University of Louisville
ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

12-2018

Landmine detection using semi-supervised learning.

Graham Reid
University of Louisville

Follow this and additional works at: https://ir.library.louisville.edu/etd

 Part of the Computational Engineering Commons

Recommended Citation

Reid, Graham, "Landmine detection using semi-supervised learning." (2018). *Electronic Theses and Dissertations.* Paper 3132.
https://doi.org/10.18297/etd/3132

This Master's Thesis is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

LANDMINE DETECTION USING SEMI-SUPERVISED LEARNING

By

Graham Reid
B.S., Computer Engineering and Computer Science, University of Louisville, 2014

A Thesis Submitted to the Faculty of the JB Speed School of Engineering of the University of Louisville in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computer Science

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

December 2018

Copyright 2018 by Graham Reid

All rights reserved

LANDMINE DETECTION USING SEMI-SUPERVISED LEARNING

By

Graham Reid
B.S., Computer Engineering and Computer Science, University of Louisville, 2014

A Thesis Approved On

September 11, 2018

by the following Thesis Committee:

Hichem Frigui, Ph.D., Advisor

Amir A. Amini, Ph.D.

Olfa Nasraoui, Ph.D.

ABSTRACT

LANDMINE DETECTION USING SEMI-SUPERVISED LEARNING

Graham Reid

September 11, 2018

Landmine detection is imperative for the preservation of both military and civilian lives. While landmines are easy to place, they are relatively difficult to remove. The classic method of detecting landmines was by using metal-detectors. However, many present-day landmines are composed of little to no metal, necessitating the use of additional technologies. One of the most successful and widely employed technologies is Ground Penetrating Radar (GPR). In order to maximize efficiency of GPR-based landmine detection and minimize wasted effort caused by false alarms, intelligent detection methods such as machine learning are used. Many sophisticated algorithms are developed and employed to accomplish this. One such successful algorithm is K Nearest Neighbors (KNN) classification. Most of these algorithms, including KNN, are based on supervised learning, which requires labeling of known data. This process can be tedious. Semi-supervised learning leverages both labeled and unlabeled data in the training process, alleviating over-dependency on labeling. Semi-supervised learning has several advantages over supervised learning. For example, it applies well to large datasets because it uses the topology of unlabeled data to classify test data. Also, by allowing unlabeled data to influence classifica-

tion, one set of training data can be adopted into varying test environments. In this thesis, we explore a graph-based learning method known as Label Propagation as an alternative classifier to KNN classification, and validate its use on vehicle-mounted and handheld GPR systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION AND OVERVIEW

Landmine detection is a challenging but essential endeavour to the protection of civilian and military lives. While mines are fairly easy to plant, they are relatively difficult to detect. This is in part due to the advent of plastic explosives that cannot be discovered using energy-based detection methods, such as the kind used in metal detectors. Instead, present-day methods rely on signal analysis and image processing techniques to identify unique shapes and signals that are representative of landmine signatures beneath the earth. These methods are employed in both vehicle-based and handheld sensing devices.

Many landmine detectors employ numerous sensing methods simultaneously. Some of the most frequently used are Electromagnetic Induction (EMI), and more recently, Ground Penetrating Radar (GPR). The latter is more advantageous in detecting low-metal and non-metal targets which EMI is incapable of sensing. GPR also has the advantage that its signals can be reconstructed into an image. This image can then be analyzed using image processing and machine learning for the purpose of recognizing visual patterns within the radar.

There are three primary steps involved in landmine detection with GPR: prescreening, feature extraction, and alarm classification. Prescreening refers to the process of screening large quantities of data to identify anomalies. The goal of a prescreener is to be fast and to catch as many suspicious areas as possible. This means that a prescreener should prioritize recall at the potential expense of preci-

sion. Prior to prescreening, the raw GPR data may be preprocessed using various normalization methods and other techniques that reduce noise. After prescreening, the samples are positionally clustered, and single confidence values are assigned to collective positional areas of ground to speed up the more time-consuming process of feature extraction and classification. The output of the prescreener is a set of alarms, or areas of interest, to be fed into a classifier. The classifier typically employs techniques directly on the preprocessed GPR data to yield feature vectors which are used as observations evaluated by a learning algorithm. Classification refers primarily to the use of machine learning techniques to discriminate between potential targets and false alarms.

Classifiers that have been used for GPR based landmine detection generally fall into two categories: model-based, and instance-based. Model-based classifiers construct generative models that attempt to capture the underlying properties that define the data, and classify by estimating how much a test point conforms to assumptions made about the properties of the data. Instance based classification does not attempt to capture the properties of the data. Instead, representatives of data points are learned during training. These representatives, whose class labels are known, are then used during testing by comparing them to test points and using their similarity to determine the labels of test points. Both of these methods of classification are supervised, which means that they require training in order to distinguish mines and clutter. These learning methods are thus highly dependent upon accurate training data. During testing, label assignment is impacted only by a test point's comparison to the training data, without insight into the context surrounding that test point. Supervised learning does not take unlabeled data into account when assigning labels, potentially missing contextual clues in the test data that may aid the learning process.

In this thesis, we investigate the use of semi-supervised learning with labeled

and unlabeled data for classification. In particular, we examine the performance of a graph-based, instance-based learning method called Label Propagation (LP) as a new approach to the classification step. We evaluate the results of various implementations and experiments using LP, and compare them to nearest neighbor classification, which is another instance-based learning method that has proven successful in landmine detection.

Although LP does not require much training, it still must learn a set of representative prototypes. We explore several methods for learning prototypes, such as Multiple Instance Learning (MIL) and hierarchical clustering. Additionally, we experiment with fuzzy and possibilistic labeling, which has proven useful in landmine detection classification.

We demonstrate our methods on both vehicle-mounted and handheld GPR devices. We show that an LP classifier can consistently match or outperform a kNN classifier using the same set of prototypes and labels, assuming that the prototypes accurately represent the data. While our primary interest is in applying LP as a classifier, we also demonstrate its feasibility as a prescreener.

The organization of the rest of this thesis is as follows. In chapter 2, we discuss existing implementations of classification systems in both vehicle and handheld landmine detection. These systems include various preprocessing methods, feature descriptors and learning algorithms. In chapter 3, we describe our proposed Label Propagation classification method, including prototype learning and algorithm implementation. In chapter 4 we cover the results of our proposed methods, and discuss the implications of those results. Finally, in chapter 5 we draw conclusions about our results, and outline potential future experiments to improve them.

CHAPTER 2

RELATED WORK

2.1 Background

Historically, metal detectors were used to detect landmines. However, several difficulties have demonstrated that EMI sensing alone is insufficient for detecting landmines and other explosive devices. For example, EMI sensors are overly sensitive to metallic debris and have high false alarm rates in areas where there is a large amount of shrapnel and other miscellaneous metal material [1]. GPR is much less susceptible to this kind of contamination. Also, many modern landmines are composed of little or no metal, yet GPR is shown to be capable of detecting these landmines despite their non-metallic composition, while EMI is not [2]. Further, the typical range of depths under the earth at which landmines are buried is shallow enough to be precisely portrayed by GPR at high resolutions [3]. Ultimately, studies have shown that applying both EMI and GPR outperforms the use of either used alone [4]. GPR is not without its share of problems however. For example, GPR is susceptible to high signal reflection at the surface of the earth, commonly referred to as "Ground Bounce" [5]. The existence of the ground bounce is an important consideration for both vehicle and handheld GPR analysis.

Handheld GPR differs from vehicle-mounted GPR in several ways. Where vehicle GPR utilizes several dozen channels emitting and collecting signals to construct a GPR image, a handheld GPR uses significantly less. This means that the reconstructed image is limited to two dimensions, which restricts the available algorithms

that can be used [6]. Despite the smaller number of channels, a handheld GPR device may be used to sweep repeatedly over the same area of ground multiple times, where a vehicle-mounted GPR device generally passes over an area of ground only once. Another and more significant trait of handheld GPR is the effect of operator use on the reconstructed image. A vehicle-mounted GPR device is held quite steady relative to the ground assuming that the ground is level. Handheld GPR is susceptible to height variations which can occur if the operator is not holding the device steadily. Another possibility is that the operator could be descending or ascending a hill, where it is more difficult to keep the device at even distance with the ground. These are all scenarios that can cause the reconstructed GPR image to appear shaky and produce noise. These inconsistencies in the data can result in noisier features, making discriminative algorithms less reliable [7]. This has led to suggestions of a system that uses the handheld device as a guide, while ultimately relying upon the operator's discretion in determining whether an alarm is due to error or not [8]. Hence, a large focus of many of the currently developed handheld GPR detection algorithms are intended to operate as prescreeners serving the purpose of flagging as many alarms as possible, without taking extra time to identify or validate the alarms, which is what a classifier might do.

GPR image construction works as follows: The signals transmitted by GPR reflect back from the ground and are received by sensors on a device. These signals are interpreted as a function of time. This interpretation is referred to as an A-scan, and can be plotted as a line depicting time, or depth, along one axis, and signal strength along another axis. An example of an A-scan is shown in figure 1. One A-scan is collected for each position at which signals are emitted and received. These A-scans can be concatenated to form a 2-dimensional image, referred to as a B-scan. A group of A-scans form a B-scan which can be visualized as a 2-D image. Figure 2

shows an example of a B-scan. The A-scans (hereafter referred to as ”samples”) are lined up along the x-axis.

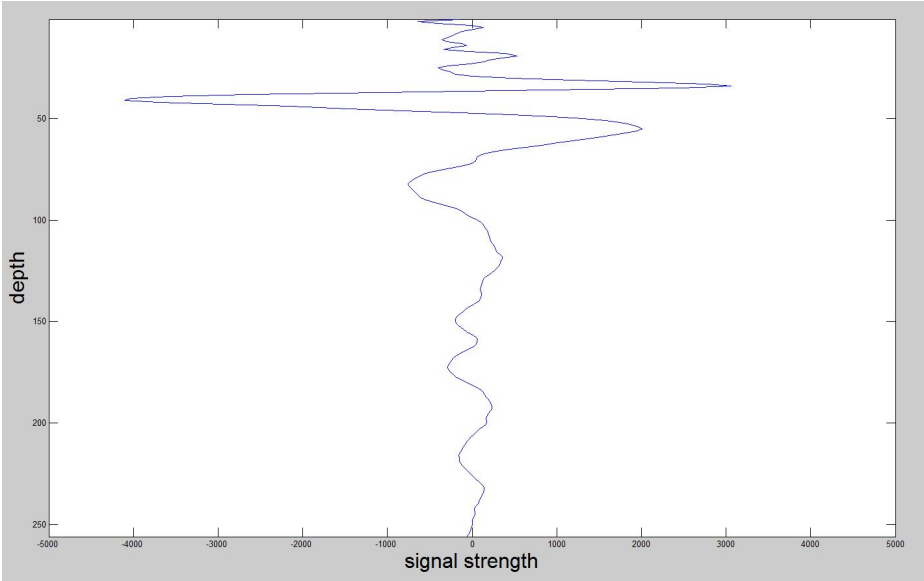

Figure 1. Example of an A-scan. The large variation seen around depth 50 is caused by ground bounce.

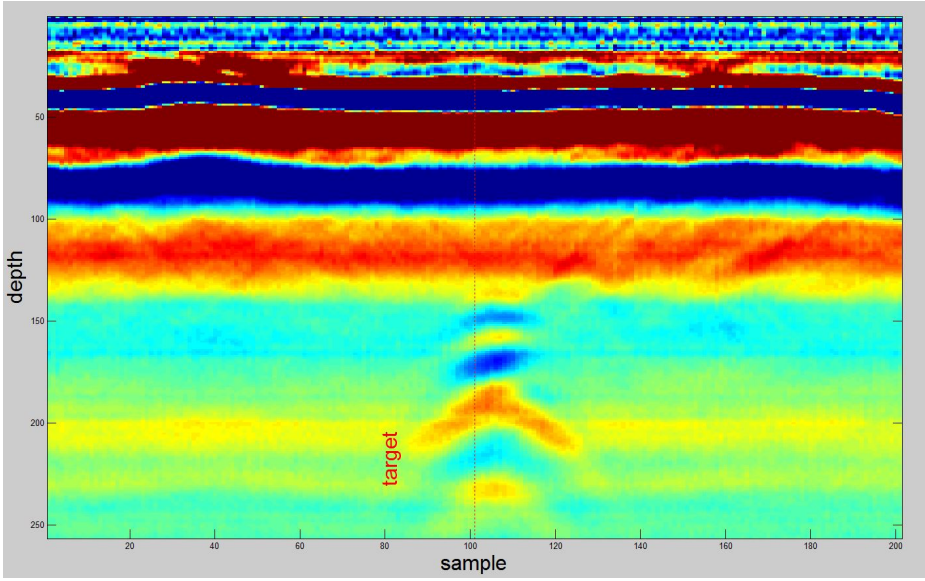

Figure 2. Example of a B-scan consisting of 200 A-scans with a deep target located around scan 100. These A-scans are uniformly sampled to be at least 1 cm apart. The colors depict signal strength.

2.2 Prescreening

Several prescreeners have been employed in vehicle GPR. One commonly used prescreening method is based on the Least Means Squared (LMS) [9, 5, 10, 11]. This LMS prescreener works by modeling the GPR data and removing an estimated mean from each sample. This approach is useful for removing ground bounce and suppressing areas with consistently high energies. Other techniques that have been explored in prescreening include median filtering, ground alignment, and Gaussian Mixture Models [12].

In the domain of handheld GPR, several new techniques have been explored for prescreening, such as curvelet filters [13], background adaptive division filtering [14], and deep belief networks [15]. Prescreeners are typically used as a first pass for data to go through before being evaluated by a more discriminative and computationally intense classifier. However, if said classifier is fast enough, it may be used in place of the prescreener [16].

2.3 Feature Extraction

After the prescreener identifies areas of interest as potential alarms, features are extracted from these areas and used to represent the alarms in a way that can be discriminated by a classifier. Features, or more specifically feature vectors, refer to numeric vectors that characterize individual instances of data. By describing an observation as a vector, it can be compared with other observations by means of a classifier which can then label and differentiate the original data.

In the case of landmines, features are used to capture the hyperbolic shape characteristic of a landmine signature. From figure 2, one can see an example of this hyperbolic pattern found in a target signature. It is often the objective of a particular feature representation to capture this hyperbolic shape as a sequence of rising, flat,

and falling edges. This is often done by quantizing the edge presence within a B-scan image. By using image filtering with edge filters corresponding to 45, 0, and -45 degree gradients, one can measure the amount of rising, flat, and falling edge, respectively, contained in the image.

Edge Histogram Descriptor (EHD) is one such feature extraction method that uses edge filters and has been successfully used in the context of landmine detection [10, 17, 18, 19]. EHD features capture the salient edge content of an image and represent it by concatenated histograms.

Figure 3 illustrates the process of extracting EHD features from GPR image data. First, the GPR image is divided by depth into N overlapping sub-windows. Then, four different edge filters are applied to each of the sub-windows. Each of these filters corresponds to a type of edge: horizontal, vertical, diagonal, and anti-diagonal. This filtering produces four new images per sub-window – one per type of edge. Each of these new images represents the strength of that type of edge at each pixel. By taking the maximum intensity value over all four images, each pixel votes for whichever edge type is most prominent at that location. This gives us a matrix of indices that represents the strongest edge type for each pixel. We then construct a histogram for each of these matrices that represents the proportion of each type of edge in each sub-window. Often, a threshold is used to require that the maximum of the four images at a pixel location be relatively large, otherwise the pixel is counted as "nonedge", which will be a fifth edge type. Finally, we concatenate the histograms of the sub-windows together to obtain our EHD feature vector of length $(N * 4)$, or $(N * 5)$ if a non-edge category is used.

