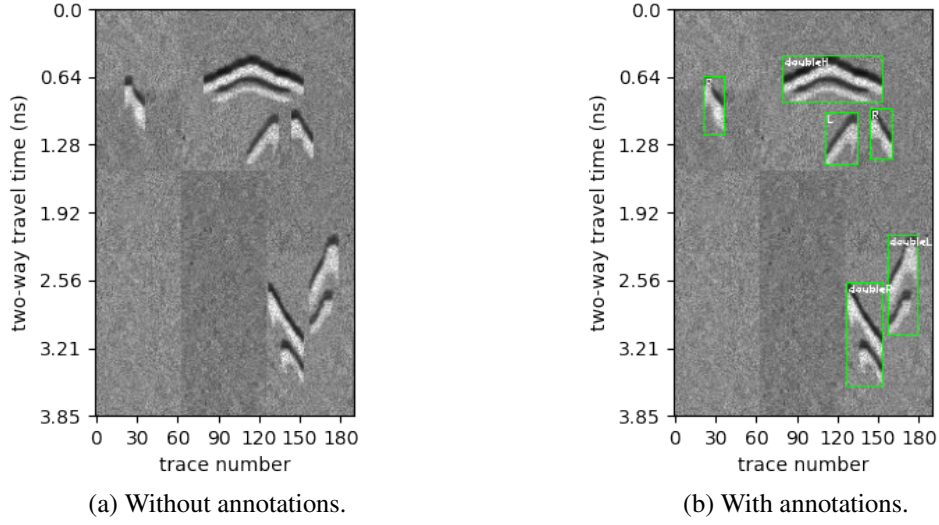


Figure 3.20: Example synthesized B-scan, with and without annotations.

3.6 YOLOv5 architectures applied in Detection of Hyperbolic Shapes in GPR Data

To follow the state-of-the-art in GPR data processing, the application of neural networks became compulsive. The application of the most recent version of YOLO networks had the intent of accurately detecting hyperbolic artifacts and estimating their depth and position, in real-time.

3.6.1 Datasets

Due to the small amount of real GPR data available, the models were trained on completely synthetic datasets, synthesized following the methodologies described in Section 3.5.2. Since every synthetic image was randomly generated, the best way to split training and validation sets was to separate the hyperbolic artifacts used in the B-scan generation. The train/validation split at artifact-level is presented in Table 3.1. Separating training and validation sets at artifact level meant that when generating synthetic B-scans, e.g., for the training dataset, the artifacts would come exclusively from the training artifacts set. The training/validation split was done with a ratio of about 70/30%, as these are the common values for deep learning [6]. The only type of synthetic artifact (GAN-generated) used to train/validate the YOLOv5 models were the synthesized double hyperbolas.

The training datasets were progressively created and adapted to optimize the networks' predictions on real data. Table 3.2 summarizes the created datasets.

The dataset used to test the performance of all trained models was composed of the real B-scans after background removal and normalization (example in Figure 3.5b).

Table 3.1: Train/Validation dataset split at artifact level.

Artifact	Train		Validation	
	Real	Synthetic	Real	Synthetic
Right arm	26	-	10	-
Left arm	25	-	10	-
Hyperbola	22	-	13	-
Double right arm	15	-	7	-
Double left arm	10	-	5	-
Double hyperbola	20	49	7	25

Table 3.2: Characterization of the datasets used in the training YOLOv5 Nano and YOLOv5 Small. **Artifacts** indicates the used artifacts. R and S define the artifact's origin: real or synthetic. **# classes** represents the number of classes of the datasets. **Train/Val set size** is the number of generated images used for training and validating the networks, respectively.

Dataset	Image size	Artifacts	# classes	Train set size	Val set size
1	(75, 300)	all (R)	6	1400	600
2	(300, 300)	all (R)	6	1400	600
3	(75, 300)	all (R)	2	1400	600
4	(300, 300)	all (R)	2	1400	600
5	(75, 300)	double hyperbolas (R)	1	1000	200
6	(75, 300)	double hyperbolas (R & S)	1	1000	200

3.6.2 Models

As stated in Section 2.2.3.1, YOLO networks are mostly used for their real-time capabilities, as with low inference times they achieve proper accuracy values. Despite the misleading names, YOLOv4 and YOLOv5 architectures came to the public almost at the same time (one month apart), and both improve the YOLOv3 (version 3). The choice between YOLOv4 or YOLOv5 came down to the framework, as they are both state-of-the-art architectures. YOLOv5 uses *Python*'s neural network package *PyTorch*, hence is very easy to install. Furthermore, it comes with a very explicit start guide with hand-on-tutorials.

The used YOLOv5 models were the Nano and the Small. Their architectures are very similar. YOLOv5 Nano has a smaller architecture, hence it has a faster inference time, but its accuracy is lower (evaluated on the *Common Objects in Context* (COCO) dataset, by Jocher. G [86]). YOLOv5 Small is architecturally bigger and, consequently, slower, but achieves higher accuracy values [6]. The performance comparison of all YOLOv5 models can be visualized in Figure 3.21.

The YOLOv5 models share the same architectural core. Their backbone is a Cross Stage Partial (CSP) DenseNet [7, 101]. DenseNets were designed to reduce the problem of vanishing gradients in deep CNNs, boost feature propagation and reinforce feature re-utilization [7]. The basic architecture of a DenseNet is visible on Figure 3.22.

CSP DenseNets separate the feature map of the base layer not only to send it through the dense block but also directly to the next stage in the CNN [7]. This aimed to reduce computational

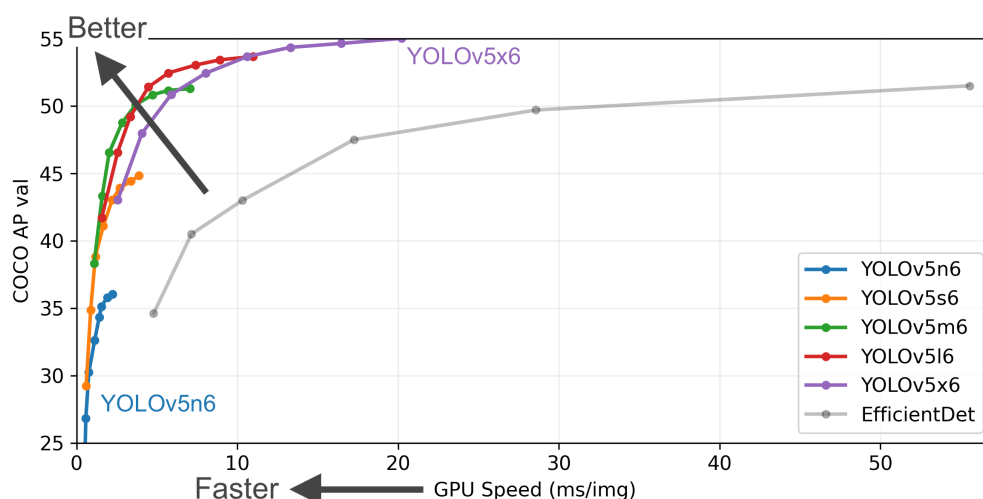


Figure 3.21: Performance analysis of YOLOv5 models (from [6]).

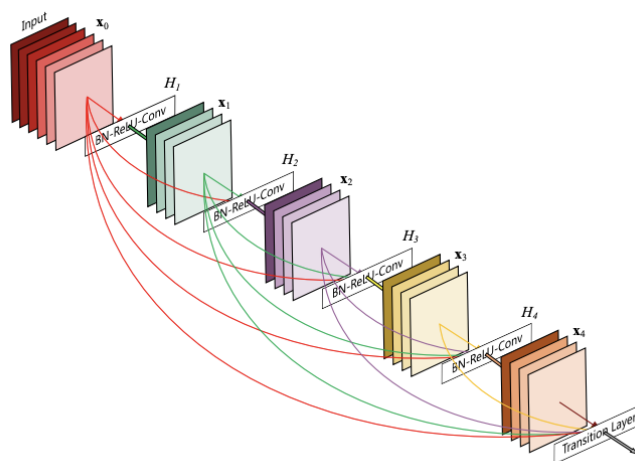


Figure 3.22: DenseNet architecture (from [7]). The colored squares represent convolutional layers.

bottlenecks and improve the network's learning. Furthermore, this type of backbone addresses the duplicate gradient problems in other large CNNs by reducing the number of parameters and the number of floating-point operations per second (FLOPS) [7]. This metric is of the utmost importance for the real-time performance of the YOLOv5. The YOLOv5's backbone has 5 dense blocks. The difference between the Nano and Small models is that the latter has a 50% wider backbone [6], resulting in more trainable parameters.

The YOLOv5's neck is a PANet. Its main function is to aggregate the intermediate feature maps collected by the backbone, using a series of downsampling, upsampling, and fusion steps [7, 102]. These strongly enhance the models' ability to detect features at different scales [102]. PANet's basic architecture is visible on Figure 3.23.

YOLOv5's full architecture is summarized in Figure 3.24. As mentioned before the width and

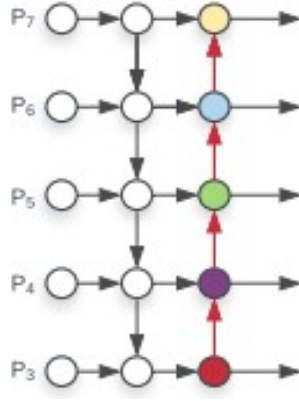


Figure 3.23: PANet architecture (from [7]). Each P_i block represents an feature map outputted by a dense block.

depth of the network depend on the model, however, the difference between YOLOv5 Nano and YOLOv5 Small is merely 50% in width.

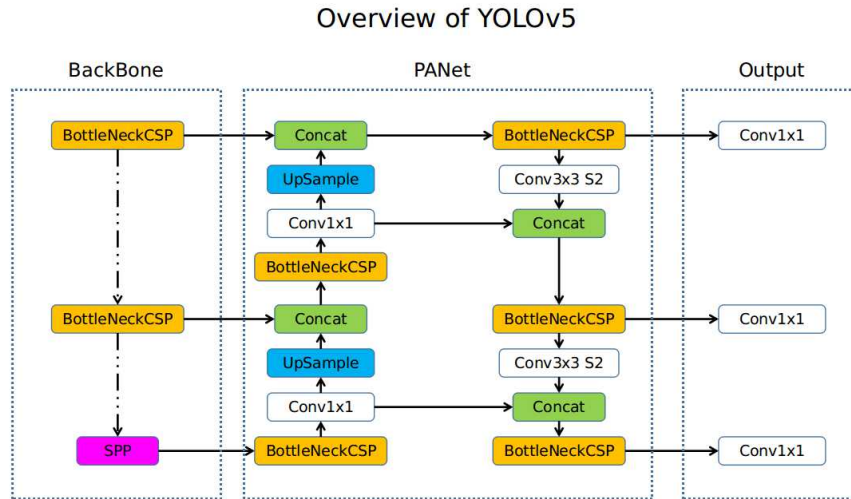


Figure 3.24: YOLOv5's basic architecture (from [6]).

The training hyperparameters of the models were kept at their default values (proposed by the YOLOv5 authors [6]) and are summarized in Table 3.3.

The network configurations trained on datasets 1 to 4 (cf. Table 3.2) were trained for 100 epochs and used the weights from a pretrained model on the COCO dataset [86] (transfer learning). The remaining configurations were trained for 20 epochs and had random weight initialization.

The output of object detection models is composed of a list of bounding boxes with confidence levels and predicted classes. As so, during training, YOLO networks have three separate loss terms: classification loss, confidence loss, and localization loss [103, 104]. Classification

Table 3.3: Hyperparameters used to train YOLOv5 networks (Nano and Small).

Hyperparameter	Value
Learning Rate	0.0001
Optimizer	SGD
Momentum	0.937
Weight decay	0.0005
Box loss gain	0.05
Class loss gain	0.5
Object loss gain	1.0
IoU training threshold	0.2

loss is the cross-entropy loss regarding the class of the detected object [103, 104]. Confidence loss measures the confidence of object presence [103, 104] through binary cross-entropy [104]. Localization loss is the bounding box regression loss [104] and measures the errors in the predicted bounding box locations and sizes [103]. It is a mean square error loss [103, 104]. The global loss is a weighted⁹ sum of the loss terms.

YOLOv5 requires 640×640 images as input with bounding box annotations in the format $[x_c, y_c, b_w, b_h]$, where x_c and y_c are the relative coordinates of the bounding box center, and b_w and b_h are its relative width and height. This meant that all images used in model training required a pre-treatment to be resized and re-annotated. These transformations, applied to the example of Figure 3.20, resulted in Figure 3.25.

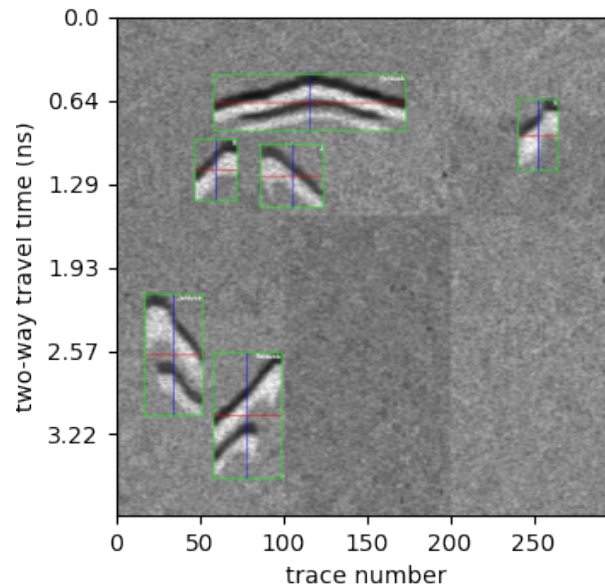


Figure 3.25: Example synthesized B-scan after resize and re-annotation steps. The green boxes with white text represent the original bounding boxes. The red and blue crosses represent the YOLO-format labelling.

⁹The loss weights used for the trained models are given in Table 3.3 as *Box/Class/Object loss gain*.

As previously mentioned, the outputs from the YOLOv5 models are bounding boxes (in the same format as the input annotations), with confidence levels and classes.

3.7 Integration of YOLOv5 Models with Classic Vision Techniques to Detect Hyperbolic Shapes in GPR Data

Algorithm 5 presents an overview of the methodologies applied to enable the successful integration of classic vision techniques with state-of-the-art YOLOv5 networks.

