Lund University

Thesis Project

Department of Mathematical Statistics

Surface Classification with Millimeter-Wave Radar for Constant Velocity Devices using Temporal Features and Machine Learning

David Montgomery
Gaston Holmén

-

Abstract

Classification of surfaces in the near field using millimeter-wave radar commonly considers the use of polarization based methods for road condition monitoring. When a surface consists of larger structures one instead wishes to monitor the surface topography. Analysis of scattering from rough surfaces is highly complex and relies on prior knowledge of surface structure. In this work a device moving at constant velocity is considered. By constructing a set of slow and fast time based features a machine learning classifier is used to distinguish grass target surfaces from asphalt, gravel, soil and tiled surfaces. It is found that using estimated autocovariances and average envelope shapes make for efficient features and that a small fully connected neural network classifier adequately manages to determine the surface type. The found model is accurate yet parsimonious and could be implemented with limited hardware requirements. Application of a median filter onto the sequence of classifier predictions effectively suppresses outlying predictions. This model can find use in autonomous devices that have tasks performed on designated surface types, such as in autonomous lawn mowers.

Acknowledgements

First of all we would like to thank Acconeer for providing us with the opportunity to do our thesis project in collaboration with the company. Furthermore we would like to thank the members of the algorithm team at Acconeer - Dr. Peter Almers, Dr. Rikard Nelander, Dr. Bo Lincoln, Erik Månsson and Dr. Daniel Jung - for all your insightful discussions as well as your helpful and friendly disposition. A special thanks to our supervisor at Acconeer Peter Almers for his unwavering dediction and input throughout the making of this work, as well as for his infectious enthusiasm. We would also like to thank our supervisor at the university, Andreas Jakobsson, for his guidance in this project. One could not ask for better supervisors and coworkers.

Contents

Abbreviations

ANN Artificial Neural Network. 35–37, 45, 53

BF Beat Frequency. 9, 14, 19, 24, 25, 39

CNN Convolutional Neural Network. 40, 42, 53

DFT Discrete Fourier Transform. 9, 24–27, 29, 30, 54

DNN Deep Neural Network. 23, 37, 38, 42, 45, 47, 53, 54

IQ In-phase and Quadrature. 4, 7, 14, 15, 21, 26, 61, 62

LDA Linear Discriminant Analysis. 33, 35, 53

LOO Leave-One-Out. 33, 41, 43, 49, 51, 53, 57

LR Logistic Regression. 35, 36

LSTM Long Short Term Memory. 40, 42, 53

PCA Principal Component Analysis. 31, 33, 42, 53

PCR Pulsed Coherent Radar. 3, 4, 7, 9, 15, 16, 18, 19, 57

RCS Radar Cross Section. 10, 11, 14, 22, 24, 28, 57

RF Radio Frequency. 1, 7, 35

RRE Radar Range Equation. 10, 11, 21

SNR Signal-to-Noise Ratio. 10, 14, 20, 21

SVM Support Vector Machine. 30, 33, 35

ZMUV Zero-Mean Unit Variance. 34

Chapter 1

Introduction

Radars have several highly attractive properties: commonly, they are active systems independent of lighting conditions, are very fast and have high precision. Furthermore, implemented at millimeter-wave wavelengths, radar sensors can be designed as low-power devices with no moving parts in a favorable form factor (Lien et al., 2016).

Radar technology has been around since the 1930s (Watson-Watt, 1945), and has since developed into a well-established field of engineering. Although the typical use case in radio frequency (RF) radars regard detection and tracking of large objects at far distances, such as for air and marine monitoring, new areas of applications have emerged during recent years, posing very different engineering challenges. Some of these are outlined in (Amin, 2017), where close-range radars are used for applications such as vital signs monitoring (Kuo et al., 2016), gesture recognition (Lien et al., 2016) and tumor imaging (Klemm et al., 2011), to name a few. Furthermore, millimeter-wave radars have lately become more inexpensive, largely due to their widespread adoption in the automotive industry (Frenzel, 2018), making such devices an attractive option in a wide range of low-cost applications.

In this work, we investigate whether millimeter-wave radar can be used for surface classification of rough surfaces. Two broad antenna beams illuminate a target scene, and using a feature extraction procedure and machine learning techniques, the returning echoes are analyzed to determine the surface type. This work focuses on the use case of determining if a surface is grassy or not, but could be extended to other surface types.

1.1 Motivation and Previous Work

Autonomous robots have found increasing use in numerous devices, from helping customers navigate stores (McSweeney, 2018) to keeping floors clean (Sanfacon, 2017) and mowing lawns (Udelhofen, 2018). A common challenge in such systems is keeping the robot in bounds. This commonly means for the robot to be aware where on a two-dimensional map it is currently located. In certain applications staying in bounds rather involves remaining on a set of allowed surface types, such as for autonomous robot lawn mowers remaining only on areas covered in grass. In such devices one may be content with knowing that the robot roams around remaining on its designated surface type rather than having knowledge of its exact position, and hence a surface classification method would be of great use.

Surface classification can also be used in autonomous devices as a supporting system. A robot vacuum cleaner could for instance make use of such a system in numerous ways, such as for avoiding liquid spills or using surface-dependent cleaning programs. One could easily imagine other use cases where knowledge of which surface type an autonomous device is on would be a great convenience.

The task of surface classification can be achieved in many ways. Taking inspiration from the recent advances in computer vision (Liu et al., 2018), surfaces could be examined optically and separated by their differences in texture - their spatial organization of image elements (Do and Vetterli, 2002). Another method is employing direct surface contact. In (Giguere and Dudek, 2011), surface identification was performed for low velocity mobile robots using a small metallic rod with an attached accelerometer, capturing motion signatures when traversing a surface. Researches, perhaps motivated by experiments showing bats capable of distinguishing surface roughness by using their echolocation (Schmidt, 1988), have also experimented with ultrasonic methods for use in road condition monitoring (Bystrov et al., 2016), (Mckerrow and Kristiansen, 2006).

In this work, we are interested in distinguishing grass surfaces from non-grass surfaces using millimeter-wave radar. The application most closely resembling this objective is perhaps the use of millimeter-wave radar for road condition monitoring, where one attempts to determine the state of a road for safety and driver assistance purposes.

Several methods involving measuring polarizations have been proposed for this purpose. In (Finkele et al., 1995), an array of radar transmitters and receivers with different polarities were used sequentially to illuminate the same surface area. Other bistatic setups (configurations with transmitter and receiver spatially separated) for road condition monitoring were considered

in (Kees and Detlefsen, 1994) and (Finkele, 1997). Monostatic experiments, with transmitting and receiving antenna at the same location, have found varying degrees of success. In (Viikari et al., 2008) and (Häkli et al., 2013), a monostatic setup was used to compare returning strengths of differently polarized radar radiation.

These polarization-based methods are founded on measuring the polarization effects of adding a thin layer of ice or water onto a flat asphalt surface. Since the reflectivity of polarized radiation depends both on the polarization and the index of refraction of the reflector one can create a setup founded in well established physics to determine the surface type. These methods rely on illuminating a planar surface, and can thus not effectively be used for the application considered in this report where target surfaces may have large variations in height such as in lawns. In the present case of distinguishing grass we are instead more interested in capturing the *topography* of a surface, not captured as slight differences in polarization.

Backscattering from rough surfaces have been studied in for instance (Fung and Pan, 1987) and (Fung et al., 1992), clearly showcasing the difficulties in the modeling of scattering processes from irregular targets. In these papers, a surface is modeled having a Gaussian height distribution with a correlation function determining its shape. Such modeling requires detailed prior knowledge of the surface at hand and does not capture the many complexities arising in real-world surface structures and could hardly be of any great use for this project. In (Scharf et al., 2018), the frequency dependence of monostatic surface backscattering was evaluated for a simulated surface with similar assumptions, and shown to be in accordance with an experimental setup. Surfaces were distinguished by using the Fraunhofer criterion, which occurs as angular dependent constructive and destructive interference at specific combinations of frequency and roughness. This model is however unsuitable for the present carrier frequency of 60 GHz and the large surface variations in the target surfaces investigated in this report.

As far as the authors know, this paper presents the first attempts of classification of grass target surfaces using millimeter-wave radar with the sensor placed in a device moving at constant velocity. Furthermore, we employ a temporal feature extraction process and pass extracted features to a machine learning classifier for surface classification in a manner which has seemingly not been done before in literature for this type of problem.

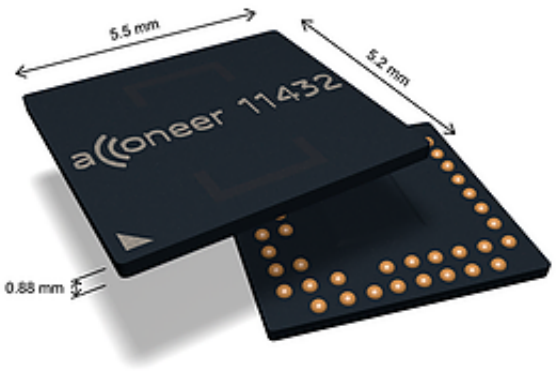

Figure 1.1: Image of the 60 GHz PCR radar sensor used in this work (retrieved from *www.acconeer.com/products* on 14/1/18).

1.2 Objective

In this work we, wish to explore whether it is possible to create an effective classification scheme from millimeter-wave radar response acquired during device movement. The classification task will be to accurately distinguish a grass surfaces from a selection of non-grass surfaces commonly found adjacent to lawns. All experiments will be performed using two 60 GHz pulsed coherent radar (PCR) sensors from Acconeer, see Figure 1.1, mounted at the front of device moving straight at a constant velocity.

To this end, the Python programming language will be utilized together with Scikit learn and Keras with a TensorFlow backend for development (Pedregosa et al., 2011), (Chollet, 2015), (Abadi et al., 2015).

1.3 Thesis Outline

The outline of this thesis is as follows:

Chapter 2: This chapter explains the fundamentals of PCR systems, and describes how distance, reflectivity and radial velocity can be extracted from the radar response. The method in which radar measurements are carried out is explained, as well as the IQ demodulation procedure.

Chapter 3: In chapter 3, we proceed with describing the data collection method. Sensor settings, such as pulse length and sampling frequency, are discussed.

Chapter 4: Chapter 4 presents the feature extraction process. Temporal features, such as the autocovariance, are investigated for accurate surface classification. At the end of the chapter, one of the tested extraction methods is selected for further use.

Chapter 5: In this chapter, a few different classifiers are tested for prediction accuracy. A selection of two linear and four non-linear machine learning models are tested. At the end of the chapter, one machine learning classifier is selected as the most suitable.

Chapter 6: In chapter 6, data augmentation and outlier suppression is discussed.

Chapter 7: Chapter 7 examines the capabilities of the found model in real and artificially created test scenarios. The found results, as well as some error sources and limitations, are discussed.

Chapter 8: In chapter 8, conclusions of this work are presented and possible directions of future work are discussed.

Chapter 2

Radar System Overview

In order to build a model capable of accurately capturing distinguishing features from target surfaces, one must first understand the origin and structure of the received signal. In this chapter a few fundamental concepts in radar systems are introduced that explain how a target's range, velocity, and reflectivity arise in a PCR system. The mixing and IQ demodulation procedures are also discussed, which explain how radar measurements are performed.

2.1 Target Properties from Pulsed Radar Response

Fundamentally, a radar operates by radiating RF electromagnetic energy and listening if the transmitted energy generates any echoes (Skolnik, 2009). By analyzing properties of returning signals, it is then possible to obtain information regarding the scattering targets. This may involve the distance and angle at which scatterers are located, or at which velocity they are moving in. With a sufficiently high angular and range resolution, it is even possible to discern shapes and sizes of targets.

Standard radars are active systems, meaning that they have a radiating antenna and hence do not depend on any ambient radiation (Richards, 2014). Radar systems can be realized in many ways, however in this report a PCR system is used. This means that a sequence of short coherent wavelets are transmitted towards a target scene to determine its properties. One such transmitted wavelet pulse $x_T(t)$ has some carrier frequency, f_c, and envelope, $A(t)$, as in

$$x_T(t) = A(t)\sin(2\pi f_c t). \tag{2.1}$$

After various scattering processes, the returning signal is captured by either an array of antennas, or just a single antenna.

2.1.1 Single-Antenna Considerations

Commonly, in larger radar systems, many receiving antennas capture the returning radiation. This allows for distinguishing radar targets not only in range, but also in angle. In this work, a single receiving antenna is used, meaning that the radar has no intrinsic method of determining at which angle a scatterer is located. Thus, only information in range can be captured by this sensor.

To understand what information can be obtained from such a one-dimensional signal, a single point scatterer some distance away from the transmitter is here considered. If a radar pulse on the form of (2.1) is transmitted towards it, the returning signal will be a delayed pulse on the form (Richards, 2014)

$$y(t) = CA(t - B)\sin(2\pi f_c(t - B)) \tag{2.2}$$

where B is some delay related to the distance to the target and C a constant corresponding to the loss of energy from transmission to reception of a pulse. For the purposes of this report the properties of this delay is of particular interest. In the next two sections it is shown how distance and velocity can be deduced from this parameter using absolute measurements of B and relative changes in B between pulse measurements. The C variable carries information about the reflectivity of a scatterer, described lastly.

2.1.2 Radar Distance Measurements

If a point scatterer was present at a distance d from the transmitting antenna, a wavelet will return $2d/c$ seconds after transmission, where c denotes the speed of light. The 2 appears to accommodate for the pulse traveling forth and back between the sensor and the scatterer. Hence, the transmitted signal is delayed according to

$$\begin{aligned} y(t) &= CA(t - 2d/c)\sin(2\pi f_c(t - \frac{2d}{c})) \\ &= CA(t - 2d/c)\sin(2\pi f_c t - \frac{4\pi d}{c} f_c). \end{aligned} \tag{2.3}$$

By measuring the delay B in (2.2), either as a phase shift or a shift in envelope, one may simply calculate the distance through

$$d = \frac{Bc}{2}. \tag{2.4}$$

Thus, by examining the time delay between pulse transmission and reception, we can deduce the distance to the target scatterer.

2.1.3 Doppler Frequency in Pulsed Radar

If the target scatterer is moving from or towards the transmitting antenna, a shift in frequency will occur in the received pulse $y(t)$. For velocity v, the transmitted frequency, f_c, will be shifted to f_r according to the well known Doppler formula (Ridenour, 1947)

$$f_r = \frac{c+v}{c-v} f_c \tag{2.5}$$

where a positive v indicates that the scatterer is moving towards the transmitter. The frequency shift f_d, also known as the *beat frequency* (BF), is then

$$f_d = f_r - f_c = \frac{2v}{c-v} f_c \approx \frac{2v}{c} f_c = \frac{2v}{\lambda_c} \tag{2.6}$$

where λ_c is the carrier wavelength. This shift means that an approaching target has a slightly increased returning frequency, and conversely that a receding target has a slightly decreased frequency. In this project, a pulsed radar is used, meaning that measurements are done repeatedly using short wavelets instead of transmitting one continuous wave. What does this mean for the Doppler shift?