Another edge-based feature representation, which has been applied successfully to landmine detection, is the Histogram of Oriented Gradients (HOG) [20, 21, 22, 23]. HOG can be considered as a generalization of EHD. Instead of restricting the edge

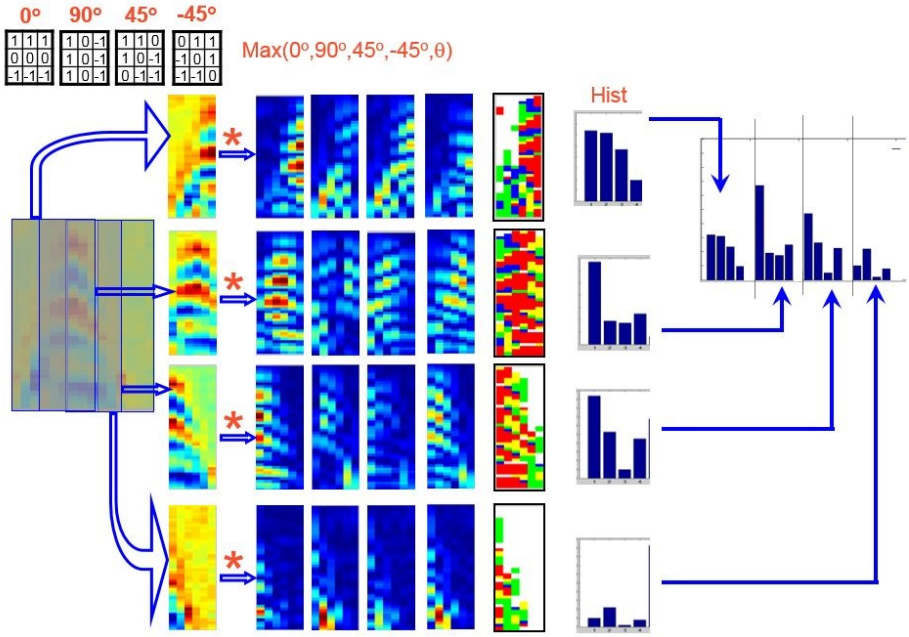

Figure 3. Illustration of the EHD feature extraction performed on landmine GPR image data with no nonedge.

type to four specific directions, each pixel is associated with a quantized gradient direction.

Local Binary Patterns (LBP) is another feature type that has been used in landmine detection [20, 23]. Instead of using gradient features like EHD and HOG, LBP uses binary patterns that depict comparisons of a central pixel's intensity value to its neighboring pixels. Fisher Vectors (FV) have also been explored in landmine detection [24]. FV uses variation from statistical models to characterize image data. Finally, frequency domain features have also been used to describe GPR images for landmine detection [25, 26]. In this thesis, we focus on EHD features to represent the alarms. However, other features could be easily integrated.

2.4 Discrimination Algorithms

Once the features are decided and extracted for all data that passed the pre-screener, a classifier, which can label a signature as either mine or false alarm, must be learned. Classification is a supervised learning technique that involves training and testing. These steps vary depending on the classification algorithm used. Training is the process of using a collection of labeled data to learn a mapping between the distribution of the features of the data and the labels associated with those features. Testing is the process of using the learned mapping to predict the label of new test samples.

To reduce the sensitivity of the results to the selected training and test subsets, typically, the validation of a classification algorithm is performed using multiple cross-validation folds. These folds are partitions of the original dataset into mutually exclusive learning/testing subsets of data. For instance, using 10 fold cross validation, 9 out of 10 subsets of the data are used for training, while the 10th subset is tested. The results of these 10 folds are then combined and scored to evaluate a classifier's performance.

Alarm classification is a subject of research that has been well explored. Examples of classifiers that have been used include K-Nearest Neighbors (KNN) [19] Support Vector Machines (SVM) [24, 27], and Hidden Markov Models (HMM) [21, 18, 28, 29, 30]. In this thesis, we compare our proposed LP classifier with the provenly successful KNN classifier.

2.4.1 The K-Nearest Neighbor (KNN) Classifier

KNN is a simple algorithm that classifies a test sample by comparing it to training data with known labels. Like most classifiers, KNN involves both training and testing. Training in this case refers to the process of deciding on a set of example

vectors to use as anchor points. Testing involves comparing an unknown test sample to the example vectors. A confidence value is assigned to the test point based on the labels and proximity of the k closest training samples. A visual illustration of how KNN works is presented in figure 4. In this figure, the label of a test sample is calculated by identifying the (k=4) nearest neighbors and combining their labels using either a simple or weighted average.

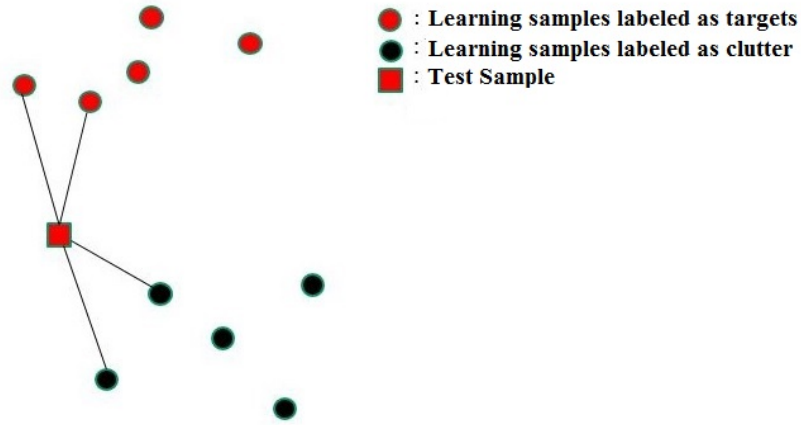

Figure 4. Illustration of KNN classifier. The label of a test point is calculated by combining the labels of its (k=4) nearest neighbors.

Ideally, a KNN classifier could compare all test data to all training data to identify the labels of test data. However, in most cases the training data are too large and would cause comparison of test and training data to become slow and inefficient. For this reason, training points are summarized by few representatives. These representative prototypes will have assigned labels learned from the training data they represent.

Other classifiers, such as SVM [24, 27], HMM [21, 18, 28, 29, 30], Random Forest [16], Neural Networks [31], and Bayesian Relevance Vector Machines [32] learn the distribution of the training data and identify boundaries that can separate observations from different classes in the feature space. Comparatively, training a KNN

classifier is relatively simple, which typically results in faster training.

While KNN may be an attractive classifier due to its ease of use, interpretability, and generalization ability, its performance is dependent upon the quality of the labeled samples learned during training. This means that careful selection of these prototypes during training is key for good classification results. In some cases, it may be sufficient to choose these samples manually, but in many other cases it is better to learn them automatically from a training dataset. Methods that can learn prototypes from training data include clustering (k-means, fuzzy c-means), Multiple Instance Learning (MIL) [21, 31, 33, 34], Self Organizing Maps (SOM) [19], and transfer learning [35].

After a set of training prototypes is learned, they are labeled based on what class they belong to. A simple way of labeling these prototypes is using binary labels. If a prototype represents a group of alarms that include more targets than clutter, then it will be labeled as a target. Otherwise, it will be labeled as clutter. Another approach is to use fuzzy labeling. Fuzzy labeling assigns a value between 0 and 1 depending on the target to clutter ratio of the alarms assigned to its cluster. Fuzzy labeling has been applied to landmine detection in many classifiers [36, 34, 37, 38, 31].

Possibilistic membership is another labeling alternative. Possibilistic labeling accounts for the possibility that a particular training prototype may not belong to any available class label. With this, the requirement that the memberships of a prototype to all class labels sum to one is relaxed. Instead, a prototype is assigned a "soft" label that reflects its membership to each class label and which only expresses the likelihood of belonging to a particular class. Like fuzzy labeling, possiblistic membership has been extensively applied in landmine detection [24, 19, 36].

During classification, confidence values are assigned to observations indicating whether they belong to one class or another. Similar to the labels of the training

prototypes, a test point's confidence may be a binary 0 or 1, or it could be any value within the range of all unique values assigned by the classifier. Once testing is finished, this range is used to plot a Receiver Operating Characteristic curve [39].

2.4.2 Label Propagation

The classification methods mentioned in section 2.4 are all supervised machine learning algorithms. This means that these algorithms use only labeled training data to learn a mapping that can predict the label of a new test sample. Label propagation, however, is a semi-supervised machine learning method. Semi-supervised learning algorithms also use labeled training data; However, they differ from supervised classifiers because they utilize additional unlabeled data to assist labeled data during training and/or testing. In addition to being semi-supervised, Label Propagation is a graph-based algorithm that computes pairwise similarities between labeled and unlabeled data points, converts these similarities to edges in a graph, and uses this graph together with the labels of the training data to compute the labels of the unlabeled points. Recent applications for which Label Propagation has been used include Saliency estimation [40] and object tracking in video data [41, 42, 43, 44]. In many of these applications, LP is used as an intermediary and high-speed classifier to identify objects between frames. A slower and more powerful classifier is used to recognize objects in the initial frame.

LP, like KNN, uses the pairwise similarities between observations to assign labels to test samples. However, KNN only considers the distance between a test point and its closest training points when assigning its confidence value. LP instead uses the pairwise similarities between all data to assign a confidence value to any test point. This includes other test points and other prototypes that aren't necessarily nearest neighbors. Figure 5 illustrates how this process differs from KNN (as illustrated in

figure 4). LP includes more information in confidence assignment by constructing a fully connected graph over all points (training and testing), and using the topology of the graph to assign an appropriate confidence value to each of the unlabeled points. This means that as long as the features describe the data accurately, the more unlabeled data one includes in testing, the more reliable one can expect classification to be.

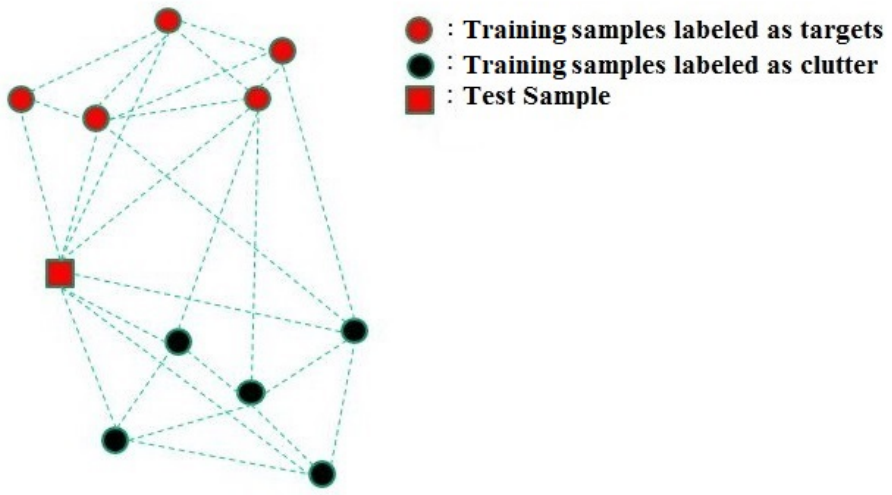

Figure 5. Illustration of LP classification. By constructing a fully connected graph over all points, the topology of the data is used to propagate a label onto the test point. For clarity, not all edges are shown here.

In the following, we explore four different implementations of the Label Propagation. The first three of which are iterative, while the last algorithm offers a closed form solution.

2.4.2.1 Zhu and Ghahramani (Iterative Label Propagation) Algorithm

The first of these methods uses an iterative approach proposed by Zhu and Ghahramani (2002) which repeats three steps: (1) propagate labels through graph, (2) row-normalize graph, and (3) clamp labels of training data.

Let $X_L = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{N_L}\}$ represent N_L feature vectors of the labeled sam-

ples, and $Y_L = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_{N_L}\}$ be the labels of those samples. Let $X_U = \{\mathbf{x}_{N_L+1}, \mathbf{x}_{N_L+2}, \cdots, \mathbf{x}_{N_L+N_U}\}$ be the feature vectors of the N_U feature vectors of the unlabeled data. First we calculate the pairwise distances between all pairs of instances in the union set $\mathbf{X} = \{X_L \cup X_U\}$. We obtain the distance matrix $\mathbf{D}$, where $\mathbf{d}_{ij}$ is the distance between observations $\mathbf{x}_i$ and $\mathbf{x}_j$. We then construct a fully connected graph over $\mathbf{X}$ where nodes represent the $\mathbf{x}_i \in \mathbf{X}$, and the edge between nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, w_{ij}, is proportional to the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. We use the following equation to calculate these similarities.

$$w_{ij} = \exp\Big(-\frac{{d^2}_{ij}}{\sigma^2}\Big) \tag{1}$$

The value of σ in (1) is a parameter that is used to control the distance at which two nodes will influence each other. The smaller σ, the closer two nodes must be to affect one another significantly.

Next, we calculate the diagonal degree matrix $\mathbf{D}^g$ that is simply the sum of the edge weights for each row in $\mathbf{W}$:

$${D^g}_{ii} = \sum_j w_{ij} \tag{2}$$

We then append N_U zeros onto the end of the Y_L to get $Y^0 \leftarrow \{Y_L, 0, 0, \ldots, 0\}$. Then, the algorithm propagates the labels while clamping the values of the prototypes to their known labels, and repeats until convergence. This is outlined in algorithm 1.

Algorithm 1: Zhu and Ghahramani (Iterative)

Compute affinity matrix $\mathbf{W}$ from (1)
Compute diagonal degree matrix $\mathbf{D}^g$ from (2)
Intialize $Y^{(0)} \leftarrow \{Y_L, 0, 0, \ldots, 0\}$
Iterate:
1: $Y^{t+1} \leftarrow (\mathbf{D}^g)^{-1}\mathbf{W}Y^t$
2: $Y^{t+1}_{1,\ldots,N_L} \leftarrow Y_L$
Until convergence of Y

Step 2 in the "Iterate" portion of algorithm 1 serves to clamp the known labels Y_L while step 1 propagates the class labels based on the topology of the data.

2.4.2.2 Label Propagation based on Jacobi Iteration

The Jacobi Iteration algorithm for LP [45] is inspired by the Jacobi iterative method for linear systems. In this algorithm, we allow Y_L to change based on the topology of the data and the initial labels of the known points ${Y_L}^{(0)}$. This algorithm uses a parameter α to control the influence of the initial labeling. It starts by computing a matrix $\mathbf{A}$ using

$$a_{ii} \leftarrow I_{[l]}(i) + \mu {\mathbf{D}^g}_{ii} + \mu\epsilon \tag{3}$$

where $\mathbf{D}^g$ is computed using (2), and $I_{[l]}(i)$ is the identity matrix over the l labeled points. In (3), $\mu = \frac{\alpha}{1-\alpha}$, where $\alpha \in (0, 1)$ is a coefficient that controls the degree to which labels of labeled samples are allowed to change. Our initial labels Y^0 are the same as in algorithm 1. With the known labels allowed to change, the LP calculation changes as well. The complete is outlined in algorithm (2).