Algorithm 5 Overview of the integration of YOLOv5 models with classic vision methodologies to detect hyperbolic shapes in GPR data

Require: x , pre-processed B-scan

```

1:  $BBoxes \leftarrow \text{YOLOv5}(x)$ 
2:  $x_{bb} \leftarrow \text{crop}(x, BBoxes)$   $\triangleright$  Crop the B-scan's portions bounded by the bounding boxes
3:  $x_{th} \leftarrow \text{AGT}(x_{bb}, m, s_t)$   $\triangleright$  Automatic Global Thresholding
4:  $x_{CCF} \leftarrow \text{CCF}(x_{th}, c_p)$   $\triangleright$  Connected-components filtering
5: for each  $x_{Vwin}$  in  $\text{VerticalSlidingWindow}(x_{CCF})$  do
6:    $x_{ed} \leftarrow \text{skeleton}(x_{Vwin})$ 
7:    $lines \leftarrow \text{HoughLines}(x_{ed})$ 
8:    $lines_F \leftarrow \text{FilterLines}(lines, dif)$ 
9:    $intersects \leftarrow \text{Intersection}(lines_F)$ 
10:  if  $intersects$  is not None then
11:     $apex_{xy} \leftarrow \text{FilterIntersections}(x_{th}, intersects, dif)$   $\triangleright$  Only one apex per bounding box
12:    return
13:  end if
14: end for

```

As mentioned in Section 3.6.2, the output of the YOLOv5 networks is composed of a set of bounding boxes with confidence levels and classes. To perform hyperbola isolation and apex detection from these outputs, the B-scan's area bounded by the predicted bounding boxes had to be cropped. The cropped B-scan portions were then forwarded to the classic vision algorithm. Given their smaller size the fact that they were mainly composed of objects of interest, some adaptations had to be done to the classic vision methodologies.

Firstly, the AGT margin, m , that determined the maximum percentage of the binary image that an object of interest could occupy, was widened from 0.18 to 0.25. Given the fact that the input image (cropped B-scan) was the predicted location (area) of an artifact of interest, smaller margins would result in very small candidates and the disappearance of important artifact features. On the other hand, bigger margins would result in candidates being disturbed by background noise.

Given the dimensions of the input image, the morphological operations (after the AGT algorithm) had a great impact on the features, leading to partitions in the artifact's hyperbolic signature. As so, the percentage that ruled the size of the dismissed connected-components, c_p , was lowered from 0.6 to 0.2. Bigger values of c_p resulted in the elimination of hyperbolic features, whilst lower values did not eliminate unwanted (background noise-generated) components.

The sliding window's direction was also changed. Horizontal sliding windows only made sense in continuous processing of B-scans, where multiple objects could be present on the same image and/or where the objects could be clipped during the B-scan generation. In this case, the inputs were discrete, i.e., bounded areas that predicted the location of an object. Nonetheless, the objects could be composed of multiple hyperbolic signatures, thus the application of a vertical sliding window¹⁰ allowed every hyperbolic signature of the same object to be considered individually. The vertical window size was 40 pixels with an overlap of 5% (2 pixels).

The final algorithmic adjustment was made to the intersection filtering. After the master points' filtering, all detected master points were collapsed into the master point with the smallest y-coordinate, independently of their cluster. This resulted in a single prediction per bounding box: the one that gave the best estimate for the apex of the shallowest hyperbolic reflection.

The small input dimensions contributed to the adjustment of multiple fine-tunable parameters: the absolute minimum value of votes of the lines detected by the *Hough* Transform was raised from 10 to 17; the maximum line y-intercept difference that resulted in line aggregation was decreased from 5 to 2 pixels; and the cluster neighbourhood of the interception point filtering was decreased from a 5×5 to a 3×3 window.

3.8 Conversion from pixel to real-world coordinates

The estimation of the depth and positions of the predicted objects differed depending on the object detection algorithm.

The x-coordinate of the predicted apexes of the classic vision (solo) algorithm was converted to position coordinates by means of the traces-per-meter ratio (tpm) (estimated through the calibration step mentioned in Section 3.1) and Equation 3.15.

The center x-coordinate of the predicted bounding boxes of the YOLOv5 (solo) algorithm was converted to position using Equation 3.15.

The position estimates of the combined methodologies were computed using Equation 3.16, where x_{coord} is the apex's x-coordinate and $bbox_{tlx}$ is the x-coordinate of the bounding box's top left corner.

$$p = \frac{x_{coord}}{tpm} \quad (3.15)$$

$$p = \frac{x_{coord} + bbox_{tlx}}{tpm} \quad (3.16)$$

The y-coordinate of the predicted apexes of the classic vision (solo) algorithm was converted to depth by means of the time/depth sampling intervals (calculated in Section 3.1) and Equation 3.17, where t_s is the time sampling interval, d_s is the depth sampling interval, v is the medium's EM propagation velocity and y_{coord} is the apex's y-coordinate.

¹⁰A transposed version of the horizontal sliding window of Figure 3.9.

To estimate the depth of the YOLOv5 (solo) algorithm's predictions, the y-coordinate of the bounding boxes' top-left corner was converted to depth values using Equation 3.17.

The depth estimates of the combined methodologies were computed using Equation 3.18, where $bbox_{tly}$ is the y-coordinate of the bounding box's top left corner.

$$d = t_s \cdot \frac{y_{coord} \cdot v}{2} = d_s \cdot y_{coord} \quad (3.17)$$

$$d = t_s \cdot \frac{(y_{coord} + bbox_{tly}) \cdot v}{2} = d_s \cdot (y_{coord} + bbox_{tly}) \quad (3.18)$$

Chapter 4

Results and Discussion

4.1 Results

4.1.1 Expected Results

As explained in the previous Chapter, the GPR surveys were captured in a controlled environment, where the location of the reflective objects was known (cf. Figure 3.1). Furthermore, from the analysis of the hyperbolic signatures of the collected B-scans (cf. Figure 3.2), it was possible to visualize which objects provided detectable reflections and which did not, and with that information determine the ground-truth, i.e., the expected responses from the developed algorithms. The ground-truth of the captured B-scans is available on Table 4.1. The object's absolute position (p_{GT}) represents the distance from the start of the survey until the object's central point (where the reflected hyperbolic signatures peak).

Although Configurations 6 and 7 had three different objects (cf. Figures 3.1f and 3.1g), the GPR could not capture them successfully. On the B-scan from Configuration 6 (Figures 3.2f), the presence of an elongated object results in a weak and distorted hyperbolic reflection. On the other hand, the elongated object from Configuration 7 is completely indistinguishable from the B-scan's background (Figure 3.2g). As explained in Chapter 2, when survey lines are conducted parallel or near-parallel to the object's main axis (which was true for Configurations 6 and 7), the hyperbolic reflections are often weak and unpredictable.

4.1.2 Metrics

As the algorithms utilize different methodologies, the prediction format also differs. The classic image processing algorithms predict the location of the hyperbola by its apex, whilst the YOLOv5 architectures infer the hyperbola's location by means of a bounding box with a confidence level.

The ground-truth is the same for all methodologies (summarized in Table 4.1). To ensure correct performance assessment and comparison between different methodologies, the evaluation metrics must be standardized.

Table 4.1: ground-truth number of objects (N_{GT}) per configuration (**Conf.**), their depth (d_{GT}) and their absolute position (p_{GT}).

Conf.	N_{GT}	d_{GT} (cm)	p_{GT} (cm)
1	2	10.00	23.25
		10.00	98.25
2	2	9.00	13.75
		9.00	20.75
3	2	9.00	13.75
		9.00	52.50
4	2	9.00	13.75
		9.00	49.65
5	3	10.00	23.25
		10.00	41.5
		10.00	98.25
6	2	10.00	23.25
		10.00	98.25
7	2	10.00	23.25
		10.00	98.25
8	3	10.00	23.25
		10.00	67.75
		10.00	98.25

To assess the systems' performance on detecting artifacts, the *Precision* (Equation 4.1) and *Recall* (Equation 4.2) metrics are used. *Precision* evaluates how correct the model's predictions are. *Recall* measures the percentage of correctly identified ground-truth objects. Ideally, both measures would have a value of 100%.

$$P = \frac{TP}{TP + FP} \quad (4.1)$$

$$R = \frac{TP}{TP + FN} \quad (4.2)$$

To further understand these measures, the terminologies true positives (TP), false positives (FP) and false negatives (FN), must be defined. True positives represent the correct object predictions, either by its apex or by its bounding box. False positives are predictions that should not be, i.e., that don't belong to the ground-truth. False negatives are ground-truth objects that are undetected by the algorithms.

On the classic system, an apex estimate is said to be correct, if its coordinates lie within the hyperbola originated by the ground-truth object's reflection. On the CNN-based system, the measure of prediction correctness during training/testing is the intersection over union (IoU) with respect to the ground-truth bounding boxes.

Intersection over union is defined as the ratio between the intersection area of the ground-truth bounding box (BB_{GT}) and the predicted bounding box (BB_P), and their union (see Equation

4.3). When applied to object detectors, correct box predictions are those, and only those, where the IoU value is above the IoU threshold, t_{IoU} [68].

$$IoU = \frac{area(BBox_{GT} \cap BBox_P)}{area(BBox_{GT} \cup BBox_P)} \quad (4.3)$$

The aforementioned metrics allow the system's performance evaluation when it comes to object detection. To assess the performance with respect to object depth and position, the absolute errors ($\Delta\epsilon^i$) of the true positive predictions ($\hat{o}^i$), when compared with the ground-truth values (GT^i), are calculated using Equation 4.4.

$$\Delta\epsilon^i = |GT^i - \hat{o}^i| \quad (4.4)$$

4.1.3 Classic Image Processing Techniques to Detect Hyperbolic Shapes in GPR Data

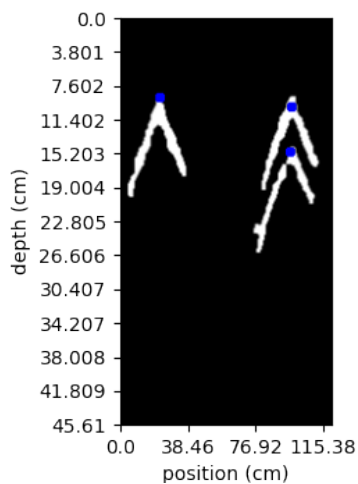
The evaluation of the performance of the classic image processing algorithm is split into two categories: hyperbola isolation and apex detection.

Ideally, the hyperbola isolation techniques should only isolate the top hyperbola, as that is the artifact that characterizes the object's position and depth. Although in most configurations the hyperbola isolation functioned properly, in some configurations secondary hyperbolas were found. The presence of artifacts without vertices in the binary masks was the main source of false positive apex detections. Anyhow, the apex detection methodologies mostly correctly predicted the location of the hyperbola's apexes.

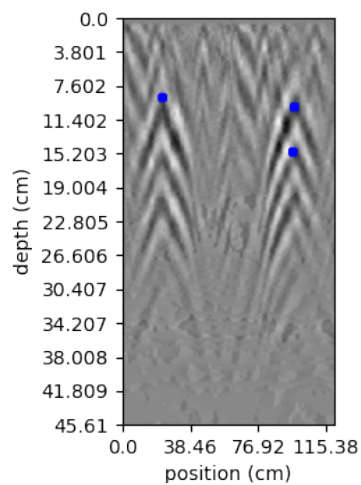
The results of the classic image processing algorithm are summarized in Table 4.2. The depth (d_o) and position (p_o) estimates are only performed for the true positive predictions. The binary masks and the predicted hyperbola apexes are presented in Figure 4.1.

The analysis of Table 4.2 and Figure 4.1, provides information on the source of erroneous estimates (FP and FN). The majority of FP predictions originate in secondary hyperbolas (Figures 4.1a-4.1b, 4.1i-4.1j and 4.1o-4.1p). Partial hyperbolas result in unpredictable behaviour, with either true apex prediction (TP) (Figures 4.1c-4.1d), erroneous apex prediction (FP) (Figure 4.1k-4.1l) or correct artifact dismissal (Figure 4.1k-4.1l). Overall, the algorithm has a *Precision* of 81.0% and a *Recall* of 94.4%.

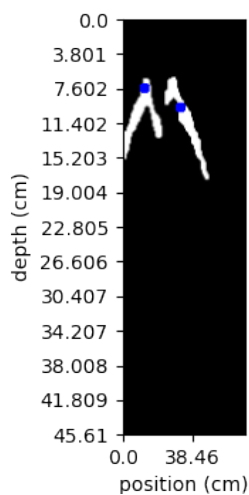
Table 4.3 presents the absolute depth and position errors of the true positive apex predictions.



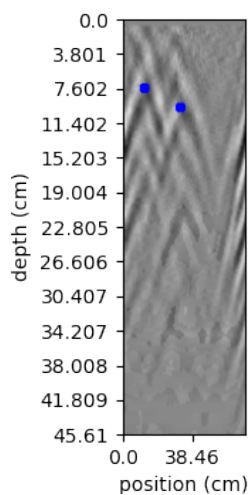
(a) Hyperbola masks from Configuration 1.



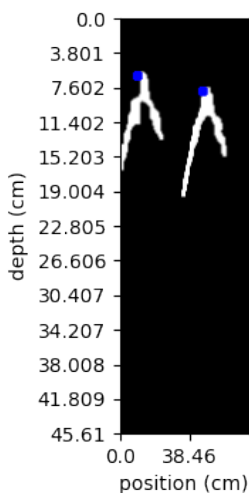
(b) Detected hyperbola apexes on B-scan from Configuration 1.



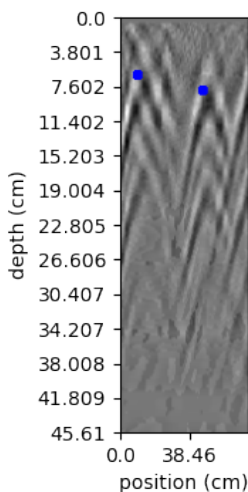
(c) Hyperbola masks from Configuration 2.



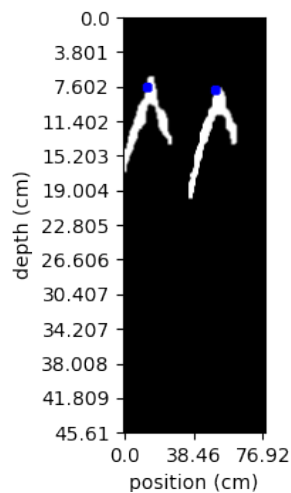
(d) Detected hyperbola apexes on B-scan from Configuration 2.



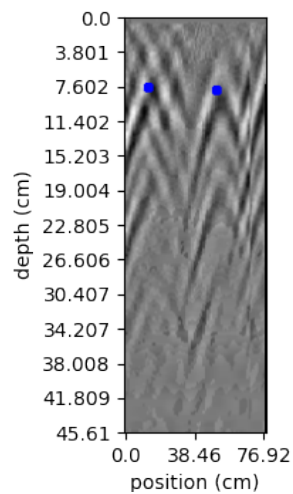
(e) Hyperbola masks from Configuration 3.



