Delft University of Technology
Master of Science Thesis in Embedded Systems

Privacy preserving detection and classification of playground users.

Bernard Bekker

TU Delft Delft University of Technology

Privacy preserving detection and classification of playground users.

Master of Science Thesis in Embedded Systems

Embedded and Networked Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Bernard Bekker
bernardbekker@student.tudelft.nl
Bernard@BernardBekker.nl

22nd May 2020

Author
Bernard Bekker (bernardbekker@student.tudelft.nl)
(Bernard@BernardBekker.nl)
Title
Privacy preserving detection and classification of playground users.
MSc Presentation Date
29-05-2020

Graduation Committee

Dr. ir. Fernando Kuipers (chairman)	Delft University of Technology
Dr. Marco Antonio Zúñiga Zamalloa (direct supervisor)	Delft University of Technology
Dr. Lydia Y. Chen	Delft University of Technology

Abstract

Large cities in the Netherlands, like Rotterdam, have hundreds of playgrounds, but local governments have little information on how children and adults use them. Usage data can help create playgrounds that better fit with the residents' needs by identifying what elements of a playground are the most popular, and which do not get used. The AMS Institute and municipality of Rotterdam have asked us to develop a system to collect usage information of playgrounds without recording personally identifiable information. The system requirements are to record the locations of individual users and to estimate if a user is a child or an adult. It must do this without requiring an external power source and without a broadband internet connection.

After considering different types of sensors (including computer vision, radio, sound, and mechanical acoustic signals), We decided to use a mmWave radar sensor due to its ability to provide accurate localization, easy installation, and low energy usage without recording identifiable information. We use a commercially available mmWave radar sensor that we configure to localize people in a 30m by 20m area when placed at the perimeter. We use the radar point cloud output from the device for classification by calculating statistics that we use as features for our classifier. We evaluate classifiers based on SVM, Random Forrest, fully connected and recurrent neural nets. We also analyze different methods for combining radar point clouds captured over time from the same person.

We collected 100.000 radar point clouds of adults, children, and bicyclists at real playgrounds, split into a training and validation data set. We show that our SVM classifier achieves an accuracy of 79% on *single* radar frames, and 93% on the combined results of *10 second* long sequences of our validation dataset. The classifier requires $<$ 1KB of memory and little processing power. Meaning it can execute on an embedded platform powered by a solar panel.

Preface

I want to thank all the people that supported me during this project. First of all, a special thanks to my supervisor Marco Zúñiga, who went above and beyond his duties as a supervisor, and recruited his entire family to help test the system. Another word of gratitude goes to the people who made this project possible, and helped set up a public deployment: Irma, Frank, Lisbeth en Nancy at Gemeente Rotterdam, and Thijs at AMS. Without them, this would have just been a boring piece of theoretical research in an office. I also want to thank Jodi Kooijman, who helped me gain a new perspective during some of the tough moments during my thesis, and whose constructive feedback helped improve this report. And finally, I would like to thank my parents. Who always pushed and supported me to improve myself and do what I thought was right. Without their encouragement I wouldn't have been at this point here today.

Bernard Bekker

Delft, The Netherlands
22nd May 2020

Contents

Chapter 1

Introduction

In this report, we detail the design of a privacy-preserving system for monitoring the usage of playgrounds. We started this project in cooperation with the municipality of Rotterdam, which has 1319 playgrounds within its borders. These playgrounds provide significant public benefits as places for exercise and meeting each other, improving the social cohesion and the health of residents in the city. However, since space in the city is scarce, and the upkeep of the playgrounds is expensive, the investments in these playgrounds must be efficient and fit the residents' needs. To be able to make these investments efficiently, the municipality requires information on how the playgrounds are currently used, and how the needs of the residents change over time.

Currently, the municipality has little information available about how children and adults use their playgrounds and what playground elements are popular. To provide this information, we want to design a system for automated, continuous measurement of the usage patterns in a playground.

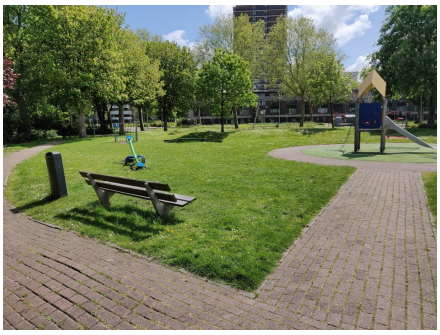
The municipality is not able to use existing methods using cameras since their local and national rules prevent the use of privacy-invasive methods unless there is a pressing security need. If our method is a success, it could lead the way to a broader use throughout the city, replacing the cameras currently in place.

1.1 Problem description

The system must be able to detect the presence of individual people, how long they stay at the playground, and what parts of the playground they use. Also, we want to be able to estimate if someone is a child or an adult. This data needs to be usable for statistical purposes to determine which playground designs drive the most usage.

Gemeente Rotterdam selected four reference sites that are part of the "move-fit" and "plug&play" initiatives. Both these initiatives feature interchangeable elements on the playground. For instance, equipment like the slide in figure 1.1a can be replaced by a swing. The goal of the sensor deployment at these locations is to determine which elements are the most popular.

Compared to general crowd monitoring, monitoring at these playgrounds has a unique set of problems. Children are an extra privacy-sensitive group, and for this reason, using cameras is prohibited. Since it's a public space, we can't ask



(a) **The plug and play playgrounds have interchangable pieces of play equipment that have a standardized footprint. Equipment can be changed while keeping the same foundations. These parks have large open spaces in between the large, wooden pieces of equipment.**

(b) **Movefit are temporary public fitness equipment. It consists of prefabricated pieces that can be quickly placed and removed in one piece. The area is small and well defined, with open areas surrounding it.**

the users to carry a device or self-report. The reference locations don't have power or data infrastructure for electronic sensors. Furthermore, the system needs to be able to withstand the environmental conditions at a playground: harsh weather, curious children, and the occasional soccer ball. And to be able to install the system widely, it must be affordable for a municipality.

In this work, we focus on developing the core technology of the system that gathers the location data, and performs a classification of the type of user. Developing the casing and statistical analysis tools is future work.

We converted the basic description from the municipality into a set of technical requirements:

1. The system has to perform two functions:
   (a) It has to track the location of individual users with an accuracy of 1m.
   (b) It has to detect if a person is a child or an adult. Extended goals include detecting specific classes of users such as bicyclists, wheelchair users, baby strollers, and pets.
2. The system needs to work with the constraints of a playground and should be easy to deploy for it to be worth the effort for the municipality.
   (a) It must operate on a playground of at least $20\,\mathrm{m} \times 20\,\mathrm{m}$
   (b) It must work without a wired external power supply. It can use a long-lasting battery, or a small solar panel to gather its energy.
   (c) It must transmit results over a wireless data communication network.
   (d) It must be able to be integrated in a casing that is robust, weatherproof, and aesthetically pleasing.
   (e) It must not require extensive modifications to the play equipment.

(f) It must not require the user to adjust their behavior or carry a device.

3. To do this, we need to conform to regulations and constraints:
   (a) The device must adhere to Dutch radio emission regulations.
   (b) The device must adhere to the GDPR Privacy regulations.
4. The system should be affordable in order to be widely deployed. (<€2000 per unit material cost)

Privacy requirements

The goal of this project is to create a privacy-preserving system. To do this, we should define what we mean with privacy-preserving, as there are many different interpretations of privacy. We use the interpretation of the authors of [49], which consider privacy to concern both the collection, processing and the use of information about a data subject. Some existing methods that are widely deployed in public, measure, but do not record sensitive information. These systems do not fulfill our requirements, as the act of measurement itself is considered a privacy invasion under this definition.

However, every system that collects information about an area can be (ab)used to extract some identifying information about the persons in that area [56]. So rather than privacy-preserving being a binary property, we should consider the degree of privacy provided based on a set of metrics. One of these metrics is between how many people can be distinguished: a system that can distinguish between people in a large group is less privacy-preserving that a system that can only distinguish between two (groups of) people. Our goal is to build a system that physically can only sense the data needed to distinguish between the groups we are interested in. At a minimum, the data collected should not be able to be used to uniquely identify a general member of the population in order to qualify as privacy-preserving.

1.2 Contributions

The contributions in this report are:

1. We present an analysis of sensors based on different physical signals. We identify mmWave radar as the type of sensor best fitting our requirements.
2. An analysis of how to use mmWave data for classification. We analyze data representations, features, and machine learning methods. We design a classifier using radar point clouds and an LSTM neural network.
3. We collected real-world data at playgrounds. We use the data to show that our classifier achieves a 91% accuracy at distinguishing between children, adults, and bicyclists.

The problem of playground monitoring is approached from a first-principles perspective taking into account our design requirements. An analysis of all person-sensing methods is given from which mmWave radar is chosen for our design. This is presented in chapter 2.1.

Localization using mmwave FMCW radar is a mostly solved problem, but the classification of objects is not, especially on hardware with limited resources. In chapter 2.3 we consider the state of the art in classification used in the application of self-driving cars. We develop our own method based on point cloud features and an LSTM neural network, this work is presented in chapter 4.

We have worked on the design of a system that can be deployed at public playgrounds, and had conversations with lawyers and government employees to work towards a real-world application. This has resulted in significant progress towards a broader implementation at playgrounds in Amsterdam and Rotterdam. This work is presented in chapter 3. However, doing all the work required for a public deployment took longer than the time available in this project, and finishing it is future work.

1.3 Terminology

In this report, we often refer to planes, directions and angles. These always refer to the coordinate system in figure 1.2.

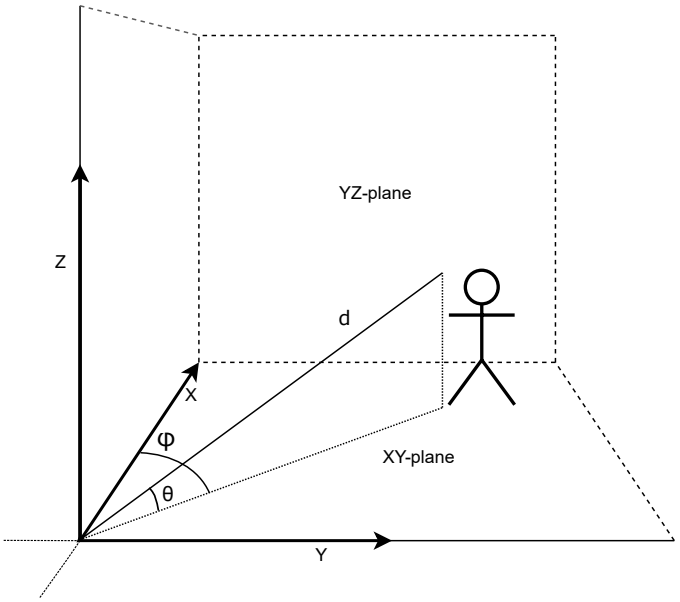


Figure 1.2: **The coordinate system and angles used in this report. We always use the reference frame of the sensor, unless specified otherwise. A location in the coordinate system can be defined by both the carthesian coordinates (xyz), or by the polar coordinates of distance (d), Azimuth (Φ) and Elevation (θ).**

Chapter 2

State of the art and Background

Our assignment does not specify what technology to use, and we did not find previous work on our exact topic. However, similar work has been done on person detection and classification using many different sensing techniques. To find the most suitable sensor technology for our application, we considered what signals could be used for detection, localization, and classification, and performed a literature study of how these signals have been used for person detection. This work is presented in section 2.1. In section 2.2, the chosen sensor technology, mmWave radar, is explained in more detail. In section 2.3, we show the state of the art in the area of object detection using mmWave radar data.

2.1 Person sensing methods

In this section, we present methods for detecting persons in the area of a playground. We present a broad overview of all methods, and we explore the most promising techniques that match our requirements in more detail.

Before considering specific sensors, we identify what signals can be used to detect people. We have identified three usable types of signals: Electromagnetic radiation, sound waves, and mechanical pressure waves. For each signal, we consider by which methods they can be measured, and how these methods fit our requirements.

2.1.1 Sound waves

People produce, absorb, and reflect sound waves when talking, walking, and playing. Prior research has shown methods that can locate people[31] and classify their activities [29] based on generated sounds. However, these methods record sound in the human speech range, and since speech can be identifiable and confidential, this violates the privacy requirement. To avoid privacy concerns, we could use a method that works outside the frequency range used for human speech of 100 Hz to 8 kHz. Low-frequency signals (< 100 Hz) are limited in use for localization: The range resolution is limited by the bandwidth

(BW) of the signal according to $resolution = v_{soundinair}/BW$, limiting us to a best-case accuracy of 3 m at 100 Hz.

Ultrasonic echolocation (> 20 kHz) has widespread use for the detection of objects underwater by boats and submarines, and imaging of human organs. By using an array of transducers, the range and azimuth of objects can be determined. This technique is popular underwater, but the use in air is limited because the propagation properties of sound in air reduce the range and accuracy. Commercial ranging sensors are limited to a 6m range in open air [25]. Academic machine learning-based techniques do not measure range directly but make location estimations by monitoring the acoustic properties of complex reflections [48, 19]. However, these papers focus on relatively small indoor environments, and the technique may not scale to a large playground.

Based on this research, it seems that using an ultrasonic sensor approach might be possible, the range will likely be too limited for the size of the area of interest in our application, and require multiple sensors. A system using machine learning for localization will likely require extensive calibration for each playground, making the installation more difficult.

2.1.2 Mechanical waves

When a person makes contact with the ground or any play equipment at the playground, they cause vibrations in the ground and the structures. By instrumenting the playground with vibration sensors, we could detect when play equipment is used. We hypothesize that amplitude, frequency, and phase of the vibrations can carry extra information about the person and the activity. For instance, the natural resonance frequency of a structure will change based on the added weight of the person using it. However, placing a sensor on a structure will only monitor that single structure, potentially requiring a large number of sensors to instrument each piece of play equipment. This violates the requirements of easy installation and to not modify the play structures. In addition, such a system does not monitor open spaces such as football fields.

