Universidade de Aveiro
2019

Departamento de Eletrónica,
Telecomunicações e Informática

José Pedro Baião Castanheira

Aprendizagem Automática Aplicada à Deteção de Pessoas Baseada em Radar

Machine Learning Methods for Radar-Based People Detection

Universidade de Aveiro
2019

Departamento de Eletrónica,
Telecomunicações e Informática

José Pedro Baião Castanheira

Aprendizagem Automática Aplicada à Deteção de Pessoas Baseada em Radar

Machine Learning Methods for Radar-Based People Detection

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Francisco José Curado Mendes Teixeira, Investigador do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e da Doutora Ana Maria Perfeito Tomé, Professora Associada do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor Joaquim João Estrela Ribeiro Silvestre Madeira
Professor Auxiliar do Departamento de Engenharia Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Eugénio Alexandre Miguel Rocha
Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro

Doutor Francisco José Curado Mendes Teixeira
Investigador do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos /
acknowledgements

Aos meus pais, José Manuel Castanheira e Maria de Lurdes Baião, agradeço por me terem dado a oportunidade e incentivado a seguir o meu percurso académico. Foram o meu maior apoio durante todos estes anos de estudo, e sem eles, nada teria sido possível.

Aos meus irmãos, João e Miguel, e a toda a minha família, agradeço por estarem sempre presentes, ainda que muitas vezes longe, e por todo o carinho que sempre me deram.

Agradeço profundamente aos meus orientadores, Prof. Doutor Francisco Curado e Prof. Doutora Ana Tomé, todos os ensinamentos que me transmitiram e que foram essenciais para o desenvolvimento desta dissertação. Agradeço também as correções minuciosas e por me incentivarem sempre a fazer melhor.

A todos os meus colegas do Laboratório IRIS, agradeço a boa disposição e o apoio dado no trabalho. Em particular, quero destacar o Eurico Pedrosa e o Edgar Gonçalves, por terem contribuído largamente para a evolução do trabalho, e pela paciência e ajuda fundamental que deram durante a fase de testes experimentais, desenvolvimento de código e estudo da tecnologia do radar.

À Catarina, quero agradacer por me ter motivado, pelos vários conselhos que foi dando ao longo de todo o trabalho, e por ter estado sempre presente.

Aos meus amigos quero agradecer todos os bons momentos que passámos, todas as brincadeiras e todo o divertimento que proporcionaram ao longo destes anos.

A todos os meus colegas no curso e na universidade, agradeço o companheirismo e a entreajuda.

Finalmente, deixo uma palavra de gratidão a todos aqueles que, de uma forma ou de outra, me tenham apoiado durante o meu percurso académico.

Esta dissertação foi realizada no âmbito do Projeto RETIOT, *"Reflectometry Technologies to Enhance the Future Internet of Things and Cyber-Physical Systems"*, financiado pelo Fundo Europeu de Desenvolvimento Regional, através do Programa Operacional temático "Competitividade e Internacionalização", e pela Fundação para a Ciência e Tecnologia (FCT), sobre o contrato POCI-01-0145-FEDER-016432.

Palavras Chave

Deteção de Pessoas, Ambientes Interiores, Radar, Aprendizagem Automática, Robótica Móvel

Resumo

A presente dissertação descreve o desenvolvimento e implementação de um sistema baseado em radar que tem como objetivo detetar e distinguir pessoas de outros objetos que se movem num ambiente interior. Os métodos de deteção e distinção exploram os dados de radar que são processados por um sistema que abrange a aquisição e pré-processamento dos dados, a extração de características, e a aplicação desses dados a modelos de aprendizagem automática especificamente desenhados para atingir o objetivo de classificação de alvos.

Além do estudo da teoria básica de radar para o desenvolvimento bem-sucedido desta dissertação, este trabalho contempla uma componente importante de desenvolvimento de *software* e testes experimentais. Entre outros, os seguintes tópicos foram abordados nesta dissertação: o estudo dos princípios básicos do funcionamento do radar e do seu equipamento; processamento de sinal do radar; técnicas de remoção de ruído, extração de características, e segmentação de dados aplicada ao sinal de radar; implementação e calibração de hiper-parâmetros dos modelos de aprendizagem automática para sistemas de classificação; estudo de métodos de deteção e seguimento de múltiplos alvos.

A aplicação para deteção de pessoas foi testada em diferentes cenários interiores que incluem o radar estático ou transportado por um robot móvel. Esta aplicação pode ser executada em tempo real e realizar deteção e classificação de múltiplos alvos usando algoritmos básicos de segmentação e seguimento. O estudo do impacto da deteção de múltiplos alvos no funcionamento da aplicação é apresentado, bem como a avaliação da eficiência dos diferentes métodos de classificação usados.

As possíveis aplicações do sistema de deteção proposto incluem a deteção de intrusão em ambientes interiores e aquisição de dados anónimos para seguimento e contagem de pessoas em espaços públicos tais como hospitais ou escolas.

Keywords

People Detection, Indoor Environments, Radar, Machine Learning, Mobile Robotics

Abstract

The present dissertation describes the development and implementation of a radar-based system with the purpose of being able to detect people amidst other objects that are moving in an indoor scenario. The detection methods implemented exploit radar data that is processed by a system that includes the data acquisition, the pre-processing of the data, the feature extraction, and the application of these data to machine learning models specifically designed to attain the objective of target classification.

Beyond the basic theoretical research necessary for its sucessful development, the work contamplates an important component of software development and experimental tests. Among others, the following topics were covered in this dissertation: the study of radar working principles and hardware; radar signal processing; techniques of clutter removal, feature exctraction, and data clustering applied to radar signals; implementation and hyperparameter tuning of machine learning classification systems; study of multi-target detection and tracking methods.

The people detection application was tested in different indoor scenarios that include a static radar and a radar dynamically deployed by a mobile robot. This application can be executed in real time and perform multiple target detection and classification using basic clustering and tracking algorithms. A study of the effects of the detection of multiple targets in the performance of the application, as well as an assessment of the efficiency of the different classification methods is presented.

The envisaged applications of the proposed detection system include intrusion detection in indoor environments and acquisition of anonymized data for people tracking and counting in public spaces such as hospitals and schools.

Contents

List of Figures

List of Tables

List of Acronyms

ADC Analog-to-digital Converter

AoA Angle of Arrival

ANN Artificial Neural Network

API Aplication Programmming Interface

BO Bayesian Optimization

DBSCAN Density-based Spatial Clustering of Applications with Noise

DSP Digital Signal Processor

EA Evolutionary Algorithms

FT Fourier Transform

FFT Fast Fourier Transform

FMCW Frequency-Modulated Continuous Wave

FoV Field of View

GLCM Gray-Level Co-Occurrence Matrix

GUI Graphical User Interface

IF Intermediate Frequency

ML Machine Learning

MIMO Multiple-input Multiple-output

OS Operative System

TI Texas Instruments

RADAR Radio Detection and Ranging

RCS Radar Cross Section

RF Random Forest

ROS Robot Operating System

RNN Recurrent Neural Network

Chapter 1

Introduction

Radar, which stands for **Ra**dio **D**etection **A**nd **R**anging, was designed to detect targets, and measure their respective distances and velocities. This is accomplished by emitting radio waves that are reflected by other objects. Radar, as a sensory technology, had its initial major developments in World War II, were it was used to detect missiles and airplanes. The fundamental theoretical principles, however, were formulated years before. The first radar systems were limited by the technology of the time, in particular the limited power and frequency on the signal generator, which in turn restricted the detection range and demanded large antennas [1]. Radars were mostly used in the military industry, but in the post-war period the radar devices continued to be refined, and currently can achieve high resolutions in range and velocity due to the increase in the frequencies transmitted and received by the radars, and the availability of powerful digital signal processing hardware. This makes radars appealing to integrate in modern civil applications, such as civil and maritime navigation, meteorology and medicine, and also the automobile industry in the context of autonomous navigation [2].

Autonomous navigation is a research area which has been drawing increasing attention in the latest decades. Autonomous navigation is not only focused on self-driving cars, but in any kind of robotic agent that can move on its own. Boosted by the developments on artificial intelligence, computer processing power, sensory technology, and safety concerns regarding the automotive mobility, this area promotes a world where robotic systems can travel in environments where the presence of people is typical, and execute tasks in a way that was not possible before. Such tasks might include survivor detection in emergency scenarios, intrusion detection, mapping, etc. In order to navigate in any scenario without the aid of humans, robotic agents need to have a mean to observe the surrounding environment. Hence, sensory devices, such as video cameras, lidar, radar or sonar need to be applied in such systems, and the algorithms for data extraction and processing need to be developed to aim the robot with the necessary information to make autonomous decisions.

In order to provide robotic agents with the necessary intelligence to distinguish objects based on the received information, one modern approach is the use of machine learning (ML) to train a computer to learn patterns from the stored data. In fact, ML has shown that with the necessary and right type of data, and with the adequate models, computers can perform complex decisions based solely on the data they have processed so far. These tasks include speech and image recognition, medical diagnosis such as the analysis of skin lesions, among others.

1.1 Motivation

The utilization of standard sensors such as cameras, laser or infrared in order to detect the presence of people and other objects in a given scenario of interest is often hampered by the diversity of environmental effects such as dust, fog, rain, darkness, strong heat or fire. In contrast, radar signals are immune to these disturbances. Additionaly, radar systems can be easily parameterized to deal with a diversity of operational conditions. In recent years, the number and diversity of applications of radar sensors has been increasing dramatically motivated by the emergence of inexpensive, self-contained development kits. These devices permit rapid prototyping and testing of radar-based solutions and their reduced cost and dimension make them affordable for academic research and permit their integration in small, inexpensive robots.

Despite the aforementioned advantages, robust feature extraction from radar data is still an open problem because the radar physics is prone to generate a diversity of artifacts that pose significant challenges to conventional signal processing techniques. This problem is especially dramatic in indoor environments due to the ubiquity of obstacles (clutter) normally present in these scenarios.

The present dissertation introduces a novel ML approach integrated with mobile robotic platforms to address the problem of radar-based detection of moving people in an indoor scenario. To the best of my knowledge, none of the alternative works address the specific problem that is tackled in this dissertation: the detection of people (walking and on wheelchair) in indoor environments based on statistical measures of radar Doppler data acquired by static or mobile robots. The envisioned applications include elderly monitoring and rescue operations in vision-denied environments. The dissertation does not discuss other sensor alternatives (e.g. vision) because the radar is intended to work as a complementary instrument or as a replacement sensor in scenarios where the alternative sensors are prone to fail.

1.2 Objectives

The envisaged application aims to distinguish people moving amidst other objects, using a small, low-cost, MIMO radar. This will be achieved by extracting data from moving targets with the radar. The data will be used to generate datasets, which will be used to train classification models to be able to distinguish people moving amidst other objects. The application intends to detect not only people walking but also people moving in wheelchairs. Different optimization techniques will be applied to try to maximize the performance of the ML models. Lastly, the developed application is intended to function in real time, with the radar deployed by a moving robot, and detect and classify more than one target simultaneously.

1.3 Main contributions

The main contribution of this dissertation consists in the introduction of a new, robust approach for classification of moving objects based on radar data, with the ability to distinguish people from other moving agents in an indoor environment. The specific achievements of this work include:

- The exploitation of the Doppler histograms representative of the velocity distribution of the points that constitute the point cloud acquired by the radar (originated from the multiple scatterers of a given object) as a feature that permits a robust distinction between moving rigid bodies and people.

- The utilization of the radar cross section of targets as a classification feature, and the derivation of a practical method for its computation.

- The demonstration of the radar-based classification system ability to work not only with static radars but also to be deployed and operated autonomously by mobile robots. For such, a dedicated standalone radar kit was assembled and tested in the present work.

- The development and implementation of basic algorithmic tools for clutter removal and outliers detection that can be used as building blocks of more advanced tracking and data association methods for people detection and tracking.

1.4 Dissertation structure

This dissertation is structured in the following six chapters:

- **Chapter 1** - This is an introductory (and current) chapter of the dissertation. The chapter starts by succinctly explaining the radar purpose and inner workings and gives a brief description of its history. The applications of radar are presented, and a connection is established with the use of ML in these same applications. The text moves on to explain the advantages of the development of people detection applications with radars, and presents a summary of the steps implemented to achieve it.

- **Chapter 2** - Basic radar principles, particularly the theory behind the FMCW radar technology, are described in this chapter. The equations of range, velocity and angle estimation and resolution are explained, and the concept of radar cross section is introduced. The main radar hardware and software tools employed in this work are also presented.

- **Chapter 3** - This dissertation focuses on the use of ML, therefore, several concepts of this field are explained in this chapter, as well as the software tools used. Related work on people detection with radars and ML is also presented.

- **Chapter 4** - The development of an application for people detection with a static radar is described in this chapter. The data received from the radar is described, and the process of feature extraction and processing is detailed. Then, the results of the models performance are plotted and compared.

- **Chapter 5** - This chapter details the implementation of methods to enhance the classification models performance, such as the implementation of techniques to optimize the hyperparameters of the models. A method to reduce the dimension of the features used is also described in this chapter.
- **Chapter 6** - Describes the development of an application for people detection using a radar dynamically deployed by a moving robot. This chapter also describes the implementation of a real-time version of the developed application, and the detection of multiple people. The implications and problems originated by the detection and classification of more than one target at the same time are also analyzed.
- **Chapter 7** - Presents the main conclusions of this work and a discussion of the attained results. The chapter also mentions the related work that resulted from this dissertation, such as papers and participation in conferences, and outlines possible future work directions, giving a proper justification on how they might improve this work.

Chapter 2

Radar technology and methods

The present chapter describes the radar basic working principles, particularly the FMCW radar technology. The chapter also presents the main types of radar-based approaches for people detection and tracking.

2.1 Basic radar terminology

For the convenience of the reader this section presents the basic concepts and terminology about radar systems, that will be used throughout this dissertation.

- In radar terminology, any object that can be detected by the radar is often designated as **target**.
- The scalar measure representative of the radial velocity of the target, i.e. its velocity projected onto the central axis of the radar transmitter, is normally designated as **Doppler** since it is computed based on the frequency shift (Doppler effect) incurred by the radar wave due to the target velocity.
- A **frame** constitutes an elementary structure of reflectivity data that characterizes the state of the target in terms of its distance (range) and radial velocity (Doppler), relatively to the radar system.
- The **multiple-input multiple-output** (MIMO) radar technology considered in this work permits the discrimination of simultaneous reflections from different points in the plane of the radar wave by exploiting the beam-formimg capability of the MIMO system.
- In the present context, a **point cloud** is a collection of points corresponding to individual reflections from a scene acquired at a given instant of time and represented in different positions of the radar sensor grid; these points may correspond to different reflections from a single object or from multiple targets.
- **Clutter** is the result of unwanted echoes in the radar field of view (FoV), that may result, for example, from multi-paths of the radio waves reflections.
- **Angle of arrival** (AoA) is the angle between a reflected signal from an object and the central axis of the radar transmitter.

- A **phased array radar** is a radar device with several transmitting antennas and receiving antennas, that can provide signals with different phase shifts. This technique allows the detection of the AoA of multiple reflection points.

2.2 Radar basic working principles

Radar is a technology that permits extracting range, velocity and angle from objects, through electromagnetic waves in the radio frequency spectrum, called radio waves. The way radar systems achieve this is by emitting these radio waves and then receiving and detecting them when they are reflected by other objects as illustrated in figure 2.1. Modern

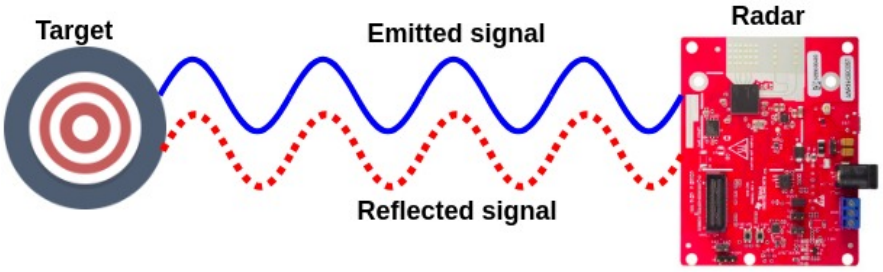


Figure 2.1: Schematic of a radar device detecting a target.

radar systems can also track, identify, image and classify targets, while at the same time suppressing background noise, also known as clutter. There are different types of radar systems, but in general these must include a transmitting antenna, receiving antenna, and signal processor. The transmitting antenna is responsible for generating the microwave signal, while the receiving antenna is the subsystem that receives the electromagnetic waves reflected by the objects (targets of interest or clutter). The signal received is ultimately analyzed by the signal processor [3].

2.2.1 Radar Cross Section

RADAR Cross Section, usually referred by its initials RCS, and represented by the Greek letter σ, is a parameter that, in general terms, describes the energy reflected by an object. The RCS can be interpreted as the effective reflecting area of the target as seen by the radar although its value is not only a function of the size but also of other properties of the target such as its shape and constitutive materials; for example, a small object with a metallic surface can reflect more energy to the radar than a larger target built from soft materials. RCS is a parameter that is often used to distinguish classes of radar targets. Typical values of RCS for each specific type of object - cars, people, ships - are well known and tabulated, and therefore can aid the detection of a specific object. Several formulas to compute the RCS can be found in the literature, and usually relate distance of the measured object with the power of the signal received and transmitted, as well as signal-to-noise ratio.

2.3 Detecting radar targets with multiple reflection points

Due to the characteristics of the electromagnetic waves propagation and the high resolutions attained by radars, these devices are capable of sensing multiple reflection points from different targets. These reflection points constitute point clouds, where each point can represent a vector of data, containing its **range**, **azimuth**, and **radial velocity** (range rate), among other features. Real life targets are thus represented not by a single point but by multiple ones. Figure 2.2 shows the radar geometry schematic. R_{min} and R_{max} are the minimum and maximum range (respectively); $-\phi_{min}$ and $+\phi_{max}$ are the minimum and maximum measurable angle (respectively), and ϕ_n is the azimuth; r_n and $\dot{r_n}$ represent respectively the range and the radial velocity (Doppler) of one point at a given instant in the radar coordinates frame. This figure also illustrates the ability of the radar system to estimate the bi-dimensional velocity of a target, v_n, based on successive detections of its reflections points (point_{n-1} and point_n).

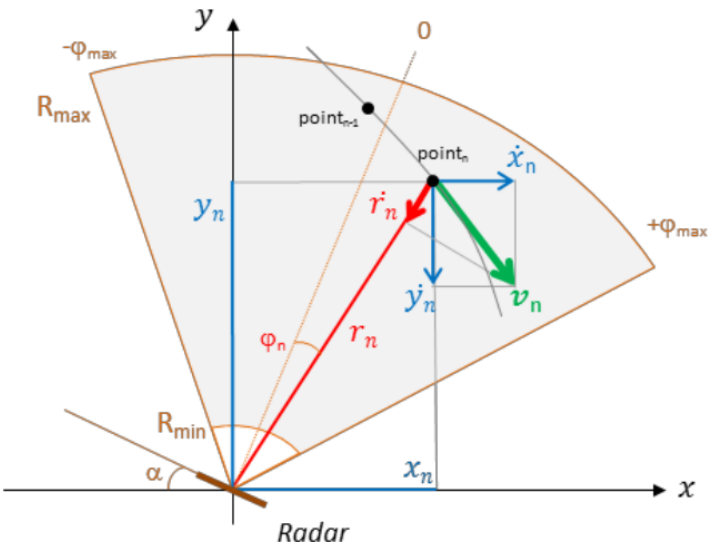


Figure 2.2: Radar geometry in 2D. Source [4].

2.4 FMCW radar

2.4.1 Fundamentals of FMCW radar

Radar systems can be divided into several categories, regarding their characteristics. This dissertation will solely focus on FMCW radar, which will be the one used in this work. FMCW (radar) stands for **F**requency **M**odulated **C**ontinuous **W**ave. FMCW works by emitting a continuous wave with a time-varying frequency. One of the most typical kinds of modulation applied is the sawtooth modulation in which the frequency increases linearly with time [5]. A wave of this type is called a **chirp**. Figure 2.3 shows this typical modulation, as well as some other parameters namely **bandwidth** and **sweep time**. Figure 2.4 shows the signal evolution over time for a single chirp. The bandwidth can be defined as the difference between the maximum and minimum frequency of the chirp. The sweep time is the time necessary for the chirp to traverse all the programmed bandwidth [6].

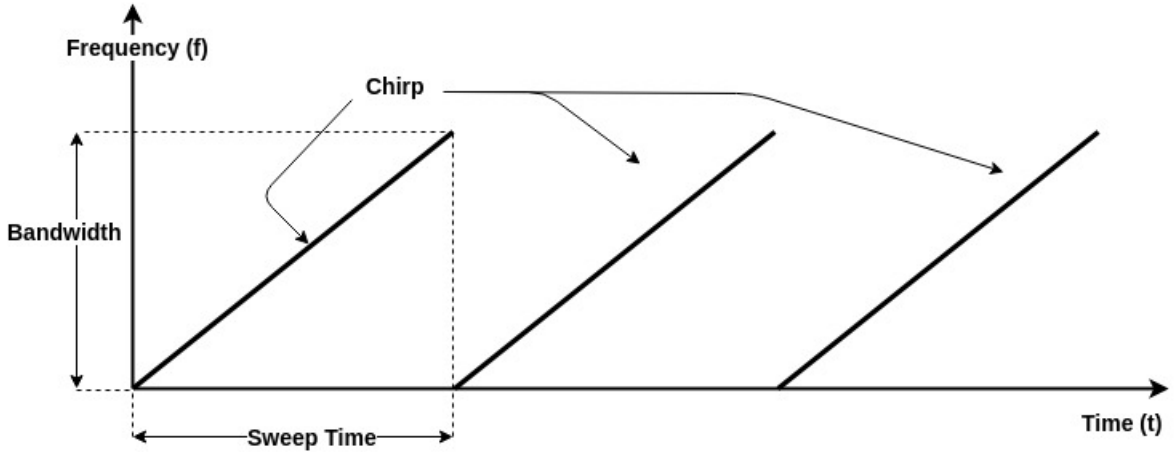


Figure 2.3: Time-frequency representation of chirps with sawtooth modulation.

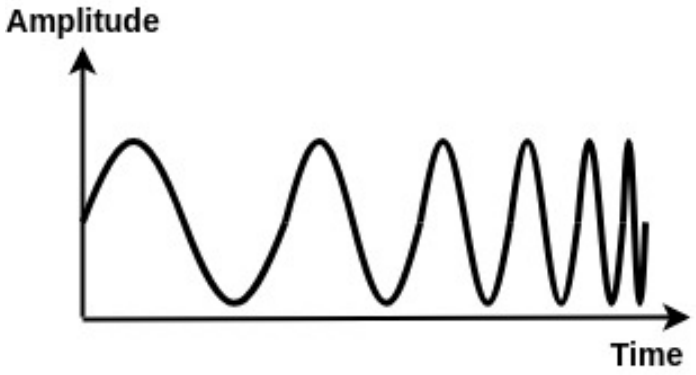


Figure 2.4: Signal evolution over time for a single chirp.

An FMCW radar includes a synthesizer that generates the chirps. The chirps are transmitted by the transmitting antenna(s) and the corresponding reflections from the objects

present in the radar FoV are received by the receiving antenna(s). The emitted signal and the received one are combined by the **mixer**, to generate the so called **Intermediate Frequency** (IF) signal [6]. A schematic representation of this structure is depicted in figure 2.5.

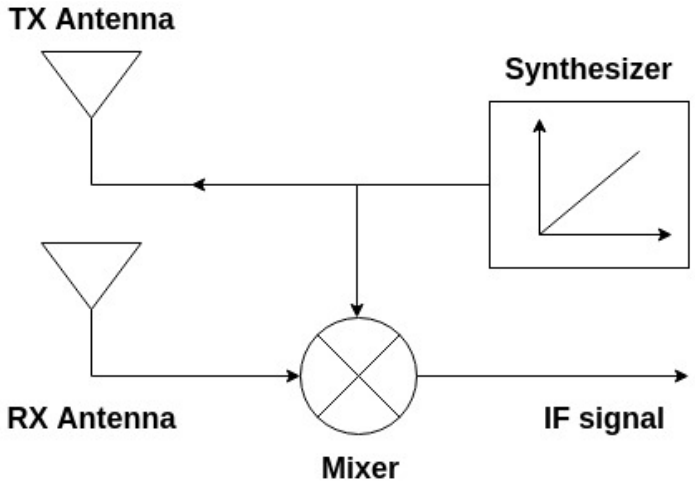


Figure 2.5: Schematic of a radar device with one transmitting antenna and one receiving antenna. Adapted from [6].

2.4.2 Range estimation and resolution

Range estimation

The radar emits a continuous wave signal whose frequency increases linearly with time during the chirp period (sweep time). The reflected signal is a chirp with the same characteristics of the emitted chirp, but delayed in time. Because of this delay, which is a function of the distance to the object, when mixed, there will be a difference in frequency, between the original and reflected chirp, which will be the frequency of the IF signal. Figure 2.6 depicts this rationale, where on the top left side it is possible to see the emitted and received chirp and the resulting IF signal (bottom left); on the right side there is a similar image, but the chirps are further apart on the time domain, which makes the resulting IF signal (bottom right) to have a different (higher) frequency.

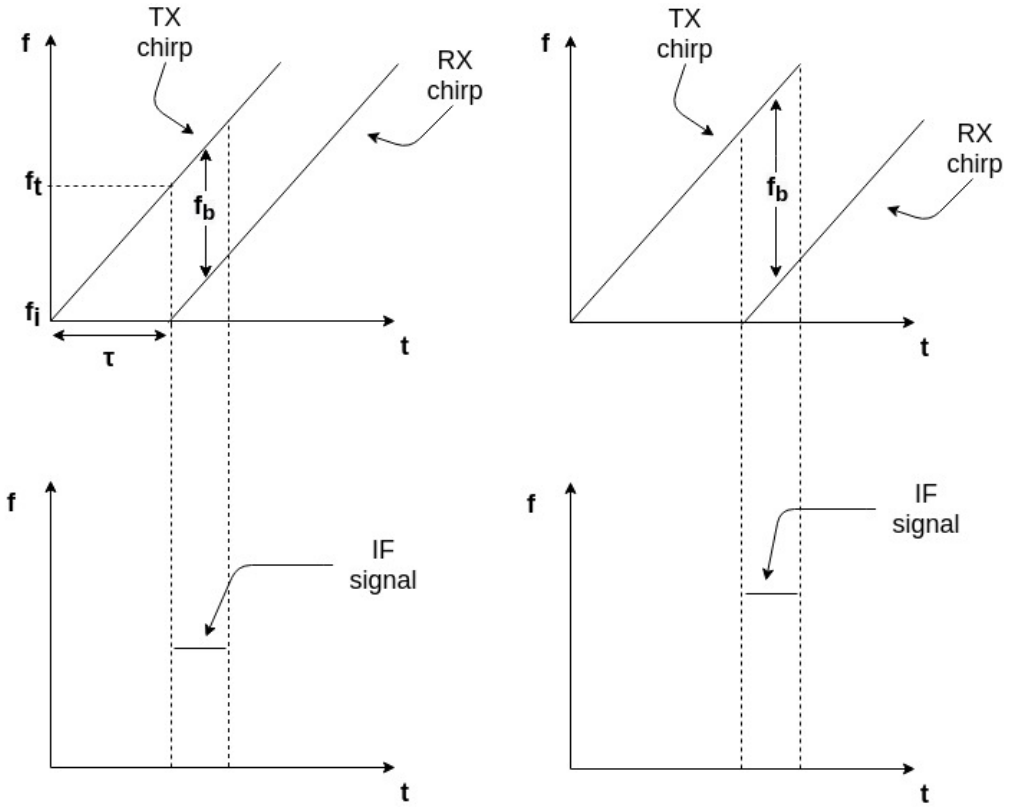


Figure 2.6: Transmission (Tx) and reception (Rx) chirp (top left and right) and resulting IF signal (bottom left and right). f_b is the (beat) frequency of the IF signal; τ is the time of flight; f_i represents the initial frequency and f_t represents the frequency of the emitted chirp after τ seconds. Adapted from [6].

An object standing at a constant distance from the radar produces an IF signal with a constant frequency. Consider d as the distance between the radar and the object, and c as the speed of light in vacuum. The time of flight, represented by τ, can be calculated as

$$\tau = \frac{2d}{c}. \tag{2.1}$$

Consider S as the slope (quotient between bandwidth and sweep time) of the chirp, which means how much frequency increases by unit of time. The beat frequency (f_b) is equal to the subtraction of the frequency at point f_t and point f_i. In this context, f_i is the initial frequency, and f_t is the frequency of the emitted chirp after τ seconds. The value of f_t can be derived by the following equation

$$f_t = f_i + S\tau. \tag{2.2}$$

Considering equation (2.1), we can rewrite the previous equation as

$$f_t = f_i + S\frac{2d}{c}. \tag{2.3}$$

The respective IF signal frequency, represented by f_b, is calculated as

$$f_b = f_t - f_i = S\frac{2d}{c}. \tag{2.4}$$

The above equation shows that in principle it is possible to calculate the distance from the radar to an object using the frequency of the IF signal. The **Fourier Transform** (FT) is a mathematical operation that transforms a function in the time domain into its constituent frequencies (function in the frequency domain), and the **discrete-time Fourier transform** (DFT) is an implementation of FT [7]. The calculation of the frequency of the IF signal is done with a **Fast Fourier Transform** (FFT) over the IF signal. The FFT applied to resolve the targets in the range domain is called, in the literature, the **1D-FFT** or **range-FFT**. Consider that the ADC sampling data of the chirps is stored as columns of a matrix. The FFT is performed over each column to resolve the targets in range by determining the corresponding frequency peaks. Figure 2.7 serves as an illustration of this concept. If multiple objects are at different distances from the radar, the IF will not be a pure tone, but a sum of tones. Each tone frequency (peak in the range-FFT) corresponds to the distance of each object.

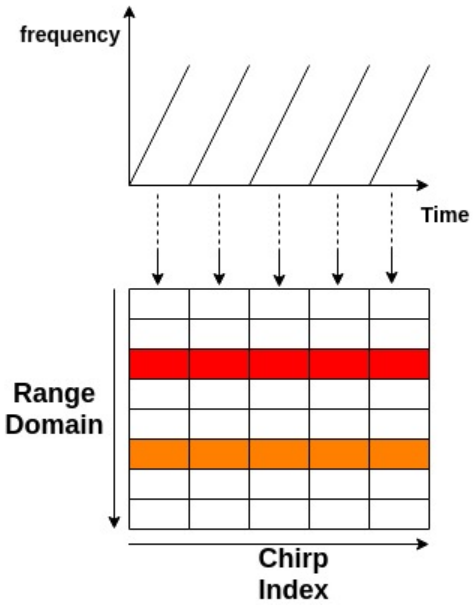


Figure 2.7: 1D-FFT to resolve targets in the range domain. The top image is a frequency-time plot of the emitted chirps. The bottom image represents each chirp stored as a column of a matrix. Each square in a column represents a bin of the DFT. Each column represents a set of frequency bins of the DFT applied to a single chirp, and each row represents the bins with the same index of different chirps of a frame. Each colored square represents the location of a frequency peak. The vertical dimension is usually designated as *fast time* because the duration of the chirp is very small; the horizontal dimension is usually designated as *slow time* because the duration of a set of chirps is very large when compared to the duration of a single chirp. Adapted from [6].

Range resolution

Range resolution refers to the ability of resolving two closely spaced objects, or in other words, how close can two objects be for the radar to distinguish them in the range domain. The frequency of the IF signal results from an FFT performed in the range domain. If two objects are too closely spaced, then the FFT will yield only one peak in the frequency domain. Figure 2.8 serves as an illustration of this. This representation considers a small observation time window.

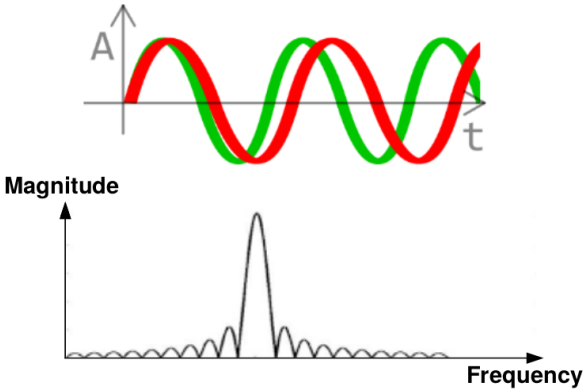


Figure 2.8: Amplitude-time representation of two IF signals with similar frequencies (top) and corresponding peak generated by FFT in the range domain (bottom), considering a small observation time window. Adapted from [6].

To resolve distinct objects in these conditions, the solution is to increase the size of the observation time window, which increases the duration of the IF signal, and allows the FFT to resolve the two objects by determining distinct peaks in the frequency spectrum. Such rationale is illustrated in figure 2.9. This representation considers a larger observation time window.

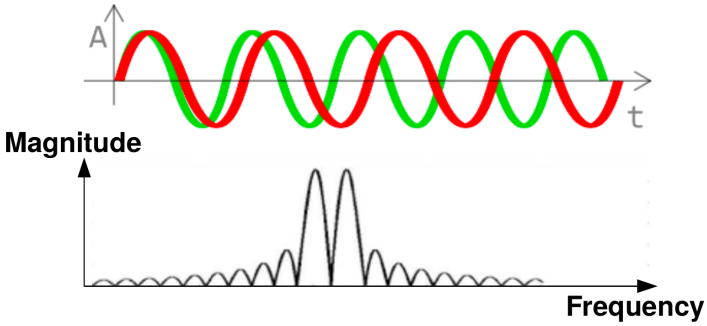


Figure 2.9: Amplitude-time representation of two IF signals with similar frequencies (top) and corresponding peaks generated by FFT in the range domain (bottom), considering a large observation time window. Adapted from [6].