The use of short wavelets means that each individual transmitted pulse becomes compressed by the Doppler shift. This compression is miniscule, on the scale of fractions of picoseconds, and thus insignificant for any target with reasonably moderate velocity. For a PCR system with sampling frequency, F_s, and sampling period, $T_s = 1/F_s$, the Doppler shift can instead of a shift in frequency be viewed as a phase shift from one pulse to the next. Between two distance measurements a target moves a distance vT_s. Hence, each pulse travels a distance $2vT_s$ less than the preceding one. This distance corresponds to $2vT_s/\lambda_c$ wavelengths, providing a phase change, $\Delta\phi$, per pulse of

$$\Delta\phi = \frac{4\pi}{\lambda_c} v T_s. \tag{2.7}$$

For a target moving at a constant pace, the BF arising from the constant Doppler phase shift is thus $f_d = 2v/\lambda_c$, as was found in equation (2.6). Thus, by taking a discrete Fourier transform (DFT) of the sequence of T_s-spaced measurements for one specific distance, we can estimate the velocity from the peak frequency f in the DFT as

$$v = \frac{\lambda_c f_d}{2} \approx \frac{\lambda_c f}{2}. \tag{2.8}$$

2.1.4 Power Dissapation

The C factor in (2.2) describes the ratio of the emitted power that is received after scattering. To get a feel for how a target's reflectivity relates to this factor, we can make the following thought experiment. First, we assume that the transmitted power, P_t, is radiated isotropically by a point source, meaning that the power density is distributed uniformly by the source over the surface of a sphere. When our radar sphere has radius R, the power density at the sphere surface becomes (Amin, 2017)

$$P_d = \frac{P_t}{4\pi R^2} \tag{2.9}$$

where $4\pi R^2$ is the surface area of the sphere. If a scatterer is present at range d with radar cross section (RCS) σ and gain G_t, the fraction of the radar power reflected by the scatterer is given by

$$P_e = \frac{P_t G_t \sigma}{4\pi d^2}. \tag{2.10}$$

Then, provided that the scattering process is isotropic and lossless, the power reflected back at the radar receiver is

$$P_f = \frac{P_e}{4\pi d^2} = \frac{P_t G_t \sigma}{(4\pi d^2)^2}. \tag{2.11}$$

Finally, only a portion $P_r = P_f A_r$ of the power is captured by the receiver, where A_r is a function of transmission wavelength and receive antenna gain. Including this into (2.11), we obtain the *radar range equation* (RRE) relating the transmitted and received power as

$$P_r = \frac{P_t G_t A_r \sigma}{16\pi^2 d^4}. \tag{2.12}$$

It is worth pointing out that this argument only holds for scatterers of limited sizes, and that if the scattering target for instance was a parabolic antenna, a waveguide or an infinite plane, this relationship between the transmitted and received power would not be valid. The RRE states that power dissipates rapidly, by a factor $1/d^4$, with range. Hence, increasing the distance to a target object by a factor 2 will return only 1/16 of the power otherwise received. According to (Richards, 2014), this rate in real-world scenarios is typically between $1/d$ to $1/d^4$, so that the process may not be as lossy as the RRE predicts. If the noise power, N, remains constant regardless of target distance, the *signal-to-noise ratio* (SNR), defined as SNR $= P_r/N$ quickly decreases with range.

We also see from the RRE that the power returned is governed by the RCS, σ, of the target. The RCS describes how reflective a target is, determined by its size and shape as well as its dielectric properties. RCS can be regarded as a fictious area that describes the intensity of the wave reflected back to the scatterer. This area corresponds to the projected area of a perfectly conducting sphere, with a diameter of several wavelengths, whose echo strength would match that of the target if we were to replace the target with the sphere (Knott, 1993).

However, as the reflectivity and thus the RCS can vary with both distance and viewing angle, this fictious metal sphere subsequently changes size depending on where in space it is located. Thus, our best recourse is to simply regard the RCS as a measure of the intensity of the radar echo expressed in terms of an area.

2.2 Matched Filter

In the preceding section, it was shown that the scattering response carries information about the target range, the reflectivity, and the radial velocity. This section explains how the radar captures the returning signals to generate raw radar data through *matched filtering*.

Matched filtering involves convolving the returning signal with a copy of the transmitted signal. Since the radio channel is assumed to be linear and time invariant (any Doppler shift is insignificant on the transmission-reception time scale) the received signal consists of a sum of weighted and time shifted transmitted signals. An internal pulse, called an analysis pulse, is generated as a delayed copy of the transmitted pulse. The received signal and the generated signal are then multiplied and summed to form one measurement point $m(\tau)$, where τ is the internal delay of the analysis pulse.

Mathematically, we can describe matched filtering as a convolution

$$m(\tau) = x_T(-t) * y(t) = \int_{-\infty}^{+\infty} x_T(t-\tau)y(t)dt \tag{2.13}$$

where $x_T(t)$ is the analysis pulse as defined in (2.1) and $y(t)$ is the returning wavelet. If the incoming signal, $y(t)$, stems from a single scattering point target, then $y(t)$ is on the form of (2.2). With the envelope, $A(t)$, as a rectangular envelope[1] of duration L

[1]The rectangular envelope is commonly used for maximal power transmission (Richards, 2014), however the Acconeer radar system employs an envelope more similar to a raised cosine. The rectangular envelope is used here for mathematical convenience.

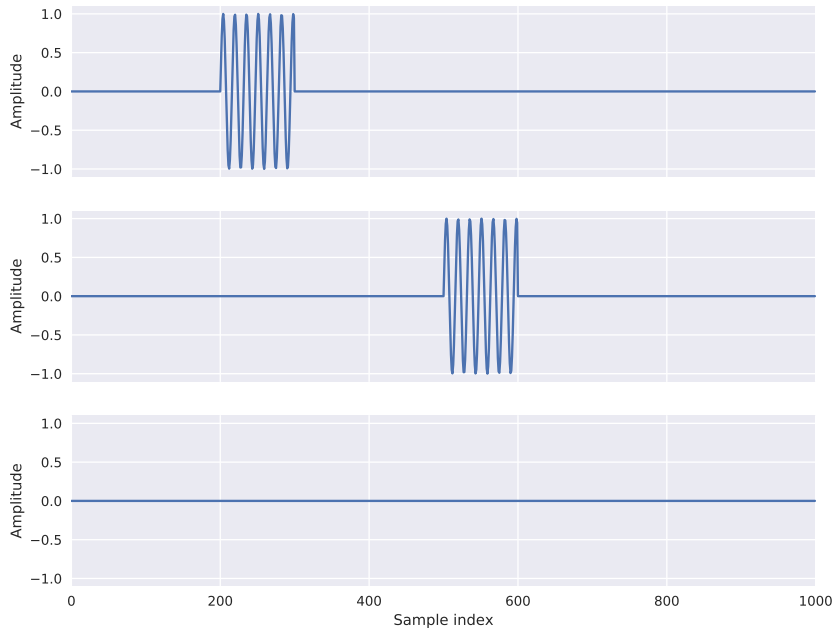

Figure 2.1: Analysis pulse $x_T(t-\tau)$ (top), received pulse $y(t)$ (mid) and multiplication output $x_T(t-\tau)y(t)$ (bottom). As the signals do not overlap the multiplication yields only zero values.

$$A(t) = \begin{cases} 1 & \text{if } 0 \leq t < L \\ 0 & \text{otherwise,} \end{cases} \tag{2.14}$$

$m(\tau)$ will be on the form

$$m(\tau) = \begin{cases} 0 & \text{if } |\tau - 2d/c| \geq L \\ (2.16) & \text{if } |\tau - 2d/c| < L \text{ and } \tau \leq 2d/c \\ (2.17) & \text{if } |\tau - 2d/c| < L \text{ and } \tau > 2d/c \end{cases} \tag{2.15}$$

$$\frac{C}{2}\left((\tau + L - B)\cos(2\pi f_c(\tau - B)) - \frac{1}{2\pi f_c}\sin(2\pi f_c(\tau - B))\right) \tag{2.16}$$

$$\frac{C}{2}\left((B + L - \tau)\cos(2\pi f_c(\tau - B)) + \frac{1}{2\pi f_c}\sin(2\pi f_c(\tau - B))\right). \tag{2.17}$$

The full derivation of (2.15), (2.16) and (2.17) can be found as an appendix. This procedure is illustrated in Figures 2.1 and 2.2. The analysis

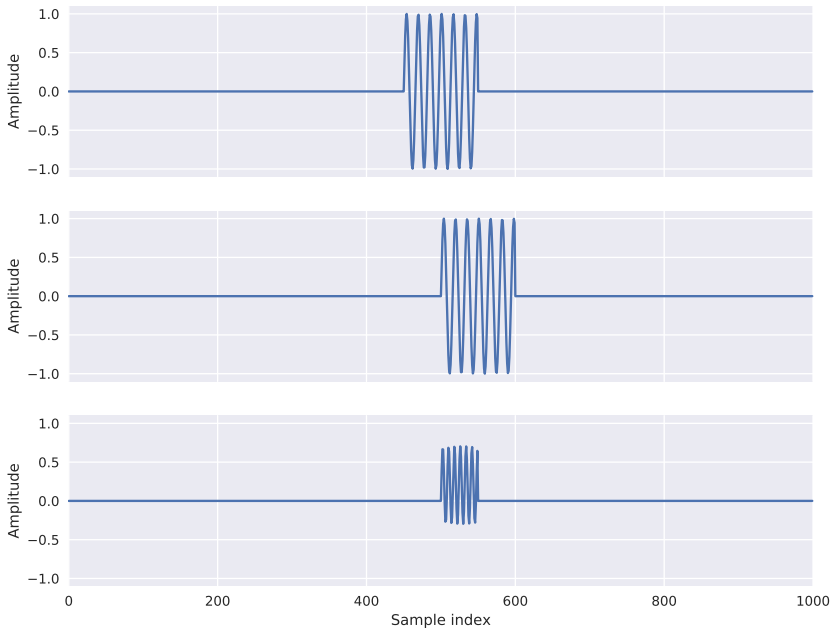

Figure 2.2: Analysis pulse $x_T(t-\tau)$ (top), received pulse $y(t)$ (mid) and multiplication output $x_T(t-\tau)y(t)$ (bot). The overlapping signals produce non-zero values after multiplication.

pulse is shown in the top plot, the received pulse in the center and the result of the elementwise multiplication at the bottom. These two figures differ in that the internal delay, τ, of the analysis pulse is different, so that no overlap occurs in the first figure while significant overlap happens in the second.

In Figure 2.3, a full numerical computation of $m(\tau)$ has been performed, shown in part in Figures 2.1 and 2.2. Note that this result was obtained for a noise-free mixing of a signal from a single theoretical point scattering target. It is thus clear that with complicated continuous structures, target scenes produce complex outputs at the receiver.

2.3 IQ Demodulation

Even though single-antenna receivers have no angular resolution, it was shown above that the radar echo carries useful information. By measuring the temporal shift from transmission to reception, we can calculate the

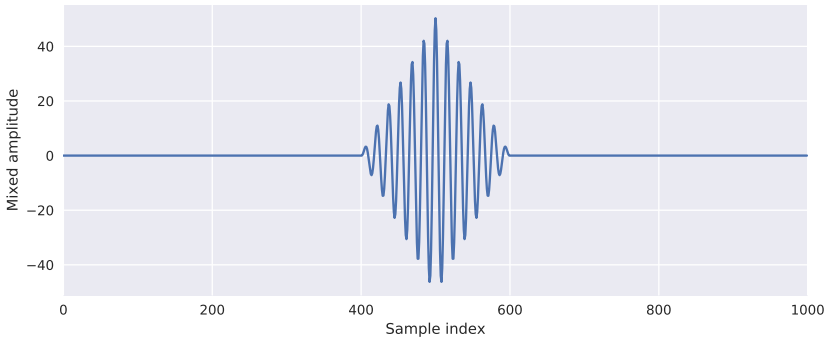

Figure 2.3: The matched filtering output $m(\tau)$ is produced by summing the overlap of the analysis and received pulse for all lags τ.

distance to a scatterer, and by examining the phase shift between pulses, we may find the BF and thus the radial velocity of the scatterer. Finally, the scatterers' dielectric properties are brought forward through the RCS, showing up as a scaling factor after matched filtering.

Although the raw data signal produced by matched filtering holds the information we are interested in, it commonly has a low SNR (Richards, 2014) and depends on the carrier frequency as can be seen in equations (2.16) and (2.17), and in Figure 2.3. To get rid of the carrier frequency, some transformation of the raw signal is usually performed. In this work, we are particularly interested in resolving small Doppler shifts, and thus we wish to closely monitor phase shifts from one measurement to the next. One effective data representation of raw data that facilitates accurate phase tracking involves splitting the signal into its *In-phase* and *Quadrature* (IQ) components (Lien et al., 2016).

The In-phase channel mixes the raw signal with an oscillator at the carrier frequency f_c, and the Quadrature channel with the same frequency but with a 90° phase shift from the In-phase channel. After low-pass filtering, the IQ components are interpreted as a complex number, $I(t) + jQ(t)$. IQ demodulation shifts the information bearing part of the mixed signal to baseband, meaning that an echo waveform on form $A(t)\sin(2\pi f_c t + \phi(t))$ after demodulation has the form $A(t)e^{j\phi(t)}$, effectively eliminating the carrier frequency. Thus, after matched filtering and IQ demodulation we obtain a set of complex range measurements that we will denote as a *radar sweep*. The details of the IQ demodulation scheme can be found in the appendix.

Chapter 3

Data Acquisition and Preprocessing

In the preceding chapter, it was shown how certain metrics can be extracted from a target scatterer using its radar response, where the values the radar produces after matched filtering and IQ-demodulation was on the form of complex IQ radar sweeps. In this chapter, we continue with describing how measurements are acquired, structured, and preprocessed. Some PCR system settings are discussed.

3.1 Data Representation

A common representation of data acquired by multi-antenna radars is the *datacube* (Richards, 2014). The first dimension of the datacube consists of the radar sweeps, as described in section 2.3. Radar sweeps are formed by estimating the time-of-flight of a returning wavelet from a target scene, as described in section 2.2. This process forms the shortest time frame possible with sample spacing on the picosecond scale for millimeter-wave radar, and is thus commonly referred to as the *fast time* scale. New radar sweeps are acquired at a rate set by the sampling frequency. This sequence of radar sweeps form the second dimension of the datacube. Due to its much slower rate, it is aptly referred to as the *slow time* scale. Finally, the last dimension is constructed from the antenna array. This representation of data acquired by radar arrays are often referenced in journal papers when describing, for instance, beam forming or Doppler processing algorithms (Gentile and Donovan, 2018).

We are thus working with two time scales simultaneously. Previously, the variable t described the fast time scale, but it will from this point be referred

to as the slow time scale to accomodate for this new dimension. Instead, we will denote the fast time scale with its corresponding range.

The PCR system considered in this work only has a single receiving antenna, meaning it does not capture angular information, as discussed in section 2.1.1. Thus, we are left with data representing only fast and slow time dimensions. Two such systems are used in parallel, but are not synchronized to one another. This essentially means that the sensors take turn in capturing radar sweeps as opposed to listening to the same echo wavelets. Angular information is still omitted, but using two sensors, surface characteristics can be more accurately captured.

For our intents and purposes, we may concatenate the two discrete sensor outputs to form a *data matrix*. If a radar sensor's output with fast time index $d = 1...Z$ and slow time index $t = 1...Q$ is described by $r(d,t)$, we form a data matrix $\mathbf{D}$ with sensors $r_1(d,t)$ and $r_2(d,t)$ through

$$\mathbf{D} = \begin{bmatrix} r_1(0,0) & r_1(1,0) & \cdots & r_1(Z,0) & r_2(0,0) & \cdots & r_2(Z,0) \\ r_1(0,1) & r_1(1,1) & \cdots & r_1(Z,1) & r_2(0,1) & \cdots & r_2(Z,1) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ r_1(0,Q) & r_1(0,Q) & \cdots & r_1(Z,Q) & r_2(0,Q) & \cdots & r_2(Z,Q) \end{bmatrix}, \quad (3.1)$$

or, more succinctly, as samples $r(n,t) = \mathbf{D}_{n,t}$ with $n = 1...2Z$. Each collected data matrix is built up of $Q = 50,000$ slow time samples.