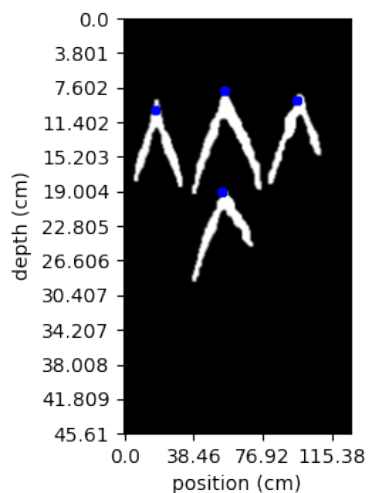
(f) Detected hyperbola apexes on B-scan from Configuration 3.



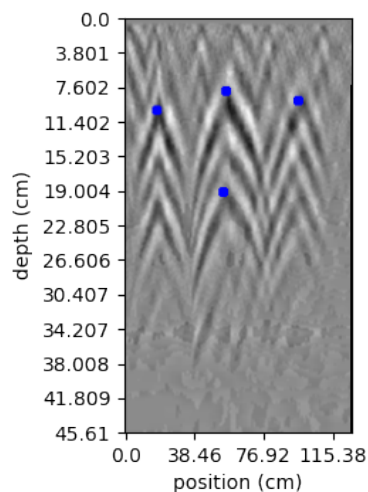
(g) Hyperbola masks from Configuration 4.



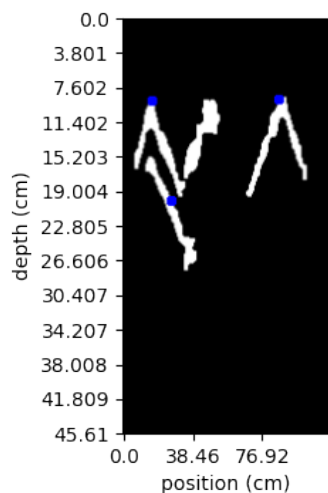
(h) Detected hyperbola apexes on B-scan from Configuration 4.



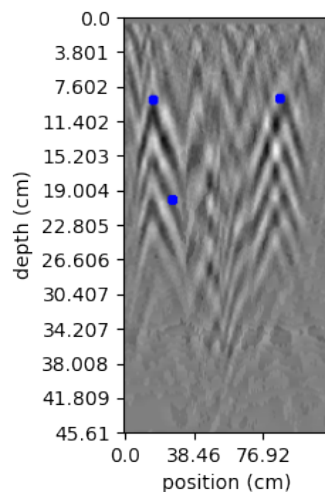
(i) Hyperbola masks from Configuration 5.



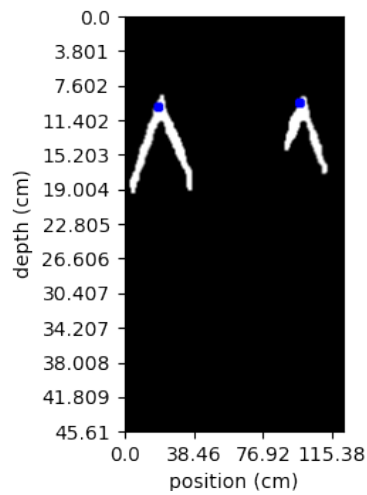
(j) Detected hyperbola apexes on B-scan from Configuration 5.



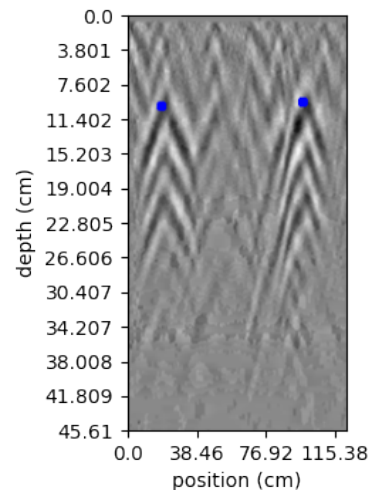
(k) Hyperbola masks from Configuration 6.



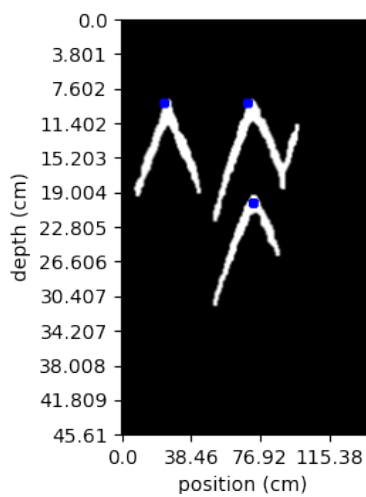
(l) Detected hyperbola apexes on B-scan from Configuration 6.



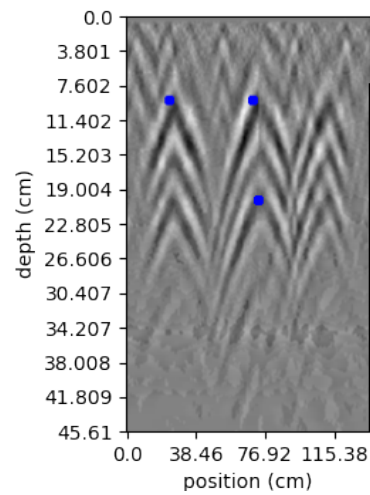
(m) Hyperbola masks from Configuration 7.



(n) Detected hyperbola apexes on B-scan from Configuration 7.



(o) Hyperbola masks from Configuration 8.



(p) Detected hyperbola apexes on B-scan from Configuration 8.

Figure 4.1: Results of the application of the classic image processing algorithm, on all surveyed configurations.

Table 4.2: Summarized results of the classic image processing algorithm: false negative (**FN**), false positive (**FP**) and true positive (**TP**) apex predictions, and depth (d_o) and position (p_o) estimates of the true positive predictions of each configuration (**Conf.**).

Conf.	FN	FP	TP	d_o (cm)	p_o (cm)
1	0	1	2	8.37	23.08
				9.37	97.69
2	0	0	2	7.09	12.31
				9.08	32.31
3	0	0	2	5.96	10.00
				7.52	46.15
4	0	0	2	7.24	13.85
				7.52	51.54
5	0	1	3	9.51	18.46
				7.52	56.92
				8.51	96.92
6	0	1	2	8.51	16.92
				8.37	86.92
7	0	0	2	9.37	19.23
				8.93	97.69
8	1	1	2	8.66	24.62
				8.66	70.77

Table 4.3: Absolute depth error ($\Delta\epsilon_d$) and absolute position error ($\Delta\epsilon_p$) of true positive apex predictions (**TP**). **Conf.** is the surveyed configuration.

Conf.	TP	$\Delta\epsilon_d$ (cm)	$\Delta\epsilon_p$ (cm)
1	2	1.63	0.17
		0.63	0.56
2	2	1.91	1.44
		0.08	11.56
2	2	3.04	3.75
		1.48	3.5
4	2	1.76	0.1
		1.48	1.89
5	3	0.49	4.79
		2.48	15.42
		1.49	1.33
6	2	1.49	6.33
		1.63	11.33
7	2	0.63	4.02
		1.07	0.56
8	2	1.34	1.37
		1.34	3.02

The average absolute depth error was 1.41 cm, with a standard deviation of 0.72 cm. The average absolute position error was 4.18 cm, with a standard deviation of 4.52 cm.

4.1.4 YOLOv5 architectures applied in Detection of Hyperbolic Shapes in GPR Data

As mentioned in Chapter 3, 6 different datasets were generated for the YOLOv5 models to train on. That resulted in a total of 12 training rounds (6 per model). To ease the visualization and interpretability of the presented data in Section 4.1.4.1, only the training and validation rounds of the best performing model will be summarized.

4.1.4.1 Training and Validation

The training and validation rounds of the YOLOv5 models were completely performed on synthetic data. The YOLOv5 framework [6], provides a great interface to visualize and evaluate the model's training and validation evolution. In addition, it also automatically computes multiple evaluation metrics, namely *Precision*, *Recall*, *F1-score*, *Precision × Recall* curve and *Mean Average Precision* (mAP).

4.1.4.1.1 Metrics

The *F1-score* summarizes *Precision* and *Recall* in one single measure (see Equation 4.5). *Precision*, *Recall* and *F1-score*, when used in object detector networks, are usually plotted as function of different bounding box confidence thresholds.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4.5)$$

The *Precision × Recall* curve, is a plot that correlates *Precision* and *Recall*, at different IoU thresholds (t_{IoU}). This metric is mostly used to compute the model's mAP.

The average precision (AP) of a dataset class is represented by the area under the *Precision × Recall* curve (AUC), of that class, for a specific t_{IoU} value. In multi-class training rounds (e.g. using dataset 1), the mAP is calculated as the average of the AP of all classes. On the other hand, in single-class training rounds (e.g. using dataset 6), the mAP has numerically the same value as the AP. Similarly to AP, the mAP is evaluated as a function of the IoU threshold, and consequently, the correct terminology is $mAP@t_{IoU}$.

4.1.4.1.2 Training evaluation

The best performing model was the YOLOv5 Nano, which trained for 20 epochs on dataset 6 (one class, with GAN-generated artifacts). This model was trained to only detect double hyperbolas.

As stated before, all the metrics and plots are automatically computed/generated by the YOLOv5 framework [6]. Figure 4.2 summarizes the training results by training epoch.

The first six plots, from left to right, show the progression of the training and validation loss functions, by epoch. The training and validation classification loss (*train/val cls_loss*) is always

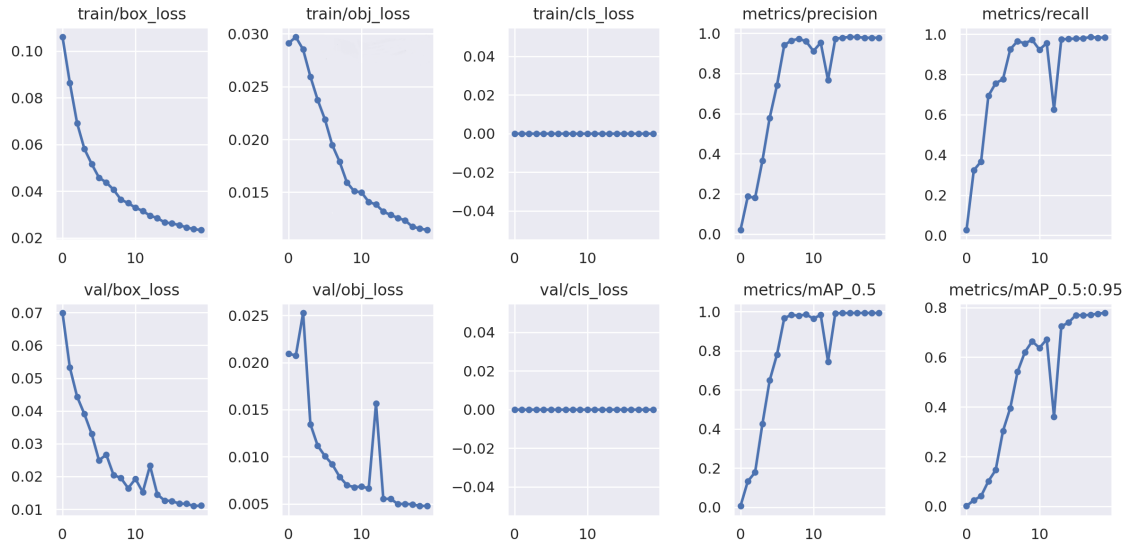


Figure 4.2: Training and validation results of YOLOv5 Nano trained on dataset 6. The horizontal axis of all plots represents the training epochs.

zero as there was only one class. The training and validation localization and confidence losses (*train/val box_loss* and *train/val obj_loss* respectively) present an exponential decrease throughout the training epochs. At the end of the 20 training epochs, all loss functions are reaching stability i.e. their rate of change is approaching zero.

The last four plots, from left to right, present the evolution of the different validation metrics. *Precision* and *Recall* were evaluated for validation confidence threshold of 0.001 (the default value of the YOLOv5 framework). *Precision*, *Recall* and *mAP@0.5*, present values close to 100% by the end of the 20 epochs. The *mAP@0.5:0.95* (*metrics/mAP_0.5:0.95*), represents the validation *mAP* for *IoU* thresholds ranging from 0.5 to 0.95, and seems to reach equilibrium at $\sim 75\%$, by the end of the training epochs.

4.1.4.1.3 Post-training validation

After the training epochs were completed, the YOLOv5 framework runs a final validation round, where *Precision*, *Recall* and *F1-score* are computed for different confidence thresholds (steps of 0.001). Figures 4.3a, 4.3b and 4.3c, show the variation of these metrics with the confidence threshold.

Figure 4.3d shows the *Precision* $\times$ *Recall* curve for an *IoU* threshold of 0.5. The AUC gives a *mAP@0.5* of 99.3%. This *mAP@0.5* value shows that at an *IoU* threshold of 0.5 the number of false positives and false negatives is very low, resulting in a model with high *Precision* and high *Recall*.