Consider N as the number of points of a DFT, and F_s as the sampling frequency of the

ADC. The frequency resolution (Δf) can be written as

$$\Delta f = F_s/N. \tag{2.5}$$

The total number of points of the DFT is expressed by the following equation, where T represents the sweep time

$$N = F_s T. \tag{2.6}$$

Replacing N in equation 2.5, we derive

$$\Delta f = \frac{1}{T}. \tag{2.7}$$

The variable Δf represents the minimum frequency difference for two objects to be separated in the range domain. Consider now two objects at distance d_1 and d_2 from the radar. The beat frequency of each one is defined as

$$f_{b1} = \frac{2d_1 S}{c} \quad \text{and} \quad f_{b2} = \frac{2d_2 S}{c}. \tag{2.8}$$

Consider Δd as the minimum distance required between two objects in order to enable the radar to distinguish them in the range domain. Likewise, the difference between the frequency of the respective IF signals (Δf_b) will be the minimum frequency difference that permits the two objects to be discriminated. The bandwidth is represented by B. Hence, the range resolution can be derived as follows

$$\begin{aligned} \Delta f_b &= \frac{2(d_1 - d_2)S}{c} \\ \Leftrightarrow \frac{1}{T} = \frac{2\Delta d S}{c} &\Leftrightarrow \frac{1}{T} = \frac{B}{T}\frac{2\Delta d}{c} \Leftrightarrow \Delta d = \frac{c}{2B}. \end{aligned} \tag{2.9}$$

In order to discriminate two objects based on this principle, it is necessary to increase the bandwidth. Assuming the slope remains unchanged, increasing the bandwidth implies to increase the duration of the IF signal.

Table 2.1 shows some typical values for bandwidth and the respective range that is achieved.

Bandwidth	**Range Resolutions**
4GHz	3.75 cm
2GHZ	7.5 cm
1GHz	15 cm
600 MHz	25 cm

Table 2.1: List of range resolutions and the respective bandwidth [6].

Maximum distance measurable

The maximum distance (d_{max}) that can be unambiguously measured by the radar is dependent on the maximum frequency of the IF signal. Because the IF signal is digitized to be further processed on the DSP, the maximum frequency of the IF signal is limited by the ADC sampling rate (F_s). In order to respect the Nyquist theorem, the following condition applies [1]:

$$F_s \geq 2\frac{Sd_{max}}{c}. \tag{2.10}$$

We conclude that the ADC sampling frequency limits the maximum unambiguous range of the radar, which can be defined by the following equation

$$d_{max} = \frac{F_s c}{2S}. \tag{2.11}$$

2.4.3 Velocity estimation and resolution

Velocity estimation

Although a single chirp is enough to determine the range of one or more objects, to measure velocity requires the use of more than one chirp. If a radar transmits two chirps closely spaced in the time domain, the range-FFT corresponding to each one of those chirps will have the same peak on the frequency spectrum, but with different phases. By examining the phase of the IF signal, it is possible to see that the IF signal is very sensitive to small displacements in the object range. A small motion of the object does not change the frequency of the IF signal, at least not enough to be detected using an FFT. However, the resulting peaks will have different phases, and thus velocity can be determined. The difference in phase corresponds to the velocity of the object. An FFT on the sequence of phasors corresponding to the range FFT peaks is called the **2D-FFT** or the **Doppler-FFT** [6]. See figure 2.10 as an illustration.

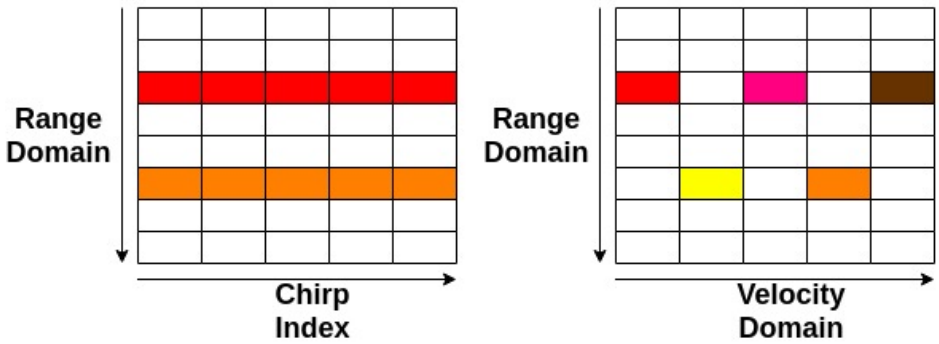


Figure 2.10: 2D-FFT to resolve targets in the velocity domain. Each colored square represents the location of a frequency peak. Adapted from [6].

[1] The Nyquist theorem states that the sampling frequency should be at least the double of the highest frequency present in the signal being sampled. However, the radar devices used in this work perform **complex** sampling, therefore, the sampling frequency may be equal (or larger) than the highest frequency present in the signal being sampled.

Consider the phase difference between two chirps as $\Delta\phi$, the time of flight as τ (defined in equation 2.1), λ as the wavelength, d as the distance traveled by the target, and f as the frequency of the transmitted chirp, and the time interval between two chirps as T_c. $\Delta\tau$ is proportional to the distance difference between successive positions of the target (Δd). The above variables are related by the following equations:

$$\lambda = \frac{c}{f} \quad \text{and} \quad \Delta\tau = \frac{2\Delta d}{c}. \tag{2.12}$$

Based on the previous equations, and knowing that the phase difference is proportional to $\Delta\tau$, it can be derived as follows

$$\begin{gathered} \Delta\phi = 2\pi f \Delta\tau \\ \Leftrightarrow \Delta\phi = \frac{2\pi 2\Delta d}{c} f \Leftrightarrow \Delta\phi = \frac{4\pi\Delta d}{\lambda} \Leftrightarrow \Delta\phi = \frac{4\pi v T_c}{\lambda}. \end{gathered} \tag{2.13}$$

From (2.13) we can easily derive velocity using the phase difference as variable, as shown in the next equation

$$v = \frac{\lambda\Delta\phi}{4\pi T_c}. \tag{2.14}$$

Velocity resolution

In order for an FFT to separate two frequencies, the difference between them, $\Delta\omega$, must be larger than $\frac{2\pi}{N} F_s$ rad/sample, where N is the number of samples. In the present context, N represents the number of chirps used for FFT computation. The frequency is the derivative in time of the phase, thus, we can assume the equality $\Delta\phi = \Delta\omega T_c$. Based on these premises, and considering equation (2.7), the velocity resolution (v_{res}) is derived as follows

$$\begin{gathered} \Delta\omega > \frac{2\pi}{N} F_s \\ \Leftrightarrow \Delta\phi > \frac{2\pi}{T_c N} T_c \Leftrightarrow \Delta\phi > \frac{2\pi}{N} \Leftrightarrow \frac{4\pi v T_c}{\lambda} > \frac{2\pi}{N} \Leftrightarrow v > \frac{\lambda}{2NT_c} \implies v_{res} = \frac{\lambda}{2T_f}, \end{gathered} \tag{2.15}$$

where T_f is equal to NT_c (number of chirps multiplied by chirp time) which corresponds to the frame length.

Maximum velocity measurable

An unambiguous measurement of velocity requires that the modulus of the phase is smaller than π. From equations 2.13 and 2.14 we can derive the expression that defines how the maximum speed of an object relatively to the radar can be calculated, using two chirps separated by a time interval equal to T_c, considering that the expression in 2.13 has to be smaller than π

$$v_{max} = \frac{\lambda}{4T_c}. \tag{2.16}$$

Discriminating two objects at the same distance

We have seen before that two objects that are approximately equally distanced from the radar may not be distinguished. A possible solution is to increase the range resolution by increasing the bandwidth of the chirp. Another option is to use the Doppler-FFT to distinguish these same objects based on their respective velocities. By emitting and receiving a succession of chirps, and if the objects move with different velocities and the difference of velocities between the different objects is larger than the velocity resolution, the Doppler-FFT will detect multiple peaks, and thus, the different objects are revealed.

2.4.4 Angle estimation and resolution

Angle estimation of a single target

A visual representation of the AoA is presented in figure 2.11.

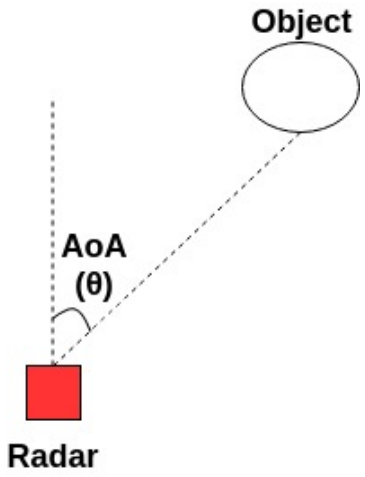


Figure 2.11: Schematic representation of the AoA concept.

To be able to determine the AoA it is necessary to use more than one receiving antenna. As discussed before, a small change in the distance of an object to the radar results in a phase change on the received signal in each receiving antenna. This phase change enables the estimation of the AoA. See figure 2.12.

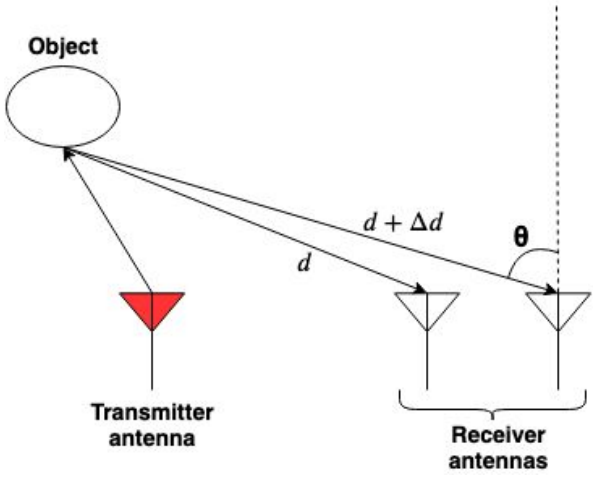


Figure 2.12: Two receiving antennas used to determine the AoA.

Consider the AoA as θ. Consider also d as the distance of an object being detected by the radar to a receiving antenna. Δd is the additional distance between this same object and another receiving antenna (see figure 2.12). Through trigonometric calculus one can derive the following equation:

$$\Delta d = l \sin(\theta), \tag{2.17}$$

where l is the distance between the antennas. The phase change is derived as:

$$\Delta\phi = \frac{2\pi l sin(\theta)}{\lambda} \tag{2.18}$$

Hence, the AoA can be computed as:

$$\theta = \sin^{-1}\left(\frac{\lambda\Delta\phi}{2\pi l}\right) \tag{2.19}$$

Equation 2.18 shows that $\Delta\phi$ is a nonlinear function of θ, and this nonlinearity increases with the angle θ; $\sin(\theta)$ is approximately equal to θ when θ has a small value. Thus, when determining the AoA, the accuracy is higher when θ is smaller. In other words, the farther an object is from the central axis of the radar transmitter, the larger is the uncertainty relatively to its angle.

Angle estimation of multiple targets

When multiple targets are present at the same range and with the same velocity (stationary or not), using the aforementioned method based on one transmitting and two receiving antennas is not sufficient to discriminate them in terms of AoA because both the objects appear in the same frequency bin of the 2D-FFT and the phase of their phasors is combined in both the receiving antennas. In order to resolve them in terms of AoA it is necessary to have a large number of receiving channels (physical or virtual receiving antennas). An FFT on the sequence of phasors corresponding to the Doppler-FFT peaks across the sequence of channels can resolve the multiple targets by measuring the corresponding phases at each channel. This is called the **3D-FFT** or **angle-FFT**, see figures 2.13 and 2.14.

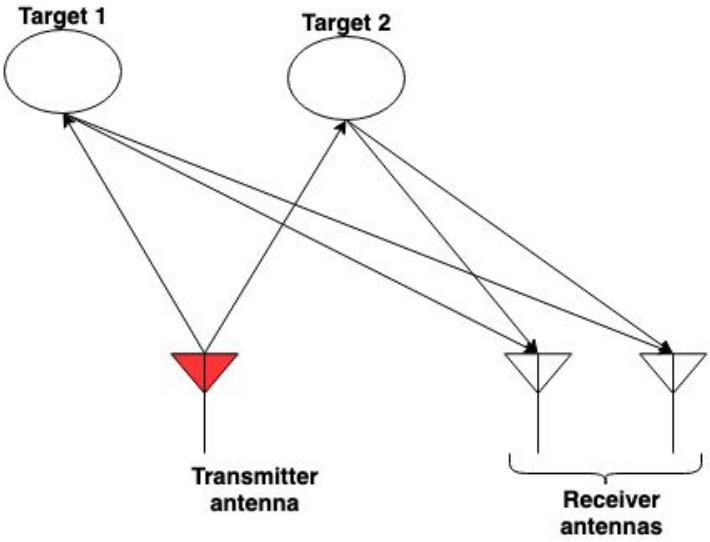


Figure 2.13: Two receiving antennas used to determine the AoA of two objects with the same range and Doppler.

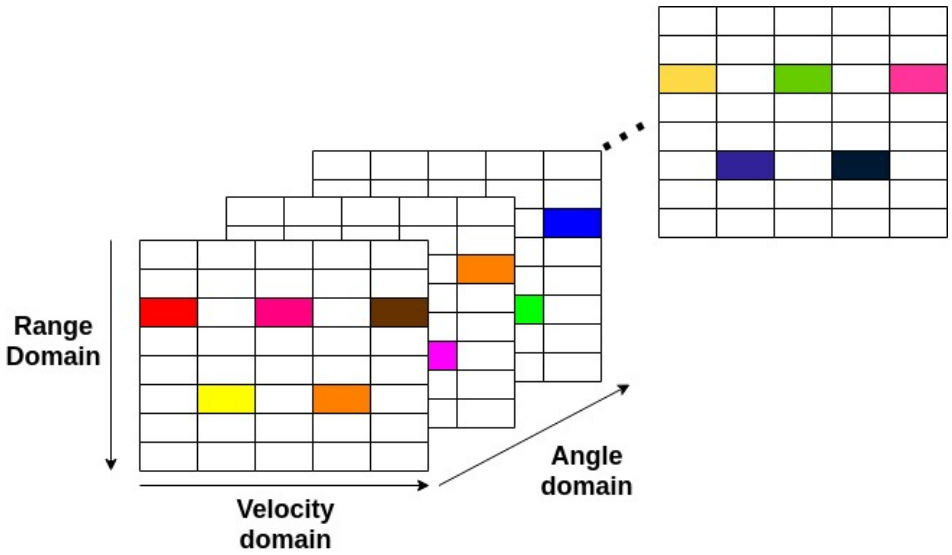


Figure 2.14: 3D-FFT to resolve targets in the angle domain. Each colored square represents the location of a frequency peak. The 3D-FFT is applied over a large number of antennas, therefore, some matrices on the image were occluded (with suspension points) to represent this idea.

Angle resolution

Angle resolution is the minimum necessary angle separation between two objects to appear as separate peaks in the angle-FFT. Angle resolution is defined by equation 2.20

$$\theta_{res} = \frac{\lambda}{N_a l \cos(\theta)}, \text{ for } |\theta| < 90^{\text{o}}, \tag{2.20}$$

where θ_{res} represents the angle resolution, N_a is the number of receiving antennas, l is the distance between the antennas, and θ is the AoA. We conclude that a larger number of receiving antennas yields a better angle resolution, and the angle resolution deteriorates as θ increases.

Maximum angle measurable

As in the case of velocity, unambiguous measurement of angles requires that the modulus of the phase change must be smaller than π. Hence, from equation 2.18 we can establish the following condition

$$\frac{2\pi l sin(\theta)}{\lambda} < \pi. \tag{2.21}$$

Rearranging the inequality above, and solving for θ we can derive the equation that translates the maximum angle, θ_{max}, that may be measured by the device

$$\theta_{max} = \sin^{-1}\left(\frac{\lambda}{2l}\right). \tag{2.22}$$

Thus, the maximum FoV of a radar is defined by the relation between the wavelength of the signal and the spacing between the antennas. The largest FoV possible is $\pm 90^{\text{o}}$, which can be achieved with a spacing between the antennas of $l = \frac{\lambda}{2}$.

2.4.5 Parameters trade-off

Optimally, we wish for a radar device with the best resolution and range in all the different parameters (distance, velocity and angle). However, such is not possible due to the fact that the increase of some parameters often results in the degradation of others. Table 2.2 serves as a summary of the parameters that affect resolution and maximum value of range, velocity and angle.

Requirement	**Resolution**	**Range**
Distance	Bandwidth	Chirp slope and ADC sample rate
Velocity	Frame period	Sweep tine
Angle	Num. of Rx antennas and spacing	Spacing between the Rx antennas

Table 2.2: Parameters that affect both the resolution and maximum value of the device's range, velocity and angle measured.

Consider that we have requirements for range resolution and maximum velocity measurable. For this, we have to fix a value for bandwidth (range resolution) and sweep time (maximum velocity). With this, we are also imposing a value for the slope of the chirp. Hence,

if we also have a requirement for maximum range, the ADC of the device must be able to support the necessary IF bandwidth, otherwise, a trade-off between slope and bandwidth or sweep time has to be made.

Besides ADC sampling rate, other device specific requirements affect the parameters of the chirps. One of these requirements is the memory of the device. In order to permit the application of Doppler-FFTs, the radar must first calculate the range-FFTs and to store all of them in memory for all the chirps in a given frame. This affects the number of chirps that may be used in a frame. The device may also have specific idle times between chirps, which may affect either the number of chirps in a frame, or the frame time.

2.5 Radar-based approaches for people detection and tracking

The present section describes the state of the art in terms of radar-based approaches for people detection and tracking together with an overview of related applications for indoor environments. In general, the approaches described in the literature to detect and track moving persons can be divided into the following categories, depending on the main data types and the signal processing techniques applied:

- Doppler-based detection followed by range/Doppler/azimuth based tracking;
- Range/Doppler-based detection followed by range/Doppler/azimuth based tracking;
- Returned signal strength-based detection.

The next subsections provides a detailed description of each one of the aforementioned methods.

2.5.1 Doppler-based detection followed by range/Doppler/azimuth based tracking

This approach separates moving people from clutter based essentially on Doppler data, representative of the velocity of the target relatively to the radar; static objects can be easily classified as clutter due to their zero-Doppler attribute while non-zero Doppler measurements associated to people (or any other object) motion are used to initiate and propagate target tracks in time.

To enable tracking, one has to perform data association. Recall the concept of frame seen in the beginning of the chapter; a frame can be seen as a snapshot in time of the environment which is under the FoV of the radar. The data association problem consists in applying an interframe correspondence between different objects after detecting them in successive frames. Data association performed for each target relies on the Doppler, but also on information regarding the target position in 2D, including its range and bearing provided by a phased array radar. This approach has demonstrated its efficacy to track multiple people with a single, low-cost MIMO radar but it fails in two relevant aspects:

- It cannot distinguish moving people from other moving targets of approximately the same dimensions (or radar cross section);
- It cannot detect or track a non-walking person even if he/she is performing some task (seated or standing).

This approach is illustrated by the demonstrations provided by the Texas Instruments manufacturer of the xWR1642 and xWR1443 families of FMCW radars; see, e.g., [8] and [4]. Figure 2.15 shows a screenshot of the application developed by Texas Instruments for people detection and tracking [9].

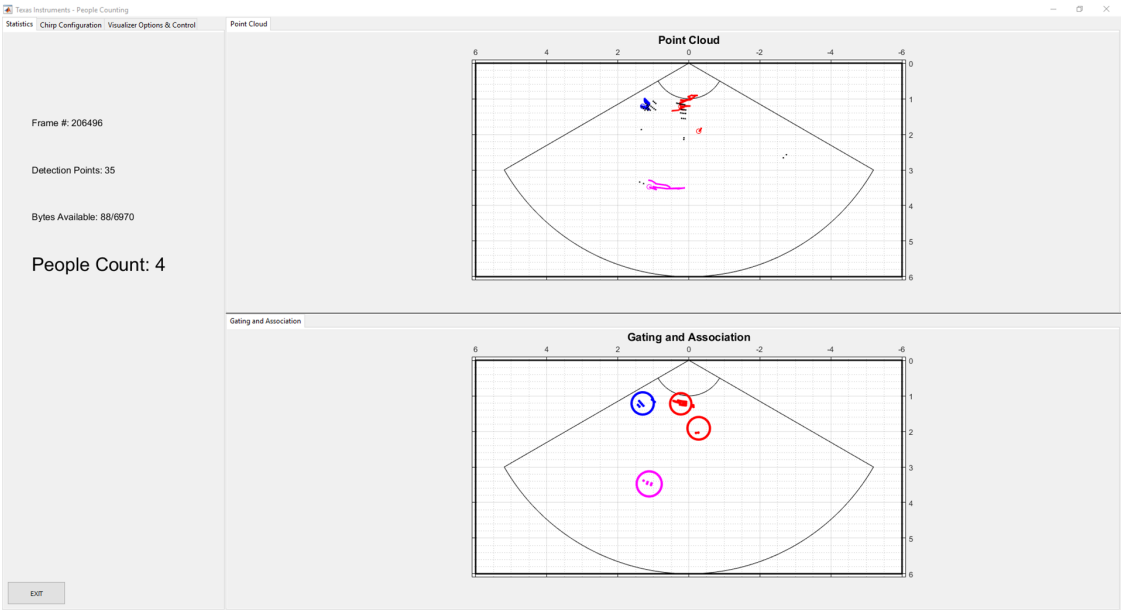


Figure 2.15: Screenshot of the TI application for people detection and tracking. Image from *People Counting User's Guide*, Texas Instruments, 2018.

The former approach is also adopted in the works of [10] and [11]. In [10], FMCW radar technology is used, with a bandwidth of 100MHz, were the purpose is to minimize the effect of radar multipaths, result of the signals reflected by humans that are also reflected (subsequently) in the walls and furniture, and to extract only the echoes relative to human motion. The authors effectively achieve this through Doppler prepossessing. After applying an FFT on successive reflected signals received in each observation time window (the authors call it slow-time) to obtain beat-spectrum, another FFT is applied again, at each spectrum, to estimate Doppler frequency at each range. Then, using a high-pass filter and inverse transforms, stationary targets (walls and furniture) response can be removed. In [11], a pipeline characterized by beamforming, background subtraction and Högbom CLEAN algorithm allows to detect the targets in clutter intensive environments. The tracking stage is implemented by a Markov Chain Monte Carlo Data Association (MCMCDA) together with an Extended Kalman Filter. The authors use a low power MIMO FMCW radar constituted by 8 receiving antennas and 4 transmitting antennas. The application is tested in indoor environments characterized by a metal frame construction, which increases clutter and augments the challenge of both detecting and tracking humans. The tests were performed with different trajectories and some with two people simultaneously walking in front of the sensor, and eventually crossing paths. The results showed that the application was able to successfully track the real targets.

2.5.2 Range/Doppler-based detection followed by range/Doppler/azimuth based tracking

This approach exploits the extended velocity profile (or velocity dispersion) characteristic of human motion. In this context, the term **profile** is used to describe the physical and kinematic properties of a target in the range, angle and velocity dimensions. This exploits the fact that humans do not move as rigid objects but rather present typical oscillating movements of their members and body; for example, there is a typical spreading and contraction pattern of the velocity profile, with a sinusoidal character, in the case of movement of arms and legs of a walking pedestrian. This approach is obviously more robust and holds a larger application potential in most scenarios where humans are required to interact with other mobile agents (including mobile robots, small vehicles, and other machines with moving parts in general). Typically, during a short time interval (for example, a radar frame of a few tens of milliseconds), walking pedestrians present Doppler profiles with multiple, distinct velocity components centred on a single point in the range domain (due to his small velocity, a walking human practically does not change its position during such a short interval). This is in contrast with the compact, point-shaped profile presented by a rigid moving body (a mobile robot or a car) in the Doppler domain due to its constant velocity during an equivalent period of time.

An interesting implementation of the Range-Doppler detection approach is described in [12]. The type of device used in this work is a 24GHz automotive radar sensor with a bandwidth of 150 MHz, and the type of modulation applied allows for a simultaneous high estimation accuracy of range and velocity. The authors apply a Support Vector Machine to classify pedestrians. For such, they propose three feature extraction schemes with increasing computational complexity and latency corresponding to increasing classification success rates. The features used in the simplest implementation are the range-velocity profiles and associated dispersion metrics extracted from the set of reflections obtained in a single radar measurement. A more elaborate solution exploits data from multiple buffered radar measurements. The most advanced implementation complements the radar returns with additional feedback information provided by a Kalman-based tracker. In the three solutions, the true positive rates for pedestrians are, 45%, 54%, and 61%, respectively.

Another potential application of the Range-Doppler detection approach consists of exploiting the patterns of hands movements of a person in order to detect her presence and even to interpret his gestures (for example as remote commands for a robot). This method is illustrated by the work of [13]. In this work, a hand gesture recognition system is developed using a ML algorithm fed with data from a radar to permit steering small mobile robots without direct human operation. The authors use an AWR1642 radar device from TI, using a bandwidth of 2 GHz, which yields a range resolution of approximately 7.5cm. The velocity resolution is 0.058m/s. The clutter was filtered by bypassing the objects with low peak value (intensity). Because learning models demand that the data fed into the model is the same size, and the point cloud retrieve by the radar may have a variable number of points, the coordinates of each target are averaged into a single point. The several points in gesture recognition are combined with each other yielding the average shape of the gesture. The gestures considered are: swipe left, swipe right, push, and pull. The ML implementation of gesture recognition relies on decision trees providing a mean success rate (true positives) of 96%.

2.5.3 Returned signal strength-based detection

This approach exploits measurements of signal intensity returned from the target that may be used directly, as in the case of classification schemes based on the computed absolute RCS of targets, or indirectly by exploiting the relative intensity of the echos corresponding to distinct scatterers of the same target. RCS is a property of the target's reflectivity, commonly used to detect and classify airplanes in a wide variation of ranges. In principle, the technique can be applied to distinguish terrestrial targets, because cars, motorcycles, bicycles, and pedestrians present distinctive RCS values. In practice, the method may be difficult to apply due to the large dependence of RCS measurements on the orientation of the target relatively to the radar; see, e.g., [14]. To the best of our knowledge RCS has not been applied before to detection and tracking of people. The aforementioned indirect method is illustrated by the collection of methods proposed in [15] that exploits the 2D radar intensity image of a small object in order to derive a model of its bi-dimensional shape. This is combined with measures of its Doppler spectrum distribution to build a feature vector used by a fuzzy membership function that maps the acquired features onto the corresponding target class. As such, the solution integrates techniques pertaining to the two above-mentioned approaches in order to implement a robust classification system. The success rate (true positives) achieved with this system varies between 40% and 95% depending on the orientation of the pedestrian trajectory (from radial to lateral) relatively to the radar axis.

2.6 Micro-Doppler for people detection

Other related works (e.g. [16]) have focused on the use of **micro-Doppler** data to detect human motions. Micro-Doppler (effect) is defined as phase shift on the returned radar signal, that is a result of the target's micro-motion dynamics, such as mechanical vibrations. Despite the high potential of the approach, exploitation of this effect with the radar kits on hand is not suitable for the current application given the demanding signal processing power required to resolve the micro-Doppler components induced by the targets while simultaneously dealing with the envisaged sensing requirements of maximum detection range, and maximum unambiguous velocity of different moving objects. Another issue with this approach is raised by the inherent vibrations of mobile robots (carrying the radar sensors) that induce micro-Doppler effects susceptible of masking those of the targets.

2.7 Hardware and software tools

2.7.1 Board Texas Instruments AWR1642BOOST-EVM

The board used was the AWR1642BOOST-EVM (figure 2.16) from Texas Instruments, that possesses an AWR1642 single chip 76-GHz to 81-GHz automotive radar, and integrates also a C67x DSP and ARM® Cortex™-R4F MCU. More details of this board are available at the following link: `http://www.ti.com/lit/ug/swru508b/swru508b.pdf`. The AWR1642 board possesses buttons and leds which allow the programmer to quickly develop a simple user interface; it has XDS110-based JTAG with serial-port interface that is used for flash programming, a UART-to-USB to control, configure, and provide real-time data visualization. It also has a CAN connector that enables direct interface with car units.

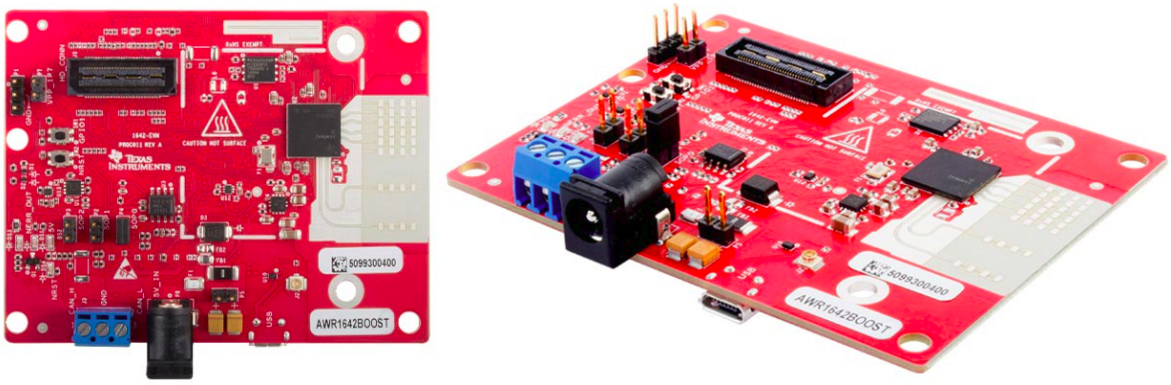


Figure 2.16: Board AWR1642BOOST-EVM from Texas Instruments. Image from `http://www.ti.com/tool/AWR1642BOOST`, online, accessed July 2019.

Other alternative devices

Other alternative devices are common in the market. Texas Instruments itself has a panoply of other similar devices with different valences. Examples are the AWR1443, the AWR1243, and AWR1642BOOST-ODS. The devices are similar between them and also similar to the one used in this work. The differences rest on technical details of the boards such as the antennas positioning and the SDK used. The selected board was used because it had already been demonstrated in basic demos by TI to be adequate for people detection, and to be a a versatile board in terms of configuration for multi-purpose applications.

2.7.2 Matlab

The present work was partly accomplished using Matlab software. Specifically, the version of Matlab used was **Matlab R2018b**, with the *Robotic Systems Toolbox* installed.

Matlab is a programming language and computer software that is particularly useful for simulation of mathematical calculus. The use of native data structures such as matrices, and the built-in functions for plotting data make it an adequate candidate to be used on the stage of processing the collected data. The *Robotics Systems Toolbox* enabled the processing of data acquired while using ROS.

2.7.3 ROS

ROS, which stands for Robot Operating System, is an open source middleware for robotic systems. ROS is distributed under different versions. The version used in this work is the *melodic* version, supported by the build system *catkin*. The ROS package **rosbag** is particularly useful. This package allows the creation of files with extension "bag", which are files were it is possible to store the data received with the respective timestamps. Consequently, this also permits to playback the data with the correct timing and enables the offline processing of the data. ROS software was used to store the data captured and feed it to the preprocessing algorithms.

2.7.4 UniFlash 4.4.0

UniFlash is a software tool from TI that allows to transfer the binaries of the applications, that one wants to execute, to the flash memory of a wide set of TI boards.

2.7.5 mmWave SDK Demo

The mmWave SDK Demo is an out-of-box application developed by TI that processes radar data - performs FFT to resolve range and Doppler; if several antennas are connected, it performs beam forming to solve for angle of arrival; applies CFAR for noise deletion, and sends the data through TLVs to the host computer, via the USB port. TLV is a type of data/message encoding, where the data is divided in three blocks: the *type* block, the *length* block and finally the *value* block. The *type* block has a fixed length and defines the type of information the message represents. The *length* field is also of fixed length, and defines the length of the value field. Finally the *value* field contains the body of the message itself. The mmWave SDK Demo is often used in conjunction with the application mmWave Demo Visualizer to see in real time the result of this processing.

The way the data is processed by this application is defined in a configuration file (extension ".cfg"). This file allows the user to send commands to the target board.

Chapter 3

Machine Learning applied to radar data

This chapter presents the main ML applications in the context of radar systems. First, the key concepts of ML are clarified, and then the ML techniques applied to radar data are described.

3.1 Basic principles of machine learning

With the developments on informatics and communication systems, the flow and quantity of data, particularly on the Internet, has increased tremendously. Examples of this are the fact that Walmart processes 1 million transactions per hour, and has databases containing more that 2.5 petabytes (2.5×10^{15}) of information; YouTube has one hour of video being uploaded to its platform every second [17]. This enormous amount of data calls for methods that can substitute the humans in the processing stage and to be able to extract patterns from these data completely autonomously. Associated with this, is the problem of having tasks for which it is not feasible to code a solution. Examples of this are spam filtering, face and speech recognition, text translation, and so forth. Hence, it is necessary to follow a different approach. ML allies the existence of such large amounts of data with the need to develop algorithms that can find solutions to these problems in an automated manner, and to make predictions of future data using knowledge acquired from training data. As stated in [17], the goal of ML is to discover methods that can automatically detect patterns, and use them to predict future data or other outcomes of interest. ML intuition comes from the fact that virtually all learning problems can be formulated as (complex) mappings between inputs and outputs. Mathematically speaking, we search for a "good" function $F : X \times y \rightarrow O$, where X is the set of all possible inputs, y represents the labels, and O the set of all possible outputs. For the aforementioned examples, we want to map an email to spam or not spam, an image to the identified faces, a speech signal to a text sentence, and so on [18].

The question is now how does a computer learn these mappings. The learning method depends on the data that is used. If the data samples contain both the input and the output, supervised learning can be applied. If only the inputs are available, then unsupervised learning can be applied. For other cases, where there is no direct access to the correct output, but to a measure of the quality of a particular output $o \in O$ following some particular input $i \in I$, then reinforcement learning is applied [18].