Algorithm 2: LP Based on Jacobi Iteration

Compute affinity matrix $\mathbf{W}$ from (1)
Set $w_{ii} \leftarrow 0$
Compute diagonal degree matrix $\mathbf{D}^g$ from (2)
Choose value for $\alpha \in (0, 1)$ and let $\mu = \frac{\alpha}{1-\alpha}$
Compute diagonal matrix $\mathbf{A}$ using (3)
Intialize $Y^{(0)} \leftarrow \{Y_L, 0, 0, \ldots, 0\}$
Iterate:
1: $Y^{t+1} \leftarrow \mathbf{A}^{-1}(\mu \mathbf{W} Y^{(t)} + Y^{(0)})$
Until convergence of Y

A large value for α will allow the topology of both the labeled and unlabeled points to influence the classification, and a small value of α will allow the initial labeling of the prototypes to influence the classification.

2.4.2.3 Label Propagation based on Label Spreading

LP with label spreading relies on the Laplacian of the degree matrix $\mathbf{D}$ which results in a similar but different result as algorithm 2. The technique used is referred to as "label spreading" [45], in which each node receives a contribution to its label from each of its neighbors, in addition to a contribution by the original labeling of the prototypes.

First, we compute the degree matrix $\mathbf{D}^g$ as in (2). Next we compute the graph Laplacian of $\mathbf{D}^g$ using

$$\mathcal{L} \leftarrow (\mathbf{D}^g)^{-1/2}\mathbf{W}(\mathbf{D}^g)^{-1/2} \tag{4}$$

Once again we initialize $Y^{(0)}$ as in algorithm 1 and decide on a value for the parameter α to control the influence of the data's topology against initial labeling of the prototypes. This gives us the following algorithm:

Algorithm 3: LP based on Label Spreading

Compute affinity matrix $\mathbf{W}$ using (1)
Set $w_{ii} \leftarrow 0$
Compute diagonal degree matrix $\mathbf{D}^g$ using (2)
Compute normalized graph Laplacian using (4)
Intialize $Y^{(0)} \leftarrow \{Y_L, 0, 0, \ldots, 0\}$
Choose value for $\alpha \in [0, 1)$
Iterate:
1: $Y^{(t+1)} \leftarrow \alpha\mathcal{L}Y^{(t)} + (1-\alpha)Y^{(0)}$
Until convergence of Y

Similar to algorithm 2, α directly affects the influence of the topology and inversely affects the initial labeling $Y^{(0)}$.

2.4.2.4 Zhu and Ghahramani (Closed Form Label Propagation Algorithm)

Zhu and Gharhamani proposed an alternative approach to the iterative algorithm outlined in algorithm 1. This approach proves that repeating the iterative steps from algorithm 1 can be shown to converge to a closed form solution [46].

First, we construct a probabilistic transition matrix $\mathbf{T}$ from the similarity matrix computed in (1) using

$$\mathbf{T}_{ij} = P(j \rightarrow i) = \frac{w_{ij}}{\sum_{k=1}^{N_L+N_U} w_{kj}}. \tag{5}$$

Essentially, (5) represents a random walk, where each element in $\mathbf{T}$ is the likelihood of randomly moving from node j to node i. We then row normalize the matrix using

$$\bar{\mathbf{T}}_{ij} = \frac{\mathbf{T_{ij}}}{\sum_{k=1}^{N_L+N_U} \mathbf{T}_{ik}} \tag{6}$$

$\bar{\mathbf{T}}$ is a square matrix of size $N_L + N_U$ then can be split into four sub-matrices:

$$\begin{bmatrix} \bar{\mathbf{T}}_{LL} & \bar{\mathbf{T}}_{LU} \\ \bar{\mathbf{T}}_{UL} & \bar{\mathbf{T}}_{UU} \end{bmatrix} \tag{7}$$

In (7), $\bar{\mathbf{T}}_{LL}$ consists of all transitions from a labeled point to another labeled point. $\bar{\mathbf{T}}_{UL}$ consists of all transitions from an unlabeled point to a labeled point, etc.

It has been proven [46] that the labels of the unlabeled points can be computed using the following equation:

$$\mathbf{Y_U} = \left(\mathbf{1} - \bar{\mathbf{T}}_{\mathbf{UU}}\right)^{-\mathbf{1}} \bar{\mathbf{T}}_{\mathbf{UL}} \mathbf{Y_L} \tag{8}$$

The complete process is outlined in algorithm (4).

Algorithm 4: Closed Form Label Propagation

Compute transition matrix $\bar{\mathbf{T}}$ using (5) and (6)
Split $\bar{\mathbf{T}}$ into four sub-matrices as per (7)
Compute labels of unlabeled data $\mathbf{Y_U}$ using (8)

CHAPTER 3

LANDMINE DETECTION USING SEMI-SUPERVISED LEARNING

In this chapter, we start by motivating the advantages of using semi-supervised learning as an alterative to supervised learning for landmine detection. We validate this by using Label Propagation (LP) as an alternative to a K-Nearest Neighbor (KNN) based classifier. We outline our architecture which includes preprocessing, prescreening, feature extraction, and discrimination steps. Our architecture will be demonstrated on vehicle-mounted and hand-held landmine detection systems. The intermediate steps involved are illustrated using visualizations of landmine data.

3.1 Motivations

The goal of a discriminative algorithm in landmine detection is to correctly separate a set of alarms into mines and clutter. KNN classification is an example of a discriminative algorithm that does this well [10, 19]. In this thesis, we provide an alternative classifier to KNN that takes advantage of available unlabeled data.

In domains where the underlying characteristics of the data can vary between datasets, it is critical to have a discriminator that is robust. KNN classification has proven to be robust, especially in the context of detecting landmines from vehicle GPR data [19]. This is due in large part to the development of systems which train a KNN classifier by summarizing a larger training set of data with a smaller set of representatives. It is this summarization that prevents classifiers from over-fitting the

learning data and allows these systems to generalize well to new datasets.

Label Propagation shares many of the same traits as KNN. However, it has the advantage of incorporating unlabeled data during the testing phase. In GPR based landmine detection, there is an abundance of unlabeled data that is often discarded during prescreening or at other stages in the landmine classification process. We seek to utilize this data during classification to see if it may enhance the accuracy of a learning algorithm. As a semi-supervised learning method, LP offers an approach that combines labeled and unlabeled data to aid classification.

We consider using our learning method both as a discriminator and as a prescreener. Traditionally, prescreeners act as fast, simple methods of filtering larger quantities of data into fewer regions of interest. We demonstrate the feasibility of LP as a prescreener for handheld GPR datasets, given that our application of LP is fast enough to operate upon raw GPR data in real-time.

We validate our system by comparing it to KNN on several GPR based landmine datasets, taken from both vehicle and handheld systems.

3.2 Preprocessing

Using a prescreener directly on unfiltered GPR data is prone to high rate of false alarms due to anomalies within the data such as noise and uneven terrain. Therefore, several preprocessing steps are used to filter, normalize, and align the signal data.

Figure 6 illustrates all of the steps involved in our approach. The first step in preprocessing is to uniformly sample the signal data. An operator may slow down or speed up as the device is in use, causing the positional spacing between samples to be uneven. Uniform sampling reduces the number of samples so that their Universal Transverse Mercator (UTM) coordinates cannot be closer than a threshold. This way,

we can be sure that any two groups of adjacent samples collected that have equal size cover the same amount of positional space.

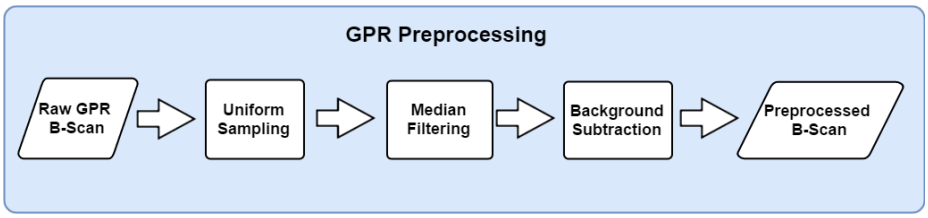

Figure 6. Flowchart illustrating proposed GPR preprocessing method on handheld data.

In the next step, the sampled data is concatenated to form a B-scan image. We median filter this image to remove speckle noise and small discontinuities within it. Noise is much more significant across samples than across depths, partially due to the uniform sampling. Therefore, we filter each depth independently.

Next, we perform a background subtraction similar to what has been used on vehicle GPR data [36]. In real time, the GPR data are received as A-scans which are retrieved every 1 cm of forward advance by the device due to our uniform sampling. We retain a B-Scan $\mathbf{R}_{bg}$ as background data (stored in a buffer) which precedes the current kth scan by a distance N_{dist}. We use this B-Scan to learn the statistics of the background of scan k (μ_k, σ_k), then, we normalize the current A-scan k using

$$\bar{s}_k = \frac{s_k - \mu_k}{\sigma_k} \tag{9}$$

In (9), s_k is all of the values collected at the kth A-scan, μ_k and σ_k are the depth-wise mean and standard deviation computed from $\mathbf{R}_{bg}$. Figure 7 shows the relative locations of these regions during background subtraction. While s_k is the current scan, $\mathbf{R}_{bg}$ is stored in the buffer. We perform this background subtraction over all scans collected from the lane GPR data, one scan at a time.

Normalizing with background subtraction helps eliminate constant (not alarm

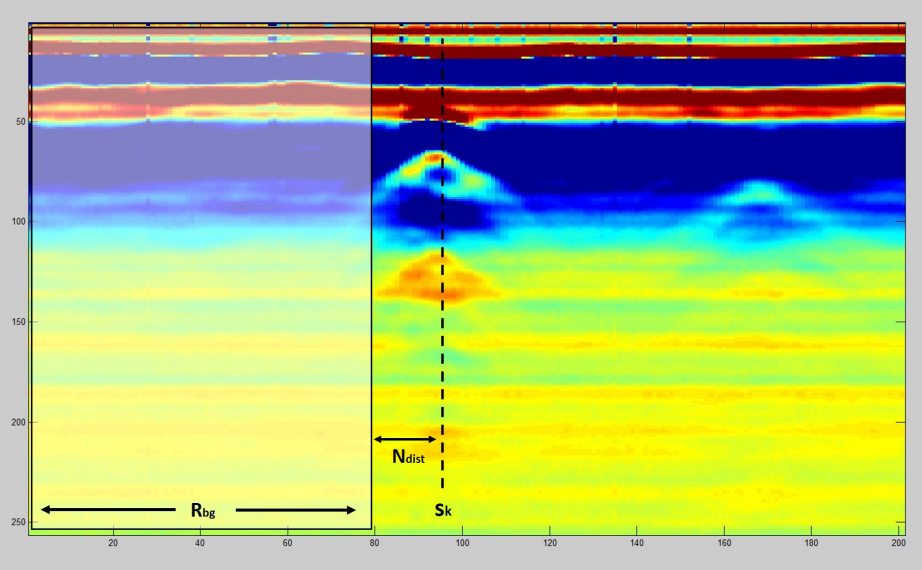

Figure 7. Illustration of the location of the background buffer $\mathbf{R}_{bg}$ relative to the current scan s_k that is being normalized.

related) energies across samples, such as the high energy caused by the ground bounce. Dividing $\mathbf{s}_k$ by σ_k normalizes the strength of the signal energies across depths. This is necessary because targets buried deeper in the ground tend to have weaker energy. Figure 8 illustrates the B-Scans of the target from figure 7 before and after normalization.

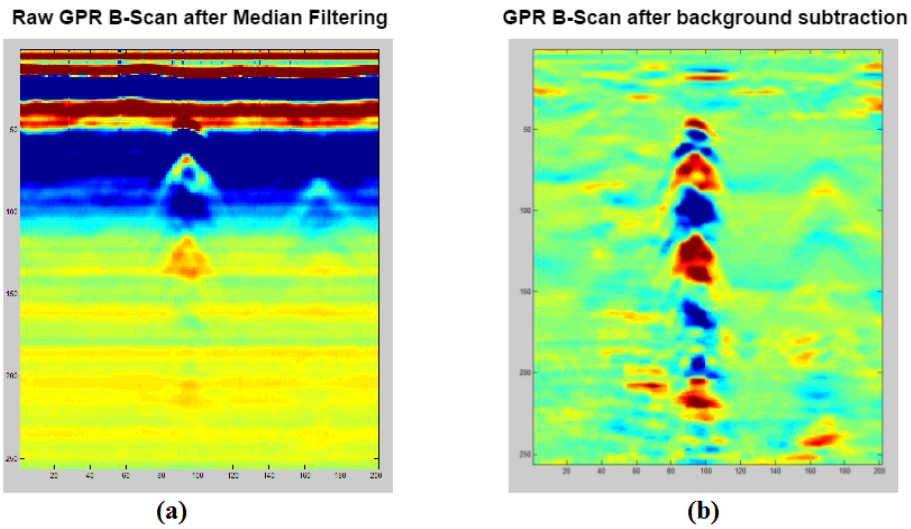

Figure 8. A sample GPR B-Scan (a) before background subtraction and (b) after background subtraction.

3.3 Edge Histogram Descriptor (EHD) Feature Extraction

After data preprocessing, our system employs an EHD feature extractor to capture the hyperbolic shape that is characteristic of landmines in time-domain GPR data. EHD captures the spatial distribution, degree, and orientation of edges as they appear in GPR data.

Our implementation of EHD follows a similar process as those described in prior landmine detection applications [19]. This approach extracts edge histograms that measures the frequency of occurrence of edge orientations at a particular sample within GPR data. Our time-domain GPR data are evaluated as a 2-D grayscale image, and EHD captures the frequency and direction of edges within it.

We apply 2-D EHD edge operators as the hand-held GPR data has a single channel. Each region, i.e. B-Scan, $\mathbf{R}_{tst}$ that we evaluate spans 21 scans across (a center scan plus 10 scans before and after it). This region is subdivided by depth into several overlapping windows, as demonstrated in figure 9. Let S_{xy} be the 2-D signature of one of these windows for a given region. We compute each of four categories of edge strengths: horizontal, vertical, diagonal, and anti-diagonal.

Following this computation, we apply a similar background normalization as the one described in section 3.2 to each of these edge strength values independently. Figure 10 demonstrates the benefit of performing this additional normalization. Without normalization, horizontal edge activity dominates the other features, even when no visible edge activity is present. Subtracting the mean computed from background for each edge type normalizes the edges so the differences in edge presence are compared rather than overall edge strength.

If the maximum of the edge strengths after normalization is greater than a predetermined threshold θ_G, the corresponding pixel is considered an edge pixel, otherwise, it is considered a nonedge pixel.

