Ghost target classification using scene models in radar

David Wadmark, Anton Sedin

Master's thesis
2021:E41

Lund Institute of Technology
Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Master's Theses in Mathematical Sciences 2021:E41
ISSN 1404-6342

LUTFMS-3420-2021

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/

Abstract

In surveillance contexts, radars can be used to monitor an area, detecting and tracking moving objects inside it. Monitored areas in urban environments often contain many surfaces that reflect radar waves, which can have the undesired consequence of a single object producing multiple tracks due to multipath propagation effects. This thesis considers a method of identifying if a track is produced by a real object, or if it stems from multipath effects. The proposed method works by creating a machine-learning-based classifier and modelling the monitored scene over time. Tracks are assigned features based on their characteristics and the state of the scene model in regards to their position. These features are then used as inputs to the classifier model to produce the classification. We propose four machine learning-based classifier models, with two different sets of structures and features used. The classifier models are compared to a naive classifier model for reference.

The proposed models all outperform the naive classifier, although some of them are biased. As for the usefulness of the scene model, the results are mixed but show promise. We believe that the scene model can improve classification performance further with more and better data.

Keywords

radar surveillance, multipath, machine learning, classification

Acknowledgements

We acknowledge the fantastic support and council that was provided by our industry supervisors Dr. Sebastian Heunisch and Dr. Aras Papadelis, as well as our unofficial industry supervisors Dr. Stefan Adalbjörnsson, Dr. Anders Mannesson and Daniel Ståhl. We want to thank them for all the great discussions, remarks and advice, and for providing superb feedback on the report with short notice.

We want to thank our supervisors at Lund University, Prof. Andreas Jakobsson and Dr. Björn Olofsson for providing great advice and feedback on the report. We also want to thank our examiners Prof. Rolf Johansson and Senior Lecturer Johan Lindström for critiquing and grading this thesis.

We want to thank André Nüßlein for proofreading our report, which led to its readability being greatly improved.

Finally, we want to give a heartfelt thank you to our families for their unconditional support.

Abbreviations

FMCW - Frequency Modulated Continuous Wave
IF - Intermediate Frequency
ANN - Artificial Neural Network
RCS - Radar Cross Section
SNR - Signal-to-Noise Ratio
FFT - Fast Fourier Transform
DFT - Discrete Fourier Transform
LOGOCV - Leave-One-Group-Out Cross-Validation

Contents

1

Introduction

1.1 Background

Short range radar sensors have become a field of interest in the surveillance industry in recent years. Unlike video surveillance systems, radar-based surveillance systems can operate independent of lighting conditions and they do not suffer from the same large decrease in performance in foggy, rainy and snowy conditions. Objects are detected by transmitting an electromagnetic signal from the radar, and collecting the reflections from the surroundings. By using certain transmit patterns and combining multiple receiver antennas in an array, the range, radial velocity and angle of detected objects can be deduced. However, there is no direct way to tell if a signal comes from a direct reflection from the object or if the signal has been reflected on other surfaces. This is a big problem in radar surveillance systems that are mounted in urban environments with many reflecting surfaces like walls, containers or cars.

1.2 The multipath problem

The problems caused by ambiguous reflections of radar signals can be illustrated by Figure 1.1. In this fictive scenario, the radar is able to trace the target over time by measuring the position and the velocity of the target. We will refer to this trace as a track. Radar signals can also be reflected in other surfaces like the wall in this case, either before or after hitting the target. We will refer to this as the multipath phenomenon. This can cause additional tracks to appear although there is only one target in view of the radar. These ghost tracks can cause false alarms, for instance if they appear in a zone where no people are allowed. In any real installation, the environment is typically more complex than in Figure 1.1 where there is only one perfectly flat wall. This means that radar signals can often travel in an exceedingly large number of ways from the radar to a target and back. Therefore it is not an easy task to construct an accurate geometrical model of the reflections.

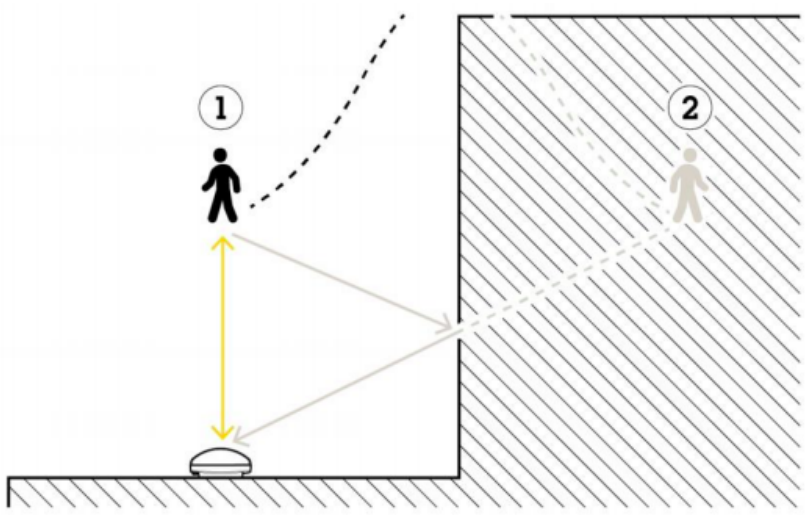

Figure 1.1 Illustration of the multipath phenomenon. The radar signals travel along two different paths back to the radar after reflecting of the target causing two tracks to appear from one target.

1.3 Purpose

This thesis aims to investigate the possibility of using radar data to reliably classify whether or not a detected target is in fact real, or a result of the multipath phenomenon. More specifically, we will try to solve this problem using machine learning methods, and investigate whether the classification can be aided by constructing a model of the monitored scene over time.

Scope

We make no attempt to distinguish between the type of objects such as pedestrians, cars and bikes in this thesis, only whether the detected target is a ghost target or a real target. All objects are treated the same, no matter what type of object it is. We also assume that the radar is stationary. The investigation will be done using a single radar sensor with proprietary signal processing and tracking software provided by our industry supervisors. This thesis will not consider any changes to these systems.

1.4 Previous work

Previous attempts have been made to classify ghost targets from radar signals, but predominantly in automotive settings where the radar is mounted on a vehicle, and is therefore not stationary [Ryu et al., 2018], [Prophet et al., 2019], [Kraus et al., 2020]. Attempts have also been made to infer the position of walls using a stationary radar by making some assumptions about multipath signals [Nüßlein, 2021].

2

Theory

An introduction to radar signal processing and some machine-learning concepts will be presented in this chapter for readers unfamiliar with these concepts. The first sections will cover the fundamentals of radar technology and radar signal processing. Then, the tracker used in this thesis and the multipath problem are described in more detail. Finally, some concepts from machine learning that are used in this thesis are presented.

2.1 Radar fundamentals

Radar is an acronym for Radio Detection and Ranging. The technology was originally developed to detect and locate aircraft in World War II [Sim, 2014] . Nowadays, radar technology is used in a wide range of applications such as flight control systems, radar astronomy and meteorological precipitation monitoring.

Radar systems use radio waves to determine the range, angle or velocity of objects. They are composed of a transmitter and receiver. The transmitter emits radio waves which in turn are reflected or scattered by objects, and the receiver captures the reflected or scattered waves. There are many types of radars which use different transmitting schemes, but in this thesis exclusively Frequency Modulated Continuous Wave (FMCW) radars were used, which will be covered in the next section.

2.2 FMCW radar

Range measurement

To measure the range from the radar to an object, an FMCW radar transmits a signal called a chirp [Smith, 1997]. A chirp is a sinusoid where the frequency increases linearly with time. The frequency and amplitude of a chirp as a function of time can be seen in Figure 2.1.

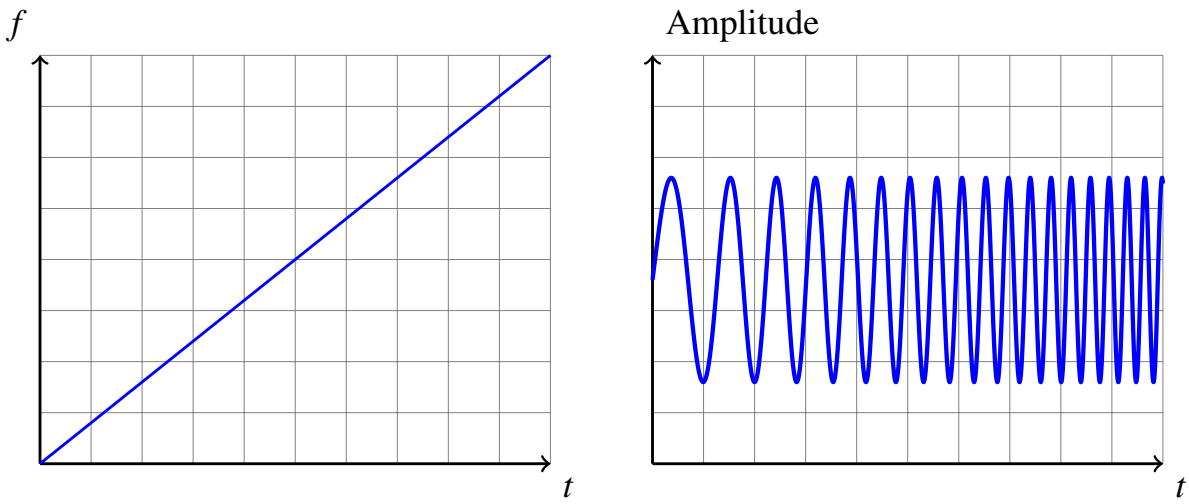

Figure 2.1 Illustration of a chirp. Left: frequency as a function of time. Right: amplitude as a function of time.