3.1.1 Supervised Learning

In supervised learning, we have training data (that will be used to train the agent) encoded as pairs (i,o). The function $F : I \rightarrow O$ is often dependent in a (sometimes large) set of parameters, and the learning goal is to adjust these parameters in order to fit the data. To further understand the concept of supervised learning, and of ML in general, some concepts need to be clarified:

- **Feature**: Is a characteristic of the data, to be taken into account by the learning method. For the case of spam filtering, it might be the type of words in the email, the number of images, etc.
- **Data sample**: Is a sample of the data that is being used to train the learning model or to test its ability to classify the data. For the example of spam filtering, a data sample is a single email message.
- **Label**: Is the output for a data sample. For the example of spam filtering, it is *spam* or *not spam*. Often, the term *class* is used as a synonymous.
- **Dataset**: Is the collection of all data samples, along with the respective labels for the case of supervised learning, or without them, for the case of unsupervised learning.

Learning models are methods to determine the class of functions from the dataset, while the learning algorithm is how the parameters of the equation are determined to best fit the data. In the context of supervised learning there are two types of predictive models: **classification** and **regression**.

Classification

Classification is used in cases where the label is defined by a discrete (nominal or categorical) set of values. Graphically speaking, for the case of classification we want to find the line that separates the several classes at play. A visual representation of such line is presented in figure 3.1.

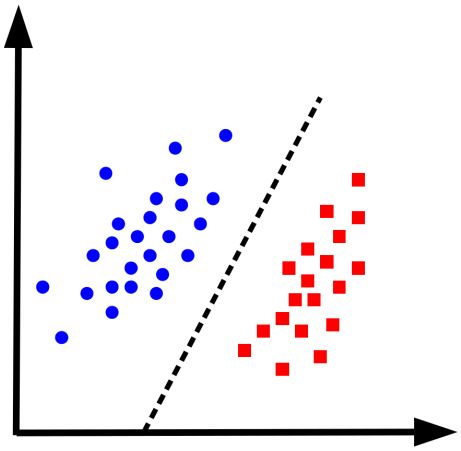


Figure 3.1: Classification applied to a dataset with two distinct classes (and two features).

If the number of classes is equal to 2, this is called binary classification. When there are more than 2 classes, then it is called multiclass classification. When the labels are not mutually exclusive, then we are facing a multi-label classification problem, where multiple labels may be assigned to a single data sample.

Regression

Regression is used when the label is defined by a continuous interval of values. Graphically, we want to derive the line that best translates the evolution of the data. In figure 3.2, the line generated aims at best fitting all the outputs for the respective inputs. According to this model, further inputs will have an output that rests around the regression line.

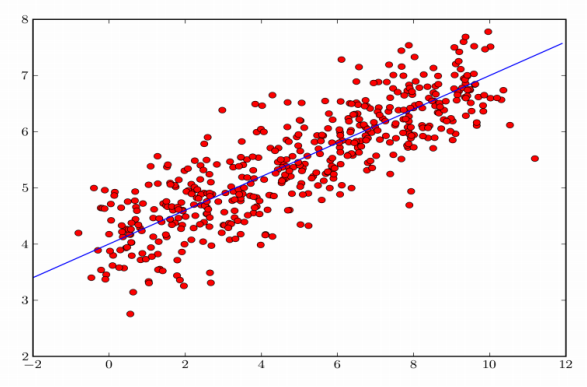


Figure 3.2: Linear regression applied to a dataset. Source [19].

An example of regression is to predict tomorrow's stock market price given current market conditions and other possible side information [17]. A variant of regression, known as ordinal regression, occurs when the label space has some natural order, like grades A to F.

3.1.2 Unsupervised Learning

In unsupervised learning, also referred to as knowledge discovery, the goal is to determine if there is any relation between the different features, and to segment the data in different groups, or, in other words, cluster the data in groups, and detect outliers [18]. The classes (or clusters) in which the data is divided are not explicit *a priori*, and there is no obvious error metric, unlike in supervised learning where we can compare the output or estimation of the model with the real value [17]. Figure 3.3 shows the data before the application of unsupervised learning, and the result after. As shown, points near each other were assigned to the same cluster, while far away points were assigned to different clusters, which resulted in two different clusters being discriminated.

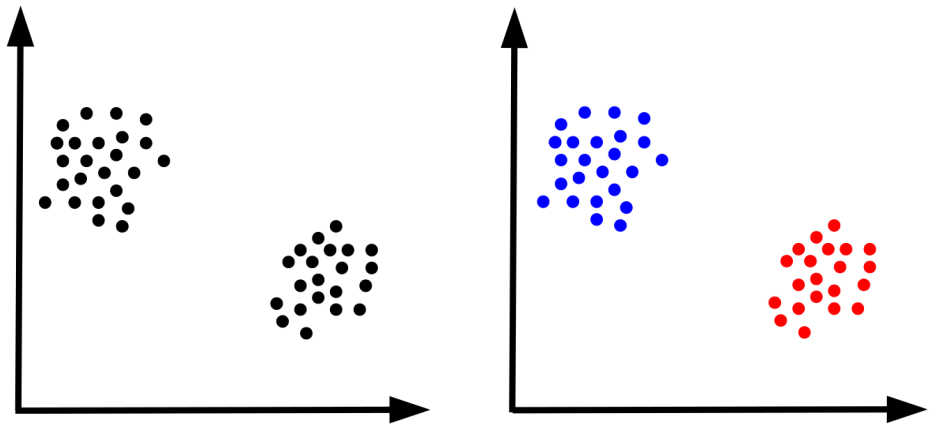


Figure 3.3: Data before being clustered (left) and after being clustered (right). The two clusters generated are represented with different colours.

Unsupervised learning is closer to the way humans and animals learn. It is also more widely applicable than the supervised version, due to the fact that labeled data is expensive and requires a human expert to label it [17].

3.1.3 Reinforcement Learning

As stated earlier, in reinforcement learning there is no direct access to the correct output for a given input, but there is however a measure of the quality of that output. This is often referred to as the reward, which can be positive or negative [18]. Reinforcement learning functions as a loop, where the intelligent agent interacts with the environment - percepts the state of the world, executes the action it considers to be the most appropriate one, and is then rewarded. Note that the notion of reward is different from its original meaning. As said before, this reward can be negative. The goal of the agent is to perform the actions that maximize its cumulative reward over time.

The values of the reward are related to the purpose of the task at hands. Consider a system that is learning to play some game, for example chess, and receives negative reward when it looses a piece, and positive reward when it eliminates a piece from its opponent.

3.2 Machine Learning techniques applied to radar data

A surge of interest in ML for radar has been witnessed in the last few years. This is illustrated in the recent special issue [20] containing a collection of papers that illustrate the state of the art on ML applications to radar and radio communications; the most popular classes of algorithms employed are **deep learning** and **reinforcement learning** but some classical approaches such as **support vector machines**, are also included. Among the problems addressed in the papers are: learning of operational parameters of radio-frequency (RF) devices and communication channels (RF interference, target, and clutter effects) for improved medium access control; radar waveform classification; identification and adaptation of network topology. The special issue also addresses the need for publicly available repositories of sample RF data for ML training and validation and presents some pioneering efforts developed towards the constitution of such knowledge base. However, the number of ML applications devoted to object recognition and tracking is still scarse. The operation of radars installed on mobile robots, particularly in indoor environments, is also still poorly covered in the literature. Some notable exceptions are [21], [22], and [23].

A comprehensive revision of the use of radar systems together with the application of ML for people detection tasks can also be found in [24]. The authors present the main applications of radar in indoor environments, and detail different types of radar feature extraction and types of features. Examples of this are **handcrafted features**, such as physical features like the Doppler bandwidth (characteristic of the target motion) or the stride rate; speech-inspired features like the mel-frequency are also a possible type of feature to extract. Other type of feature extraction is **data-driven feature learning** like the application of Principal Component Analysis (PCA), a technique of data transformation that allows to derive the directions in feature space with the most variance. In that manuscript the authors study the performance of different Deep Neural Network learning models, such as autoencoders and Convolutional Neural Networks.

3.3 Software tools

3.3.1 Python

Python is an interpreted, high level object oriented programming language, that has been widely used because of its simple syntax. The version of this language, that was used in this work, was *Python 3.6*. **scikit-learn** (version 0.20.3) is a popular open source python library focused on ML. It provides an extensive set of APIs for data mining and data analysis. This tool was used to implement and test the learning models presented in this work, and also to implement the feature extraction methods on the radar data.

Chapter 4

People detection with static radar

To discriminate a moving person from other targets, we intend to establish parameters that allow to distinguish the classic motion of people from the motion of other targets, and use them to feed and train classification models. We rely on the Doppler property of the data for this. The purpose is to determine how the Doppler varies along time and to try to visualize patterns of motion. In this context, as it is possible to store the properties of each point of the point cloud along the several frames, it is intended to see the distribution of the values of Doppler of the points, while following a specific path. To visualize this distribution, the data is plotted in 3D histograms, with the first dimension being the Doppler value, the second dimension the time instant or frame, and lastly, because it is a histogram, the third dimension is the number of points within a given Doppler range (bin) at each instant of time.

To analyze this pattern and to collect the data needed, several tests were performed. In the following subsections of this chapter the setup and the tests are thoroughly described, as well as the mechanisms used to segment the point cloud and store the data. A visual analysis of the data is also performed.

4.1 Scenario configuration

Firstly we define the configuration used for data acquisition. The tests with the radar were performed in a wide open area, free of obstacles. The host computer communicates with the sensor via a Raspberry Pi 3 (model B+) connected directly to the sensor. The radar is placed at a height of 1.9 meters (in order to stand taller than most people) with a downwards inclination of about 10 degrees. This placement of the device aims at minimizing the number of people that can be occluded by other people present in the area. Figure 4.1 shows the physical setup of the radar device placed on the tripod.

In table 4.1 details of the chirp used can be found.

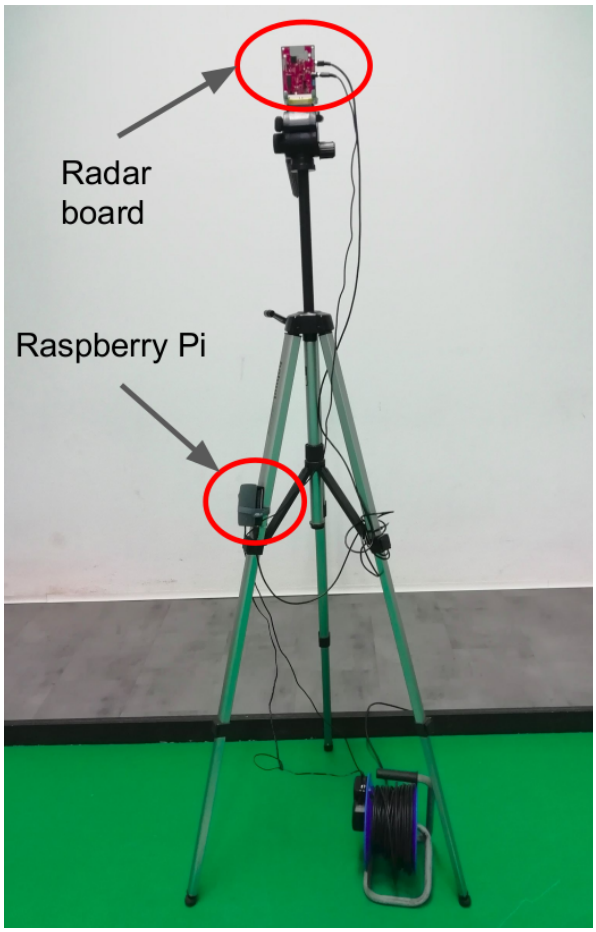


Figure 4.1: Radar device placed on the tripod, connected with the Raspberry Pi device.

Chirp Parameter (Units)	**Value**	**Chirp Parameter (Units)**	**Value**
Start Frequency (GHz)	77	Maximum unambiguous range (m)	5
Slope (MHz/us)	60	Maximum radial velocity (m/s)	5.2936
Samples per chirp	128	Azimuth resolution (degrees)	14.5
Chirps per frame	256	Velocity resolution (m/s)	0.0827
Frame duration (ms)	50	Number of transmitting antennas	2
Sampling rate (Msps)	2.5000	Range resolution (m)	0.0488
Bandwidth (GHz)	3.0720	Number of receiving antennas	4

Table 4.1: Chirp parameters. A similar table to this one can be found in the Texas Instruments documentation.

4.2 Tests performed

In order to enable a learning model to distinguish people from other targets, representative data sets of both classes of targets have to be provided. Therefore, tests were made with people, but also with other targets, such as dummies with a dimension close to that of an adult. Furthermore, with the intent of making this application as inclusive as possible, people with limitated mobility were also taken into account by performing tests with a person driving an automated wheelchair.

Each elementary test in this experiment consisted of a target moving in a specific direction. In the test field, 4 types of directions were considered: from one corner to the opposite one (left to right and right to left); from the top centre of the field to the bottom centre; from the lateral centre of the field to the other lateral centre. For each direction, the two ways were considered, yielding therefore a total of 8 different trajectories. For each trajectory, 10 different tests were performed, for a total of 80 tests for each specific target. Figure 4.2 shows a schematic representation of the trajectories traversed by the targets used in the tests. Notice that, although the trajectories are represented as straight lines, while performing the data acquisition tests, it is not guaranteed that the agents will travel straight lines. This is only an approximation of the real trajectories.

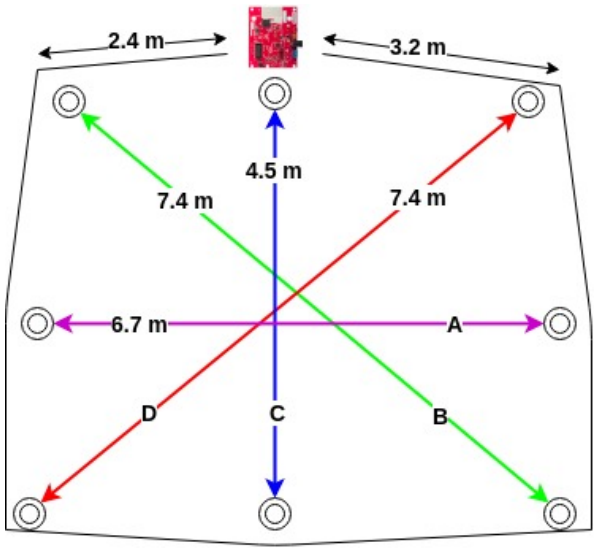


Figure 4.2: Schematic of the test field with the trajectories considered represented by bidirectional non-black arrows, and the limits of the FoV represented by circumferences. The length of each path is represented in meters.

Three different types of targets were considered: person walking, person in a wheelchair, and two different dummies. A picture of one of the dummies is shown in 4.3, as well as the wheelchair and the radar device.

Figure 4.3: Wheelchair (left) and one of the dummies (right) used in the tests.

4.3 Point cloud retrieved from sensor

Using the AWR1642 board, for each point of the point cloud, the device provides (among others) the following data that is used by the current application:

- **Range**: the radial distance (meters), applied in the clustering of the point cloud;
- **Relative intensity**: the relative intensity of the signal in that specific point, used in the computation of the RCS (feature extraction);
- **Radial velocity (Doppler)**: in meters per second, used in the computation of the velocity histograms to be used in the classification stage (feature extraction);
- **Angle of arrival (azimuth)**: the angle that the point makes with the central axis of the radar transmitter, in degrees, used to compute the projection of the radial distance in the axis of the radar coordinates frame. Applied in the clustering of the point cloud.

In any frame, the points of the point cloud closer than 0.5 meters from the sensor are eliminated. The reason behind this is that the device can not correctly distinguish points too close to the device, hence, bellow a range of 0.5 meters, the radar is prone to detect a lot of clutter.

After retrieving the point cloud and the associated frames and recording these data, we can playback the tests made and visualize the point cloud. Figures 4.4 and 4.5 show two consecutive snapshots of two point clouds retrieved from different frames on the same test.

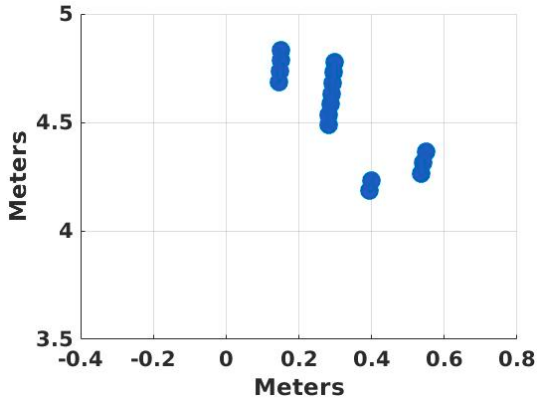


Figure 4.4: Snapshot of the point cloud of a person walking in front and towards the radar (path C), with the right side of the point cloud more advanced than the left side.

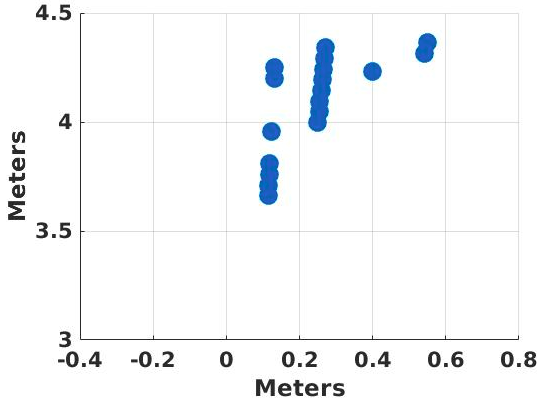


Figure 4.5: Snapshot of the point cloud of a person walking in front and towards the radar (path C), with the left side of the point cloud more advanced than the right side.

The test consisted in a person walking in front and to the radar (position $(0, 0)$ in the graph.). The first and second snapshot are a few frames apart. The first snapshot shows a right side to be more advanced (closer to the horizontal axis) than the left side. This is resultant from the movement of the arms; in this case, the left arm (not the right arm, remember that the person is moving to the $(0, 0)$ position) of the person performing the test is advanced relatively to the body, and the right arm is lagging behind the body. Opposite to this, in the second snapshot the left side is more advanced, which results from the fact that the right arm is advanced relatively to the body and the left arm is lagging behind the body. This means that we can visually observe the pendulum like motion of the arms.

The point cloud of a person walking should have a periodic dispersion (when the arms and legs are further away from the body) and contraction (when the arms and legs are in line with the rest of the body) of the points. Other targets, such as robots, which move as a block, do not have this periodic dispersion and contraction of the points of their point cloud. Instead, the points of their point cloud will always be concentrated around a central position. Figure 4.6 serves as an illustration of this.

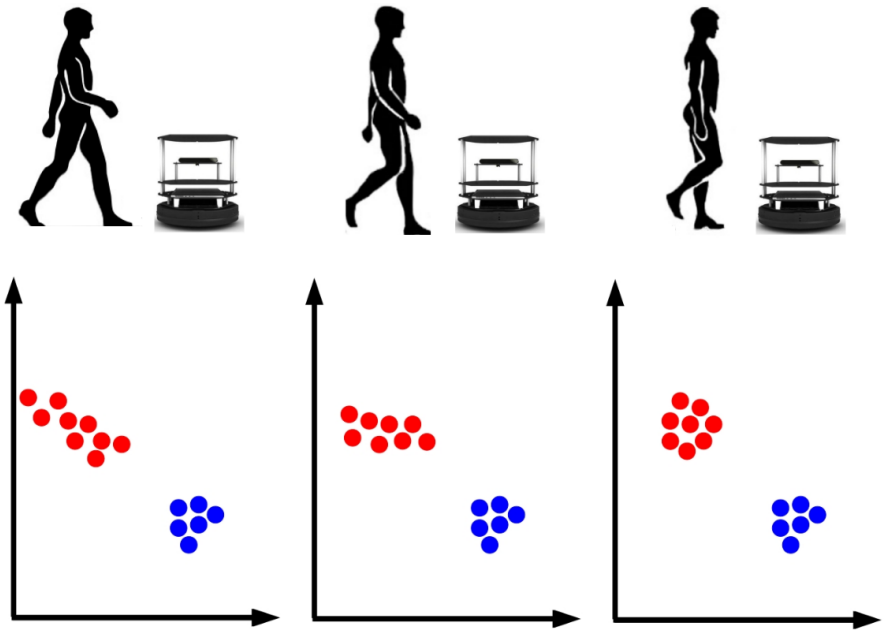


Figure 4.6: Illustration of a person walking and a robot moving and the respective point clouds.

4.4 Clustering - DBSCAN

When performing measurements using the radar, clutter may coexist with the point cloud of a target of interest. It is then important not only to detect this clutter, but to eliminate it, ensuring that these points will not be included in the creation of the dataset. The obvious way to segment the point cloud corresponding to a target is by using an appropriate clustering method. Clustering methods have the advantage of being able to determine clusters with an arbitrary shape (spherical, linear, elongated, etc), and such is the case of the type of point clouds examined in this work. The method chosen in the current implementation was DBSCAN (Density-based spatial clustering of applications with noise) [25]. DBSCAN is a clustering method that relies solely on two base parameters: minimum number of points and maximum distance between them for a set of points to be considered a cluster. Henceforth, minimum number of points will be designated as *MinPts* and the maximum distance will be designated by *Eps*. DBSCAN intuition follows the rationale used by humans to visually detect clusters in a point cloud. Take, for example, figure 4.7; the figure contains 3 images of point clouds, each one with a different number of clusters, with different forms and densities. A human being can rapidly identify all of the clusters in the images, because all of them have typical densities that are constant through the cluster.

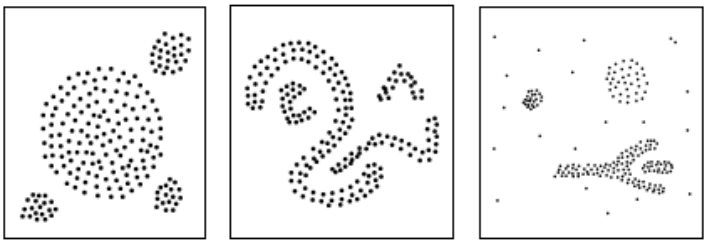


Figure 4.7: Three different point clouds with clusters of different forms and densities. Source [25].

This is exactly what DBSCAN tries to uncover. The key ideas of DBSCAN is that, for each point, in a given radius (or Eps) the neighborhood has to contain at least a minimum number of points (*MinPts*), or in other words, the density in the neighborhood of a given point has to exceed some threshold. The shape of a cluster is defined by the choice of the distance function used, as well as the value attributed to the Eps and MinPts [25]. Note that the DBSCAN algorithm can deal with any distance function (euclidean, Manhattan, Chebyshev, etc) and with any dimensional space (1D, 2D, 3D, and so on). According to the reference on [25], there are 6 definitions one has to consider in order to understand DBSCAN:

1. The Eps neighborhood of a point q is defined by every point that stands at a distance from q that is smaller or equal to Eps;

   DBSCAN algorithm makes a distinction between **core and border points**. Core points are those that are located in or near the center of a cluster. These points have a high density of points on their surroundings. Border points, as the name suggests, are points located near the border of the cluster, and so, are not surrounded by a density of points comparable to the core points. As such, in order for border points to obey to the aforementioned condition, we would have to set the value of MinPts to a particularly small value, which won't be representative of the cluster. The next definition aims at solving this issue, while using an adequate value for MinPts.

2. A point q is directly **density reachable** from a point p if the distance from q to p is smaller or equal to Eps, and if the Eps neighborhood of p has at least a number of points equal to MinPts;

3. A point q is density reachable from a point p if there is a chain of points p_1, p_2, ..., p_n, such that p_{i+1} is directly density reachable from p_i;

4. A point p and q are **density-connected** if there is a point o such that both p and q are density reachable from o;

5. Considering a point cloud, a cluster is defined as a non-empty subset of points that obey the two following conditions:

   For any point p and q, if p belongs to cluster C, and q is density-reachable from p, then q also belongs to C;

   If a point p and q belong to C, then p is density connected to q.

6. Noise is defined as any point in the point cloud that does not belong to any cluster.

4.4.1 Algorithm inner working

The way the algorithm works, and how it is implemented particularly in the context of this dissertation, is described in this subsection. The algorithm starts by choosing a point randomly. Then, all the points located at a distance from the original point smaller or equal to Eps are counted. If the counted number of points is larger or equal to MinPts, then the original point is considered part of a cluster, and their neighbors are also part of that cluster. After, the points in the vicinity of each of the neighbors of the original point are determined. This means the cluster is being expanded, until the border points are reached. At this moment, the cluster is defined. This process is applied to the other points of the point cloud which have not been attributed to any cluster. When all the points have been processed, the ones that were not associated to any cluster are considered noise points.

The number of clutter points in the point cloud is unpredictable, and nothing can assure that, while using the aforementioned parameters for DBSCAN, there will not be more than one cluster detected. To deal with this issue, because the tests are performed with one target at a time, we consider that even if multiple clusters are detected, the cluster of interest will be the one that has the largest number of points (the largest cluster), and we will only consider for analysis the data from the points that belong to that same cluster.

4.4.2 Choosing the values for Eps

There are several approaches to assign the best values to the two parameters that define DBSCAN, and the vast majority of them are empirical. The parameter MinPts is defined according to some established rule, based on experiments, but Eps can be calculated using a well defined theoretical rationale. Inside the several clusters, the distance between a point and its closest point is constrained in a small interval of values. However, the distance between a noise point and its closest point is very large when compared to the previous case. Figure 4.8 depicts a graph that shows how the distance increases, from the point that has the smallest distance to its nearest neighbor, to the point that has the largest distance to its nearest neighbor. Until the first 91 points the Eps almost does not increase, which may indicate that the points are all located inside clusters, where the distances have small variations. From this point on, the distance increases rapidly, which indicates that those points are now in the domain of the noisy points. The chosen Eps should be the one that is better able to allow the detection of the possible clusters, and make a distinction between them and noise. Therefore, in this case, the Eps should be between the one of the 91th point and the 92th point. This part of the graph is defined as the "elbow" of the graph. Notice that this approach is only effective in simplified scenarios, such as the one considered in this work. In more complex, cluttered scenarios, where the clusters are not so clearly separated, the elbow of the graph may not be easy to identify.

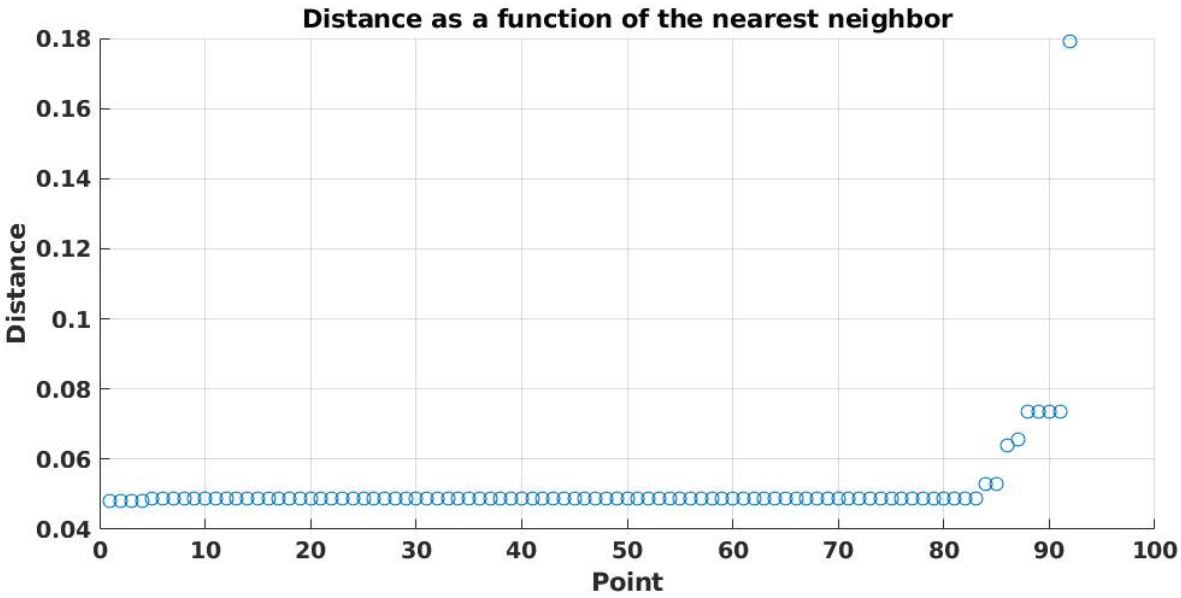


Figure 4.8: Distance of each point to its closest neighbor sorted in ascending order.

To determine an adequate Eps, we should then calculate the elbow of the graph. This can be seen as a good approach, because it dynamically calculates the most well suited Eps. However, because the number of points and their disposition on the radar FoV changes from frame to frame, Eps needs to be re-calculated in every frame. This implies an increase of computational complexity which decreases execution speed and can be particularly problematic in the context of a real time application. For this reason, in the present application, instead of dynamically calculating the Eps, the parameter values were chosen based on practical experiments. The values are $MinPts = 5$ and $Eps = 0.5$.

4.5 Data acquisition and feature extraction

The velocity of the points belonging to the point cloud of a frame are processed to build a Doppler histogram. Based on this, the target kinematics is represented as an image by the concatenation of the Doppler histograms of the successive point cloud of all frames. For a better visualization, the values of the bins are color coded. In this work, the term **first order features** is used to define a type of data that is extracted through basic signal processing of raw data samples. Based on this definition, the present method considers two types of first order features: the Doppler bins resulting from the histogramic representation and the RCS. Based on this approach, the data acquired in each test is transformed into a single dataset constituted by 151 features (150 velocity bins plus one value for RCS) per frame and with a varying number of data samples equal to the number of frames. A visual representation of an offline implementation of this data processing scheme can be found in figure 4.9.

Figure 4.9: Schematic representation of data acquisition and feature extraction process. Notice that the attribution of a label to a frame is supervised by a human agent.

In the current experiment, the complete dataset consists of a total of 24455 frames. Out of these 24455 frames, 11285 correspond to people moving, and the remaining 13170 correspond to other moving targets. Table 4.2 shows the distribution of frames for the different paths and different agents. *Mannequin 1* is the term used to define the mannequin shown in figure

4.3, and *Mannequin 2* is the term used to define the other dummy.

	People walking	People in wheelchair	Mannequin 1	Mannequin 2
Path A	2111	1660	1756	1240
Path B	1158	1454	1819	1334
Path C	1264	1561	1992	1810
Path D	940	1137	1591	1628

Table 4.2: Distribution of frames for different paths and agents. *Mannequin 1* is the mannequin used (shown in figure 4.3) and *Mannequin 2* is the other dummy used.

4.6 Representation of movement

The data is recorded using ROS, and is stored in a file with extension ".bag", which allows the data to be later processed and also played back with the original time intervals. The pattern of velocities of the different parts of the body of a walking person can be summarized by a histogram representative of the distribution of the Doppler measures acquired in each frame. A histogram is then a snapshot of the kinematic state of a person at a specific time. The concatenation of the histograms obtained for successive frames provides a 3D histogram in which the first dimension represents Doppler, the second dimension the frame (time), and the third the number of points. The limits of the velocity axis, $+/-6$ meters per second, were chosen considering that in the indoor scenario under study, targets will not move with higher velocities. As such, the histogram is constituted by a set of 150 bins in order to obtain a velocity resolution of about 0.08 m/s, which is the velocity resolution of the radar device and the one used with the present device configuration. In each frame, the total number of points is normalized (to 1), which results in the summation of the points across the several bins in the same frame yielding 1. According to this representation, figure 4.10 shows the typical pattern of a person walking. In this specific case, the person was moving in front and towards the sensor (path C).

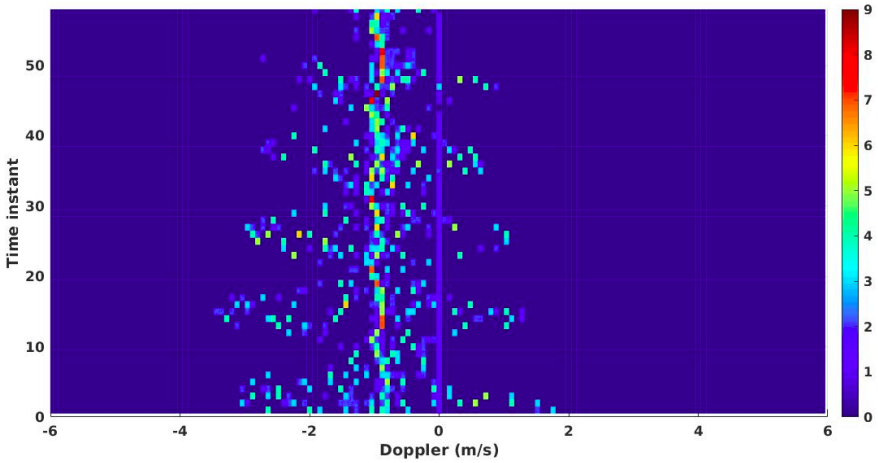


Figure 4.10: Histogramic representation of Doppler measures of a person walking in front and towards the radar (path C). The colour bar on the right of the histogram represents the absolute number of points.

The plot clearly shows the dispersion of velocity values associated to the walking person. As can be seen in figure 4.10, the points are centered around the -1 m/s value, but there is a dispersion both in the negative and positive direction with a periodical variation. This is in accordance with the motion of a person. The majority of the points, which are a result of reflections on the torso, have all the same speed. The points corresponding to the arms and legs have different velocities from the rest of the body. As the arms travel in opposite directions at each moment, they generate a dispersion both in the positive and negative direction. For other directions, such as diagonal or transversely relatively to the radar (paths A, B and D), the pattern is similar, although with different amplitudes.