3.2 Measurement Setup

As mentioned above, two sensors each capture one data matrix per measurement session. Sensor data is acquired while the robot is moving at a constant pace, as illustrated in Figure 3.1. Furthermore, two mounting angles are used, where one sensor is facing directly towards the ground while the other has a 22.5° forward tilt. The reasoning behind this setup is for each sensor to capture different aspects of the surface below; the sensor directed straight downwards may capture a larger component of the specular (i.e. mirror-like) reflection from the ground plane, while the tilted may capture more diffuse reflections. This concept is closely related to the surface ruggedness illustrated in Figure 3.2. It is worth noting that the half power beam width of the sensor is roughly 60° (Acconeer, 2018), meaning that an angular width of about 60° is illuminated by each sensor.

A final consideration is that as the sensors are placed on the inside of the robot plastic chassis, there is a risk for interference in the region between the plastic and the antenna. To avoid this, a small mount was 3D-printed so that

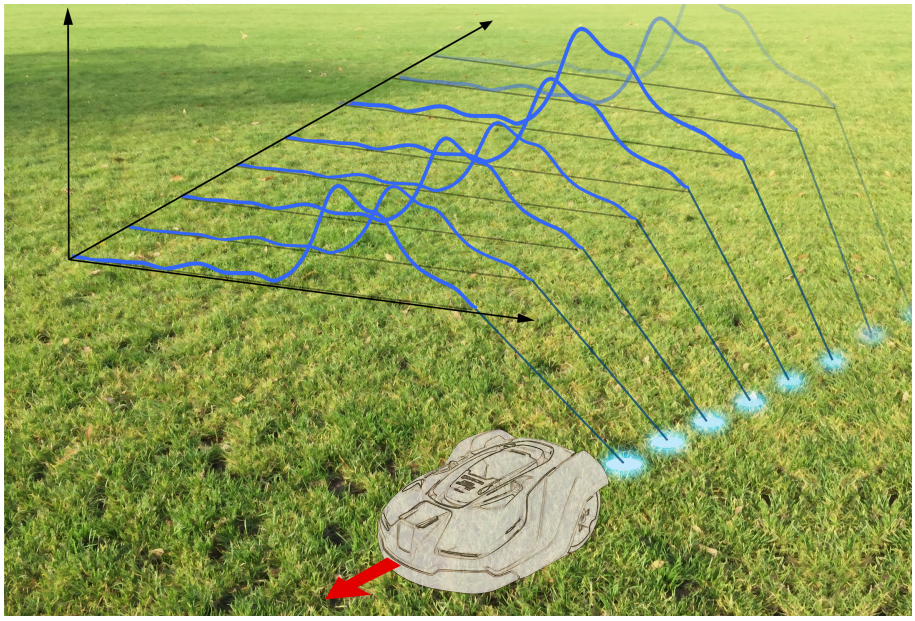

Figure 3.1: Illustation of the data acquisition process. Radar sweeps are collected during constant velocity robot movement and stored in a data matrix. For visualization purposes, the absolute values of the complex radar sweeps are shown and the spatial distance between radar sweep measurements have been greatly exaggerated.

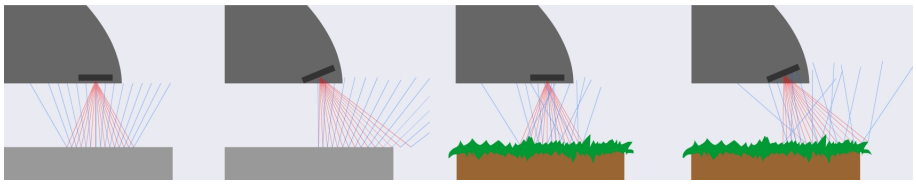

Figure 3.2: By using two PCR sensors with different mounting angles more diverse information about a target surface is obtained. The ratio between specular and diffuse reflectivity is dependent on surface roughness, as illustrated in the figure.

the distance to the plastic was $\lambda_c/4$. Doing this means that any undesired wavelets propagating back and forth in this region interfere destructively due to the change in phase and the superposition principle of electromagnetic radiation (Griffiths, 2018).

3.3 Target Surfaces

In this work, we wish to see if we can create a binary classifier capable of distinguishing grass from non-grass surfaces. For the application of keeping a robot lawn mower in bounds, it is natural to select other surfaces that commonly border lawns. The selection of surfaces was made with this in mind, and is presented in table 3.1, along with the number of measurement sessions per surface, each acquiring one data matrix with 50,000 slow time samples. Each session was taken either on a different day or in a different location than the rest. Note that the total sampling time amounts to $50,000 \cdot 42/(F_s[\mathrm{s}^{-1}] \cdot 60) = 175$ minutes.

Surface	Data matrices
Grass	18
Asphalt	6
Gravel	6
Soil	8
Tiles	4
Total	42

Table 3.1: Measured surface types and number of captured data matrices per surface type.

3.4 Measurement Settings

The Acconeer PCR system has several user-defined settings. In this section, a few key parameters of these are discussed, namely the range swath, the slow time sampling rate, and the wavelet duration. Although finding the appropriate sensor settings requires some level of trial-and-error due to hardware and software limitations, a good starting point can be found from a theoretical standpoint.

3.4.1 Sampling Rate

When working with radars and moving targets, it is important to select a sampling frequency, F_s, capable of resolving the maximum speed that an investigated target can attain. If the sampling frequency is too low, aliasing occurs, distorting the frequency spectrum (Lindgren et al., 2014). In order to assure that aliasing does not occur, one must select a sampling rate at least twice the maximal frequency component received, f_{max}. This limit is commonly known as the Nyquist limit (Proakis and Manolakis, 2014). From section 2.1.3 we find that the velocity, v, the carrier wavelength, λ_c and the BF, f_d are related through $v \approx \lambda_c f_d/2$. Combining this result with the Nyquist limit, we require from the sampling rate that

$$F_s \geq 2f_{max} = 2f_{d,max} \approx \frac{4v_{max}}{\lambda_c}. \tag{3.2}$$

The maximum velocity, v_{max}, above is the *radial* velocity. In the measurement setup depicted in Figure 3.3, a vehicle is moving forward at constant velocity, v_0, having a sensor mounted at the front with a 22.5-degree tilt and a 60 degree angular spread. The maximal velocity component that is orthogonal to the sensor occurs at the far end of the radar's view, at a $22.5° + 30° = 52.5°$ tilt angle. With $\lambda_c = c/f_c = 0.005$ m and the maximum orthogonal velocity component, $v_{\perp,max}$, the requirement above can be written as

$$F_s \geq \frac{4v_{\perp,max}}{\lambda_c} = \frac{4v_0 \sin(52.5°)}{\lambda_c} \approx 190 \text{ Hz}. \tag{3.3}$$

Thus, in order to avoid aliasing, the sampling rate, F_s, should be at least 190 Hz.

3.4.2 Wavelet Duration

The length, or duration, of the transmitted wavelet pulses in a PCR system determines the bandwidth. Taking a Fourier transform of a short wavelet

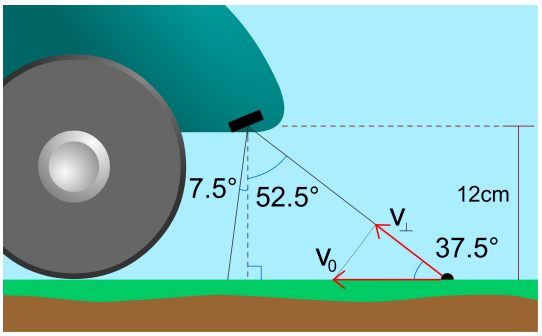

Figure 3.3: Sensor placement of the 22.5°-degree tilted sensor in the robot chassis. The figure indicates the furthest point visible by this sensor.

produces a wider spectrum, and a longer pulse length conversely produces a narrower spectrum. Assuming the same amplitude, a longer wavelet corresponds to transmitting more energy and thus receiving a signal with better spectral resolution and SNR, while a shorter wavelet has better spatial resolution. Thus, the wavelet duration parameter becomes a tradeoff between spatial resolution and SNR. A reasonable strategy is therefore to start with a short wavelet and increase its duration until a reasonable SNR is obtained.

3.4.3 Range Swath

Finally, the radar measures power over some range interval, also called the *range swath* of the radar (Richards, 2014). Again, a tradeoff appears. If a short range swath is selected, we can increase the sampling rate and still stay within the allowed hardware transmission speed limitations. However, if the range swath becomes too short, we may miss useful information we could have collected outside the chosen interval.

Thus, a reasonable strategy is to select the most critical interval and then increase the sampling frequency to its hardware allowed limit, keeping in mind that the sampling frequency should exceed the limit found above in section 3.4.1. As the distance to the surface plane is roughly 12 cm in the measurement setup, the furthest distance illuminated is $12/\sin(37.5°) \approx 20$ cm from the sensor. We should therefore at least have a range swath spanning this region.

The three settings for the three parameters discussed in this section are summarized in table 3.2.

Setting	Value
Wavelet duration	50 ps
Sampling frequency	200 Hz
Range swath	7 to 23 cm

Table 3.2: Sensor settings. The wavelet duration is set as short as possible while maintaining a reasonable SNR.

3.5 Downsampling in Fast Time

Using the settings in table 3.2, we obtain 331 range samples per radar sweep in the 7-23 cm range swath, and as samples are equidistantly spaced, they are separated by $(230\ [\text{mm}] - 70\ [\text{mm}])/331 \approx 0.48$ mm. After examining a number of radar sweeps, it is clear that closely spaced points are highly related, and that including all datapoints for analysis is redundant. This correlation occurs in part due to IQ demodulation, as it involves low pass filtering in fast time (see appendices), making closely spaced points highly correlated. With this redundancy of information in mind, we may downsample by some integer factor, D, without significant loss of information. Downsampling measurements $r_1(d,t)$ and $r_2(d,t)$ by a factor D can be expressed as

$$r_{1,D}(d,t) = r_1(dD,t), \quad \text{and} \quad r_{2,D}(d,t) = r_2(dD,t) \tag{3.4}$$

and the data matrix $r_D(n,t)$ is formed correspondingly.

3.6 Sweep Normalization

One hardware quirk of the radar sensors used in this project is that their gain, found in A_r in the RRE, can vary significantly from one sensor to the next. This means that faced with the same target at the same distance, two sensors exhibit a similar sweep structure, but possibly with a different scaling. Assuming a model's training data has been collected with one sensor, it could therefore be difficult to successfully perform classifications with the same model using a sensor which has a different gain. A way to overcome this issue is to perform radar sweep normalization as a preprocessing step. There are multiple ways of performing such a normalization.

A simple strategy is to simply divide each sweep with its maximum absolute value. While this may intuitively seem like a good idea, it eliminates signal evolution structures between consecutively captured sweeps - if neighboring sweeps vary in received signal strength, we wish to capture such a

behavior. A better solution is to instead have the normalization method depend on several consecutive sweeps. Normalizing using multiple sweeps thus maintains some of the relative amplitude structure. Constructing a smaller gain independent data matrix, $x(n,t)$, consisting of K downsampled sampling points and T consecutive sweeps, starting from slow time sample, T_m, we can normalize by the average amplitude through

$$x(n,t) = \frac{r_D(n, T_m + t)TK}{\sum_{n=0}^{K-1} \sum_{\tau=0}^{T-1} |r_D(n, T_m + \tau)|}. \tag{3.5}$$

Thus, we can perform gain normalization while, at least partially, maintaining any structures or patterns present in the returning sweep energy. The drawback of this normalization is that information pertaining to absolute measurements of target surfaces' RCS is lost in the process.

Chapter 4

Feature Extraction

In this chapter, we describe our feature extraction process. This process is an intermediary step between data preprocessing and surface classification, where we calculate metrics that are used for classification. After introducing some potential features, we compare the accuracy of a few feature ensembles using a linear classifier to determine which ensemble should be used for further analysis.

4.1 Why Feature Extract?

Machine learning algorithms fundamentally aim to extract useful patterns in data. In our case, data consist of the complex output of two radar receivers and the useful patterns we search for are patterns capable of distinguishing the surface type as either grass or non-grass.

We found in chapter 2 that a single radar sweep only provides information about ranges and reflectivities but cannot determine target velocities. Due in part to the normalization process done in the preprocessing step, such measures alone will hardly suffice to make an efficient predictor. Instead, information from several sequential sweeps should be used to generate a prediction. In order to make use of the information content present in the sequence of radar sweeps we have two main options.

The first involves using a complex nonlinear machine learning algorithm, such as a deep neural network (DNN) with a large number of hidden layers that figures out how to calculate efficient temporal metrics on its own, provided large bulks of unmodified data. Algorithms exist that indeed can learn to analyze frequency components, for instance for speech recognition (Graves et al., 2013), but tend to be too computationally expensive to realistically implement when hardware is limited.

Instead, we utilize that we know that taking the DFT in slow time provides velocity estimates for a given distance through BFs, as was explained in 2.1.3. The scaling of each frequency bin thus provides an estimate of the amount of velocity component at a given range. Such estimates are rich in information regarding the topography of a surface structure, as it accounts for the RCS of a surface at different velocities and thus different angles of incidence.

A second option is thus to utilize this knowledge and perform *feature extraction*. By extracting a smaller set of features pertaining to the frequency content of the given data, we can significantly reduce the complexity of the model. Furthermore, the feature extraction process allows us to pinpoint what data characteristics we wish to monitor, and subsequently which data characteristics we believe are of value. This ensures that the machine learning algorithm generates its predictions using patterns in metrics selected by the authors, rather than finding patterns from raw input data. For these reasons the focus of this report is using feature extraction based machine learning.

4.2 Features

Calculating velocity-based features require the use of a sequence of multiple sweeps. How long should one such sequence be? If T radar sweeps are used per classification, the rate of classifications per second, F_c, produced relates to the sampling rate through

$$F_c = \frac{F_s}{T}. \tag{4.1}$$

Hence, the parameter T becomes a tradeoff between feature accuracy and classification rate. The more samples used per classification, the better the extracted features become at the cost of a lower classification frequency. Conversely, we may be able to generate feature estimates more rapidly by setting T to a lower value, but will in the process end up with worse feature estimates.

In this section, four different feature types are discussed aimed at capturing the geometric characteristics of a target surface. From a given data matrix $x(n, t)$ consisting of $T = 25$ consecutive sweeps normalized as described in section 3.6, we construct for each feature type a corresponding feature vector. We can then concatenate different feature vectors to form longer sets of features as we please.

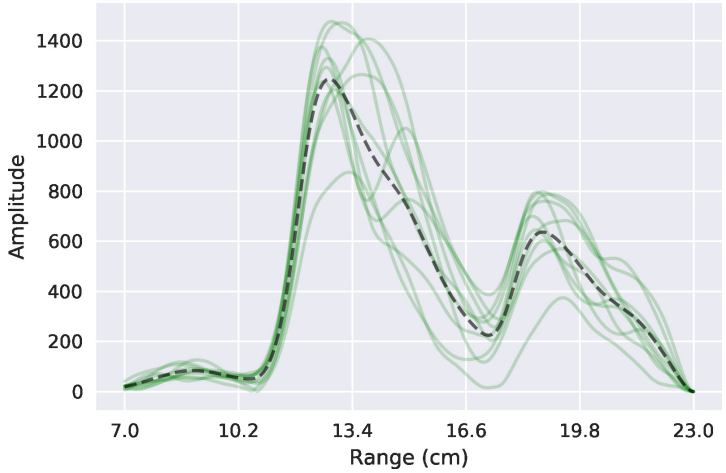

Figure 4.1: By averaging the absolute values of several consecutive sweeps we obtain a measure of the average sweep shape. The dashed line shows the average of individual absolute values of radar sweeps, shown as solid semi-transparent lines.