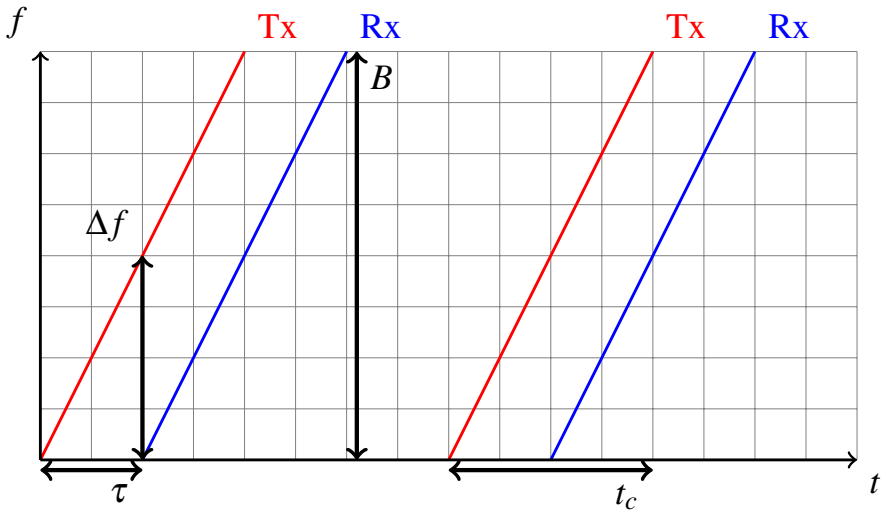

Figure 2.2 Transmitted signal (Tx) and received signal (Rx).

If there is an object within range of the radar, the chirp will be reflected from the object and a time-delayed chirp will be received by the radar. This transmit-receive pattern is illustrated in Figure 2.2.

The received signal is then mixed with the transmitted signal to a so-called intermediate frequency (IF) signal. This is a sinusoid where the frequency and phase is the difference of the frequencies and phases of the received and transmitted signal:

$$x_{IF}(t) = \sin\left[(\omega_2 - \omega_1)t + (\phi_2 - \phi_1)\right] \tag{2.1}$$

where ω_1 and ω_2 are the frequencies of the transmitted and received signal, and ϕ_1 and ϕ_2 are the phases of the transmitted and received signal.

The IF signal will have frequency

$$f_0 = \omega_2 - \omega_1 = S\tau = \frac{2Sd}{c} \tag{2.2}$$

which is equivalent to Δf in Figure 2.2. Here, S is the rate of change of the chirp frequency, d is the distance to the interfering object, and c is the speed of light. From equation (2.2) the distance from the radar to an interfering object can be deduced. We will refer to this distance as the range of an object.

Doppler

To deduce the radial velocity of an object, multiple chirps are transmitted in rapid succession. With the assumption that the object is traveling much slower than the speed of light, the resulting IF signals of each chirp will have approximately the same frequencies, but their phases ($\phi_2 - \phi_1$ in equation (2.1)) will be different for non-zero radial velocities. This is because the distance d to the object changes between chirps due to the movement of the object. This change in distance Δd is small, but significant in relation to the wavelength λ of the chirp. The relation between phase difference $\Delta\phi$ and difference in distance Δd is

$$\Delta\phi = \frac{4\pi\Delta d}{\lambda} \,. \tag{2.3}$$

By transmitting chirps separated by a time of t_c and inserting the relation $\Delta d = vt_c$ in equation (2.3) we have

$$\Delta\phi = \frac{4\pi v t_c}{\lambda} \implies v = \frac{\lambda\Delta\phi}{4\pi t_c} \tag{2.4}$$

where v is the radial velocity of an object. The signals received by transmitting a set of chirps will be referred to as a frame from here on.

Azimuth

To determine the angle of arrival θ of an object, several equally spaced receiver antennas are used. The incoming signals are assumed to be plane waves. The scenario is illustrated in Figure 2.3, where the incoming signals to each of the receiver antennas (black rectangles) are represented by the red arrows.

For non-zero angles, the received signals at each of the antennas will have a difference in phase which we call ω. From this phase difference, the angle of arrival of the received signal can be deduced using the relation

$$\omega = \frac{2\pi d \sin\theta}{\lambda} \implies \theta = \arcsin\left(\frac{\lambda\omega}{2\pi d}\right) \,. \tag{2.5}$$

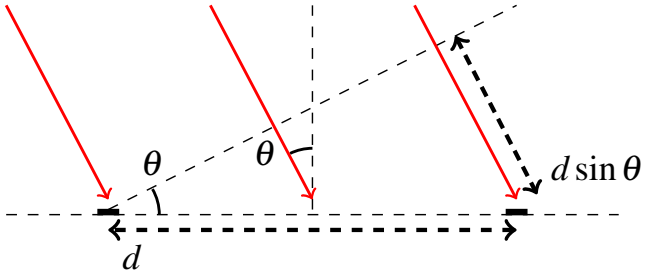

Figure 2.3 Incoming signal (red arrows) from an object at an angle θ to some receiver antennas (black rectangles).

Fourier transforms

When several targets appear in front of the radar the transmitted chirps are reflected multiple times, leading to the IF signal becoming a sum of sinusoids. To identify the individual sinusoids, a Fourier transform is applied to convert this time-domain signal into the frequency domain. From here the independent sinusoids that make up the IF signal can be deduced by looking at where the peaks of the transformed signals are. Since the converted signal is complex, each value containing both an amplitude and a phase, we can also get the initial phase of each sinusoid by looking at the phase of its peak in the frequency domain.

Doing this in practice means using a fast Fourier transform (FFT) [Duhamel and Vetterli, 1990], which is a computationally efficient algorithm that computes the discrete Fourier transform (DFT) of a sequence. When an IF signal is generated from a receiving antenna, it is transformed using an FFT and the results are stored in a vector. This is repeated for each chirp in a frame, the resulting vectors being stored as rows in a matrix. For each range bin another FFT is applied, creating a range-velocity plot. These steps are repeated for every receiver antenna, creating a 3D-array where a final FFT can be applied for each range-velocity bin. Thus, the range, radial velocity, and azimuth angle of individual objects can be detected in a scene with multiple reflections. These steps can be seen in Figure 2.4. For further information, we refer to [Rao, 2017].

Signal strength

There are two measures commonly used in radar sensors for describing the strength of the reflected signal. The radar cross-section (RCS) denoted by σ, is a measurement of how well an object reflects radio waves. It interprets the object as if it were a perfectly reflecting sphere, the cross-sectional area of which is the dimension of the measure. While RCS formally describes the detected object rather than the strength of the recieved signal, the two are correlated as can be shown by the radar range equation [Richards, 2005]:

$$P_r = \frac{P_t G_t \sigma A_e}{(4\pi)^2 r^4} \implies \sigma = \frac{(4\pi)^2 r^4 P_r}{P_t G_t A_e} \tag{2.6}$$

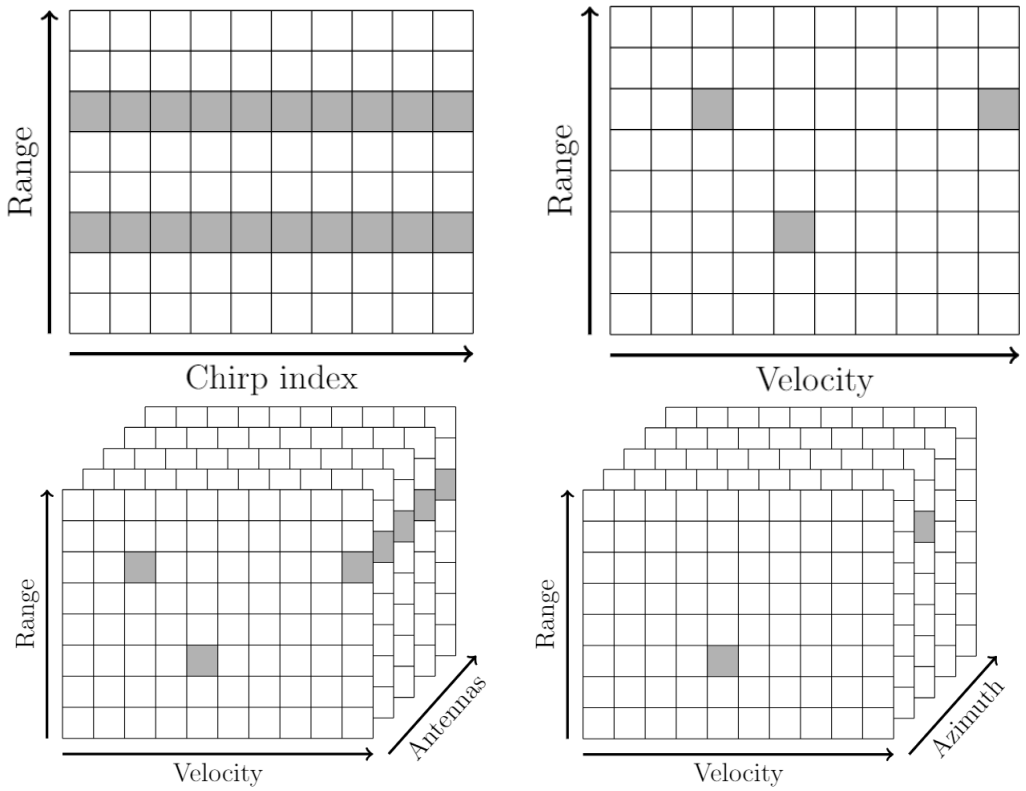

Figure 2.4 Signal processing steps. (top left) range-chirp index matrix, (top right) range-velocity matrix, (bottom left) range-velocity-antenna cube, and (bottom right) range-velocity-azimuth cube.

where P_t and P_r are, respectively, the power to the transmitting antenna and the received power. A_e is the effective area of the receiving antenna, a measure of its geometric area and efficiency. Finally, G_t is the gain of the transmitting antenna, and r is the distance from the radar to the object.

Another, more straight-forward way to describe the strength of the reflected signal is the signal-to-noise ratio (SNR). For a bin i, in a set of bins I, we define this ratio as:

$$\mathrm{SNR}_i = \frac{P_{r,i}}{\tilde{P}_{r,I}} \tag{2.7}$$

where $P_{r,i}$ is the received power in a specific bin and $\tilde{P}_{r,I}$ is the mean noise power estimated here as the median of the received power in all bins. Note that SNR does not take the distance to the detected object into account, meaning a small object close to the radar could have a higher SNR than a large object further away.

2.3 Tracker

The purpose of the tracker is to follow the position of a target over multiple frames. The high velocity resolution of millimeter wave radars means that an object like a car, person, or even a small animal will give off multiple detections. This is not only

due to the spatial extent of the object, but also due to different parts of the detected object having different radial velocities, for example the swaying limbs of a person. This phenomenon is known as a micro-Doppler effect [Chen et al., 2006]. After the Fourier transforms, each frame contains a point cloud of detections, with each point containing polar coordinates, radial velocity, RCS and SNR values. By grouping these detections into clusters and connecting them over time, tracks are created. In this thesis, this was done by proprietary tracking software, which we were given access to by our industry supervisors. We will refer to this software as the tracker.