In order to detect people in open areas, footsteps could be detected by seismic waves [52]. However, typical playground design uses impact absorbing materials for children's safety, which would reduce the signal.

In conclusion, using mechanical waves for usage monitoring would likely result in an incomplete overview of the usage patterns, or require many sensors to instrument every aspect of the playground.

2.1.3 Electro magnetic radiation

EM radiation, either in the form of (visible) light or radio waves, offers many methods of detecting objects at a distance. Every person emits Ifra red (IR) radiation from their body heat and can reflect external visible or radio signals. We will first look at systems using light, and then evaluate systems using radio. Both signals have intrinsic advantages and disadvantages. Light-based systems require a line of sight between the sensor and the person, while RF signals can pass through some obstacles. But RF suffers from signal reflections known as multipath, which can cause severe detection errors.

For light-based systems, we identified three types of sensors, with many different implementations:

- visible light cameras.
- IR sensors: IR cameras, PIR sensors, and Thermopile arrays.
- Depth sensors: LIDAR and Time of flight (TOF) cameras.

Many academic and commercial camera-based area monitoring systems exist. However, these systems do not fit our privacy requirements. The captured image will contain not only the data we are interested in but also other information that may infringe on people's privacy. In addition, a camera mounted on the perimeter of the playground can only view the Y/Z plane while we are interested in localization in the X/Y plane as shown in figure 1.2. Depth in the Y direction can only be estimated, based on perspective, or a stereovision system. This translation means we need a lot of information in a dimension we are not interested in, in order to estimate the dimension we are interested in. Placing a camera above the play area is not feasible for an area of $20\,\mathrm{m} \times 20\,\mathrm{m}$, as such a camera would have to be mounted several meters above the ground to contain the entire area in its field of view. Although camera systems can be modified to be more privacy-preserving by reducing their resolution, this will always come at the cost of localization performance.

IR sensors can detect the presence of people based on radiated body heat. Compared to visible light cameras, this offers two advantages: IR cameras work during the night and, because the objects of interest can be separated from the background based on heat, less identifiable information needs to be gathered [6]. There are three types of IR sensors in common use: IR cameras, Thermopile arrays, and PIR sensors. IR cameras are high-resolution cameras working in the thermal IR spectrum. Besides the advantages listed earlier, the disadvantage compared to visible light cameras is a reduced resolution and higher cost.

Thermopile arrays are a lower cost, lower power alternative, but are currently limited to a 32x24 pixel resolution [32]. But research has shown that this can be enough to classify people [34]. PIR sensors are simple sensors with a single sensing element. These sensors only detect the presence of movement.

Although IR sensors offer an improvement over visible-light cameras for the detection of people, they suffer from the same localization issue. They can not accurately determine the distance from the sensor without an additional sensor.

Time of flight (TOF) sensors are light-based distance sensors that send a light pulse and measure the time of flight of the reflection [43]. The different implementations of this technique can be clustered into two groups: TOF Cameras and LIDARS. TOF cameras use a wide-angle illuminator to illuminate the entire scene and use camera-like optics to determine the angle of arrival. The camera detects the time delay between when the illuminator is turned on, and the light is received at the image sensor. These types of sensors have a limited range and are susceptible to ambient light, limiting them to indoor use. LIDARs use or one or multiple modulated LASERs and sensor elements that scan an area using mirrors. The focused beam of light makes them less susceptible to ambient light. High-quality LIDARs have a superior range, accuracy, and

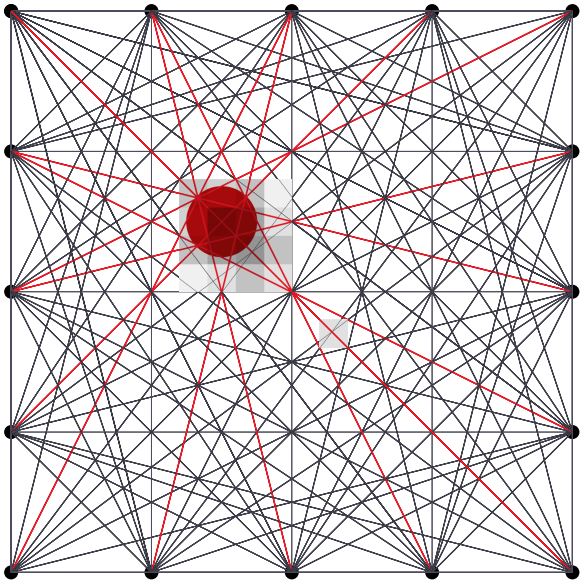


Figure 2.1: **Operating principle of Radio Tomographic Imaging. Every node measures the RSS of radio transmissions from every other node. Radio links that are obstructed by an object in its fresnel zone (in red) have a reduced RSS. By using a model of the radio channel, the locations of obstructions can be estimated. In this example, we use the average signal strength of all links passing over a pixel to color it.**

resolution compared to any other sensor. The downsides of LIDARs are their cost and fragility due to delicate moving parts. This makes LIDARs infeasible for widespread adoption in playgrounds. However, ongoing development into solid-state lidars might lower the cost and complexity in the near future [43].

Radio systems

There are passive and active methods for people detection using radio sensors. Passive radio systems that do not transmit their own signals are commonly used for crowd monitoring in cities or stores, for instance, by monitoring WiFi or Bluetooth broadcast messages sent by smartphones as shown in [44]. However, since our target audience includes young children, we cannot assume that those will carry a smartphone.

For active systems, we identify two main types of systems:

- Tomography (Radio attenuation)
- RADAR (Radio reflection)

Radio tomography uses the attenuation of radio signals between many transceivers to detect objects inside an area. This technique is explained in figure 2.1. The achieved accuracy and resolution scales with the number of transmitters and receivers. Over 200 nodes may be required to achieve the required localization accuracy on a $20\,\mathrm{m} \times 20\,\mathrm{m}$ field [57] making a tomographic system unfeasible in our situation.

Radar systems use reflections of signals rather than attenuation. By measuring differences between the transmitted signal and the received reflections, a determination of the location, velocity, and size of objects in the field of view is made. RADAR technology is a large field of research with many types of radars. Differences exist based on the type of signal they emit and the different topologies of antennas. Exploring all combinations is outside the scope of this project. Instead, we focus on three methods that have been used previously in short-range person detection contexts. These methods are continuous wave (CW) radar, pulse-doppler, and frequency modulated continuous wave (FMCW) radar.

Continuous-wave radar measures the signal strength and Doppler shift of radar reflections. This allows it to do detailed measurements of the movements of the body parts of a person. It is commonly used for movement detection, identification based on gait [9, 42, 27], fall detection [30] and vital sign (heart rate and breathing) monitoring [13]. However, a continuous wave radar can not measure range and can not separate the returns from multiple people. This makes a CW radar not useful for us, unless in combination with other sensors.

Pulse-doppler and FMCW radars have similar capabilities: both can sense the distance and velocity of an object. By using a MIMO (Multiple Input Multiple Output) antenna array, the angle of arrival of the reflections can be measured. The range and azimuth angles map directly to the X/Y plane, making them well suited to our use case. Pulse doppler radars can have a long-range but have a dead-zone near the radar, while FMCW radars are well suited to short ranges [39]. Therefore, for our range and power requirements, an FMCW radar offers a better trade-off .

FMCW radars are popular for use in driver assistance systems in cars where they are used to detect and localize moving objects, including pedestrians and bicyclists. Stationary objects can be filtered as a method to remove reflections from the ground and background objects. A specific type of FMCW radars is the mmWave radar. These radars use signals with a frequency range between 30 GHz and 100 GHz. The high frequency allows for a large bandwidth, resulting in sub-meter accuracy and small antennas. Within this frequency range, two bands have internationally been assigned to license-free use. This has lead semiconductor manufacturers to create affordable, integrated sensors for these frequency bands.

Radar is not without downsides. The high-frequency radar pulse only works in line of sight, an object can be hidden behind another object in front of it. Radars also suffer from multipath; the signal can be reflected by more than one object before returning to the sensor. This means that even after filtering all stationary objects, the signal can still contain a lot of noise that are reflections of the moving object.

2.1.4 Tradeoff

Out of the methods considered, we consider radar the most promising. A radar sensor can perform detection, localization, and tracking from a single vantage point. Camera systems have the same property but offer a worse privacy trade-off. All other systems require multiple sensors to be placed throughout the playground.

In the next section, we will inspect mmWave radar systems in more detail.

2.2 mmWave radar background

mmWave radars use FMCW (Frequency modulated continuous wave) technology in the millimeter wavelength frequency band (30 - 300 GHz). The radar can measure distance, radial velocity, and azimuth angle of multiple objects in view. The sensor chosen uses the 60Ghz band, which is a free to use frequency band in the Netherlands [2] and many other countries.

Due to the architecture of the sensor, many parameters of the radar signal transmitted are software configurable. When selecting these parameters, a tradeoff needs to be made between the range and resolution of the measurements. In order to understand the output of the radar, and how to configure the radar, an understanding of its working principles is needed.

2.2.1 Operating principle

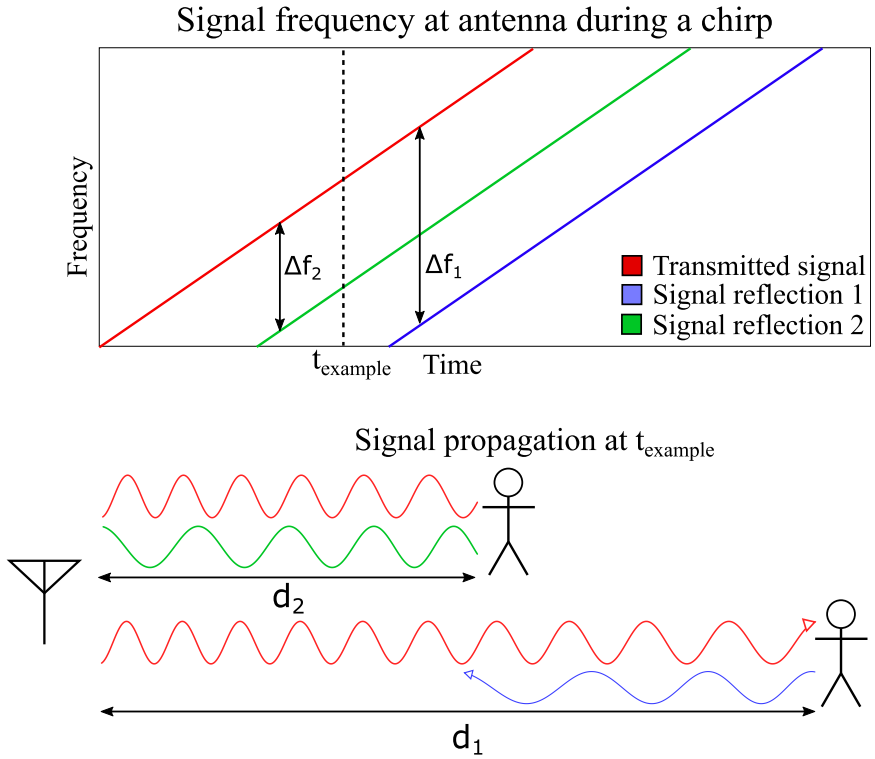


Figure 2.2: **A FMCW radar uses a chirp of increasing frequency to determine the range to a target. Due to the signal travel time, The frequency of the transmitted signal and received reflection will be shifted by an amount linearly related to the distance of the object.**

An FMCW (Frequency modulating continuous wave) radar measures distance to objects based on the time taken for a radio reflection to travel from the sensor to the object and back. The radar transmits a radio signal called a chirp, which increases in radio frequency at a constant rate S. This signal travels distance d to the detected object and back. Due to the round trip time, the received

signal will have a frequency that lags by $\Delta f = S\frac{2d}{c}$. By recording the frequency difference between the transmitted and received signal, the sensor calculates the distance.

Chirp processing

To determine the frequency difference Δf, the sensor uses a hardware circuit that subtracts the frequency currently being transmitted f_t from the received signal f_r. The circuit outputs a much lower frequency IF (intermediate frequency) signal, with a constant tone for each object detected. The frequency of these tones is related to the distance according to equation 2.1.

$$d = \frac{\Delta f \cdot c}{2S} \tag{2.1}$$

In a real-world scenario, the IF signal will be a composition of many tones from all the objects in the scene and background noise. A Fourier frequency decomposition is used to determine the frequencies present. The IF signal is digitized by sampling it with ADCs and converted to the frequency domain by a complex FFT (Fast Fourier Transform) operation. The FFT operation outputs complex numbers, with the absolute value representing the amplitude of a frequency, and the angle the starting phase of the signal.

The radar's range resolution is limited by its ability to accurately determine the frequency components of the signal, and the frequency ramp speed S. Based on Fourier theory, the maximum resolution achievable in the frequency decomposition is limited by the time duration of a chirp. Since the duration and ramp together determine the bandwidth, the range resolution is directly proportional to the transmitter's frequency bandwidth. The frequency ramp and duration can be configured for an implementation, but bandwidth is limited by the hardware and frequency usage regulations.

The phase value (ϕ) calculated by the FFT transformation, is the same as the phase shift between the transmitted and the received signal. The phase shift is caused by the distance of the object, but since the wavelength (λ) is only about 2.5 mm, this can only be used to measure small changes in distance of less than one wavelength. The phase shift is related to the distance according to equation 2.2.

$$\phi = \frac{4\pi d}{\lambda} \mod 2\pi \tag{2.2}$$

2.2.2 Calculating the velocity and angle of objects

An FMCW radar doesn't only sense the distance of objects, but also velocity and angle of arrival. It does this by sending multiple chirps from multiple antennas in one frame.