4.2.1 Envelope Estimation

First of all, we may characterize a sweep by its envelope form; that is, the shape of the absolute values of the radar sweeps. We do this by simply calculating the absolute value averages in slow time to construct the estimated envelope shape $\hat{x}(n)$ through

$$\hat{x}(n) = \frac{1}{T}\sum_{t=0}^{T-1}|x(n,t)|. \tag{4.2}$$

In Figure 4.1, we see what such an averaging process yield. Individual variances are suppressed forming a stable estimate of the sweep shape. With K fast time samples, the estimate of the expected envelope feature vector becomes

$$\mathbf{f}_x = \begin{bmatrix}\hat{x}(0) & \hat{x}(1) & \dots & \hat{x}(K-1)\end{bmatrix}. \tag{4.3}$$

4.2.2 Fourier Transform

As discussed in section 2.1.3, we can relate the velocity of a scatterer to its BF through (2.8). When an entire target scene is illuminated, computing a DFT in the slow time domain thus provides a *velocity profile* of the scene at every range, indicating the amount of each velocity is present for the range in

question. Due to the sampling settings outlined in chapter 3, no significant aliasing in the frequency spectrum should occur in theory, as the sampling rate was set with the Nyquist criterion in mind. In Figure 4.2, the DFTs of radar IQ data are shown for all surface types considered in this work. For visualization purposes, the square root of the absolute values of the DFT are shown. For grass (the top two plots), the frequency content is contained mainly in the vicinity of the zero-frequency bin, while the other materials have larger components at higher frequencies.

As for the feature vector, a T-step DFT along the slow time axis is first computed for each range, n, as

$$\mathbf{d}_n = \text{DFT}\big\{x(n,t)\big\}_t = \sum_{t=0}^{T-1} x(n,t) \exp\Big(-2\pi i \frac{kt}{T}\Big), \quad k = 0, ..., T-1. \tag{4.4}$$

After this, the transformations for all ranges are concatenated into a one-dimensional feature vector. This feature vector must be real-valued, as the machine learning classifiers considered in this work use real-valued inputs. The feature vector is formed by taking the absolute values of the frequency components through

$$\mathbf{f}_f = \begin{bmatrix} |\mathbf{d}_0|^\Delta & |\mathbf{d}_1|^\Delta & \dots & |\mathbf{d}_K|^\Delta \end{bmatrix}, \tag{4.5}$$

where $|\cdot|^\Delta$ denotes elementwise absolute values, and K is the number of fast time samples.

4.2.3 Autocovariance in Range

A second method to investigate time-domain dependencies is through the autocovariance function. This method adds the benefit of allowing for a specification of the number of covariance lags, q, which subsequently determines the number of features. Figure 4.3 displays absolute values of autocovariances estimated from 5000 sweeps for all surface types considered. As the most even surface, the tiled pavement shows the highest autocovariances, whereas grass - the arguably most uneven surface - indicates a low autocovariance. It should be noted that since only absolute values are shown, phase information has been lost in this plot.

In order to have a reasonable classification rate, see (4.1), only $T = 25$ sweeps are used for generating one feature vector. The biased estimate of the autocovariance function can be obtained through (Jakobsson, 2015)

$$\hat{r}_n(k) = \frac{1}{T} \sum_{t=k}^{T-1} \big(x(n,t) - \hat{\mu}_n\big)\big(x(n,t-k) - \hat{\mu}_n\big)^* \tag{4.6}$$

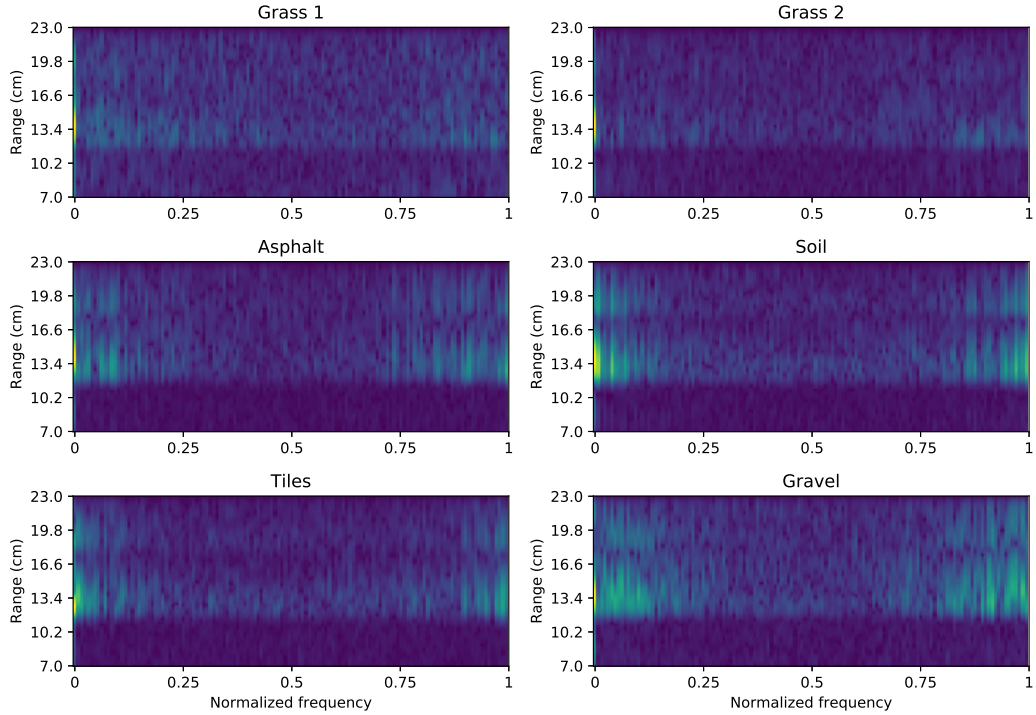

Figure 4.2: For each material a DFT is computed over 128 sweeps in slow time. For visualization purposes the square root of the absolute values of the complex-valued DFT is shown, highlighting differences in frequency content of the different surfaces.

where

$$\hat{\mu}_n = \frac{1}{T}\sum_{t=0}^{T-1} x(n,t). \tag{4.7}$$

The bias in this estimate is completely inconsequential as all features later are normalized to zero mean unit variance. Autocovariance estimation through (4.6) is based on the assumption that $x(n,t)$ are measurements of a wide-sense stationary process. For a rough real-world terrain, this assumption is optimistic. However, the estimates may nonetheless make for useful metrics for classification as they feature slow time frequency content. The features may be formed as

$$\hat{\mathbf{r}}_k = \begin{bmatrix} \hat{r}_0(k) & \hat{r}_1(k) & \dots & \hat{r}_{K-1}(k) \end{bmatrix}. \tag{4.8}$$

Noting that the autocovariance sequence at 0 lag produces only real numbers, as any complex number $z = a + bi$ multiplied with its conjugate has zero

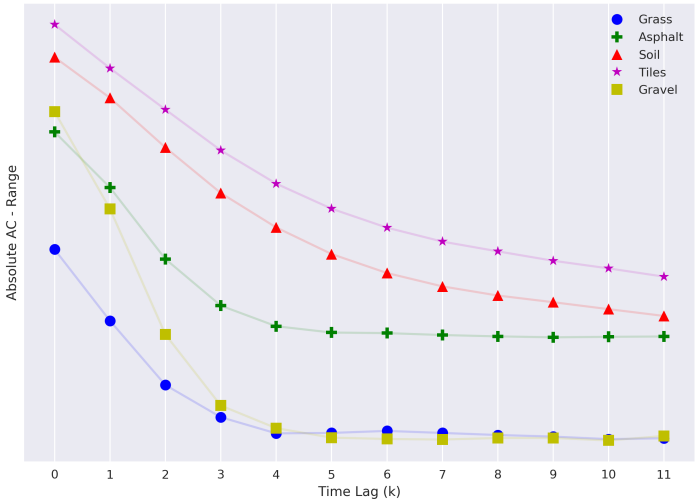

Figure 4.3: The first 11 autocovariance lags have been computed from a large number of slow time samples taken at roughly 14 cm for each surface considered in this work. The absolute values of the covariance lags are shown here for illustrative purposes. Note that only the zeroth lag is real-valued and that the others are complex valued.

imaginary part $\mathrm{Im}(zz^*) = \mathrm{Im}((a+bi)(a-bi)) = \mathrm{Im}(a^2+b^2) = 0$, we may form the full real-valued feature component for q autocovariance lags $\mathbf{f}_r$ as

$$\mathbf{f}_r = \begin{bmatrix} \hat{\mathbf{r}}_0 & \mathrm{Re}(\hat{\mathbf{r}}_1) & \mathrm{Im}(\hat{\mathbf{r}}_1) & \ldots & \mathrm{Re}(\hat{\mathbf{r}}_q) & \mathrm{Im}(\hat{\mathbf{r}}_q) \end{bmatrix}. \tag{4.9}$$

4.2.4 Autocovariance in Energy

Although the sweep normalization process rendered absolute measurements of RCS pointless, we are still able to investigate any time-dependent structure through the autocovariance function. First, we estimate the energy in each radar sweep $v(t)$ and the average energy $v_a(t)$ in T number of sweeps, and then estimate the real-valued autocovariance sequence $h(k)$ as

$$v(t) = \frac{1}{N} \sum_{n=0}^{N-1} x(n,t) x^*(n,t) \tag{4.10}$$

$$v_a = \frac{1}{T} \sum_{t=0}^{T-1} v(t) \tag{4.11}$$

	Abs.	AC-E	AC-R	DFT	Feats.	SVM Acc. [%]
Config 1	X				28	92.23
Config 2	X	X	X		174	97.44
Config 3				X	700	97.37

Table 4.1: Three feature configurations of the four introduced features are compared. An X indicates that the feature is used in the configuration. Above, Abs indicate estimated envelopes, AC-E and AC-R autocovariances in energy and range, respectively, and DFT the discrete Fourier transform. Ideally we want as few features and as high linear separability, shown through an SVM accuracy score, as possible.

$$h(k) = \frac{1}{T}\sum_{t=k}^{T-1}\big(v(t) - v_a\big)\big(v(t-k) - v_a\big)^*. \tag{4.12}$$

As $h(k)$ only consist of real values, the energy autocovariance feature vector $\mathbf{f}_h$ is formed as

$$\mathbf{f}_h = \begin{bmatrix} h(0) & h(1) & ... & h(q-1) \end{bmatrix}, \tag{4.13}$$

where q is the number of covariance lags.

4.2.5 Reducing the Range Swath

Before performing any feature processing, data is downsampled by a downsampling factor of $D = 20$ as was described in section 3.5. However, after examining a number of radar sweeps, it is clear that ranges below approximately 11 cm only contain little useful information. This can clearly be seen in the dark regions of the DFTs in Figure 4.2. Similarly, little information is found at longer ranges. Therefore, we extract samples from an intermediate region when downsampling, modifying equation (3.4) to

$$r_{1,D}(d,t) = r_1(50 + dD, t), \quad \text{and} \quad r_{2,D}(d,t) = r_2(50 + dD, t) \tag{4.14}$$

for $d = 0, ..., 13$. With this definition, the downsampled radar sweeps consist of 14 evenly spaced range indices from ranges 9.4 to 21.9 cm. Thus, the resulting data matrix used for generating one set of features has $14 \cdot 2$ complex ranges for the two sensors, and $T = 25$ slow time sweeps.

Figure 4.4: Origin of the 174 features in features in feature configuration 2.

4.3 Tested Feature Combinations

Four different feature extraction methods have now been described - the average signal shape, the autocovariance in range and energy and the Fourier transform. Each method generates multiple features. However, we are not limited to feed our model with features taken from a single one of the methods. In table 4.1, results from three feature configurations are tested. Each of these combinations was tested in a support vector machine classifier (SVM, described in further detail in chapter 5), to evaluate which was the most efficient. The first one, which simply involves the averaged sweep envelope, requires much fewer features than the other two combinations - 28 positive real numbers for the 28 ranges. But looking at its performance, these features are not enough to reach the accuracy attained in the other two cases.

By instead selecting autocovariances with 2 lags as features the accuracy increases by over 5 percentage points. Here we also include the averaged absolute values of sweeps as features, as this information is lost when subtracting the mean during the computation of the autocovariances. This configuration generates a feature vector with 174 elements, as illustrated in Figure 4.4. 28 of the features come from the absolute values and 6 from the energy autocovariance. The majority of the features come from the autocovariance in range. For each of the 28 range bins there is one real-valued feature representing variance followed by two complex autocovariances for each range bin. This should give us $28 + 2 \cdot 28 = 84$ features, but as the real- and imaginary parts of complex numbers are split up in two features, we obtain $28 + 2 \cdot 2 \cdot 28 = 140$ features. The total number of features therefore becomes $28 + 6 + 140 = 174$.

We can also choose to look at the Fourier transform, as in configuration 3. Using the DFT, we reach a similar accuracy as in configuration 2 but get some 700 features. The 700 features are a result of computing a discrete Fourier transform along the slow time of all 28 ranges over 25 samples[1]. The length of each transform is 25, and by concatenating them all into one feature vector, we end up with $28 \cdot 25 = 700$ elements.

[1]Ideally, the DFT should be computed as an FFT over a number of slow time samples which is a power of 2. However, to get a fair comparison of the feature extraction methods, we utilize the same number of samples for all methods.

The high accuracy and low number of features of configuration 2 makes it the clear top performer of the three, and will be used for the remainder of this report for further investigation.

4.4 Feature Principal Component Analysis

Through the feature extraction methods described in previous sections, we obtain high dimensional feature vectors. Getting an intuitive feel for such high-dimensional data is difficult as direct plotting is limited to three dimensions.

Principal component analysis (PCA) is a classical technique in statistical data analysis which takes a large set variables from a multivariate dataset and finds a smaller set of variables with less redundancy. Critically, PCA finds a rotated orthogonal coordinate system such that the elements of the set become uncorrelated (Hyvasrinen et al., 2004). By projecting elements onto the principal axes corresponding onto the directions of maximal variance we obtain a good approximation of the original data in a lower dimension.

After having chosen to proceed with feature configuration 2 in table 4.1, the 174 dimensions can be reduced to 2 dimensions using PCA. By projecting feature vectors onto the two directions of maximal variance, we get a good visualization of the separability of the different surfaces as seen in Figure 4.5. Note that each dot in this plot corresponds to one feature vector, and that the figure represents features extracted from the full set of 175 minutes of data sampled at 200 Hz.