Clustering and association

The first step in the tracking process is to determine which detection points stem from the same object, and group them together. This is called clustering and is done using a process that takes the position and RCS values of the detections into account. Next these clusters are given possible associations to existing tracks in the previous frame based on their velocities and positions. Finally, out of these possible associations, the best combination of tracks and clusters are chosen based on minimizing the distances between them.

Updating tracks

Updating the existing tracks is based on a version of the well-known Kalman filter [Kalman, 1960]. If a track has an associated cluster, the new position will be a weighted average of its predicted position and the position of the cluster. In cases where a track has no association to a cluster, the tracker will perform dead-reckoning to predict the current position of the object. If the track is not associated with a new cluster frequently over a certain amount of frames, it will no longer be considered alive, and subsequently be removed. Any remaining clusters that have not been associated to any existing track are used to create new tracks.

2.4 Multipath fundamentals

Given a radar sensor with an object visible to it, the direct path is the straight line from the radar to the object. The multipath phenomenon is the result of the radar signal not traveling along this direct path, instead taking an indirect path after being reflected by one or more other surfaces in the scene. The signal can take an indirect path before and/or after being reflected off the object, which will alter the position of the ghost object. Since the perceived range to the ghost target is equal to the distance the signal has traveled, and the direct path is the shortest, a ghost object will always be further away than the real object that created it. Therefore, given a frame with multiple tracks, it can be assumed that the one closest to the radar is a real track. For a more detailed description of the geometrical aspect of the multipath phenomenon we refer to [Nüßlein, 2021].

As an electromagnetic signal always loses power from a reflection on most materials, there is a limited number of reflections that can occur before the signal can no longer be distinguished from noise by the radar. It also means that the RCS of a real track will always be larger than the RCS of any ghost tracks that have been generated from that same object. Because of this it can also be assumed that the track with the largest RCS value in a scene is a real track. It is worth noting that the assumptions outlined in this section are not always true, as an object which is not visible by the radar can produce reflections in surfaces that are visible to the radar.

2.5 Machine learning methods

Decision tree classifiers

Given a set of samples described by feature vectors $\mathbf{x}$ and a set of labels y which describes the categorical value of the feature vectors, a decision tree classifier [James et al., 2017] can be used to predict the label y for a feature vector $\mathbf{x}$. When fitting a decision tree to a dataset (a set of feature vectors and corresponding labels), the decision tree recursively splits the dataset into two sets by examining the value of some feature, where the split minimizes or maximizes some metric. In this thesis, the metric used was the Gini impurity [Laber and Murtinho, 2019]. With a set of samples having J different possible labels, and letting p_i denote the proportion of samples labeled with class i, the Gini impurity for this set is calculated as

$$I_{\mathrm{G}} = \sum_{i=1}^{J} \left(p_i \sum_{k \neq i} p_k \right) . \tag{2.8}$$

In a binary classification problem such as the one in this thesis, the Gini impurity can be simplified to

$$I_{\mathrm{G}} = p_1 p_2 + p_2 p_1 = 2 p_1 p_2 = 2 p_1 (1 - p_1) . \tag{2.9}$$

An optimal split which minimizes the Gini impurity is then a split where one set only contains samples of class 1 and the other set only contains samples of class 2. An example of a decision tree fit to the dataset used in this thesis can be seen in Figure 2.5. Two features, "track lifetime" and "closest" which are explained further in Table 5.1 were used to create the tree, and the depth was limited to 2.

Random forest classifiers

A random forest is an ensemble classifier which combines the result of several sub-classifiers by majority voting, where the most commonly predicted label by the sub-classifiers is chosen as the final prediction [James et al., 2017]. The sub-classifiers in a random forest are decision trees, where in the case of this thesis, the correlation between the sub-classifiers was reduced by using bagging (Bootstrap Aggregating)

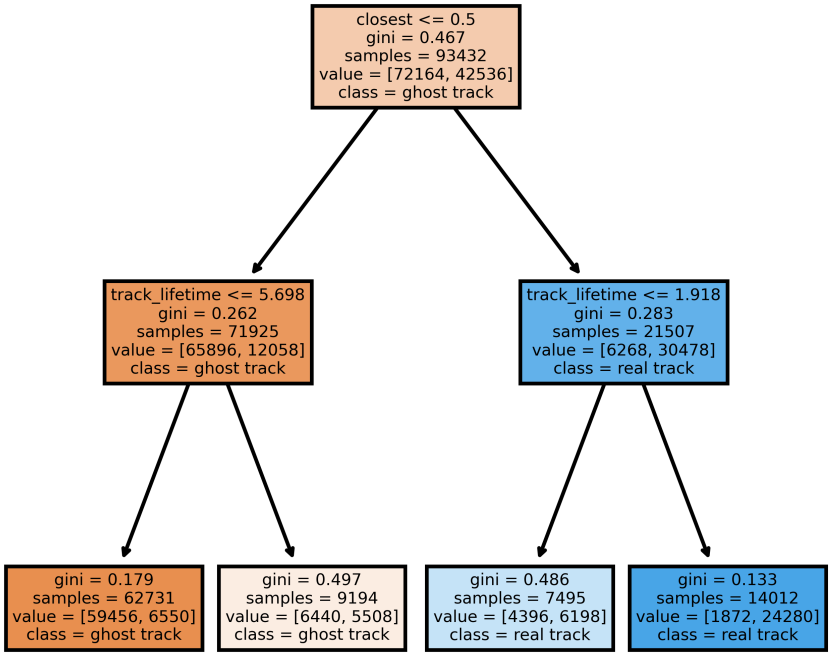

Figure 2.5 Example of a very simple decision tree fit to the dataset used in this thesis. The tree has a depth of 2 and only 2 features were used to create the tree.

and boosting (or feature bagging) [James et al., 2017]. Bagging is the practice of sampling, with replacement, a subset of the dataset used for training, and then using that subset to fit the model instead of the original training set. Boosting is the practice of restricting the features considered when making a split in a decision tree to a random subset of the features. In this thesis, the number of features considered in the random forests used were $\sqrt{n_f}$ rounded to the nearest integer, where n_f is the number of features in the feature vectors $\mathbf{x}$. This is a common choice for classification tasks [Hastie et al., 2008].

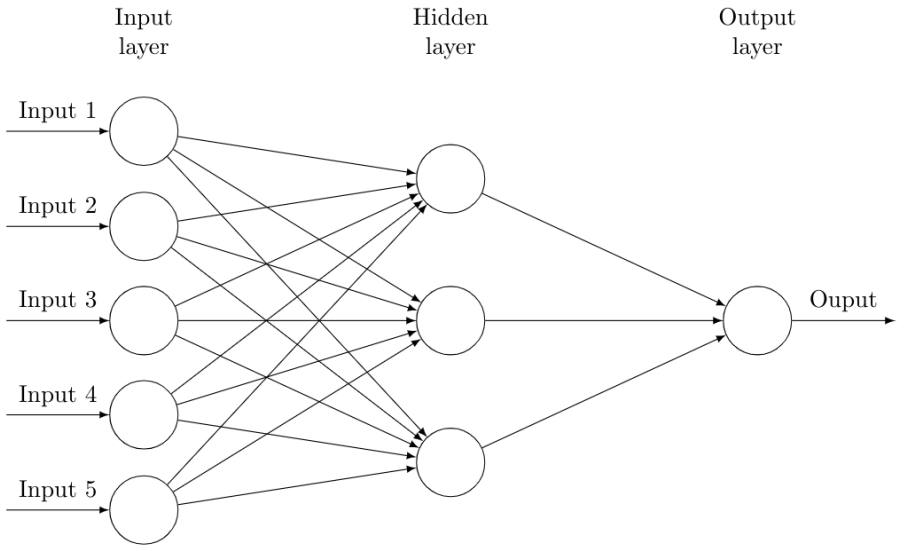

Figure 2.6 Schematic of a simple artificial neural network.

Artificial neural networks

An artificial neuron is a function which maps an input vector $\mathbf{x}$ to an output scalar y according to

$$y(x) = \phi(\mathbf{w}^{\mathrm{T}}\mathbf{x}) = \phi\left(\sum_{j=0}^{m} w_j x_j\right) \tag{2.10}$$

where $\mathbf{w}$ is a vector of weights with dimension $m \times 1$ and ϕ is an activation function (see next section). Mapping an input vector $\mathbf{x}$ to several neurons will produce an output vector $\mathbf{y}$ with an element y_k for every neuron:

$$y_k(x) = \phi(\mathbf{w}_k^{\mathrm{T}}\mathbf{x}) = \phi\left(\sum_{j=0}^{m} w_{kj} x_j\right) . \tag{2.11}$$

The result of combining several artificial neurons like this is commonly known as an artificial neural network (ANN) [Gurney, 1997]. An ANN is usually divided into layers where the output vector $\mathbf{y}$ from one layer is used as input to the next. The structure of an example ANN is visualized in Figure 2.6. This particular network accepts 5 input values and produces one output value.

Activation functions

The main purpose of using activation functions in artificial neural networks is to add non-linearity to the network. Without activation functions, artificial neural networks are nothing more than linear transformations of the input. Many activation functions also keep the outputs of neurons from growing in an unbounded fashion. This can cause ANNs to become unstable and can cause computational problems.

Three main activation functions were used in this thesis; ReLU, sigmoid and softmax [Sharma, 2017]. ReLU is short for Rectified Linear Unit and is defined as

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{2.12}$$

In reality, sigmoid functions refer to a class of functions but in the context of machine learning it typically refers to the logistic function:

$$\text{Sigmoid}(x) = \frac{1}{1+\exp(-x)} \tag{2.13}$$

The softmax function is a generalization of the logistic function beyond the one-dimensional case. Consider a vector $\mathbf{x}$ consisting of K real numbers. The softmax function produces a K-dimensional output where the output at index i is defined as:

$$\text{Softmax}(\mathbf{x}_i) = \frac{\exp(x_i)}{\sum_j^K \exp(x_j)} . \tag{2.14}$$

Supervised learning

With several input vectors $\mathbf{x}_k$ and corresponding targets y_k it is possible to adjust the weights in an ANN to estimate a function which maps $\mathbf{x}_k$ to y_k. This process is called supervised learning. For this process two things are necessary; a loss function and an optimization algorithm. The loss function computes a measure of how poor the network output $\hat{y}_k = f(\mathbf{x}_k)$ is in relation to the label y_k. A commonly used loss function in classification contexts is the cross entropy loss function [Brownlee, 2020].