In order to estimate velocity, the sensor performs multiple chirps in quick succession in a radar frame, as seen in figure 2.3. The radar data is from each chirp is stored in a 3-dimensional data structure with the range data in the first dimension, chirps in the second, and antennas in the third dimension. This data structure called the *radar cube* is shown in figure 2.4. If an object at distance d is moving, the small change in distance in each chirp changes the phase of the complex value at the range bin corresponding to d. The values in

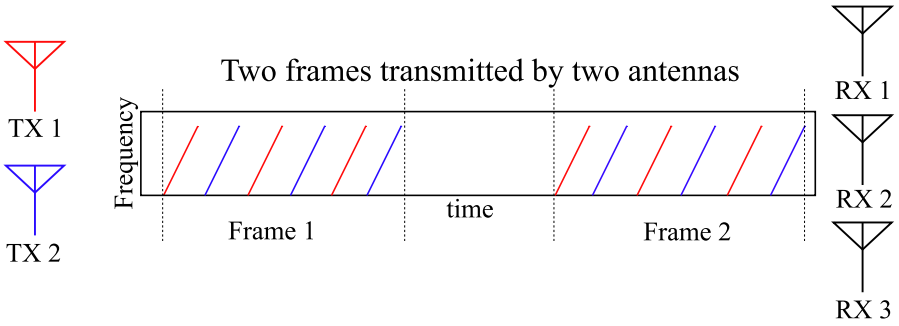


Figure 2.3: **The radar transmits multiple chirps in a frame. Multiple frames follow each other after a delay. The chirps are transmitted from multiple antennas and received by multiple antennas.**

the chirp dimension can be used as samples of a signal with its own frequency, the frequency of this signal (f_{chirps}) is related to the object's velocity according to 2.3.

$$v = \frac{\lambda f_{chirps}}{2} \tag{2.3}$$

Since there might be multiple objects at different velocities at the same distance range, we again use an FFT to determine the constituent frequencies for the entire chirp/velocity dimension of the radarcube. The duration of the frame determines the resolution of the velocity measurement. Using the phase places a limit on the maximum speed that can be detected: If an object moves more than half a wavelength between two chirps, the direction of movement will be determined incorrectly. The velocity where an object moves more than half a wavelength between two chirps is considered the maximum unambiguous velocity for the chirp configuration. Velocity estimation is used to help with the tracking of objects, and to reject static objects. The sensor can only measure radial velocity, the velocity vector moving towards or away from the sensor.

The sensor captures data from multiple antennas spaced half a wavelength apart to determine the angle of arrival. The distance from the antennas to the object will be slightly different for each antenna. Assuming that the object is far away, this results in a difference in the phase of the range-FFT output (ϕ) described by equation 2.4.

$$\Delta\phi = \frac{2\pi l \sin\theta}{\lambda} \tag{2.4}$$

Where θ is the angle of arrival. This phase difference can then be used to perform an FFT on the antenna dimension in the same way as described for the velocity measurement. The radar can have multiple transmitter antennas to increase the effective amount of antennas. Every transmitter/receiver pair forms a *virtual* antenna, improving the resolution. The transmitters have to take turns transmitting, as shown in figure 2.3.

Our sensor can measure both the Azimuth angle and the Elevation angle because one antenna is placed higher than the others. The azimuth is calculated with 12 (virtual) antennas, while the elevation has only two antennas. The azimuth angle is used to generate the radar cube, and the elevation can be calculated for specific points of interest.

The final result after processing range, velocity, and azimuth is a 3-d radar cube in these dimensions, where each position has a signal strength value associated with it (see figure 2.4). The computations are performed on a DSP (Digital signal processor). The first processing step, the range calculation, is performed on the ADC samples directly after the chirp. The FFT result is stored in a piece of memory reserved for the radar data. Once all chirps are processed, the velocity and angle calculations can be performed in between two frames.

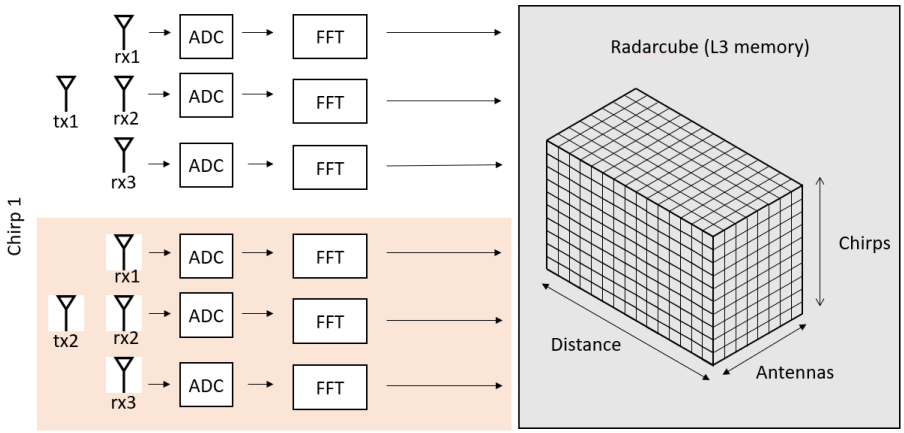


Figure 2.4: **Capturing a chirp on a system with 2 TX and 3 RX antennas. The two TX (Transmitter) antennas chirp after eachother. Every RX (Receiver) antenna receives the chirp simultaneously. The DSP performs a range FFT whose results are stored in the radar cube before the next chirp. The angle of arrival and object velocities can be calculated by performing an FFT in the antennas and chirp dimensions respectively.**

2.2.3 Pointcloud calculation

For many applications, we are most interested in the positions in the radarcube with the highest signal strength. These locations represent the location of objects in the detection area. A data structure representing only these points is a point cloud. A point cloud is a list of tuples containing a distance, azimuth angle, radial velocity, received signal strength and optionally the elevation angle. The points are selected with a peak detector, that returns positions in the radar cube with a signal to noise value exceeding a threshold. A visual example of a point cloud is given in figure 4.2.

Generating a point cloud is more efficient than calculating values for the radarcube. By performing peak detection directly after the distance FFT operation, the velocity and angle processing steps only need to be performed on the areas of interest, reducing the memory and processing requirements. The measurements from all chirps are combined before the peak finding step to improve the signal to noise ratio and to filter out stationary objects. Beamforming algorithms can be used to find the angle and velocity with a higher resolution than by detecting peaks in the radarcube [55]. These algorithms don't return

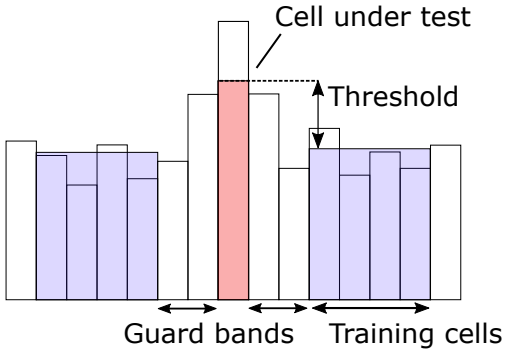


Figure 2.5: **CA-CFAR peak detector. The value of the cell under test is compared to the average value of the training cells plus a threshold value. Guard bands are excluded from the training cells to avoid the sidelobes of the peak from affecting the average. The number of training cells, guard bands and the height of the threshold can be modified.**

accurate signal strength values but have a narrower peak.

In our sensor, points are extracted using a CA-CFAR (Constant False Alarm Rate) peak detector [23]. This detector is shown in figure 2.5. Signals originating from the environment are filtered by ignoring objects with no velocity.

2.3 Object classification methodologies

In this section, we analyze the current state of the art in object detection using mmWave data. We included papers performing all types of object detection, even if the classes are different from those we are interested in, as person detection is a subset of object detection. mmWave radar data can be represented at different abstraction levels. The classification methods found use the radar cube or point cloud representations. How these are created is described in section 2.2.3. Here we describe the state of the art using both abstraction levels.

2.3.1 Classification methods based on radar cube data

The radar cube has many features similar to an image. It is a fixed size grid in multiple dimensions. The structure has locality: features belonging to the same object are close together. And the shape of an object will stay the same when it is moved in the radar cube. Because of these similarities to an image, many image classification methods are used on them.

Most of these methods have a common starting point. They start with finding an area of interest and taking a 2d slice from the radar cube, for instance, by using distance and velocity dimensions of a single azimuth bin. Effectively, this results in a regular image. They then classify the object in the slice with machine vision techniques. These techniques use manually derived features [38], automatically derived features [37] or neural networks.

The current state of the art for image classification is convolutional neural networks [8]. These networks scan the image with multiple filters to find the presence of patterns. By using consecutive layers that combine the simple patterns into more and more complex shapes the network can detect the presence of objects. During the training step, the network learns the optimal filters to use. These networks have enabled the current machine vision revolution, and are used in the current best performing radar systems. [35] uses a 2-d convolutional neural network on radar cube slices to classify objects and is the current best performing system using 2d slices.

2.3.2 Classification methods based on point clouds

Research into point cloud classification has not been limited to radar data. Instead, most research has been done with a depth camera or LIDAR data [18]. Some of these techniques have been transferred to radar point clouds.

Using a point cloud as input is problematic for traditional machine learning methods. Most methods have a fixed vector of input features where each value has a fixed meaning or with a fixed relationship between the inputs [8]. For instance, in an image, the relationship between pixels is determined by their position in the 2d grid. If all pixels in an image are randomly shuffled, it is impossible to determine the subject of the picture. But point clouds don't have a fixed amount of points, nor is there a fixed ordering that captures the relationship between the points. An ordering based on the location of the points is unstable, small changes in the location of the points will cause big changes in a 1-d ordering [10].

To make a feature vector for the classifier, we need to create a fixed set of features independent of the ordering and number of points. Several methods

have been proposed to create a fixed set of features for radar point clouds:

- Use symmetric functions to generate features. [47, 45]
- Direct feature learning from point clouds. [46, 16]
- Voxel coloring/rasterization. [58]

Symmetric functions are mathematical functions whose output are independent of the ordering and amount of inputs. They can operate on all points in the point cloud. These features are typically statistical properties such as min, max, mean, variance, et cetera. The most common method is to hand-select a list of functions and to create a classifier based on the output of these functions as features. This has been used in [47, 45]. The classifier architecture can be any type of classifier that operates on a feature vector such as SVM, decision trees, or fully connected neural networks [8].

The most influential works have been Pointnet [10] and Pointnet++ [41]. Pointnet uses a method for learning symmetrical functions to apply on the point cloud while training the network. The pointnet architecture applies a multilayer perceptron network to each point in the point cloud. This network maps the 4 dimensions of a point to a much larger embedding space. It uses the same weights for each point. It then applies a symmetrical function featurewise to all outputs, creating a single 1d feature vector. The authors use the max function but show that any symmetrical function can be used if the multilayer perceptron and embedding space is large enough. This results in a single feature vector representing the entire point cloud. After generating the final feature vector, a multilayer perceptron performs the final classification. Pointnet++ hierarchically applies the feature vector generation process to groups of nearby points (based on spatial distance) and repeats this process on a larger distance scale on the outputs until a single feature vector remains.

The authors of [46, 16] transfer the technique of pointnet++ to radar point clouds for use in automotive applications. The authors did not release their code, but since the pointnet++ source code is available online, we tried to replicate the results found in [46]. By replicating the steps described in the paper, using our data, we found that the network overfits to our data. We expect that our dataset was too small for the size of the network. And that our relatively sparse point cloud does not have many shape features. The network size is also far beyond what can be executed on an embedded platform, requiring a GPU with gigabytes of memory.

Voxel coloring has been used in mID [58] for unique person identification using a mmWave sensor. A fixed-size bounding box is placed around a tracked object, and voxels are colored when a point from the point cloud is inside the voxel. This creates a rough outline of the person. The voxels are flattened using a fixed ordering, and an LSTM network identifies the person by using multiple frames. The authors show that this works at a range of up to 5 meters.

2.3.3 Classification methods using both point and radar cube data

In [33], the authors combine both point clouds and radar cube data. The algorithm selects a smaller 3-D section of the radar cube of 5 range bins, 5 width

bins, and 32 velocity bins for each point in the point cloud. A series of 3d and 1d convolutional and pooling layers extract features from the section. These features describe the velocity distribution of the radar cube around the point. These features are combined with the features from the point cloud tuple (location, velocity, RCS) in a fully connected neural net to calculate output scores. Finally, the points are clustered, taking into account the class predictions, to arrive at the final classification. The authors show it outperforms [38] which is discussed above, and [47] which uses only point clouds. The authors use a high-end automotive radar sensor to generate the radar cube and point cloud. A high-end PC with an Nvidia TITAN V GPU is used to perform the predictions in 0.04 seconds per frame. We consider this method state of the art for radar-based object detection without resource constraints, but out of range for our power budget.

2.3.4 Takeaways

Most of the previous work in object classification with mmWave radars have been in the context of road user detection for cars. The best results in this field have been achieved with high-resolution radars and complex models executed on powerful hardware [33, 46]. Our contribution will be to develop a system that classifies using limited resources, using the properties of static surveillance to our advantage: Our sensor is static, and people stay in the field of view of the sensor for a time in the order of minutes rather than seconds. In [58], we have seen the use of an LSTM network to combine sparse data from multiple frames into better features. We plan to use such a technique to improve our prediction when using sparse or low-resolution data. While using the radar cube offers more information and the best classification results, it also has higher computational requirements than using the point cloud. It might not be possible to use the radar cube with our hardware. Using point cloud features is a computationally efficient way to use point clouds for classification.

Chapter 3

System design

In this chapter, we describe the implementation of our measurement and data capture system. We describe the used sensor and its configuration in section 3.1. To process the radar data, we wrote software tools that can store, visualize, and modify the radar data. These tools are described in section 3.2. Our method of data collection, and what datasets we used for the design of our classifier is described in section 3.3.

We performed a conceptual design of a standalone capture system to be used in playgrounds to show that this system can operate without external power, use wireless connectivity, and with costs below the limits set in the requirements in section 1.1. This is described in section 5.1.