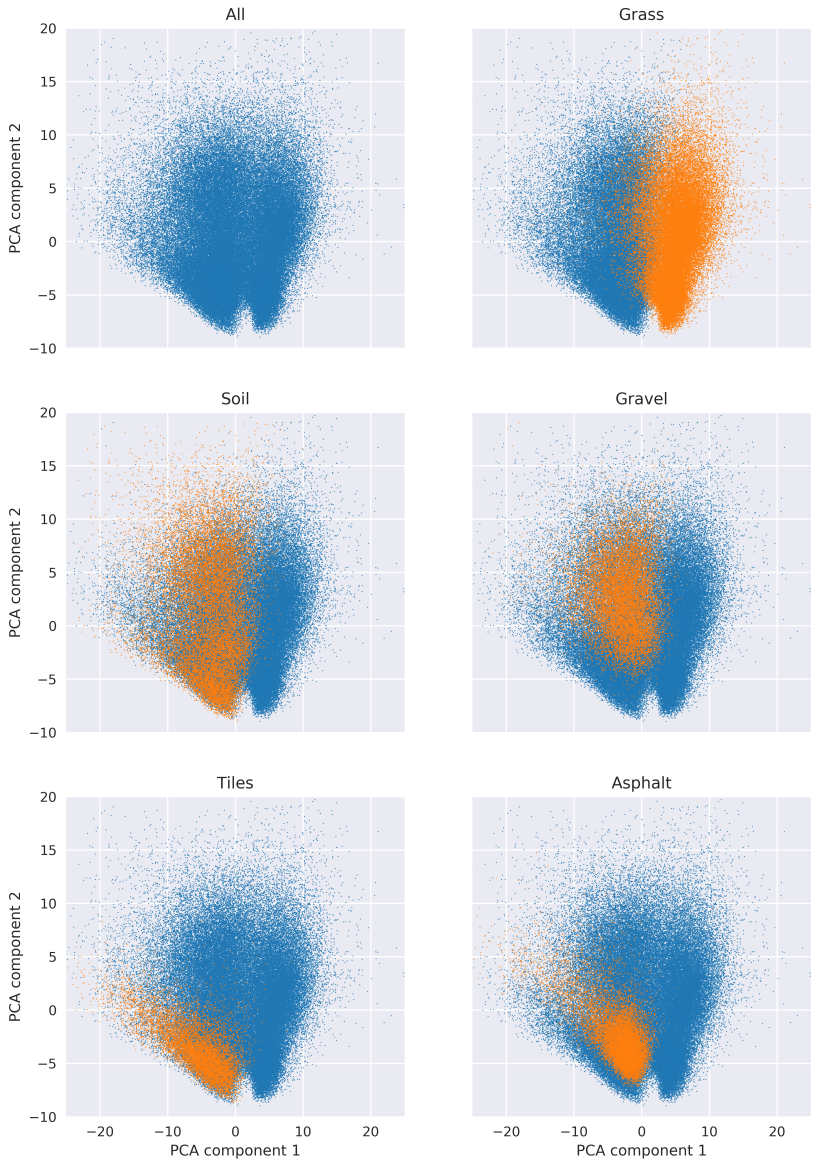

Figure 4.5: By performing a principal component analysis on the feature vectors we project 174 dimensional feature vectors onto two dimensions corresponding to directions of maximum variance. This allows for us to visualize how separable the different surface types are. In each plot, the samples corresponding to the titled surface are highlighted.

Chapter 5

Machine Learning

In this chapter, a few classification models are described and evaluated. Two linear and four nonlinear models for supervised learning are compared using leave-one-out (LOO) cross validation on the acquired data. The workflow for all models apart from the last follows the chart in Figure 5.1 where extracted features are used for classification.

5.1 Linear Models

Linear classifiers perform a classification decision based on the value of a linear combination of input data (Santos, 2018). In the binary case, this corresponds to dividing the dataset with a single hyperplane, splitting data into two categories. New test data is predicted to belong to a category based on which side of the hyperplane it is on. In this work, two linear models are considered: linear discriminant analysis (LDA) and a support vector machine (SVM) with a linear kernel.

5.1.1 Linear Discriminant Analysis

Linear discriminant analysis is an algorithm commonly used for dimensionality reduction (Raschka, 2014), but can also be used for classification purposes. LDA projects given data onto a subspace in a manner that separates the classes as much as possible.

LDA bears many similarities to PCA in that it is based on finding eigenvectors with the largest corresponding eigenvalues, with the key difference that PCA disregards class labels whereas LDA does not. For further reading about LDA, we refer to (Raschka, 2014).

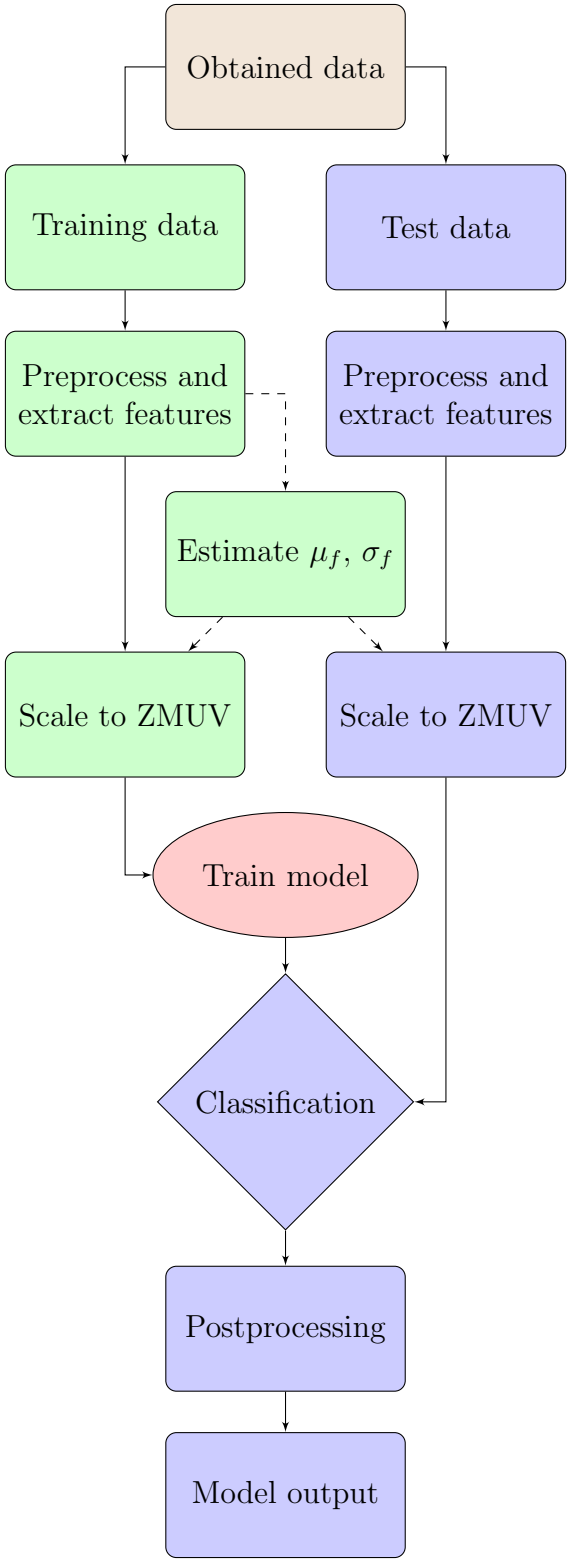

Figure 5.1: Flowchart of the procedure of generating classifications. Both training- and test data are preprocessed and scaled so that each feature has zero mean and unit variance (ZMUV). The training data is used to obtain model weights, which become inputs to the classifier. Using these weights, predictions are made on the test data. These predictions then go through a postprocessing which results in the final output.

5.1.2 Support Vector Machines

Support vector machines are one of the most popular linear models for supervised learning. The simplest SVM does not introduce any nonlinearities through any kernel tricks, and generates a linear hyperplane that separates two sets of labeled data similarly to LDA. Unlike LDA however, an SVM is not an eigen-based method, making SVMs less prone to outliers. Finding a good hyperplane can be done in numerous ways. An SVM finds what it regards as the best hyperplane by optimizing after parameters that maximize the high-dimensional distance between the hyperplane and sample points closest to it (Boswell, 2002).

5.2 Nonlinear Models

In many cases, data cannot be satisfactorily separated by a linear classifier. In the binary case this means that two classes in a dataset cannot be properly split by any one hyperplane. For such data non-linear algorithms may perform better. In this work, four different nonlinear machine learning classifiers are tested: random forest (RF), logistic regression (LR) and two types of artificial neural networks (ANNs).

5.2.1 Random Forest

Random forest (RF) is an ensemble learning method that can be used for both regression- and classification problems. The name stems from that the method is based on a collection of randomly initiated *decision trees*. A decision tree is structured as a sequence of simple questions, or decision rules. These rules typically consider whether its input is equal to or smaller than some value. The answers to these questions form a path in the decision tree, leading to an end node which corresponds to a prediction.

Random forest segments the training data into n parts, and induces a decision tree from each group of data. Thus, there are n predictors that work independently, and by selecting the most common prediction, random forest yields a robust result with little risks of overfitting due to the combined results of multiple decision trees. On top of that it offers a very high accuracy in a wide variety of applications, while still maintaining an intuitive model structure that allows us to, for instance, estimate which features are important (Breiman, 2002). In Google's *Project Soli*, random forests were used for millimeter-wave radar gesture recognition with impressive results (Lien et al., 2016).

5.2.2 Logistic Regression

Logistic regression (LR) is similar to linear regression where the parameters $\{b_i\}$ are optimized to model the linear relationship between the inputs $\{X_i\}$ and the continuous output Y

$$Y = b_0 + \sum_{i=1}^{n} b_i X_i. \tag{5.1}$$

However, logistic regression is rather used for modeling a binary variable, $Y_{log} \in \{0, 1\}$. Therefore, a continuous output is unsuitable. Instead, the expression in (5.1) is mapped to the open interval $(0, 1)$ using the sigmoid function, defined as $\sigma(z) = \frac{1}{1+e^{-z}}$. Thus, LR maps the input variables to

$$Y_{log} = \frac{1}{1 + \exp(-(b_0 + \sum_{i=1}^{n} b_i X_i))}. \tag{5.2}$$

This value can be interpreted as the probability that a feature vector with a set of features, $\{X_i\}$, belongs to class 1. Note that for $Y \to \infty$ we have that $Y_{log} \to 1$ and for $Y \to -\infty$ that $Y_{log} \to 0$. The model parameters in LR are trained in such a way that Y becomes large for input combinations representing samples of class 1, and small for samples of class 0.

For further details about LR, including details on how its parameters are optimized, we refer to (Shalev-Shwartz and Ben-David, 2016).

5.2.3 Artificial Neural Networks

Artificial neural networks constitute a class of nonlinear models designed to mimic biological neural systems (Rojas, 1996). ANNs consist of multiple layers of neurons, or nodes. The networks are structured with an input layer followed by one or more hidden layers and an output layer (Logan, 2017). Figure 5.2 illustrates a simple network which takes a feature vector $\mathbf{x}^{(0)} = [x_1^{(0)}\ x_2^{(0)}]^T$ as input. By multiplying the feature vector with a set of weights $\mathbf{w}^{(1)}$ the features are propagated through the network, generating a set of new node values. In the network in Figure 5.2, the output from the input layer, forwarded to the hidden layer, becomes

$$\mathbf{x}^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{21}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} \\ w_{13}^{(1)} & w_{23}^{(1)} \end{bmatrix} \cdot \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix}. \tag{5.3}$$

When the nodes in the hidden layer receive the propagated values, they may range anywhere from negative to positive infinity. Using an *activation function* f, the node transforms the input to a more suitable format in terms of

whether the node should be "active" or not (or, in biological terms, whether the neuron should fire) (Kriesel, 2007). The function f also introduces nonlinearity in the model, making ANNs capable of solving nonlinear problems.

In (5.3) we omit the bias term in the hidden layer labeled "1" in Figure 5.2. A bias term is an external, constant input to an input or hidden layer. The bias term is independent of inputs from the preceding layer and increases the flexibiliy of a model, as it allows for translations of the activation function (Kohl, 2010).

Next, the output from the activation function is propagated with a new set of weights $\mathbf{w}^{(2)}$

$$\mathbf{x}^{(2)} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(2)} & w_{21}^{(2)} & w_{31}^{(2)} \\ w_{12}^{(2)} & w_{22}^{(2)} & w_{32}^{(2)} \end{bmatrix} \cdot \begin{bmatrix} f(x_1^{(1)}) \\ f(x_2^{(1)}) \\ f(x_3^{(1)}) \end{bmatrix}. \tag{5.4}$$

The propagated values in $\mathbf{x}^{(2)}$ are the inputs to the output layer which after application of the activation function produces the output that is interpreted as a classification prediction. The model weights $\mathbf{w}^{(i)}$ of a neural network are set through some version of the backpropagation algorithm during the model training phase. For details about backpropagation, we refer to chapter 7 in (Rojas, 1996).

By introducing more than one hidden layer, an ANN can be called a deep neural network (DNN). With increased number of hidden layers, DNNs are capable of extracting more complex patterns from data such as for image recognition (Szegedy et al., 2018) or modeling of speech (Hinton et al., 2012). Since we in the preceding chapter performed feature extraction, it should in this case not be necessary to have more hidden layers than two, since useful features should already readily available from the feature extraction stage.

Hyperparameters of Neural Networks

The sizes of input and output layers are determined by the classification problem, but the internal structure of a DNN classifier can be structured freely. The number of hidden layers and the number of nodes therein should be selected with care, as increasing the model size rapidly increases the computational complexity and the number of trainable parameters.

While the number of hidden layers and the number of nodes in each layer relates to the architecture of the network, there are several parameters that are related to the training process. The *batch size* specifies how many samples are propagated through a network between each model weight update. A small batch size has the benefit of requiring little memory and converging rapidly, but at the same time impairs the gradient estimate (Brownlee,

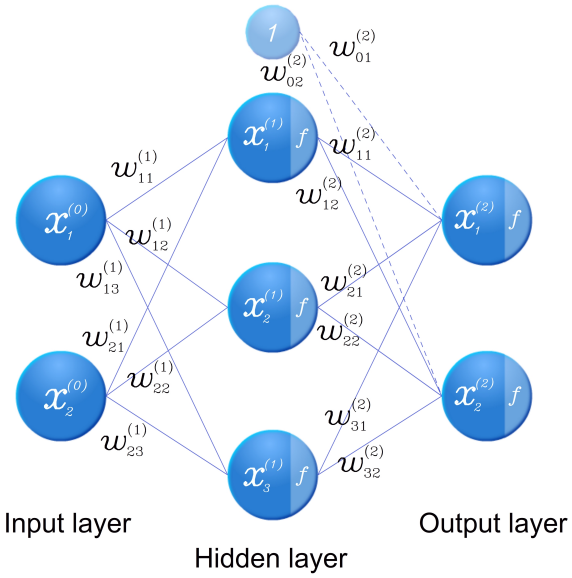

Figure 5.2: A simple neural network with 2 inputs, 2 outputs, and one hidden layer. The top node in the hidden layer is a bias term which is added for increased network flexibility.

2017). With a larger batch size the gradient is more accurately estimated but convergence is slower.

When it comes to reducing overfitting, *dropout* is a method commonly utilized in neural networks. By randomly disabling some of the hidden layer nodes during the training phase the model is forced to become more general, not relying on any specific set of nodes for accomplishing its target. A node n in layer k is disabled by setting its weights $w_{nj}^{(k)}$ to 0, where j ranges from 1 to the number of nodes in the sequent layer. For any node in a layer that features dropout, the *dropout rate* specifies the probability that the node will be disabled.

If the total number of feature vectors is m, and the batch size is b, there will be $\lceil m/b \rceil$ forward and backward propagations, and an equal number of model weight updates. Each batch is only propagated once, but to extend the model training process further, we can specify number of *epochs*. This hyperparameter determines how many times each batch will be fed through the model. One epoch is often not enough for the weights to fully converge (Kriesel, 2007), but increasing the number of epochs naturally increases the training time. Furthermore, too many epochs may put the model at risk of overfitting. In (Prechelt, 1998), this issue is addressed, along with proposed strategies to avoid it.