Consider a classification task where c is the target class index, and $\hat{\mathbf{y}}$ is the output from the network, where each element $\hat{y}_c$ in $\hat{\mathbf{y}}$ represents the score of class c. This score is first approximated as a probability with the softmax function from equation (2.14), and then the negative logarithm of the likelihood of the data is calculated:

$$\text{L}(\hat{\mathbf{y}}, c) = -w_c \log\left(\frac{\exp(\hat{y}_c)}{\sum_j \exp(\hat{y}_j)}\right) . \tag{2.15}$$

Here w_c represents the weight given to class c.

The backpropagation algorithm described in more detail in [Goodfellow et al., 2016] can be used to calculate the gradients of the loss function with respect to the weight of every neuron by recursively applying the chain rule of derivatives. When the gradients are known, the weights of the network can be iteratively updated towards a minimum of the loss function. In this thesis we exclusively used the network optimization algorithm known as Adam, which is described further in [Kingma and Ba, 2014].

Dropout

To avoid overfitting the neural networks to the training sets, a regularization technique known as dropout [Srivastava et al., 2014] was used. In the implementation of dropout that was used in this thesis, random elements of the input vector $\mathbf{x}$ are set to zero after multiplication with the weights of each layer except for the final layer. Each element is set to zero according to a Bernoulli distribution $Z_i^k \in \mathrm{B}(p)$ where k is the index of the layer and i is the index of the element in the vector. All Z_i^k are independent. The value of p used in this thesis was $p = 0.3$.

2.6 Scene mapping

In order to build a model of the scene visible to the radar, a map needs to be created. [Thrun et al., 2006] defines a map m as a list of objects in a scene, along with their properties:

$$m = \{m_1, m_2, ..., m_N\} \tag{2.16}$$

With N objects in the scene, each m_n with $1 \leq n \leq N$ specifies a property. Depending on the type of map, the index n either specifies a certain feature, or a specific location in the scene.

Occupancy grid maps

Occupancy grid maps are a probabilistic approach to mapping that is popular in the field of robotics [Siciliano and Khatib, 2007]. The scene is simplified into a discrete, two-dimensional grid of independent cells, where m_i denotes the grid cell with index i. We define the map as the union of all grid cells:

$$m = \bigcup_i m_i \tag{2.17}$$

and each cell can be given its own posterior probability of being occupied by an object:

$$p(m_i|z_{1:t}, x_{1:t}) = p(m_i|z_{1:t}) \tag{2.18}$$

where $z_{1:t}$ is the set of all sensor measurements, and $x_{1:t}$ is the path of the robot [Thrun et al., 2006]. The equality in equation (2.18) follows from the fact that the radar is static and its position is considered known in our case. Expanding on this expression for the occupancy probability we get

$$p(m_i|z_{1:t}) = \frac{p(z_t|m_i)p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{2.19}$$

and applying Bayes rule to the measurement model $p(z_t|m_i)$ we get

$$p(m_i|z_{1:t}) = \frac{p(m_i|z_t)p(z_t)p(m_i|z_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1})} \,. \tag{2.20}$$

By calculating the odds of occupancy instead of the probability of occupancy we can cancel probabilities that are difficult to calculate like $p(z_t)$ and $p(z_t|z_{1:t-1})$:

$$\begin{aligned}\frac{p(m_i|z_{1:t})}{p(\neg m_i|z_{1:t})} &= \frac{p(m_i|z_t)}{p(\neg m_i|z_t)}\frac{p(m_i|z_{1:t-1})}{p(\neg m_i|z_{1:t-1})}\frac{p(\neg m_i)}{p(m_i)} \\ &= \frac{p(m_i|z_t)}{1-p(m_i|z_t)}\frac{p(m_i|z_{1:t-1})}{1-p(m_i|z_{1:t-1})}\frac{1-p(m_i)}{p(m_i)}\end{aligned} \tag{2.21}$$

By taking the logarithm of the odds of occupancy from equation (2.21) and denoting it $l_t(m_i)$ we arrive at an additive and thus more stable representation of occupancy in grid cell $\mathbf{m}_i$ at time t:

$$\begin{aligned}l_t(m_i) &= \log\frac{p(m_i|z_t)}{1-p(m_i|z_t)} + \log\frac{p(m_i|z_{1:t-1})}{1-p(m_i|z_{1:t-1})} - \log\frac{p(m_i)}{1-p(m_i)} \\ &= \log\frac{p(m_i|z_t)}{1-p(m_i|z_t)} + l_{t-1}(m_i) - l_0(m_i)\ .\end{aligned} \tag{2.22}$$

The expression $l_{t-1}(m_i)$ represents the log-odds of occupancy of grid cell m_i at time $t-1$ and l_0 represent the prior log-odds of occupancy of the same grid cell. Thus, all we need to construct the map is the prior occupancy probability and a way to calculate $p(m_i|z_t)$.

This can be done with a so-called inverse sensor model [Thrun et al., 2006]. Given a measurement, this sensor model updates the map according to a probability distribution based on the accuracy and characteristics of the sensor. One way to do this is by raising the occupancy odds in the cells around the measurement point, and also lowering the occupancy odds in the cells between the measurement point and the position of the sensor. It is worth noting that the inverse sensor model assumes that there is no correlation between the occupancy of a cell and the occupancy of its neighboring cells.

2.7 Evaluation metrics

Classification

To evaluate the classification results, precision, recall and F_1 score were used. These metrics are calculated for each class in the classification problem, and can be averaged over all classes to get a combined evaluation for all classes. Denoting TP as true positives, FP as false positives and FN as false negatives, we can state the formulas for calculating precision and recall:

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{2.23}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}\ . \tag{2.24}$$

Precision can be seen as a measure of how many of the selected items that are relevant, and recall as a measure of how many of the relevant items that are selected. F_1-score (or traditional F-measure) is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \,. \tag{2.25}$$

This measure computes an evaluation of a classification result as a single number, which is practical. However, it weighs precision and recall equally irregardless of their consequences for the specific problem and should thus be used with caution.

3

Data

A common sentiment in the area of machine learning is that a model can only be as good as the data it is trained with. This is of course true, as the model can only be trained to make predictions about samples that are already in the dataset used and there are no guarantees for the quality of predictions on data from new scenarios and contexts. Another aspect is that the dataset needs to contain enough samples for the model to be able to accurately capture the information contained within the current dataset. Therefore, a large and diverse dataset is essential to produce a robust and accurate machine learning model.

3.1 Gathering data

Recording Data

The data used in this thesis was recorded with a 60 GHz radar designed to be used for surveillance. The radar parameters can be found in Table 3.1. The data was collected by recording scenes with a camera and collecting radar data at the same time, the recorded video being used to aid with annotation. We will refer to a specific recording from one of these scenes as a scene or a sequence. Some of the scenes resemble typical radar installations that a customer might use, but most of the scenes are chosen specifically to produce at least some amount of multipath signals. The recordings can be divided into two subsets; 13 short ones around one to three minutes each, and three long ones around 10 minutes each. They mostly feature people walking in the scene. Some recordings also contain cars and bikes, but in this thesis we make no attempt to distinguish between the type of objects.

Previously recorded data

In order to minimize the amount of time that had to be spent on recording, we were given access to a large collection of proprietary data by our industry supervisors. Unfortunately, only a few scenes were found that were deemed useful for our purposes. The scenes were usually very short, and containing few multipath tracks in them. Many of them also had their static detections filtered out, which we needed

Table 3.1 Parameters for the radar used to record the data used in this thesis.

Radar type	FMCW
Operating frequency	60 GHz
Frame rate	10 frames/s
Maximum range	138 m
Range resolution	0.765 m
Velocity resolution	0.098 m/s
Azimuth accuracy	1°

as will be shown in the next chapter. All the publicly available datasets found were of no use to us, as they were mostly intended for use in development of autonomous vehicles, meaning the radar was not stationary.

3.2 Processing and annotating data

All detections above a certain velocity threshold were removed in order to remove false detections which most likely originated from signal noise caused by hardware in the radar. This threshold was chosen large enough to include walking persons and slow-moving cars. The data from each scene was processed by the proprietary tracking software outlined in Section 2.3 in order to create tracks. These were then manually annotated using the video recorded from each scene. The tracks which were derived from a person walking were annotated as real. Tracks which obviously originated from sources other than real detections or multipath effects were annotated to be ignored. These were usually generated by stationary objects swaying in the wind, like trees or flags. In scenes where this was prevalent, the areas where these tracks originated were marked, and all tracks created in those areas were automatically annotated to be ignored. Finally, all tracks which were not annotated as real tracks or flagged to be ignored were automatically annotated as ghost tracks.

An issue with annotating this way is that it is prone to human error. The camera used to record the video could not capture the entire scene, meaning people were sometimes moving outside of its field of view. Even when people are captured by the camera, it is not always evident which track stems from a real target, as can be seen in Figure 3.1. Another issue is that tracks can blend together, meaning a track can start from a multipath effect and then at one point it transitions to tracking a real target, or vice versa. In this case we chose to annotate the track as real or ghost depending on what the source of the track was for the majority of its lifetime.