Differently from the former, the motion pattern of a non-person target, for example a mannequin, presents a reduced Doppler variability. As the mannequin moves as a block, the great majority of the points will have the same Doppler value (like the torso of the person in the previous case). As such, the visual representation of the Doppler dispersion over time will result in almost single line, with barely no dispersion in the Doppler axis. Figure 4.11 shows the Doppler distribution for the case of the mannequin. However, notice that there is a small dispersion of Doppler of the mannequin, which is caused by small oscillatory movements of its pending arms combined with a certain component of rotation of the body during towing of the dummy.

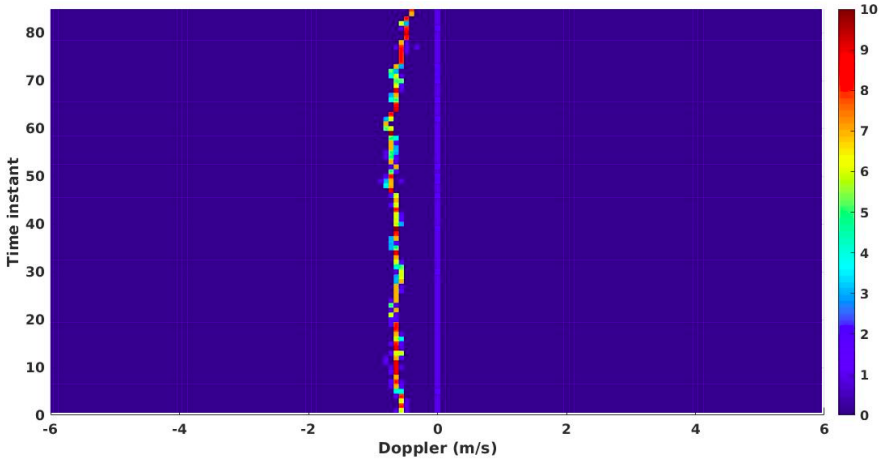


Figure 4.11: Histogramic representation of Doppler measures of a mannequin moving in front and towards the radar (path C). The colour bar on the right of the histogram represents the absolute number of points.

The pattern of motion of a person moving in a wheelchair is similar to the one of a mannequin. A person moving in an automated wheelchair doesn't show movement of arms and limbs, and so, he/she moves as a block. Figure 4.12 shows the typical motion of a person in an automated wheelchair.

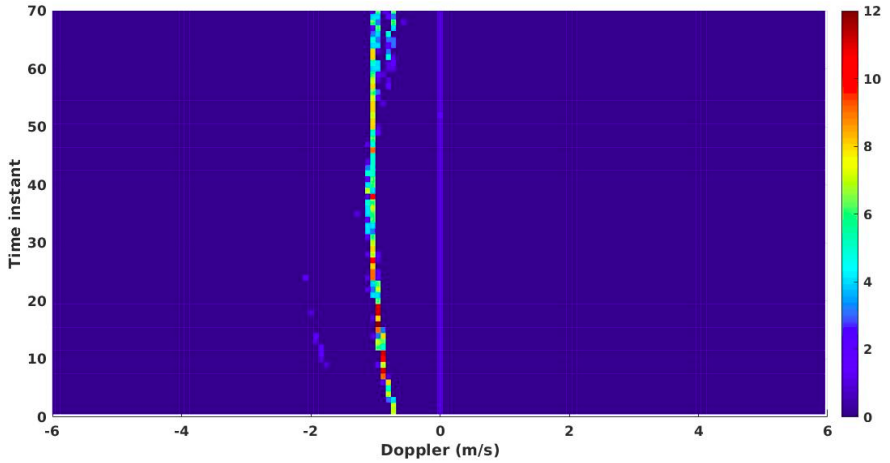


Figure 4.12: Histogramic representation of Doppler measures of a person in an automated wheelchair moving in front and towards the radar (path C). The colour bar on the right of the histogram represents the absolute number of points.

4.7 Calculation and usage of RCS as a feature

We have previously introduced the concept of RCS. In this work, as it was not possible to find a clear definition of the RCS in the TI documentation, the formula shown in equation (4.1) was derived from basic radio-frequency principles to compute the RCS. The expression derived is

$$\sigma = \frac{(4\pi)^3 d^4 P_r}{\lambda^2 G_t G_r P_t} \ , \tag{4.1}$$

where d denotes the distance to the target (the range mentioned in section 4.3), P_t and P_r represent the power emitted by the radar and the received power, respectively, λ is the wave length of the radar signal, and G_t and G_r are the transmission and reception gains, respectively[1].

The value for P_t is 12.5 dBm, which is later converted to Watts. The transmission and reception gains, G_t and G_r, have the same value of 9dB. Because the radar chirp used has a bandwidth of approximately 4 GHz (sweeping the frequencies between 77 GHz and 81 GHz), in order to calculate the RCS, the wave length used is based on the central frequency, 79 GHz, which corresponds to a wave length of 0.0038 meters. Since the received signal is attenuated by a series of filtration steps in the DSP pipeline of the radar, the power of the digitized signal after FFT processing needs to be compensated in order to obtain the total power reflected by the target. This compensation is implemented according to equation (4.2) as explained

[1]The values of RCS effectively used in this work were amplified (multiplied) by a factor of 2.4. This is an unwanted effect due to an error present in the Matlab code used to calculate this feature. The classification tests were repeated using the values of RCS without the amplification of this factor. The results achieved were similar to the ones attained with the multiplication factor (variations of less than 1% which are considered negligible).

below; see also [14].

$$P_r = P_{FFT} - [20log_{10}(2^{(nbits-1)}) + 20log_{10}(\sum_{i=0}^{N-1} \omega_i) - 20log_{10}(\sqrt{2})], \tag{4.2}$$

where P_{FFT} stands for the power of the signal after ADC and FFT processing (the relative intensity mentioned in section 4.3), *nbits* is the number of ADC bits used (12 in this case), the ω_i summation compensates the attenuation effect of windowing prior to the application of the FFT. Equation (4.2) is used as a correction factor [14]. The resulting value for power received is later converted from dBm to Watts.

The RCS for a target is usually defined as a single value. However, a target generates multiple points that appear in the point cloud, and the RCS is calculated for each one of them. To extract a single value for RCS, the average of the target's RCS is derived, by calculating the mean of the RCS of all points in the point cloud of the target. The RCS is calculated in every frame, and thus can be different from frame to frame.

Figure 4.13 shows how the values of RCS evolve as a function of distance, for the three types of targets: people, people in wheelchair, mannequin. Two immediate conclusions can be deducted from the graph. The first one is that the RCS increases with distance, due possibly to a overcompensation by a nonlinear gain of the signal spread loss associated to the distance, and a linear amplification introduced by the artificial factor of 2.4, in equation (4.1). The second is that the RCS from the mannequin or the wheelchair is systematically and significantly different from the one of a person, but the RCS of the mannequin and the wheelchair does not differ significantly. Thus, the RCS can be a relevant feature to distinguish a person from a mannequin, but not a mannequin from a person in a wheelchair. For other trajectories, this pattern holds.

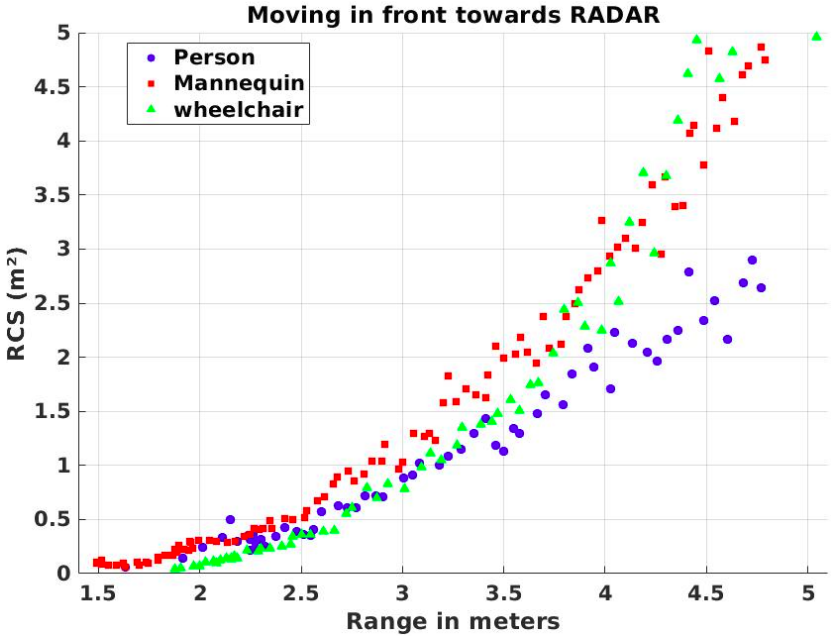


Figure 4.13: Representation of the RCS values as a function of distance to the radar. The path being traveled is in front and towards the radar (path C). Even at a short range there is a clear difference between some types of targets, such as the wheelchair and the mannequin.

4.8 Learning models applied

The learning models used in this work are implemented in the well-known Python ML toolkit named **scikit-learn**. Each model is dependent on a set of parameters. The values chosen for each of the parameters of each model will be presented in the sections detailing the performance of each one of the models in the generated dataset. The parameters for which the respective value is omitted are considered to have the default value defined in the aforementioned library.

In the subsequent section, all the models go through a phase of training and cross validation. The models then go under a testing stage, to derive their generalization error. In this work, 75% of the dataset (equivalent to 18341 frames) were used for training and the cross validation stage, and the remaining were applied in the testing stage. The dataset was previously shuffled, and the division in cross validation and test set was done in a stratified manner. When the datasets are stratified, it means that the percentage of each class in both the training and test set is equal to the percentage of each class in the original dataset. In this specific context, stratified sampling is not strictly necessary, because the dataset is fairly balanced; 53.9% of the data samples are from non-people moving, and the remaining 46.1% is related to people moving. This means that only by performing random sampling we would probably achieve in both the cross validation and test set the same distribution of samples as in the original dataset. However, if the dataset is unbalanced, then the distribution of classes in each of the sub datasets might be very different from the original one. Although this last case is not the one we are dealing with, stratified sampling can be still applied just to make sure the distribution of classes remains equal as the original one.

4.8.1 Artificial Neural Network

Artificial Neural Networks (ANNs) are a class of learning models applied in a wide range of scenarios, that has shown impressive results. ANNs are defined by being a structure constituted of several layers, each embodied by a set of processing units called the **neurons** or **nodes**. Each node in one layer connects with all the other nodes in the next layer, as depicted in figure 4.14. ANNs were inspired in biological neural networks, thus the strong resemblance with them. The main advantages in the use of ANNs are its capability to deal with non linear problems and the high processing parallelism that is evident in each layer by the existence of more than one node.

An ANN works by receiving, in each node, the outputs from the previous nodes; these outputs are combined in a function called **activation function**, and are passed on to the next node (in the next layer). Each input is associated with a weigth, meaning that the connection between a node and one in the next layer has a respective weight. The training of the neural networks consists in adjusting the connection weights in proportion to the difference between the produced output and the correct one - the **error** [26]. The error is thus a function of the weights, that yields an hypersurface with an irregular shape, with many peaks, saddle points and minima. With the aid of an adequate learning algorithm, the global minima, or a local minima of this function can be found [26].

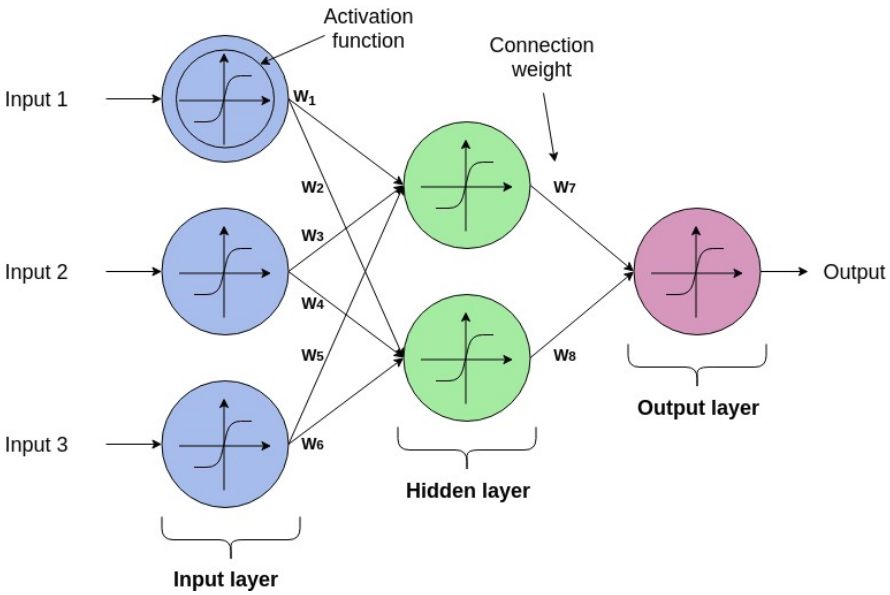


Figure 4.14: Structure of an ANN, with 1 hidden layer of 2 nodes, plus an input layer with 3 nodes and an output layer with one node.

4.8.2 Random Forest

In order to understand the concept of Random Forest (RF), it is convenient to have an adequate grasp of what is a **decision tree**. A decision tree is a decision model used in ML [27]. In the context of decision trees, a node represents a test on an attribute, and each branch represents a specific outcome of that test. Each set of nodes that have the same number of ancestor nodes constitute a **level** of the tree; each node without descendants is called a **leaf node**. At each level, one or more features of the input data are used to perform the partition. Each leaf node represents a possible output. A generic representation of a decision tree can be found in figure 4.15.

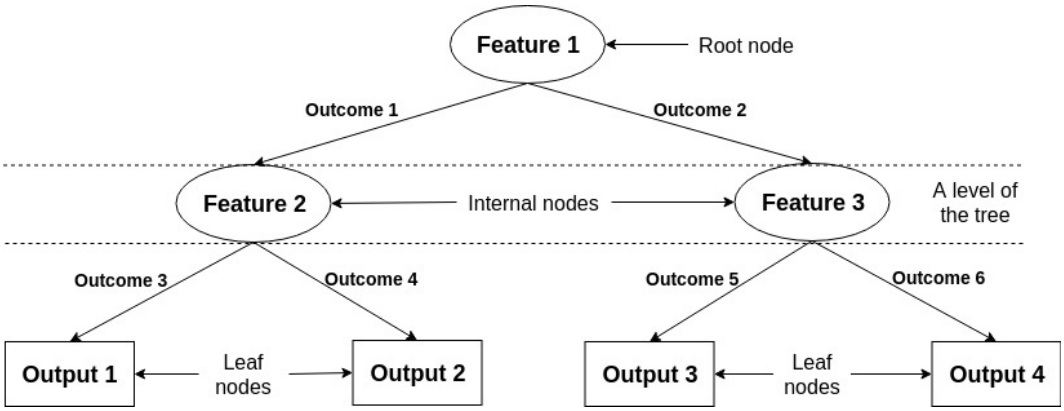


Figure 4.15: Schematic representation of a decision tree.

Decision trees have the advantage of being easily interpreted by humans, handle both numerical and categorical data and to mimic the way humans think and reason. Nonetheless,

decision trees suffer from high variance; the decision trees output depends on the feature that is used at each level. A method with this characteristic is not robust. To reduce variance, one possible approach is to train different trees on different training sets and then to take into account all the outputs by averaging them all (numerical data) or by choosing the output with the majority of the votes (categorical data). This increases the statistical accuracy of the decision trees. The problem with this approach is that often there is no access to several training sets. To solve this, we can use bootstrap, which consists in random sampling with replacement. The method of using bootstrap to create several training tests and then train a plurality of decision trees is called **bagging**. Although bagging decreases variance, it has the disadvantage of creating similar trees. Suppose there is a very influential (or strong) feature. Most of the trees will choose this feature for the top split. Thus, as said, the trees generated will be similar. Hence, to tackle this issue, RF are used. In the RF model, each tree only considers a subset of the features when deciding which feature to use in a split. Considering p as the total number of features, the typical number of features considered at each split when applying the RF model is approximately equal to $\sqrt{p}$. This process serves the purpose of decorrelating the trees, reducing variability and hence make the model more reliable.

4.8.3 Model's hyperparameters and parameters

When training different models, one concern is the **hyperparameter** tuning. Hyperparameters are the parameters that need to be defined before training the model, or in other words, parameters that are not fitted by the model whilst training. On the other hand, **parameters** is the name used to define the parameters that are learned from the training data. In the case of a neural network, the parameters are the weights of the network, that will be fitted during training in order for the model to achieve the best possible performance. The hyperparameters can be the number of hidden layers, the number of hidden nodes in each hidden layer and the activation function in each node.

As mentioned, the hyperparameters are not learned from the data available. Furthermore, there is no defined formula for the definition of the hyperparameters. The definition of the hyperparameters is empirical, meaning that only through experience it is possible to reach an adequate value for it. There are different searching techniques to explore the best hyperparameters. At this stage of the work, we will perform manual search, which means to manually test a set of different hyperparameters and verify what are the ones that work best for the available data.

4.9 Validation and selection of best model

4.9.1 Cross Validation

After training (possibly) several models, it is necessary to select the best model. One way to achieve this is by training the models and then derive its misclassification rate on the same training stage. Misclassification rate is defined as the number of wrong outputs divided by the total number of outputs. However, as the model uses this data to train, when testing, this data is not new to the model, and so, it will have a higher probability of being correctly classified. What is really needed is to determine the **generalization error**, that is, the misclassification rate on data the models have not processed before [17]. In order to compute this generalization error, the model must be tested in a dataset that has not been

used in the training stage. To achieve this, we can partition the dataset used in **training** and **validation set**. The validation set is used to select the best model. The original dataset can be further partitioned in a **test set**. After the best model is selected, we can train it using both the training and **cross validation set**, and test it on the test set, and extract its final estimation accuracy.

K-Fold Cross Validation

The aforementioned approach is well-suited when we possess a large dataset. Otherwise, when partitioning the dataset into training, validation and test set, we might not have enough data samples in the training set to train the models, and won't also have enough data in the validation and testing set to determine its future performance. A possible approach to solve this is **K-Fold cross validation**. The intuition of K-Fold cross validation consists in dividing the dataset into K folds. For each fold $k \in \{1, ..., K\}$, we train the model in all the folds but the kth one. The kth fold is used to test the model, in a round-robin fashion [17]. Figure 4.16 shows a schematic representation of this rationale; the dataset is divided in four folds, and in each iteration we test the model in each different fold, for a total of 4 iterations.

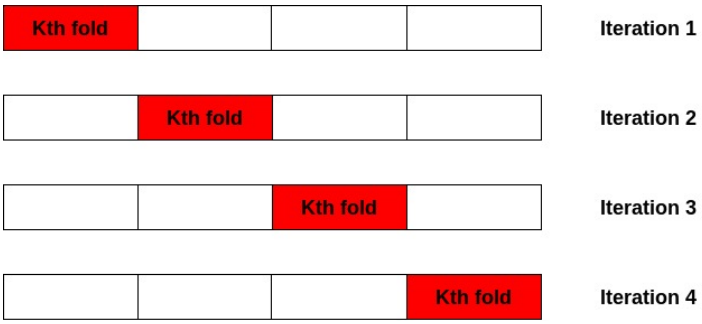


Figure 4.16: K-fold cross validation schematic.

Then the error is stored when the model is tested in each fold, and it is averaged. This will be the test error.

In the present work, the part of the dataset that is used for K-Fold cross validation (75% of the original dataset) will be divided in 5 folds, which is a typical number of folds used.

4.9.2 Artificial Neural Network

The first model applied was an artificial neural network, using as features the histogram data and the RCS. The chosen values for the hyperparameters were the following:

- **Activation function**: ReLU, which stands for **Re**ctified **L**inear **U**nits, is a function defined as $f(x) = max(0, x)$
- **Number of hidden layers and number of neurons**: 4 layers, with (from the first layer to the last) 100, 70, 50 and 20 neurons
- **Learning rate**: 0.001

- **Maximum iterations**: 10000
- **Random state**: 42

In this work we define the performance of the models as being the **accuracy**. Following the standard nomenclature of ML, accuracy is defined as the quotient between the total number of correctly classified samples and the total number of samples, $\frac{TP+TN}{N}$, where TP and TN stands for true positives and true negatives (respectively), and N represents the total number of examples. The average performance attained was 84.9%. The average error is the complement of this value. When training and testing the model, if we remove the RCS from the feature space and re-trained the model, the average performance dropped to 82.5%.

4.9.3 Random Forest

The second learning model applied was a RF, with the following hyperparameters:

- **Number of estimators**: 500
- **Maximum depth**: 600
- **Random state**: 0

The average accuracy attained was 86.2%, which is better than the one achieved by the neural network. As in the case for the neural network, when the RCS is not used the accuracy decreases, in this case for a value of 83%.

4.9.4 Selection of best model

Given the results of the K-Fold cross validation, we can conclude two things: the first one is that the RF model is superior to the ANN both in the case the RCS is used as a feature and in the case the RCS is not used; the second conclusion is that the RCS has a visible impact on the performance of both models, and its absence decreases the accuracy of both models.

Based on the above described results, the model chosen is the RF. This model is re-trained using the train and validation set, and tested on the remaining 25% of the dataset. The final performance achieved is 86% using the RCS, and 82.9% without RCS. Table 4.3 shows the confusion matrices of the RF. Table 4.4 shows the confusion matrices of the ANN when trained with 75% of the dataset and tested in the remaining 25%. The performance attained with the ANN was 85.9% using the RCS and 81.5% not using the RCS. Notice that, although only the accuracy is referenced, all the other metrics (such as precision, recall, etc) can be derived from the confusion matrices.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	2916	387
Person	470	2341
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	2821	482
Person	563	2248

Table 4.3: RF confusion matrices.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	2890	413
Person	448	2363
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	2732	571
Person	559	2252

Table 4.4: Artificial neural network confusion matrices.

4.10 Discussion of results

From the results achieved in this chapter, several conclusions can be deduced. The DBSCAN algorithm was considered an adequate method to apply in this type of data (point cloud), and allows to correctly segment the point cloud and isolate the target of interest. Although there are methods to theoretically define the *Eps*, it was shown that it is possible to achieve an adequate clustering efficacy with values chosen empirically.

The motion of people can be visually detected by looking at the evolution of the position of the points of the point cloud over time. By representing the distribution of velocity values as a bi-dimensional histogram, and by concatenating the histograms derived from each frame, we achieve a three-dimensional view of the velocity distribution pattern of each target while traversing some trajectory. Each bi-dimensional histogram, along with the calculated RCS from the target of interest, can be used to feed an ANN or a RF, which can classify the target moving with a high level of accuracy (around 86%). Moreover, as seen, the RCS contributes to the improvement of the performance of the classification models.

Chapter 5

Feature extraction refinement and model optimization

5.1 Using more than one frame in each data sample

While each data sample consists of only one frame, the learning model is processing patterns from instantaneous representations (snapshots) of Doppler dispersion rather than from a sequence of Doppler metrics representative of the motion pattern. The wide instantaneous dispersion over the Doppler axis is indeed a characteristic of the motion of a person, but the cadence of this same dispersion is also characteristic of the movement. So far, this implementation does not take this aspect into consideration. In order to enable the algorithm to try to capture the pattern of motion, it is necessary to provide it with more than one frame at each iteration. Hence, each data sample needs to have more than one frame. To achieve this, successive frames are concatenated, generating a new data sample where the number of elements is given by the the expression: $151 \times N_{frames}$, where N_{frames} represents the number of frames that compose a single data sample. Figure 5.1 illustrates this process for two frames per data sample.

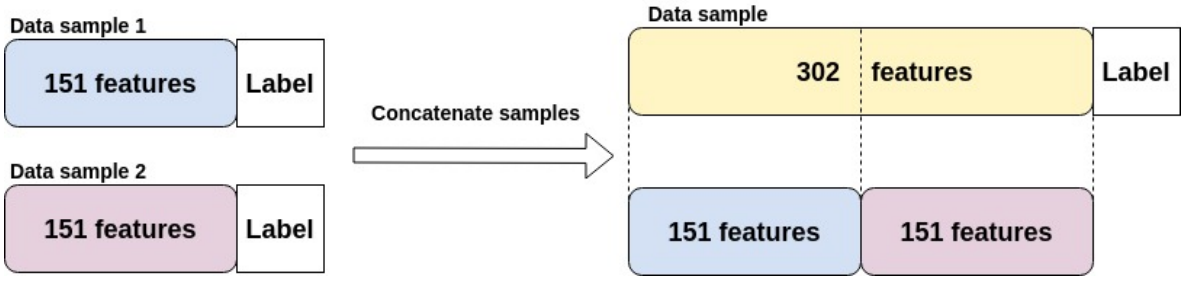


Figure 5.1: Concatenation of two data samples, to generate a data sample with two frames.

We will opt to apply the two classification models used in the previous chapter, and check how the number of frames impacts their performance. For such, the ANN and RF will be trained with the new training set (75% of the dataset), and tested on the test set (25% of the dataset). We start by creating a dataset with two frames per data sample and analyze its impact on the performance. The new dataset is created using a "sliding window"

approach. While retrieving the frames of the histogram generated by a test, the first frame is concatenated with the second one, the second frame is concatenated with the third, the third with the fourth, and so on. This way, we minimize the reduction of the dataset. The new dataset has a total of 24125 data samples, each constituted by a total of 302 features.

5.1.1 Advantages of using more than one frame

Using the same neural network architecture as described in the previous chapter, the performance was 85.9% using only one frame, and increased to 86.8% with two frames. Regarding the RF, using the same parameters as the ones before, the performance increased to 89.1%. This is indicative that more than one frame does increase the ability of the models to classify correctly the target that originates the radar returns. It is useful to analyze how much performance increases for a larger number of frames. Figure 5.2 shows how the number of frames impacts both the RF and ANN, from 1 frame up to 10 frames. Table 5.1 shows the number of data samples for each type of dataset and the number of features in each data sample.

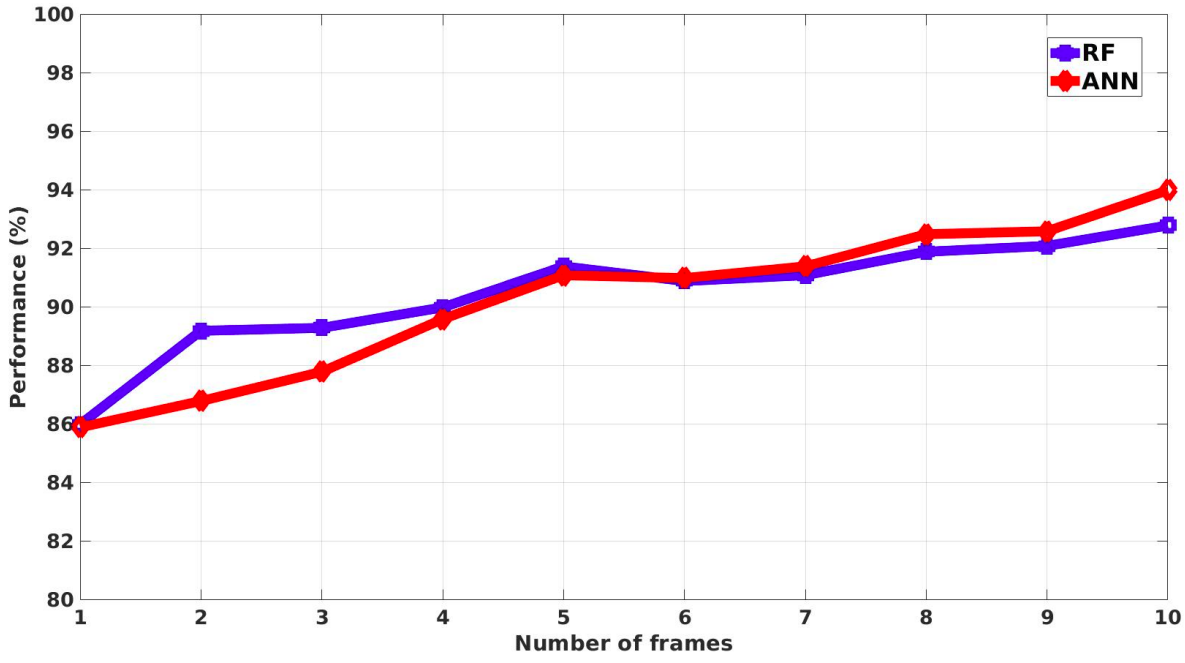


Figure 5.2: Influence of the number of frames used in the performance of the models.

Number of frames	Number of data samples	Number of features
1 frame	24455	151
2 frames	24125	302
3 frames	23920	453
4 frames	23448	604
5 frames	23102	755
6 frames	22772	906
7 frames	22426	1057
8 frames	22096	1208
9 frames	21751	1359
10 frames	21421	1510

Table 5.1: Size of dataset in data samples and features, as a function of the number of frames per data sample.

As can be seen in the plot, the RF has a better performance than the ANN until the 6th frame. From this point on, the ANN performs better than the RF.

Note that the performance comparison with different number of frames may not be fair because it is done using always the same ANN and RF configuration, which were found to be (through experiments) the best configurations possible while using one frame per data sample. For other types of datasets, there may be better configurations that will yield an even better performance. Another important aspect to note is that the size of the current dataset decreases as we increase the number of frames per data sample, which means that there are fewer data samples to train and test the models. However, even considering these aspects, the improvement of performance with the increase of frames is evident.

5.1.2 Disadvantages of using more than one frame

The usage of more frames per data sample for classification yields disadvantages regarding the time needed to extract these features. Each frame takes 0.05 seconds to be acquired, thus, using more frames implies that the system will take a longer time performing measurements; after the measures have been acquired, the feature extraction algorithms will have to perform more computations (when compared to the case where only one frame per data sample is used) because there are more frames from where to extract features. Thus, it will take a longer time to acquire the necessary features to use in the classification stage. Therefore, a trade-off needs to be made between performance of the system and feature extraction latency. The effects of the use of more frames per data sample on the classification execution speed will be evaluated in section 5.3.

5.2 Feature dimension reduction and Haralick features

The number of first order features used is considerably large; even when using only one frame per data sample, we have to consider 151 features. Furthermore, the vast majority of the bins in the dataset have a Doppler value of zero, which might indicate some redundancy or excess of features in the dataset, and thus, drawing an approach which is able to reduce the number of features is desirable.

The reduction of features aims at extracting statistical information from the frames, and compress it into a smaller subset of features. To achieve this we extract a set of features from the generated histograms, called the **Haralick features**. In this context, they are considered **second order features** because they are extracted from first order features previously defined. Each set of Haralick features is extracted from a data sample with only one frame. Haralick features are a set of high order statistics often applied in the detection of textures in images, by exploring the distribution of gray values in the pixels of the images and the spatial relation between them. These features are computed from a gray-level co-occurrence matrix (GLCM), which is a method of quantifying the spatial relation of neighboring pixels in an image. The Haralick features computed from the GLCM have been successfully applied in a myriad of applications, from skin texture analysis, to plant leaf classification and analysis of MRI and ultrasound images of organs [28]. In the present work, instead of exploiting gray levels, we apply the same concepts to process the relative frequency of Doppler values represented by the histograms of Doppler radar data.

The GLCM is an m by m matrix, where m is the number of different gray intensity values. Each value in the matrix is calculated by counting how often a pixel with intensity (gray value) value i occurs in a defined spatial relationship, P, to a pixel with value j. Often, the spatial relationship considered is the immediate right, or by other words, the pixel that stands immediate to the right of the other pixel. Nonetheless, other relationships can be defined, but in this work, the immediate right will be the one considered. To summarize, each element (i, j) of the GLCM is the number of times a pixel with value j appeared in the immediate right of a pixel with value i. Figure 5.3 shows an example of how one can derive the GLCM from an image.

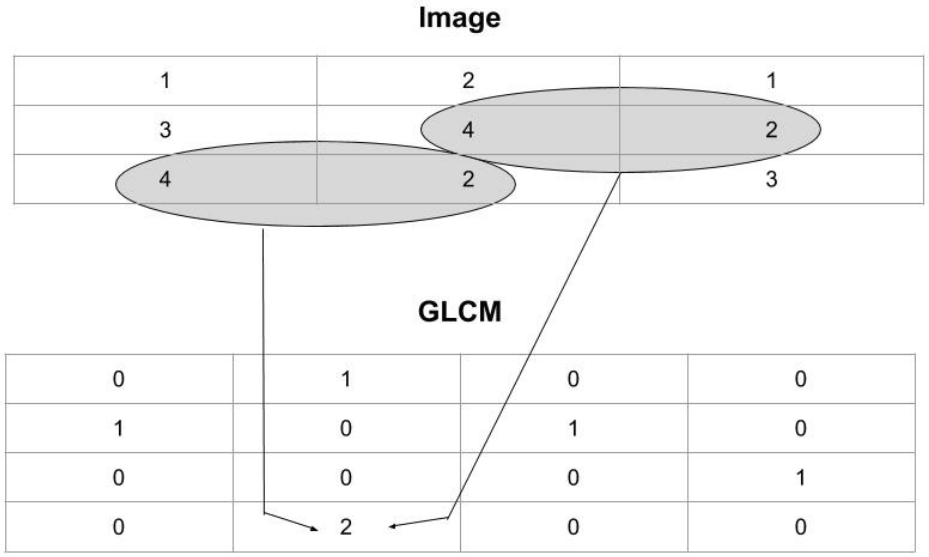


Figure 5.3: Visual example of the derivation of a GLCM from an image.

At the top we have an image, represented as a matrix where in each cell (pixel) is registered the gray intensity. Because the maximum intensity is 4, the GLCM (represented in the lower part of the same figure) will be a 4 by 4 matrix. To avoid looking at all the pixels in the image, pay special attention to the marked numbers on the image. The pixel with intensity 2 is horizontally adjacent to the pixel 4 two times. Thus, in position $(4, 2)$ in the GLCM, the

value will be 2. The same rationale applies to all the other positions in the matrix.