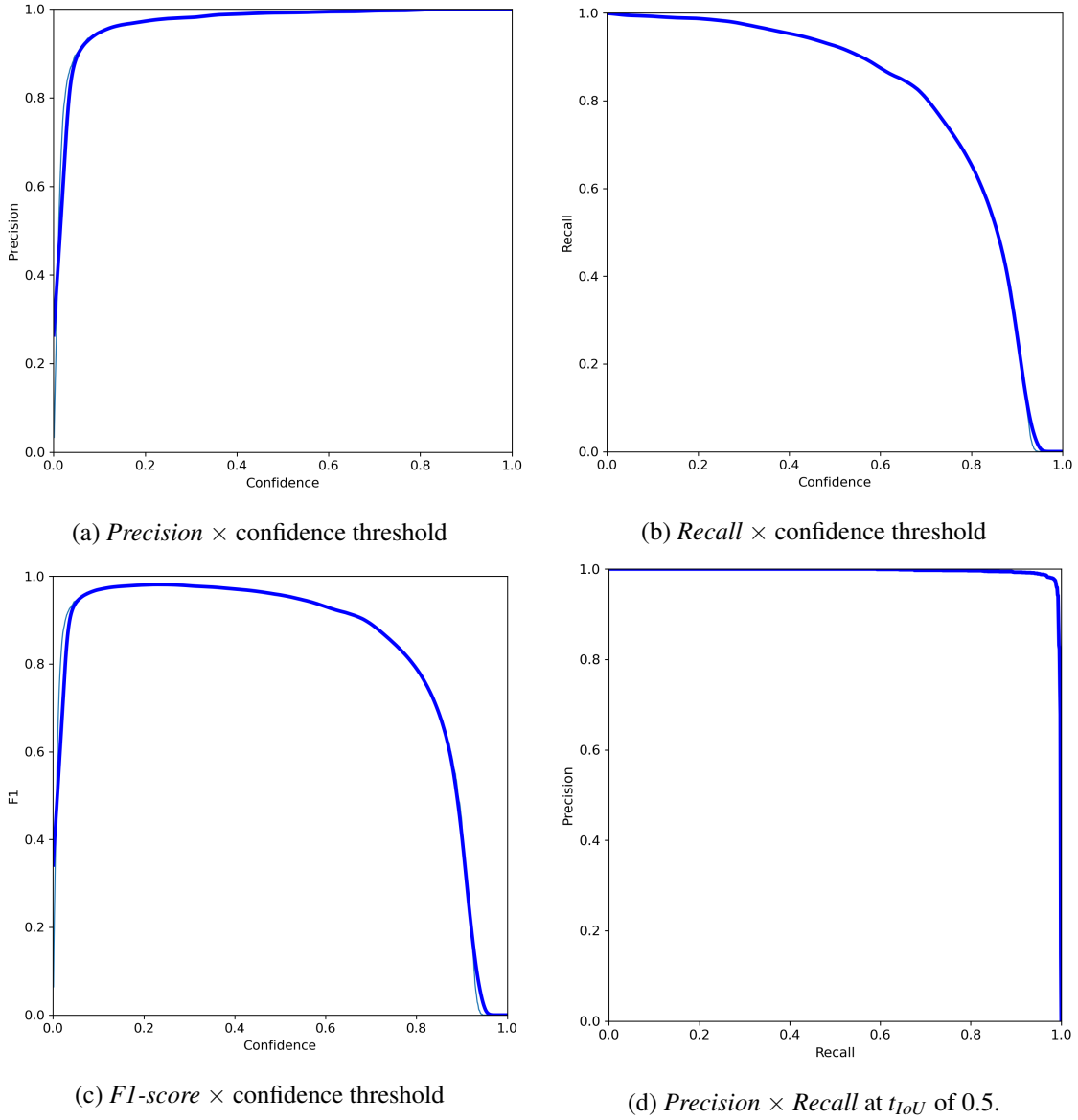


Figure 4.3: Post-training validation metrics.

4.1.4.2 Testing

The model's testing was performed on the real B-scans, surveyed on configurations 1 to 8 (cf. Figure 3.1). Similarly to the classic vision algorithm, the B-scans were pre-processed with a background removal filter and a normalization step, guaranteeing the same testing data for both algorithms.

The model's testing parameters, i.e., IoU threshold and confidence threshold, are manually defined. Magalhães, S. et al [105] state that the optimal confidence threshold is the one that maximizes the *F1-score* of the object detector. From Figure 4.3c, that value is 0.228. The standard IoU threshold amongst the object detector community is 0.5 [105, 68] however, the authors of the YOLOv5 framework suggested the value of 0.45 [6].

Ideally, around one object there should only be one bounding box. The model was trained to detect double hyperbolas, however, on the real B-scans objects are characterized by triple hyperbolas. After a first inference round, it was noticeable that around true hyperbolic signatures, often there were two different bounding boxes: one bounding the top double hyperbola, and one bounding the lower double hyperbola (cf. Figure 4.4a). To overcome this problem, the YOLOv5's predictions were post-processed in such a way that if two predicted bounding boxes had an IoU¹ bigger than 0.2, they were converted into a single prediction box. This box was defined as the smallest box that embraced the original bounding boxes (cf. Figure 4.4b). For simplicity purposes, the post-processing step is going to be addressed as '*IoU merging*'.

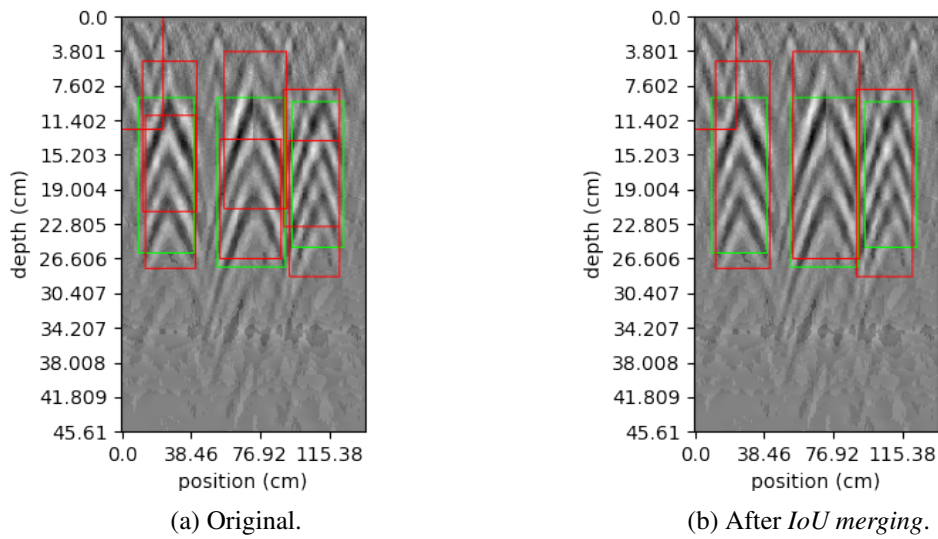


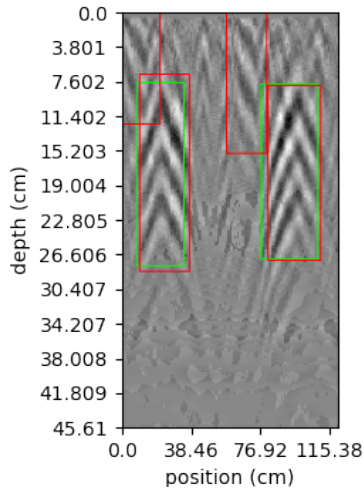
Figure 4.4: Visualization of the *IoU merging* technique. Green boxes are the ground-truth and red boxes are predicted bounding boxes (before and after the merging).

The post-processed predicted bounding boxes, with a confidence threshold of 0.228 and a IoU threshold of 0.45, are visible in Figure 4.5. Table 4.4 presents the prediction's summary for every configuration, using the aforementioned thresholds.

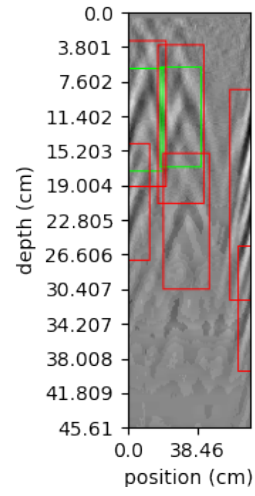
As a result of the low confidence value, the number of false positive predictions is high (24), which results in a *Precision* of 42.86%. On the other hand the *Recall* was 100%, as there were no undetected ground-truth objects (i.e. zero false negatives). The testing *Precision* differs a lot from the validation *Precision* at confidence threshold of 0.228 ($\sim 95\%$, cf. Figure 4.3a), because of the dataset difference. Training and validation were performed on synthetic B-scans, whilst testing was performed on real B-scans. Synthetic B-scans and real B-scans have different styles of artifacts, hence the optimal confidence threshold proposed by Magalhães, S. et al [105], does not perform well in our system.

Through observation of the confidence values of the true positive bounding boxes, the confidence threshold increased to 0.6. The post-processed bounding box predictions are presented

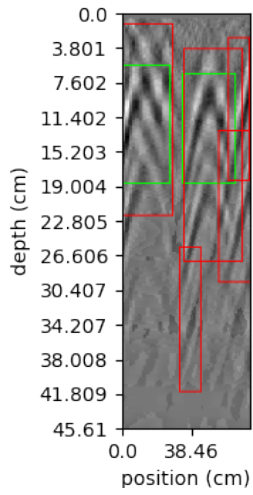
¹In this case, the IoU is computed between the two candidate bounding boxes, and not with the ground-truth boxes.



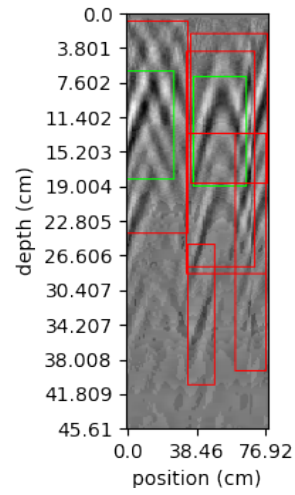
(a) Configuration 1.



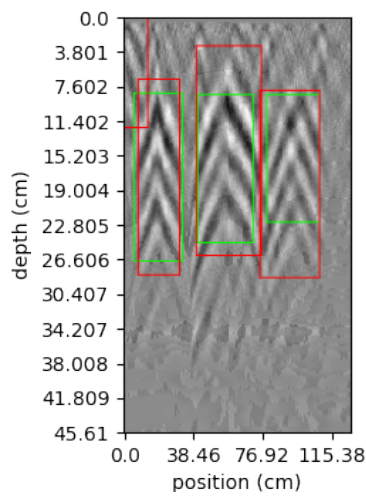
(b) Configuration 2.



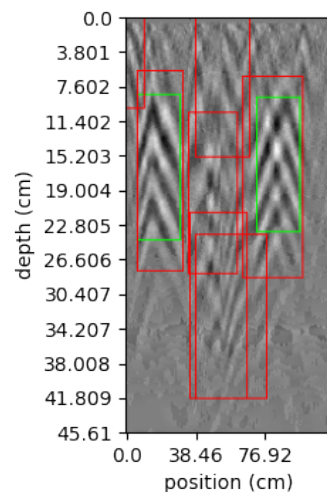
(c) Configuration 3.



(d) Configuration 4.



(e) Configuration 5.



(f) Configuration 6.

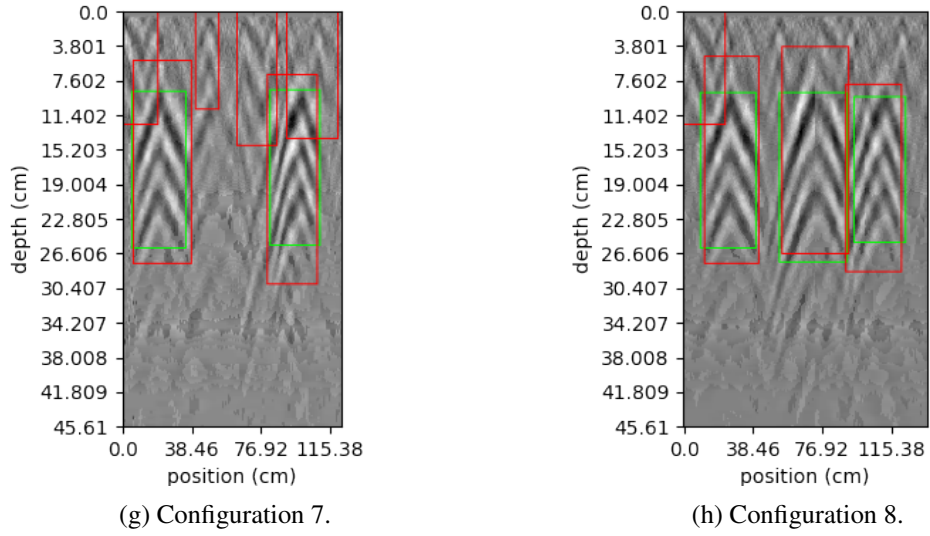


Figure 4.5: Post-processed bounding box predictions with a confidence threshold of 0.228 and a IoU threshold of 0.45. The green boxes represent the ground-truth and the red boxes represent the model's predictions.

Table 4.4: Summarized results of the YOLOv5 Nano, trained on dataset 6, using a confidence threshold of 0.228 and an IoU threshold of 0.45: false negative (**FN**), false positive (**FP**) and true positive (**TP**) bounding box predictions, and depth (d_o) and position (p_o) estimates of the true positive predictions of each configuration (**Conf.**).

Conf.	FN	FP	TP	d_o (cm)	p_o (cm)
1	0	2	2	6.84	23.85
				8.06	96.15
2	0	4	2	3.19	10.77
				3.65	29.61
3	0	3	2	1.22	14.23
				3.95	50.77
4	0	4	2	0.91	17.31
				4.26	52.69
5	0	1	3	6.84	20.00
				3.19	58.85
				8.06	92.69
6	0	5	2	5.93	19.62
				6.54	81.92
7	0	4	2	5.47	22.31
				6.99	94.62
8	0	1	3	5.02	27.31
				3.95	73.85
				8.06	100.62

Table 4.5: Summarized results of the YOLOv5 Nano, trained on dataset 6, using a confidence threshold of 0.6 and an IoU threshold of 0.45: false negative (**FN**), false positive (**FP**) and true positive (**TP**) bounding box predictions, and depth (d_o) and position (p_o) estimates of the true positive predictions of each configuration (**Conf.**).

Conf.	FN	FP	TP	d_o (cm)	p_o (cm)
1	0	1	2	6.84	23.85
				8.06	96.15
2	1	1	1	5.81	10.77
3	0	1	2	1.22	14.23
				3.95	51.54
4	0	1	2	0.91	17.31
				2.28	57.31
5	0	0	3	6.84	20.00
				3.19	58.85
				8.06	92.69
6	0	0	2	5.93	19.62
				6.54	81.92
7	0	1	2	5.47	22.31
				6.99	94.62
8	0	1	3	5.02	27.31
				3.95	73.85
				8.06	100.62

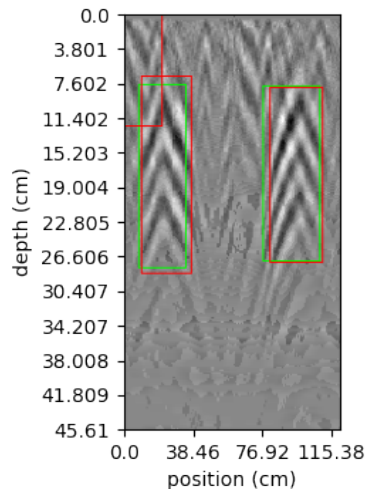
in Figure 4.6. Table 4.5 summarizes the predictions of every configuration, using the renewed confidence threshold.

The model's post-processed predictions present much less false positives (6), but now fail to detect a ground-truth object. This translates to a *Precision* of 73.9% and a *Recall* of 94.4%.

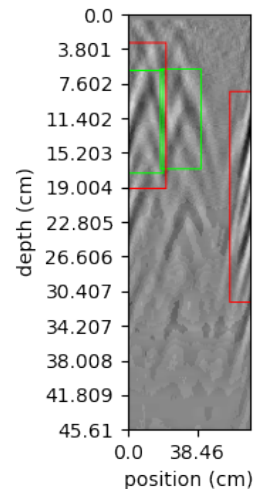
Tables 4.6 and 4.7, present absolute depth and position errors of the true positive predictions, for inferences with a confidence threshold of 0.228 and 0.6, respectively.