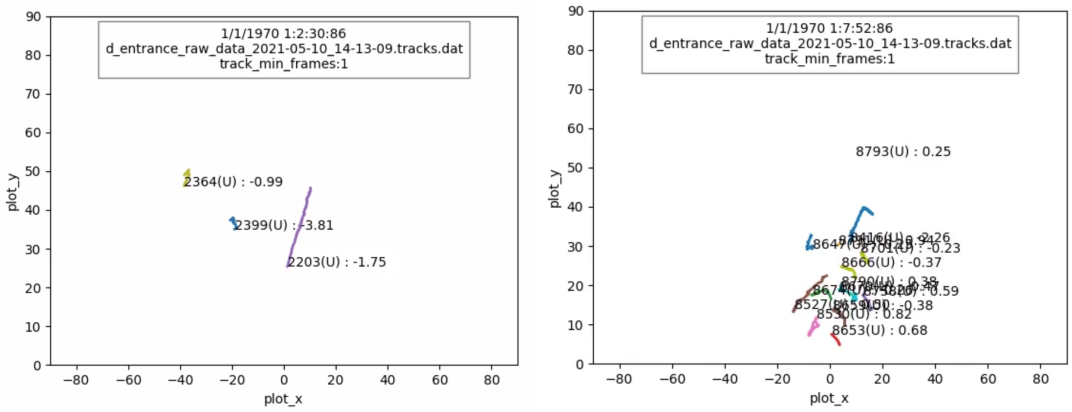

Figure 3.1 Plots of tracks generated by the tracker. A simple frame (left) where track 2203 is real, and a more problematic frame (right) where it is difficult to tell which tracks are real.

Table 3.2 The distribution of samples in the dataset.

Samples from real targets	Samples from ghost targets	Total samples
21 268	72 164	93 432

3.3 Structure of dataset

The dataset used in this thesis is divided into samples where each sample is made up of features belonging to one track in one frame. These datapoints can for example be the lifetime of the track, the position of the track and an estimate of its RCS value. A single track will therefore produce multiple samples, one for each frame that it is active. It is important to note that classification is done by sample, and not by track, meaning that a track can be classified as real in one frame and a ghost in the next frame. The distribution of the samples in the dataset is shown in Table 3.2.

4

Scene Model

Multipath propagation is dependent on the specific geometry of reflective surfaces in the scene that is being monitored. Given the simple scenario seen in Figure 1.1, only ghost tracks can appear beyond the wall to the right of the radar. However, the radar has no way of knowing this. It could be mounted in a completely different environment where there is no wall on its right side. In fact there are very few assumptions that can be made about the monitored scene. Therefore, trying to classify a track using its position without any knowledge of the scene would be ineffective. Inversely, if certain information specific to a scene can be mapped accurately by the radar, that information would be a strong tool to use for classification.

4.1 Scene-specific information

One type of scene-specific information that would be of use is the position of large reflective surfaces, like the walls of a building. If a track appears behind one of these surfaces it is very likely caused by multipath effects. In most cases these objects will not change over time, however one could easily imagine a few examples where this is not the case. If a radar unit was set up to monitor a construction site or a parking lot for large vehicles, the reflective surfaces could change over time if for example a large truck parks in front of the radar. Therefore, the scene model implementation should preferably be able to handle environments that behave somewhat dynamically.

Another way to utilize information specific to the monitored scene is to map the areas of the scene where real tracks tend to appear, and where they do not. An obvious problem here is that we do not know which tracks are real, but we can create features out of the the assumptions that were derived in Section 2.4 to make good guesses. In each frame that contains tracks, it is possible to identify which track is the strongest one, closest one, or has existed the longest. If the position of these tracks are stored, an approximate map over where assumed real tracks and ghost tracks frequently appear can be generated.

4.2 Mapping reflective surfaces

After a literature study, occupancy grid mapping was chosen as the method of modeling the reflective surfaces in the scene. It has an inherent advantage in that it not only maps areas of the scene which are occupied, but also areas that are unoccupied. This is of interest as real tracks should only appear in the unoccupied areas of the scene. Occupancy grid maps are also suited to map a dynamic environment over time, as shown by their popularity in the field of mobile robotics.

The grid cells in the map were defined in a Cartesian coordinate system where each grid cell represented an area of 1 m^2. We found that this relatively low resolution produced the most optimal map from a visual standpoint, while maintaining a relatively low level of computational complexity.

Extracting static detections

In order to map the reflective surfaces, all detections with an absolute radial velocity over a certain threshold were filtered and removed. This velocity threshold was chosen so that only the detections between the smallest positive and the smallest negative velocity bins were extracted. Since even moving objects can result in detections with no radial velocity, either due to micro-Doppler effects or by moving concentrically around the radar, a simple filter was implemented which removed any detections within 2 meters of a track. The remaining detections were then assumed to be caused by static objects and were used to build a map of the scene.

Occupancy grid map implementation

The gathered data was mapped by our own implementation of an occupancy grid map, that functioned in the following way. A grid map is created with a given size and resolution, where each cell has the occupancy probability p_0. For each frame in the recorded scene, and each static detection not close to a track in that frame, the grid map is updated with the detection according to the inverse sensor model described in the next section.

Inverse sensor model

As described in [Thrun et al., 2006], inverse sensor models can be learned by for example sampling from a sensor model and fitting a neural network to the generated samples. However, creating the best possible inverse sensor model was not the focus for this thesis. Instead, we constructed our own inverse sensor model with the purpose of being simple and computationally efficient. The algorithm is described below:

1. Receive detection at polar coordinates (r_0, θ_0).

2. Sample N_s pairs of coordinates from the distributions $r \in N(r_0, \sigma_r^2)$ and $\theta \in N(\theta_0, \sigma_\theta^2)$.

3. For each sampled point, update the grid cell containing the point with probability $p = p_0 + \frac{\alpha_m}{N_s}$ according to equation (2.22).

4. For each sampled point, use Bresenhams line algorithm [Bresenham, 1965] to find all grid cells in a straight line from the point to the radar at the point of origin. For all these grid cells, update the occupancy value with probability $p = p_0 - \frac{\beta_m}{N_s}$ according to equation (2.22).

p_0 is the prior probability of occupancy and σ_r^2 and σ_θ^2 are known beforehand from the forward sensor model. N_s, α_m and β_m are chosen with regards to the computational complexity and to how much the occupancy probability is updated when a detection is received. Because the radar in our model is stationary and detections are accumulated at a high frequency in some grid cells, α_m and β_m were chosen to be much lower than what was first theorized to be appropriate values.

Dynamicity in static detections

Static detections from reflective surfaces were initially thought to be time invariant, as long as those surfaces were not moved. Previous analysis by our industry supervisors had found that the static detections would be dominated by objects that reflect radio waves well, like house corners or metal poles, meaning large sections of walls would be invisible. This is due to the poor angle resolution of the used radar, meaning it is difficult to resolve individual detections for objects in the same range-doppler bin.

However, an interesting phenomenon was discovered when testing the mapping algorithm. When a person moved through the scene, the reflective surface behind the person would become visible to the radar. Only a cursory analysis was made to find the cause of this phenomenon, but one possible explanation is that it stems from the fact that when an object obscures a large flat surface, the gap created in the surface makes it easier to resolve the detections next to it in the Fourier transform.

4.3 Feature density maps

In order to find the areas where real tracks are appearing, we map the spatial and temporal density of the features described in Section 4.1. These feature density maps are also represented using Cartesian oriented cells in a grid, which are updated as follows:

1. Receive track in grid cell (i, j).

2. If the value of the relevant feature is true in the grid cell: Update the grid cell with probability $p = 0.5 + \alpha_s$ according to equation (2.22).
   If the value of the relevant feature is false in the grid cell: Update the grid cell with probability $p = 0.5 - \beta_s$ according to equation (2.22).

Table 4.1 Parameters for the feature density maps

Grid dimensions	240x120 m
Cell dimensions	1x1 m
α_s	0.01
β_s	0.002

This process results in a map similar to the occupancy grid map, but with the purpose of mapping common or a lack of common occurrences of certain features.

5

Features

Features are the properties of the tracks that were fed into the classifiers. They were chosen in order to capture only the most relevant information in the data, this selection process is described in Section 6.8. The features are divided into two types, track-specific features which were extracted from the data generated by the tracker, and scene-specific features which were extracted from the scene models.

5.1 Track-specific features

The track-specific features mentioned in this report are shown with a brief explanation in Table 5.1. These features are defined for each track in each frame. The *strongest*, *closest*, and *longest* features are binary, and computed by comparing the given track with other tracks currently active in the same frame. Similarly, *relative SNR* is a comparative measure, and is calculated by taking the SNR of the given track, and dividing it with the highest SNR of all the tracks in the frame. The *range std* and *doppler std* features are created by calculating the standard deviation of the respective measure for all the detections in the cluster belonging to that track. Finally, the *track lifetime* feature is the amount of time in seconds that the tracker has been able to keep the track alive.

Some other track-specific features that were considered are *weakest*, *furthest* and *shortest*. These features can be thought of as approximate indicators for ghost targets. As the names suggest, these features are essentially the opposites of the other binary features *strongest*, *closest* and *longest*. However, when deciding which features to use as described in Section 6.8, these features were found to not increase performance, and was therefore not included in the final models. A possible explanation for this is that a multitude of ghost tracks are often created from one real track, and these binary features can only be active for one track at a time, essentially missing many of the ghost tracks.

Table 5.1 Explanation of track-specific features.

Feature	Explanation
track lifetime	The amount of time in seconds the tracker has kept the track alive
range std	Standard deviation of range of detections belonging to the track
doppler std	Standard deviation of Doppler of detections belonging to the track
relative SNR	SNR of track divided by the highest SNR of all current tracks
strongest	1 if RCS of track is highest of all current tracks, otherwise 0
closest	1 if range of track is lowest of all current tracks, otherwise 0
longest	1 if lifetime of track is highest of all current tracks, otherwise 0

5.2 Scene-specific features

The scene-specific features were created by inputting the position of the track into the scene model, and are listed in Table 5.2. The three density features are created by finding the cell that represents the tracks current position in the corresponding feature density map, and taking the value of that cell. The *occupancy probability* feature is created in the same way from the occupancy grid map of the reflective surfaces. Finally, the *occlusion score* feature is designed to indicate whether there is a clear line of sight from the radar to a track, or if the track is occluded by a static object. It is created using the occupancy grid map of the reflective surfaces and is calculated as follows:

1. Receive track in grid cell (i, j).
2. Use Bresenhams line algorithm [Bresenham, 1965] to find all grid cells in a straight line from cell (i, j) to the grid cell containing the radar.
3. The occlusion score is the maximum value of the occupancy probability of these grid cells.

Table 5.2 List and explanation of scene-specific features.