The hyperparameters of the DNN model were optimized using the free

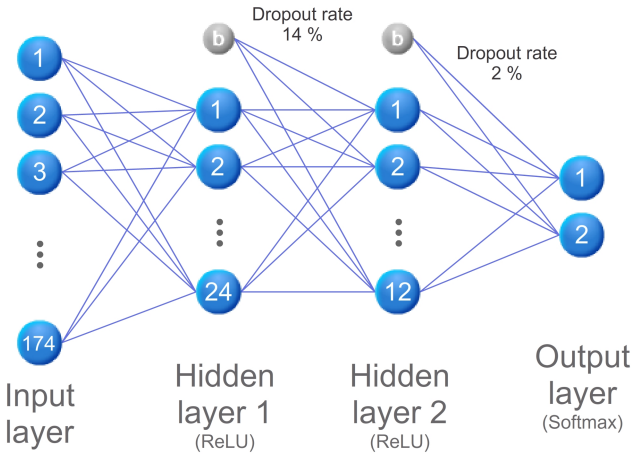

Figure 5.3: Using Hyperas a network with two hidden layers was optimized in terms of number of nodes in each layer, dropout rate, batch size and optimization algorithm. The resulting network has 25 and 13 nodes (including bias terms) in the hidden layers, and dropout rates of 14 and 2 percent, respectively.

optimization tool Hyperas (available at *https://github.com/maxpumperla/hyperas*). Hyperas selected the number of nodes, the dropout rates, the batch size and the optimization algorithm (from the three optimization algorithm options of RMSprop, Adam, and Stochastic Gradient Descent natively available in Keras).

With the selections made by Hyperas, the network in Figure 5.3 is obtained. The two hidden layers have 24 and 12 nodes with dropout rates of 14 and 2 percent, respectively. Both layers have the activation function $f(x) = \max(0, x)$ often referred to as rectified linear unit (ReLU) and the output layer has a softmax activation function. Furthermore, the batch size in the learning phase is 32, and the number of epochs is set to 20. Finally, the optimization algorithm preferred by Hyperas is RMSprop.

5.2.4 Convolutional Neural Network with Long Short Term Memory

In previous models normalized data went through a feature extraction process before going into model training. For this model, on the other hand, no feature extraction is performed. Instead, several consecutive radar sweeps are used as input. We can view the sequence of slow time samples for any one range as a one-dimensional time series containing the velocity information found in the BFs as was discussed in section 2.1.3. Previously we utilized this temporal information by calculating Fourier transforms and estimating

autocovariance coefficients for a few lags, but we may also exploit this temporal behavior in a more immediate way.

Recurrent neural networks are models that work with sequential data by having feedback within individual layers in the network structure (Karim et al., 2018). The problem with these networks, however, is that they suffer from a quickly vanishing or exploding gradient, and can only sustain a short term memory (Pascanu et al., 2013). A way to combat this is to use a neural network layer type called long short term memory (LSTM).

LSTM layers have previously been used successfully for classifications in radar applications. For instance, in (Jithesh et al., 2017) the method was used in a classification model that was able to distinguish multiple classes of flying targets with high accuracy. The theory behind these layers are thoroughly described in for example (Hochreiter and Schmidhuber, 1997). Another successful approach for time series classifications is convolutional neural networks (CNNs) (Karim et al., 2018). In (Capobianco et al., 2018), time series of radar data were preprocessed and used as input to a CNN. The network was used to predict what types of vehicles were driving past a radar sensor and managed to do so with a good success rate.

A combination of the LSTM layer with a CNN is proposed in (Karim et al., 2018). This proves to be a significant improvement from just using CNNs when classifying time series. The architecture of the model we use in this work is similar to this classifier with only a few tweaks of parameter values. The model essentially concatenates the outputs from an LSTM network and a network consisting of three one-dimensional convolution layers. For more details we refer to (Karim et al., 2018).

5.3 Model Evaluation

When evaluating a machine learning model, one must decide how a dataset should be split into data used for training and data used for evaluation. There exist a great number of strategies to split data into these two (Raschka, 2016). One of the simplest ways of dividing the dataset is to randomly select a portion of samples to use for training and use the remainder for evaluation of model performance. These two sets are commonly referred to as the training and the validation set.

There is one chief issue with this random-selection methodology. If we, for instance, are predicting using features from a small data matrix found from a specific grass sample, the model has trained on a large portion of not only other lawns on different days, but also from the *same* lawn on the *same* day. This means that a model under investigation has trained on very

similar samples with very high resemblance to what it is currently attempting to classify. The authors consider this to be cheating, as such a scenario is not particularly realistic. It would be more informative to test how the model manages when trained only with samples from other lawns without help from its neighboring samples, as this is what the model would be faced with in any real-world scenario. For this reason, we will employ leave-one-out (LOO) cross validation next explained.

5.3.1 Leave-One-Out Strategy

For this project 42 data matrices were collected from surfaces described in table 3.1. Each surface was acquired either on a separate day or in a separate location from the others. When evaluating how well a model perform we would like to test it in as realistic of a scenario as possible, where only training data from other days and locations are available.

Because of this we use the leave-one-out cross validation strategy. This strategy involves cycling through each of the 42 data matrices and leaving out one and training on the remainder, as shown in Figure 5.4. We then evaluate model performance on the left-out data matrix. This strategy ensures that when we classify a surface, the model has not seen any data from the same measurement session and we thus get a more realistic performance providing an indication of the model's robustness. The LOO strategy also has the benefit that all data for the most part is used in training, instead of only a portion as is required in the random selection method (Raschka, 2016). In table 5.1 the LOO accuracy scores of all six models are listed for comparison.

5.3.2 Selecting a Model

From table 5.1 we can see that all models perform well on asphalt and tiled surfaces. Gravel, soil and grass are occasionally harder to classify correctly. Looking back at the PCA in Figure 4.5 this should not come as a great surprise, as these are the main surface types with significant overlap of their two principal components. As for gravel, there is one data matrix that sticks out - Gravel 2. The accuracy of this measurement sessions is well below the average accuracy regardless of model. This could be because this particular gravel data matrix contained characteristics not captured by the other gravel surfaces resulting in several misclassifications. It also possible that some temporary problem occured in the measurement setup.

The LSTM and CNN model has a great accuracy span ranging from 59% to having multiple perfect scores. While it has a leading median score of 99.8%, it also contains several surfaces where it performed very poorly. This

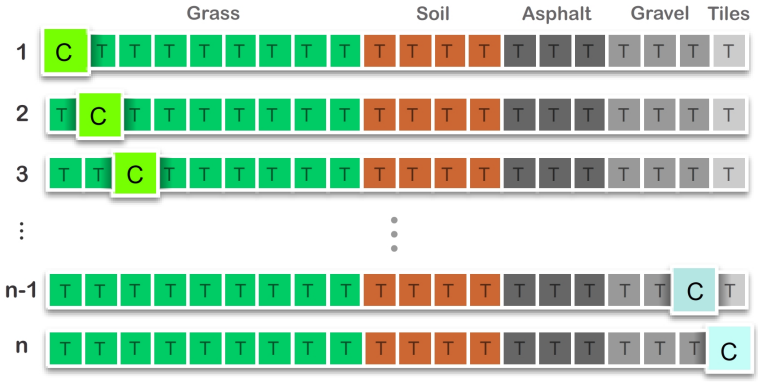

Figure 5.4: With the leave-one-out strategy data is split up into n parts. The model evaluation is then performed in n steps. Each time, one part of the data is excluded from the training and evaluated upon. The T's mark which parts of the data that are used for training, and the C's show which data matrix is used for classification.

gives it the highest standard deviation among all the methods, suggesting it is not as robust as its competitors. It is possible that this could be remedied with a little bit of fine-tuning, but due to a limited amount of time we disregard this model, making the DNN-model the top performer with the greatest median and mean as well as the lowest standard deviation. Thus, for the remainder of this report, the DNN model is used.

Material	LR	RF	LSTM CNN	SVM	LDA	DNN
Grass 1	97.9	98.05	59.0	95.75	96.8	97.45
Grass 2	99.8	99.5	83.8	96.2	100.0	99.95
Grass 3	94.35	86.75	94.5	91.4	92.6	95.3
Grass 4	95.7	96.45	100.0	91.35	93.65	97.8
Grass 5	96.65	97.25	97.4	93.95	95.9	95.55
Grass 6	96.35	98.8	95.9	92.95	97.2	99.3
Grass 7	99.9	99.85	99.9	99.65	99.95	100.0
Grass 8	97.0	96.65	97.5	96.85	92.7	96.95
Grass 9	97.75	97.7	99.8	97.8	96.3	98.75
Grass 10	99.2	98.65	99.9	99.0	97.65	99.7
Grass 11	99.35	98.7	99.9	99.25	97.55	99.8
Grass 12	99.6	99.4	99.9	99.6	99.2	99.65
Grass 13	100.0	100.0	100.0	100.0	99.85	100.0
Grass 14	96.35	96.8	98.6	95.6	89.85	98.1
Grass 15	97.6	95.45	99.8	93.1	87.4	99.05
Grass 16	97.55	98.0	99.8	95.8	94.05	98.85
Grass 17	95.4	92.85	98.4	94.6	89.9	95.0
Grass 18	97.35	94.4	96.9	96.55	93.65	95.6
Asphalt 1	100.0	100.0	99.9	100.0	100.0	100.0
Asphalt 2	99.95	100.0	99.0	100.0	100.0	100.0
Asphalt 3	100.0	100.0	99.7	100.0	99.95	99.2
Asphalt 4	99.9	99.85	100.0	99.9	99.95	100.0
Asphalt 5	100.0	100.0	100.0	100.0	100.0	100.0
Asphalt 6	100.0	100.0	100.0	100.0	99.9	99.95
Gravel 1	99.45	99.85	99.8	99.9	99.2	99.7
Gravel 2	82.65	97.1	86.6	88.2	89.3	94.3
Gravel 3	99.95	99.55	100.0	100.0	99.95	99.75
Gravel 4	98.95	99.6	99.9	99.55	99.9	99.8
Gravel 5	99.85	99.8	99.9	99.95	100.0	99.7
Gravel 6	99.75	99.7	100.0	99.75	99.9	99.55
Soil 1	99.85	100.0	99.7	99.95	99.7	99.9
Soil 2	99.35	99.6	99.4	99.85	99.4	99.75
Soil 3	99.75	99.85	100.0	100.0	99.9	99.85
Soil 4	97.8	96.3	88.9	98.9	97.4	95.95
Soil 5	96.25	95.45	100.0	96.55	96.55	93.95
Soil 6	96.35	94.0	99.3	96.85	95.5	99.85
Soil 7	92.65	95.15	99.3	90.25	92.4	100.0
Soil 8	94.75	95.65	84.7	90.95	92.9	95.15
Tiles 1	99.35	99.4	100.0	99.55	99.3	99.35
Tiles 2	99.95	99.75	99.9	99.95	99.6	99.95
Tiles 3	99.95	99.9	100.0	100.0	100.0	99.95
Tiles 4	99.95	99.7	99.5	99.95	100.0	94.45
Mean	97.96	97.99	97.06	97.37	97.02	98.5
Median	99.35	99.4	99.8	99.4	99.2	99.68
SD	3.05	2.63	7.23	3.3	3.58	1.96

Table 5.1: LOO accuracies for all collected data matrices evaluated for accuracy using each of the six machine learning classifiers.

Chapter 6

Postprocessing and Data Augmentation

In the previous chapter, a DNN classifier was found to be the top performing model for the surface classification problem investigated. In this chapter, two improvements to this model are introduced. First, we use a basic data augmentation technique to increase the size of the training data. Then, outlier suppression and detection methods are discussed.

6.1 Data Augmentation

A recurring problem in training ANNs is that there simply is not enough training data for proper convergence (Lemley et al., 2017). Too little data will make a network prone to overfitting, which means that it becomes highly biased to what it has seen in training and that it will subsequently perform poorly on any validation or test set. In the preceding chapter, dropout was introduced for this particular purpose. Examining the cross validation accuracies it is clear that reasonable accuracy is attained without any further means.

However, by *augmenting* the data, we can further increase the training set size, and subsequently potentially increase model performance. Data augmentation is the process of supplementing a dataset with similar data created from the same dataset. How one augments a dataset is completely dependent on the data set. In for instance computer vision, data augmentation often involves rotating, translating, blurring or in some other way modifying existing images (Lemley et al., 2017).

In the present case of increasing the number of smaller data matrices with T slow time samples from a given data matrix with Q slow time samples,

A	LOO Acc.
1	98.50
5	98.54

Table 6.1: Leave-one-out accuracies with and without data augmentation.

we can allow for overlapping batches. Previously, when batches of length T were generated from Q slow time samples, every T sweeps produced one data matrix providing a total of Q/T matrices for analysis. However, noting that any sequence of T radar sweeps is a valid data matrix, we may form *overlapping* matrices from every T/A samples, where A is some integer factor selected so that T/A becomes an integer. This produces $(A-1)(Q/T-1)$ additional matrices providing a total P number of data matrices according to

$$P = \frac{Q}{T} + (A-1)\Big(\frac{Q}{T} - 1\Big) = \frac{AQ}{T} - A + 1. \tag{6.1}$$

Setting $A = 5$, one data matrix consisting of 50,000 slow time samples normally yielding 2,000 smaller data matrices with $T = 25$ slow time samples, instead produces 9996 matrices. Thus, with this method we are able to increase the number of training data by almost a factor of 5. In table 6.1 the effects of data augmentation in terms of performance with $A = 5$ is compared to having no augmentation.

6.2 Outlier Suppression and Change Detection

Even with an optimized model with tons of training data, erroneous predictions are inevitable in any real-world scenario. Prediction probabilities are produced by the model rapidly, 8 times a second for a sampling rate of 200 Hz and a batch size $T = 25$ according to equation (4.1). With such errors present in the prediction confidences of the model, what is a reasonable strategy to detect when a change in surface has occurred while still allowing for occasional erronous predictor outputs?

Many elaborate statistically appealing methods for change detection are presented in (Basseville and Nikiforov, 1993). These methods require some basic assumptions on the data it attempts to detect a change from, such as for the data having constant probability distribution before and after a parameter change occurs. This, however, renders these methods difficult to use in the present use case, as we are dealing with the output of an artificial neural network producing predictions with unknown structure. This

is reinforced by examining what the model outputs, see the top Figure in 7.3. Predictions remain extremely stable for long periods of time, with occasional outlying predictions every now and then. This makes, as far as we can tell, the methods presented in (Basseville and Nikiforov, 1993) unsuitable for detecting change at the softmax output.

The perhaps most effective way to suppress outliers in a sequence of data is through some form of median filtering (Yin et al., 1996). Median filters are robust against impulsive-type noise, a property that cannot be achieved by traditional linear filtering techniques. The regular form of a median filter simply takes the median of current and previous datapoints, resulting in an output significantly less sensitive to inconsistencies (Pearson, 2002). For a sequence, $\{p_k\}$, of predictions produced by the DNN we apply a median filter of length L to render a final prediction confidence at time instance i (Yin et al., 1996)

$$\hat{p}_i = \text{median}\{p_k\}_{k=i-L+1}^{i}. \tag{6.2}$$

By specifying some decision threshold, $\xi \in [0, 1]$, a classification, c_i, can be produced as

$$c_i = \begin{cases} 0 & \text{if } \hat{p}_i < \xi \\ 1 & \text{otherwise.} \end{cases} \tag{6.3}$$

The drawback of using median filtering, or any filtering for that matter, is that the detection of a surface change is delayed by a few steps. Thus, L must be selected so that the device has not moved too far before the change detection has occurred, but also such that the filter is still capable of sufficiently suppressing outlying predictions.