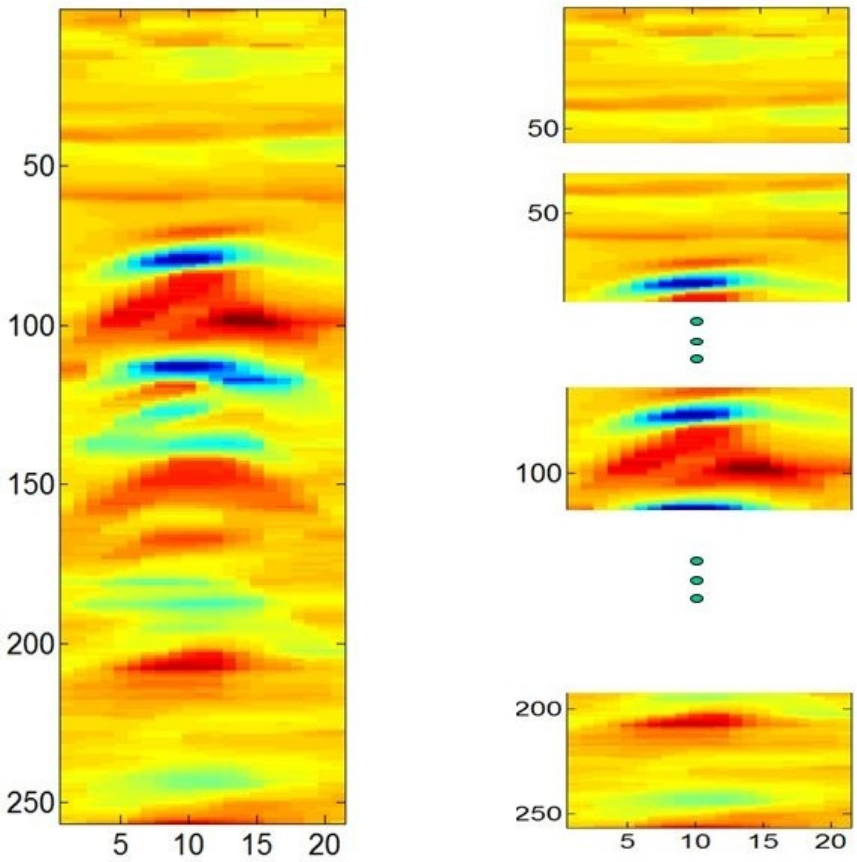

Figure 9. (a) Example of preprocessed mine GPR region and the (b) overlapping subwindows into which it is divided. From each subwindow a corresponding signature and EHD feature vector will be extracted.

Computing a single histogram over S_{xy} would fail to capture the hyperbolic shape we are interested in, because histograms do not take into account spatial information. For instance, a histogram-based feature cannot discriminate between a signature that has a diagonal edge followed by an anti-diagonal edge and one that has an anti-diagonal edge followed by a diagonal edge.

To circumvent this, we vertically subdivide S_{xy} into four overlapping subimages S_{xy_i}, $i = 1, ..., 4$. These subimages overlap in order to capture sufficiently large edges, and also to reduce the sensitivity of the feature representation to size and locational variations between signatures. For each S_{xy}, a four-bin edge histogram

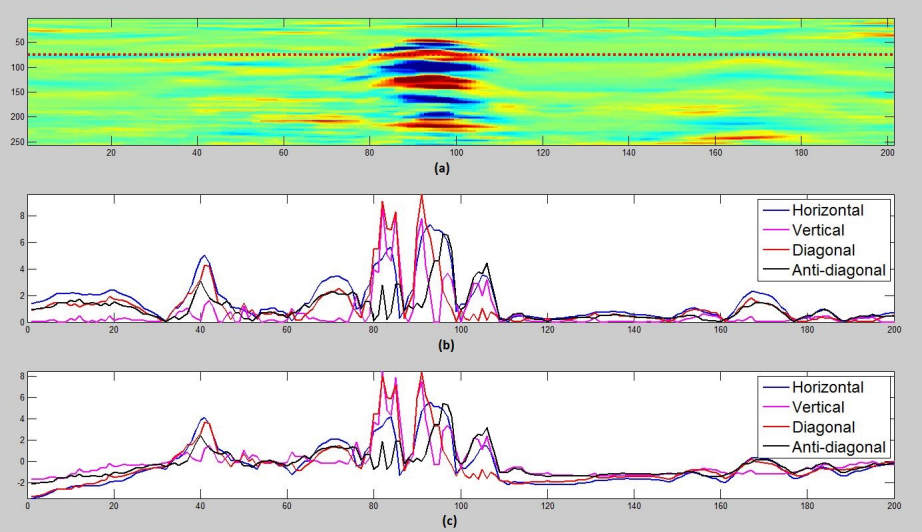

Figure 10. (a) Example preprocessed GPR image. (b) calculated edge amount at each scan from (a), at the depth marked by the dotted red line. (c) normalized edge amount calculated at the same locations.

H_{xy_i} is computed. The four bins correspond to the four edge categories.

The EHD final feature extracted from S_{xy} is defined as the concatenation of the 4 four-bin histograms, i.e.,

$$\text{EHD}(S_{xy}) = [H_{xy_1}, H_{xy_2}, H_{xy_3}, H_{xy_4}] \tag{10}$$

This process of extracting a 16-dimensional EHD(S_{xy}) from a signature S_{xy} is demonstrated on a mine signature in figure 11. We label the individual features h_i, v_i, d_i, and a_i, to denote the horizontal, vertical, diagonal, and anti-diagonal edge components of H_{xy_i}

3.4 Data Summarization and Representation

All of the classifiers that we use in our experiments are instance-based. This means that we achieve classification of our test points by comparing them with other data. To this end, it is desired that we have as much data as possible with which to compare our test points. However, landmine detection is a real-time process, and if

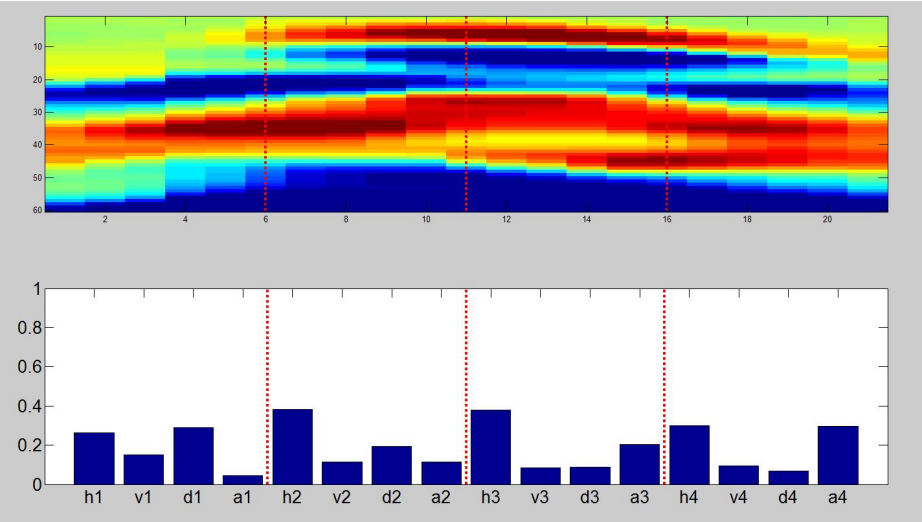

Figure 11. Example of a mine signature with its calculated feature vector EHD(S_{xy}). Subimages and their calculated histograms H_{xy_i} are separated by the red dotted lines (subimages have slight overlap and aren't exclusively separated by the red lines).

the amount of data to compare is too large, classification will be too slow to keep up with the GPR device. To take advantage of the large amounts of available training data and keep computations to a minimum, we summarize our labeled data by a smaller set of representative prototypes as will be described in section 3.4.2.

3.4.1 Locating Mine Signatures within Alarms

Summarizing our data requires that we first partition them into mine signatures and clutter signatures. Our original input data are a set of alarms. Each of these alarms contains multiple signatures that vary by depth, and has a ground truth indicating whether the alarm is a mine or clutter. If an alarm is a mine, it is not known which signatures contain the hyperbolic shape signifying a target. Thus, an algorithm is needed to locate the depth location at which the signature should be extracted.

We explore two methods to identify the most likely target signature from each alarm. The first one uses a simple condition that assigns a confidence value, based

on the diagonal edge strength in the left sub-image and the anti-diagonal edge in the right sub-image, to all signatures belonging to all alarms. It then thresholds based on this condition to acquire only the strongest signatures from each mine.

Our second method uses Kernel Density Estimation (KDE) [47] to locate the mine signatures within alarms that are labeled as mines. Let $P^- = \{p_1^-, p_2^-, \ldots, p_k^-\}$ be a set of pre-learned prototypes summarizing clutter alarms. We estimate the KDE of each signature x_i within an alarm using

$$KDE(x_i) = \frac{1}{z} \sum_{j=1}^{k} exp(-\beta \| x_i - p_j^- \|) \tag{11}$$

In (11), β is a resolution parameter learned during summarization of the clutter signatures, and z is a normalization factor. Signatures with low assigned KDE (close to zero) are selected as the most likely depth corresponding to an alarm's target signature.

3.4.2 Learning Prototypes

After the mine signatures have been extracted from their alarms, we use clustering to reduce the number of mines and clutter signatures to a smaller number of representatives which will allow our classifiers to perform efficiently. We perform this clustering once for mine signatures and once for clutter signatures to obtain separate sets of representatives. These representative prototypes constitute our final set of training points which will be used by our classifiers during testing.

We explore two clustering methods. The first method uses Agglomerative Hierarchical Clustering since the size of the training data after thresholding is typically small enough to employ this technique, and agglomerative clustering is shown to be generally superior to partitional clustering when the size of the training data is small [48]. We also use complete linkage in order to ensure variation in the final set of

prototypes.

Our second method uses Self-Organizing Maps (SOM) for clustering. SOM has proven successful in training classifiers for landmine detection, and is well-suited for preserving the topology and representation of data during clustering [19].

3.4.3 Labeling

When assigning labels to indicate which prototypes are mines and which are false alarms, the easiest way to distinguish between the two is using binary, or crisp, labeling. In this setting, mine prototypes are assigned a label of 1, and false alarm prototypes are assigned a label of 0 or -1.

Prototypes outputted by a training method do not equally represent their class labels however. For example, an ideal mine signature describes a clear hyperbolic shape. But many mine signatures do not conform to this expectation. The benefit of training is that exceptions and subtleties are captured by a variation of prototypes, which broadens the representation of possible mine appearances captured through training. Further, representatives have varying importance. Some capture the most obvious features of a mine, while others may capture less apparent indicators.

Binary labeling fails to reflect the significance of individual prototypes. It is therefore necessary to use some other method of soft labeling to convey this additional information about our prototypes. Ideally, the more important prototypes should be assigned a higher label, and during classification these prototypes should have more influence on the assigned confidence values.

We use fuzzy labeling as a way to convey the significance of the learned prototypes. Fuzzy labeling serves to provide uncertainty during classification by allowing our prototypes' labels to fall into any range of numbers. We use a high value (close to 1) to indicate a strong mine, and a low value (close to 0) to indicate a weak mine.

This helps establish a ranking of the prototypes learned during training that can influence the confidence values assigned by the classifier.

For our experiments, several different methods of fuzzy labeling are used. Our first approach uses a prescreener assigned confidence to label prototypes. This prescreener is detailed in chapter 4. We also use the KDE from section 3.4 as a labeling method, where we use a set of clutter signatures learned during training to estimate the KDE of all prototypes and assign their KDE as their label.

Finally, we look into a labeling approach that considers the nearest training signatures to each prototype to assign its label. Let y_i be the label of the ith closest training signature to a prototype $\mathbf{p}$. We compute $\mathbf{p}$'s label using the average label of the n closest training signatures, where each signature is given a label of 1 for target and 0 for clutter. We calculate a prototype p's label y_p using

$$y_p = \frac{\sum_1^n y_i}{n} \tag{12}$$

We also considered possibilistic classification in our experiments. A standard KNN classifier cannot discriminate between cases where a test point's neighbors are all equally close and cases where its neighbors are all equally far away. This can cause a classifier to label a test point that is very close to all of its neighbors to have the same label as a point that is very far away from the exact same neighbors. We use possibilistic classification to alleviate this by relaxing the constraint that the sum of a test point's membership to all classes must sum to 1. During data summarization, we calculate the mean and standard deviation of the distances of training points to each cluster center (i.e. learned prototypes). Let μ_c and σ_c denote the calculated mean and standard deviation for prototype c. After calculating a test point's distance from c, we normalize it using

$$\bar{d} = \frac{max(0, |d - \mu_c|)}{\sigma_c} \tag{13}$$

The normalized distance $\bar{d}$ will be 0 if the test point is close to c relative to μ_c. Then, we convert this value to a weight using

$$w = \frac{1}{\bar{d} + 1} \tag{14}$$

We use w to adjust the contribution of prototype p to the overall confidence of the test point. This method of labeling allows us to use the posterior distribution of the training data in confidence assignment during testing.

3.4.4 Graph Construction

In section 2.4.2, we detailed graph construction over a set of observations, which is required to perform label propagation. When applied to landmine detection, we construct a graph that includes all training (labeled) prototypes and one or more test (unlabeled) signatures. In this context, prototypes/signatures are the nodes, and the distances between them are the edge weights in the graph.

In equation (1), we use a control parameter σ to limit the influence of adjacent edges such that the smaller the value of σ, the closer two points must be to influence each other's labels. We explore few options to assign a value to σ.

The simplest method is by experimentation, where we vary σ and evaluate the results. We compare ROCs on specific test datasets and select the best value for σ. Another approach employs the heuristic suggested in [46] that uses minimum spanning trees to decide σ. Specifically, by constructing a minimum spanning tree over all training points, we find the shortest edge d connecting prototypes belonging to separate classes within the tree. Following the assumption of normal distribution, we let $\sigma = d/3$.

We also investigate the use of KNN trees as an alternative method of graph construction. With KNN trees, we construct a sparse graph rather than a fully connected one, by constructing edges only between an observation and its k nearest neighbors, and setting the weight of all edges to 1. This method spaces the nodes evenly so that each training point's label is influenced equally by each of its neighbors. Constructing the KNN tree replaces equation (1) from section 2.4.2. With this method, we use experimentation to determine a good value of k to use when constructing the KNN tree.

3.4.5 Unlabeled Data

One of the advantages of LP is its utilization of unlabeled data in the process of label assignment. Unlabeled data are any data where the labels are not known or are not observed. Given the ample data available during landmine detection that are not used, we propose several methods of acquiring and utilizing unlabeled data to aid in classification.

Unlabeled data can be collected at different points during system operation. We explore two options. First, we consider extracting unlabeled data from a region slightly preceding the location of the alarm reported by the prescreener. Using this approach, we seek to examine whether including data that spatially precedes an alarm (ie have the same soil properties) will help in propagating the labels of the known data. We hypothesize that by creating a context around the test samples in the feature space, we can use unlabeled samples to bridge gaps between the labeled samples and test samples. A visual example demonstrating where unlabeled data would be extracted relative to a test region is presented in figure 12.

The second option that we explore to collect unlabeled data uses known training points with ambiguous labels (e.g. weak mines that are more similar to clutter

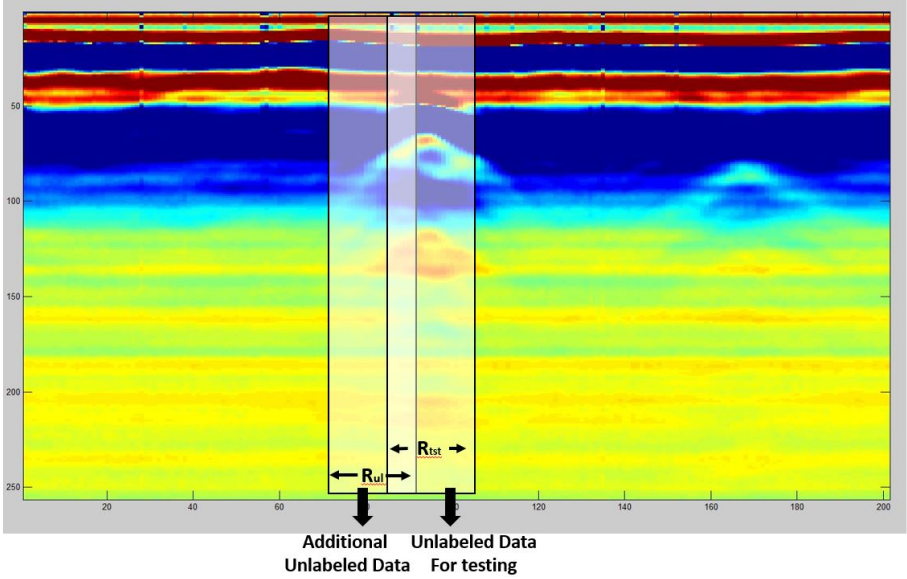

Figure 12. Example of how contextual unlabeled data can be extracted. $\mathbf{R}_{tst}$ is the region from which features are extracted for testing. $\mathbf{R}_{ul}$ is where additional unlabeled data will be extracted to aid LP in label assignment.

or clutter signatures that look like mines) as unlabeled data. We hypothesize that using these points as unlabeled data, despite knowing their actual labels, provides more flexibility by using them as ambiguous samples. Instead of committing them to a specific label, we let them act only as bridges connecting test data to representative prototypes.