Feature	Explanation
occupancy probability	The occupancy probability in the grid map
occlusion score	Approximate probability that a track is occluded
strongest density	Aggregated density of "strongest" feature
closest density	Aggregated density of "closest" feature
longest density	Aggregated density of "longest" feature

6

Model structures and evaluation

6.1 General structure

Two types of classifier model structures were considered in this thesis. In one of the structures, only the track-specific features from Section 5.1 were used for classification. This model structure is visualized as in Figure 6.1. The classifier block in this structure was implemented with a random forest in classifier model 1 and with a multilayer perceptron in classifier model 3, both of which are described later in this Section.

The other structure can be seen in Figure 6.2. In this structure, a model of the scene is created and scene-specific features are extracted from the model as in Section 5.2. The classifier block in this structure was implemented with a random forest in classifier model 2 and with a multilayer perceptron in classifier model 4, both of which are described later in this Section. All machine learning models were implemented in the Python programming language using the PyTorch framework.

6.2 Naive classifier

To be able to compare our models with a baseline model we created a simple naive classifier. The classifier predicts a track to be a real track if the track is currently the closest one to the radar of all active tracks in the frame, otherwise it predicts the track to be a ghost track. This prediction is identical to the value of the feature "closest" described in Table 5.1. This is a naive approach because if there are several real tracks active at once, then only the closest one of the real tracks will be classified correctly. It could also happen that the radar does not have a clear line of sight to a target, and the closest track is in fact a ghost track caused by reflections.

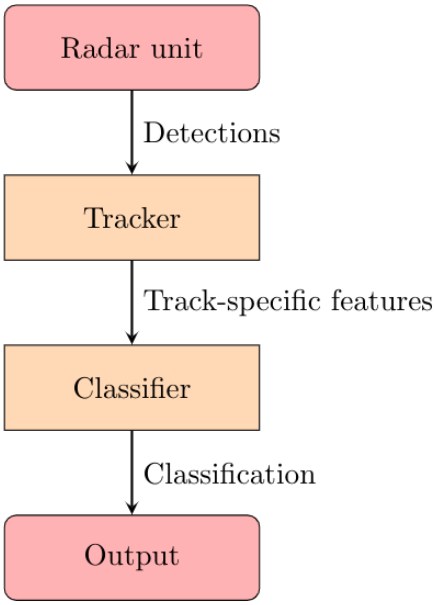

Figure 6.1 Model structure of classifier with no scene model.

6.3 Classifier model 1 — Random forest with only track-specific features

Classifier model 1 was a random forest classifier, where only the track-specific features were used. The number of decision trees used in the random forest was 100. As mentioned in Section 2.5, the number of features considered in the decision trees were $\sqrt{n_f}$ rounded to the nearest integer where n_f is the total number of features. Because the dataset used contained more samples of ghost targets than real targets, the samples were weighted $1:2$ so that each sample of a real target was weighted equal to two ghost targets when making a split in the decision trees to make the dataset more balanced.

6.4 Classifier model 2 — Random forest with scene-specific features

Classifier model 2 was a random forest classifier, where both track-specific and scene-specific features were used. Apart from the difference in the features used, classifier model 2 had exactly the same structure as classifier model 1.

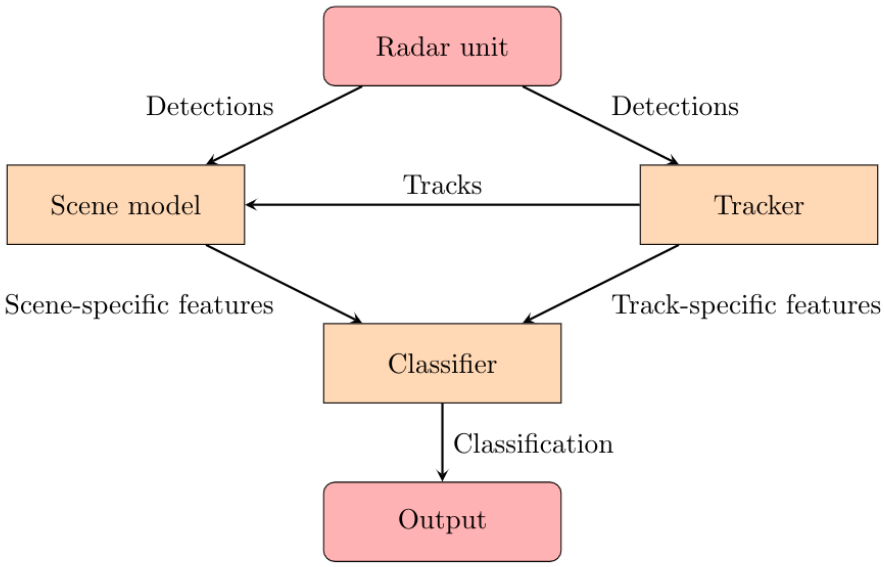

Figure 6.2 Model structure of classifier which uses a scene model to improve classification performance.

6.5 Classifier model 3 — Multilayer perceptron with only track-specific features

Classifier model 3 consisted of a multilayer perceptron where only track-specific features were used. Three hidden layers were used where each hidden layer consisted of 128 nodes. Dropout as described in section 2.5 was used in all of the hidden layers when training the network with a dropout rate of 0.3. ReLU was used as the activation function in all of the layers. The loss function used was the cross entropy loss function described in Section 2.5. The loss function weights were 1 for samples from ghost targets and 2 for samples from real targets, meaning that every sample from a real target had twice as much impact on the loss as a sample from a ghost target. The Adam optimization algorithm [Kingma and Ba, 2014] was used when training the model. The learning rate used to update the network weights was $\eta = 10^{-4}$. The network was trained for 50 epochs, meaning that on average, every sample was used to update the network weights 50 times.

6.6 Classifier model 4 — Multilayer perceptron with scene-specific features

Classifier model 4 consisted of a multilayer perceptron with exactly the same structure as classifier model 3, the only difference being that scene-specific features were used in addition to track-specific features.

6.7 Evaluation

In order to evaluate the performance of the classifiers, leave-one-group-out cross-validation (LOGOCV) was used. LOGOCV is a special case of leave-one-out cross-validation [Magnusson et al., 2020], the difference being that LOGOCV omits a group of samples in each training set instead of a single sample. The groups in our case were the different recordings in our dataset. Data from all but one recording was used to train the classifier model, and then precision, recall and F_1-score were calculated for the two classes, as well as the weighted and unweighted averages of these metrics over the classes. This was then repeated for all the recordings in the dataset. Finally, the metrics were averaged over all the splits, with the splits weighted equally regardless of the length of the recording being evaluated. We consider this method to be the harshest but most fair method of evaluating the classification performance.

Another method of evaluating the classification would be to split every track in every frame randomly into a training set and evaluation set regardless of which recording the track is from. We will refer to this as random split evaluation. This yields a much better classification performance, but tells us little about how the classifier would perform in a completely new setting.

6.8 Feature selection

In order to determine which features were useful for the classification task, the models were trained on a random subset of the dataset with all potential features included. Then, new models were trained on the same dataset where one feature at a time was left out and the classification performance of these models were compared to the performance of the model with all potential features included. If no significant reduction in performance could be measured when excluding a feature, that feature was deemed not worth including in the final models. The resulting features are the ones described in Chapter 5. The reason for using random split evaluation here was to make the process computationally feasible. This selection of features is not guaranteed to give the best results when using LOGOCV evaluation, but it is an approximation of which the most optimal features to include are.

6.9 Model parameter tuning

Due to the computational complexity of the chosen cross-validation evaluation method, a true grid search could not be performed for the model parameters such as the structure of the neural networks and the learning rate used when training the neural networks. A grid search is the process of training and evaluating a model with every possible combination of some discrete set of parameter values (every point in the parameter space). The number of the decision trees used in the random forests

as well as the number and size of the layers in the multilayer perceptrons were chosen large enough such that an increase in the size of the models would yield no significant improvements in the classification performance. To make this feasible this was done without cross-validation, by instead using the random split method of evaluation. Small models were trained and evaluated, iteratively increasing the model sizes until no significant increase in performance could be measured.

7

Results

7.1 Classification results

Naive classifier

The naive classifier does not require any training data, and thus LOGOCV evaluation as described in Section 6.7 is not necessary, but it was still used in order to obtain the most fair comparison with the other classifier models. The classification results from the naive classifier using LOGOCV evaluation can be seen in Table 7.1 The success of the naive classifier depends completely on the amount of datapoints which only have one track in a given frame, as the naive classifier will be very accurate for such datapoints. However, if multiple real tracks are frequently active at the same time in one frame, then the naive classifier will incorrectly classify all but one of these real tracks as ghost tracks. This would lead to a poor recall score for real targets in the evaluation.

Table 7.1 Classification results for the naive classifier.

Class	precision	recall	F_1 score
Ghost target	0.89	0.90	0.89
Real target	0.75	0.74	0.73
weighted average	0.86	0.86	0.85

Classifier model 1 — Random forest with only track-specific features

The classification results for the random forest with only track-specific features using LOGOCV evaluation can be seen in Table 7.2. The class-weighted F_1 score is better than the naive classifier which is the main metric considered here when evaluating the models. It can also be noted that the precision and recall scores are higher for both the "Ghost target" and "Real target" classes individually.

Table 7.2 Classification results for classifier model 1.

Class	precision	recall	F_1 score
Ghost target	0.90	0.94	0.91
Real target	0.82	0.75	0.77
weighted average	0.88	0.89	0.88

Classifier model 2 — Random forest with scene-specific features

The classification results using LOGOCV evaluation for the random forest with scene-specific features included can be seen in Table 7.3. Including scene-specific features in the random forest classifier improved the performance slightly compared to using only track-specific features, but the improvement is quite small. A possible explanation is that many of the recordings in the dataset were too short for the scene-specific features to provide relevant information.

Table 7.3 Classification results for classifier model 2.

Class	precision	recall	F_1 score
Ghost target	0.90	0.95	0.92
Real target	0.86	0.75	0.79
weighted average	0.89	0.90	0.89

Classifier model 3 — Multilayer perceptron with only track-specific features

The multilayer perceptron classifier with exclusively track-specific features also outperforms the naive classifier in F_1 score using LOGOCV evaluation as can be seen in Table 7.4. However, by noting that the recall score is high for ghost targets but low for real targets, we can conclude that this classifier is skewed towards classifying targets as ghost targets. One possible explanation is that the dataset was imbalanced with more ghost targets represented, and weighting real targets to be more important was not enough to overcome this imbalance.