Chapter 7

Results and Discussion

In this section the best models' capabilities are tested and discussed. Up to this point the model has been evaluated on labeled data, providing a clear ground truth to evaluate the models' predictions on. In this chapter the model will instead make predictions on unlabeled data and instead output its confidence in its prediction. We then discuss various topics with regards to the results found and some overall considerations.

7.1 Real and Artificial Transition Regions

While we previously mainly have focused on LOO accuracies, we have not in detail studied what the neural network softmax output that we base our classification on looks like. We are specifically interested in examining how well performance is when the robot moves across a *transition region*, a region where the surface type changes from grass to non-grass and vice versa. Ideally, the prediction should rapidly change according to the new target surface.

This can be accomplished in two different ways. The first revolves around creating a test set through concatenation of sequences of radar sweeps from a few different data matrices. By selecting alternating surface types, we artificially generate transition region data which we can predict on. By examining these predictions we get a feel for what output the predictor could generate when moving from one surface to the next. The second method is to use real-world measurements from when the robot traverses an actual transition region. We should see that when the robot has reached the transition its predictions change accordingly.

In the first test, samples from four surfaces are concatenated into a sequence: grass, asphalt, grass and tiles, where the two grass samples were

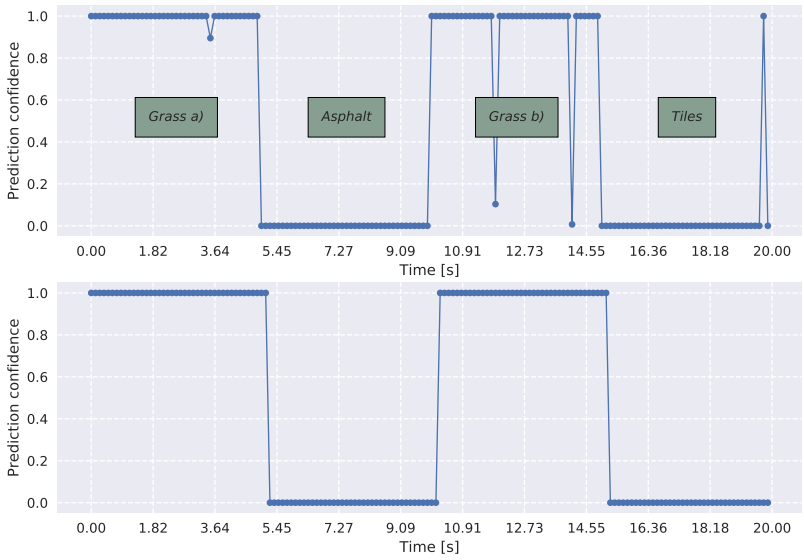

Figure 7.1: Predictions on an artificial transition region created using samples from four different regions. The bottom figure shows the median filtered predictions with filter length $L = 5$.

taken from different data matrices. The predictions are shown in the top part of Figure 7.1. In the lower part of this figure, the predictions are median filtered with filter length $L = 5$ as was discussed in section 6.2. The outliers are suppressed, rendering accurate predictions of the different surfaces.

Two more challenging surfaces, according to table 5.1, are soil and gravel. In Figure 7.2 predictions on artificial transitions including these are shown. Something worth noting in the two figures is that the surface transition is delayed by two steps after the median filtering. It is important to be aware of what distance this corresponds to in the physical world, since we do not want to detect an edge too long after it has already passed. Each prediction uses 25 samples, and with a delay of two predictions this means a 50 sample delay. With the sampling rate at 200 Hz (see table 3.2), this means a 0.25 second delay in the edge detection, which for a robot traveling at the speed of $v = 0.3$ m/s means a spatial delay of $0.3 \cdot 0.25 = 0.075$ m. This distance may be reduced by increasing the classification rate. This could of course be done by altering T or F_s in (4.1), but also through making predictions more frequently by allowing the model to make a prediction at intermediate

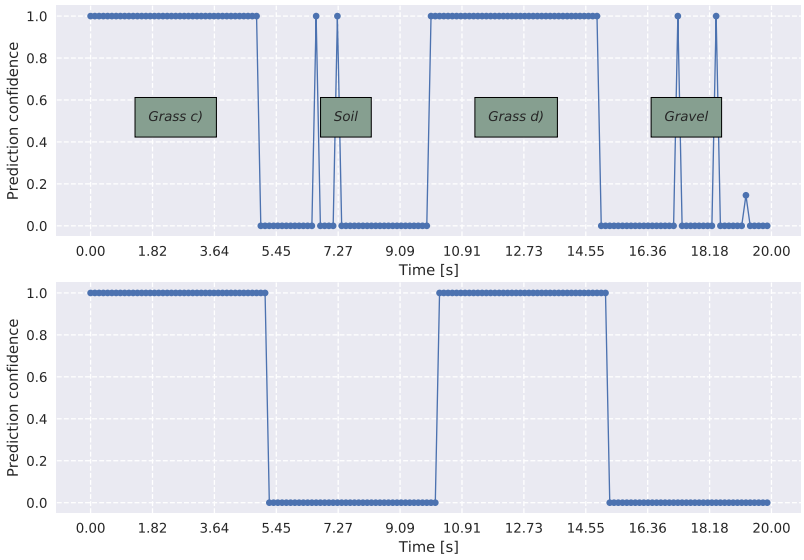

Figure 7.2: Predictions on an artificial transition region created using samples from four different regions.

points, given the T *most recent* sweeps.

Looking at the high LOO accuracies in table 5.1, the solid classifications accuracies in artificial transitions should not come as a surprise, as the LOO accuracies and the artificially created predictions were created similarly. What can be seen that we didn't know from table 5.1 is that not only does the model classify correctly, but usually does so with a very high confidence.

To test the model further, data was collected when the test robot moved from one surface to another. Figure 7.3 shows a transition from grass to tiles, and Figure 7.4 a transition from grass to gravel. Focusing on the median filtered predictions, we see that outside the transition region, the model works well for classifying either surface. Looking at the transitions sections, we see a rather sharp switch in prediction confidence. Nonetheless, this switch is not immediate. This is likely due to that the wide combined field of view of the two sensors captures both regions simultaneously, and thus the model tries to make a prediction on data obtained from a mixture of the two surface types.

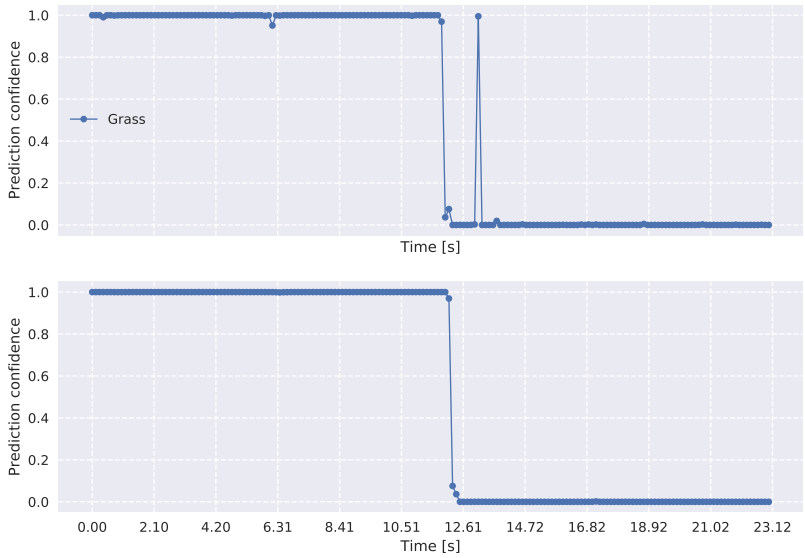

Figure 7.3: Predictions from a real-world test where the robot traversed a transition from grass to a tiled pavement.

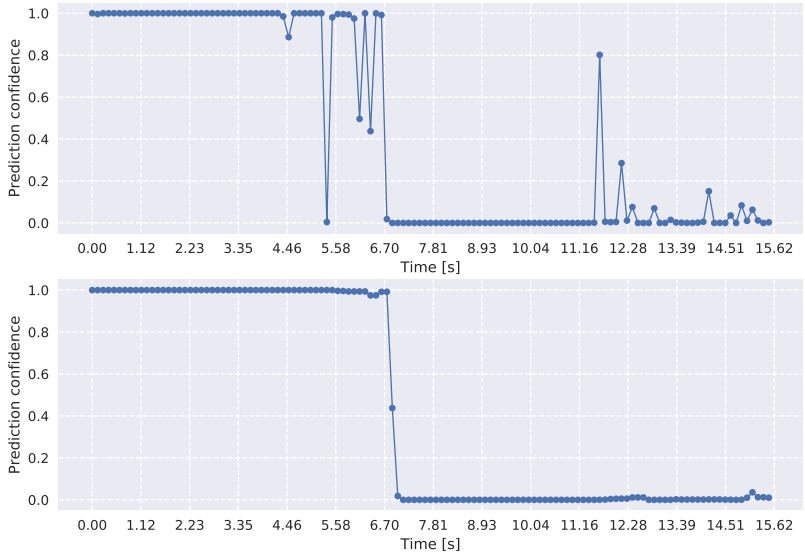

Figure 7.4: Predictions from a real-world test where the robot traversed a transition from grass to gravel.

7.2 Feature Extraction and Model Selection

Table 5.1 is in many ways a very telling one. Using the LOO cross validation strategy each data measurement was classified without using any of the samples from the measurement session the data was taken from. Six different methods of classification were evaluated; two linear and four nonlinear.

First and foremost, it is noted that each tested method performed at the very least *decently*. One may argue that the LSTM model is unstable or that using LDA produces lower accuracy predictions, but they nonetheless generated average accuracies above 97%. Considering that these are the lower-performing classifiers and the remaining ones perform even better suggests that the performed feature extraction captures, or at least maintains, most of the crucial information in the original data. The PCA-plot in Figure 4.5 served as a proof of concept to this already at an early stage, as some degree of separation can be seen in only two dimensions. Furthermore, as linear models managed to separate data decently we know that the information content is readily available without the usage of more advanced nonlinear classifiers.

A deep learning puritan might argue that no feature extraction should be performed when working with deep neural networks; a network should be able to find good data characteristics on its own. Hence, an ANN may yield a better result by omitting the feature extraction process as this unavoidably discards some information from the original data.

Since much of the information needed to distinguish grass from non-grass surfaces is in the slow time dimension a neural network needs to be exposed to multiple sweeps to extract such information. This can, if we don't allow for feature extraction, either be done through concatenation of several sweeps, or through some version of a recurrent neural network. The first option is computationally expensive, as the network needs to figure out how to compute useful features from the set of complex data matrix which certainly would require a deep neural net. The second was investigated in this report through the CNN and LSTM model from (Karim et al., 2018). While it for many surfaces did perform very well, it showed lackluster performance on others indicating poor robustness, while simultaneously being the most complex of the investigated models. It may nonetheless be of interest to investigate if robustness can be improved for this model. However, based solely on the results found in this report we argue that the reduced complexity and increased control makes the feature extraction process worthwhile for the task of distinguishing grass from non-grass surfaces.

On a different note, the final choice of classifier is debatable. The DNN classifier was chosen mainly due to its high stable LOO scores in table 5.1.

Nonetheless linear models performed well too, but didn't quite reach the accuracies of the DNN model, implying that the feature vectors are at least *approximately* linearly separable. Placing more effort on the linear models through testing more types of linear classifiers and optimizing their hyperparameters more carefully might make one of the linear models the top choice.

While linear classifiers require only a dot product for making a prediction, one should not be deterred by the seemingly high complexity of the DNN model. Once trained, making a prediction using our DNN model is only a matter of computing 3 fairly small matrix multiplications and $24 + 12 + 2 = 38$ calls to an activation function. This is done quickly even with limited hardware resources, making for a parsimonious classifier. The heavy work lies in the model training phase, which is significantly more computationally complex.

7.3 Errors and Uncertainties

For this project, a total of 42 data matrices were captured from 5 different surface types, see table 3.1. The number of matrices acquired from each surface was based on having a reasonably balanced dataset with a similar number of grass and non-grass samples. Having a balanced (or near-balanced) measurement set ensures that the classification is not biased either way; with more samples of one class the classifier will naturally tend towards the surface type it has seen more of in training. This however leaves other surfaces with less similar training data, and it is well possible that more measurements of the lower-performing soil and gravel surfaces are needed.

Another risk when collecting data is obtaining too little of it. By looking at how little improvement the data augmentation made in table 6.1, it would seem that obtaining $Q = 50,000$ sweeps per data matrix was enough. However by increasing the number of data matrices collected we increase *data diversity*, capturing a wider range of grass types.

A more technical uncertainty regards the assumption of a constant velocity over different surfaces. The robot used in this project kept a somewhat steady pace regardless of surface type, but when faced with rough terrain or steep hills the robot's velocity altered a fair bit. As both the DFT and autocovariance requires a consistency in the spacing between sample points, we cannot tolerate too large variations in velocity. Hence, to sample at regular distances the sampling could be controlled by positional feedback from the robot rather than a fixed sampling frequency.

Previously we mentioned that predictions can be made more frequent by continuously keeping track of the T most recent samples. A similar idea can

be applied to the normalization process. Even if we only use $T = 25$ sweeps for feature extraction, we are not limited to use only those for the sweep normalization. By in addition using sweeps from further back in time, a more stable mean energy estimate is achieved. The estimate could in theory be updated for each new sample.

7.4 Surface Variances

One particularly challenging aspect of adequately classifying surfaces is that the space of realistically occurring surface variations is essentially infinite. As any lawn owner can testify a lawn changes drastically over the course of its life span depending on grass height, weather conditions, tear and temperature. One would certainly need an enormous training dataset to capture all such variations. And that would still only account for one single lawn. It wouldn't take much for the classifier to enter uncharted territory, as the machine learning classifier solely bases its predictions on training feature vectors.

However, we see from our results that accurate classification is possible when acquired data is reasonably similar to what has been seen in training. A robot *can* learn to identify surfaces as being grassy or not provided that it has previously been exposed to somewhat similar surfaces.

To remedy for the dangers of using an unchecked neural network classifier one should consider using some supporting sanity-check system in conjunction with the algorithm suggested in this thesis to improve robustness. Checking that returning signals are of reasonable strengths or that the estimated distance to the ground remains unchanged are suggestions.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this work the possibilities of using millimeter-wave radar for surface classification for rough target surfaces were explored. It was found that by extracting features exploiting temporal behavior and combining this with a deep neural network classifier it is possible to perform binary classification distinguishing grass covered surfaces from four other types of surfaces commonly found adjacent to lawns with an accuracy of 98.5 % using LOO cross validation. Application of a median filter onto the classifier outputs effectively suppressed prediction outliers, providing good results on artificial and real-world test data.

The feature extraction procedure along with the neural network classification and median output filtering show promise for use in real world applications. Such applications may involve autonomous lawn mowers and autonomous vacuum cleaners.

8.2 Future Work

There are many possible areas of improvement for the classification scheme presented in this work. It would be interesting to examine which sensor angles are most useful, and to further investigate what PCR system settings are best suited for rough surface classification. In this work sets of radar sweeps were normalized as a preprocessing step due to different sensors having different gain. Through consistent calibration one could circumvent this issue and use differences in RCS to a greater extent than was possible in this work. Furthermore, collecting a larger and more diverse dataset could show if a network becomes capable of generalizing grass from non-grass surfaces when

a dataset contains a wider range of surface examples, or if this model falls short in such situations.

On the modeling side it would be interesting to attempt classification without any feature extraction at all and a very deep neural network, or further examine recurring neural network models. It would likewise be of interest to use linear classifiers with other types of extracted features. Lastly a velocity-based sampling scheme should be considered for future use ensuring equidistant sampling.