3.5 Discrimination Algorithms

Our experiments use instance-based classification methods. Specifically, we explore the use of KNN and several variations of the LP classifier as detailed in section 2.4.

Our system operates in two different modes. The first is as a discriminative classifier to accept or reject alarms that have been reported by a prescreener algorithm. This is currently the only option for data collected using a vehicle mounted GPR system. In the other mode, we use the proposed algorithm as a prescreener

that processes all raw data and flags potential regions. This scenario is specific to handheld datasets.

3.5.1 Discriminative Classifier Mode

As a discriminative classifier, our system receives as input a set of alarms identified by a prescreener. A subset of these alarms is used to train the classifier and learn its optimal parameters. The remaining subset is used for testing and validation. We experiment with this setting on both vehicle and handheld datasets. We compare the performance of the different variations of the proposed LP algorithms to the prescreener that generated the alarms. We also compare our results to those generated by a KNN classifier receiving the same set of alarms.

3.5.2 Prescreener Mode

We demonstrate the feasibility of our system in a prescreener setting due to its computational efficiency. In this setting, we use a different prescreener that uses a simple condition to extract a set of alarms to train our system. This data is then summarized to identify a set of learned prototypes.

After learning and labeling the prototypes, we begin prescreening our test dataset to identify target locations. The input to our prescreener is raw GPR data. At each location, we extract features for the test samples and features from samples that precede it as unlabeled data. Our prescreener will then assign a confidence value to each spatial location in the raw GPR data.

Each of the samples tested by the prescreener has a corresponding UTM coordinate. Figure 13 shows what this looks like from a bird's eye perspective as the operator swings the device. Each UTM coordinate has a confidence value assigned by the prescreener. The circle is the "halo" region representing the surface area under

which a target is buried. Any high confidences inside this area are considered a "hit".

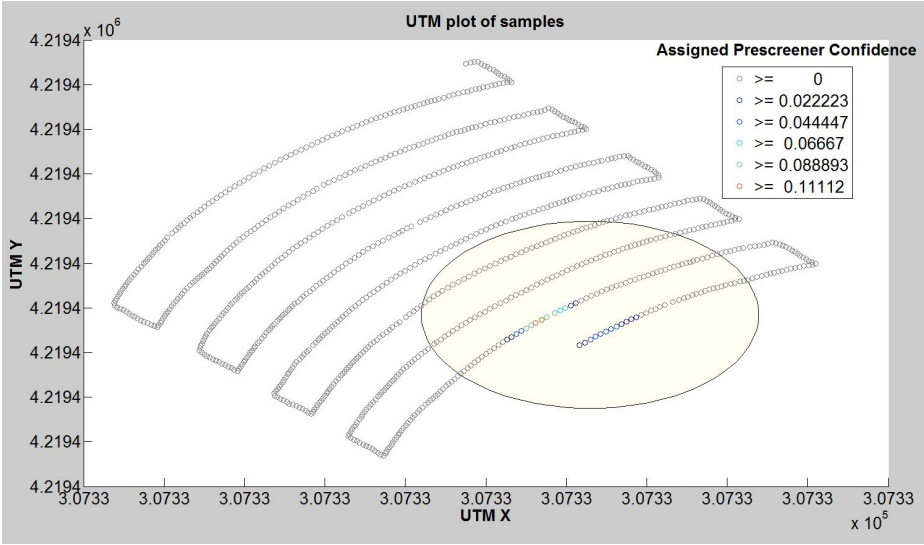

Figure 13. Positional plot of prescreener assigned confidences by UTM coordinate.

After a confidence has been assigned at each position within the GPR data, we cluster the confidences based on their spatial location to reduce the number of alarms outputted to a classifier.

To evaluate the performance of the proposed LP prescreener, we compare it to a computationally efficient prescreener which we use to collect our training data. We also compare LP to a KNN prescreener which uses the same learned prototypes.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, we validate our proposed system on data collected using both vehicle and handheld GPR systems. We first demonstrate the performance of LP on vehicle datasets, and then explore various settings and experiments as outlined in chapter 3. Next, we detail the implementation of our system on an experimental handheld GPR system. We follow the same experimental analysis as per our vehicle settings, but using data collected with a handheld system. For this application, we look into using LP as both a discriminative classifier and as a prescreener, and compare those options with a similar KNN-based system as well as a simple rule-based prescreener. We use Receiver Operating Characteristic (ROC) curves [39] to analyze the performance of our classifiers and compare them to existing algorithms. An ROC plots the accuracy rate of a classifier against its false alarm rate for all range of values assigned by the classifier. This allows classifiers to be compared at different tolerances.

4.1 Landmine Detection Using Vehicle Mounted GPR array

We first investigate optimizing our proposed LP classifier on datasets collected using vehicle mounted GPR. Vehicle-based GPR landmine detection systems are more mature and stable than those of handheld GPR. Therefore, we first demonstrate the validity of our system on these datasets.

TABLE 1

Statistics of the Datasets

	Area (m^2)	Site	Number of Targets	Number of Lanes	Number of Alarms
Dataset 1	28238.5	1	1738	5	1930
Dataset 2	28210.4	1	1736	5	2346
Dataset 3	113286.2	2	1377	8	1847
Dataset 4	113059.7	2	1376	8	2286

4.1.1 Datasets

We performed our experiments on vehicle-mounted GPR using four datasets. All of these datasets were collected in 2015. Each of these datasets was obtained with a GPR device collecting signal data over 51 channels.

The first dataset spanned 28238.5 square meters covering 1738 targets. The second set used a different version of the sensing hardware, and spanned 28210.4 square meters covering 1736 targets from the same site. The third dataset was collected from a different site than the first two sets. This dataset spanned 113286.2 square meters and covers 1377 targets. The fourth and final dataset was taken from the same site as the third dataset, but spanned 113059.7 square meters covering 1376 targets, and used the same hardware as dataset 2. Each site contained a variety of targets which included no-metal, low-metal, and high-metal objects. These objects range from inactive landmines to miscellaneous clutter. An adaptive LMS prescreener [9] was run on each of these datasets to obtain a set of alarms, which we then tested independently using 10-fold cross validation. By dividing alarms from each lane into 10 folds, we could train and test using the same dataset.

4.1.2 EHD Feature Extraction for Vehicle Mounted GPR

In section 3.3, we explained our feature extraction as applied to handheld GPR. Our vehicle implementation used a similar EHD feature representation, but executed

on 3-D GPR data. Because the vehicle GPR spans 51 channels of signal data, we used an identical implementation as in other vehicle-based systems that use EHD features [19], which capture edge information within a 3-D cube of GPR data per alarm. A visual example of this is presented in figure 14.

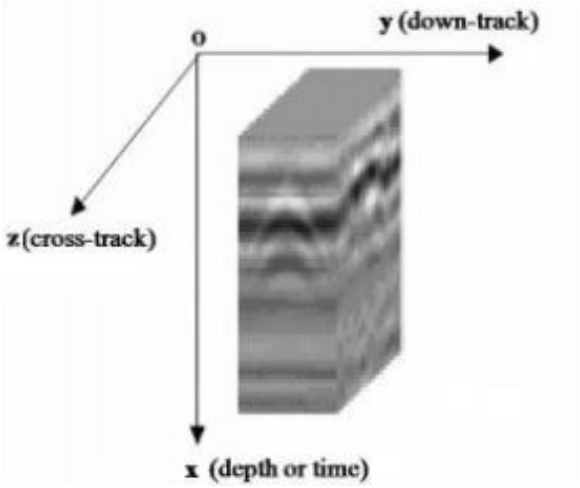

Figure 14. Visual example of 3-D GPR data.

Let $S_{xy}^{(z)}$ be a downtrack 2-D signature similar to S_{xy} from section 3.3, along the zth channel, or plane, of a 3-D vehicle signature $S(x, y, z)$, spanning 15 scans. Figure 15 shows a visual example of this kind of signature. We compute $\text{EHD}(S_{xy}^{(z)})$ using 7 overlapping windows, and subsequently obtain 7 histograms: $H_{xy_i}^{(z)}$, $i \in 1, \ldots, 7$. We compute these histograms over N_c channels, giving us N_c downtrack EHD feature vectors. We then obtain an overall downtrack EHD^d by averaging the individual channel EHD feature vectors, giving us

$$\text{EHD}^d(S_{xy}^{(z)}) = [\bar{H}_{xy_1}, \bar{H}_{xy_2}, \bar{H}_{xy_3}, \ldots, \bar{H}_{xy_7}] \tag{15}$$

where $\bar{H}_{xy_i}$ is the cross-track average of the ith window over N_c channels calculated by

$$\bar{H}_{xy_i}^{(z)} = \frac{1}{N_c} \sum_{z=1}^{N_c} H_{xy_i}^{(z)}. \tag{16}$$

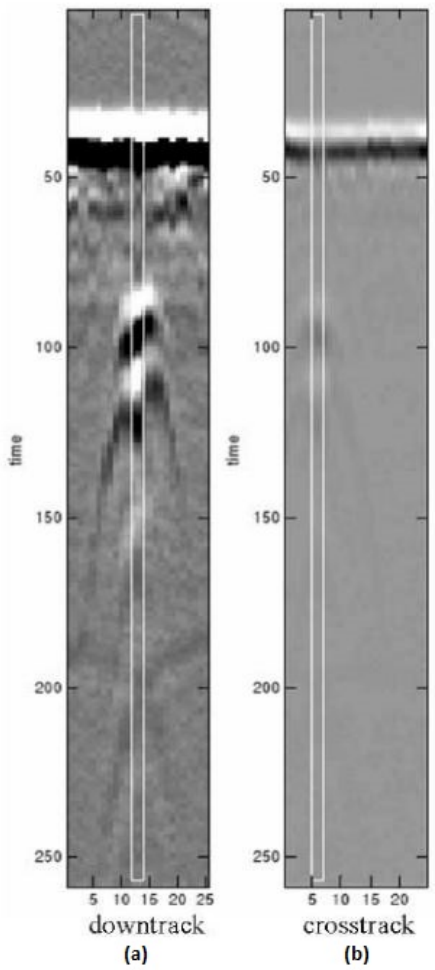

Figure 15. Visual example of (a) downtrack and (b) crosstrack b-scans taken from GPR data.

Each of the 7 downtrack histograms is a 5-dimensional EHD feature vector, giving us a 35-dimensional downtrack EHD feature vector.

We followed a similar process to calculate the cross-track component of our EHD feature vector EHD^c. We fixed the scans to obtain multiple cross-track 2-D signatures $S_{xz}^{(y)}$, $y = 1, ..., N_s$. Due to the low resolution across channels, we take only one histogram per signature. Finally, we obtained our cross-track component by averaging over the signatures

$$\bar{H}_{xz}^{(y)} = \frac{1}{N_s} \sum_{y=1}^{N_s} H_{xz}^{(y)}. \tag{17}$$

After concatenating the down-track and cross-track EHD components, the final EHD feature vector of our 3-D alarm $S_{xy}^{(z)}$ is a 40-Dimensional vector.

$$\text{EHD}(S_{xyz}) = [\text{EHD}^d(S_{xyz})\ \text{EHD}^c(S_{xyz})] \tag{18}$$

4.1.3 Label Propagation

We compare the proposed LP discriminator with the possibilistic KNN classifier. Both classifiers learn a set of prototypes from the training data. For this step, we use the Self-Organizing Maps (SOMs) [49] that is used within the KNN classifier [19]. In addition to learning the prototypes, this approach also learns two independent sets of fuzzy labels. The first label can be considered as the posterior probability of the prototype in the class of targets and the second one as the posterior probability in the class of clutter.

4.1.4 Graph Construction

The next step of our LP classifier is to construct the graph. We explored several methods for this task as detailed in section 3.4.4. The first method involved varying σ used in equation (1) experimentally. Using dataset 1, we tried σ from 0.01 to 0.1 in increments of 0.01. For each σ, we test the LP classifier using 10 fold cross-validation. Our results have indicated that using $\sigma = 0.1$ yielded the best results.

Next we considered the heuristic for choosing σ from [46]. We used the Kruskal algorithm for constructing minimum spanning trees (MST) to create an MST over all labeled data. Then, we found the smallest edge d connecting a target node to a clutter node, and set $\sigma = d/3$.

We applied this method for each cross-validation fold. Thus, sigma can vary between folds. Ideally this will optimize the similarity matrix $\mathbf{W}$ from equation (1) to

each fold's training set independently. We found that using a constant value for sigma worked best. We experimentally found that 0.1 seemed to show the best results.

Finally, we explored a KNN graph implementation as outlined in section 3.4.4. We experimented with different values of k ranging from 5 to 15 by an increment of 5, and evaluated performance on dataset 1. We found that the results of using a KNN graph were close, but using $\sigma = 0.1$ had better performance. Thus, for the rest of our experiments we proceeded using a σ based graph.

4.1.5 Analysis of the different LP variations

We next compared the different implementations of LP described in section 2.4.2. Up until now our LP experiments have been done exclusively using the closed form LP from section 2.4.2.4 due to its speed. In this section, we compared the closed form LP version with each of the other algorithms listed in section 2.4.2. For each of the iterative algorithms we use 100 iterations, as most of the test points converged to under a threshold $\epsilon < 0.01$ in about 25 iterations. For LP based on Jacobi Iteration, we determined experimentally that the best performance was observed with $\alpha = 0.05$, which places more emphasis on the initial labeling of the prototypes than on the topology of the data. Contrarily, we found that LP based on Label Spreading performs better with a higher value of alpha. Specifically, we found that $\alpha = 0.95$ performs best.

Figure 16 compares the ROC's of all 4 versions of the LP algorithms. As it can be seen, all algorithms, except for LP based on Label Spreading, have similar performance. We thus continued our experiments using the closed form LP, as it is the most efficient and does not require convergence criteria.