Table 7.4 Classification results for classifier model 3.

Class	precision	recall	F_1 score
Ghost target	0.86	0.98	0.91
Real target	0.91	0.63	0.73
weighted average	0.87	0.90	0.87

Classifier model 4 — Multilayer perceptron with scene-specific features

As can be seen in Table 7.5, including scene-specific features in the multilayer perceptron classifier does not improve the performance when using LOGOCV evaluation. Just like in model 3, the recall score for real targets are low, but the recall score for ghost targets is very high. We can see that there is an interesting difference between the models 1 and 2 which use a random forest and the models 3 and 4 which uses a multilayer perceptron. The difference is that the classification results are more balanced for models 1 and 2, while model 3 and 4 classify more samples as ghost targets. A possible explanation for this is that there are a few features which have a particularly strong correlation with the class of the sample. Due to the inbalanced dataset, the multilayer perceptrons can transform these features such that most of the samples get classified as ghost targets. But a random forests consists of many decision trees which each only use a subset of the features. If these particularly informative features are not selected when fitting a particular decision tree, the classification result could be worse for that particular tree, essentially meaning that the results are more random and thus more balanced.

Table 7.5 Classification results for classifier model 4.

Class	precision	recall	F_1 score
Ghost target	0.86	0.97	0.91
Real target	0.92	0.62	0.73
weighted average	0.87	0.89	0.87

Random split evaluation

Classification results when using random split evaluation for model 1 and 2 can be seen in Tables 7.6 and 7.7 respectively. The random split evaluation method does not say anything about how well the classifier performs on unseen data from completely new scenes. However, by comparing the results from model 1 and model 2 and noting that the performance is better for model 2 we can conclude that the scene specific features contain some information that is useful for the classification task. This observation combined with the marginal performance gain from including scene-specific features when using LOGOCV evaluation could mean that the scene-specific features contain at least some useful information, but do not seem to generalize well to new scenes.

7.2 Visualization of scene models

In this section we will show some examples of occupancy grid maps and feature density maps produced in this thesis. The maps are shown with their state at the

Table 7.6 Classification results using the random split evaluation method for classifier model 1.

Class	precision	recall	F_1 score
Ghost target	0.93	0.96	0.94
Real target	0.84	0.75	0.80
weighted average	0.91	0.91	0.91

Table 7.7 Classification results using the random split evaluation method for classifier model 2.

Class	precision	recall	F_1 score
Ghost target	0.97	0.99	0.98
Real target	0.95	0.90	0.92
weighted average	0.97	0.97	0.97

end of the recorded scenes. All maps in this section are produced with the same mapping parameters, which can be seen in Table 4.1 and 7.8.

Table 7.8 Parameters for the occupancy grid map

Grid dimensions	240x120 m
Cell dimensions	1x1 m
p_0	0.5
σ_r^2	0.25 m
σ_θ^2	0.75°
α_m	0.01
β_m	0.0005
N_s	10

One of the scenes where data for our thesis was recorded is shown in Figure 7.1. The radar faces the corner of a building, and there is a metal container in front and to the left of the radar. Data was collected while we walked around in front of the radar for a period of approximately two minutes. In the occupancy map shown in Figure 7.2, the walls of the building, the container in the left part of the photo, and the metal furniture in front of the radar are clearly visible. However, the map also shows static objects beyond the walls of the building. We believe that these objects are the results of multipath effects. Two feature density maps of the same scene can be seen in Figure 7.3. The left image shows the density map of the *closest* feature, and in this case the density map seems to show where real targets have been present (dark area) and where ghost targets have been present (light area) in the scene. The right image in Figure 7.3 shows the density map of the *strongest* feature. It appears to show a higher density around the area where the container is located.

Another example of a recorded scene can be seen in Figure 7.4. The scene con-

Figure 7.1 Photograph (left) and aerial view (right) of one of the recorded scenes. The red triangle in the right image represents the position and orientation of the radar. The blue rectangle represents the metal container that was missing in the original image.

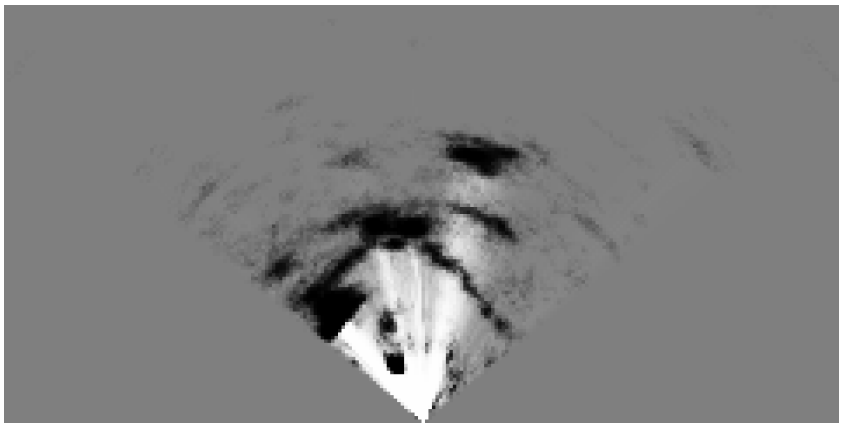

Figure 7.2 Occupancy grid map produced by recording the scene shown in Figure 7.1. The walls of the building, the container and the metal furniture are visible in the map.

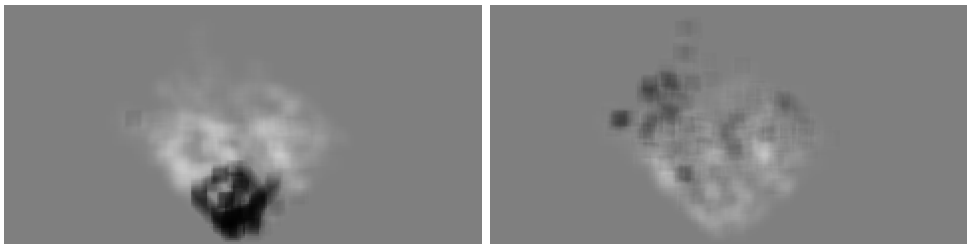

Figure 7.3 Density maps of "closest" feature (left) and "strongest" feature (right).

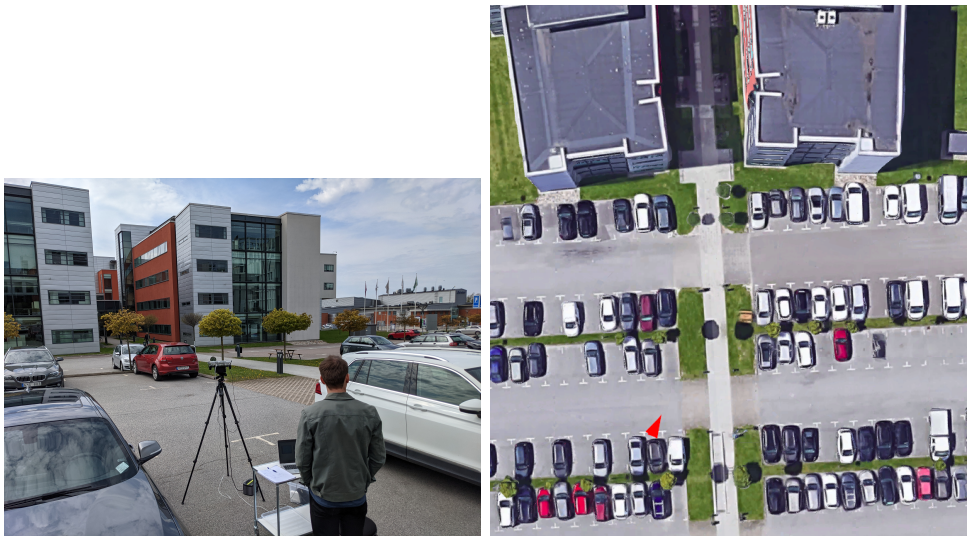

Figure 7.4 Photograph (left) and aerial view (right) of one of the recorded scenes. The red triangle in the right image represents the position and orientation of the radar.

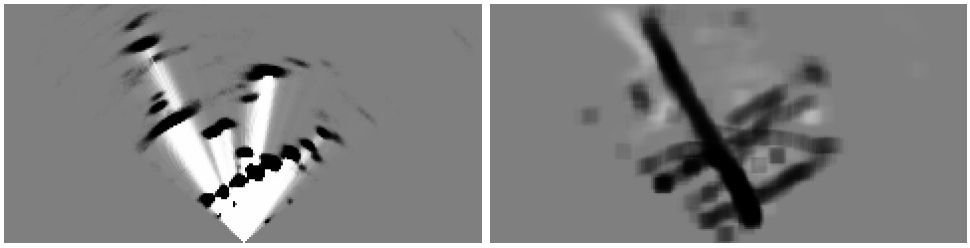

Figure 7.5 Occupancy map (left) and density map of "closest" feature (right) produced by recording the scene in Figure 7.4.

sists of a half-full parking lot with a footpath through the middle of it and some trees next to the path. Data was collected for approximately 10 minutes, during which some people walked in the parking lot and along the footpath in the middle of the scene. Figure 7.5 shows the occupancy map to the left and density map of the *closest* feature to the right. The reason that the occupancy map looks more sharply defined than the occupancy map in Figure 7.2 is that the map has been created over a longer period of time, and has thus been updated with more detections. In the occupancy map, a problem with the *occlusion* feature can be seen. The cars and trees in front of the radar resemble a wall or fence in the map, and thus all tracks appearing behind this line of trees and cars will have a high *occlusion* feature value even though they are not really occluded by a reflective surface. The left image in Figure 7.5 shows what a feature density map might look like in a more realistic scene. The footpath in the middle is clearly visible, where the presence of real targets is high relative to the rest of the scene.

8

Discussion

8.1 Limitations

Quality of data

A significant problem with the data collected and used in this thesis is that some sequences do not really resemble realistic scenarios. Some sequences of data were recorded with the specific purpose of eliciting ghost tracks, and this data can certainly be used to train classifier models that only use track-specific features. However, when building a model of the scene it is preferable to have longer and more realistic sequences. This is because of the fact that more detections are collected to improve the quality of the maps of the static environment and the feature density maps. Longer and more realistic sequences of data are necessary to improve the quality of the scene-specific features created, and furthermore, to improve the performance of the classifier models that use scene specific-features.