For the inferences with a confidence threshold of 0.228 (Table 4.6), the average absolute depth error is 4.55 cm, with a standard deviation of 1.90 cm, and the average absolute position error is 4.81 cm, with a standard deviation of 4.83.

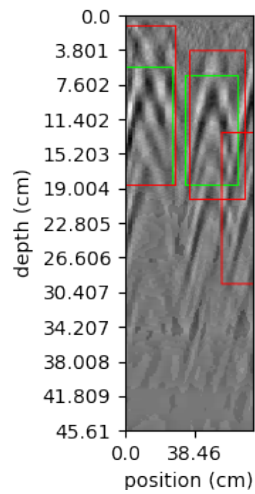
The inferences with a confidence threshold of 0.6 (Table 4.6) presented an average absolute depth error of 4.62 cm, with a standard deviation of 2.02 cm, and an average absolute position error of 4.80 cm, with a standard deviation of 4.94 cm.



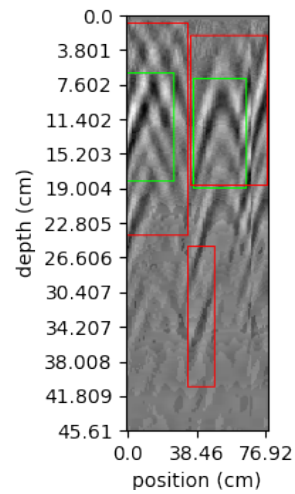
(a) Configuration 1.



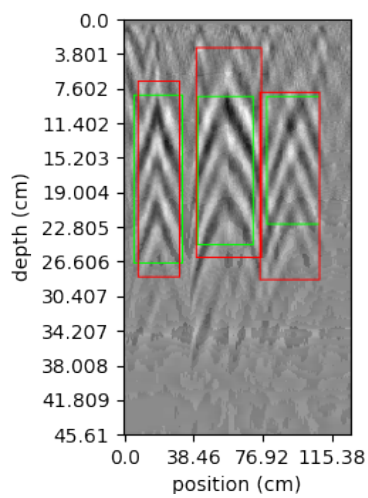
(b) Configuration 2.



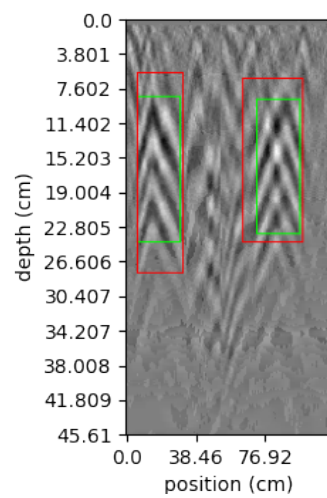
(c) Configuration 3.



(d) Configuration 4.



(e) Configuration 5.



(f) Configuration 6.

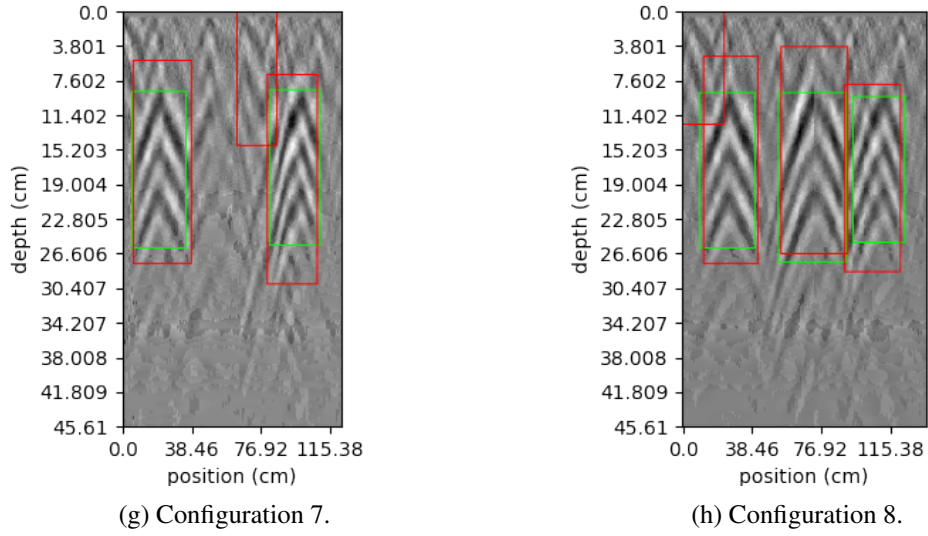


Figure 4.6: Post-processed bounding box predictions with a confidence threshold of 0.6 and an IoU threshold of 0.45. The green boxes represent the ground-truth and the red boxes represent the model's predictions.

Table 4.6: Absolute depth error ($\Delta\epsilon_d$) and absolute position error ($\Delta\epsilon_p$) of true positive bounding box predictions (**TP**), of the inferences with a confidence threshold of 0.228. **Conf.** is the surveyed configuration.

Conf.	TP	$\Delta\epsilon_d$ (cm)	$\Delta\epsilon_p$ (cm)
1	2	3.16	0.6
		1.94	2.1
2	2	5.81	2.98
		5.35	8.86
3	2	7.78	0.48
		5.05	1.73
4	2	8.09	3.56
		4.74	3.04
5	3	3.16	3.25
		6.81	17.35
		1.94	5.56
6	3	4.07	3.63
		3.46	16.33
7	2	4.53	0.94
		3.01	3.63
8	3	4.98	4.06
		6.05	6.1
		1.94	2.37

Table 4.7: Absolute depth error ($\Delta\epsilon_d$) and absolute position error ($\Delta\epsilon_p$) of true positive bounding box predictions (**TP**), of the inferences with a confidence threshold of 0.6. **Conf.** is the surveyed configuration.

Conf.	TP	$\Delta\epsilon_d$ (cm)	$\Delta\epsilon_p$ (cm)
1	2	3.16	0.6
		1.94	2.1
2	1	5.81	2.98
3	2	7.78	0.48
		5.05	0.96
4	2	8.09	3.56
		6.72	7.66
5	3	3.16	3.25
		6.81	17.35
		1.94	5.56
6	3	4.07	3.63
		3.46	16.33
7	2	4.53	0.94
		3.01	3.63
8	3	4.98	4.06
		6.05	6.1
		1.94	2.37

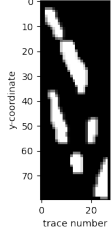
4.1.5 Integration of YOLOv5 Models with Classic Vision Techniques to Detect Hyperbolic Shapes in GPR Data

The YOLOv5 network used in the combined methodologies was the best performing model (YOLOv5 Nano, trained for 20 epochs, on dataset 6). As explained in the previous Section, the predicted bounding boxes that provided the best results had a base confidence of 0.6. Moreover, all bounding box predictions suffered a post-processing step that merged two bounding boxes that had an IoU bigger than 0.2 (*IoU merging*). The final bounding box predictions were presented in Figure 4.6, of Section 4.1.4.2.

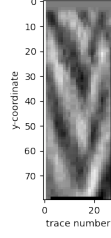
As mentioned in Section 3.7, the processed bounding box predictions were used to crop the original B-scans. The resulting images were fed to the classic vision algorithm for hyperbola isolation and apex detection. The results of the hyperbola isolation (binary masks) and apex detection (predicted apexes) algorithms, applied to the cropped B-scans, are presented in Figure 4.7.

As is visible from the analysis of Figure 4.7, the hyperbola isolation and apex detection applied to the cropped B-scans provides satisfactory results. When the cropped image has an hyperbolic signature, the classic vision algorithm is able to detect its apex accurately. Furthermore, in most cases where only partial signatures or background noise are present in the image, the algorithm does not predict any apexes. The only exception is the predicted apex of bounding box 1 of configuration 7 (Figures 4.7ag and 4.7ah), where a noise-generated inverted hyperbola is detected, inducing a false positive prediction.

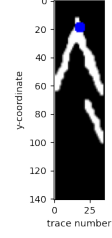
As mentioned in Section 4.1.4.2, the post-processed bounding boxes, predicted by the YOLOv5



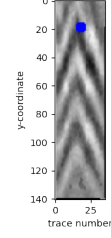
(a) B.Box 1 (Conf. 1).



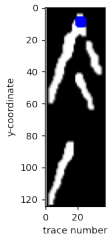
(b) B.Box 1 (Conf. 1).



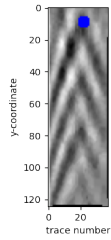
(c) B.Box 2 (Conf. 1).



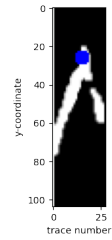
(d) B.Box 2 (Conf. 1).



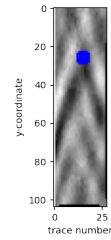
(e) B.Box 3 (Conf. 1).



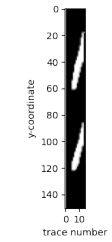
(f) B.Box 3 (Conf. 1).



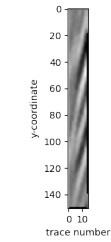
(g) B.Box 1 (Conf. 2).



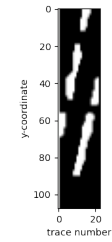
(h) B.Box 1 (Conf. 2).



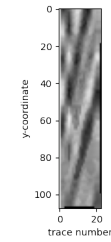
(i) B.Box 2 (Conf. 2).



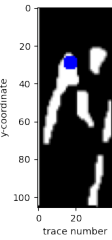
(j) B.Box 2 (Conf. 2).



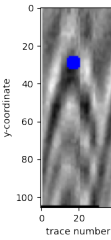
(k) B.Box 1 (Conf. 3).



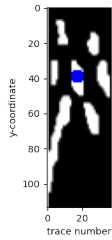
(l) B.Box 1 (Conf. 3).



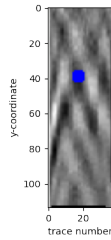
(m) B.Box 2 (Conf. 3).



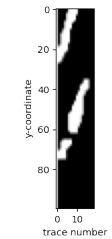
(n) B.Box 2 (Conf. 3).



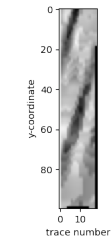
(o) B.Box 3 (Conf. 3).



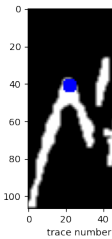
(p) B.Box 3 (Conf. 3).



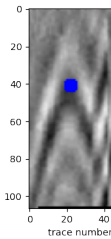
(q) B.Box 1 (Conf. 4).



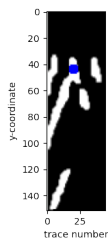
(r) B.Box 1 (Conf. 4).



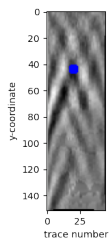
(s) B.Box 2 (Conf. 4).



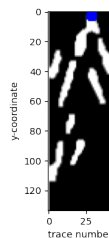
(t) B.Box 2 (Conf. 4).



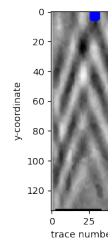
(u) B.Box 3 (Conf. 4).



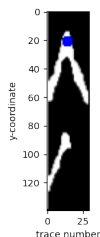
(v) B.Box 3 (Conf. 4).



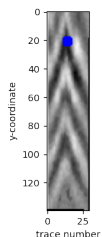
(w) B.Box 1 (Conf. 5).



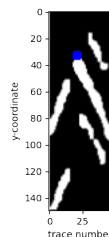
(x) B.Box 1 (Conf. 5).



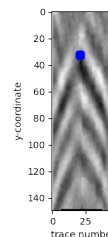
(y) B.Box 2 (Conf. 5).



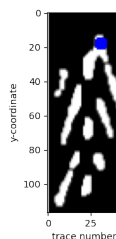
(z) B.Box 2 (Conf. 5).



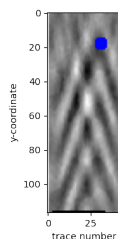
(aa) B.Box 3 (Conf. 5).



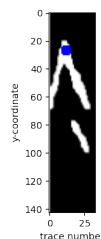
(ab) B.Box 3 (Conf. 5).



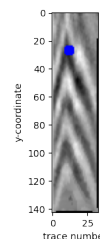
(ac) B.Box 1 (Conf. 6).



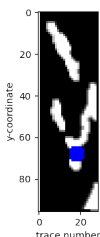
(ad) B.Box 1 (Conf. 6).



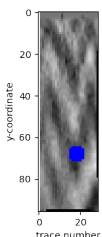
(ae) B.Box 2 (Conf. 6).



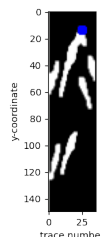
(af) B.Box 2 (Conf. 6).



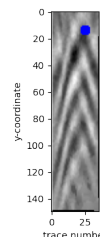
(ag) B.Box 1 (Conf. 7).



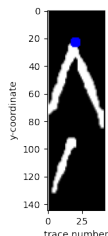
(ah) B.Box 1 (Conf. 7).



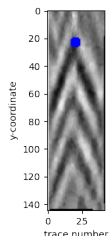
(ai) B.Box 2 (Conf. 7).



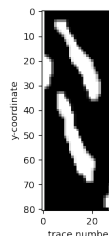
(aj) B.Box 2 (Conf. 7).



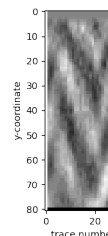
(ak) B.Box 3 (Conf. 7).



(al) B.Box 3 (Conf. 7).



(am) B.Box 1 (Conf. 8).



(an) B.Box 1 (Conf. 8).