3.1 Sensor hardware

We have chosen to design the system around a Texas Instruments mmWave sensor for the reason that the sensors in this series have both the analog RF circuitry and a digital signal processor integrated into one Integrated Circuit [53]. The integration of these components makes it easier and less expensive to design a PCB compared to other systems using separate parts.

We evaluated two sensors ICs: the Texas instruments IWR1642 and IWR6843. Both devices have the same basic layout shown in figure 3.1, but differ in the used frequency band and number of antennas. The IWR1642 uses the 76 GHz band, which is license-free for use in automotive and traffic control applications. Our application does not fall under that definition. Furthermore, we found in testing that the IWR1642 had an effective range for detecting persons of only 16m, and could only separate people in azimuth at least 2m apart. During the project, Texas instruments released the IWR6843 for evaluation, which improves on these aspects. It uses the 60 Ghz band, which is free to use at less than 10 dBm transmit power and 13 dBm/MHz spectral density [2]. The IWR6843 has an additional TX antenna, increasing the azimuth separation ability by 50%. And the range was measured at over 50 m.

To get started quickly, we used the IWR6843ISK dev kit with onboard antennas. The IWR6843ISK also can measure the elevation of a radar return by having one antenna placed higher than the others, as seen in figure 3.2. The configuration of the antennas results in a field of view of 110° azimuth and 44°

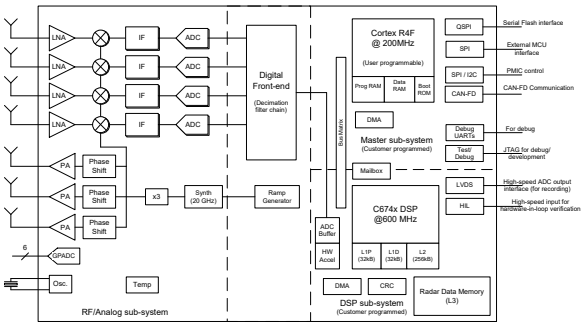


Figure 3.1: **Overview of the IWR6843 components [53].**

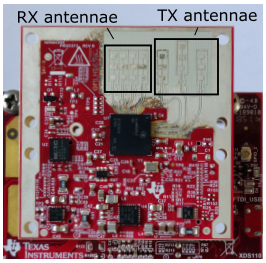


Figure 3.2: **IWR6843ISK dev kit. It has 3 transmission (TX) and 4 receiver (RX) antennas.**

elevation.

3.1.1 Radar configuration

To process the ADC values to point clouds, we use a code example by Texas Instruments [1]. It processes the raw data into point clouds on the DSP as described in chapters 2.2. The processing chain is parameterized, meaning we can configure the radar for our use case.

We configured the radar chirp to have an unambiguous range of 43.7 m and a maximum velocity of 21.6 m/s with a range resolution of 0.085 m and a doppler resolution of 0.9 m/s. The range is selected to reach an effective 30m range. The additional range is to prevent objects further way to show up as *ghost* images in the area due to the range unambiguity. The velocity limit is chosen to reflect the maximum speed of fast bicyclists and mopeds inside cities.

The device can separate up to 3 returns in azimuth and measure the elevation of a return, but not separate multiple returns in height. Thus, two objects at the same range and velocity, but at a different azimuth can be detected separately. But two objects above each other at the same distance, velocity and azimuth can't be detected separately, and instead an average value will be resolved.

The parameters for this configuration are a chirp that starts at 60 GHz and sweeps 1.7 GHz at a slope of 42.8 MHz/µs. This results in a chirp length of 40 us. Every frame consists of 48 chirps that are alternatingly sent by the TX antennas. 10 frames are recorded per second.

The parameters for the CFAR peak detection have been tuned to result in a high number of points for one person by increasing the number of sidebands and lowering the threshold. The parameters were determined experimentally to an averaging window of 8 bins, 4 bin guard bands, and a threshold of 9 dB.

To be able to track and classify the objects in the scene, the individual radar reflections are clustered into objects. We use the DBSCAN [14] clustering algorithm. DBSCAN creates clusters in two steps: First, it searches for a core point that is surrounded by a minimum number of other close points. Then, it

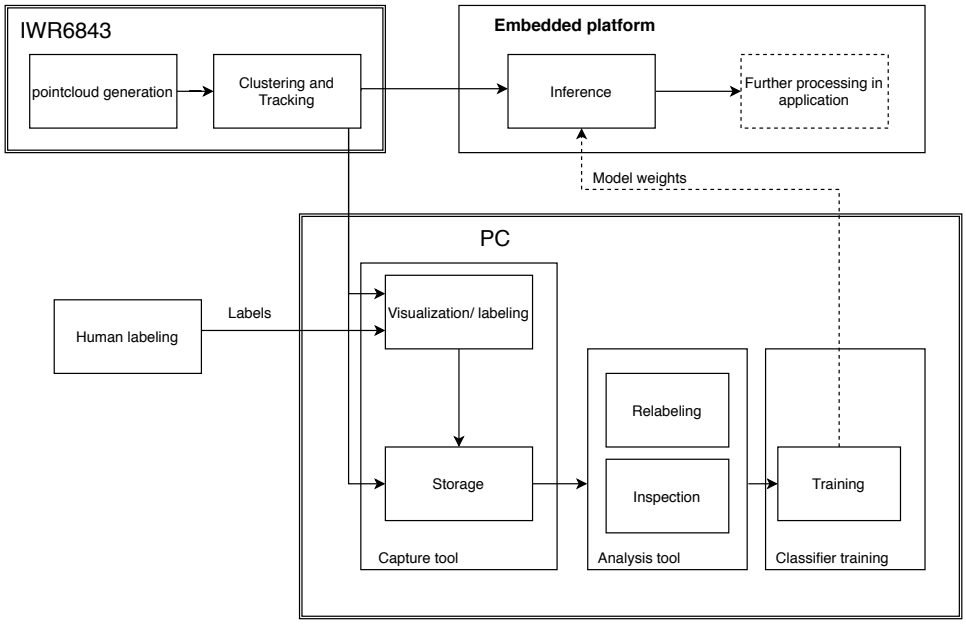


Figure 3.3: **A schematic overview of the test setup.**

recursively adds nearby points within a distance threshold. A cluster is made when the cluster's size and density exceeds a threshold value.

In order to track objects over time, a location prediction is made for each point by assuming a constant-acceleration. The location prediction is used as the core point for the new cluster. Texas instruments includes an implementation of DBSCAN with Kalman filter motion prediction. Rather than writing a new implementation, we choose to use this system and only tweak its parameters. To tune the clustering algorithm, a repeatable process to evaluate the performance is required. We modified and compiled the algorithm for use on a desktop pc and iterated on the parameters on prerecorded radar data. We tuned it to be slightly oversensitive, creating too many clusters, with a post-processing step that filters out false positives. The updated parameters are loaded onto the embedded hardware.

3.2 Tools and Processing

To use the radar output, we need a method of capturing, processing, storing, and visualizing the data. The tools need to store the radar point clouds together with ground truth labels so that they can be used to train the classifier. The overall structure of how these tools work in the complete system is depicted in figure 3.3. We developed two applications on a shared base: A capture application, and an analysis application. We used a separate script for model training.

The capture application connects to the radar, configures it, and captures the radar data. It then visualizes the point clouds in 3d, shows the clusters they are assigned to, and shows the location and classes of the detected clusters.

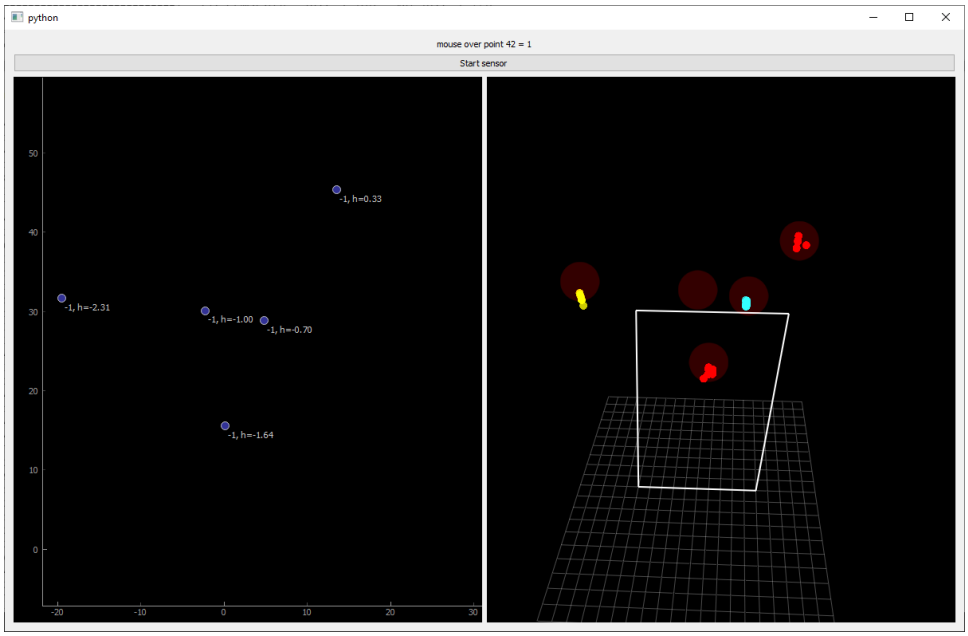


Figure 3.4: **The monitoring and labeling application.**

The user can label the clusters with a class in real-time by using the mouse and using keyboard shortcuts. Since the clustering algorithm does not always cluster correctly, we have given the user the option to correct these mistakes by breaking clusters up into sections or labeling them as background noise to be ignored. When the cluster is no longer tracked, a trace containing the entire history of that cluster is stored on the hard drive. The application also saves the raw radar data and can replay these raw data files. This has proven useful when iterating on configurations.

The analysis tool can playback the point cloud traces to inspect the data and labels visually. The user can "scroll" through the data to find a specific point of interest. The user can update the labels by comparing them to a reference video. The analysis application is shown in figure 3.4.

The capture application is written in Python. The USB connection to the sensor exposes two virtual UART ports for commands and data. The application configures the sensor with text commands over the command port and receives binary data via the data port.

The GUI is created using the QT framework [54]. QT's Asynchronous processing support is used for simultaneous data capture and presentation. The application is split up into asynchronous parts that communicate with signals: Parsing, visualizing, storage and classification.

The radar data arrives in the form of a binary data structure. The data structure arrives in packages, when a complete package is received, it is parsed using the Construct library [5], which parses a raw binary stream from a data structure definition. Point clouds are shown in a GPU accelerated 3d view using Pyqtgraph [40]. Points are given colors to signify their cluster. Users can pan, rotate, and zoom the viewing area to inspect the point cloud. Labels, point

cloud data, a position trace, the creation/deletion time and other metadata of a cluster is stored together in a *point of interest* data structure. When no information for a cluster is received for 5 seconds, the data is stored to disk. The data is written in the msgpack format, a format similar to the commonly used JSON, but which stores data in a binary format to be more storage efficient. We run the classifier in a separate thread to give live output in the visualization.

Training the classifier happens offline using the msgpack files stored by the capture application. We use scikit-learn, a machine learning library, to implement the single-frame machine learning algorithms. For the neural network algorithms, we use Tensorflow [15]. Tensorflow models can be exported to C code for inference on embedded systems using Tensorflow lite. To train these models, we used a server with a GPU using Google Colab.

The tool source code is available on GitHub: `https://github.com/BBekker/mmWavePOI`

3.3 Data collection

The classification system requires a large training dataset to learn to differentiate the classes. Our classifier is a supervised learning architecture. Meaning that it learns how to identify a class based on a lot of examples. Every example has both the recorded data, and a label of the correct class. To achieve the best performance, the dataset should contain examples of all the situations that are expected when deployed. We collected datasets for training and validation on location at playgrounds and other public locations. For this, we used the IWR6843 sensor mounted on a tripod and powered by a battery. To compensate for differences in sensor location, the sensor height and angle were measured when deploying the sensor. Points are mapped from the radar reference frame to a global reference frame, taking into account the sensor height and angle. To be able to perform data collection on location, the data capture method needs to fulfill radio emission regulations. We chose the IWR6843 because it uses a free to use frequency band. The IWR6843ISK dev kit has been certified to fulfill the radio emission requirements for limited deployment.

3.3.1 Datasets

We collected 5 datasets at the TU Delft campus and a school playground. Datasets that include children, were collected in a closed playground with small group of children whose parents were informed and consented. These datasets contain 4 different adults and 5 different children. Data collected at the TU Delft contains many different Adults and Bicyclists.

Dataset 1: Adults and children on a school playground without giving them any instruction on what action to perform, other than to stay in the roughly 20m x 30m area. This most closely mimics a real-world scenario and includes behavioral features.

Dataset 2: Adults and children playing together in a small area.

Dataset 3: Adults and children stand at the exact same location 5m in front of the sensor while moving in place.

Dataset 4: Bicyclists moving in a zig-zag pattern inside a 20m x 30m area.

Dataset 5: People walking and bicycling at the TU Delft campus. This dataset consists of over a hundred bicyclists and pedestrians but no children.

When analyzing the datasets one big issue stood out: In the first dataset of people acting without instructions we found that peoples behavior is influenced by the physical design of the area. Children will play near the play equipment, while adults stand on the side and bicyclists only move in open spaces. The classifiers will learn where on the playground each class is most common and uses this information in the classification. But this does not generalize to other playgrounds with a different layout. To overcome this issue we need to either test at a lot of different playgrounds, dont use the location information in the classifier, or create datasets that force the classes to move everywhere. Since we cant test at a large amount of playgrounds due to required permissions and time constraints, and we want to preserve the potential benefits of including location data, we restricted dataset 2 to a smaller 10m x 10m area where subjects are asked to use the entire space equally. The high density of use reduces the clustering algorithm's performance, resulting in worse input for the classifier.