Another consequence of these sequences recorded with the purpose of eliciting ghost tracks is that the dataset is quite imbalanced, with about three times as many samples from ghost tracks compared to real tracks. An attempt was made to mitigate this, with weighted loss functions and samples as described in Chapter 6. Regardless, the classifier models that were trained in this thesis have a propensity towards classifying samples as ghost tracks. To make a classifier more suited to realistic scenarios, more realistic data would be needed for the training process.

One final issue with the data used in this thesis is the limitations of the proprietary tracking software and the manual annotation process that are outlined in Section 3.2. These limitations would lead to a small amount of tracks being annotated incorrectly. A solution was briefly discussed where ambivalent or problematic tracks would be pruned or split by manually altering the data. This was however deemed too time-consuming for the marginal gain that would probably be achieved.

Evaluation of scene model

Evaluating and tuning the maps in the scene model from a visual standpoint only, is far from the optimal way of maximizing their performance. A literature study pro-

duced some methods for evaluating grid maps, most promising being the solution proposed by [Schwertfeger et al., 2010], however it would take quite some time to implement. Since the purpose of this thesis is to classify ghost tracks, and not to build as accurate a scene model as possible, it was deemed satisfactory to use its effect on the classification as the only metric of its performance.

This however leads to another issue. Since the scene models are built iteratively over time, the quality of the scene models and thus also the scene-specific features, will be low for all samples early in the recording. In a realistic scenario, the radar will monitor the same environment for months, perhaps even years. Assuming mapping parameters are chosen so that the scene model converges after a week, all tracks after that point will be classified using roughly the same converged scene model. The effects of this could have been investigated by training and evaluating the classifiers where all scene-specific features are extracted from the final converged scene models at the end of the recording. Additionally, some perfect and some purposefully bad scene models could have been generated and tested in the same way to see how much this impacted the classification results.

Scene model tuning

One issue with the proposed scene-specific features is that they require choosing the value of many parameters. For instance, to create the map of the static environment the value of seven parameters need to be chosen: σ_r, σ_θ, N_s, p_0, α_m, β_m, and the grid map resolution. σ_r and σ_θ can be assigned reasonable values by looking at the radar specifications, but the other five need to be determined by using them in the mapping algorithms and evaluating the results. Creating these maps for all the recorded sequences requires significant processing time. It was therefore considered infeasible to attempt a grid parameter search, and the parameters were tuned until the scene specific features looked good visually. This method is of course not guaranteed to provide the best model performance in the end, but it was deemed to be the only feasible alternative considering the time and computational resources that were available to us.

8.2 Future work

Not much previous research has been published on the problem of classifying ghost tracks with a static radar. That meant we could not build upon previously proposed solutions, but instead we had to propose our own solutions, prototype them and evaluate them. Naturally there was not enough time to try all of the ideas that we had, and so we prototyped the solutions that seemed the most promising, which in the end resulted in our model proposals. We believe however, that there is much room for improvement given more time.

Classifier

An area of improvement for the classifier is to better utilize the information contained in the temporal evolution of the features of a track, perhaps by using a Recurrent Neural Network. Another alternative is to represent the features of a track as spectrograms which are used as input to a convolutional neural network. Another way of classifying tracks could be to perform a classification after the track has been active for some fixed period of time, and then never change the classification. As many ghost tracks are short lived, and samples with a short current lifetime are prone to be classified as ghost tracks, this could alleviate some incorrect classifications of real tracks at the beginning of their lifetime.

Scene model

In its current implementation, the scene model uses the data to build its maps immediately as it is available. However, in a real implementation, the scene model could be built over a much longer time span than the data in this thesis has allowed. Therefore, it would be possible to take a slower approach to mapping. One way is to wait until a track has died, and then analysing it over its entire lifespan before building the scene model using that information. This could potentially be used to filter out tracks which are more uncertain from being added to the feature density maps.

The occupancy grid map could be improved by adding a filtering step before the scene-specific features are generated. Objects like trees and metal poles generate a lot of static detections, but they don't necessarily block detections from people walking behind them. An extreme example of this can be seen in Figure 7.5, where the row of trees create what the scene model treats as a solid wall. Finding a way to identify which static objects are obscuring objects behind them, and which are not, would mitigate this problem. The occupancy grid maps produced could also be used to find lines in the maps and thus more rigidly infer the position of walls and large reflective surfaces in the scene. One could also expand on the work in [Nüßlein, 2021] to try and detect reflective surfaces by looking at the positions of ghost tracks in the scene, and guessing where reflections have occurred. Finally, removing the assumption that cells in the grid maps are independent could produce better maps at a cost of more computationally complex map updates [Thrun et al., 2006].

9

Conclusion

The proposed classifier models prove especially effective in detecting ghost tracks, but a significant proportion of the real tracks are misclassified as ghost tracks, especially in the case of model 3 and 4. If the objective is to minimize false alarms, this is preferable over having a low recall score for ghost tracks. Regardless we believe that this flaw could be mitigated by either using balanced sampling of datapoints when training the models, or by extending the dataset with more balanced data.

We believe that many of the comparative features proposed such as *closest* and *relative SNR* provide a solid base on which many different types of classifiers can be constructed. It is also our opinion that the scene-specific features extracted from the scene models has the potential to improve the classification results. But to be able to utilize and evaluate these features, data recorded over longer periods of time will be necessary, preferably from radar installations in real environments.

From a purely visual perspective, the occupancy grid maps produced surprisingly accurate maps for many of the recorded scenes. The mapping algorithm is quite simple and customizable, although it can be difficult to tune the parameters so that the algorithm produces accurate maps for multiple scenes with just one set of parameters.

We believe that this thesis shows it is possible to reliably classify ghost tracks with a stationary radar using our proposed method. However, there is a lot of potential for better results with higher quality data, and the proposed improvements to our models.

Bibliography

Bresenham, J. E. (1965). "Algorithm for computer control of a digital plotter". *IBM Systems Journal* **4**:1, pp. 25–30. DOI: 10.1147/sj.41.0025.

Brownlee, J. (2020). *A gentle introduction to cross-entropy for machine learning*. https://machinelearningmastery.com/cross-entropy-for-machine-learning (Accessed 2021/06/07).

Chen, V., F. Li, S.-S. Ho, and H. Wechsler (2006). "Micro-doppler effect in radar: phenomenon, model, and simulation study". *IEEE Transactions on Aerospace and Electronic Systems* **42**:1, pp. 2–21. DOI: 10.1109/TAES.2006.1603402.

Duhamel, P. and M. Vetterli (1990). "Fast fourier transforms: a tutorial review and a state of the art". *Signal Processing* **19**:4. ID: 271605, pp. 259–299.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press. ISBN: 9780262035613.

Gurney, K. (1997). *An Introduction to Neural Networks*. UCL Press. ISBN: 1857285034.

Hastie, T., R. Tibshirani, and J. Friedman (2008). *The Elements of Statistical Learning (2nd ed.)* Springer. ISBN: 0-387-95284-5.

James, G., D. Witten, T. Hastie, and R. Tibshirani (2017). *An Introduction to Statistical Learning: with Applications in R, Corr. 7th printing 2017*. Springer Texts in Statistics, pp. 303–330. ISBN: 1461471370.

Kalman, R. E. (1960). "A new approach to linear and filtering problems". *Transactions of the ASME-Journal of Basic Engineering*.

Kingma, D. P. and J. Ba (2014). "Adam: a method for stochastic optimization". English.

Kraus, F., N. Scheiner, W. Ritter, and K. Dietmayer (2020). "Using machine learning to detect ghost images in automotive radar". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–7. DOI: 10.1109/ITSC45102.2020.9294631.

Laber, E. and L. Murtinho (2019). "Minimization of gini impurity: np-completeness and approximation algorithm via connections with the k-means problem". English. *Electronic notes in theoretical computer science* **346**.

Magnusson, M., A. Vehtari, J. Jonasson, and M. Andersen (2020). "Leave-one-out cross-validation for bayesian model comparison in large data". In: Chiappa, S. et al. (Eds.). *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Vol. 108. Proceedings of Machine Learning Research. PMLR, pp. 341–351.

Nüßlein, A. (2021). *Multipath modelling for 60ghz short range radar*.

Prophet, R., J. Martinez, J.-C. F. Michel, R. Ebelt, I. Weber, and M. Vossiek (2019). "Instantaneous ghost detection identification in automotive scenarios". In: *2019 IEEE Radar Conference (RadarConf)*, pp. 1–6. DOI: 10.1109/RADAR.2019.8835603.

Rao, S. (2017). *Introduction to mmwave sensing: fmcw radars*. Texas Instruments. URL: https://training.ti.com/sites/default/files/docs/mmwaveSensing-FMCW-offlineviewing_0.pdf (visited on 2021-06-21).

Richards, M. A. (2005). *Fundamentals of Radar Signal Processing*. McGraw-Hill Professional, pp. 42–50. ISBN: 0071444742. DOI: 10.1036/0071444742.

Ryu, I.-h., I. Won, and J. Kwon (2018). "Detecting ghost targets using multilayer perceptron in multiple-target tracking". *Symmetry*. DOI: 10.3390/sym10010016.

Schwertfeger, S., A. Jacoff, C. Scrapper, J. Pellenz, and A. Kleiner (2010). "Evaluation of maps using fixed shapes: the fiducial map metric". In: pp. 344–351. DOI: 10.1145/2377576.2377638.

Sharma, S. (2017). *Activation functions in neural networks*. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (Accessed 2021/06/07).

Siciliano, B. and O. Khatib (2007). *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, pp. 1136–1150. ISBN: 354023957X.

Sim, P. (2014). *Making waves: Robert Watson-Watt, the pioneer of radar*. https://www.bbc.com/news/uk-scotland-tayside-central-27393558 (Accessed 2021/02/26).

Smith, S. W. (1997). *The Scientist & Engineer's Guide to Digital Signal Processing*. California Technical Pub. ISBN: 0966017633.

Srivastava, N., G. Hinton, A. Krizhevsky, and R. Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting". *Journal of Machine Learning Research* **15**:56.

Thrun, S., W. Burgard, and D. Fox (2006). *Probabilistic robotics*. MIT Press, Cambridge, Mass. [u.a.] ISBN: 0262201623.