Finally, we explored the use of additional unlabeled data. We implemented and tried each of the settings described in section 3.4.5. We compare these settings with

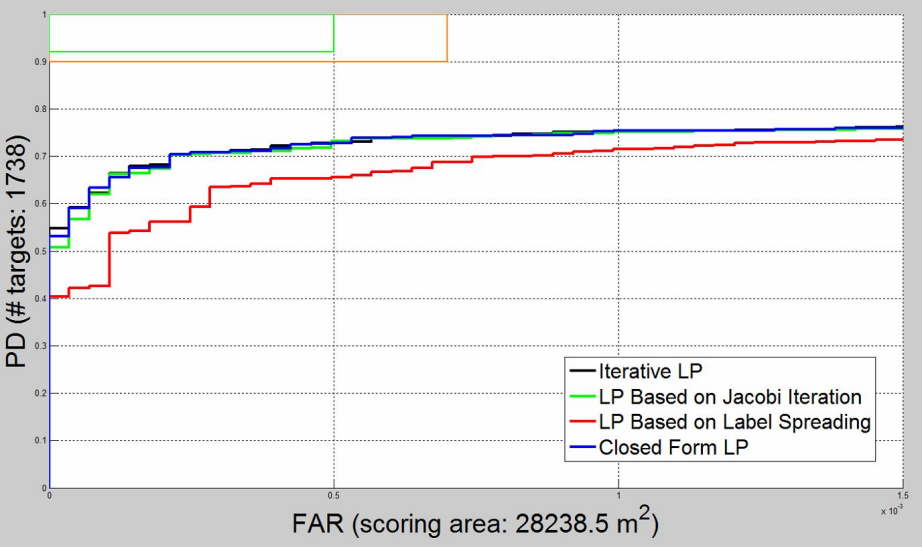

Figure 16. ROC's obtained using all 4 LP algorithms on Dataset 1: Iterative LP, LP based on Jacobi Iteration, LP Based on Label Spreading, and Closed Form LP.

an LP classifier that uses no additional unlabeled data (which we refer to as LP1), and an LP classifier that tests all (10) signatures collected at a particular sample at once (LP10). The results are shown in figure 17. While most of the options did not affect the results significantly, we did observe a slight improvement when we used a large number of additional context data. Thus, we reported results on the vehicle GPR using the context as additional unlabeled data.

With our optimized settings, we compared the performance of the standard KNN algorithm [19] and the proposed LP classifier on all of our vehicle datasets. We observed in figure 18 that LP either improves upon or matches a possibilistic KNN classifier on all datasets.

4.2 Landmine Detection Using Handheld GPR device

Having proved the efficacy of LP on vehicle GPR data and its improvement over a possibilistic KNN classifier, we will repeat the above experiments on a handheld GPR dataset. Unlike the vehicle GPR, the handheld device is still in its early stages

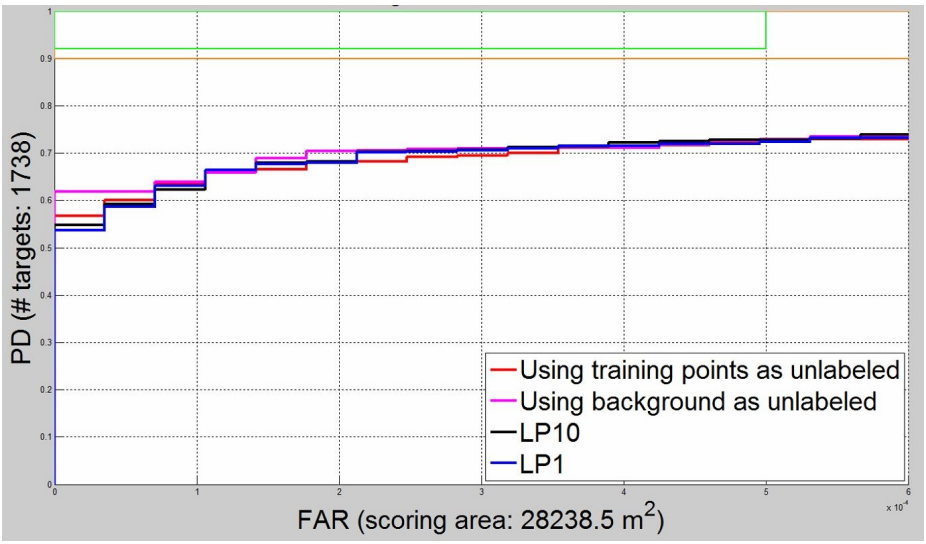

Figure 17. ROC's obtained by incorporating additional unlabeled data into the LP algorithm to aid in classification on Dataset 1.

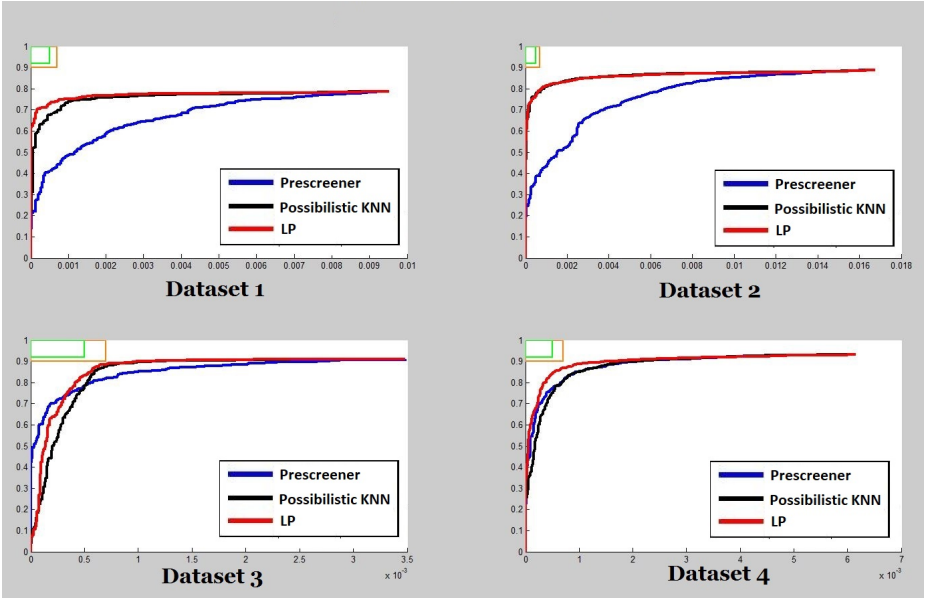

Figure 18. Results from applying a prescreener (blue), possibilistic KNN classifier (black), and LP classifier (red) on 4 datasets.

of development and no standard prescreener or algorithms have been integrated into this system. Hence, we devised our own methods of prescreening, feature extraction, prototype learning/training, and labeling methods prior to comparing KNN and LP classifiers. Afterward, the same experiments as from section 4.1 are run and evaluated.

4.2.1 Datasets

The dataset for the handheld device consisted of raw GPR signal data collected from one channel. Scans were collected as the operator moved the sensor down-track. Each down-track position contains 256 time samples. The down-track scans were uniformly sampled to have one scan per 1 cm. Similar to the vehicle datasets, we used 10-fold cross-validation to perform both training and testing on the same data. The dataset that we use to validate our algorithms was obtained in 2015 on a site spanning 919.8 square meters and containing 350 targets.

4.2.2 Prescreener

We devised a simple conditional prescreener on our handheld datasets to act as a baseline for comparison of our classifiers and as a method of reducing our data for training purposes. This prescreener operates in the same manner as the system described in section 3.5.2. The only difference is the way in which it calculates labels for extracted signatures.

Referring to figure 11, each of the four portions of our EHD feature vector has a measured amount of horizontal, vertical, diagonal, and anti-diagonal edges. These measurements are denoted as h, v, d, and a, respectively, followed by the number of the portion from which that measurement was taken. Thus, it follows that h_1, v_1, d_1, and a_1 represent the first four features of EHD(S_{xy}).

Our simple prescreener employed a condition that can detect the hyperbolic shape of a mine signature. Figure 19 shows an example of a signature to which a prescreener should assign a high confidence value. From this figure, we observe the high diagonal in the first two portions of the signature, denoted by d_1 and d_2, and the low antidiagonal a_1 and a_2 indicating the presence of an upward slope. We also observe the low diagonal in the last two portions denoted by d_3 and d_4, and the

high anti-diagonal a_3 and a_4, indicating a downward slope in the latter portions of the signature. The presence of both conditions indicate that a hyperbolic shape is present within the signature.

Our prescreener used the above idea to determine if a hyperbolic shape is present within an EHD signature. Specifically, it checks to see if the signature meets several conditions. These conditions evaluate whether the diagonal edge in either of the first two portions of the signature is greater than the amount of anti-diagonal, or that the opposite is true in the latter two portions. If this condition is passed, then

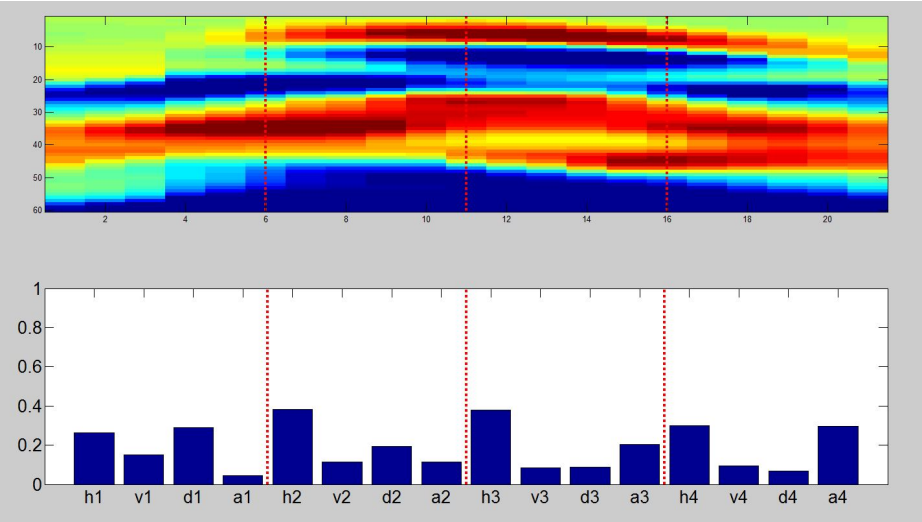

Figure 19. Example of mine signature with high diagonals d_1 and d_2 and anti-diagonals a_3 and a_4.

the product of the diagonal edge on the left subimage with the antidiagonal edges on the right subimage can be used as an indication of the likelihood of a target. The resulting prescreener is outlined in algorithm 5.

Algorithm 5: EHD Prescreener

1: **if** $((d_1 > a_1) + (d_2 > a_2) + (a_3 > d_3) + (a_4 > d_4)) < 2$ **then**
2: $\quad c = 0$
3: **else**
4: $\quad c = (d_1 + d_2) * (a_3 + a_4)$
5: **end if**

We see from line 1 of algorithm 5 that each of the four possible conditions yields a logical 1 if the condition is passed and 0 if it is not. In figure 19, we can see that each of these conditions would pass, yielding a result of 4 which passes the required threshold of 2. If a signature does not pass this requirement, the confidence will be set to 0. If it passes, we use the calculation in line 4 to assign a confidence value to the signature. If this signature has both a high amount of diagonal edge activity in the first two portions of the signature, and a high amount of anti-diagonal edge activity in the last two portions, it will be assigned a high confidence value.

In our experiments we used this conditional prescreener as a baseline with which to compare our more sophisticated classifiers. When run on our handheld dataset, it produced 3185 alarms.

4.2.3 Training Methods

We learned our prototypes using the two methods discussed in section 3.4. We took the training samples outputted from the EHD prescreener, separated them into targets and clutter according to the ground truths, and then passed these as input into either of our training algorithms.

We first extracted all signatures from our alarms using our feature extractor described in section 3.3. We then used our prescreener from section 4.2.2 to assign a confidence to all signatures from all alarms. Each of our training methods takes a different approach to identifying the mine signatures within the positive alarms.

Our first method keeps only the signatures with assigned confidences higher than a threshold t_{mine}. We applied a similar thresholding to the false alarm instances to reduce the number of repetitive and noisy signatures. We used an upper threshold t_{upper} and required that false alarm instances have a prescreener assigned confidence less than this value, as well as a lower threshold t_{lower} which they must be above. For

our experiments, we selected our mine threshold $t_{mine} = 0.04$, and our false alarm thresholds $t_{upper} = 0.1$ and $t_{lower} = 0.01$.

Our second method combined MIL and KDE to assign density values to instances within bags (alarms). All instances belonging to negative bags are summarized by few prototypes. To maintain diversity within the clutter and background objects, we first divide the clutter data into 7 groups according to the strength of their diagonal and antidiagonal edges. After the instances have been separated, the first group contains all instances with low edge activity, and the last group contains all instances with very high amounts of edge activity. Finally, we clustered each group independently using SOM to reduce the number of observations per group to 100.

Each of these groups is then independently used to estimate the Kernel Density of each of the positive bags. The densities are then summed across groups to determine which instances within each positive bag are of low density, and thus have a high likelihood of being mines. Experimentally, we found that for the handheld dataset, summing the densities estimated using only the two groups with the weakest (least edge activity) false alarms gave the best results in establishing a contrast in the kernel density between positive and negative instances in positive bags. An example illustrating the advantage of combining these two groups on our handheld dataset is presented in Figure 20. In this figure, brighter pixel color indicates higher density. A high density means that a signature extracted at that location was estimated as clutter.

After calculating the densities of all observations belonging to all bags, we thresholded observations from positive bags by density using a threshold $t_{density}$. We used $t_{density} = 0.05$. The remaining observations were thresholded once more using the same thresholding as in method 1.

Finally, we clustered the results of each of our training methods to obtain our

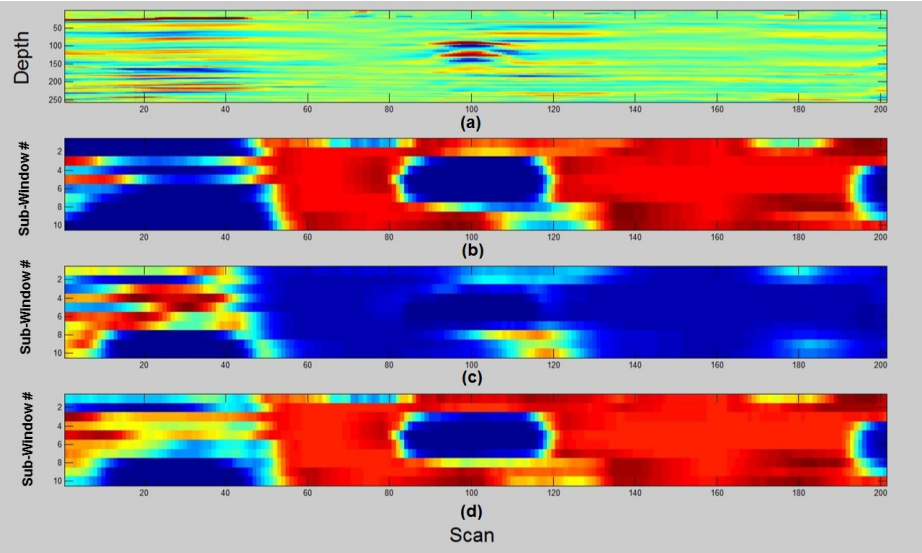

Figure 20. Example of combined kernel density for a target: (a) Preprocessed image. (b) KDE estimated using first group of negative clusters. (c) Using second group. (d) Sum of (a) and (b).

final set of representatives. Our first method used Hierarchical Agglomerative Clustering (HAC) on both the positive and negative observations to reduce the number of prototypes to 100 clutter and 100 mine representatives. Our second method used SOM which also provided us with 100 mine prototypes and 100 clutter prototypes, giving us 200 total representative prototypes constituting our outputted prototypes used as labeled data during classification.

After deciding our method of learning representative prototypes, and our method of classifying test points, we still required a way of summarizing the assigned confidences to test points (sub-windows) within an alarm, in order to assign it an overall confidence. We did this by assigning the alarm a confidence equal to the average of the 3 sub-windows with the highest confidences.