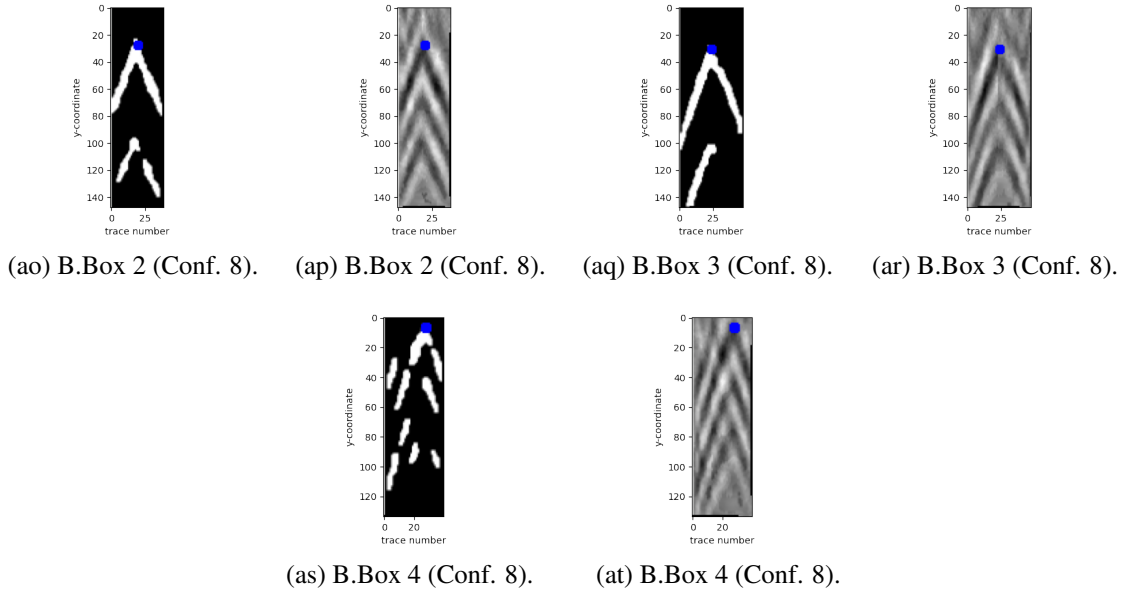


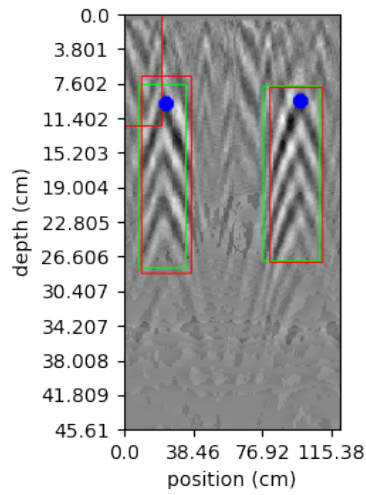
Figure 4.7: Binary masks and cropped B-scans (in pairs) of the bounding boxes, predicted by the YOLOv5 Nano. The blue dots represent the apex predictions.

model, present 17 TP bounding boxes, i.e., objects correctly detected, and 6 FP bounding boxes (background noise). The classic vision methodologies, applied to the B-scans cropped from those 23 bounding boxes, achieve a *Precision* of 94.4% ($TP = 17$, $FP = 1$), and a *Recall* of 100% ($TP = 17$, $FN = 0$).

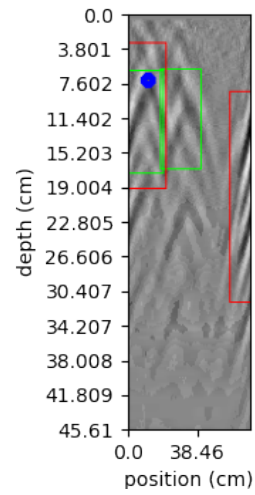
The overall results of the combined methodology are summarized on Table 4.8. Figure 4.8 presents the ground-truth bounding boxes, the post-processed predicted bounding boxes and object's apex predictions.

The combined methodologies achieve an overall *Precision* of 94.4% (1 FP) and *Recall* of 94.4% (1 FN). Table 4.9 presents the absolute depth and position errors for the true positive apex predictions.

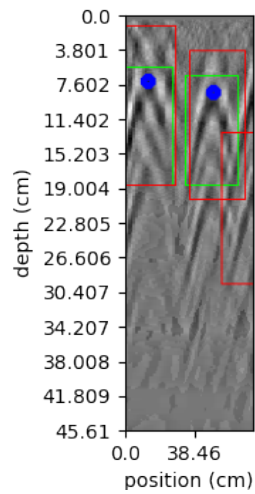
The combined methodologies presented an average absolute depth error of 0.82 cm, with a standard deviation of 0.54 cm, and an average absolute position error of 4.46 cm, with a standard deviation of 4.60 cm.



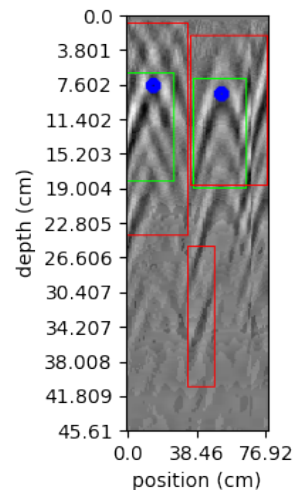
(a) Configuration 1.



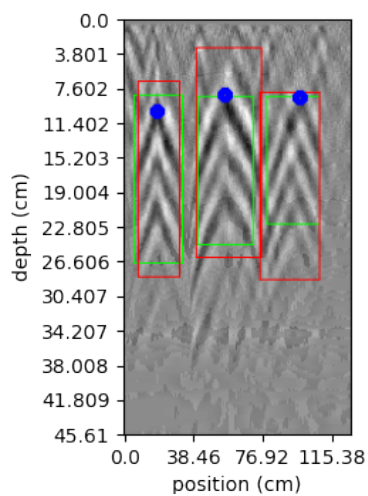
(b) Configuration 2.



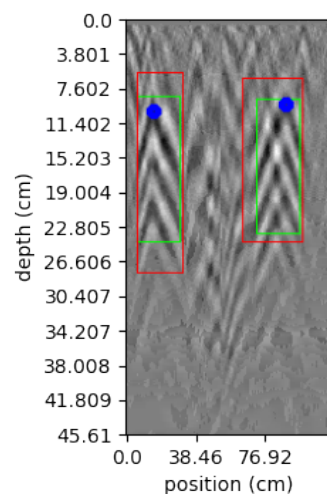
(c) Configuration 3.



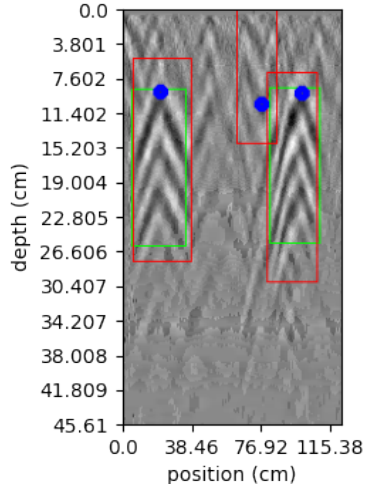
(d) Configuration 4.



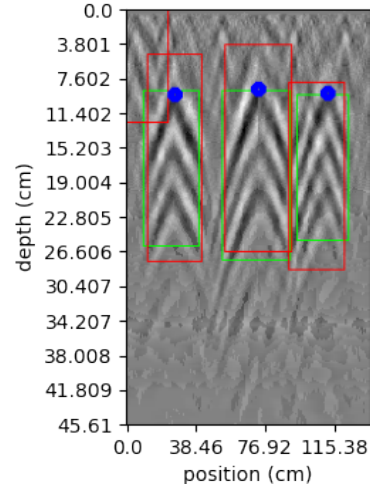
(e) Configuration 5.



(f) Configuration 6.



(g) Configuration 7.



(h) Configuration 8.

Figure 4.8: Apex predictions of the combined methodologies. The green boxes represent the ground-truth, the red boxes represent YOLOv5's predictions and the blue dots represent classic vision's apex predictions.

Table 4.8: Summarized results of the combined methodologies: false negative (**FN**), false positive (**FP**) and true positive (**TP**) apex predictions, and depth (d_o) and position (p_o) estimates of the true positive predictions of each configuration (**Conf.**).

Conf.	FN	FP	TP	d_o (cm)	p_o (cm)
1	0	0	2	9.88	23.85
				9.58	98.46
2	1	0	1	7.30	11.54
3	0	0	2	7.30	13.08
				8.51	49.23
4	0	0	2	7.75	15.38
				8.67	53.08
5	0	0	3	10.19	19.23
				8.36	59.92
				8.67	98.46
6	0	0	2	10.19	16.15
				9.43	89.23
7	0	1	2	9.12	21.54
				9.27	100.00
8	0	0	3	9.43	27.69
				8.82	73.85
				9.27	112.31

Table 4.9: Absolute depth error ($\Delta\epsilon_d$) and absolute position error ($\Delta\epsilon_p$) of the true positive apex predictions (**TP**), of the combined methodologies' predictions. **Conf.** is the surveyed configuration.

Conf.	TP	$\Delta\epsilon_d$ (cm)	$\Delta\epsilon_p$ (cm)
1	2	0.12	0.6
		0.42	0.21
2	1	1.7	2.21
3	2	1.7	0.67
		0.49	3.27
4	2	1.25	1.63
		0.33	3.43
5	3	0.19	4.02
		1.64	15.42
		1.33	0.21
6	2	0.19	7.1
		0.57	9.02
7	2	0.88	1.71
		0.73	1.75
8	3	0.57	4.44
		1.18	6.1
		0.73	14.06

4.2 Discussion

This Section provides a thorough analysis of the results of the proposed methodologies: classic vision, YOLOv5, and the combination of both.

As mentioned before, the YOLOv5 architectures (Nano and Small) were trained in different training/validation datasets (characterized in Table 3.2). Table 4.10 summarizes the performance of all the dataset-architecture pairs in terms of false negatives, false positives, true positives, *Precision*, *Recall* and *F1-score*. The models' bounding box confidence threshold was adjusted based on the predictions' confidence when the confidence threshold was set to the value that maximized the *F1-score* (as suggested by Magalhães, S. et. al [105]).

From the analysis of Table 4.10, it is clear that the employment of the YOLOv5 Small model was an over-complex architectural choice, as it was continuously outperformed by its simpler version, the YOLOv5 Nano. This most likely occurred because the YOLOv5 Small network was overfitting the training data. Overfitting, in this case, arose from two problems: firstly, the objects to be detected were very simple relative to the network's complexity; secondly, the training/validation datasets had a different 'style' (synthetic) from the testing set (real).

The B-scan's dimensions were also a factor that affected the model's performance. Datasets 2 and 4 (the 300×300 B-scan version of datasets 1 and 3, respectively) were constantly outperformed by their 75×300 version. To fulfill the input requirements of the YOLOv5 architectures, all training and testing B-scans had to be resized to 640×640 images, possibly leading to deformed hyperbolic shapes. In the case of square inputs (datasets 2 and 4), the disfiguration was

Table 4.10: Performance assessment of all different training rounds and YOLOv5 models. **Dataset** is the training and validation dataset. **Model** is the YOLOv5 architecture used. **FN**, **FP** and **TP** are the total number of false negatives, false positives and true positives in the testing set. **Pr.**, **Rec.** and **F1** are the dataset-model pair overall *Precision*, *Recall* and *F1-score*.

Dataset	Model	FN	FP	TP	Pr. (%)	Rec. (%)	F1(%)
1	Nano	9	17	9	34.62	47.37	40.00
	Small	18	9	0	0	0	-
2	Nano	16	15	2	11.76	11.11	11.43
	Small	16	39	2	4.88	11.11	6.78
3	Nano	8	11	10	47.62	55.56	51.32
	Small	15	8	3	27.27	16.67	20.69
4	Nano	16	14	2	14.29	11.11	12.50
	Small	16	32	2	5.88	11.11	7.69
5	Nano	2	1	16	94.12	88.89	91.43
	Small	17	1	1	50.00	5.56	10.01
6	Nano	1	6	17	73.91	94.44	82.92
	Small	16	2	2	50.00	5.56	10.01

much smaller than for rectangular inputs. The fact that the testing set (real B-scans) was only composed of rectangular-shaped B-scans, made the YOLOv5 models trained on squared inputs unable to learn how to correctly characterize and estimate real hyperbolic reflections.

The number of different classes in the training set also affects the model's predictions on real data. Datasets 1 and 2 were trained to distinguish 6 classes, datasets 3 and 4 to distinguish 2, and datasets 5 and 6 to distinguish 1 class from the background. Predictions on the testing set were considered correct, i.e., true positive, if any type of bounding box had an IoU of 0.45 (w.r.t. the ground-truth), independently of the predicted class. However, the datasets with more classes resulted in an over-perfectionist model, with very low generalization capability. Figure 4.9 shows an example where the model detected small hyperbolic reflections and ignored the main (TP) hyperbolic signatures. As can be seen from Table 4.10, the lower the number of classes the best the performance of the models on the real B-scans.

Overall, the best performing models were the YOLOv5 Nano networks trained on datasets 5 and 6 (highlighted rows in Table 4.10). These datasets were composed of 75×300 synthetic B-scans, exclusively scattered with double hyperbolas. Dataset 5 was generated with real hyperbolic signatures, whilst dataset 6 also used GAN-generated double hyperbolas.

Although the model trained on dataset 5 has a better F1-score (91.43%) than the one trained on dataset 6 (82.92%), its *Recall*, average absolute depth error and average absolute position error are poorer. The model trained on dataset 5 has an average absolute depth error of 5.31 cm, with a standard deviation of 2.60 cm, as opposed to the model trained on dataset 6 which has an average absolute depth error of 4.62 cm, with a standard deviation of 2.02 cm. Moreover, the first model has an average absolute position error of 4.96 cm, with a standard deviation of 5.39 cm, whilst the latter has an average absolute position error of 4.80 cm, with a standard deviation of 4.94 cm. Lastly, the first model has a *Recall* of 88.89% whilst the latter achieves 94.4%.

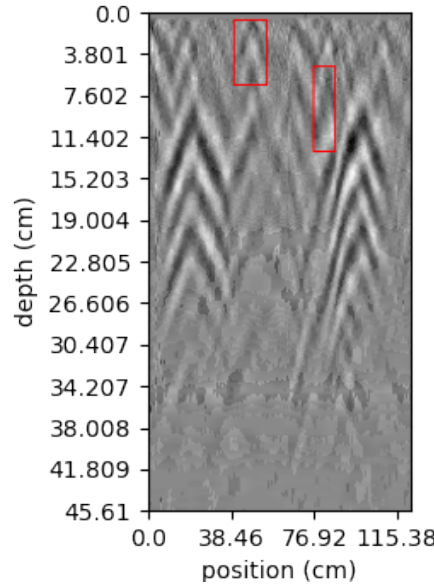


Figure 4.9: Example of 'over-perfectionist' bounding box predictions. The model was a YOLOv5 Nano, trained on dataset 2 (6 classes).

As the final goal of the system is to aid the accurate mapping of the subsurface, the absolute depth and position errors, and *Recall* are the metrics of most importance. The low absolute depth and position errors result in a more accurate estimation of the location of the object. *Recall* represents the percentage of the ground-truth objects that were detected. As so, it was considered that the best performing YOLOv5 model was the one trained on dataset 6, as it delivered better scores in the aforementioned metrics. The *Precision* deficit could always be minimized if an additional validation step was applied to the models' outputs (as was the case for the combined methodologies).

The classic image processing algorithm, applied to the complete B-scans, performed better than the YOLOv5 models: it achieved a *Precision* of 81%, a *Recall* of 94.4%, an absolute depth error (in average) of 1.41 ± 0.72 cm and an absolute position error (in average) of 4.18 ± 4.52 cm.