Appendices

IQ Demodulation

IQ demodulation involves the process of transforming a signal on form

$$x(t) = A(t)\sin(f_c t + \phi(t)) \tag{1}$$

to complex form without the carrier frequency f_c (Lee, 1991)

$$x_{IQ}(t) = A(t)e^{i\phi(t)}. \tag{2}$$

IQ demodulation thus aims to extract the information bearing part of $x(t)$ found in the envelope $A(t)$ and phase shift $\phi(t)$ from (1). Most classical radars follow the demodulation scheme depicted in Figure 1. The first step is to split the signal in (1) into two separate channels. In the upper channel, the signal is multiplied by a coherent sinusoid with the same carrier frequency as the received pulse. The result of this multiplication can be rewritten as

$$A(t)\sin(f_c t + \phi(t)) \cdot 2\sin(f_c t) = \tag{3}$$

$$A(t)\cos(\phi(t)) - A(t)\cos(2f_c t + \phi(t)) \tag{4}$$

yielding one baseband sinusoid and one sinusoid at the double carrier frequency.

The next step in the demodulation scheme is to low pass filter to get rid of the undesired high frequency term term in (4) leaving only $A(t)\cos(\phi(t))$. This constitutes the *In-phase* channel $I(t)$.

Similarly, in the bottom channel in Figure 1, $x(t)$ is multiplied by a signal of the same carrier frequency. However, this time, the signal is shifted 90° in phase[1]. We can rewrite this product as

$$A(t)\sin(f_c t + \phi(t)) \cdot 2\cos(f_c t) = \tag{5}$$

$$A(t)\sin(\phi(t)) + A(t)\sin(2f_c t + \phi(t)). \tag{6}$$

[1]This shift is technically a *Hilbert transform*, as the sinusoid is shifted by $\pi/2$.

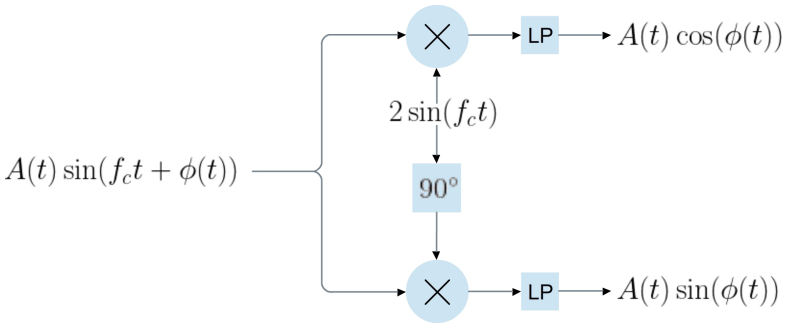

Figure 1: Schematics for a typical IQ demodulation.

After low pass filtering we obtain $A(t)\sin(\phi(t))$ which we define as the *Quadrature* channel $Q(t)$. Finally, by interpreting $I(t)$ and $Q(t)$ channels as the real- and imaginary parts of a complex number, respectively, we obtain our IQ representation as

$$I(t) + iQ(t) = A(t)\Big(\cos(\phi(t)) + i\sin(\phi(t))\Big) = A(t)e^{i\phi(t)}. \tag{7}$$

Proof of 2.15, 2.16 and 2.17

For signals $y(t)$ and $x_T(t)$ defined as

$$\begin{aligned} y(t) &= CA(t-B)\sin(\Omega(t-B)) \\ x_T(t) &= A(t)\sin(\Omega t) \end{aligned} \tag{8}$$

with angular carrier frequency $\Omega = 2\pi f_c$, where $A(t)$ is defined as

$$A(t) = \begin{cases} 1 & \text{if } 0 \leq t < L \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

the output $m(\tau)$ of the matched filtering process becomes

$$m(\tau) = \int_{-\infty}^{+\infty} y(t)x_T(t-\tau)dt \tag{10}$$

$$= C\int_{-\infty}^{+\infty} A(t-B)A(t-\tau)\sin(\Omega(t-B))\sin(\Omega(t-\tau))dt. \tag{11}$$

This integral can be split into three cases:

(i) When $|\tau - B| \geq L$, i.e. when no overlap occurrs.

(ii) When $|\tau - B| < L$ and $\tau \leq B$.

(iii) When $|\tau - B| < L$ and $\tau > B$.

The first case, (i), clearly results in the integral becoming 0. Case (ii) can be written as

$$m(\tau) = C\int_{B}^{\tau+L} \sin(\Omega(t-B))\sin(\Omega(t-\tau))dt \tag{12}$$

$$= \frac{C}{2}\int_{B}^{\tau+L} \cos(\Omega(\tau-B)) - \cos(\Omega(2t-B-\tau))dt \tag{13}$$

$$= \frac{C}{2}\Big((\tau+L-B)\cos(\Omega(B-\tau)) - \frac{1}{2\Omega}\Big[\sin(\Omega(2t-B-\tau)\Big]_{B}^{\tau+L}\Big) \tag{14}$$

$$
\begin{aligned}
&= \frac{C}{2}\Big((\tau + L - B)\cos(\Omega(B-\tau)) \\
&- \frac{1}{2\Omega}\big(\sin(\Omega(\tau + 2L - B)) - \sin(\Omega(B-\tau))\big)\Big)
\end{aligned} \tag{15}
$$

Selecting L as a whole number n of wavelengths λ, the argument of the second sinusoid can be simplified into

$$\Omega(\tau + 2L - B) = \Omega(\tau - B) + 2\frac{2\pi f n\lambda}{c} \tag{16}$$

$$= \Omega(\tau - B) + 2\frac{2\pi f nc}{cf} \tag{17}$$

$$= \Omega(\tau - B) + 4\pi n. \tag{18}$$

This means that, as the sinus function is odd and 2π periodic, (15) can be rewritten as

$$\frac{C}{2}\Big((\tau + L - B)\cos(\Omega(\tau - B)) - \frac{1}{\Omega}\sin(\Omega(\tau - B))\Big). \tag{19}$$

For (iii), we get a similar integral differing in the integral limits

$$m(\tau) = \int_{\tau}^{B+L} \sin(\Omega(t - B))\sin(\Omega(t - \tau))dt \tag{20}$$

which is calculated similarly to (ii) to yield

$$\frac{C}{2}\Big((B + L - \tau)\cos(\Omega(\tau - B)) + \frac{1}{\Omega}\sin(\Omega(\tau - B))\Big). \tag{21}$$

Thus, the full mixing output can be written as

$$m(\tau) = \begin{cases} 0 & \text{if } |\tau - B| \geq L \\ (19) & \text{if } |\tau - B| < L \text{ and } \tau \leq B \\ (21) & \text{if } |\tau - B| < L \text{ and } \tau > B. \end{cases} \tag{22}$$

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from *tensorflow.org*.

Acconeer (2018). *A111 – Pulsed Coherent Radar (PCR)*. Acconeer. v 1.3.

Amin, M. G. (2017). *Radar for indoor monitoring*. CRC Press, first edition.

Basseville, M. and Nikiforov, I. V. (1993). *Detection of abrupt changes*. Prentice Hall.

Boswell, D. (2002). Introduction to support vector machines.

Breiman, L. (2002). Random forests. *Machine Learning*, 45(1):5–9.

Brownlee, J. (2017). A gentle introduction to mini-batch gradient descent and how to configure batch size. *https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size*. Retrieved 14/1/18.

Bystrov, A., Hoare, E., Tran, T.-Y., Clarke, N., Gashinova, M., and Cherniakov, M. (2016). Road surface classification using automotive ultrasonic sensor. *Procedia Engineering*, 168:19–22.

Capobianco, S., Facheris, L., Cuccoli, F., and Marinai, S. (2018). Vehicle classification based on convolutional networks applied to fmcw radar signals. *Advances in Intelligent Systems and Computing*, pages 115–128.

Chollet, F. (2015). Keras. Software available from *https://github.com/fchollet/keras*.

Do, M. and Vetterli, M. (2002). Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance. *IEEE Transactions on Image Processing*, 11(2):146–158.

Finkele, R. (1997). Detection of ice layers on road surfaces using a polarimetric millimetre wave sensor at 76 ghz. *Electronics Letters*, 33(13):1153–1154.

Finkele, R., Schreck, A., and Wanielik, G. (1995). *Polarimetric road condition classification and data visualisation*, pages 1–5. IEEE.

Frenzel, L. (2018). Millimeter-wave single-chip radar sensors get practical. *https://www.electronicdesign.com/automotive/millimeter-wave-single-chip-radar-sensors-get-practical*. Retrieved 14/1/18.

Fung, A., Li, Z., and Chen, K. (1992). Backscattering from a randomly rough dielectric surface. *IEEE Transactions on Geoscience and Remote Sensing*, 30(2):356–369.

Fung, A. K. and Pan, G. W. (1987). A scattering model for perfectly conducting random surfaces. *International Journal of Remote Sensing*, 8(11):1579–1593.

Gentile, R. and Donovan, M. (2018). Building and processing a radar data cube. *https://www.mathworks.com/company/newsletters/articles/building-and-processing-a-radar-data-cube.html*. Retrived 14/1/18.

Giguere, P. and Dudek, G. (2011). A simple tactile probe for surface identification by mobile robots. *IEEE Transactions on Robotics*, 27(3):534–544.

Graves, A., Mohamed, A.-R., and Hinton, G. (2013). *Speech recognition with deep recurrent neural networks*, pages 1–5. IEEE.

Griffiths, D. J. (2018). *Introduction to electrodynamics*. Cambridge University Press.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., and Sainath, T. e. a. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1–9.

Hyvasrinen, A., Karhunen, J., and Oja, E. (2004). *Independent Component Analysis*. John Wiley and Sons.

Häkli, J., Säily, J., Koivisto, P., Huhtinen, I., Dufva, T., Rautiainen, A., Toivanen, H., and Nummila, K. (2013). *Road surface condition detection using 24 GHz automotive radar technology*, pages 702–707. IEEE.

Jakobsson, A. (2015). *An Introduction to Time Series Modeling*. Studentlitteratur AB, second edition.

Jithesh, V., Sagayaraj, M. J., and Srinivasa, K. G. (2017). Lstm recurrent neural networks for high resolution range profile based radar target classification. *2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*.

Karim, F., Majumdar, S., Darabi, H., and Chen, S. (2018). Lstm fully convolutional networks for time series classification. *IEEE Access*, 6:1662–1669.

Kees, N. and Detlefsen, J. (1994). *Road surface classification by using a polarimetric coherent radar module at millimeter waves*, pages 1–4. IEEE.

Klemm, M., Gibbins, D., Leendertz, J., Horseman, T., Preece, A. W., Benjamin, R., and Craddock, I. J. (2011). *Development and Testing of a 60-Element UWB Conformal Array for Breast Cancer Imaging*, pages 3077–3079. IEEE.

Knott, E. F. (1993). *Radar cross section measurements*. Springer US, first edition.

Kohl, N. (2010). Role of bias in neural networks. *https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks*. Retrieved on 14/1/18.

Kriesel, D. (2007). A brief introduction to neural networks. *http://www.dkriesel.com*. Retrieved on 14/1/18.

Kuo, H.-C., Lin, C.-C., Yu, C.-H., Lo, P.-H., Lyu, J.-Y., Chou, C.-C., and Chuang, H.-R. (2016). A fully integrated 60-ghz cmos direct-conversion doppler radar rf sensor with clutter canceller for single-antenna noncontact human vital-signs detection. *IEEE Transactions on Microwave Theory and Techniques*, 64(4):1018–1028.

Lee, J. P. (1991). *I/Q Demodulation of Radar Signals with Calibration and Filtering*.

Lemley, J., Bazrafkan, S., and Corcoran, P. (2017). Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869.

Lien, J., Gillian, N., Karagozler, M. E., Amihood, P., Schwesig, C., Olson, E., Raja, H., and Poupyrev, I. (2016). Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics*, 35(4):1–19.

Lindgren, G., Rootzen, H., and Sandsten, M. (2014). *Stationary stochastic processes for scientists and engineers*. CRC Press Taylor & Francis Group.

Liu, L., Chen, J., Fieguth, P., Zhao, G., Chellappa, R., and Pietikäinen, M. (2018). From bow to cnn: Two decades of texture representation for texture classification. *International Journal of Computer Vision*.

Logan, S. (2017). Understanding the structure of neural networks. *https://becominghuman.ai/understanding-the-structure-of-neural-networks-1fa5bd17fef0*. Retrieved on 14/1/18.

Mckerrow, P. and Kristiansen, B. (2006). Classifying surface roughness with ctfm ultrasonic sensing. *IEEE Sensors Journal*, 6(5):1267–1279.

McSweeney, K. (2018). Lowe's introduces autonomous service robots to retail stores — zdnet. *https://www.zdnet.com/article/lowes-introduces-autonomous-retail-service-robots/*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. pages 1310–1318.

Pearson, R. (2002). Outliers in process modeling and identification. *IEEE Transactions on Control Systems Technology*, 10(1):55–63.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Prechelt, L. (1998). Early stopping - but when? *Lecture Notes in Computer Science*, pages 55–69.

Proakis, J. G. and Manolakis, D. G. (2014). *Digital signal processing*. Prentice Hall, third edition.

Raschka, S. (2014). Linear discriminant analysis - bit by bit. *https://sebastianraschka.com/Articles/2014_python_lda.html*. Retreived on 14/1/18.

Raschka, S. (2016). Model evaluation, model selection, and algorithm selection in machine learning. *https://sebastianraschka.com/blog/2016/model-evaluation-selection-part1.html*. Retrieved on 14/1/18.

Richards, M. A. (2014). *Fundamentals of radar signal processing*. McGraw-Hill Education.

Ridenour, L. N. (1947). *Radar System Engineering*. McGraw-Hill, first edition.

Rojas, R. (1996). *Neural networks*. Springer.

Sanfacon, N. (2017). Robots' disruptive force on the vacuum cleaner industry. *https://www.gapintelligence.com/blog/2017/robots-disruptive-force-on-the-vacuum-cleaner-industry*. Retrieved on 14/1/18.

Santos, L. A. (2018). Linear classification · artificial inteligence. *https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/linear_classification.html*. Retreived on 14/1/18.

Scharf, P. A., Iberle, J., Mantz, H., Walter, T., and Waldschrnidt, C. (2018). Multiband microwave sensing for surface roughness classification. *2018 IEEE/MTT-S International Microwave Symposium - IMS*, pages 934–937.

Schmidt, S. (1988). Evidence for a spectral basis of texture perception in bat sonar. *Nature*, 331(6157):617–619.

Shalev-Shwartz, S. and Ben-David, S. (2016). *Understanding machine learning*. Cambridge University Press.

Skolnik, M. I. (2009). *Radar handbook*. McGraw-Hill.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2018). *Going deeper with convolutions*, pages 1–9. IEEE.

Udelhofen, G. (2018). Robotic lawn mower market continues to grow. *https://www.greenindustrypros.com/lawn-maintenance/mowing/article/12414137/robotic-lawn-mower-market-continues-to-grow*. Retreived on 14/1/18.

Viikari, V., Varpula, T., and Kantanen, M. (2008). *Automotive Radar Technology for Detecting Road Conditions. Backscattering Properties of Dry, Wet, and Icy Asphalt*, pages 276–279. IEEE.

Watson-Watt, R. (1945). Radar in war and in peace. *Nature*, 156(3959):319–324.

Yin, L., Yang, R., Gabbouj, M., and Neuvo, Y. (1996). Weighted median filters: a tutorial. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(3):157–192.