Dataset three removes all variations in location and clustering quality, guaranteeing a high signal to noise ratio and no features based on location or movement. But, these datasets may not generalize to other activities or locations. Therefore, we only use this dataset for validation. Dataset four is added to increase the amount of bicyclist data, in the same conditions as dataset one, two and three. Dataset five is a large dataset with many different people, but does not exactly capture the type of users and activities we are interested in. This dataset is only used in training.

Data split

The captured data is combined and split into a training and classification dataset. The training dataset is used to train the classifier, and the validation set is used to evaluate the classifier's performance. The typical way of creating a machine learning experiment would be to shuffle the dataset and split based on a fixed ratio [8]. Multiple splits can be used to perform cross-validation [8]. However, we found that in our situation we achieved an unreasonably high accuracy due to data leaking [26]. Because ten samples are taken every second, two samples taken directly after each other will be very similar. If the dataset is split randomly, these very similar samples will be in both the training and validation dataset.

To prevent this issue, our datasets are recorded separately. Every dataset mentioned earlier consists of multiple captures, and we use some captures for training and some for validation. We aimed for a 9:1 split between training and validation data. The amount of data in our training and validation datasets is shown in table 3.1.

3.4 Playground sensing sytem hardware design

Testing and training the classifier uses a laptop to process the data. But, when deployed in playgrounds, the system needs to be standalone. To show that this

Dataset	Adult	Child	Bicycle
Training	38 701	37 573	15 714
Validation	3 903	4 063	1 355

Table 3.1: **Number of individual pointclouds captured for each class. The total dataset represents 2.7 Hours of data.**

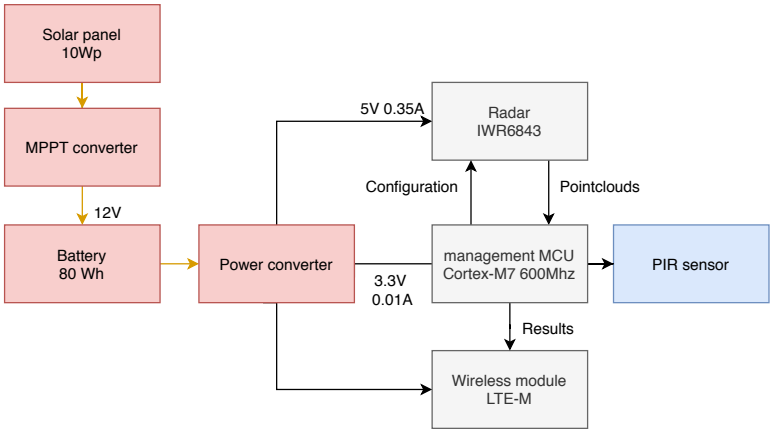


Figure 3.5: **Schematic design of the inference system that can be deployed at playgrounds.**

is possible, we conceptually designed the electrical system of the playground sensing system. In order to fit our requirements, the system should be able to operate without an external power supply, should be affordable and small enough to be easily deployed.

To supply our system with power, we can use a solar panel. But, to keep the system reasonably sized and easy to deploy, we should strive to have the smallest possible solar panel. To determine the required solar panel, we analyzed the energy usage requirements of the system.

The radar sensor is measured to use 1.7 W when operational. To reduce the energy consumption, a PIR sensor can be used that only enables the radar when movement is detected in the area. Any processing will need to happen on a device with either a very low power sleep state or fast power on, such that it can be kept in a low power state when not active. To achieve this, we use a cortex-m7 microprocessor with a <100 mW power usage when active and <1 mW when in standby. After the data is processed, the location and classification data is sent to a server periodically. Transmitting x/y location and class every second requires approximately 15KB per person hour. This usage pattern fits best with LTE-M. LTE-M can transmit at up to 375Kbps when active while having a standby current of less than 20 µA [24]. This means that the data can be transmitted once per hour with insignificant additional power usage. The total power usage is a maximum of 1.8 Watt when active and <1 mW when in standby.

To provide the required energy, we choose a 10 Wp (Watt-peak) solar panel as the power source. The solar panel has a size of 355 mm × 255 mm [3]. The

amount of energy provided by the solar panel will change during the year with the seasons. The energy generated when placed in Rotterdam is estimated using PVGIS [22]. It generates 10 Wh per day during the winter and 40 Wh per day during the summer. An 40 Wh battery can be used to buffer energy during nights and periods of low energy.

This enables 5 hours of active operation during the winter and 20 hours during the summer. We expect that in most cases, this should be sufficient energy. As during the dark winter nights, we also expect less activity at the playgrounds. But in a high traffic location, a larger solar panel can be used.

The antennas are placed 2m above the ground surface with a slight tilt downwards. A sensor placed too low to the ground is at risk of being completely blocked by the body of a nearby person, while a sensor too far above head height has a blindspot nearby. The sensor can be placed on a pole or attached to an existing pole, such as a streetlight. The casing can be opaque in the visual spectrum but should have low attenuation and minimal refraction of radar signals to not affect the measurement. Non-polar polymers are a suitable material, for instance polypropylene [4].

Cost estimate

As stated in our requirements in the introduction, the system needs to be affordable for broad adoption. We set a limit of €2000 material costs per device. In table 3.2 we show the costs for the electrical components at €439.70, leaving over €1500.- for the mechanical components.

Component	Part number	Cost
Sensor module	IWR6843ISK	€125.-
Solar panel	Phaesun sun plus 10	€36.40
Battery	Enerpower 12V 4.5Ah - LiFePo4	€52.95
Solar charge controller	estimate	€50.-
LTE-M Modem	XB3-C-A2-UT-001	€64.35
Microcontroller	NXP IMXRT1064	€11.00
PCB and power regulation	estimate	€100.-
	total	€439.70

Table 3.2: **Cost estimate for the electrical parts in the sensor design.**

Chapter 4

Classification

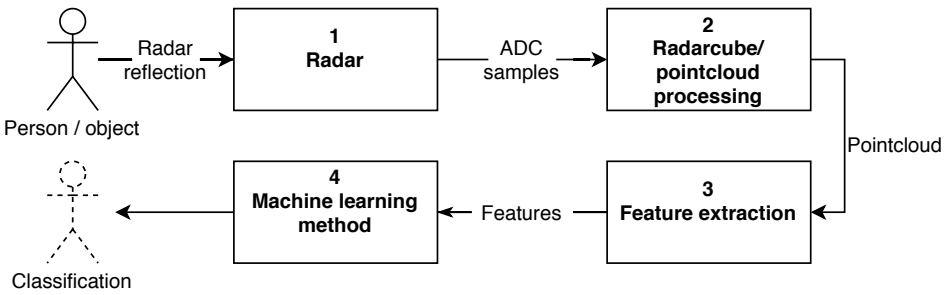


Figure 4.1: **Schematic overview of the steps taken to classify a person.**

In this chapter, we explain how we classify the three classes we are interested in: adults, children, and bicyclists. We do this by detecting the unique differences in the radar output for each class.

The processing pipeline has multiple steps to go from the person to a classification, as shown in figure 4.1. To determine how we can differentiate between the classes, we will first consider what measurable physical differences exist between the classes, and how they could affect the radar reflection. This is reported in section 4.1. Based on these differences, we set out to design a classifier. The first step is to determine what data representation we can use from our radar. We decide to use point cloud output, the reasoning for this choice is explained in section 4.2.

We analyzed multiple methods for classification using pointclouds in chapter 2.3. Based on our hardware limitations, we chose to use a feature based method. These features have to describe the physical differences identified before. We review multiple features in section 4.3, and decide on a feature vector to describe the point cloud. Finally, a machine learning classifier uses the feature vector to make a classification. We created classifiers based on multiple machine learning methods and evaluated their performance in section 4.4.

4.1 Identification hypothesis

We hypothesize that we can detect if someone is a child, adult, or bicyclist of any age, based on the physical and behavioral differences between these classes. There is one exception: there is no perfect way to define a hard cutoff between adolescent children and adults based on apearance. Therefor, our child class encompasses prepuberecent children, and the adult class is post-puberty. For adolescents we accept both classes, and they are not included in our datasets. The physical differences have to result in differences in the radar reflection for them to be measurable. We have identified four properties that can be measured:

Height Height is the main difference between adults and children. An adult will be taller than a child, but bicyclists might not be recognizable based on height. Taller people will have a larger extent and higher maximum height of the pointcloud. We can calculate the height of a radar reflection by converting the measured elevation and distance to cartesian cordinates.

Radar cross-section The radar cross-section (RCS) is the amount of radio energy reflected by an object. The RCS scales with the size of the object, but also changes due to the different reflectivities of materials. An adult should have a larger radar cross-section than a child. And a bicyclist on a metal bike should also have a significantly larger radar cross-section than a person walking.

Limb movement An adult, child, or bicyclist will have periodic limb movement when moving. The frequency of this movement is related to the limb length and the movement speed. We hypothesize that a child's limb movements will oscillate at a higher frequency than an adult to achieve the same body velocity. And the limb movements of a bicyclist will be completely different. These oscillations might be visible in doppler, position, and even signal strength measurements. Oscillations in the doppler dimension are often used for classification with CW radar [9, 42, 27].

Macro movement We expect that the different classes will have different movement patterns. For instance, a bicyclist moving more quickly and children having more erratic movements than adults. These large scale movement patterns can give behavioral clues to the subjects class.

4.2 Radar data representation

To classify our subjects, we need to extract the information from the radar reflection that represents the features of our three classes. There are two levels of abstraction that we can use as inputs for our classifier: either the radar cube or a point cloud.

As described in chapter 2.2, the radar cube is a direct representation of FMCW radar data. The radar cube represents all data that was captured by the ADC's, converted into distance, velocity, and azimuth dimensions. The point cloud does not represent all data but instead is a sparse representation of only the reflections with the highest signal to noise ratio. Because the radar

cube contains more information than a point cloud, it is the preferred representation to use for classification, but it might not be possible to generate it on our hardware. We face three hardware limitations to using the radarcube:

- The available RAM on our device.
- The available processing time for each frame.
- The data transfer rate to the next stage of processing.

The IWR6843 has 768KB RAM for radar data. The ADC captures 512 $samples*$ 12 $antennas * 24\, chirps * 4\, bytes = 576KB$ of data each frame. A radar cube using 16-bit RSSI values would use an additional 288KB, exceeding the available memory. In order to send this data at 10hz, an average data transfer rate of 23Mbps is required. Moreover, since the data cannot be double-buffered, it needs to be sent in bursts after the radar cube is calculated. The IWR6843 has a high-speed LVDS peripheral for this use case, but LVDS is generally not supported on microcontrollers or mobile processors. In conclusion, calculating and using the whole radar cube is not a feasible path, given our hardware constraints.

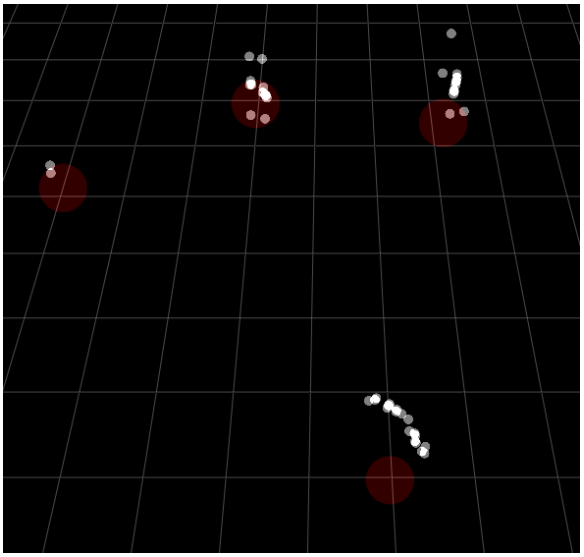


Figure 4.2: **Point clouds of four children playing. The grid size is 1 meter. Brightness represents signal strength, velocity is not shown.**

Instead of using the radar cube we have to use point clouds. We can configure the radar to generate a point cloud with more details. When only using the point cloud for localization, we might configure the CFAR algorithm to return only a single point. But because our goal is to use information about the shape and movement of individual body parts, we have configured the CFAR algorithm to return multiple points for one person. The points represent different spots on the person's body. An example of these point clouds can be seen in figure 4.2. The number of points depends on the distance, and if other people are nearby. The distribution of the number of points in one sample of a person in our dataset can be seen in figure 4.3.

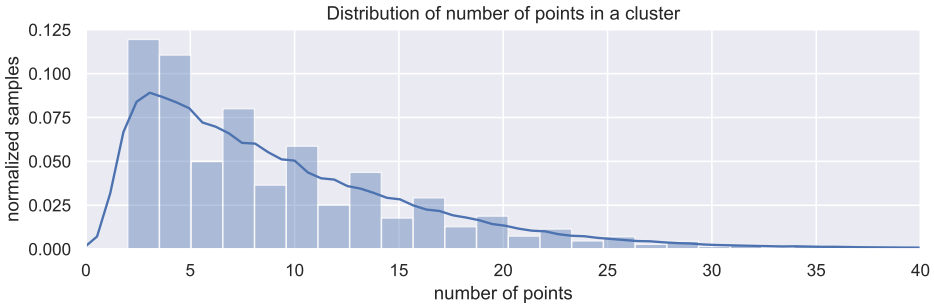


Figure 4.3: **Distribution of number of points in a pointcloud. Most pointclouds used for classification are between 5-10 points.**

A hybrid solution would be to use CFAR processing and clustering to detect areas of interest and generate a small radar cube around the area of interest from the ADC samples. Texas instruments does not support this method at this time. We explored the possibility of implementing this ourselves but concluded that the amount of mathematics and engineering effort required was out of scope for this project.