After obtaining the GLCM, the next step is to calculate the Haralick features. The literature defines a total of 14 features [29]. In our work, the features extracted from each frame are the standard deviation (not an Haralick feature), entropy, contrast, energy, and homogeneity. These Haralick features were chosen due to their relatively simple interpretation and because of the fact of being the most used features. Bellow follows a description of each one of these features, as well as the equations used to calculate each of the features. In the following equations (5.1) to (5.4), i represents the row index and j represents the column index of the GLCM. Consider N as the sum of the elements of the GLCM. If the GLCM is divided by N, then $p(i,j)$ represents the joint probability that a pair of points in the original image (used to derive the GLCM) obey to the relation P for the pair of pixels with value represented by the indexes of i and j in GLCM. K is used to define the maximum gray intensity (or the highest row/column number of the GLCM).

- **Entropy**: Is a measure of the "disorder" (or "randomness") of the image. Entropy has value 0 when the probability that any pair of pixels obeys to relation P is 0 (all $p_{i,j}$ values are 0), and has value $2log_2K$ when all pairs of pixels have the same probability of obeying to relation P ($\forall_{i,j}$, $p_{i,j}$ has the same value).

$$Entropy = -\sum_{i}^{K}\sum_{j}^{K}[p(i,j)log_2(p(i,j))] \tag{5.1}$$

- **Contrast**: Is a measure of the local variations present in the image, or by other words, the variation between the reference pixel and the neighbor pixels. A large contrast is a result of large intensity differences. Contrast is equal to 0 when the image is constant, and the maximum value is $(K-1)^2$.

$$Contrast = \sum_{i}^{K}\sum_{j}^{K}|i-j|^2p(i,j) \tag{5.2}$$

- **Energy**: This feature measures the repeatability of pairs of pixels. A large value for energy occurs when the image has a periodic or constant form. The range of values for energy is [0,1].

$$Energy = \sum_{i}^{K}\sum_{j}^{K}p(i,j)^2 \tag{5.3}$$

- **Homogeneity**: Is a measure of the similarity between the distribution of the elements in the image and the diagonal of the GLCM. This feature has values that are contained in the interval $[0,1]$. The value of 1 is attained when the GLCM is a diagonal matrix.

$$Homogeneity = \sum_{i}^{K}\sum_{j}^{K}\frac{p(i,j)}{1+|i-j|} \tag{5.4}$$

In order to extract the aforementioned features, the histogram of each frame is converted in a gray-level-co-occurence (GLCM) matrix, using matlab function *graycomatrix*. Then, using function *graycoprops* and passing the generated matrix as argument, we can extract the

contrast, the energy and the homogeneity. Standard deviation and entropy can be calculated using the Matlab function *std* and *entropy*, respectively. Using these methods, each frame of 150 values is converted into a frame of only 5 features.

5.2.1 Classification based on Haralick features

The new generated dataset, composed by the retrieved Haralick features, has a total of 24455 data samples, like the dataset used in the previous chapter. Each data sample has 6 features (the 4 Haralick features aforementioned, plus the standar deviation and the RCS). Using the same RF and ANN as before, and applying the same k-fold cross validation methodology as applied in section section 4.9, the average performance attained for the RF was 62.3% and for the ANN was 67.4%. After testing the models on the testing set (25% of the dataset), the final values attained were 62.9% for the RF and 66.8% for the ANN. However, when ignoring the RCS, the final performance of the ANN was 64.1%, which is worse than when using the RCS, but the performance of the RF increased to 66%. Table 5.2 and table 5.3 show the confusion matrices of the ANN and RF, respectively.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	2665	638
Person	1390	1421
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	2630	673
Person	1525	1286

Table 5.2: ANN confusion matrices while using Haralick features.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	2101	1202
Person	1069	1742
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	2501	802
Person	1276	1535

Table 5.3: RF confusion matrices while using Haralick features.

We conclude from these results that the chosen Haralick features combined with the standard deviation, although allowing for a more compact feature representation, do not enable a good model performance, especially when compared to the original dataset. The low performance of the classification models while using the Haralick features can be explained by the fact that these features are normally retrieved from bi-dimensional data, while in this case we have exploited only uni-dimensional data (one frame). As seen in subsection 5.1.1, the use of more than one frame per data sample could possibly lead to a better performance of these models when using the Haralick features. Another aspect to note is that, when the RCS is used as a feature, the ANN can perform better than the RF on this dataset, but while using the original dataset (with the velocity bins) this was not the case.

It is also important to bear in mind that the architecture of the models was not optimized for this dataset. Other configurations may yield better results than the ones presented.

5.3 Performance assessment of different methods of feature extraction and classification

Until this point, this dissertation covered three main methods of feature extraction:

- Retrieving the histogram values of velocity and the RCS of each frame, and using the data from each frame as a single data sample;
- Apply the same method as above, but use more than one frame per data sample;
- Apply a feature reduction method, namely the extraction of Harralick features from the velocity bins, and use them in conjunction with the RCS as a single data sample.

We have compared the methods in terms of classification performance, but have yet to assess and compare the computational effort of the three methods. For such, we adopt as a metric of computational effort the execution time of the overall classification process.

In order to compute the overall execution time, we need to measure the execution time in two different stages; firstly we have to measure the execution time of the code that is responsible to perform feature extraction (implemented in Matlab); secondly, we have to measure the execution time of the classification models using such features (implemented in Python). The application code was executed in a **Fujitsu** laptop, with the following characteristics:

- **Laptop:** Fujitsu LIFEBOOK S Series
- **Model:** S752 Notebook
- **RAM:** 8 GB
- **Processor:** Intel® Core™ i5-3230M CPU @ 2.60GHz * 4
- **OS:** Ubuntu 18.04.1 LTS

5.3.1 Timing metrics

Notice that the feature extraction process is applied in each frame. Thus, from frame to frame, given that the number of points and amount of clusters is different, the execution speed might also be different. Therefore, to derive an effective estimate of the performance of the code, it is necessary to measure the execution time for different frames and for different types of tests (person walking, mannequin moving, etc). For the specific case of feature extraction, the function used to derive the execution time is the *timeit* function of Matlab. After creating a handle of the function whose execution time we want to measure, we simply pass this function as an argument to the *timeit* function. This function calls the specified function multiple times and returns the average of the measurements, in seconds, which is the **wall-clock time** for the execution of the specified code. The wall-clock time is defined also as the elapsed real-time between the start and the end of a program. Considering the wall-clock time is a better metric than considering the CPU time. The CPU time only accounts for the time the processor is handling the program, and not for the memory access time and eventual input/output interaction. If a computer is performing a lot of tasks at the same time, using the wall-clock time may yield biased/unrealistic results. However, if the computer used is

only executing the desired program, all the resources will be available to the program, and the processing time will take in account only the execution of that program, and naturally the execution of some other operative system (OS) related tasks (which we will consider negligible). The *timeit* function also considers first-time costs. First-time cost is the time taken to execute the code for the first time. In fact, executing a script for the first time may take longer than executing for a second (or third, etc) time, due to different reasons; after executing the code for the first time, the processor may be working at a higher speed, which makes it faster to execute on the second time; the impact of the cache is also considerable, in the first time the code is executed, a lot of data may not be present in the cache, so the access to the memory will take a longer time, but on the second time, a lot of data might be already in cache, so the memory access will be processed at a higher speed.

5.3.2 Timing procedure

To have a robust estimate of the execution time of the feature extraction process, we have to measure the execution time for the feature extraction process in several different frames from different tests, several times each one. We consider one test of each target traveling each one of the eight trajectories defined in section 4.2, and compute the execution time of the feature extraction process for every frame. Finally, we average all the retrieved execution times to obtain a final average execution time. Note that the *timeit* function executes the same script multiple times, and yields as the execution time of a script the average of the execution times attained.

In order to measure the execution time of the classification process, we also consider the wall-clock time, by using the built-in Python library *time*. As opposed to the feature extraction case, for the classification process it is not necessary to measure the execution time when classifying different frames. The frames when in the form of data samples, are conceptually equal to each other; they have the same length and the same type of data. Therefore, theoretically, all data samples should take the same time to classify. In practice, this might not be the case, only due to computer related issues, such as the number and type of tasks the computer OS is handling at the specific moment it is also classifying a data sample, but not because of the data sample *per se*. Due to these reasons, to have a statistically valid estimation of the execution time for the classification process, we have to classify a data sample several times, and it can be always the same data sample. We will classify a data sample 1000 times and average the execution times of each turn. This is our estimate of the classification execution time. We apply this process both for the ANN and for the RF, and consider each model execution time individually. Also, keep in mind that it is irrelevant if the data sample chosen for classification is correctly or wrongly classified, because, as described before, we are only interested in measuring the time necessary to output a classification result for a specific sample, not in the result of the classification.

5.3.3 Overall execution time

It is important to note that we are not actually considering the time necessary to train the model using the different types of data samples that can be derived from the different types of feature extraction. The model training is a procedure to be processed off-line and only once. Therefore, the time necessary for training is not relevant in this case.

The overall execution time will be the result of the summation of the feature extraction execution time plus the classification (by the ANN or the RF) execution time. Notice, however, that in a real time application, we would have to take in account also for the time necessary for the radar to retrieve a frame, which is 0.05 seconds, and not only the time for feature extraction and classification.

5.3.4 Extracting and classifying radar data from a single frame

The average feature extraction time was 38.1 milliseconds (ms), with the minimum time being 32.0 ms and the maximum being 67.4 ms. In the classification stage, the RF takes around 40.1 ms to classify a data sample, while the ANN takes around 0.5 ms. Thus, we obtain a final value of 78.2 ms for the feature extraction and classification using RF, and 38.6 ms for feature extraction and classification using the ANN. Table 5.4 summarizes the execution time for this type of feature extraction and classification.

Process	**Execution time (ms)**
Feature extraction (average)	38.1
Feature extraction (maximum)	67.4
Feature extraction (minimum)	32.0
RF classification	40.1
ANN classification	0.5
Overall processing using RF	78.2
Overall processing using ANN	38.6

Table 5.4: Execution times for extracting and classifying radar data with the bins and RCS from a single frame.

5.3.5 Extracting and classifying radar data from more than one frame

When using more than one frame per data sample, the expectation is that both the feature extraction process and the classification will take a longer time to compute, because more features will have to be extracted to create a single data sample, and the classification will have to take into account more features to output a result. In this subsection we examine how the increase in the number of frames per data sample impacts (and possibly increases) the execution time for feature extraction and classification. As done in section 5.1, we will consider an increasing number of frames, from 2 to 10. For each different number of frames per data sample, we will measure the execution time for feature extraction and classification using the same methods as used in the previous subsection.

Figure 5.4 shows how the execution time of the classification by the ANN and RF evolves as a function of the number of frames.

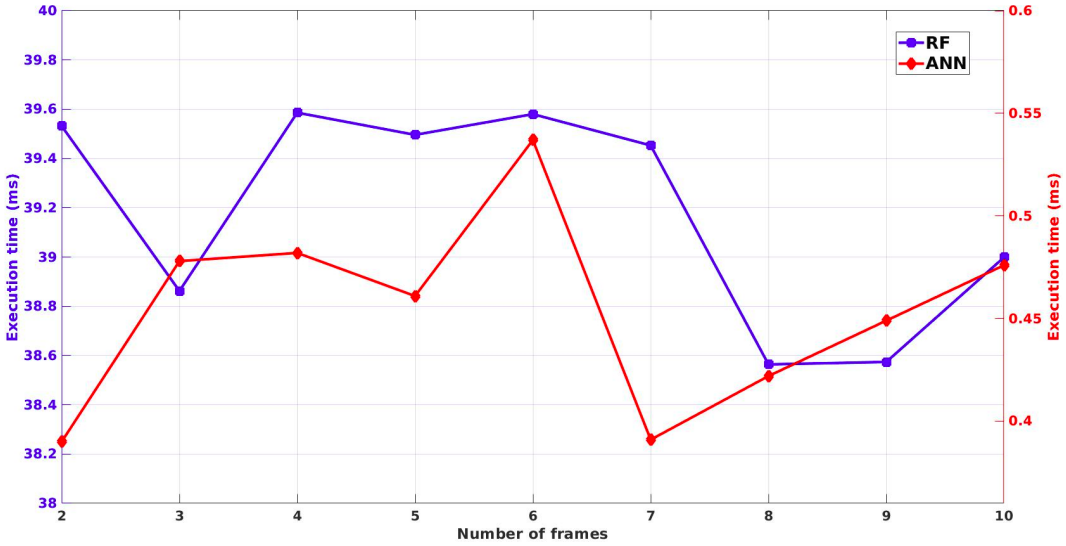


Figure 5.4: Execution times of both models classification, as a function of the number of frames. Notice the two different scales on the vertical axis.

Figure 5.5 shows the time for feature extraction, and ANN and RF overall classification time.

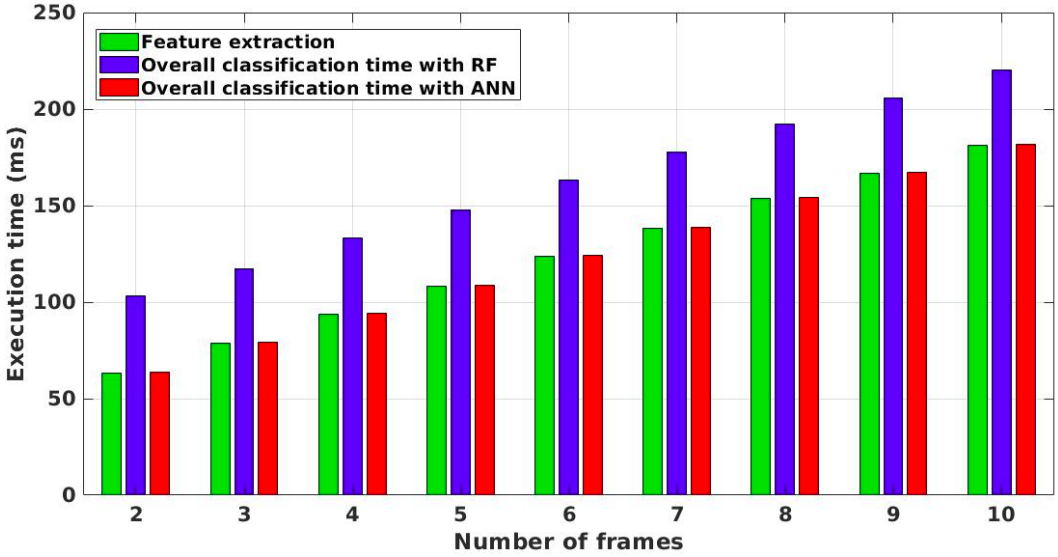


Figure 5.5: Execution times of feature extraction and overall classification time for the ANN and RF, as function of the number of frames.

From the plotted graphs we can firstly conclude that the models classification process seems to not be visibly affected by the increase in the number of frames. In fact, for the case of the ANN processing, the minimum time measured was 0.39 ms with 2 frames, and the maximum was 0.537 ms with 5 frames. The average time was 0.454 ms and the standard deviation was 0.0047 ms. With the maximum number of frames (10) the time for classification was 0.476 ms. The ANN is faster than the RF at performing classification. The second

conclusion is that the feature extraction time increases almost linearly with the number of frames, for both models. The maximum time necessary to extract frames was 181.4 ms, for 10 frames, and 63.5 ms for 2 frames. Table 5.5 summarizes the mentioned execution times in tabular form, for the case of the ANN.

Process	Number of frames	Execution time (ms)
Feature extraction (maximum)	10	181.4
Feature extraction (minimum)	2	63.5
ANN classification (minimum)	2	0.39
ANN classification (maximum)	5	0.537
ANN average classification	-	0.454
ANN classification	10	0.476

Table 5.5: Execution times for extracting and classifying radar data with the bins and RCS from more than one frame.

5.3.6 Extracting and classifying radar data based on Haralick features from a single frame

The third and last method whose timing execution was measured was the classification process based on Haralick features using only one frame plus the standard deviation and the RCS. Table 5.6 shows the execution times for feature extraction and classification using the ANN and the RF.

Process	Execution time (ms)
Feature extraction (average)	48.9
RF classification	40.2
ANN classification	0.2

Table 5.6: Execution times for extracting and classifying radar data based on Haralick features from a single frame.

5.3.7 Comparison of results

The extraction of features from only one frame for each data sample was, as expected, faster than using more than one frame. The difference between the fastest (1 frame) and the most time consuming version (10 frames) is 143.3 ms. Since the total time including the frame acquisition and classification in the worst case considered (RF plus 10 features) remains bellow 800 ms, the total classification time is compatible with its use in real-time applications. In the case of the classification process, the execution time did not increase with the number of features. The use of Haralick features requires more time to process than using directly the 150 bins of histograms. The extraction of Haralick features was still faster than creating a data sample composed by more than one frame. In the classification stage, the data samples composed by Haralick features were the fastest ones to classify while using the ANN. For the case of the RF, the classification of Haralick features took approximately the same time as classifying the velocity bins and the RCS of one frame per data sample.

5.4 Hyperparameter optimization

In subsection 4.8.3 we briefly mentioned the existence of different techniques to search for the optimal model's hyperparameters. Manual search is of course the most basic searching technique. However there are other more effective techniques, such as those described bellow: **grid search** and **random search**.

Note that the manual search used to find the optimal parameters for the ANN and the RF was done considering the dataset with one frame per data sample and using the velocity bins and the RCS. The best model found using manual search was then applied to the other types of datasets. As mentioned before, different hyperparameters may achieve better results in different datasets. In this section we will exploit only the hyperparameter optimization for the dataset using the velocity bins of one single frame per data sample plus the RCS value.

5.4.1 Grid search

Grid search is a simple searching technique were, for each hyperparameter that we want to tune, we define a list of values; the set of combinations of values for all hyperparameters is called the grid. Grid search builds a model for every combination of hyperparameters specified in the grid. The performance of each model is recorded, and in the end we can retrieve the hyperparameters that yielded the best performance according to some predefined metric.

Neural network optimization

For the case of the neural network, the parameters that were tuned were the **activation function**, the **solver for weight optimization**, the **learning rate**, the **number of hidden layers** and the **number of neurons** in each layer. Table 5.7 shows the set of values used for each of the hyperparameters. Notice that not all combinations of number of hidden layers and hidden nodes per layer (network architecture) were tested.

Parameter	Values
Activation function	ReLU, Tanh, Logistic
Solver	SGD, Adam
Learning rate	0.0001, 0.001, 0.01, 0.05, 0.1
Number of hidden layers	1, 2, 3, 4, 5, 6, 7
Number of hidden nodes per layer	1, 2, 3, 4, 5, 6, 7, 10, 20, 50, 70, 100, 200, 500

Table 5.7: Range of values used in grid search for different ANN hyperparameters.

The total number of combinations (or networks) is 360. The best performing network attained an accuracy of 86.3%, and had the following hyperparameters:

- **Activation function:** ReLU
- **Solver:** Adam
- **Learning rate:** 0.01
- **Number of hidden layers:** 3
- **Number of hidden nodes per layer (from 1st to last layer):** 100, 70, 10

Random forest optimization

In the case of the RF, different models are generated with different combinations of hyperparameters. Table 5.8 shows the hyperparameters that were modified and the respective set of values. The number of different models generated was 225.

Parameter	Values
Number of estimators (trees)	1, 5, 10, 20, 50, 100, 200, 250, 500, 650, 800, 1000, 1200, 1500, 2000
Maximum depth	1, 5, 10, 20, 50, 100, 200, 250, 500, 650, 800, 1000, 1200, 1500, 2000

Table 5.8: Range of values used in grid search for different RF hyperparameters.

The best performance attained was 86.4% accuracy, achieved with the following hyperparameters:

- **Number of estimators:** 1200
- **Maximum depth:** 150

5.4.2 Random search

Grid search is widely used because it is simple to implement, it is a process easy to parallelize, and its reliable in low dimensional spaces (one dimension or two dimensions), meaning that a good solution is easy to find. However, grid search has the disadvantage of being computationally expensive, because it tests a lot of different parameter combinations. Also, as mentioned in [30], some hyperparameters do not matter much for the performance of the models, but because grid search tests all combinations, it "wastes" time on not so important hyperparameters. Random search can be more efficient than performing trials on a grid, especially in high dimensional spaces. Empirical evidence shows that random search can find models that are as good or even better than the ones found with grid search, while only using a fraction of the time.

Random search works by extracting independent draws from a parameter uniform distribution. Because it is based on random sampling, the technique may cope better with non important hyperparameters. Consider each hyperparameter value combination as a point in the parameter space. Consider also a particular case were we only have two parameters. In figure 5.6, the image at the left shows how the points are distributed in the parameter space using the grid technique. In contrast, at the right we see how the points are distributed in the random layout. As we can see, in the grid layout, the points are more evenly distributed. However, if we project the points in any of those dimensions, most of the points will be projected onto the same location and thus cover a smaller space of each parameter spectrum. On the other hand, in the random layout, the projection of the points in each parameter dimension covers a lot more space of the specific parameter range, which in turn, makes it account for more variability in combinations of different parameter values and to have more sensibility to different parameter impact.

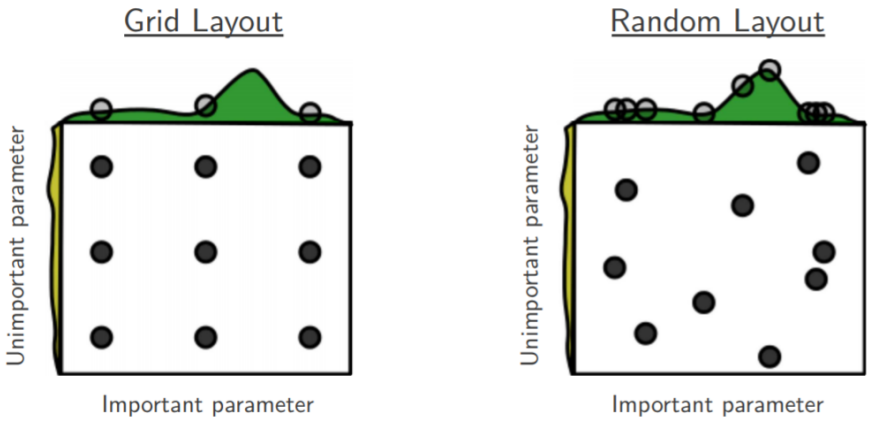


Figure 5.6: Grid and random search over 9 trials. Grid layout (left) and random layout (right). Source [30].

Neural network optimization

To perform random search, we simply provide the same list of parameter values as provided for the grid search (see table 5.7), and we define how many different parameter settings we want to be sampled. Originally we had 360 different combinations. In random search we are going to retrieve 100 random samples and calculate their individual performance. The learning rate is randomly generated, with values in the interval]0, 1]. The best performing combination was the one listed bellow, with an accuracy of 86.02%.

- **Activation function:** ReLU
- **Solver:** SGD
- **Learning rate:** 0.23
- **Number of hidden layers:** 6
- **Number of hidden nodes per layer (from 1st to last layer):** 500, 500, 200, 100, 50, 5

Random forest optimization

To apply random search for the hyperparameter optimization in the RF model, we generate a random (integer) number for the maximum depth of the trees and the number of estimators. The ranges of each hyperparameter are the same as those presented in table 5.8. Note that, because the value of each parameter is defined randomly, the random search technique is exploiting hyperparameter values that were not considered in the grid search case, simply because these same values were not defined in the parameters list of values. We perform a total of 112 searches. In practice, this means we consider 112 different hyperparameters combinations, which is about half of the number of combinations of hyperparameters for the RF model when compared with the grid search case. The performance attained was 86.2%, achieved with the following hyperparameters:

- **Number of estimators:** 1017
- **Maximum depth:** 337

5.4.3 Discussion and conclusions on hyperparameters optimization

The results achieved in this section show that the hyperparameters retrieved by grid and random search led to a model performance that is, for both models, only slightly better (less than 0.5% better) than the models that use the hyperparameters found through manual search. Grid and random search were executed in a remote HP Proliant device, with two (2x) Intel Xeon 2.5GHz processor and 32GiB of RAM for a total availability of 12 cores and 24 execution threads. Grid and random search required 4 and 1 day (respectively) to perform all the necessary computations to retrieve the best ANN hyperparameters, whereas for the case of the RF, both techniques required several hours to calculate the optimal hyperparameters. Using manual search, in a few trials it was possible to retrieve the hyperparameters presented in subsection 4.9.2 and subsection 4.9.3. This leads to the conclusion that by performing a small number of tests, a good set of hyperparameters can be found, and thus, grid and random search are disadvantageous techniques to use in this case because they require a long time to perform all the computations. However, grid and random search have the advantage of searching for the optimal hyperparameters autonomously. In conclusion, manual search may be considered an adequate searching technique to use in this case.

Chapter 6

People detection by a mobile robot

The previous chapter showed the efficacy of classification models when applied to distinguish people moving from other moving targets, based on data acquired by a radar device positioned on a tripod. For static scenarios, where the objective is to identify targets that appear in a well defined area, that method is sufficient. However, a more dynamic scenario envisages a small robot that carries the device, and is able to position itself and to move in an indoor space and eventually acquire and classify data while moving. To achieve and test such system, we start by installing the aforementioned radar in a TurtleBot, and test the performance of the models with the newly acquired data, with the radar-equipped robot immobilized in an initial set of trials. TurtleBot is a small low cost robot, suited for navigation in indoor environments. More details about this robot can be found in `https://www.turtlebot.com/turtlebot2/`. Figure 6.1 shows the robot used, with the radar device installed. The radar is placed with an inclination of 10 degrees facing upwards. This makes sense if we recall that when the radar was positioned in the tripod, the device was facing downwards with a slope of 10 degrees. As the radar pose is now symmetric to the previous one, we consider also a symmetric inclination.

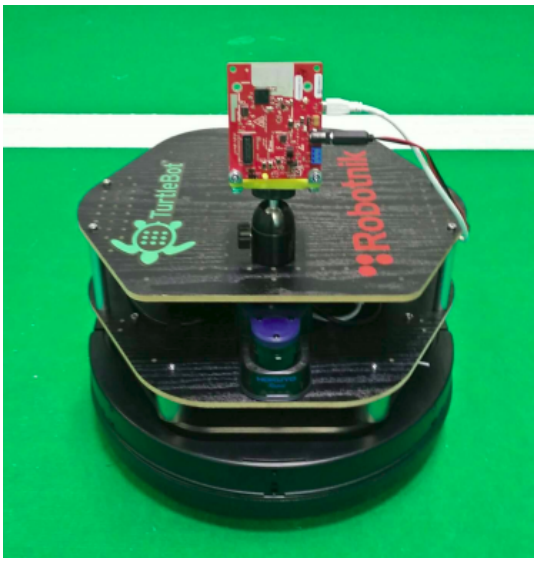


Figure 6.1: TurtleBot with radar installed at the top of it, with an upwards inclination of 10 degrees.

6.1 People detection by a motionless robot

A new series of tests was performed with the radar in the new position. The targets trajectories considered were the same as the ones mentioned in the fourth chapter. This time, only walking people, people in wheelchair and a mannequin were involved in the tests. In the previous scenario (radar installed in the tripod) we used two different mannequins representative of the non people case. In this scenario, only one mannequin is used. The number of tests performed with this mannequin was double relatively to the number of tests done in the fourth chapter, in order to ensure that about half of the dataset contains data samples labeled as non people targets. In the case of the wheelchair, half of the tests were done with a larger velocity than the other half, to provide tests with different velocities. Table 6.1 shows the distribution of frames for the different agents and paths.

	People walking	People in wheelchair	Mannequin
Path A	1762	3710	4557
Path B	1633	4214	4256
Path C	1899	4730	5285
Path D	1114	2810	4460

Table 6.1: Distribution of frames for different paths and agents, acquired with the robot immobilized

Figures 6.2 to figure 6.4 show the velocity patterns of a person walking, a person in a wheelchair, and a mannequin moving, using data acquired with the radar positioned in the robot. As it is possible to see, the patterns are similar to the ones obtained in the fourth chapter.

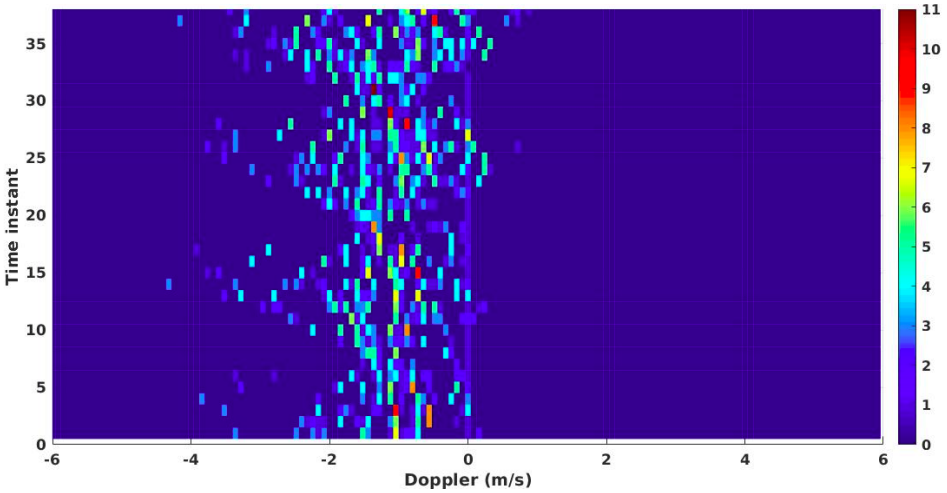


Figure 6.2: Histogramic representation of Doppler measures of a person walking in front and towards the radar (path C) installed on the robot. The colour bar on the right of the histogram represents the absolute number of points.

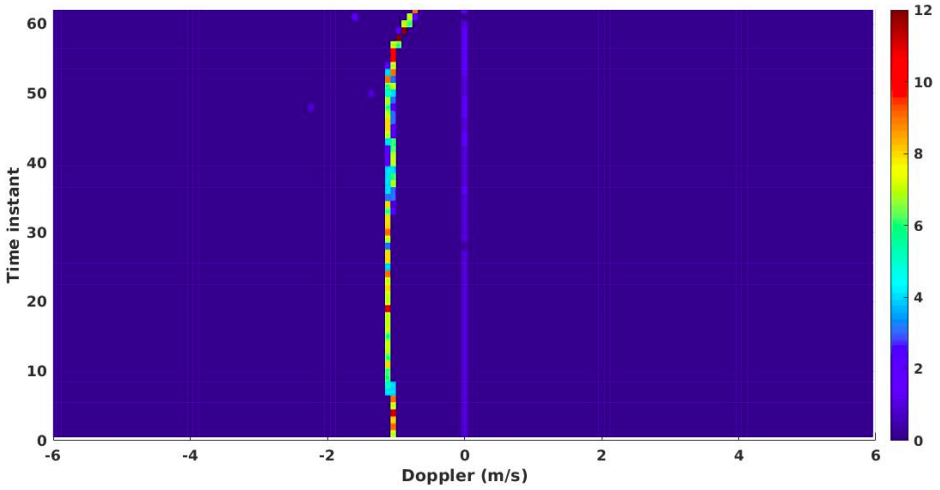


Figure 6.3: Histogramic representation of Doppler measures of a person moving in an automated wheelchair in front and towards the radar (path C) installed on the robot. The colour bar on the right of the histogram represents the absolute number of points.

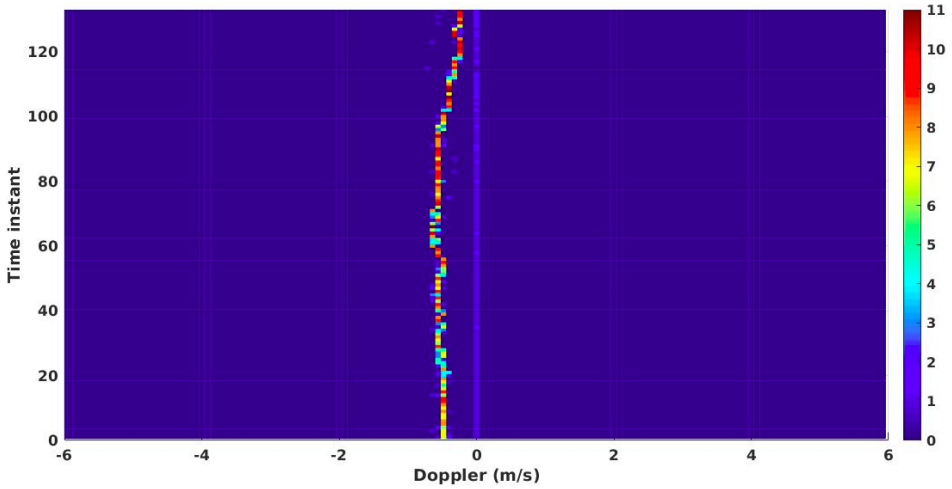


Figure 6.4: Histogramic representation of Doppler measures of a mannequin moving in front and towards the radar (path C) installed on the robot. The colour bar on the right of the histogram represents the absolute number of points.

The newly created dataset, which comprises only one frame in each data sample, has a total of 40430 data samples, of which 21872 correspond to people moving. The models were trained in 75% of the dataset and tested with the remaining data samples. Although the targets move with different absolute velocities, we want the classification models to base their performance on the dispersion of velocity and the motion pattern, rather than the absolute velocity. In other words, we don't want to exploit the absolute velocity as a feature, because this could be misleading and result in biased models of the targets. That is the reason why the tests with the wheelchair were performed with different velocities.

The classification models used were the ANN and RF with the same configuration as men-

tioned in section 4.9. The classification based on the ANN, using as features the normalized histogram bins as well as the RCS, achieved an accuracy of 96.1%. Using the same ANN with similar hyperparameters but excluding the feature RCS, the performance was 92.0%. Using the RF, the accuracy attained was 95.3%. As in the case of the ANN, when the RCS is not used as a feature the accuracy decreases, in this case for a value of 91.9%. The confusion matrices of the ANN and Random Forest are represented in table 6.2 and 6.3, respectively.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	4487	175
Person	217	5229
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	4357	305
Person	506	4940

Table 6.2: ANN confusion matrices with data acquired in a motionless robot.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	4450	212
Person	263	5183
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	4316	346
Person	468	4978

Table 6.3: RF confusion matrices with data acquired in a motionless robot.