We compared the quality of our prototypes by classifying our test points using a crisp KNN classifier with $k = 10$, similar to the KNN classifier used on the vehicle datasets. The results of using this classifier are presented in figure 21. We perceived no significant difference in the results between our two training methods, and thus

proceeded using our HAC based training method, as it is the faster and simpler of the two training methods.

We also found that varying the number of prototypes and value of k yielded no definitive improvements over our original choices of $k = 10$ with 200 total prototypes.

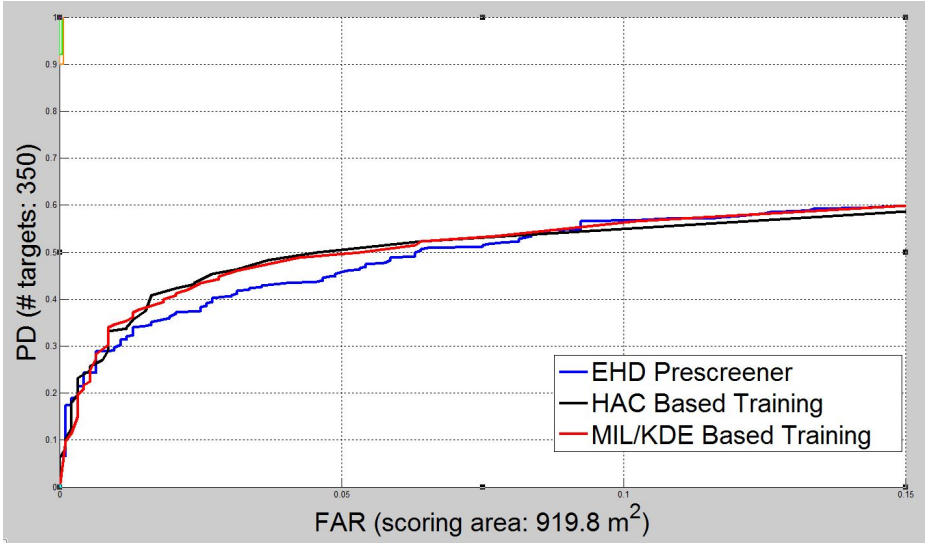

Figure 21. Results of applying training methods from section 3.4 to learn prototypes that are used for a KNN classifier on a handheld dataset.

4.2.4 Fuzzy and Possibilistic Membership

In another experiment, we evaluated the results of using our fuzzy labeling methods described in section 3.4.3 by comparing them on a KNN classifier using the same settings that were used for the classifiers from section 3.4. The results of these labeling methods are presented in figure 22. The differences in performance are slight, but we concluded that using KNN based labeling resulted in the best performance. Thus, for all remaining experiments, we used this as our labeling method for our LP classifier.

We also compared a possibilistic KNN classifier with our fuzzy LP classifier. During data summarization in our training methods we learned the mean and standard deviation of each prototype's cluster, μ_c, and σ_c. During testing, to reduce the

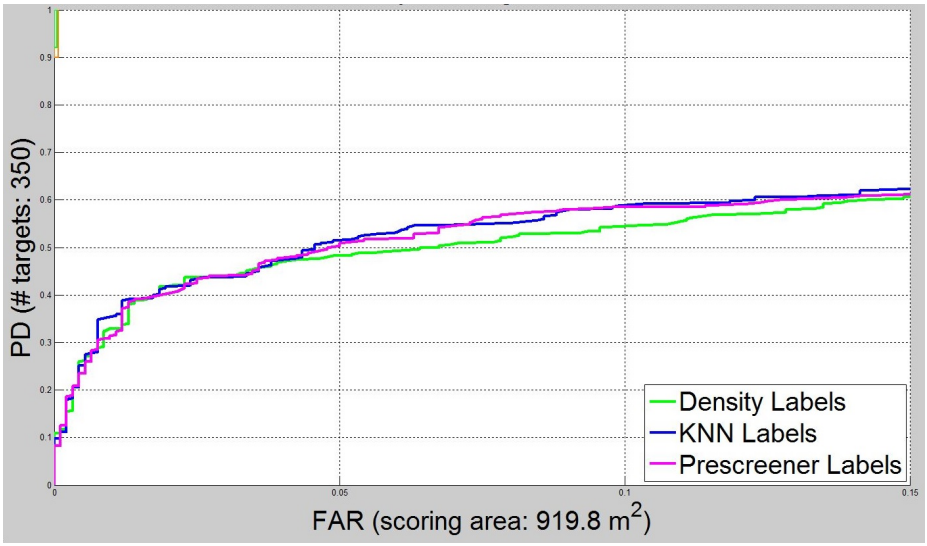

Figure 22. Comparison of 3 prototype labeling methods for the KNN classifier.

effect of noise and outliers, we weighed the contribution of each nearest neighbor. To calculate the weight for the kth nearest labeled point x_k^l to a particular test point x_u, we used

$$w(x^u, x_k^l) = \frac{1}{\frac{max(0,|d_k - \mu_c|)}{\sigma_c * m} + 1} \tag{19}$$

In (19), d_k is the Euclidean distance between x^u and x_k^l, and m is a constant used to amplify or dampen the effect of σ_c. Through experimentation, we found that $m = 6$ seemed to yield the best classification results. Equation 19 is meant to reduce the significance of a labeled prototype that is too far from the test sample even when the prototype is among the k nearest neighbors to the test point.

Finally, the weight of the kth nearest labeled point is multiplied by its class label. These labels are then summed over all k nearest labeled points using

$$y^u = \sum_{i=1}^{k} y_i^l * w(x^u, x_i^l) \tag{20}$$

where y^u is the confidence value assigned to the test point, and y_i^l is the unweighted label of the ith nearest labeled prototype.

We combined our KNN and density labeling methods with our possibilistic

labeling method on our KNN classifier. Our KNN labels are more positive for mine prototypes, and more negative for false alarm prototypes. Our density labels are vice versa, so we combine them using

$$y^u_{(combined)} = y^u_{(KNN)} * (1 - y^u_{(density)}) \tag{21}$$

to obtain our final assigned label for the test point x^u.

Figure 23 shows that possibilistic labeling performed similarly to fuzzy labeling using KNN labels. Thus, we continued our experiments using fuzzy KNN labels, due to its simplicity.

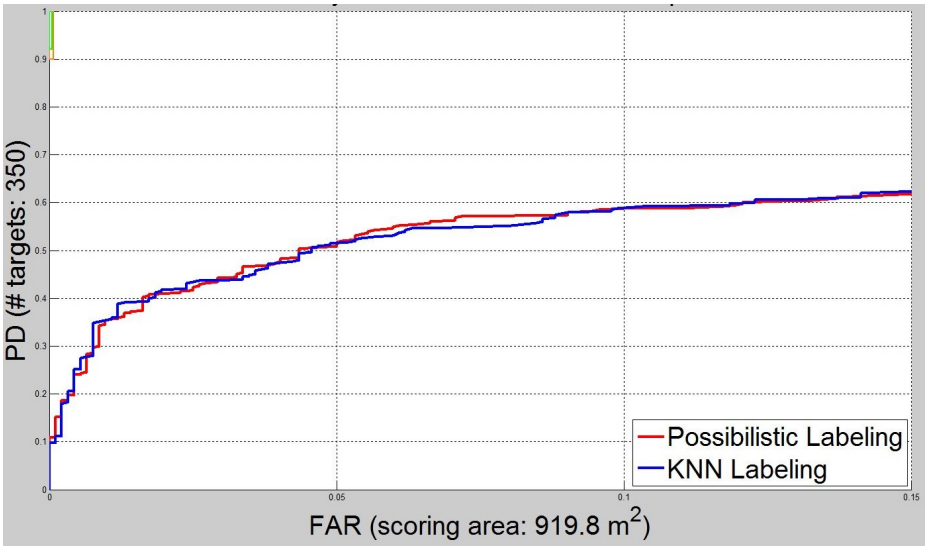

Figure 23. Comparison of results from using Possibilistic and Fuzzy Memberships.

After deciding upon the use of KNN based fuzzy labels as the optimal labeling method for our KNN classifier, and our HAC based training method for learning prototypes, we applied these settings to an LP classifier.

4.2.5 LP Experiments

Our first experiment using LP on our handheld dataset sought to find an appropriate value for σ from equation (1). Similar to section 4.1, we experimentally

chose a value for sigma. We found that $\sigma = 0.01$ performed best on our handheld dataset. We once again apply our MST heuristic, as well as a KNN tree graph method, which we determined experimentally performed best with $k = 10$, and compared the ROCs of each and found that using $\sigma = 0.01$ was the best option. The results of these experiments are presented in figure 24.

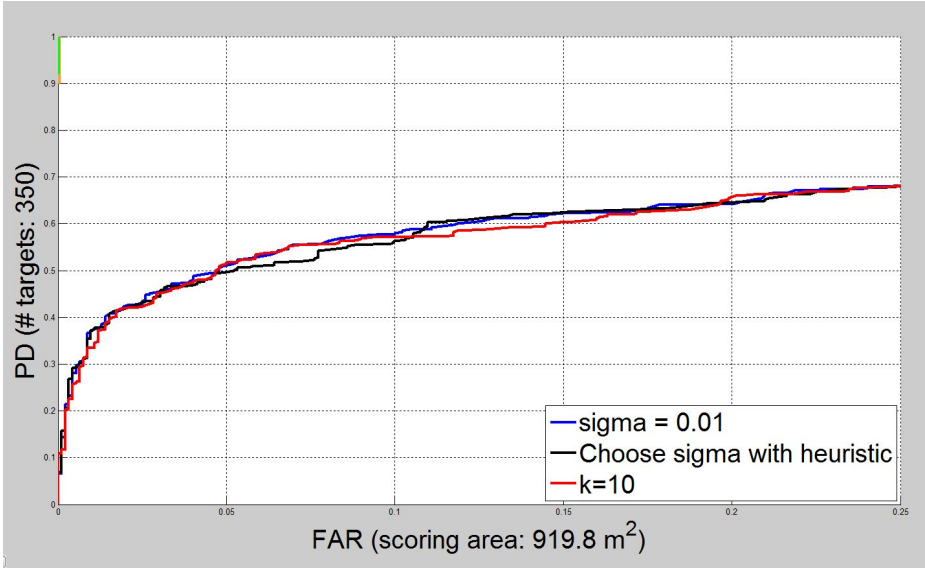

Figure 24. Comparison of experimentally selecting σ, heuristically choosing σ, and using a KNN tree as our graph, for our LP classifier.

Similar to our vehicle GPR experiments, we applied each of our different LP algorithms from section 2.4.2 on our handheld dataset. Once again we found that all LP algorithms performed equally well, with the exception of LP based on Label Spreading, which performed worse.

We experimented with including additional unlabeled data to aid our LP classifier, similar to our vehicle experiments. The results are presented in figure 25. We found that the inclusion of unlabeled data did not have a significant effect on the results of testing, and hence we continue without using unlabeled data.

Comparing the best of our KNN and LP experiments, we obtained the results in figure 26. We see that while LP did not definitively outperform the KNN classifier,

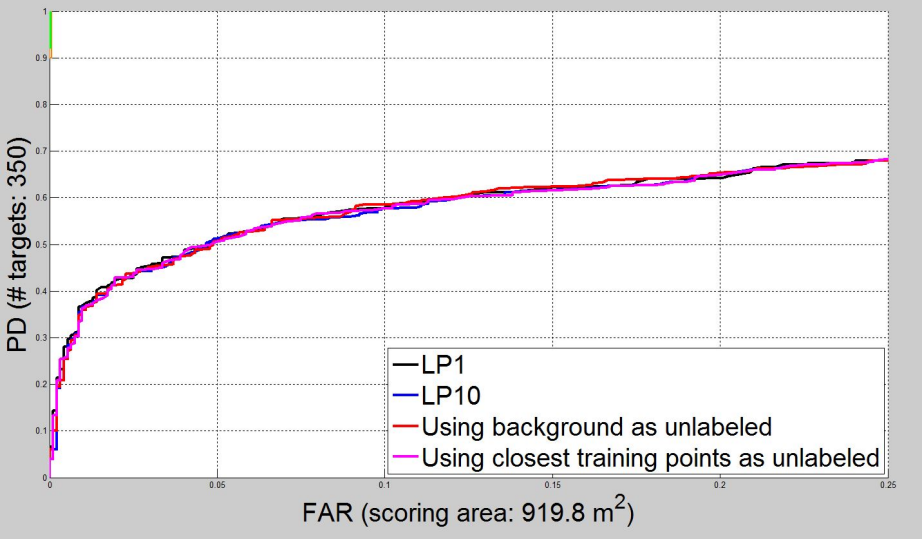

Figure 25. Results of including unlabeled data with LP on handheld dataset.

it still matched it in performance for this dataset.

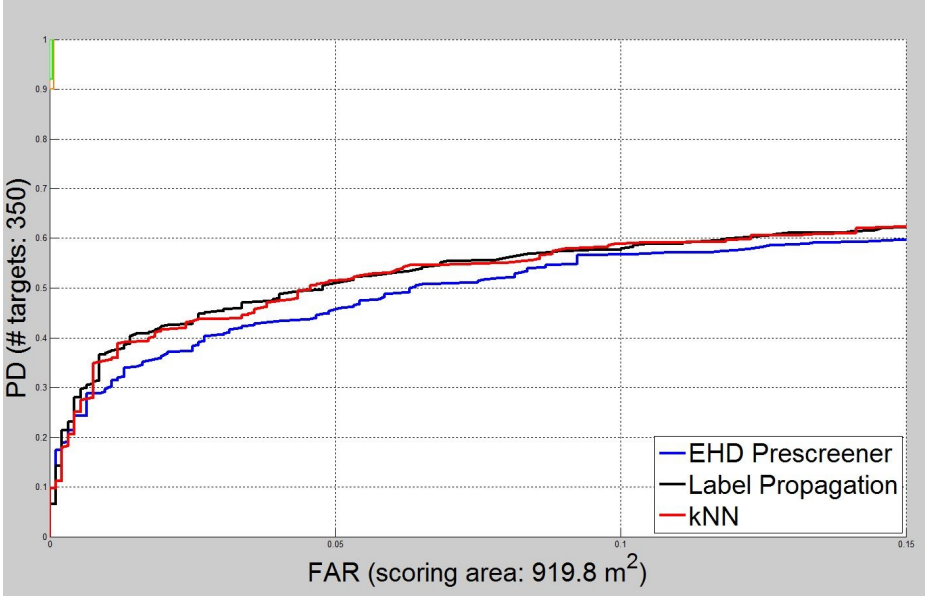

Figure 26. Results of comparing best KNN and LP classifiers.

4.2.6 LP as a Prescreener

We also applied our LP classifier to our handheld dataset as a prescreener instead of as a classifier. For this experiment, we obtained our prototypes from a

separate handheld dataset. The dataset that we used for training was collected in 2014, covered 441.7 square meters and contained 174 targets.

We kept all of our settings the same as what was used for discriminative classification, for both training and testing. However, we did adjust our thresholding during training in order to obtain lower confidence mines, since we tested over the entire GPR data, and not just locations with relatively high energy. Specifically, we changed the mine threshold to $t_{mine} = 0.01$.

We applied both a KNN and LP prescreener using the same settings as before, provided both prescreeners with the same set of prototypes, and compared performance with the conditional prescreener from section 4.2.2.