4.3 Feature selection

Description	symbol	Importance
Mean distance	μ_d	13%
Mean azimuth angle	μ_ϕ	16%
Mean radial velocity	μ_v	6.4%
Mean height	μ_z	3.2%
Mean SNR	μ_{SNR}	2.0%
std x location	σ_x	7.9%
std y location	σ_y	9.6%
std radial velocity	σ_v	2.1%
std SNR	σ_{SNR}	1.3%
var distance	σ_d^2	11%
var azimuth	σ_ϕ^2	2.8%
var radial velocity	σ_v^2	2.4%
var SNR	σ_{SNR}^2	0%
95th percentile Height	$P^{95}(z)$	8%
5th percentile Height	$P^5(z)$	3%
RCS estimate	RCS	2%
sum SNR	$\sum SNR$	3%
Number of points	$\#Points$	5%

Table 4.1: **Evaluated features. Selected features in bold. The importance column is the normalized importance in a trained random forrest classifier. This may not reflect the importance in the final model.**

We created a list of features that might have some importance based on the physical differences we identified in section 4.1 and our literature study. The evaluated features are listed in table 4.1.

As described in section 2.3.2, the features derived from a point cloud must be based on symmetrical functions. Symmetrical functions do not depend on the order of inputs and operate on any number of inputs. Before applying the symmetrical functions, we can apply transformations on the individual point tuples, for instance, to convert the polar coordinates to cartesian cordinates.

The symmetrical functions we use are mean, standard deviation, variance, and percentile. Mean values are important for relationships between variables. For instance, distance and RCS.

Standard deviation and variance show the size of the object in one dimension for the x and y location. We opted not to use the total extend because all our classes are narrow compared to the resolution of the sensor. If the measurements have Gaussian noise, the expected measured total extent will be larger when increasing the number of samples due to the outliers, but the variance and standard deviation stay the same.

We use variance in the velocity dimension to detect limb movement. During a walking cycle, the left and right arms and legs move in opposite directions in a cyclical pattern. This results in a maximum amount of variance when the limbs are moving at their highest speed, and a low variance when slowing down at their maximum extent.

We evaluated features by testing if they are significantly different for each class, the importance assigned in a classifier, and by the effect on the network accuracy when the feature is included.

We found that the height of a person to be a robust feature. To determine the height of the person, we use the height of the individual point cloud points. Because the height of the points is influenced by random noise, using the maximum height of all points in a pointcloud is very noisy. The density of a point cloud for a person matches the distribution of a person's body mass. Using the 95th percentile height of the individual points is a good tradeoff between resistance to outlier points and maximizing the difference in height. The 95th percentile returns the $n * 0.95$th value in a vector of n values that are sorted from lowest to highest. When $n * 0.95$ is not an integer number, the result is interpolated. In figure 4.4, we show boxplots of the 95th percentile of the height of each point in a point cloud. These boxplots show that there is a clear separation between the heights of the classes. When multiple recorded point clouds of the same person are combined, the 95th percentile becomes more accurate, increasing the predictive power.

Counterintuitively, we found very little separation of the SNR values of children and adults. Due to the difference in size between the classes, we expected a difference in signal strength. Even after compensating for distance, angle, and velocity, we can't create a good RCS estimate from the SNR, and can't separate the two classes based on SNR. An example can be seen in figure 4.5. We hypothesize that this is because our point cloud returns only small peaks inside the reflection of a person, while the true RCS of a person should be calculated by taking the sum of the entire reflection. We did see a higher SNR for bicyclists.

For limb movement, we could not find a noticeable signal in any real-world datasets. We plotted the mean and variance of all dimensions in the point cloud

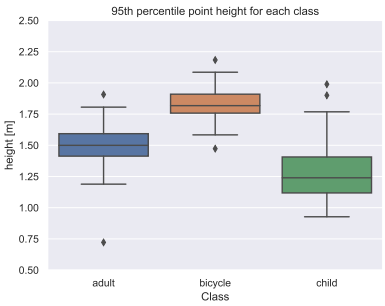


Figure 4.4: **Boxplot of the 95th percentile height of the point-cloud for clusters of each class. There is a clear difference in the height of each class.**

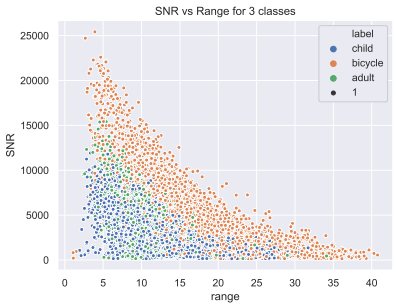


Figure 4.5: **The SNR of a cluster vs distance from sensor. To avoid differences due to azimuth and velocity, we only show detections straight ahead and over 1m/s velocity. There is a clear difference for the bicyclist vs other classes. But not between adults and children.**

over time but could not find an oscillating signal at the frequencies that would relate to walking or bicycling.

As an initial test of the predictive performance of these features in a classifier, the features are used to train a random forest classifier, and the feature importance is determined using impurity decrease [51]. This technique is not perfect, as it overestimates the performance of highly variable features and can underestimate the importance of features when two features always change together [51]. Therefore, it was only used as a guideline. The final set of features was determined by recursively leaving out features to find the optimal set. By this process, we arrived at the features selected in bold in table 4.1.

We found that we can improve many features by grouping multiple, consecutive point clouds in a larger point cloud. Thus increasing the number of points, and improving the estimation of the statistics that stay constant over time. However, combining too many point clouds removes the descriptive power of features that change over time, for instance, the mean x and y location becomes meaningless when a child has visited every point in the playground. And we lose the ability to use some of the higher frequency time-varying features. However, classifiers that work on a single frame are not able to use those features anyway.

4.4 Classifier design

We consider multiple classifier designs that use our selected features to predict the class of a subject. The classifiers are rated on multiple criteria. The first being the overall classification performance in terms of accuracy. Accuracy is defined as the percentage of correct classifications in the complete dataset. The secondary criteria is the required computational resources of the classifier. Computational resources are the required storage memory, random access memory

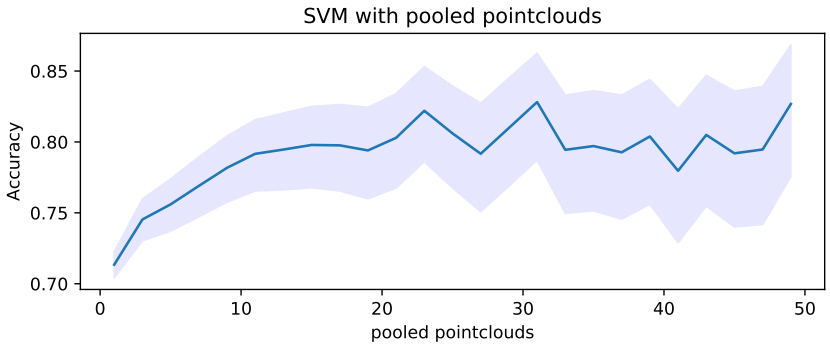


Figure 4.6: **Accuracy of the SVM classifier for different amounts of merged pointclouds.**

and processor operations. Using more of these resources make the classifier less energy-efficient, increasing the size and cost of the overall system to be deployed in playgrounds.

First, we consider classifiers that only use a single feature vector. Then we consider methods of using information from multiple feature vectors over time by averaging results and using recurrent neural networks. Finally, we summarize the results of the classifier with the best performance.

4.4.1 Clasification methods using a single feature vector

We evaluated SVM, random forest, and fully connected neural network classifiers for single feature vectors. All evaluations were made using implementations from the *Scikit-learn* python library [36]. Default values are used unless listed. For all methods, we use features derived from 10 combined point clouds. We found that merging point clouds was required for all methods to achieve good results, but using more than 10 point clouds had diminishing returns. A comparison can be seen in figure 4.6.

Support Vector Machine

We used support vector machines (SVM) [8] as a baseline machine learning method. Classifications are made by separating the feature space with a multidimensional plane that separates the classes. The plane has two parameters, a weights vector w and bias vector b. The hyperplane is defined by the set of points x where:

$$\vec{wx} - \vec{b} = 0$$

If we substitute x for our input vector, the sign of the result is the classification. Since our problem is a multiclass classification, we construct a classifier for each combination of classes and add all results for the final classification. An SVM classifier can only classify linearly separable classes, but our dataset is not linearly separable. The solution to this is called the *kernel trick*, which replaces the dot product with a nonlinear kernel [8]. We use an RBF kernel, which projects our feature vector onto an infinite-dimensional space, allowing it to form to any shape. The training process finds a set of weights that result in the

highest margin between the hyperplane and the training samples. The model can be tuned by the hyperparameters C, which controls how much outliers affect the training, and γ, which limits the effect of the RBF kernel. In practice, a low C and γ generalize better, while higher values fit the training data more closely. We use $C = 1$ and $\gamma = 0.1$.

Before classifying, we standardize our input data by scaling and adding a bias to each feature such that the mean of the distribution is zero, and the variance is 1. Without scaling, features with larger values have a higher importance, resulting in reduced performance. The scaling factors are based on the training dataset and are also applied to the validation data. We found that using normalization improved the performance of the classifier from 74% to 79%. The resulting confusion matrix is shown in figure 4.7.

SVM's have considerable secondary benefits as classifiers. Calculating the classification requires the least cycles of the techniques discussed here, and the model only requires to store w, b, γ, and the formula itself in memory.

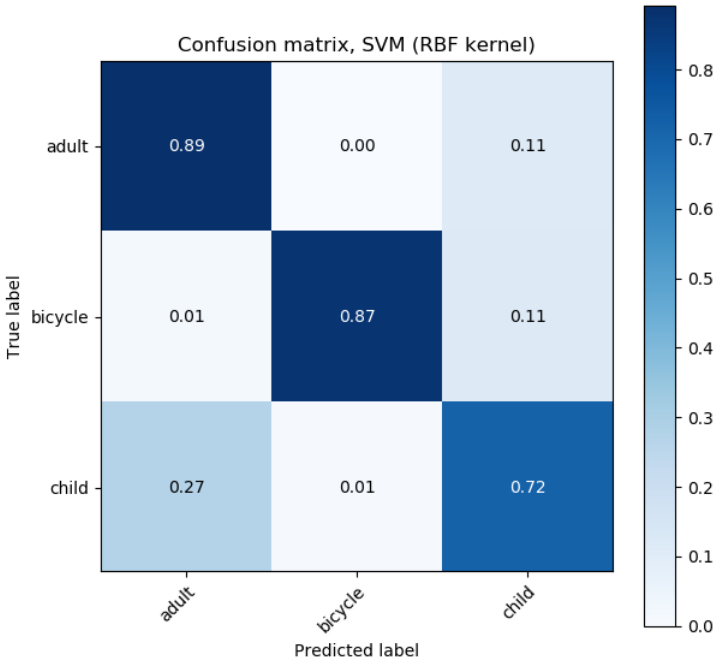


Figure 4.7: **The confusion matrix for our SVM classifier.**

Random Forrest Classifier

In a decision tree-based system, every node of the tree examins a feature of the feature set and splits the data based on a threshold value. The final leaf of the tree gives a classification to the feature vector. The random forest classifier builds many trees by adding a small random variation to the training data for each tree [7]. The final result is calculated by averaging the results from all trees. This has been shown to improve the generalization of the decision tree.

We've seen in [38] that these models can be very successful with radar point cloud features.

We use scikit-learn's "RandomForestClassifier". New branches are made when they exceed an information gain threshold. The threshold is experimentally chosen at 0.01, and we generate 50 random trees. The resulting confusion matrix is shown in figure 4.8.

In order to make a classification, all trees need to be followed, and the results averaged. To store all trees, we require $trees * nodes * size\,of\,node$ bytes of memory. Our model has a total of 1388 nodes, using at least 22KB of memory. The performance of the classifier is slightly lower than that of the random forrest classifier, with much higher computational requirements.

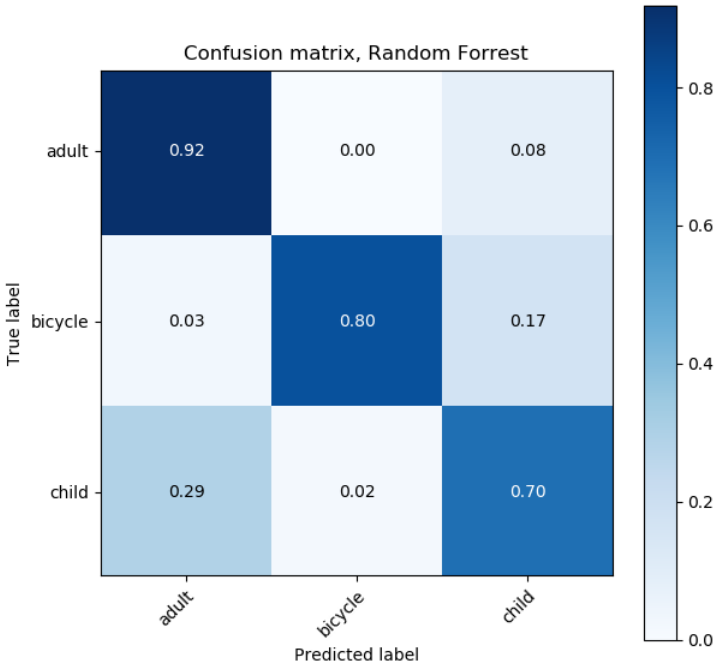


Figure 4.8: **The confusion matrix for our Random Forrest classifier.**

Fully connected neural net

Fully connected neural nets, also refered to as dense neural nets or multilayer perceptrons, consist of multiple layers of neurons, with connections between all neurons in adjecent layers [8]. The mathematical transformation in each layer is:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Where $\mathbf{x}$ is the input vector, $\mathbf{W}$ and $\mathbf{b}$ are weights and biases and $f()$ is a nonlinear activation function.

Our design uses two hidden layers of size 50. The hidden layers use the ReLu activation function. The output layers use softmax activation [17], which takes

the exponent of each output and scales them equally such that they sum to one. The model loss is evaluated using categorical cross-entropy [17]. The network is trained using the Adam [28] optimizer.

To limit overfitting, we used early stopping and L2 regularisation [8]. L2 regularisation adds a cost in the loss function to high weights and bias values. This tends to give a less complex model that generalizes better. Early stopping stops the training when performance on the validation set no longer improves.