6.2 People detection by a moving robot

Based on the successful experiments performed in classifying targets with the device in the robot, we proceed to a new experimental configuration where the robot that carries the radar is moving. As the device is moving, the tests planned are different from the previous ones. The tests to be considered are the following (refer to figure 4.2 for a visualization of the paths):

- Target going to robot and robot moving towards the target (path C).
- Target moving away from robot and robot moving towards the target (path C).
- Target moving transversely (path A) in relation to the robot, in front of it, and robot moving towards the target (path C).
- Target moving diagonally (path D and B) in relation to the robot, in front of it, and robot moving towards the target (path C).

Three different targets were tested: people, people in wheelchair, mannequin (the last is a counter example of a person). The total number os tests made is 420.

6.2.1 Dynamic detection issues

In the present scenario of tests we need to address the following implementation issue: the targets that contribute with points to the point cloud detected by the radar are only those who have an apparent movement relatively to the sensor. In these tests, since the sensor is moving, all targets have apparent movement relatively to the radar, and thus, all targets contribute with points to the acquired point cloud. The point cloud generated is thus larger than when the radar was immobilized. The first tests revealed that this increase of information (points) is enough to crash the radar signal processing, due to the excess of data to process in real-time. The solution was to alter the configuration used, and to reduce the number of ADC samples collected during sampling time, from 128 to 64.

6.2.2 Clustering limitations and basic tracking

Another consequence of the above-described issue was that the clustering method wasn't robust enough to detect and retrieve only the cluster of interest. As the sensor is moving, there will be more clusters now than when the robot was immobilized. Thus, choosing the largest cluster as the cluster of interest (as done in the previous implementations) does not guarantee that the cluster chosen corresponds to the target of interest. It may be caused by a large, static target that has now an apparent motion relatively to the radar. Therefore, in order to process the data, it is necessary to develop a method that can distinguish and select the cluster of interest and follow that cluster.

There are two aspects that characterize the clusters originated from noisy points of clutter: they are random, and can appear and disappear between successive frames; and they usually are placed on a constant position relatively to the sensor, because most of this clutter corresponds to nearby reflections from the ground where the robot is moving, whose distance to the radar remains constant. The cluster of interest, on the other hand, does not stay in the same position, because in every test performed, the target is moving (normally with a velocity superior to the robot), and also, the cluster of interest doesn't disappear between frames. Furthermore, because frames have a periodicity of 0.05 seconds, a person, wheelchair or mannequin cannot move a large distance during that period, i.e., in this time interval it is unfeasible for this target to travel large distances. Armed with these premises, we can develop a tracking method that is able to reject the clutter and false targets, and to select and follow the cluster of interest in each test. The next paragraphs detail this heuristic-based tracking method.

6.2.3 Dynamic clustering algorithm

The heuristic method adopted to implement the new clustering algorithm uses the following principles: for a given frame, we start by applying the usual clustering method using the coordinates of the points, and then choose the largest cluster. The next step is to calculate its centroid. The centroid will be used to compute the distance between the cluster chosen on the current frame and the cluster chosen on the previous frame. On the first iteration, only the storage of the current centroid is done. In the succeeding iterations, start by calculating the current centroid coordinates and then the distance to the previous cluster centroid. Every cluster centroid is determined, and then the closest one is chosen. If, however, the closest centroid is located at a distance superior to 1.6 meters, then this means that the target would have travel at a very high velocity, which is unfeasible. The value of 1.6 meters was chosen

based on practical experiments, that showed that this value was adequate. If the distance between the previous and the current cluster centroid is larger than 1.6 meters, it is concluded that the cluster chosen has no **spatial continuity** with the previous one, and thus, the script is alternating between clusters. This means the cluster chosen in the beginning or in the previous iteration was a noise cluster which disappeared in the present iteration. The solution is to go back to the first iteration, and choose a different cluster to start with. If the cluster chosen in the beginning is the cluster of interest, then the algorithm should be able to follow that cluster. Even if in some iteration it firstly chooses another one, it will correct itself and choose the right cluster again to guarantee spatial continuity. If however the cluster chosen in the beginning is a noise cluster, two things may happen:

- The noise cluster may at some iteration, disappear. Then, because spatial continuity is not ensured, the script will restart the iterations and start on a different cluster, which can be the cluster of interest.
- However, it can happen that the cluster of noise is always present, ensuring the spatial continuity. To detect this hazard and correct it, we consider that (as described before) the cluster of interest is moving in relation to the radar, even with the two elements moving, and has to travel a distance equal or higher than 1.5 meters (value empirically chosen). Normally, the noisy clusters that appear continuously in the FoV of the radar are a result from nearby reflections in the ground, that appear to be static in relation to the sensor, meaning that they are always in same position of the FoV of the radar. By simply storing the distance traversed by each cluster centroid, we can detect if the cluster chosen was a noise cluster (that appears to be static in the FoV of the radar) or was the cluster of interest.

Algorithm 1 shows the pseudo code of this tracking method.

Algorithm 1: Heuristic tracking algorithm

```
Iteration = 1; TotDistance = 0;
while there are data frames do
    if Iteration == 1 then
        Compute and store centroid of cluster;
    else
        Calculate closest centroid to previous one;
        Store centroid of cluster;
        Displacement = distance between previous and closest centroid;
        if Displacement > 1.6 then
            Iteration = 1; TotDistance = 0;
            Start on different cluster;
            Continue;
        end
        TotDistance = TotDistance + Displacement;
    end
    if Iteration == total number of frames then
        if TotDistance < 1.5 then
            Iteration = 1; TotDistance = 0;
            Start on different cluster;
            Continue;
        end
    end
    Iteration = Iteration + 1;
end
```

Another aspect to note is that the distance traversed by the centroid of the cluster (that is

being tracked at a given instant) must not be iteratively updated based on the successive values of the Cartesian distance between the two centroids, because the centroid in a static cluster always has minor coordinate changes. What this means is that, between successive frames, even if a noisy cluster is always located in the same region in relation to the radar, the cluster centroid will have always some natural coordinate changes, due to the nature of the radio reflections, that originate different points detected between successive frames, even if the cluster detected is the same as the one detected in previous frames. Therefore, if we consider the Cartesian distance, the distance traversed will always increment from frame to frame, even if the cluster is alternating between the same positions. The solution is to accumulate the coordinate differences in each iteration, and only in the end calculate the Cartesian distance. This way, if the centroid revolves around the same position, the total distance traveled will not increase.

Figure 6.5, figure 6.6, and figure 6.7 show the pattern of velocity distributions of person, person in wheelchair and mannequin (respectively) moving in direction to the radar and robot moving in direction to the person/mannequin.

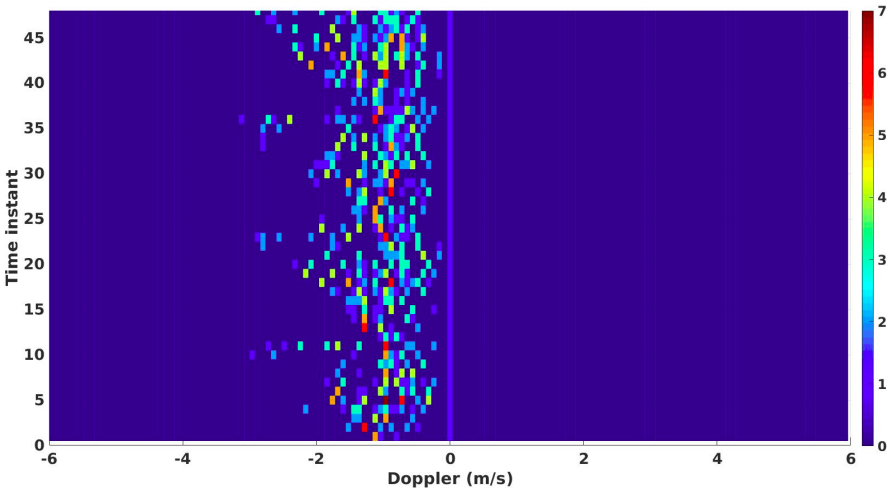


Figure 6.5: Histogramic representation of Doppler measures of a person walking in the direction of the radar and the robot moving towards the person. The colour bar on the right of the histogram represents the absolute number of points.

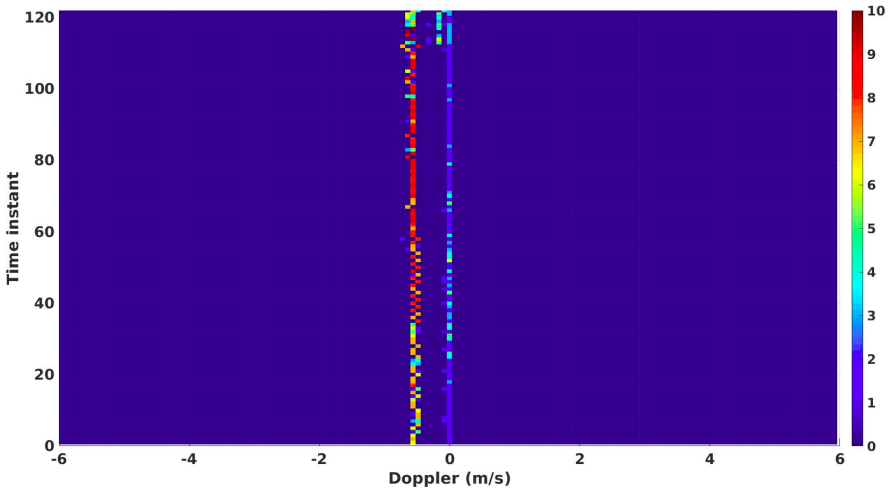


Figure 6.6: Histogramic representation of Doppler measures of a person moving in an automated wheelchair in the direction of the radar and the robot moving towards the person. The colour bar on the right of the histogram represents the absolute number of points.

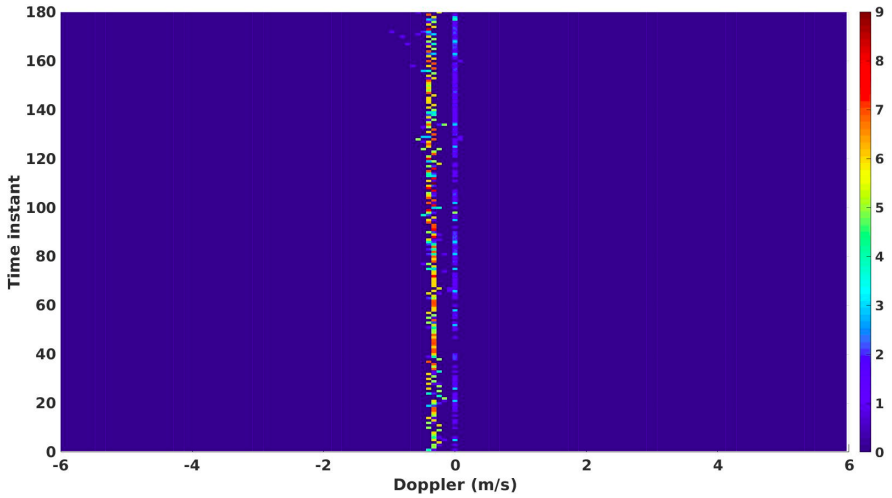


Figure 6.7: Histogramic representation of Doppler measures of a mannequin moving in the direction of the radar and the robot moving towards the mannequin. The colour bar on the right of the histogram represents the absolute number of points.

6.2.4 Classification results with dynamically deployed radar

In the current experiment, the complete dataset consists of a total of 25431 frames. Out of these 25431 frames, 12720 correspond to people moving, and the remaining 12711 correspond to other moving targets. Table 6.4 shows the distribution of frames for agents and paths, for the case of the dynamically deployed radar.

	People walking	People in wheelchair	Mannequin
Path A	1443	1615	4478
Path B	1366	1503	2409
Path C	1635	1580	2397
Path D	1460	2118	3427

Table 6.4: Distribution of frames for different paths and agents, acquired with the dynamic robot.

25% of the dataset was used for test, while the remaining was applied in the training stage. Using the ANN described in the previous chapter and considering the RCS as a feature, the accuracy attained was 97.7%, while ignoring the RCS the performance was 96.2%. For the Random Forest, the accuracy were 97.8% using the RCS as a feature, and 96.5% without RCS. The resultant confusion matrices of the ANN and Random Forest are represented in table 6.5 and 6.6, respectively.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	3135	66
Person	83	3074
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	3076	125
Person	115	3042

Table 6.5: ANN confusion matrices in the case of a moving robot.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	3133	68
Person	74	3083
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	3070	131
Person	91	3066

Table 6.6: Random Forest confusion matrices in the case of a moving robot.

6.2.5 Discussion of results attained with radar positioned on the robot

In the first approach, the robot that carries the radar is static. As it was possible to see, the histogramic representation of the Doppler measures of the different agents traveling each one of the considered paths was similar to the representation generated when the radar was positioned on the tripod. The classification results attained with this configuration were significantly better than the ones attained in the previous chapter. The acquisition of data with the radar deployed dynamically by the robot revealed that the clustering mechanism was not robust enough to deal with this data acquisition scenario. A basic heuristic tracking algorithm is enough to mitigate the problems resultant from the motion of the robot. The histogramic representation of the Doppler measures resultant from the motion of the different targets traversing the different paths, when the robot was immobilized or in motion, was similar to the one attained in the fourth chapter. The results attained were better than the ones attained with the static robot. In all cases, the RCS continues to improve the performance of the classification models.

6.3 Real time application

The results previously presented were obtained with the application functioning offline; the data is retrieved by the radar, stored in a file on the computer system, and then replayed and processed. In a realistic scenario, the application must function online, i.e., continuously retrieving and processing the data within an acceptable delay. As mentioned earlier, the data retrieval and processing part of the application is coded in Matlab. However, to programatically connect to the ROS nodes to receive the data in real time, Python is often used. For such, all the methods and data structures used in the Matlab version of the application were "translated" into Python scripts.

After training the models, the **Joblib** library from Python allows the users to store the models in files with the extension *".joblib"*. In essence, what is being stored is the Python object associated with the implemented model. In the same way the Joblib library allows the users to store their models, it also allows them to load a previously stored model. By loading the classification model, we can apply it again to classify new data, without the need to retrain again the model, because the weights of the model were stored. To perform real-time data retrieval and subsequent classification, the data retrieved using the scripts developed (by exporting the Matlab coded methods to Python) is used as input to the loaded models, and finally we can reach a classification output. If several people move in front of the radar device at the same time, only one of them will be classified, because the method presently implemented only selects one cluster in each frame. The interface for the real time application is the **rviz** GUI. The point cloud retrieved from the radar is displayed in rviz, as depicted in figure 6.8.

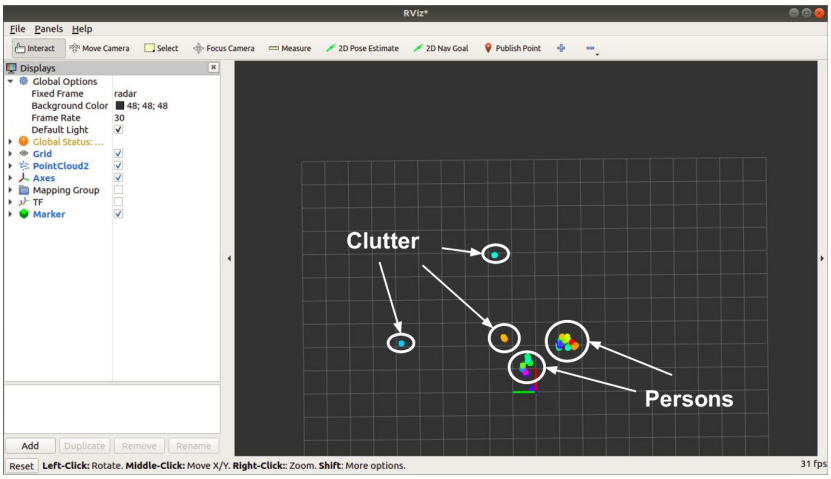


Figure 6.8: Point cloud retrieved from the radar, represented in rviz. Two people are in the FoV of the radar, and there is some clutter present in the point cloud.

After segmenting the point cloud, the largest cluster is classified. If the target is classified as a person, a green circle is drawn, otherwise a red circle is drawn, with the center positioned on the centroid of the cluster. This can be seen in figure 6.9, where a person is walking in front of the radar.

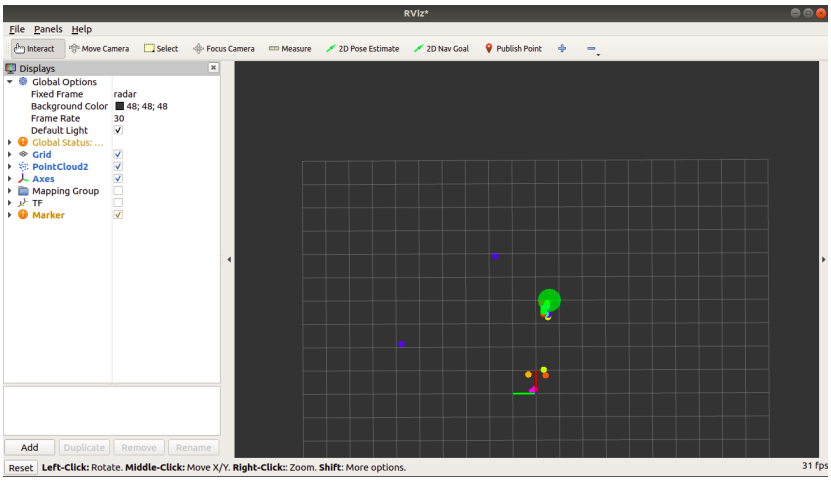
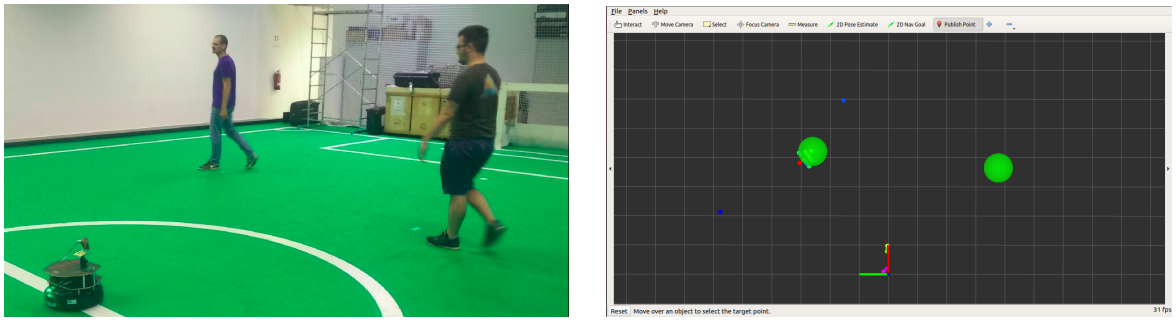


Figure 6.9: Person walking in front of the radar is detected, and a green solid circle is drawn above the points that belong to that target.

6.4 Multiple and simultaneous target detection and classification

Until this point, this dissertation only considered the case of one person/non-person classification at each time instant, applying for this purpose clustering methods to isolate an unique target. This was particularly relevant in the training stage, where it was imperative that only one target was detected at each time to ensure a clear selection of the type of target observed during the data acquisition phase. However, after we have trained the classification models, it is not necessary to ensure that only one cluster is classified at each time. In fact, because (after training) the classification models have the ability to classify correctly the targets, we can rely on the DBSCAN clustering method to isolate clusters and detect clutter, and use the trained models to classify all clusters. This is called multiple and simultaneous target classification, and can also be performed in real time. Figure 6.10 shows, at the left, two volunteers walking in front of the radar that is placed in the turtlebot in front of them, and at the right, the rviz view.



(a) (b)

Figure 6.10: Two people walking in front of the static sensor (left) and the graphical representation of this experiment in rviz (right).

As mentioned, a non-person target is marked with a red circle. Figure 6.11 shows, at the left, a person walking in front of the radar while, at the same time, a ball bouncing in the FoV of the radar. At the right of the same figure, we can see the respective target clusters being classified on the rviz GUI. Note that, although the classification models were trained with the mannequins as counter examples of people moving, they were able to classify a bouncing ball (a target they have never "seen" before) in front of the radar as non-person target.

(a) (b)

Figure 6.11: A person walking and a ball bouncing in front of the static sensor (left) and the graphical representation of this experiment in rviz (right).

6.5 Distinguish closely spaced targets

If we consider more than one target moving in front of the radar, the clustering method may fail under certain circumstances. Specifically, consider two people walking in front of the radar, depicted for example in figure 6.12.

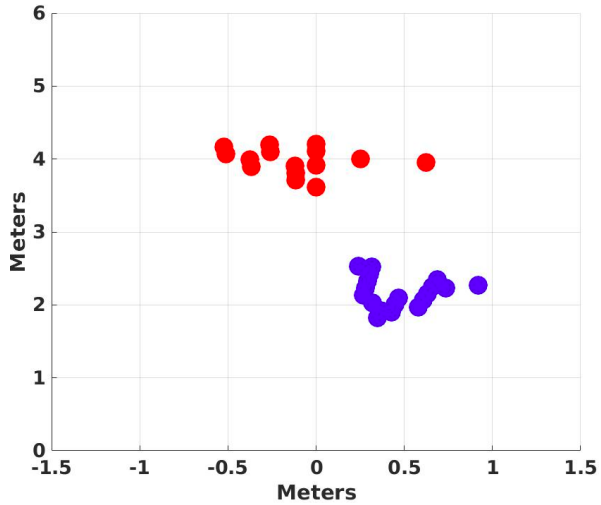


Figure 6.12: Point clouds of two people spatially apart.

In this case, the people are relatively distanced from one another, and the amount of clutter in the FoV is also minimum. In these conditions, the DBSCAN algorithm can easily detect two clusters, because, spatially, the clusters are separated. Consider now that these two people get very close to each other. Theoretically, we would want the DBSCAN method to distinguish the two different targets, but because this method relies on the spatial distance between the clusters, which is practically non-existent in this case, the two clouds are in fact misinterpreted as one cluster, as shown in figure 6.13.

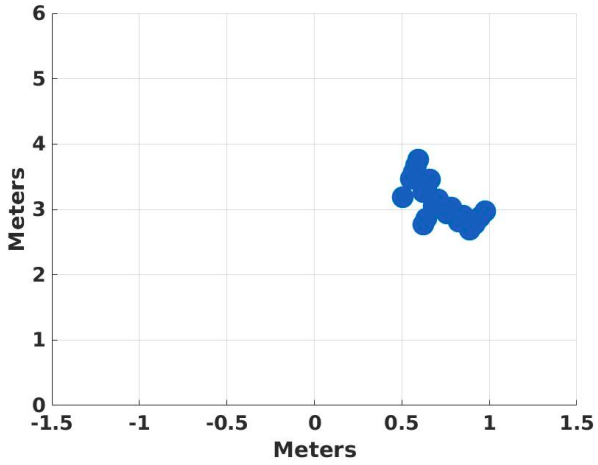


Figure 6.13: Point cloud of two people very close to each other.

Note that figure 6.12 and 6.13 both belong to the same test recorded, where two volunteers walk in front of the sensor and randomly cross paths with each other at the same time. Targets can be miss-classified as one if they are both visible from the radar but are very closely spaced. Targets can also be miss-classified as only one target if one of the targets is being occluded by the other.

In tracking applications, occlusion of one target or agent by other static or dynamic targets present in the environment is a typical, well-studied problem. This is defined as the **occlusion problem**. An effective and robust solution to this problem implies the application of advanced tracking and data association methods which have not been implemented in the present work due to time limitations. Some of these methods, envisaged as future work, include the techniques described in [4], [11] and [31]. In this work we adopted a basic, heuristic clustering method designed to cope with the occlusion problem, described bellow.

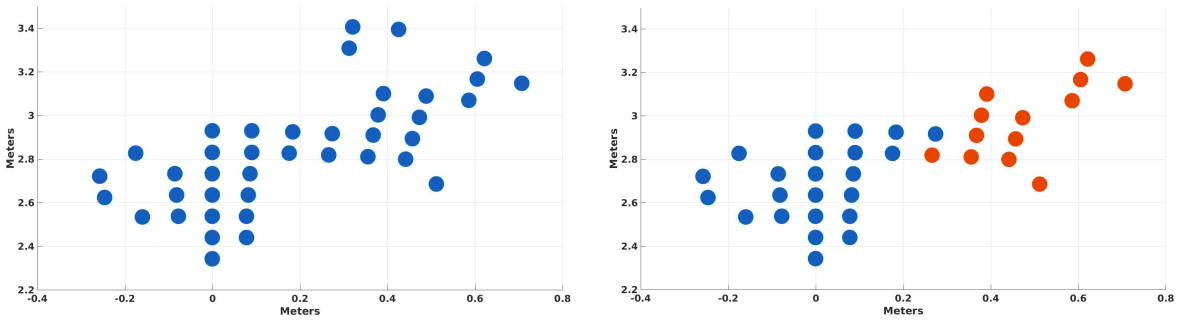
The enhanced and more complex clustering mechanism relies in additional features of the points in order to distinguish targets that are spatially close but must be associated to distinct clusters. We have used before the Doppler feature of the data to train classification models to classify targets. The Doppler can also be used as a feature for the clustering method. Consider two people closely spaced, but each one moving in opposite directions (opposite velocities); if we apply clustering using the Doppler as a feature and ignoring the positions of the points of the point cloud of each target, theoretically, in the situation represented in figure 6.13 the method should separate the two people due to their different velocities. However, we can not use only the Doppler feature to distinguish targets, because this will assume as belonging to the same clusters targets that are spatially distant from one another, but have the same velocity. Therefore, we have to consider both the position and the velocity of each target.

6.5.1 Cascaded clustering

In practice, point clouds that are spatially distant belong to different clusters. Point clouds that are spatially close may belong to different clusters if their velocity is different. In a first stage, the targets should be separated by their position (clustering in position). On a second stage, each cluster is *subclustered* using the Doppler feature (clustering in velocity). Notice that, by applying this method, the different parts of the human body can be *subclustered* in different clusters due to their different velocities. Possible solutions to this problem are presented in the next subsection. This method allows to separate clusters that were, in the first stage, assigned to only one major cluster due to their positions. We call this approach **cascaded clustering**, because we apply the different clustering methods in succession in order to separate targets. When applying the DBSCAN with the Doppler feature, we consider the values of $Eps = 0.3$ and $MinPts = 8$. As in subsection 4.4.2, these values were chosen based on experiments.

The application of clustering using the Doppler as a feature is able to distinguish people that cross paths when they are walking towards and away from the radar. Figure 6.14 shows two images of a point cloud of two people that are close to each other, obtained with the two approaches:

- cascaded clustering is not applied (left).
- cascaded clustering is applied (right).



(a) (b)

Figure 6.14: Point cloud before applying cascaded clustering (a) and after applying cascaded clustering (b).

As shown in the plots, the two targets can be distinguished. Notice also that some of the points of the point cloud that has been processed by the cascaded clustering method are not associated to the targets, whereas in the case where only spatial clustering is used, this points are associated to the targets. Therefore, information can be loss when applying the cascaded clustering method.

6.5.2 Occlusion problem and limitations of cascaded clustering

When targets move in the current experimental setup, they are prone to be occluded by having some other target between them and the sensor. Figure 6.15 shows the result of a point cloud after applying the cascaded clustering when the targets trajectories overlap. As it is possible to see, in this case the cascaded clustering method is not effective. This can be explained by the fact that the occluded target is not detected, and therefore, only points of one target are being clustered.

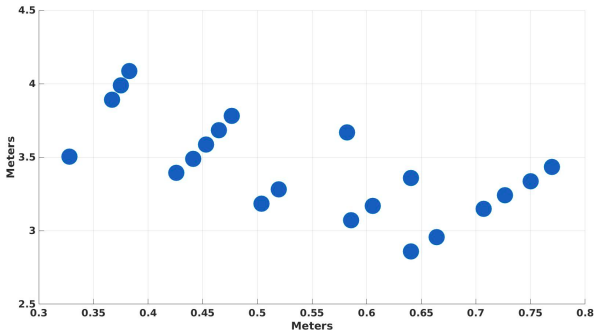


Figure 6.15: Point cloud of two people crossing transversely using the cascaded clustering technique.

Another issue with this approach is that this clustering technique is prone to sometimes wrongly divide clusters related to one target in more subclusters. This is due to the different motion of the different parts of the human body; because they have different velocities, they can be subclustered in different clusters. Although this problem can be mitigated by altering the DBSCAN parameters (Eps and $MinPts$) the solution may also reduce the efficacy of the method in distinguishing two targets too closely spaced. There is, therefore with this method, a trade-off between the detection of the motion of limbs in a single target, and the discrimination of two targets shortly distanced from one another.

6.6 Discussion of results attained with simultaneous multiple target classification

In the previous section it was demonstrated the effectiveness of the application at classifying simultaneous multiple targets. Furthermore, it was shown that the coexistence of multiple targets in the FoV of the radar can, in specific circumstances, raise issues regarding their correct detection.

When the targets are relatively distant from one another, the clustering method used is sufficient to isolate the targets, and a correct classification output can be achieved. When the targets are spatially close, the clustering method fails to segment correctly the point cloud, and multiple targets can be clustered as a single target. Another consequence of the coexistence of multiple targets is that one target can be occluded by another target, which makes its detection by the radar unfeasible in the current scenario. To resolve this limitations, a cascaded clustering method that relies on the Doppler information of the points to differentiate the different targets was developed. The clustering method is defined as being a two stage mechanism, where in the first stage the targets are separated by their relative

positions; in the second stage, each cluster defined in the previous stage is further segmented based on the Doppler information of its points. The key idea is that clusters spatially distant belong to different targets; if two clusters are close to one another, they can be separated if their velocities are different. This method is robust enough to differentiate targets that are closely spaced, except if one of the targets is being occluded by other. In this situation, it is necessary to use a tracking algorithm to be able to detect and distinguish the two targets.

Chapter 7

Summary and conclusions

7.1 Summary

7.1.1 System development and tests

As can be seen from the results presented, the performance of the classification models tested in this dissertation clearly evidence the high potential of the methods to detect people in indoor environments, even if moving in an automated wheelchair. The inclusion of the RCS of each detected target as a classification feature was shown to improve significantly the performance of the target recognition methods in both models and in the different cases considered (radar in tripod, radar immobilized in TurtleBot and radar dynamically deployed by the TurtleBot). Even using only the values of the RCS, whose calibration was beyond the scope of this dissertation, it is possible to show that the relative values of this parameter, when acquired by the same radar with fixed parameters, is sufficient to discriminate some classes of targets in a given scenario.

Although the tests with the moving TurtleBot hampered the tracking and distinction of the cluster of interest, the classification methods had a better performance in this case than when the robot was immobilized. In the experiments were the TurtleBot was immobilized, the ANN had a better performance than the RF. In the experiments where the Turtle Bot is moving, the RF had a better performance than the ANN. The RCS had a larger impact on performance in the experiments where the TurtleBot is static, comparing to the experiments where it is moving. If we consider the experiments where the radar was placed in the top of the tripod, the RF always has a better performance than the ANN. The impact of the RCS on the performance of the models in this last case was similar to when the radar is static in the TurtleBot.

We showed that the approach is effective in two distinct scenarios (robot immobilized and robot moving) which can lead to a wide range of applications in robotics. The presence of clutter and its deleterious effect, which can be typical in indoor environments, was also solved with a simple heuristic tracking algorithm.

The feature extraction process and classification can be performed within small time intervals that are adequate for real time applications. In fact, it was proven that this application can detect multiple targets simultaneously and in real time. The execution time of the application increases with the increase in the number of features in a data sample. The bottleneck in terms of execution time is located in the feature extraction process, since the classification stage, when comparing to the extraction of features, is almost non affected.

The detection of more than one target at the same time originated other problems, namely the occlusion problem and the perception of more than one target as a single target. The miss segmentation of the point cloud and the association of more than one target to a single one can be solved by developing a more robust clustering method that relies on additional characteristics of the targets besides their position in space. However, this method can only function properly if a target is not being completely occlude by other (occlusion problem). If a target is being occluded by other target, then the radar sensor will not be able to detect it, and thus it can not distinguish the two targets, only the one "visible" from the point of view of the radar.

7.1.2 Publications and presentations

Some of the results achieved in this dissertation resulted in two papers that were later published and presented in two conferences:

- *Encontro Português de Inteligência Artificial (EPIA)*, held in Vila Real, Portugal, between the 3^{rd} and 6^{th} of September. The title of the paper submitted for this conference was *"Machine learning for radar-based people detection"*. The scientific article is available at `https://link.springer.com/book/10.1007%2F978-3-030-30244-3`. See also **Appendix A**.
- *Fourth Iberian Robotics Conference - ROBOT2019*, held in Porto, Portugal, between the 20^{th} and 22^{nd} of November. The title of the paper submitted was *"Machine learning methods for radar-based people detection and tracking by mobile robots"*. See also **Appendix B**.

The developed application was also exhibited at:

- *Encontro Ciência 2019* held in Lisbon, an event devoted to interchange and presentation of the scientific work developed in Portugal,
- *TechDays* held in Aveiro, an event with the purpose of showing to the community the most recent developments in technology that is developed by different companies and investigation groups of the region.

7.2 Conclusions

One of the main results of this work is the demonstration that the Doppler of each target, calculated by the radar device, is a feature that can be effectively used to detect different types of targets, because different targets present different motion patterns. By representing the Doppler values of the points of the point cloud of each target as an histogram, the data is organized under a fixed format, which enables its application in ML methods that usually require that the different inputs have a fixed size. The RCS does not depend on the kinematics of the target, but on the physical properties of the target, which acts as complementary feature to the motion pattern, and improves the performance of the models at distinguishing different targets. The ML classification models can use this data to distinguish with a high level of accuracy the different types of targets.