However, when the same algorithm (with the small adjustments mentioned in Section 3.7) was applied to the B-scans cropped from the YOLOv5's predictions, its performance substantially increased: *Precision* of 94.4% and *Recall* of 100%. This performance improvement results from the fact that the input images only contained local information about the neighbouring area of the predicted object, making the hyperbola isolation and apex detection easier and more accurate.

By combining both methodologies (YOLOv5 and classic vision on cropped B-scans) it was possible to increase the system's overall *Precision* and diminish the average absolute depth estimation error while maintaining the *Recall*. The average absolute position estimate error slightly increased. As mentioned in Section 3.7, the YOLOv5 model used in the combined methodologies system was the YOLOv5 Nano trained on dataset 6. By closely inspecting the architecture of the combined methodologies, it is clear that the overall *Recall* is limited by YOLOv5's predictions

since the classic vision techniques are only applied to the predicted bounding boxes. These work as an extra validation step, i.e., they predict whether or not the inferred bounding boxes contain objects of interest. This validation step leads to a *Precision* improvement of the global system. Furthermore, the fact that the classic vision's inputs are smaller (cropped B-scans) leads to a more accurate apex estimation. The combined methodologies achieved a *Precision* of 94.4%, a *Recall* of 94.4%, an absolute depth error (in average) of 0.82 ± 0.54 cm and an absolute position error (in average) of 4.46 ± 4.6 cm.

The depth errors in the classic vision techniques (applied to cropped or complete B-scans) can be partially justified by the fact that hyperbolas have a width of several centimeters (cf. Figure 4.1), hence the apex predictions were valid for a range of values. In the YOLOv5 models, where the depth estimation was performed based on the predicted bounding boxes, the depth errors mainly appeared because the boxes contained areas above the hyperbola's apex (as was explained in Section 3.8, the depth estimation is based on the y-coordinate of the top left corner of the bounding box). Nevertheless, a share of the absolute depth errors in all architectures may be justified by the velocity estimation technique. As stated in Chapter 3, the relative dielectric permittivity of the medium was considered 1.6, based on a study from Kol, H. [88], which may have led to an incorrect velocity estimation.

The absolute position errors are similar between all methodologies and can be partially explained by the poor, inaccurate calibration system provided by OERAD. On every calibration round, the traces-per-meter (tpm) rate changed, making the conversion from pixels to real-world coordinates, dependent on an average estimate of the traces-per-meter ratio. Moreover, the low measurement accuracy of the experimental setup could also have contributed to a poor ground-truth estimation of the objects' position.

Chapter 5

Conclusions and Future Work

5.1 Final conclusions

The first goal of this dissertation was to develop a classic vision/image processing algorithm to detect hyperbolic reflections in GPR data, that is, to detect object's reflection. The set of applied vision methodologies resulted in an overall *Recall* of 94.44% and *Precision* of 81%. Moreover, the depth and position errors were 1.41 ± 0.72 cm and 4.18 ± 4.52 cm, respectively. Although these results could be improved by better calibration systems (e.g. depth-to-known reflector velocity estimation, encoder improvement, etc.) they prove the viability of this type of algorithm in the detection of objects in GPR data.

The second objective of this dissertation was to train and test YOLOv5 models to detect hyperbolic traces, i.e., objects in GPR B-scans. The results of Chapter 4 show that a state-of-the-art neural network approach to this detection task is able to achieve great *Recall* (94.4%) and *Precision* (73.9%) values. Furthermore, the depth and position estimates presented average errors in the order of 4.62 ± 2.02 cm, in depth, and 4.80 ± 4.94 cm, in position (results for 8 surveys collected with the OERAD's *Concretto* GPR, operating at 2 GHz, moved over ~ 1 meter survey lines, with metallic objects at depths of ~ 10 cm).

To improve the initial results of both methodologies, the best performing YOLOv5 network was linked to an adapted version of the classic vision algorithm. YOLOv5's predictions were post-validated by the renewed version of the classic algorithm, allowing a detection of GPR objects with *Precision* of 94.4% and *Recall* of 94.4%. Moreover, the objects' depth and position estimates presented errors in the order of 0.82 ± 0.54 cm, in depth, and 4.46 ± 4.6 cm, in position.

To overcome the low dataset size, and still successfully train the neural network models, data augmentation techniques were applied to the collected B-scans. Image level data augmentation techniques and GAN-generated hyperbolic reflections were the data augmentation methodologies that greatly improved the model's performance, as is supported by the results of Chapter 4.

Finally, the ultimate goal of the dissertation was to compare the performance of both models. Surprisingly, the classic-based algorithm outperforms the state-of-the-art deep learning models. Most likely, this fact is justified by dataset limitations, that did not allow the optimal training of

the YOLOv5 models. A larger dataset would probably improve the results from this approach. Nevertheless, the combination of both methodologies provided a outstanding improvement over the individual performance of the developed/implemented architectures.

5.2 Future Work

The developed systems leave room for future work, as they could be further improved to achieve better performance.

Naturally, adding more surveys to the dataset would be beneficial to the neural network approach. Furthermore, capturing longer surveys would allow the assessment of the system's real-time capabilities. All developed architectures are equipped with cyclic buffers on the *B-scan Generator* block, but due to the data limitations these features could not be properly tested.

Creating an ensemble neural network model composed of a YOLOv5 model and simple classifier could improve the performance of the CNN-based methodologies. The classifier could analyze the image area bounded by the predicted bounding boxes, and determine whether the area contained a real object or not. The classifier network only had to learn how to distinguish GPR-hyperbolic signatures from other types of artifacts. With this implementation, the object detector network should be trained to achieve worse *Precision* and better *Recall*, as the ensemble's maximum *Precision* and *Recall* are determined by the classifier network and by the object detector, respectively. The ensemble could then be connected to the adapted classic vision algorithm for apex detection.

The results of the YOLOv5-based architecture could be improved if the hyperparameters of the models were slightly altered, and new configurations were tested. Another improvement to the CNN-based architectures would be to simplify the YOLOv5 backbone. It was noticeable from the results of Chapter 4, that the networks were over-complex for the task at hand. By reducing the complexity of the backbone, the YOLOv5 networks could gain more ability to generalize. Moreover, the YOLOv5 network could be replaced by its renewed version: the YOLOv6¹. This version is faster and more accurate than the YOLOv5. Finally, using transfer learning from models trained on datasets closer, in simplicity and imaging, to the test set, could further increase the CNN's performance.

The synthetic B-scan generation could also be improved. Taking the real backgrounds, one could generate hyperbolic signatures from scratch, by sampling random points in the background and drawing hyperbolic signatures with different intensities, apertures, and lengths. Low-intensity signatures could be scattered among the background to further increase the similarity between the real and synthesized backgrounds. These refinements could improve the resemblance of synthetic and real B-scans to a point where they become almost indistinguishable. Another option would be to collect both darker and lighter parts of the real hyperbolic reflections and synthesize the B-scans with those signatures, which would be closer to reality.

¹The YOLOv6 was launched in July of 2022 in <https://github.com/meituan/YOLOv6>.

The classic image processing algorithm could also be improved by assuring that the thresholding and connected component filtering was performed inside a sliding window (as the line detection is, cf. Section 3.4), leading to an enhanced optimal threshold estimation. Following the same train of thought, a vertical sliding window could be added to the B-scan processing. The implemented sliding window (described in Section 3.4) has one sliding dimension, if a second sliding dimension was employed, weaker reflections could be detected (at deeper depths) and the inter-object interference would be greatly minimized (e.g. when defining the optimal threshold).

Finally, as mentioned before, one major refinement could be made to reduce the position and depth estimation errors: a correct velocity estimation and correct calibration of the traces-per-meter ratio. If not completely, the estimation errors would greatly decrease, to levels that in the final system (rePLANt's autonomous sapling planting robot), could be easily negligible.

References

- [1] AP Annan. Electromagnetic principles of ground penetrating radar. *Ground penetrating radar: theory and applications*, 1:1–37, 2009.
- [2] Dominik Merkle, Carsten Frey, and Alexander Reiterer. Fusion of ground penetrating radar and laser scanning for infrastructure mapping. *Journal of Applied Geodesy*, 15:31 – 45, 2020.
- [3] Craig Warren. *Numerical modelling of high-frequency ground-penetrating radar antennas*. PhD thesis, University of Edinburgh, 2009.
- [4] Shubhashree Savant. A review on edge detection techniques for image segmentation. *International Journal of Computer Science and Information Technologies*, 5(4):5898–5900, 2014.
- [5] Haldo Spontón and Juan Cardelino. A review of classic edge detectors. *Image Processing On Line*, 5:90–123, 2015.
- [6] ultralytics. YOLOv5. <https://github.com/ultralytics/yolov5> in 2022-06-21.
- [7] Jacob Solawetz. YOLOv5 new version - improvements and evaluation, 2020. <https://blog.roboflow.com/yolov5-improvements-and-evaluation/> in 2022-06-21.
- [8] Hao Liang, Linyin Xing, and Jianhui Lin. Application and algorithm of ground-penetrating radar for plant root detection: a review. *Sensors*, 20(10):2836, 2020.
- [9] United States Environmental Protection Agency. Ground penetrating radar. <https://clu-in.org/characterization/technologies/gpr.cfm> in 2022-03-07.
- [10] Christian Wolff. Ground penetrating radar. <https://www.radartutorial.eu/02.basics/Ground%20penetrating%20radar.en.html> in 2022-03-07.
- [11] How does a ground penetrating radar (gpr) work? <https://theconstructor.org/practical-guide/ndt/how-ground-penetrating-radar-gpr-work/43370/> in 2022-03-07.
- [12] Steven Koppenjan. Ground penetrating radar systems and design. *Ground penetrating radar: Theory and applications*, 1, 2009.
- [13] A Benedetto and F Benedetto. Application field-specific synthesizing of sensing technology: Civil engineering application of ground-penetrating radar sensing technology. 2014.

- [14] Manuelle Santos Góis Paixão. *Análise da acurácia das estimativas de posicionamento do nível freático e dos teores de umidade do solo com o emprego dos métodos de sísmica de refração rasa e georadar a partir de um estudo no campus da USP, São Paulo/SP*. PhD thesis, Universidade de São Paulo, 2005.
- [15] UBC EOS. Ground penetrating radar: concepts and examples, October 2006. <https://www.eoas.ubc.ca/courses/eosc350/content/methods/gpr-06.htm> in 2022-03-07.
- [16] Emanuele Forte and Michele Pipan. Review of multi-offset gpr applications: Data acquisition, processing and analysis. *Signal Processing*, 132:210–220, 2017. doi:10.1016/j.sigpro.2016.04.011.
- [17] N Smitha, DR Ullas Bharadwaj, S Abilash, SN Sridhara, and Vipula Singh. Kirchhoff and fk migration to focus ground penetrating radar images. *International Journal of Geo-Engineering*, 7(1):1–12, 2016.
- [18] Caner Özdemir, Şevket Demirci, Enes Yiğit, and Betül Yilmaz. A review on migration methods in b-scan ground penetrating radar imaging. *Mathematical Problems in Engineering*, 2014, 2014.
- [19] Fei Xie, Chivas Gi-Wah Wu, Wallace Wai-Lok Lai, and Janet Fung-Chu Sham. Correction of multi-frequency gpr wave velocity with distorted hyperbolic reflections from gpr surveys of underground utilities. *Tunnelling and Underground Space Technology*, 76:76–91, 2018. doi:10.1016/j.tust.2018.02.005.
- [20] Janet F.C. Sham and Wallace W.L. Lai. Development of a new algorithm for accurate estimation of gpr's wave propagation velocity by common-offset survey method. *NDT E International*, 83:104–113, 2016. URL: <https://www.sciencedirect.com/science/article/pii/S0963869516300391>, doi:<https://doi.org/10.1016/j.ndteint.2016.05.002>.
- [21] Toko Tanikawa, Yasuhiro Hirano, Masako Dannoura, Keitarou Yamase, Kenji Aono, Masahiro Ishii, Tetsuro Igarashi, Hidetoshi Ikeno, and Yoichi Kanazawa. Root orientation can affect detection accuracy of ground-penetrating radar. *Plant and Soil*, 373(1-2):317–327, 2013. doi:10.1007/s11104-013-1798-6.
- [22] What is an air reflection in gpr data?, December 2021. <https://www.guidelinegeo.com/help-articles/what-is-an-air-reflection-in-gpr-data/> in 2022-06-07.
- [23] James A Doolittle and John R Butnor. Soils, peatlands, and biomonitoring. *Ground penetrating radar: theory and applications*, 524, 2009.
- [24] A. Klotzsche, F. Jonard, M.C. Looms, J. van der Kruk, and J.A. Huisman. Measuring soil water content with ground penetrating radar: A decade of progress. *Vadose Zone Journal*, 17(1):180052, 2018. URL: <https://acsess.onlinelibrary.wiley.com/doi/abs/10.2136/vzj2018.03.0052>, arXiv:<https://acsess.onlinelibrary.wiley.com/doi/pdf/10.2136/vzj2018.03.0052>, doi:<https://doi.org/10.2136/vzj2018.03.0052>.
- [25] Nigel J Cassidy and HM Jol. Ground penetrating radar data processing, modelling and analysis. *Ground penetrating radar: theory and applications*, pages 141–176, 2009.