We used standardisation of the input in the same way as with SVM and random forrest. We found that adding an additional layer to the network without a standardisation step had a similar performance, but since a neural network layer is a matrix operation, and standardisation a vector operation, using standardisation is more resource efficient. The result is shown in figure 4.9.

We found that the fully connected neural network achieved high accuracy, but did not transfer well to samples outside the distribution of the training set. For instance, when some background clutter gets clustered with a person's point cloud, the classifier will give a high confidence, but possibly wrong, prediction. This is known to be especially problematic for neural networks using ReLu activation functions [20]. We tried to avoid this by including an additional "clutter" class, trained on radar reflections of anything outside our classes, and adding randomly generated feature vectors, but neither resulted in a statistically significant improvement.

To store our model in memory, we need to store the code for the model and parameters. For the first layer, $50 * 9 + 50$ parameters are required, $50 * 50 + 50$ for the second, and $50 * 3 + 3$ for the output layer. Assuming 32-bit float values, this requires 12.5KB of memory.

Of single feature vector methods, the SVM network performs best on our primary and secondary criteria of accuracy and network size.

4.4.2 Classification methods using multiple feature vectors

All methods presented until now only use the features from a single point cloud to perform a classification. However, we record the complete stay of a person in a playground, and try to track it continuously. We call this longer duration recording of a person we call a "trace". We can use these traces to create better predictions by either combining multiple predictions, or by using machine learning methods that can extract temporal features such as oscillating limb movement and macro movements described in section 4.3.

It is not always possible to create only one trace for a person. The system may lose tracking when someone moves out of range, line of sight is obstructed, or two people move too close together such that pointclouds merge. The average trace length in our training dataset is 46 samples, and there are 311 traces with a length over 100 samples.

Averaging predictions

If (a part of) the errors made by the classifier are uncorrelated, combining the probabilities from multiple predictions will create a better prediction. As more predictions are combined, the prediction will asymptotically become more accurate, with only systematic errors remaining.

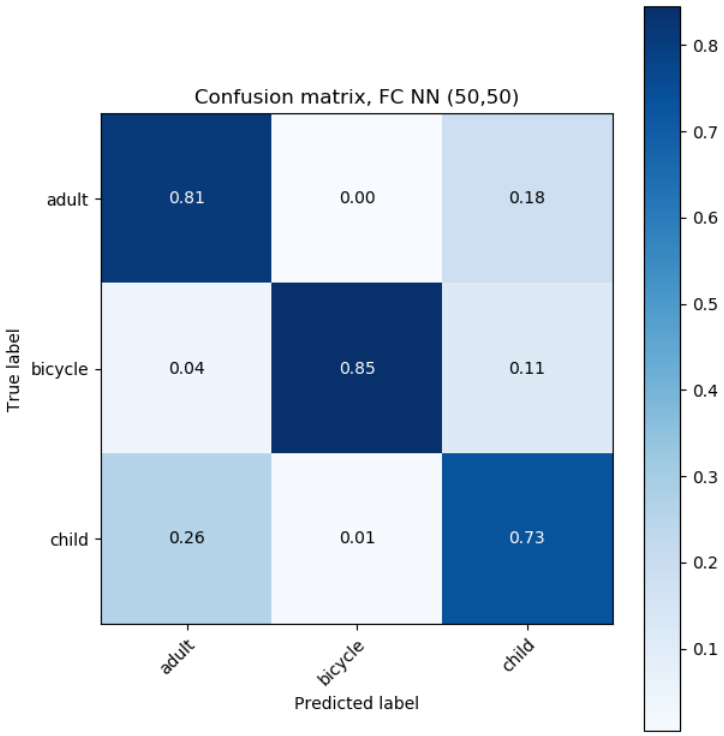


Figure 4.9: **The confusion matrix for our Fully Connected Neural Network.**

The algorithm selects a sequence of consecutive feature vectors from the same trace, and calculates the predictions for each feature vector. The output from the single feature vector classifier does not only contain a predicted label, but also probability values for each class. By using these probability values, predictions with a higher certainty wil have a larger contribution to the final result. It then averages the probabilities for each class of all the predictions.

An important note is that traces shorter than the sequence length are dropped. This changes the distribution of the dataset, giving an accuracy advantage as longer traces tend to have a better signal quality and larger point clouds.

In figure 4.10, we can see that the accuracy does increase when averaging more results with up to 98% accuracy for traces over 200 frames, or 20 seconds of data.

Besides evaluating traces with a fixed length, we also evaluated the result when averaging all predictions for traces on our dataset. By combining all data for a trace into one, our validation dataset is reduced to only 27 predictions. With such a low number of predictions, it becomes less likely that our accuracy value represents the true accuracy when tested on a different, larger dataset. We can model the uncertainty in the accuracy as a *binomial proportion confidence interval*. We use the Wilson method [12] to calculate confidence intervals for the true accuracy value. This method assumes that the probability of any prediction

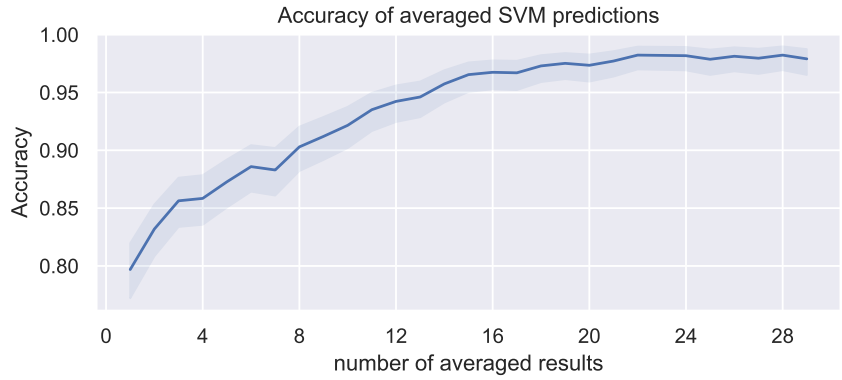


Figure 4.10: **Results when averaging the results from the SVM with increasing window, with 95% confidence interval.**

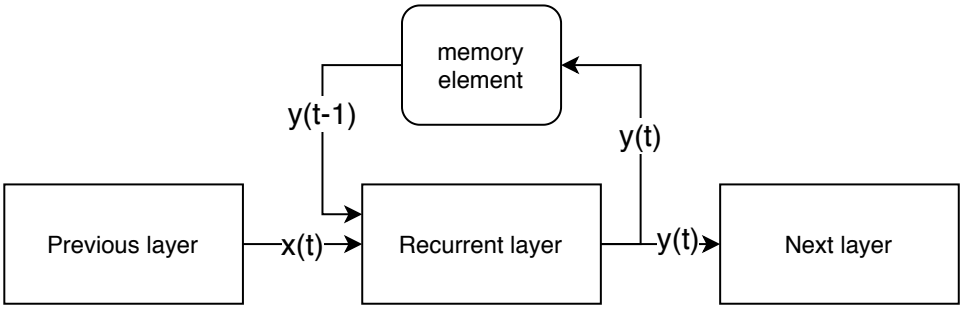


Figure 4.11: **Basic structure of a recurrent layer. The output of the previous time step is fed back into the same layer in the current time step.**

to be correct is equal to the true accuracy of the model. When combining all estimations for a trace, the accuracy is 88% on the validation set, with a 95% confidence interval of (72%, 96%).

Recurrent neural nets

Recurrent Neural Networks (RNN) can classify sequences of data, where a sequence is a list of data points. Typically, these data points represent temporal data, such as audio samples or words in a sentence, but can be any feature vector. Rather than simply averaging the results from classifying the samples individually, an RNN can extract features from the temporal data itself. For instance, determine the frequencies of an oscillating signal. Data can be fed into the network one sample at the time. The network can consist of normal layers and at least one recurrent layer. A recurrent layer receives two inputs, the output of the previous layer, and its own output at timestep $t - 1$. The output from the last step is stored in memory, and called the *hidden state*.The problem with this model is that the input from the last timestep has the most contribution to the hidden state, and earlier timesteps have exponentially less

contribution. All previous information gets "squeezed" into a fixed range by the activation function. This limits the ability to remember long sequences of this type of network [8].

This problem has been overcome by gated RNNs, the most successful versions being the LSTM [21] (Long short term memory) and GRU [11] (Gated Recurrent Unit) cells. These gated cells both introduce a new type of memory called the cell state. The cell state is designed to accumulate information over a longer time scale. It is only changed with multiplications and additions in every time step, without an activation function. The cell state is only updated selectively by the input vector based on some activation function. The GRU is a simpler implementation of a gated RNN compared to the LSTM. This means that an LSTM generally outperforms a GRU, but this comes at the cost of a higher complexity.

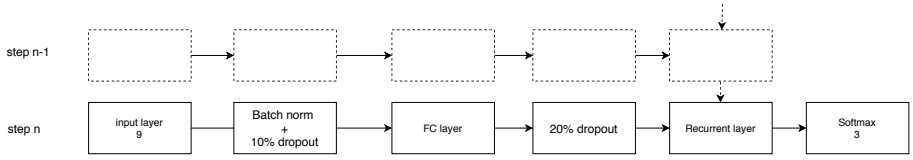


Figure 4.12: **The recurrent network design. We evaluated the network with both a LSTM and GRU recurrent layer. Layers sizes are a tradeoff between performance and network size. Batch normalization and dropout are used to improve generalisation.**

Our network is shown in figure 4.12. It uses one additional fully connected layer before the recurrent layer to map the input vector to a larger embedding space and add some nonlinearity. We use batch normalization and dropout before the first hidden layer, and another dropout layer after the first FC layer. The network is trained in batches of 64 sequences. Batch normalization normalizes the input data in the same way as the normalization step used with our SVM/Random Forrest classifier, but normalizes one batch at the time such that the entire dataset is not needed to be in memory at the same time. Dropout sets some of the values in a vector to zero with a given probability. This is an effective method to make the model more general [50].

We experimented with both GRU and LSTM recurrent layers of different sizes to make a tradeoff between computational resources and performance. We found that even minimal networks have a similar performance to larger networks. Using larger networks leads to more overfitting where the training set accuracy keeps increasing, but the accuracy on the validation set does not improve, as seen in figure 4.13. The performance of the different types and sizes of the layers is shown in table 4.2.

To prepare the input data, we use a moving window over each track of a person to create sequences of a fixed length. For a track of 500 samples with a window size of 100 samples, we create 401 sequences of size 100. Since grouping multiple point clouds together into one point cloud before extracting features was effective in single feature vector methods, we also used this technique with an LSTM network. Since grouping more point clouds reduces the sequence length for a fixed amount of radar frames, we have to find the combined optimal point of grouped point clouds and sequence length. The tradeoff between grouping

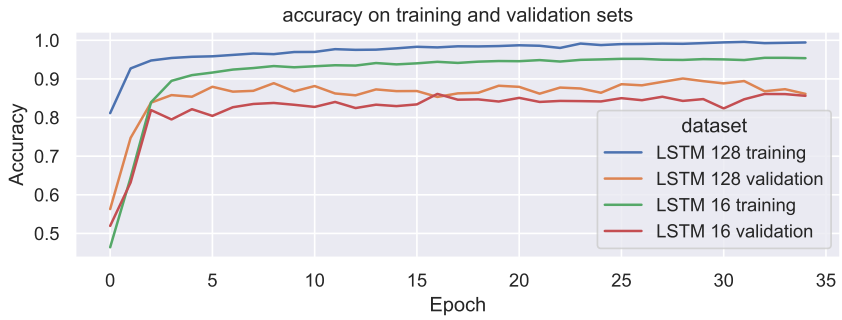


Figure 4.13: **Training and validation accuracy for a lstm of size 16 and of size 128. The larger network has a accuracy approaching 1 on the training set, but does not have a significantly higher accuracy on the validation set.**

Method/ size	16	32	50	64	100	128
GRU	0.87	0.89	0.88	0.90	0.91	0.93
LSTM	0.91	0.90	0.90	0.91	0.88	0.89

Table 4.2: **Classification accuracy of LSTM and GRU networks of different sizes on the validation set. We found no significant improvement in the LSTM performance when using networks larger than 16 cells. The GRU network improves with larger network sizes, but requires a much larger network to match the LSTM performance. The values can vary slightly based on the randomized initial conditions of the training process.**

and sequence length can be seen in figure 4.15. Compared to the single feature vector methods, the optimal amount of grouped point clouds is reduced from 10 to 2. Our hypothesis is that the LSTM can take advantage of the time-domain data preserved when only grouping two point clouds.

The confusion matrix of the size 16 network can be seen in figure 4.14. The model with layers of size 16 has 2379 parameters. If those are stored as 32-bit float, it requires 9kb of memory for storage. The model can be implemented on embedded hardware using Tensorflow lite.

4.5 Summary

Model	SVM	RF	FC NN	averaged SVM	averaged SVM	LSTM
Accuracy	0.79	0.78	0.78	0.93	0.88	0.91
Size	<1KB	22KB	12.5KB	<1KB	<1KB	9KB
Frames	10	10	10	100	all	100

Table 4.3: **Accuracy and networks size of methods tested.**

In this chapter, we have shown a complete data processing pipeline to gen-

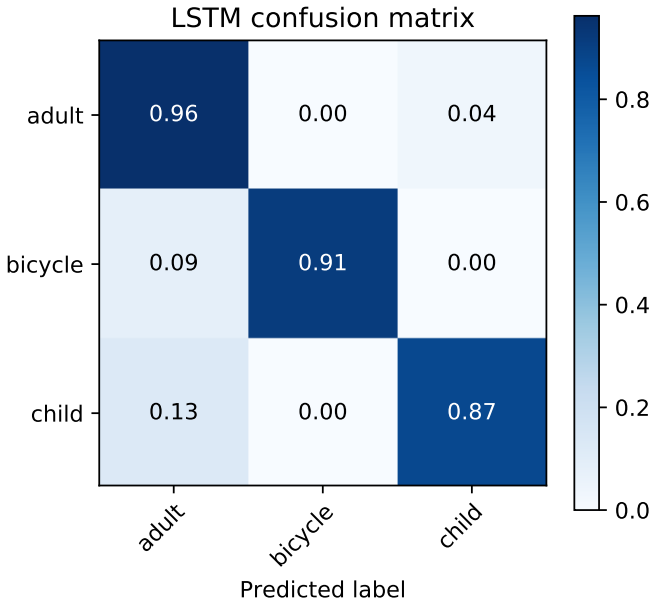


Figure 4.14: **Confusion matrix for the 16-cell LSTM network.**

erate classifications from radar data generated by the IWR6843. We use radar point clouds because our radar does not have the ability to calculate, store, and communicate a complete radar cube. We use the point cloud to extract features that represent the physical differences of our classes. We combine up to 10 point clouds to improve the accuracy of these features.