The use of more frames per data sample increases the performance of the models, which indicates that the models are sensitive to motion pattern of targets that can repeat itself in

time. The execution time of the classification models was not visibly affected by the size of data samples used. This is a consequence of the high parallelism of the models used, and the fact that the increase in the dimension of the features only impacts part of the models architecture. For example, in the case of the ANN, only the input layer changes its size with the number of features. Moreover, the different methods of feature extraction and classification take negligible time intervals to compute, which allows the application to be executed in real-time. Since the target kinematics can be represented as an image, the extraction of Haralick features was implemented in order to reduce the dimension of the data. However, this approach resulted in poor model performance. This is probably due to the fact that these features are often used in 2D dimensional data (such as images), but in this work they were applied only in 1D dimensional data (one histogram). The performance of the classification models based on Haralick features can potential be improved when extracted from bi-dimensional data samples built from multiple frames, as proposed in this work. In this application, we showed that a simple manual search to optimize the hyperparameters of the ML models was enough to attain good performances. Other searching techniques (grid and random search), which take considerable amount of time to perform all the necessary computations, were not able to significantly improve the performance of the ML models.

The type of data used can be acquired with the radar positioned in different locations. The target kinematics, as observed by the radar, is similar both in the case the radar is static or moving. This allows the application to function in a diversity of operational conditions. When the radar is being dynamically deployed by a moving robot, the observable clutter increases, which requires the application of more complex algorithms to filtrate it. The results with the moving radar were better than the ones attained with the static radar, which can be explained by the motion of the robot that leads to small variations in the radar point of view, with the potential of increasing the information extracted from the moving target. Multiple targets can be detected simultaneously, by using an adequate clustering mechanism (DBSCAN) to isolate the different point clouds of each target. The clustering method that relies only on the spatial position of the points of each target in 2D plane was not enough to discriminate different targets that are spatially close to one another. By considering the Doppler information of the points of each target, it is possible to discriminate multiple targets spatially close. However, this method is not enough to detect targets that are being occluded by others. In these conditions, only appropriate tracking mechanisms can distinguish all the targets in the radar FoV.

7.3 Future Work

The types of features and ML methods used in this work, as well as the optimization techniques and clustering methods, allowed to achieve the good results for target classification presented in the previous chapters. There are, however, different techniques that could have been applied in the development of this work, and that are detailed in following subsections.

7.3.1 Advanced methods for hyperparameter optimization

In the context of model optimization, this dissertation covered three main methods of search in the hyperparameter space, namely: manual, grid, and random search. These methods already provide a high performance of the models. Given the nature of the models and the search methods used, it is very unlikely that other optimization techniques could lead

to major improvements of performance. However, in future implementations of this type of system in different contexts, other optimization techniques could be applied due to their potential computational efficacy in terms of exploration of the hyperparameter space. Among the suggested approaches, two techniques are proposed: **evolutionary algorithms** (EA) and **Bayesian optimization** (BO).

EA are used to find optimal (or suboptimal) solutions for a given problem. This class of algorithms mimics the process of natural selection and evolution in nature; thus the name evolutionary algorithm. The key concepts of EA are **reproduction**, **mutation** and **selection**. Each possible solution is defined as a single **individual**. A set of individuals is defined as a **population**. Each individual is encoded as a string composed of characters of some alphabet (binary, integer values, etc). The initial population is generated by randomly generating these strings. The initial population is also considered the first **generation**. A **fitness function** is used to evaluate the initial population's performance in the problem domain according to some predefined metric. The probability that an individual is selected for the reproduction stage is a function of its performance. Thus, individuals that have a better performance have a higher probability of being selected for the reproduction stage. The individuals selected for reproduction are combined between each other and generate new individuals (a new generation). This means that the new individuals have shared characteristics from their parents. The offspring is further altered by means of mutation, that introduces new characteristics in the individuals, which may or may not enhance their fitness. This process is repeated until some criteria is met, such as the number of generations or a certain fitness value [32]. The idea of this approach is that the overall fitness of the individuals increases over generations (iterations) because the most fitted individuals are being used to reproduce amongst each other. This type of approach has been applied in the design of ANNs, by combining different network architectures that have shown the best performances in previous iterations. However, the method is computationally intensive due to the amount of evaluations that need to be performed to derive the best architecture, and does not guarantee that an adequate set of hyperparameters will be found.

The second approach is based on the use of BO. This is a more advanced technique when compared to the techniques used in this work. BO has been widely used in recent years and offers a different paradigm than grid or random search; while in the last mentioned methods we have to provide the values for the hyperparameters to be tested, BO is in charge of deriving precisely that. This method starts by considering an initial set of hyperparameters and their respective performance, and by using Gaussian processes, tries to predict what will be the best parameters that will yield a better performance. These parameters are tested to confirm or not the prediction of the method, and the process is repeated, until, eventually, a set of hyperparameters that allow for a satisfactory performance are found.

7.3.2 Reinforcement learning for model training

Throughout this dissertation, only two ML paradigms have been considered; supervised learning (ANN and RF) for training of the models, and unsupervised (DBSCAN) for clustering of the point cloud, although in subsection 3.1.3 we have mentioned another type of ML method: **reinforcement learning**. This type of ML method could also be used to train the ANN, RF, or any other classification model. The use of supervised learning to train and adjust the models parameters based on a specific training set implies that the data has been previously acquired and stored. With a reinforcement learning method, the data could

be acquired and the models could make predictions in real time. An autonomous agent could automatically receive some negative reward when it classifies wrongly a human as a non-human target, and a non-human by a human target.

7.3.3 Exploit different classification models

Although only two ML models (ANN and RF) have been applied in this work, others could have been tested. Neural networks are divided in different types, and in particular, **recurrent neural networks** (RNN) have specific characteristics that make them interesting candidate models to be applied. The motion of people is characterized by periodic movements that are repeated in time. The cadence of these movements is thus a characteristic of the motion of people. The models applied so far only consider and classify snapshots of the movement (or a concatenation of them if we use more than one frame per data sample), and have no sensibility for how these movements and snapshots are repeated in time. This is an important feature that should be considered by the ML models. RNNs are used to model temporal dynamic behaviors, and therefore allow to capture patterns that have a certain periodicity in time, such as the motion of people.

7.3.4 Tracking methods

We have seen that the clustering method that relies not only in the position of the points of the point cloud, but also on their velocity, can distinguish targets closely spaced. However, the occlusion problem can not be solved simply using this method. To solve this issue, the most adequate approach is in fact to apply a tracking mechanism, based for example on (Extended) Kalman Filters or Particle Filters, that can follow the different targets, even if they are temporarily occluded by some other targets. Even though these methods were not applied in this work, some exploratory work was already developed. **Appendix C** is extracted from the technical report written for the research grantship of project RETIOT, and details the implementation of a Kalman Filter that uses the position and velocity of the target measured with the radar, and outputs predictions about the target's real position. The developed Kalman Filters work with data that has been processed *a priori* by the People Counting Demo of TI. This application performs the clustering of the point cloud and returns information about each target position.

7.3.5 Future applications

This dissertation only focus on the detection of people moving. However, other applications could be implemented using the retrieved data and the datasets that were created. The detection of people that are not walking or moving, but yet perform small amplitude movements, such as people seated performing some tasks, is an interesting application. Similarly, the detection of falls is important in the context of elderly monitoring. From the data collected, other aspects related to people motion could be derived, such as the trajectories that are being traveled by each agent, which can be relevant for example in intrusion detection or patients tracking in hospitals.

Acknowledgments

This work is supported by Project RETIOT, "Reflectometry Technologies to Enhance the Future Internet of Things and Cyber-Physical Systems" funded by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework and Fundação para a Ciência e Tecnologia (FCT), under contract POCI-01-0145-FEDER-016432.

Bibliography

[1] Massimo Guarnieri. The Early History of Radar. *Industrial Electronics Magazine, IEEE*, 4:36 – 42, 10 2010.

[2] Tapan K Sarkar and Magdalena Salazar Palma. A history of the evolution of RADAR. In *2014 44th European Microwave Conference*, pages 734–737. IEEE, 2014.

[3] J.A. Scheer and W.A. Holm. *Introduction and radar overview*, pages 3–58. Institution of Engineering and Technology, 01 2010.

[4] Michael Livshitz. Tracking radar targets with multiple reflection points. `https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/1023/Tracking-radar-targets-with-multiple-reflection-points.pdf`, 2018. [Online; accessed 13-June-2019].

[5] Suleyman M. Suleymanov. Design and Implementation of an FMCW Radar Signal Processing Module for Automotive Applications. Masters thesis, University of Twente, The Netherlands., 2016.

[6] Sandeep Rao. Introduction to mmWave sensing: FMCW radars. *Texas Instruments*, 2018.

[7] James Cooley, Peter Lewis, and Peter Welch. The Fast Fourier Transform and Its Applications. *IEEE Transactions on Education*, 12, March 1969.

[8] Texas-Instruments. People Tracking and Counting Reference Design Using mmWave Radar Sensor. TI Designs: TIDEP-01000., March 2018.

[9] Texas-Instruments. People Counting User's Guide., July 2019.

[10] H. Yamada, Y. Wakamatsu, K. Sato, and Y. Yamaguchi. Indoor Human Detection by Using Quasi-MIMO Doppler Radar. In *2015 International Workshop on Antenna Technology (iWAT)*, pages 35–38, 2015.

[11] N. Knudde, B. Vandersmissen, K. Parashar, I. Couckuyt, A. Jalalvand, A. Bourdoux, W. De Neve, and T. Dhaene. Indoor Tracking of Multiple Persons With a 77 GHz MIMO FMCW Radar. In *2017 European Radar Conference (EURAD)*, pages 61–64, 2017.

[12] Steffen Heuel and Hermann Rohling. Pedestrian Recognition Based on 24 GHz Radar Sensors. volume Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications, chapter 10, pages 241–256. InTech, 2013.

[13] Christian Berkius, Max Buck, Jonatan Gustafsson, and Martin Kauppinen. Human Control of Mobile Robots Using Hand Gestures. Bachelor thesis in electrical engineering, Chalmers University of Technology. Gothenburg, Sweden., 2018.

[14] Sanoal Machado and Santiago Mancheno. Automotive FMCW Radar Development and Verification Methods. Master's thesis, Department of Computer Science and Engineering. Chalmers University of Technology. University of Gothenburg, Sweden., 2018.

[15] A. Bartsch, F. Fitzek, and R. H. Rasshofer. Pedestrian recognition using automotive radar sensors. *Advances in Radio Science*, 10:45–55, 2012.

[16] Thayananthan Thayaparan, Ljubisa Stankovic, Igor Djurovic, Suresh Penamati, and Kamisetti Venkataramaniah. Intelligent target recognition using micro-Doppler radar signatures. In Kenneth I. Ranney and Armin W. Doerry, editors, *Radar Sensor Technology XIII*, volume 7308, pages 392 – 402. International Society for Optics and Photonics, SPIE, 2009.

[17] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[18] Pierre Lison. An introduction to machine learning. `https://heim.ifi.uio.no/plison/pdfs/talks/machinelearning.pdf`. Online; accessed July, 2015.

[19] Alex Smola and SVN Vishwanathan. Introduction to machine learning. *Cambridge University, UK*, 32:34, 2008.

[20] S. Kokalj-Filipovic, M. S. Greco, H. V. Poor, G. Stantchev, and L. Xiao. Introduction to the issue on machine learning for cognition in radio communications and radar. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):3–5, 2018.

[21] Eijiro Takeuchi, Alberto Elfes, and Jonathan Roberts. *Localization and Place Recognition Using an Ultra-Wide Band (UWB) Radar*, volume 105 of *Springer Tracts in Advanced Robotics*. Springer, 2015.

[22] Dennis Barrett and Adrian Alvarez. mmWave radar sensors in robotics applications. Technical report, Texas Instruments, 2017.

[23] Sedat Dogru and Lino Marques. Using Radar for Grid-based Indoor Mapping. In *Proc. 19th IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC'2019. April24-25.*, Gondomar, Porto. Portugal, 2019.

[24] S. Z. Gurbuz and M. G. Amin. Radar-based human-motion recognition with deep learning: Promising applications for indoor monitoring. *IEEE Signal Processing Magazine*, 36(4):16–28, July 2019.

[25] 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*, KDD'96, pages 226–231. AAAI Press, 1996.

[26] I.A Basheer and M Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3 – 31, 2000. Neural Computting in Micrbiology.

[27] D.L. Gupta, A.K. Malviya, and Satyendra Singh. Performance analysis of classification tree learning algorithms. *International Journal of Computer Applications*, 55(6), 2012.

[28] Tommy Löfstedt, Patrik Brynolfsson, Thomas Asklund, Tufve Nyholm, and Anders Garpebring. Gray-level invariant haralick texture features. *PLOS ONE*, 14(2):1–18, 02 2019.

[29] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, Nov 1973.

[30] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, February 2012.

[31] Yaakov Bar-Shalom, Fred Daum, and Jim Huang. The probabilistic data association filter. *Control Systems, IEEE*, 29:82 – 100, 01 2010.

[32] Andrew Chipperfield and Peter Fleming. Evolutionary algorithms for control engineering. *IFAC Proceedings Volumes*, 29(1):1163–1168, 1996.

Appendices

Appendix A

Paper accepted and presented at conference EPIA2019

Machine learning methods for radar-based people detection and tracking *

José Castanheira[1], Francisco Curado[1][0000−0003−0239−4675], Ana Tomé[1][0000−0002−1210−0266], and Edgar Gonçalves[1]

University of Aveiro, Department of Electronics and Informatics Engineering, 3810-193 Aveiro, Portugal.
{jp.castanheira,fcurado,ana,easg}@ua.pt

Abstract. This paper describes the work developed towards the implementation of a radar-based system for people detection and tracking in indoor environments using machine learning techniques. For such, a series of experiments were carried out in an indoor scenario involving walking people and dummies representative of other moving objects. The applied machine learning methods included a neural network and a random forest classifier. The success rates (accuracies) obtained with both methods using the experimental data sets evidence the high potential of the proposed approach.

Keywords: RADAR · Locomotion Pattern · RCS · Machine Learning.

1 Introduction

1.1 Motivation

The number and diversity of applications of radar sensors have been increasing dramatically in recent years, motivated by the emergence of inexpensive, self-contained development kits that permit rapid prototyping and testing of radar-based solutions. The reduced cost and dimension of these systems make them affordable for academic research and permit their integration in small, inexpensive robots.

The immunity of radar signals to some of the main sources of noise (such as dust, fog, rain, strong heat and fire) that disrupt measurements taken with other sensors (such as cameras, laser or infrared) is one of the most important arguments for the adoption of radar as sensory devices in autonomous systems in general and in mobile robotics in particular. Robotic applications currently operational or under development include: obstacle detection and mapping of indoor environments, ground and pavement classification, terrain-referenced navigation, Simultaneous Localization and Mapping (SLAM), and detection and tracking of people and objects in motion.

* Funded by Research Project RETIOT PT2020-03/SAICT/2015 - Fundação para a Ciência e Tecnologia.

Despite the aforementioned advantages, feature extraction from radar data poses significant challenges to conventional signal processing techniques due to the diversity of artifacts that typically affect unprocessed radar signals. This problem is especially dramatic in indoor environments due to the ubiquity of obstacles (clutter) normally present in these scenarios.

This paper addresses the problem of radar-based detection of walking people in an indoor scenario using machine learning techniques. The method proposed is envisaged as a building block of a more general robust classifier and tracker of people in indoor environments that is currently under development in our laboratory. The aim of the this work is to develop a highly reconfigurable real-time location system (RTLS) that integrates static radars with radars installed on mobile platforms and Internet-of-Things (IoT) devices. The envisioned applications include elderly monitoring and rescue operations in vision-denied environments. The paper does not discuss other sensor alternatives (e.g. vision) because the radar is intended to work as a complementary instrument or as a replacement sensor in scenarios where the alternative sensors are prone to fail.

1.2 Summary of prior work

In order to establish a clear context for the methods described in the present paper, we present below a summary of the state of the art in terms of radar-based approaches for people detection and tracking together with an overview of related applications for indoor environments.

According to the main data types and the signal processing techniques applied, the approaches proposed in the literature to detect and track moving persons can be divided into the following categories:

Doppler-based detection and range/Doppler/azimuth based tracking. This approach separates moving people from clutter based essentially on Doppler data (representative of the velocity of the target relatively to the radar); static objects can be easily classified as clutter due to their zero-Doppler attribute while non-zero Doppler measurements associated to people motion are used to initiate and propagate target tracks in time; data association performed for each target relies on additional information regarding the target position in 2D, including its range and bearing provided by a phased array radar. This approach has demonstrated its efficacy to track multiple people with a single, low-cost multiple-input multiple-output (MIMO) radar but it fails in two relevant aspects: i) it cannot distinguish moving people from other moving targets of approximately the same dimensions (or radar cross section) and ii) it cannot detect or track a non-walking person even if he/she is performing some task (seated or standing). This approach is illustrated by the demonstrations provided by the Texas Instruments manufacturer of the xWR1642 and xWR1443 families of frequency modulated continuous wave (FMCW) radars; see, e.g., [8] and [6]. It is also adopted in the works of [9] and [5].

Range/Doppler-based detection and range/Doppler/azimuth based tracking. This approach exploits the extended velocity profile (or velocity dispersion) characteristic of human motion that stems from the fact that humans do not move as rigid objects but rather present typical oscillating movements of their members and body; for example, there is a typical spreading and contraction pattern of the velocity profile, with a sinusoidal character, in the case of movement of arms and legs of a walking pedestrian. This approach is obviously more robust and holds a larger application potential in most scenarios where humans are required to interact with other mobile agents (including mobile robots, small vehicles, and other machines with moving parts in general). Typically, during a short time interval (for example, a radar frame of a few tens of milliseconds), walking pedestrians present Doppler profiles with multiple, distinct velocity components centered on a single point in the range domain (due to his small velocity, a walking human practically does not change its position during such a short interval). This is in contrast with the compact, point-shaped profile presented by a rigid moving body (a mobile robot or a car) in the Doppler domain due to its constant velocity during an equivalent period of time. An interesting implementation of the Range-Doppler detection approach is described in [4]. The authors apply a Support Vector Machine to classify pedestrians, vehicles, and other objects in an urban area, using an automotive 24 GHz radar sensor with a bandwidth of 150 MHz as the measuring device. For such, they propose three feature extraction schemes with increasing computational complexity and latency corresponding to increasing classification success rates. The features used in the simplest implementation are the range-velocity profiles and associated dispersion metrics extracted from the set of reflections obtained in a single radar measurement; a more elaborate solution exploits data from multiple buffered radar measurements; the most advanced implementation complements the radar returns with additional feedback information provided by a Kalman-based tracker. In the three solutions, the true positive rates for pedestrians are, 45%, 54%, and 61%, respectively. Another potential application of the Range-Doppler detection approach consists of exploiting the patterns of hands movements of a person in order to detect her presence and even to interpret his gestures (for example as remote commands for a robot). This method is illustrated by the work of [2] that implements a hand gesture recognition system using a machine learning algorithm fed with data from a radar to permit steering small mobile robots without direct human operation. The machine learning implementation of gesture recognition relies on decision trees providing a mean success rate (true positives) of 96%.

Returned signal strength-based detection. This approach exploits measurements of signal intensity returned from the target that may be used directly, as in the case of classification schemes based on the computed absolute radar cross section (RCS) of targets, or indirectly by exploiting the relative intensity of the echos corresponding to distinct scatterers of the same target. RCS is a property of the target's reflectivity, commonly used to detect and classify airplanes in a wide variation of ranges. In principle, the technique can be applied to

distinguish terrestrial targets, because cars, motorcycles, bicycles, and pedestrians have distinct RCS values. In practice, the method may be difficult to apply due to the large dependence of RCS measurements on the orientation of the target relatively to the radar; see, e.g., [7]. To the best of our knowledge RCS has not been applied before to detection and tracking of people. The indirect method is illustrated by the collection of methods proposed in [1] that exploits the 2D radar intensity image of a small object in order to derive a model of its bi-dimensional shape. This is combined with measures of its Doppler spectrum distribution to build a feature vector used by a fuzzy membership function that maps the acquired features onto the corresponding target class. As such, the solution integrates techniques pertaining to the above-mentioned approaches 1) and 2) in order to implement a robust classification system. The success rate (true positives) achieved with this system varies between 40% and 95% depending on the orientation of the pedestrian trajectory (from radial to lateral) relatively to the radar axis.

1.3 Novelty and main contributions of the paper

In this paper we propose a new, robust approach for classification of moving objects based on radar measurements in indoor scenarios. The main novelty of the method is the exploitation of the velocity pattern associated to the collection of points (point cloud) representative of the multiple radar scatterers of a given object that permits a robust distinction between moving rigid bodies and people. Another relevant and novel contribution consists in the utilization of the radar cross section of the observed objects as a classification feature, which is shown to significantly increase the performance of the classifier.

2 Problem formulation and proposed solution

2.1 Basic terminology

- In radar terminology, any object that can be detected by the radar is often designate as *target*.
- The scalar measure representative of the radial velocity of the target, i.e. its velocity projected onto the central axis of the radar transmitter, is normally designated as *Doppler* since it is computed based on the frequency shift (Doopler effect) incurred by the radar wave due to the target velocity.
- A *frame* constitutes an elementary structure of reflectivity data that characterizes the state of the target in terms of its distance (range) and radial velocity (Doppler), relatively to the radar system.
- The MIMO radar technology considered in this work permits the discrimination of simultaneous reflections from different points in the plane of the radar wave by exploiting the beam-formimg capability of the MIMO system.
- In the present context, a *point cloud* is a collection of points corresponding to individual reflections from a scene acquired at a given instant of time and represented in different positions of the radar sensor grid; these points may correspond to different reflections from a single or multiple targets.

2.2 Problem formulation

The problem can be formulated as follows: *Given an indoor scenario characterized by the presence of a diversity of static and mobile objects, implement a method to unambiguously detect the presence of a walking person and estimate its kinematic properties using range and Doppler data acquired with a low-cost radar.*

2.3 Proposed solution

The proposed approach relies on the following principles.

The motion of people is characterized by the pendular motion of arms, as well as legs. Other objects, such as robots, usually move as a block. The pattern of motion of people is, therefore, different from the pattern of motion of other mobile agents, and thus, capturing this pattern over time and feeding it to a learning model can provide a mean to dynamically detect people and distinguish them from other objects.

Using a radar to capture the pattern of motion of an object actually implies measuring the velocity of the points that constitute the point cloud of that object and its variation along time. The velocity of a point is given by its associated Doppler value. When some object moves in front of the radar, a series of frames are recorded, and the values of Doppler are stored. In this way, the evolution of the kinematic properties of a target can be stored and indexed to time.

In addition to the utilization of the velocity pattern of the objects observed in a given scenario, we propose to acquire their RCS and use it in order to make the classification more robust.

The RADAR device used was the AWR1642-BOOST EVM from Texas Instruments. In table 1 details of the chirp can be found.

In this work, the RCS, represented by the Greek letter σ, was calculated using the following equation

$$\sigma = \frac{4\pi^3 d^4 P_r}{\lambda G P_t} \ , \tag{1}$$

Chirp Parameter (Units)	**Value**	**Chirp Parameter (Units)**	**Value**
Start Frequency (GHz)	77	Maximum unambiguous range (m)	5
Slope (MHz/us)	60	Maximum radial velocity (m/s)	5.2936
Samples per chirp	128	Azimuth resolution (degrees)	14.5
Chirps per frame	256	Velocity resolution (m/s)	0.0827
Frame duration (ms)	50	Number of transmission antennas	2
Sampling rate (Msps)	2.5000	Range resolution (m)	0.0488
Bandwidth (GHz)	3.0720	Number of reception antennas	4

Table 1. Chirp parameters and respective value.

where d denotes the distance to the object, P_t and P_r represent the power emitted by the radar and reflected by the target, respectively, λ is the wave length of the radar signal, and G is the ratio between the transmission and reception gain .

The value for P_t is 12.5 dBm, and is later converted to Watts. The transmission and reception gains have the same value of 9 dB, thus $G = 1$. Because the radar chirp used has a bandwidth of 4 GHz (from 77 GHz to 81 GHz), in order to calculate RCS, the wave length used is based on the central frequency, 79 GHz, which corresponds to a wave length of 0.0038 meters. Since the received signal is attenuated by a series of filtration steps in the digital signal processor (DSP) pipeline of the radar, the power of the digitized signal after fast Fourier transform (FFT) processing needs to be compensated in order to obtain the total power reflected by the target. This compensation is implemented according to equation (2) as explained below; see also [7].

$$P_r = P_{FFT} - [20log_{10}(2^{(nbits-1)}) + 20log_{10}(\sum_{i=0}^{N-1} \omega_i) - 20log_{10}(\sqrt{2})] + 10 \quad (2)$$

Where P_{FFT} stands for the power of the signal after analog-to-digital converter (ADC) and FFT processing, *nbits* is the number of ADC bits used (12 in this case), the ω_i summation represented compensates the attenuation effect of windowing prior to the application of the FFT, and the additive term 10 is used to convert from dBFS (dB full-scale) to dBm. Equation 2 is used as a correction factor [7]. The resulting value for power received is later converted from dBm to Watts.

The velocity of the points belonging to the point cloud of a frame are represented by a histogram. The histogram is calculated between -6 and 6 m/s with 150 bins. Therefore the bin size (resolution) is 0,08 m/s . The kinematic of the object can be viewed as an image by the concatenation of the histograms of the object point cloud in all frames. Naturally the values of the bins are color coded.

Based on this configuration, the data acquired in each test is transformed into a single dataset constituted by 151 elements (150 velocity bins plus one value RCS) and with a number of data samples equal to the number of frames. The total dataset is constituted by the collection of all the datasets obtained in all the tests. Hence, in the current experiment, the complete dataset consists of a total of 24455 frames. Out of these 24455 frames, 11285 correspond to people moving, and the remaining 13170 correspond to other moving objects.

When performing measurements using the RADAR, clutter may coexist with the point cloud of an object of interest. It is then important not only to detect this clutter, but to eliminate it, ensuring that these points will not be included in the creation of the dataset. The obvious way to segment the point cloud corresponding to a target is by using an appropriate clustering method. The method chosen in the current implementation was DBSCAN (Density-based spatial clustering of applications with noise); [3]. DBSCAN is a clustering method that relies solely on two base parameters: minimum number of points and minimum distance between them for a set of points to be considered a cluster. Henceforth,

minimum number of points will be designated as *MinPts* and the minimum distance will be designated by *Eps*. In the current application, clustering is performed not only in spacial dimension, but also in the velocity dimension. The justification of this strategy is the fact that two closely spaced objects will not be distinguished by the algorithm if we only consider spatial distance between points. However, if we consider clustering in the velocity dimension, then two objects, even if closely spaced, will be distinguished if they have different velocities. This clustering approach is applied in a hierarchical fashion. First, only spacial-based clustering is applied, resulting in an initial set of clusters. Then, to every cluster, velocity based sub-clustering is applied inside each cluster, permitting to discriminate sub-clusters of points whose velocity difference exceeds a predefined threshold. The parameter values of $MinPts$=4 and Eps=0.6, both in the spatial and velocity dimension, were chosen based on practical experiments.

It is important to remark that with the present configuration of clustering parameters, the method does not decompose the cloud of points of a single person into different clusters; it allows for the coexistence of significant velocity differences in the same cluster in order to contemplate the representation of the pendular motions of the arms and legs of the same person.

3 Experiments

3.1 Scenario configuration

The tests with the radar were performed in an indoor wide open area, free of obstacles. This simplified scenario is proposed, similarly to other works described in the literature, in order to permit a proof of concept involving the type of classification methods proposed here. The host computer communicates with the sensor via a Raspberry Pi connected directly to the sensor. The device is placed at a height of 1.9 meters (in order to stand taller than most people) with an inclination (facing downwards) of about 10 degrees. This placement of the device aims at minimizing the number of people that can be occluded by other people present in the area.

The whole test field contains blue marks that can be found on the ground. They identify the bottom and top left and right corners, as well as the centre of the field and the centre of each one of the sides. There are also blue marks to identify the 30 and 60 degree mark on the bottom and middle of the field, and the 30 degree mark at the top of the field (which is coincident with the top left and right corner). The blue marks at the bottom of the two front legs of the tripod serve the purpose of marking the position of the sensor, to ensure the tests are done with the device placed always in the same position. Figure 1 shows a schematic of the test field.

3.2 Types of tests carried out

To enable a learning model to distinguish people from other objects, representative data sets of both classes of targets had to be provided. Therefore, tests were

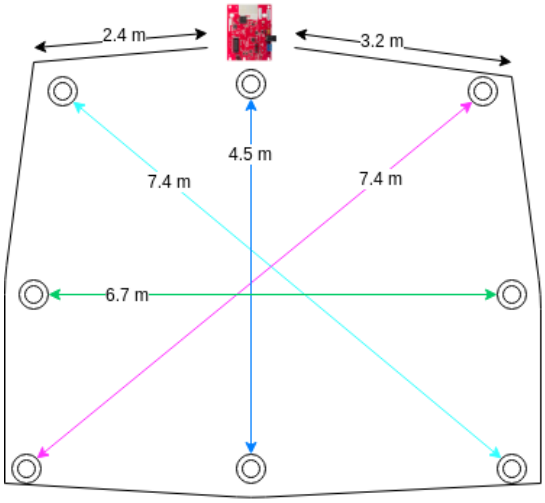


Fig. 1. Schematic of the test field with the trajectories considered represented by bidirectional non-black arrows, and the corners by circumferences. The length of each trajectory is represented in meters.

made with people, but also with objects, such as dummies with a dimension close to that of an adult. Furthermore, with the intent of making this application as inclusive as possible, people with limited mobility were also taken into account by performing tests with a person driving an automated wheelchair.

Each elementary test in this experiment consisted of an object moving in a specific direction. In the test field, 4 types of directions were considered: from one corner to the opposite one (left to right and right to left); from the top centre of the field to the bottom centre; from the centre side of the field to the other centre side. For each direction, the two ways were considered, yielding therefore a total of 8 different trajectories (see Fig. 1). For each trajectory, 10 different tests were performed, for a total of 80 tests for each specific object. Three different types of objects were considered: person walking, person in a wheelchair, and two different dummies. A picture of one of the dummies is shown in 2, as well as the radar device.

The typical velocity pattern of a person walking in front and to the device is depicted in figure 3; the plot clearly shows the dispersion of velocity values associated to the walking person. For other directions, such as diagonal or transversely, the pattern is similar. Figure 4 shows the velocity pattern of the mannequin and of a person in a wheelchair. As can be seen, the velocity pattern of the locomotion of the dummy is clearly different from that of a person. Most of the points of a person-related point cloud are located within a defined region of the Doppler, and this can be explained by the fact that most of the points are correspondent to the torso and thus move with the same velocity. However, as a result of the motion of the arms and legs, we can see there is a spread of

Fig. 2. One of the dummies used in the tests.

points along the Doppler dimension. This spread is symmetrical (both in the positive and negative side) and approximately periodic. This is accordance with the movement of the limbs, which have a velocity superior to the rest of the body and travel in opposite directions. On the other hand, the motion pattern of the dummy is rather different. As the dummy moves as a block, the great majority of the points will have the same Doppler value.

4 Learning Models Applied

The learning models used in this paper are implemented in the well-known Python machine learning library named **scikit-learn**. Each model is dependent on a set of parameters. The parameters for which the respective value is omitted are considered to have the default value defined in the aforementioned library. 25% of the dataset (equivalent to 6114 frames) were used for test, while the remaining were applied in the training stage. The dataset was previously shuffled to ensure an equal representation of both classes in both the training and testing set.

4.1 Artificial Neural Network

The first model applied was an artificial neural network. Several values for the hyperparameters were tested. The optimal performance was 85.9%. The values

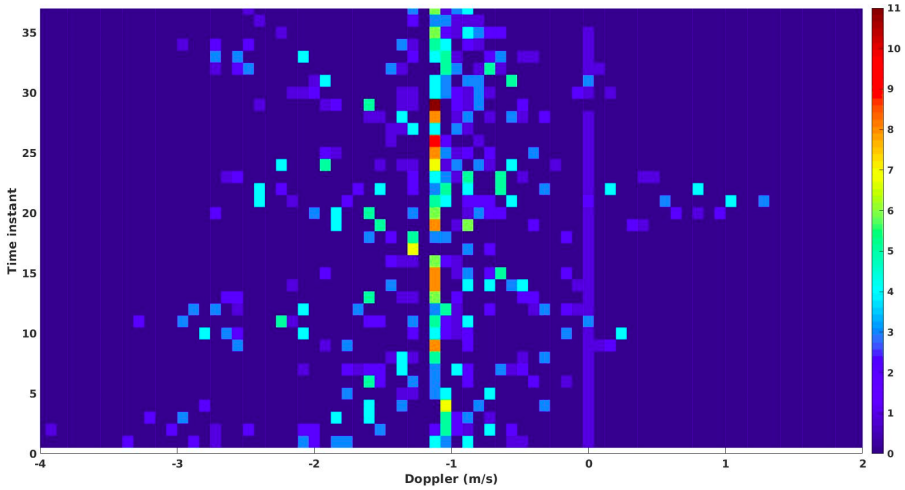


Fig. 3. Velocity pattern of a person walking in front and to the device.