- [26] Dang Hoai Trung, Nguyen Van Giang, and Nguyen Thanh Van. The application of depth migration for processing gpr data. In *E3S Web of Conferences*, volume 35, page 03004. EDP Sciences, 2018.
- [27] Andrea Benedetto, Fabio Tosti, Luca Bianchini Ciampoli, and Fabrizio D’amico. An overview of ground-penetrating radar signal processing techniques for road inspections. *Signal processing*, 132:201–209, 2017.
- [28] Xihong Cui, Li Guo, Jin Chen, Xuehong Chen, and Xiaolin Zhu. Estimating tree-root biomass in different depths using ground-penetrating radar: Evidence from a controlled experiment. *IEEE Transactions on Geoscience and Remote Sensing*, 51(6):3410–3423, 2013. doi:10.1109/tgrs.2012.2224351.
- [29] Li Guo, Yuan Wu, Jin Chen, Yasuhiro Hirano, Toko Tanikawa, Wentao Li, and Xihong Cui. Calibrating the impact of root orientation on root quantification using ground-penetrating radar. *Plant and Soil*, 395(1-2):289–305, 2015. doi:10.1007/s11104-015-2563-9.
- [30] Jean-Rémi Dujardin and Maksim Bano. Topographic migration of gpr data: Examples from chad and mongolia. *Comptes Rendus Geoscience*, 345(2):73–80, 2013. URL: <https://www.sciencedirect.com/science/article/pii/S1631071313000278>, doi:<https://doi.org/10.1016/j.crte.2013.01.003>.
- [31] Sebastian Kowalczyk, Anna Lejzerowicz, and Beata Kowalczyk. Groundwater table level changes based on ground penetrating radar images: A case study. In *2018 17th International Conference on Ground Penetrating Radar (GPR)*, pages 1–4. IEEE, 2018.
- [32] James St Clair and W Steven Holbrook. Measuring snow water equivalent from common-offset gpr records through migration velocity analysis. *The Cryosphere*, 11(6):2997–3009, 2017.
- [33] Hai Liu, Xiongyao Xie, and Motoyuki Sato. Accurate thickness estimation of a backfill grouting layer behind shield tunnel lining by cmp measurement using gpr. In *2012 14th International Conference on Ground Penetrating Radar (GPR)*, pages 137–142. IEEE, 2012.
- [34] Luigia Nuzzo. Coherent noise attenuation in gpr data by linear and parabolic radon transform techniques. *Annals of Geophysics*, 46(3), 2003.
- [35] Xiuwei Liu, Xuejun Dong, Qingwu Xue, Daniel I Leskovar, John Jifon, John R Butnor, and Thomas Marek. Ground penetrating radar (gpr) detects fine roots of agricultural crops in the field. *Plant and Soil*, 423(1):517–531, 2018.
- [36] Wenke Zhao, Emanuele Forte, Michele Pipan, and Gang Tian. Ground penetrating radar (gpr) attribute analysis for archaeological prospection. *Journal of Applied Geophysics*, 97:107–117, 2013. Ground Penetrating Radar. URL: <https://www.sciencedirect.com/science/article/pii/S0926985113000906>, doi:<https://doi.org/10.1016/j.jappgeo.2013.04.010>.
- [37] Jacob R. Turner, Roy S. Stine, and Linda F. Stine. A comparison of ground-penetrating radar, magnetic gradiometer and electromagnetic induction survey techniques at house in the horseshoe state historic site. *Journal of Archaeological Science: Reports*, 20:33–46, 2018. doi:10.1016/j.jasrep.2018.04.005.

- [38] Yuichi Nakashima, Hui Zhou, and Motoyuki Sato. Estimation of groundwater level by gpr in an area with multiple ambiguous reflections. *Journal of applied geophysics*, 47(3-4):241–249, 2001.
- [39] Yang Jin and Yunling Duan. A new method for abnormal underground rocks identification using ground penetrating radar. *Measurement*, 149:106988, 2020. URL: <https://www.sciencedirect.com/science/article/pii/S0263224119308541>, doi:<https://doi.org/10.1016/j.measurement.2019.106988>.
- [40] E. Forte, M. Dossi, M. Pipan, and R. R. Colucci. Velocity analysis from common offset GPR data inversion: theory and application to synthetic and real data. *Geophysical Journal International*, 197(3):1471–1483, 04 2014. URL: <https://doi.org/10.1093/gji/ggu103>, arXiv:<https://academic.oup.com/gji/article-pdf/197/3/1471/9265224/ggu103.pdf>, doi:10.1093/gji/ggu103.
- [41] Emerson R. Almeida, Jorge L. Porsani, Adam Booth, Alexandre T. Brunello, and Tiina Sakinen. Analysis of gpr field parameters for root mapping in brazil’s caatinga environment. In *2018 17th International Conference on Ground Penetrating Radar (GPR)*. IEEE. doi:10.1109/icgpr.2018.8441547.
- [42] Muhammed Duman and Ali Gurbuz. 3d imaging for ground-penetrating radars via dictionary dimension reduction. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING COMPUTER SCIENCES*, 23:1242–1256, 01 2015. doi:10.3906/elk-1212-157.
- [43] Gao Peng, Wang Ruiyan, Zhao Gengxing, and Li Yuhuan. The application of gpr to the detection of soil wetted bodies formed by drip irrigation. *PLOS ONE*, 15(7):1–15, 07 2020. URL: <https://doi.org/10.1371/journal.pone.0235489>, doi:10.1371/journal.pone.0235489.
- [44] Eric Sucre, John Tuttle, and Thomas Fox. The use of ground-penetrating radar to accurately estimate soil depth in rocky forest soils. *Forest Science*, 57:59–66, 02 2011.
- [45] Christofer Jonason. Automatic velocity estimation in gpr data and migration, 2020.
- [46] William W Symes. Migration velocity analysis and waveform inversion. *Geophysical prospecting*, 56(6):765–790, 2008.
- [47] Migration velocity analysis (mva). <http://sepwww.stanford.edu/sep/biondo/HTML/VelAn.html> in 2022-03-11.
- [48] Niklas Allroggen, Jens Tronicke, Marcel Delock, and Urs Böniger. Topographic migration of 2d and 3d ground-penetrating radar data considering variable velocities. *Near Surface Geophysics*, 13(3):253–259, 2015.
- [49] Lorenzo H., Pérez-Gracia V., Novo A., and Armesto J. Forestry applications of ground-penetrating radar. *Forest Systems*, 19(1):5–17, Apr. 2010. URL: <https://revistas.inia.es/index.php/fs/article/view/1163>, doi:10.5424/fs/2010191-01163.
- [50] Cmp weighted stacking, January 2017. https://reproducibility.org/RSF/book/uh/avostack/paper_html/node3.html in 2022-03-11.

- [51] Jiss Kuruvilla, Dhanya Sukumaran, Anjali Sankar, and Siji P Joy. A review on image processing and image segmentation. In *2016 international conference on data mining and advanced computing (SAPIENCE)*, pages 198–203. IEEE, 2016.
- [52] HP Narkhede. Review of image segmentation techniques. *International Journal of Science and Modern Engineering*, 1(8):54–61, 2013.
- [53] Nikhil R Pal and Sankar K Pal. A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294, 1993.
- [54] A Krishnaveni, R Shankar, and S Duraisamy. A review on various image thresholding methods for mammogram image segmentation. *Compliance Engineering Journal ISSN*, (0898-3577), 2020.
- [55] Jianbang Xiao, Qing Gao, Yuchen Ling, Jingnan Yan, and Bingzhang Liu. Research on hyperbola detection and fitting in gpr b-scan image. In *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pages 266–270. IEEE, 2021.
- [56] Wentai Lei, Feifei Hou, Jingchun Xi, Qianying Tan, Mengdi Xu, Xinyue Jiang, Gengye Liu, and Qingyuan Gu. Automatic hyperbola detection and fitting in gpr b-scan image. *Automation in Construction*, 106:102839, 2019.
- [57] Qingxu Dou, Lijun Wei, Derek R Magee, and Anthony G Cohn. Real-time hyperbola recognition and fitting in gpr data. *IEEE Transactions on Geoscience and Remote Sensing*, 55(1):51–62, 2016.
- [58] Aixa M Rivera-Rios and Elsa Leticia Flores-Marquez. Image-radargram analysis based on generalized hough transform: experimental cases. *Journal of Geophysics and Engineering*, 9(5):558–568, 2012.
- [59] Jose de Jesus Guerrero-Turrubiates, Ivan Cruz-Aceves, Sergio Ledesma, Juan Manuel Sierra-Hernandez, Jonas Velasco, Juan Gabriel Avina-Cervantes, Maria Susana Avila-Garcia, Horacio Rostro-Gonzalez, and Roberto Rojas-Laguna. Fast parabola detection using estimation of distribution algorithms. *Computational and Mathematical Methods in Medicine*, 2017, 2017.
- [60] Weibin Rong, Zhanjing Li, Wei Zhang, and Lining Sun. An improved canny edge detection algorithm. In *2014 IEEE International Conference on Mechatronics and Automation*, pages 577–582, 2014. doi:10.1109/ICMA.2014.6885761.
- [61] Mohammad Zubir Mat Jafri and Farzin Deravi. Efficient algorithm for the detection of parabolic curves. In *Vision Geometry III*, volume 2356, pages 53–61. SPIE, 1995.
- [62] Jimut Bahan Pal. Traditional lines and circles detector. 2020.
- [63] Danko Antolovic. Review of the hough transform method, with an implementation of the fast hough variant for line detection. *Department of Computer Science, Indiana University*, pages 1932–4545, 2008.
- [64] Payam Rahmdel, Richard A Comley, Daming Shi, and Siobhan McElduff. A review of hough transform and line segment detection approaches. 2015.

- [65] Somet Lee. Lines detection with hough transform, 2020. <https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549> in 2022-06-21.
- [66] Houda Harkat, Youssef Elfakir, Saad Dosse Bennani, Ghizlane Khaissidi, and Mostafa Mrabti. Ground penetrating radar hyperbola detection using scale-invariant feature transform. In *2016 International Conference on Electrical and Information Technologies (ICEIT)*, pages 392–397. IEEE, 2016.
- [67] Jian Wang and Yi Su. Fast detection of gpr objects with cross correlation and hough transform. *Progress In Electromagnetics Research C*, 38:229–239, 2013.
- [68] José Duarte Penetro Mesquita Ferreira. Cross-sensor face detection. 2021.
- [69] Umut Özkaya, Şaban Öztürk, Farid Melgani, and Levent Seyfi. Residual cnn + bi-lstm model to analyze gpr b scan images. *Automation in Construction*, 123:103525, 2021. URL: <https://www.sciencedirect.com/science/article/pii/S0926580520311055>, doi:<https://doi.org/10.1016/j.autcon.2020.103525>.
- [70] Jiabin Luo, Wentai Lei, Feifei Hou, Chenghao Wang, Qiang Ren, Shuo Zhang, Shiguang Luo, Yiwei Wang, and Long Xu. Gpr b-scan image denoising via multi-scale convolutional autoencoder with data augmentation. *Electronics*, 10(11):1269, 2021.
- [71] Xin Zhang, Liangxiu Han, Mark Robinson, and Anthony Gallagher. A gans-based deep learning framework for automatic subsurface object recognition from ground penetrating radar data. *IEEE Access*, 9:39009–39018, 2021.
- [72] Bin Liu, Yuxiao Ren, Hanchi Liu, Hui Xu, Zhengfang Wang, Anthony G Cohn, and Peng Jiang. Gprinvnet: Deep learning-based ground-penetrating radar data inversion for tunnel linings. *IEEE Transactions on Geoscience and Remote Sensing*, 59(10):8305–8325, 2021.
- [73] Haifeng Li, Nansha Li, Renbiao Wu, Huaichao Wang, Zhongcheng Gui, and Dezhen Song. Gpr-rcnn: an algorithm of subsurface defect detection for airport runway based on gpr. *IEEE Robotics and Automation Letters*, 6(2):3001–3008, 2021.
- [74] Jing Wang, Hanchi Liu, Peng Jiang, Zhengfang Wang, Qingmei Sui, and Fengkai Zhang. Gpri2net: A deep-neural-network-based ground penetrating radar data inversion and object identification framework for consecutive and long survey lines. *IEEE Transactions on Geoscience and Remote Sensing*, 2021.
- [75] Yintao Ji, Fengkai Zhang, Jing Wang, Zhengfang Wang, Peng Jiang, Hanchi Liu, and Qingmei Sui. Deep neural network-based permittivity inversions for ground penetrating radar data. *IEEE Sensors Journal*, 21(6):8172–8183, 2021.
- [76] Hui Qin, Donghao Zhang, Yu Tang, and Yuanzheng Wang. Automatic recognition of tunnel lining elements from gpr images using deep convolutional networks with data augmentation. *Automation in Construction*, 130:103830, 2021.
- [77] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

- [78] Rohith Gandhi. R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms, 2018. <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365> in 2022-06-21.
- [79] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [80] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [81] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.
- [82] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [83] Valentina Alto. Training, validation, and test set in machine learning, 2020. <https://medium.com/analytics-vidhya/training-validation-and-test-set-in-machine-learning-7fab555c1080> in 2022-07-01.
- [84] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.
- [85] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [86] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [87] Yunpeng Yue, Hai Liu, Xu Meng, Yinguang Li, and Yanliang Du. Generation of high-precision ground penetrating radar images using improved least square generative adversarial networks. *Remote Sensing*, 13(22):4590, 2021.
- [88] Hamiyet Şahin Kol. Thermal and dielectric properties of pine wood in the transverse direction. *BioResources*, 4(4):1663–1669, 2009.
- [89] TCP vs. UDP: What’s the difference?, October 2017. <https://www.lifesize.com/en/blog/tcp-vs-udp/> in 2022-06-21.
- [90] Python vs R: Know the difference, November 2021. <https://www.interviewbit.com/blog/python-vs-r/> in 2022-06-17.
- [91] Kun Liang, Yong Ma, Yue Xie, Bo Zhou, and Rui Wang. A new adaptive contrast enhancement algorithm for infrared images based on double plateaus histogram equalization. *Infrared Physics & Technology*, 55(4):309–315, 2012.

- [92] R Fisher, S Perkins, A Walker, and E Wolfart. Connected components labeling, 2003. <https://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm> in 2022-06-21.
- [93] R Fisher, S Perkins, A Walker, and E Wolfart. Skeletonization, 2003. <https://homepages.inf.ed.ac.uk/rbf/HIPR2/skeleton.htm> in 2022-06-21.
- [94] Jonathan Hui. Gan - wassertein gan wgan-gp, 2018. <https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490> in 2022-06-21.
- [95] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.
- [96] Juha Heinonen. Lipschitz functions. In *Lectures on Analysis on Metric Spaces*, pages 43–48. Springer, 2001.
- [97] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.
- [98] Lornatang. Wsgan-gp. https://github.com/Lornatang/WassersteinGAN_GP-PyTorch#wasserstein-gan in 2022-06-21.
- [99] Alpha channel, July 2012. <https://www.techopedia.com/definition/1945/alpha-channel> in 2022-06-21.
- [100] Alenrex Maity, Anshuman Pattanaik, Santwana Sagnika, and Santosh Pani. A comparative study on approaches to speckle noise reduction in images. In *2015 International Conference on Computational Intelligence and Networks*, pages 148–155, 2015. doi:10.1109/CINE.2015.36.
- [101] Do Thuan. Evolution of yolo algorithm and yolov5: The state-of-the-art object detection algorithm. 2021.
- [102] Zexuan Guo, Chensheng Wang, Guang Yang, Zeyuan Huang, and Guo Li. Msft-yolo: Improved yolov5 based on transformer for detecting defects of steel surface. *Sensors*, 22(9):3467, 2022.
- [103] Jonathan Hui. Real-time object detection with yolo, yolov2 and now yolov3, 2018. <https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088> in 2022-06-21.
- [104] Lihi Gur Arie. The practical guide for object detection with yolov5 algorithm, 2022. <https://towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843> in 2022-06-21.
- [105] Sandro Augusto Magalhães, Luís Castro, Germano Moreira, Filipe Neves Dos Santos, Mário Cunha, Jorge Dias, and António Paulo Moreira. Evaluating the single-shot multi-box detector and yolo deep learning models for the detection of tomatoes in a greenhouse. *Sensors*, 21(10):3569, 2021.