We have evaluated four different types of machine learning methods to classify based on the extracted features, in table 4.3 we show the results of these methods. We found that all single frame methods hit a limit at close to 80%. By combining all predictions for a person, we achieve 88% accuracy using an SVM. By using only sequences of 100 frames, we achieve an accuracy of 93%. By using an LSTM network, we achieve an accuracy of 91% using sequences of 100 frames.

Based on these results, the time averaging SVM classifier performs best on both or primary criteria of accuracy, and our secondary criteria of minimizing processing resources. However, the accuracy of both methods using multiple feature vectors have overlapping confidence intervals. A larger dataset is needed to conclusively state which method performs better.

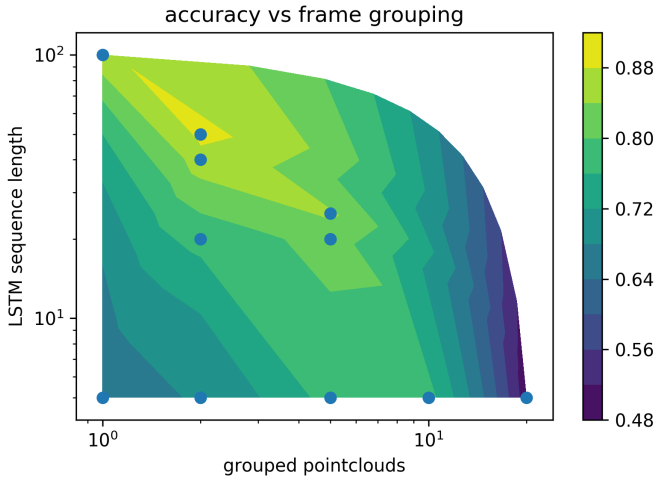


Figure 4.15: **LSTM accuracy when using different sequence lengths and point cloud groupings before feature extraction. The highest accuracy is achived with 2 grouped pointclouds and a sequence length of 50.**

Chapter 5

Conclusions

In this thesis, we have developed a measurement system for measuring playground usage using a mmWave radar sensor. In particular, we focused on the problem of predicting if a user is a child, adult, or bicyclist.

Together with the municipality of Rotterdam, we formulated a set of requirements for a system. By evaluating many different types of person detecting sensors, we conclude that the mmWave radar sensor best fits our requirements. It can detect and localize people in a 30m x 20m area with a single sensor, while powered by only a solar panel. Furthermore, it protects people's privacy as its output can not be used to identify persons uniquely.

Then, we show a classification system for predicting if a playground user is an adult or child, or if either is riding a bike. We have compared methods for representing the radar data and found that due to memory and processing requirements, point clouds are the best for our resource-constrained system. We considered what features could be extracted from the point cloud and evaluated their performance. We found that height and the size of the point cloud are the most important features for classifying children and adults. Bicyclists show a larger radar cross-section and more movement than the other classes.

To train the classifier, we have collected datasets outside the lab at a school playground and grass fields. While dividing the data into a training and validation dataset, we found that splitting the datasets randomly results in data leaking, where the same information is present in both the training and validation sets. In order to prevent this, used different data sets for training and validation.

We have compared SVM, Random Forrest, and Dense neural networks and found that all of these networks achieve an accuracy close to 80%. Then, we trained neural networks with recurrent layers with sequences of feature vectors and found that LSTM networks achieved a performance of 91% accuracy on a 10 second sequence of 100 point clouds. By averaging all classifications of a person by an SVM classifier over time, we reach an accuracy of 88% on our validation dataset, and it reaches over 97% accuracy when more than 200 pointclouds are captured for a person. The model is less than 1KB in size, and can be implemented on embedded hardware.

We conclude that the averaging SVM classifier is the best option to use for classifying children, adults and bicyclists based on radar point cloud data. We also conclude that our measurement system works in the small scale but real-

world situations we have tested the system. And that this technique can be used by municipalities to measure the usage of their playgrounds.

The tools, scripts and data used in this report is available at the project's github page: `https://github.com/BBekker/mmWavePOI`.

5.1 Future work

The main improvement on this work is to increase the size of the dataset used. Future work should focus on gathering a larger dataset with more participants. With a larger dataset, the network can be trained more accurately and can be evaluated with more confidence, removing biases due to our limited set of samples. A specific gap in our dataset is the lack of a third dataset as a test set. We evaluated multiple models based on the performance of the validation set and chose the best performing as our final model. Meaning that our model is cherrypicked for this dataset, and its performance on the validation dataset may not represent its performance on different datasets or a general deployment.

We worked with the municipality of Rotterdam and the AMS institute to organize a pilot project at a public playground to generate a much larger dataset, but were not able to finish this within the timeframe of this thesis. To set up a test at a public location, both the sensor and a method for measuring the ground truth need to be accepted to fulfill all local and national rules and regulations. We were not able to get approval in time, and for that reason, did not produce the hardware as described in section . We suggest that future work should first secure approval for a broader deployment at public playgrounds, before proceeding with the technical work. Our work lays the groundwork for that more extensive study.

A potential improvement in the classifier is to use information from the radar cube. We were not able to use this data because of the hardware restrictions of the IWR6843. If in future work, this information can be extracted, it opens the possibility for more advanced machine learning methods explored in section 2.3.2 such as 3-d convolutional neural networks.

Bibliography

[1] 68xx isk - 100m outdoor people tracking and false detection filtering. `http://dev.ti.com/tirex/explore/node?node=AKZEzABcb99p.yMh3gzLlA__VLyFKFf__LATEST`. Accessed: 2020-05-13.

[2] Regeling gebruik van frequentieruimte zonder vergunning en zonder meldingsplicht 2015, bijlage 11, subcategorie 1.

[3] Solar module phaesun sun plus 10, 2020.

[4] M. N. Afsar. Precision millimeter-wave measurements of complex refractive index, complex dielectric permittivity, and loss tangent of common polymers. *IEEE Transactions on Instrumentation and Measurement*, IM-36(2):530–536, 1987.

[5] Arkadiusz Bulski, Tomer Filiba, Corbin Simpson. Construct user guide. `https://construct.readthedocs.io/en/latest/`, 2020. [Online; accessed 21-May-2020].

[6] Amanda Berg. *Detection and Tracking in Thermal Infrared Imagery*. PhD thesis, Linköping University Electronic Press, 2016.

[7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[8] Andriy Burkov. *The hundred-page machine learning book*. Andriy Burkov, 2019.

[9] P. Cao, W. Xia, M. Ye, J. Zhang, and J. Zhou. Radar-id: human identification based on radar micro-doppler signatures using deep convolutional neural networks. *IET Radar,Sonar Navigation*, 12(7):729–734, 2018.

[10] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, July 2017.

[11] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[12] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media, 2005.

[13] Amy Droitcour, Victor Lubecke, Jenshan Lin, and Olga Boric-Lubecke. A microwave radio for doppler radar sensing of vital signs. In *2001 IEEE MTT-S International Microwave Sympsoium Digest (Cat. No. 01CH37157)*, volume 1, pages 175–178. IEEE, 2001.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD96, page 226231. AAAI Press, 1996.

[15] Martín Abadi et al. (Google Research). TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[16] Z. Feng, S. Zhang, M. Kunert, and W. Wiesbeck. Point cloud segmentation with a high-resolution automotive radar. In *AmE 2019 - Automotive meets Electronics; 10th GMM-Symposium*, pages 1–5, 2019.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[18] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey, 2019.

[19] Abbass Hammoud, Michel Deriaz, and Dimitri Konstantas. Ultrasense: A self-calibrating ultrasound-based room occupancy sensing system. *Procedia Computer Science*, 109:75 – 83, 2017. 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal.

[20] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] EU Science hub. Pvgis, 2020.

[23] Texas Instruments. *mmWave SDK user guide*, 2019.

[24] DIGI international. Digi xbee 3 cellular lte-m/nb-iot, 2020.

[25] Rohan Kapoor, Subramanian Ramasamy, Alessandro Gardi, Ron Van Schyndel, and Roberto Sabatini. Acoustic sensors for air and surface navigation applications. *Sensors*, 18(2):499, 2018.

[26] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):1–21, 2012.

[27] Youngwook Kim and Taesup Moon. Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks. *IEEE geoscience and remote sensing letters*, 13(1):8–12, 2015.

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[29] Dawei Liang and Edison Thomaz. Audio-based activities of daily living (adl) recognition with large-scale acoustic embeddings from online videos. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(1), March 2019.

[30] L. Liu, M. Popescu, M. Skubic, M. Rantz, T. Yardibi, and P. Cuddihy. Automatic fall detection based on doppler radar motion signature. In *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 222–225, 2011.

[31] Kuba Lopatka, Jozef Kotus, and Andrzej Czyzewski. Detection, classification and localization of acoustic events in the presence of background noise for acoustic surveillance of hazardous situations. *Multimedia Tools and Applications*, 75(17):10407–10439, 2016.

[32] Melexis. *MLX90640 32x24 IR array*, December 2019. Rev. 12.

[33] A. Palffy, J. Dong, J. F. P. Kooij, and D. M. Gavrila. Cnn based road user detection using the 3d radar cube. *IEEE Robotics and Automation Letters*, 5(2):1263–1270, 2020.

[34] S Parnin and MM Rahman. Human location estimation using thermopile array sensor. In *IOP Conference Series: Materials Science and Engineering*, volume 260, page 012007. IOP Publishing, 2017.

[35] K. Patel, K. Rambach, T. Visentin, D. Rusev, M. Pfeiffer, and B. Yang. Deep learning-based object classification on automotive radar spectra. In *2019 IEEE Radar Conference (RadarConf)*, pages 1–6, April 2019.

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

[37] R. Prophet, M. Hoffmann, A. Ossowska, W. Malik, C. Sturm, and M. Vossiek. Image-based pedestrian classification for 79 ghz automotive radar. In *2018 15th European Radar Conference (EuRAD)*, pages 75–78, Sep. 2018.

[38] Robert Prophet, Marcel Hoffmann, Alicja Ossowska, Waqas Malik, Christian Sturm, and Martin Vossiek. Pedestrian classification for 79 ghz automotive radar systems. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1265–1270. IEEE, 2018.

[39] D. K. A. Pulutan and J. S. Marciano. Design trade-offs in a combined fmcw and pulse doppler radar front-end. In *IEEE 2013 Tencon - Spring*, pages 567–571, April 2013.

[40] Pyqtgraph team. Pyqtgraph. `https://www.pyqtgraph.org`, 2020. [Online; accessed 21-May-2020].

[41] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[42] R. Ricci and A. Balleri. Recognition of humans based on radar micro-doppler shape spectrum features. *IET Radar, Sonar Navigation*, 9(9):1216–1223, 2015.

[43] Santiago Royo and Maria Ballesta-Garcia. An overview of lidar imaging systems for autonomous vehicles. *Applied Sciences*, 9(19):4093, 2019.

[44] Lorenz Schauer, Martin Werner, and Philipp Marcus. Estimating crowd densities and pedestrian flows using wi-fi and bluetooth. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 171–177, 2014.

[45] Nicolas Scheiner, Nils Appenrodt, Jürgen Dickmann, and Bernhard Sick. Radar-based feature design and multiclass classification for road user recognition. *CoRR*, abs/1905.11256, 2019.

[46] O. Schumann, M. Hahn, J. Dickmann, and C. Whler. Semantic segmentation on radar point clouds. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2179–2186, July 2018.

[47] O. Schumann, C. Whler, M. Hahn, and J. Dickmann. Comparison of random forest and long short-term memory network performances in classification tasks using radar. In *2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–6, 2017.

[48] Oliver Shih and Anthony Rowe. Occupancy estimation using ultrasonic chirps. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, pages 149–158, 2015.

[49] Daniel J. Solove. A taxonomy of privacy. *University of Pennsylvania Law Review*, 154(3):477–564, 2006.

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[51] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, 8(1):25, 2007.

[52] George P. Succi, Daniel Clapp, Robert Gampert, and Gervasio Prado. Footstep detection and tracking. In Edward M. Carapezza, editor, *Unattended Ground Sensor Technologies and Applications III*, volume 4393, pages 22 – 29. International Society for Optics and Photonics, SPIE, 2001.

[53] Texas Instruments. *IWR6843 Single-Chip 60- to 64-GHz mmWave Sensor*, 2020.

[54] The QT Company. Qt framework. `https://www.qt.io/`, 2020. [Online; accessed 21-May-2020].

[55] Harry L Van Trees. *Optimum array processing: Part IV of detection, estimation, and modulation theory*. John Wiley & Sons, 2004.

[56] Isabel Wagner and David Eckhoff. Technical privacy metrics: A systematic survey. *ACM Comput. Surv.*, 51(3), June 2018.

[57] J. Wilson and N. Patwari. Radio tomographic imaging with wireless networks. *IEEE Transactions on Mobile Computing*, 9(5):621–632, May 2010.

[58] Peijun Zhao, Chris Xiaoxuan Lu, Jianan Wang, Changhao Chen, Wei Wang, Niki Trigoni, and Andrew Markham. mid: Tracking and identifying people with millimeter wave radar. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 33–40. IEEE, 2019.