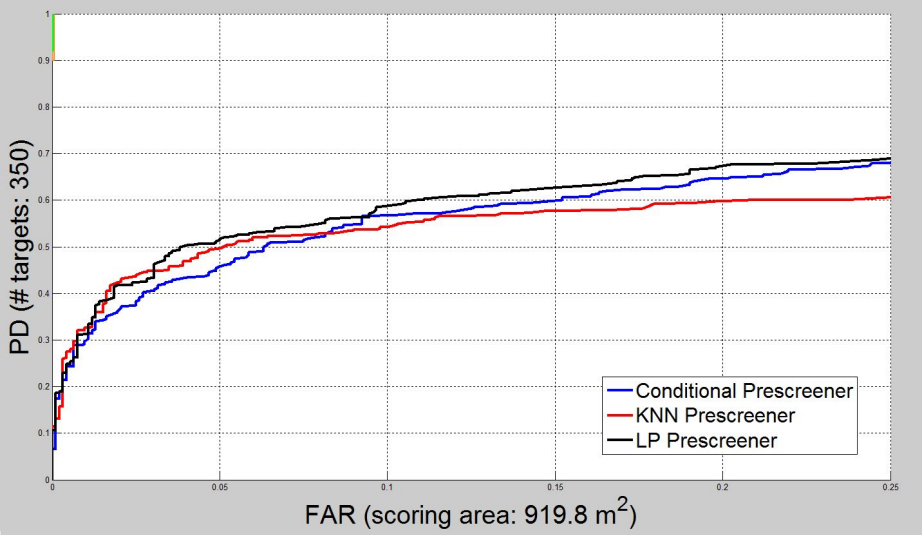

Figure 27. Results of comparing KNN and LP as prescreeners.

From figure 27, we saw that using LP as a prescreener outperformed both the conditional prescreener and KNN prescreener.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this work, we have proposed a landmine detection method based on Label Propagation classification for application in both vehicle and handheld systems.

We hypothesized that an LP classifier would outperform an identically trained KNN classifier due to its utilization of additional information about labeled and unlabeled data.

In our vehicle system, we propose an LP system that performs the following steps: First, raw GPR data are normalized via preprocessing. Next, a preselected computationally inexpensive prescreener identifies regions of interest within the GPR data. EHD features are then extracted from these areas of interest and divided into training and testing sets. Additional EHD features are extracted from a small context around these areas to be used as unlabeled data for our LP classifier. During training, our system summarizes the training data by reducing them to a set of representative prototypes via Self-Organizing Maps to characterize mine and clutter classes.

During testing, our classifier uses closed-form Label Propagation to calculate and assign labels to test points using the training prototypes and unlabeled data.

In our handheld system, we propose a discriminative system that operates similar to the vehicle LP system but with slight differences. We propose a new simple prescreener that uses EHD features to assign confidence values to samples. Our training method uses simple thresholding on the outputted confidence value

assigned by our new prescreener followed by Hierarchical Agglomerative Clustering to summarize training data. Finally, we label our summarized prototypes with a nearest-neighbor approach, detailed in section 3.4.3.

Lastly, we proposed a prescreener alternative to our handheld system, which follows all of the same steps as our handheld discriminative classifier, but takes as input the outputted trained prototypes learned from a separate dataset. In this setting, our handheld LP classifier calculates confidence values for each sample in a dataset.

Our results show that, if properly optimized, a Label Propagation classifier can consistently match or outperform a KNN classifier when using the same set of trained prototypes, if their features accurately convey the data. On the vehicle datasets, we determined that most iterative LP algorithms will converge to the same value as equation (8). We note that the most important factor in optimizing LP is choosing the correct value of σ. We also found that using a context of background data as unlabeled points can offer a slight improvement in LP performance.

On our handheld dataset, we noticed that the most important factor affecting performance is the manner in which representative prototypes are selected during training. Specifically, we found that thresholding the input data to our training methods had the largest effect on the performance of both the KNN and LP classifiers.

Our handheld experiments did not yield much variation in performance. Because the handheld datasets were collected with an experimental system, it is likely that the data are not as reliable, or further preprocessing is required to reduce the prevalence of noisy features.

As a prescreener, we found that LP performed well. This may suggest that LP performs better at rejecting false alarms than identifying mines, as the ratio of clutter to mines in prescreening is much larger than it is during classification.

Ultimately, it appears that the graph-based nature of the LP is its primary advantage that allows it to outperform possibilistic KNN classification. We noticed either slight or negligible improvements from using unlabeled data. There are several possibilities as to why additional unlabeled data did not improve the classification.

One likely explanation has to do with our application itself. In several of the applications listed in section 2.4.2, LP was used for object detection in video data. In these applications, unlabeled data were used to bridge test points from one frame to known points from an earlier frame. All frames in between are used as unlabeled data. In this case, each labeled point has a different class label and the unlabeled points are used to assign the most likely class label by propagating the labels between frames.

In our landmine detection application, targets are encountered only once. It is the equivalent of an object in a video appearing in only one frame. Unlabeled data in this case is being used to bridge between different ground truths. This may mean that there is not enough similarity between unlabeled data and labeled data to establish a bridge between test points and similar targets.

5.2 Potential Future Work

On the vehicle datasets, we suggest further analysis into why the LP outperforms the KNN. It seems that LP's graph-based nature is the primary contributor to the success of the classifier, so it may be worthwhile to explore other graph and network-based classification methods.

On the handheld datasets, raw GPR data preprocessing and feature representation should be refined. The data are too noisy to rely on subtle differences between features to distinguish between classes. Instead, our classifiers seemed to rely on simple and highly generalized patterns. As a result, our prototypes often tended to

be different variations of the same patterns, rather than unique signatures. Cleaning the GPR data will help to bring out the more subtle differences between signatures and allow more sophisticated feature representations to have greater use and be less prone to noise.

REFERENCES

[1] *A Survey of Current Sensor Technology Research for the Detection of Landmines*, volume 6, Zagreb, Croatia, Sept-Oct 1997.

[2] J. MacDonald, J.R. Lockwood, and L. Carin. *Alternatives for Landmine Detection.* RAND, Santa Monica, CA, 2003.

[3] M. P. Kolba and I. I. Jouny. Clutter suppression and feature extraction for land mine detection using ground penetrating radar. In *Antennas and Propagation Society International Symposium, 2003. IEEE*, volume 2, pages 203–206 vol.2, June 2003.

[4] Robert C. Doheny, Sean Burke, Roger Cresci, Peter Ngan, Richard Walls, and Jeff Chernoff. Handheld standoff mine detection system (hstamids) field evaluation in namibia, 2006.

[5] Luc M. van Kempen, Hichem Sahli, J. Brooks, and Jan P. Cornelis. New results on clutter reduction and parameter estimation for land mine detection using gpr, 2000.

[6] D. J. Daniels. A review of landmine detection using gpr. In *Radar Conference, 2008. EuRAD 2008. European*, pages 280–283, Oct 2008.

[7] K. C. Ho, L. M. Collins, L. G. Huettel, and P. D. Gader. Discrimination mode processing for emi and gpr sensors for hand-held land mine detection. *IEEE Transactions on Geoscience and Remote Sensing*, 42(1):249–263, Jan 2004.

[8] David J. Daniels. *Ground Penetrating Radar, Volume 1*, volume 1. Institution of Electrical Engineers, London, United Kingdom, 2 edition, 2004.

[9] P. A. Torrione, C. S. Throckmorton, and L. M. Collins. Performance of an adaptive feature-based processor for a wideband ground penetrating radar system. IEEE Trans. Aerospace and Electronic Systems (in press).

[10] H. Frigui and P. D. Gader. Detection and discrimination of land mines based on edge histogram descriptors and fuzzy k-nearest neighbors. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, Vancouver, BC, Canada, July 2006.

[11] C. R. Ratto, P. A. Torrione, and L. M. Collins. Exploiting ground-penetrating radar phenomenology in a context-dependent framework for landmine detection and discrimination. *IEEE Transactions on Geoscience and Remote Sensing*, 49(5):1689–1700, May 2011.

[12] Peter Torrione, Kenneth Morton, Jr., and Lance E. Besaw. Adaptive gaussian mixture models for pre-screening in gpr data, 2011.

[13] Julie L. White, Derek T. Anderson, John E. Ball, and Brian Parker. Curvelet filter based prescreener for explosive hazard detection in hand-held ground penetrating radar, 2016.

[14] Matthew A. Lee, Derek T. Anderson, John E. Ball, and Julie L. White. Background adaptive division filtering for hand-held ground penetrating radar, 2016.

[15] Lance E. Besaw. Detecting buried explosive hazards with handheld gpr and deep learning, 2016.

[16] P.A. Torrione, K.D. Morton, R. Sakaguchi, and L.M. Collins. Histograms of oriented gradients for landmine detection in ground-penetrating radar data. *Geo-*

science and Remote Sensing, IEEE Transactions on, 52(3):1539–1550, March 2014.

[17] J. M. Malof, K. D. Morton, L. M. Collins, and P. A. Torrione. A probabilistic model for designing multimodality landmine detection systems to improve rates of advance. *IEEE Transactions on Geoscience and Remote Sensing*, PP(99):1–13, 2016.

[18] J. N. Wilson, P. Gader, W. H. Lee, H. Frigui, and K. C. Ho. A large-scale systematic evaluation of algorithms using ground-penetrating radar for landmine detection and discrimination. *IEEE Transactions on Geoscience and Remote Sensing*, 45(8):2560–2572, Aug 2007.

[19] Hichem Frigui and Paul Gader. Detection and discrimination of land mines in ground-penetrating radar based on edge histogram descriptors and a possibilistic k-nearest neighbor classifier. *Trans. Fuz Sys.*, 17(1):185–199, February 2009.

[20] Timothy C. Havens, Kevin Stone, Derek T. Anderson, James M. Keller, K. C. Ho, Tuan T. Ton, David C. Wong, and Mehrdad Soumekh. Multiple kernel learning for explosive hazard detection in forward-looking ground-penetrating radar, 2012.

[21] A. Manandhar, P. A. Torrione, L. M. Collins, and K. D. Morton. Multiple-instance hidden markov model for gpr-based landmine detection. *IEEE Transactions on Geoscience and Remote Sensing*, 53(4):1737–1745, April 2015.

[22] P. Gader, M. Mystkowski, and Y. Zhao. Landmine detection with ground penetrating radar using hidden markov models. *IEEE Trans. Geoscience and Remote Sensing*, 39:1231–1244, 2001.

[23] D. T. Anderson, K. E. Stone, J. M. Keller, and C. J. Spain. Combination of anomaly algorithms and image features for explosive hazard detection in forward looking infrared imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(1):313–323, Feb 2012.

[24] Andrew Karem, Amine B. Khalifa, and Hichem Frigui. A fisher vector representation of gpr data for detecting buried objects, 2016.

[25] K. C. Ho, P. D. Gader, J. N. Wilson, W. Lee, and T. C. Glenn. Landmine detection using frequency domain features from gpr measurements and their fusion with time domain features, 2005.

[26] Samuel Harris, K. C. Ho, and Alina Zare. On the use of log-gabor features for subsurface object detection using ground penetrating radar, 2016.

[27] A. Hamdi, O. Missaoui, and H. Frigui. An svm classifier with hmm-based kernel for landmine detection using ground penetrating radar. In *Geoscience and Remote Sensing Symposium (IGARSS), 2010 IEEE International*, pages 4196–4199, July 2010.

[28] S. E. Yuksel and P. D. Gader. Mixture of hmm experts with applications to landmine detection. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 6852–6855, July 2012.

[29] Xuping Zhang, Seniha Esen Yuksel, P. Gader, and J. N. Wilson. Simultaneous feature and hmm model learning for landmine detection using ground penetrating radar. In *Pattern Recognition in Remote Sensing (PRRS), 2010 IAPR Workshop on*, pages 1–4, Aug 2010.

[30] O. Lohlein and M. Fritzsche. Classification of gpr data for mine detection based

on hidden markov models. In *Detection of Abandoned Land Mines, 1998. Second International Conference on the (Conf. Publ. No. 458)*, pages 96–100, Oct 1998.

[31] A.B. Khalifa and H. Frigui. A multiple instance neuro-fuzzy inference system for fusion of multiple landmine detection algorithms. In *Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International*, pages 4312–4315, July 2015.

[32] S. L. Tantum, W. R. Scott, K. D. Morton, L. M. Collins, and P. A. Torrione. Target classification and identification using sparse model representations of frequency-domain electromagnetic induction sensor data. *IEEE Transactions on Geoscience and Remote Sensing*, 51(5):2689–2706, May 2013.

[33] S.E. Yuksel, J. Bolton, and P. Gader. Multiple-instance hidden markov models with applications to landmine detection. *Geoscience and Remote Sensing, IEEE Transactions on*, 53(12):6766–6775, Dec 2015.

[34] A. Karem and H. Frigui. Fuzzy clustering of multiple instance data. In *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*, pages 1–7, Aug 2015.

[35] Kenneth A. Colwell and Leslie M. Collins. Attribute-driven transfer learning for detecting novel buried threats with ground-penetrating radar, 2016.

[36] H. Frigui, K. Satyanarayana, and P. Gader. Detection of land mines using fuzzy and possibilistic membership functions. In *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on*, volume 2, pages 834–839 vol.2, May 2003.

[37] H. Frigui, P. Gader, and K. Satyanarayana. Landmine detection with ground penetrating radar using fuzzy k-nearest neighbors. In *Fuzzy Systems, 2004. Pro-*

ceedings. 2004 IEEE International Conference on, volume 3, pages 1745–1749 vol.3, July 2004.

[38] R. Mazhar, J. N. Wilson, and P. D. Gader. Use of an application-specific dictionary for matching pursuits discrimination of landmines and clutter. In *2007 IEEE International Geoscience and Remote Sensing Symposium*, pages 26–29, July 2007.

[39] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[40] X. Tian and C. Jung. Visual saliency estimation through label propagation. *Electronics Letters*, 51(14):1073–1075, 2015.

[41] S. Tripathi, S. Belongie, Y. Hwang, and T. Nguyen. Detecting temporally consistent objects in videos through object class label propagation. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9, March 2016.

[42] B. Raducanu, A. Bosaghzadeh, and F. Dornaika. Multi-observation face recognition in videos based on label propagation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 10–17, June 2015.

[43] J. Liu, Q. Fan, S. Pankanti, and D. N. Metaxas. People detection in crowded scenes by context-driven label propagation. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9, March 2016.

[44] Y. Wu, M. Pei, M. Yang, J. Yuan, and Y. Jia. Robust discriminative tracking via landmark-based label propagation. *IEEE Transactions on Image Processing*, 24(5):1510–1523, May 2015.

[45] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[46] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, 2002.

[47] Ahmed Elgammal, R Duriswami, D Harwood, and Larry Davis. Background and foreground modelling using nonparametric kernel density estimation for visual surveil. 90, 01 2002.

[48] Ying Zhao and George Karypis. Comparison of agglomerative and partitional document clustering algorithms. Technical Report 02-014, 2002.

[49] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

CURRICULUM VITAE

NAME: Graham Reid

ADDRESS: Computer Engineering & Computer Science Department
Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION:

M.S., Computer Science & Engineering
December 2018
University of Louisville, *Louisville, Kentucky*

B.S., Computer Science and Engineering
May 2014
University of Louisville, *Louisville, Kentucky*

PUBLICATIONS:

1. Graham Reid, Hichem Frigui, "A label propagation approach for detecting buried objects in handheld GPR data," Proc. SPIE 9823, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXI, 98230M (3 May 2016)