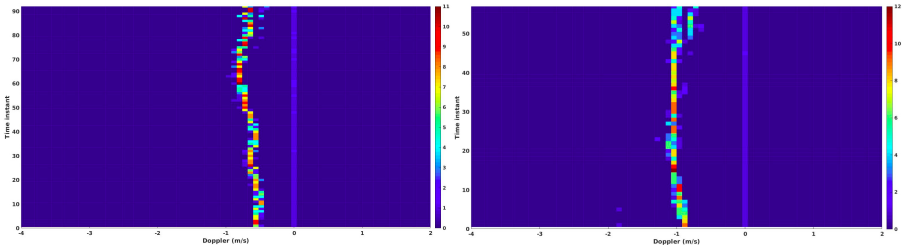


Fig. 4. Velocity pattern of a mannequin (left) and of a person in a wheelchair (right) moving in front and to the device. The Doppler scale is the same as the one in Fig. 3.

used for the artificial neural network parameters are shown in table 2. Using the same hyperparameters but ignoring the RCS as a feature, the accuracy is 81.5%. In table 3 is shown the confusion matrix both in the case the RCS is used and in the case the RCS is not used.

4.2 Random Forests

The second model applied was a Random Forest, with a number of estimators of 500 and a maximum depth of 600, and the random state is set to 0. The accuracy attained was 86%, which is better than the one achieved by the neural network. As in the case for the neural network, when the RCS is not used the accuracy decreases, in this case for a value of 82.9%. Table 4 shows the confusion matrix of the Random Forest model.

Activation function	Relu	**Num. of hl**	4
Num. of neurons (1st to last layer)	100, 70, 50, 20	**Learning rate**	0.001
Maximum iterations	10000	**Random state**	42

Table 2. Artificial Neural Network Hyperparameters. *hl* stands for hidden layers.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	2890	413
Person	448	2363
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	2732	571
Person	559	2252

Table 3. Artificial Neural Network confusion matrix.

5 Discussion of results and conclusions

As can be seen from the results presented in the previous section, the inclusion of the RCS of each detected target as a classification feature was shown to improve significantly the performance of the target recognition methods in the two studied approaches. Actually, even without using an absolute values of the RCS, whose calibration was beyond the scope of the present study, it is possible to show that the relative values of this parameter, when acquired by the same radar with fixed parameters, is sufficient to discriminate some classes of objects in a given scenario. In our opinions, this constitutes an interesting contribution of the present work.

The overall performance of the classification models tested in this study evidence clearly the high potential of the methods to detect people in indoor environments, even if moving in a wheelchair, and to support the application envisaged in our work.

The work currently in progress at our laboratory includes additional tests with other types of targets and the detection of non-walking people based on movements of the arms and hand gestures.

In future work we intend to test the same methods in more problematic scenarios (e.g. with furniture and other obstacles) in order to assess their performance and the potential adequacy of different models to the different environ-

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	2916	387
Person	470	2341
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	2821	482
Person	563	2248

Table 4. Random Forest confusion matrix.

ments. Moreover, different parameters of the classification models will be also tested, and their results evaluated.

References

1. Bartsch, A., Fitzek, F., Rasshofer, R.H.: Pedestrian recognition using automotive radar sensors. Advances in Radio Science **10**, 45–55 (2012)
2. Berkius, C., Buck, M., Gustafsson, J., Kauppinen, M.: Human Control of Mobile Robots Using Hand Gestures. Bachelor thesis in electrical engineering, Chalmers University of Technology. Gothenburg, Sweden. (2018)
3. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: 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. pp. 226–231. KDD'96, AAAI Press (1996)
4. Heuel, S., Rohling, H.: Pedestrian Recognition Based on 24 GHz Radar Sensors, vol. Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications, chap. 10, pp. 241–256. InTech (2013)
5. Knudde, N., Vandersmissen, B., Parashar, K., Couckuyt, I., Jalalvand, A., Bourdoux, A., Neve, W.D., Dhaene, T.: Indoor Tracking of Multiple Persons With a 77 GHz MIMO FMCW Radar. In: 2017 European Radar Conference (EURAD). pp. 61–64 (2017)
6. Livshitz, M.: Tracking radar targets with multiple reflection points. `https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/1023/Tracking-radar-targets-with-multiple-reflection-points.pdf` (2018), [Online; accessed 20-April-2019]
7. Machado, S., Mancheno, S.: Automotive FMCW Radar Development and Verification Methods. Master's thesis, Department of Computer Science and Engineering. Chalmers University of Technology. University of Gothenburg, Sweden. (2018)
8. Texas-Instruments: People tracking and counting reference design using mmwave radar sensor. ti designs: Tidep-01000. (Mar 2018)
9. Yamada, H., Wakamatsu, Y., Sato, K., Yamaguchi, Y.: Indoor Human Detection by Using Quasi-MIMO Doppler Radar. In: 2015 International Workshop on Antenna Technology (iWAT). pp. 35–38 (2015)

Appendix B

Paper accepted and presented at conference ROBOT2019

Machine learning methods for radar-based people detection and tracking by mobile robots

José Castanheira[1], Francisco Curado[1][0000−0003−0239−4675], Eurico Pedrosa[1][0000−0002−7954−9922], Edgar Gonçalves[1], and Ana Tomé[1][0000−0002−1210−0266]

Department of Electronics, Telecommunications and Informatics Engineering, University of Aveiro, 3810-193 Aveiro, Portugal.
{jp.castanheira,fcurado,efp,easg,ana}@ua.pt

Abstract. This paper reports a machine learning approach for people detection and tracking in indoor environments using a compact radar system deployed by a mobile robot. The set-up described in the paper includes a series of experiments carried out in an indoor scenario involving walking people and dummies representative of other moving objects. In these experiments, distinct learning models (a neural network and a random forest) were explored with different combinations of radar features to achieve person versus non-person classification.

Keywords: RADAR · Mobile Robotics · Machine Learning · People Detection

1 Introduction

1.1 Motivation

The utilization of standard sensors such as cameras, laser or infrared in order to detect the presence of people and other objects in a given scenario of interest is often hampered by the diversity of environmental effects such as dust, fog, rain, strong heat or fire. In contrast, radar signals are immune to these disturbances. Additionaly, radar systems can be easily parameterized to deal with a diversity of operational conditions. In recent years, the number and diversity of applications of radar sensors have been increasing dramatically motivated by the emergence of inexpensive, self-contained development kits. These devices permit rapid prototyping and testing of radar-based solutions and their reduced cost and dimension make them affordable for academic research and permit their integration in small, inexpensive robots.

Despite the aforementioned advantages, robust feature extraction from radar data is still an open problem because the radar physics is prone to generate a diversity of artifacts that pose significant challenges to conventional signal processing techniques. This problem is especially dramatic in indoor environments due to the ubiquity of obstacles (clutter) normally present in these scenarios.

The present paper introduces a novel machine learning approach integrated with mobile robotic platforms to address the problem of radar-based detection of moving people in an indoor scenario. To the best of our knowledge, none of the alternative works address the specific problem that we tackle in this paper: the detection of people (walking and on wheelchair) in indoor environments based on statistical measures of radar Doppler data acquired by mobile robots. The envisioned applications include elderly monitoring and rescue operations in vision-denied environments. Other related works have focused on the use of micro-Doppler data to detect human motions. Micro-Doppler (effect) is defined as frequency modulation on the returned radar signal, that is a result of the target's micro-motion dynamics, such as mechanical vibrations. Despite the high potential of the approach, exploitation of this effect with the radar kits on hand is not suitable for the current application given the demanding signal processing power required to resolve the micro-Doppler components induced by the targets while simultaneously dealing with the envisaged sensing requirements: 1) maximum detection range, and maximum unambiguous velocity of different moving objects, and 2) the inherent vibrations of mobile robots (carrying the radar sensors) that induce micro-Doppler effects susceptible of masking those of the targets. The paper does not discuss other sensor alternatives (e.g. vision) because the radar is intended to work as a complementary instrument or as a replacement sensor in scenarios where the alternative sensors are prone to fail.

1.2 Summary of prior work

According to the main data types and the signal processing techniques applied, the approaches proposed in the literature to detect and track moving persons can be divided into the following categories:

Doppler-based detection and range/Doppler/azimuth based tracking. This approach is applied in demonstrations and technical documents provided by the Texas Instruments manufacturer of the xWR1642 and xWR1443 families of FMCW radars; see, e.g., [11] and [8]. It is also adopted in the works of [12] and [7]. These methods distinguish moving people from clutter based essentially on Doppler data (representative of the velocity of the target relatively to the radar); static objects can be easily classified as clutter due to their zero-Doppler attribute while non-zero Doppler measurements associated to people motion are used to initiate and propagate target tracks in time; data association performed for each target relies on additional information regarding the target position in 2D, including its range and bearing provided by a phased array radar. The approach has been applied successfully to track multiple people with a single, low-cost MIMO radar but it fails in two relevant aspects: i) it cannot distinguish moving people from other moving targets of approximately the same dimensions (or radar cross section) and ii) it cannot detect or track a non-walking person even if he/she is performing some task (seated or standing).

Range/Doppler-based detection and range/Doppler/azimuth based tracking. This approach exploits the extended velocity profile (or velocity dispersion) characteristic of human motion that stems from the fact that humans do not move as rigid objects but rather present typical oscillating movements of their members and body; for example, there is a typical spreading and contraction pattern of the velocity profile, with a sinusoidal character, in the case of movement of arms and legs of a walking pedestrian. This approach is obviously more robust and holds a larger potential of application in most scenarios where humans are required to interact with other mobile agents (including mobile robots, small vehicles, and other machines with moving parts in general). An interesting implementation of Range/Doppler-based detection is described in [6]. The authors apply a Support Vector Machine to classify pedestrians, vehicles, and other objects in an urban area, using an automotive 24 GHz radar sensor with a bandwidth of 150 MHz as the measuring device. Another potential application of the Range-Doppler detection approach consists of exploiting the patterns of hands movements of a person in order to detect her presence and even to interpret his gestures (for example as remote commands for a robot) [2].

Returned signal strength-based detection. This approach exploits measurements of signal intensity returned from the target that may be used directly, as in the case of classification schemes based on the computed absolute radar cross section (RCS) of targets, or indirectly by exploiting the relative intensity of the echos corresponding to distinct scatterers of the same target. RCS is a property of the target's reflectivity, commonly used to detect and classify airplanes in a wide variation of ranges. In principle, the technique can be applied to distinguish terrestrial targets, because cars, motorcycles, bicycles, and pedestrians have distinct RCS values. In practice, the method may be difficult to apply due to the large dependence of RCS measurements on the orientation of the target relatively to the radar; see, e.g., [9]. To the best of our knowledge RCS has not been applied before to detection and tracking of people.

A comprehensive revision of the use of radar systems together with the application of machine learning for people detection tasks can be found in [5]. The operation of radars installed on mobile robots, particularly in indoor environments, is still poorly covered in the literature. Some notable exceptions are [10], [1], [2], and [3]. However, none of those works tackles the problem of people detection and tracking by moving robots that is addressed in this work.

1.3 Main contributions of the paper

The main contribution of this paper is the introduction of a new, robust approach for classification of moving objects based on radar measurements acquired by mobile robots in indoor scenarios. In terms of formulation and implementation, the main novelties of the work are: i) the exploitation of the Doppler histograms representative of the velocity distribution of the points that constitute the point cloud acquired by the radar, originated from the multiple radar scatterers of a

given object as a feature that permits a robust distinction between moving rigid bodies and people, and ii) the utilization of the radar cross section of targets as a classification feature; we introduce here a practical method for its computation and propose its application in order to implement more robust classifiers and trackers of people.

2 Terminology, problem formulation and solution

2.1 Basic terminology and problem formulation

Basic terminology. In radar terminology, any object that can be detected by the radar is often designate as *target*. The scalar measure representative of the radial velocity of the target, i.e. its velocity projected onto the central axis of the radar transmitter, is designated as *Doppler* since it is computed based on the frequency shift (Doppler effect) incurred by the radar wave due to the target velocity. A *frame* is a fixed-size packet of reflectivity data (chirp reflections) that characterizes the state of the target in terms of its distance (range) and radial velocity (Doppler) relatively to the radar. The multiple-input multiple-output (MIMO) radar technology considered in this work permits the discrimination of simultaneous reflections from different points in the plane of the radar wave by exploiting the beam-formimg capability of the MIMO system. In the present context, a *point cloud* is a collection of points corresponding to individual reflections from a scene acquired at a given instant of time and represented in different positions of the radar sensor grid; these points may correspond to different reflections from a single object or from multiple targets.

Problem formulation. The problem addressed here can be formulate as follows: *Given an indoor scenario characterized by the presence of a diversity of static and mobile objects, implement a method to unambiguously detect the presence of a walking person and estimate its kinematic properties using range and Doppler data acquired with a low-cost radar installed on a mobile robot.*

2.2 Proposed solution

Among other characteristics, the motion of people is characterized by the pendular movements of arms and legs. Other objects, such as robots, usually move as a block. The pattern of motion of people is, therefore, different from the pattern of motion of other mobile agents, and thus, capturing this pattern over time and feeding it to a learning model can provide a mean to dynamically detect people and distinguish them from other objects.

Another issue is that the RCS of a person (even when seated) can be distinguished from that of other objects. Hence, the RCS may be used as a feature in order to make the classification more robust.

Radar system. The RADAR device used was the AWR1642-BOOST EVM from Texas Instruments. Details of the radar signal can be found in table 1.

Radar data processing. Although the discussion of this topic is beyond the scope of this work, the RCS, represented by the Greek letter σ, can be computed as

$$\sigma = \frac{4\pi^3 d^4 P_r}{\lambda G P_t} \ , \tag{1}$$

where d denotes the distance to the object, P_t and P_r represent the power emitted by the radar and reflected by the target, respectively, λ is the wave length of the radar signal, and G is the ratio between the transmission and reception gain.

The velocity of the points belonging to the point cloud of a frame are represented by a histogram. The histogram is calculated between -6 and 6 m/s with 150 bins. Therefore the bin size (resolution) is 0,08 m/s . Based on this, the target kinematics is represented as an image by the concatenation of the Doppler histograms of the successive point cloud of all frames. For a better visualization, the values of the bins are color coded; see e.g. Figs. 3 and 4.

Based on this configuration, the data acquired in each test is transformed into a single dataset constituted by 151 elements (150 velocity bins plus one value RCS) and with a number of data samples equal to the number of frames.

Data Clustering. When performing measurements using the RADAR, clutter may coexist with the point cloud of an object of interest. It is then important not only to detect this clutter, but to eliminate it, ensuring that these points will not be included in the creation of the dataset. The obvious way to segment the point cloud corresponding to a target is by using an appropriate clustering method. The method chosen in the current implementation was DBSCAN (Density-based spatial clustering of applications with noise), [4]. DBSCAN is a clustering method that relies solely on two base parameters: minimum number of points and minimum distance between them for a set of points to be considered a cluster. Henceforth, minimum number of points will be designated as *MinPts* and the minimum distance will be designated by *Eps*. The parameter values of *MinPts*=2 and *Eps*=0.6 were chosen based on practical experiments. Although several clusters can be detected with this method, in the the tests performed, we only consider one object in front of the device at a time. Therefore in all our

Chirp Parameter (Units)	**Value**	**Chirp Parameter (Units)**	**Value**
Start Frequency (GHz)	77	Maximum unambiguous range (m)	5
Slope (MHz/us)	60	Maximum radial velocity (m/s)	5.2936
Samples per chirp	128	Azimuth resolution (degrees)	14.5
Chirps per frame	256	Velocity resolution (m/s)	0.0827
Frame duration (ms)	50	Number of transmission antennas	2
Sampling rate (Msps)	2.5000	Range resolution (m)	0.0488
Bandwidth (GHz)	3.0720	Number of reception antennas	4

Table 1. Chirp parameters and respective values.

Activation function	Relu	**Num. of hl**	4
Num. of neurons (1st to last layer)	100, 70, 50, 20	**Learning rate**	0.001
Maximum iterations	10000	**Random state**	42

Table 2. ANN hyperparameters. *hl* stands for hidden layers.

experimental set-ups for each data frame, the largest cluster is considered the cluster of interest.

2.3 Description of learning models applied

The learning models used in this paper are implemented in the well-known Python machine learning library named **scikit-learn**. Each model is dependent on a set of parameters. The parameters for which the respective value is omitted are considered to use the default values defined in the aforementioned library. The models applied were an Artificial Neural Network (ANN) and a Random Forest. The hyperparameters for the ANN are defined in table 2. The Random Forest was designed as having a number of estimators of 500 and a maximum depth of 600, and the random state is set to 0.

The ANN was chosen due to its traditional applicability to image processing and classification, and the good results it has achieved in this field. Because the statistical Doppler patterns are represented as 2D images, ANN is an adequate model to apply using this type of data. The Random Forest was chosen because it is a model different from the ANN, and allows us to investigate how this model works in this type of classification task.

3 Experiments

The reported experiments consist of a series of tests performed with a radar installed on a small robot, the TurtleBot. In order to address the problem on hand with varying degrees of complexity, two main configurations have been contemplated: one where the radar is installed on a static robot, and another with the radar deployed by a moving robot. In both cases, the moving targets (people and objects) are made to travel along the same set of pre-defined paths shown in Fig. 1.

3.1 Scenario configuration

The tests executed for radar data acquisition were performed in an indoor wide open area, free of obstacles. This simplified scenario is proposed, similarly to other works described in the literature, in order to permit a proof of concept involving the type of classification methods proposed here. The host computer responsible for recording and storing the measurements communicates with the sensor via a Raspberry Pi connected directly to the sensor. The radar device is placed on top of the TurtleBot.

A set of blue marks was placed on the ground in order to serve as a visual guide for the targets and to locate the corners of the field of view (see Fig. 1).

3.2 Types of tests carried out

Tests with radar installed on static robot. In order to enable a learning model to distinguish people from other objects, representative data sets of both classes of targets had to be provided. Therefore, tests were made with people but also with a mannequin with a dimension close to that of an adult. Furthermore, with the intent of making this application as inclusive as possible, people with limitated mobility were also taken into account by performing tests with a person driving an automated wheelchair.

Each elementary test in this experiment consisted of an object moving in a specific direction within the field of view of the radar. In the test field, four types of directions were considered: from one corner to the opposite one (left to right and right to left); from the top centre of the field to the bottom centre; from the centre side of the field to the other centre side. For each direction, the two ways were considered, yielding therefore a total of 8 different trajectories, as shown in Fig. 1. For each trajectory, 10 different tests were performed, comprising a total of 80 tests for each specific object. Three different types of objects were considered: person walking, person in a wheelchair, and a mannequin. A picture of the mannequin is shown in Fig. 2, as well as the wheelchair used, and the radar positioned on the TurtleBot.

The typical pattern of velocity distribution acquired by the radar when a person is walking in front and towards the device (trajectory C) is depicted in Fig. 3; the plot clearly shows the dispersion of velocity values associated to

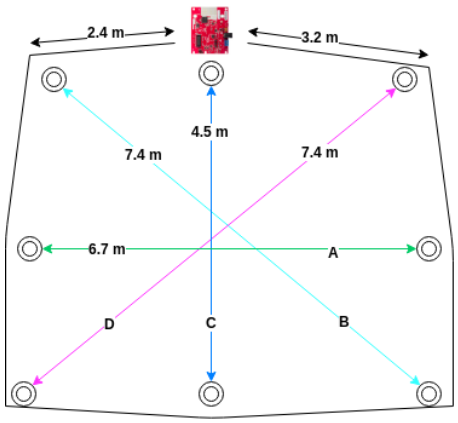


Fig. 1. Schematic of the test field with the trajectories considered represented by bidirectional non-black arrows, and the limits of the field of view represented by circumferences. The length of each trajectory is represented in meters, and each one has also a letter identifier.

the walking person. For other directions, such as diagonal or transversely to the orientation of the radar, the pattern is similar. Fig. 4 shows the velocity pattern of the mannequin and of a person in a wheelchair, moving in front and towards the radar. As can be seen, the velocity pattern of the locomotion of the mannequin is clearly different from that of a person. Most of the points of a person-related point cloud are located within a defined region of the Doppler domain, and this can be explained by the fact that most of the points represent reflections from the torso and thus move with the same velocity. However, as a result of the motion of the arms and legs, there is a clear spread of points along the Doppler dimension. This spread is symmetrical (both in the positive and negative side) and approximately periodic. This is in accordance with the movement of the limbs, which have a velocity superior to the rest of the body and travel in opposite directions. On the other hand, the motion pattern of the mannequin is rather different. As the mannequin moves as a block, the majority of the points have approximately the same Doppler value.

Tests with radar deployed by a moving robot. The tests performed with the radar deployed by the moving robot involved the same targets (people, mannequin, and wheelchair) moving along the same paths as used in the tests executed with the static robot. In the case of the tests performed with the moving robot, the robot travelled along a straight trajectory that crossed or converged with the path of the other targets; see Fig. 1.

4 Results with radar installed on the static robot

The dataset acquired in this experiment consists of a total of 40430 frames. Out of these 40430 frames, 21872 correspond to people moving (walking or moving in a wheelchair), and the remaining 18558 correspond to other moving objects. For classification purposes, 25 % of the dataset was used for test, while the remaining was applied in the training stage. The dataset was previously shuffled to ensure

Fig. 2. Wheelchair (left) and mannequin (center) used in the tests. TurtleBot with radar installed on its top (right), with an upward inclination of 10 degrees.

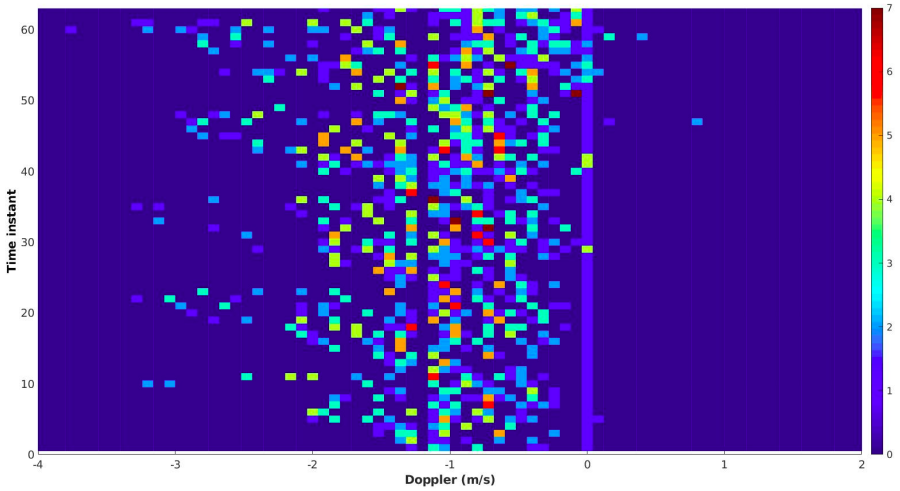


Fig. 3. Histogramic representation of Doppler measures of a person walking in front and to the device (trajectory C), with the TurtleBot immobilized. The Doppler axis only ranges from -4 m/s to 2 m/s because in this case there are no points beyond those limits. The colour bar on the right represents the absolute number of points.

an equal representation of both classes (people moving and other moving objects) in both the training and testing set.

The classification based on the ANN described in subsection 2.3, and using as features the normalized histogram bins as well as the RCS, achieved an accuracy of 96.1%. Following the standard notation of this field, accuracy is defined as $\frac{TP+TN}{N}$, where TP and TN stands for true positives and true negatives (respectively), and N represents the total number of examples. Using the same ANN with similar hyperparameters but excluding the feature RCS, the performance was 92.0%. Using the Random Forest, the accuracy attained was 95.3%. As in the case of the ANN, when the RCS is not used as a feature the accuracy decreases, in this case for a value of 91.9%. The confusion matrices of the ANN and Random Forest are represented in table 3 and 4, respectively. Note that, although we only reference the accuracy, all the other metrics (such as precision, recall, etc) can be derived from the confusion matrices.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	4487	175
Person	217	5229
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	4357	305
Person	506	4940

Table 3. ANN confusion matrices in the case of an immobilized TurtleBot.

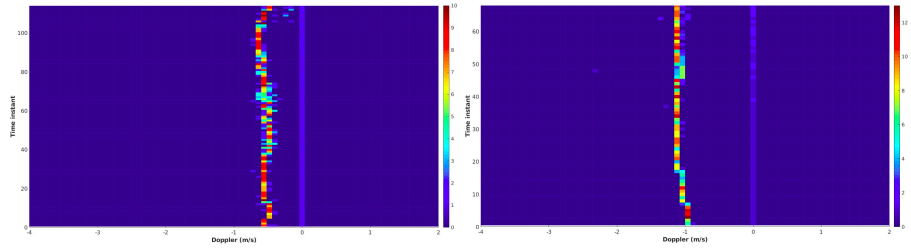


Fig. 4. Histogramic representation of Doppler measures of a mannequin (left) and person in wheelchair (right) moving in front and to the device, with the TurtleBot immobilized. The Doppler scale is the same as the one in Fig. 3. Notice the dispersion of Doppler of the mannequin, which is caused by small oscillatory movements of its pending arms combined with a certain component of rotation of the body during towing of the dummy. In contrast, the person in the automated wheelchair is practically still. The colour bar on the right of each histogram represents the absolute number of points.

5 Results with radar installed on the moving robot

5.1 Constraints

When the radar is installed on a static platform, the objects that contribute with points to the point cloud generated by the radar are only those who have an apparent movement relatively to the sensor. However, in this experiment, as the sensor is moving, all objects exhibit apparent movement relatively to the radar, and thus, all objects contribute with points to the point cloud generated. Hence, in this experiment the point cloud generated is larger than when the radar was immobilized. The first tests revealed that this increase of information (points) is enough to make the radar application provided by the manufacturer to crash. To solve this problem the radar configuration was modified, reducing the number of ADC samples collected per chirp, from 128 to 64.

One of the issues observed in this experiment was that the clustering method wasn't robust enough to detect and retrieve only the cluster of interest. Due to the above mentioned reasons, when the sensor is moving, more clusters are generated than when the robot is immobilized. Thus, choosing the largest cluster as the cluster of interest does not guarantee that the cluster corresponds to the target of interest; it may be generated by any sufficiently large object nearby,

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	4450	212
Person	263	5183
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	4316	346
Person	468	4978

Table 4. Random Forest confusion matrices in the case of an immobilized TurtleBot.

with an apparent motion. To resolve this, an heuristic-based tracking algorithm was developed in order to detect a cluster and to follow it along a sequence of frames, and to detect if the cluster chosen to initialize the track is the right cluster originated with data from the target of interest; see Algorithm 1. This algorithm considers that from frame to frame, the same cluster can not travel more that 1.6 meters, because otherwise, it would have a velocity higher than what is expected from objects moving in the environment under study. On the other hand, the total distance traversed by a cluster along the experiment must be equal or greater than 1.5 meters. Otherwise, it would mean the cluster barely moved, which can not be the case for the cluster of interest.

5.2 Results

In the current experiment, the complete dataset consists of a total of 25431 frames. Out of these 25431 frames, 12720 correspond to people moving, and the remaining 12711 correspond to other moving objects. 25 % of the dataset was used for test, while the remaining was applied in the training stage.

Using the ANN described in subsection 2.3 and considering the RCS as a feature, the accuracy attained was 97.7%, while ignoring the RCS the performance was 96.2%. For the Random Forest, the accuracy were 97.8% using the RCS as a feature, and 96.5% without RCS. The resultant confusion matrices of the ANN and Random Forest are represented in table 5 and 6, respectively.

Algorithm 1: Heuristic tracking algorithm

```
Iteration = 1; TotDistance = 0;
while there are data frames do
    if Iteration == 1 then
        Compute and store centroid of cluster;
    else
        Calculate closest centroid to previous one;
        Store centroid of cluster;
        Displacement = distance between previous and closest centroid;
        if Displacement > 1.6 then
            Iteration = 1; TotDistance = 0;
            Start on different cluster;
            Continue;
        end
        TotDistance = TotDistance + Displacement;
    end
    if Iteration == total number of frames then
        if TotDistance < 1.5 then
            Iteration = 1; TotDistance = 0;
            Start on different cluster;
            Continue;
        end
    end
    Iteration = Iteration + 1;
end
```

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	3135	66
Person	83	3074
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	3076	125
Person	115	3042

Table 5. ANN confusion matrices in the case of a moving robot.

	Predicted Not Person (with RCS)	Predicted Person (with RCS)
Not Person	3133	68
Person	74	3083
	Predicted Not Person (without RCS)	Predicted Person (without RCS)
Not Person	3070	131
Person	91	3066

Table 6. Random Forest confusion matrices in the case of a moving robot.

6 Discussion of results and conclusions

As can be seen from the results presented, the performance of the classification models tested in this study clearly evidence the high potential of the methods to detect people in indoor environments and to support the application envisaged in our work. Although the tests with the moving TurtleBot hampered the tracking and distinction of the cluster of interest, the methods had a better performance in this case than when the robot was immobilized. In all cases, the classification models had their performance deteriorated when the RCS was not used as a feature. In the experiments were the TurtleBot was immobilized, the ANN had a better performance than the Random Forests. In the experiments where the TurtleBot is moving, the Random Forests had a better performance than the ANN. The RCS had a greater impact on performance in the experiments where the TurtleBot is static, comparing to the experiments where it is moving.

We showed that the approach is effective in two distinct scenarios (robot immobilized and robot moving) which can lead to a wide range of applications in robotics. The presence of clutter and its deleterious effect, which can be typical in indoor environments, was also solved with a simple heuristic tracking algorithm.

In future work we intend to test the same methods in more problematic scenarios (e.g. with furniture and other obstacles) in order to assess their performance and the potential adequacy of different models to the different environments. Moreover, different parameters of the classification models will be also tested, and their results evaluated. The configuration of the radar device can also be altered with the purpose of detecting velocities and distances with a higher resolution or with larger ranges. The output rate can also be increased. By modifying the configuration of the radar, we can study how the different parameters affect the quality of the data and the performance of the model, and ultimately determine the best radar parameters. Another interesting appli-

cation is, besides detecting people, to detect other features such as the different trajectories travelled by each object.

Acknowledgement

This work is funded by Research Project RETIOT PT2020-03/SAICT/2015 - Fundação para a Ciência e Tecnologia.

References

1. Barrett, D., Alvarez, A.: mmWave radar sensors in robotics applications. Tech. rep., Texas Instruments (2017)
2. Berkius, C., Buck, M., Gustafsson, J., Kauppinen, M.: Human Control of Mobile Robots Using Hand Gestures. Bachelor thesis in electrical engineering, Chalmers University of Technology. Gothenburg, Sweden. (2018)
3. Dogru, S., Marques, L.: Using Radar for Grid-based Indoor Mapping. In: Proc. 19th IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC'2019. April24-25. Gondomar, Porto. Portugal (2019)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: 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. pp. 226–231. KDD'96, AAAI Press (1996)
5. Gurbuz, S.Z., Amin, M.G.: Radar-based human-motion recognition with deep learning: Promising applications for indoor monitoring. IEEE Signal Processing Magazine **36**(4), 16–28 (July 2019). https://doi.org/10.1109/MSP.2018.2890128
6. Heuel, S., Rohling, H.: Pedestrian Recognition Based on 24 GHz Radar Sensors. vol. Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications, chap. 10, pp. 241–256. InTech (2013)
7. Knudde, N., Vandersmissen, B., Parashar, K., Couckuyt, I., Jalalvand, A., Bourdoux, A., Neve, W.D., Dhaene, T.: Indoor Tracking of Multiple Persons With a 77 GHz MIMO FMCW Radar. In: 2017 European Radar Conference (EURAD). pp. 61–64 (2017)
8. Livshitz, M.: Tracking radar targets with multiple reflection points. `https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/1023/Tracking-radar-targets-with-multiple-reflection-points.pdf` (2018), [Online; accessed 13-June-2019]
9. Machado, S., Mancheno, S.: Automotive FMCW Radar Development and Verification Methods. Master's thesis, Department of Computer Science and Engineering. Chalmers University of Technology. University of Gothenburg, Sweden. (2018)
10. Takeuchi, E., Elfes, A., Roberts, J.: Localization and Place Recognition Using an Ultra-Wide Band (UWB) Radar, Springer Tracts in Advanced Robotics, vol. 105. Springer (2015)
11. Texas-Instruments: People Tracking and Counting Reference Design Using mmWave Radar Sensor. TI Designs: TIDEP-01000. (Mar 2018)
12. Yamada, H., Wakamatsu, Y., Sato, K., Yamaguchi, Y.: Indoor Human Detection by Using Quasi-MIMO Doppler Radar. In: 2015 International Workshop on Antenna Technology (iWAT). pp. 35–38 (2015)

Appendix C

Excerpt of the technical report of the research grant supported by project RETIOT

1 Application and test of Kalman Filters on the data acquired by the sensor

With the acquired data stored according to a well predefined format, it is possible to process it offline. From the mapping of the radial position of a person detected by the sensor to the Cartesian coordinates, it is possible to reconstruct the trajectory travelled. Figure 1 shows the trajectory defined by a person in the data acquisition stage. Notice that the radar is positioned in the origin of the referential, at point (0,0) with the antennas oriented towards the positive direction of Y axis. Besides showing the successive positions of the person, the velocity vector is also represented at each time instant, with a size that is proportional to the intensity.

The Kalman Filter is a mathematical method whose purpose is to use measurements acquired through time, that can potentially contain noise, and produce predictions about the real values of the state variables, which tend to be closer to the real value then the acquired measurements. Using only the data available in the TLV packets of the the detected targets, it was possible to create two Kalman filters to predict the real position of the target. The two types of filters differ in the way the velocity component is used to make the predictions. In the first approach, the velocity was used as one of the measurements to feed the filter in order to perform the position prediction. In the second approach, the velocity was used as an input to the filter.
Figure 2 shows the trajectory predicted by the Kalman Filter, using the velocity as a measurement. Notice the great similarity with the trajectory measured by the sensor in figure 1.
Figure 3 shows the trajectory predicted by the Kalman filter, using the velocity as an input to the system. Notice once again the great similarity with the trajectory measured by the radar. In both types of Kalman Filters implemented, as well as in the data retrieved by the radar, it is possible to see that, on the line defined by the equation $x = 0$, the density of points that mark the position of the target is smaller. The area where the densisity of points is samller increases as a function of the distance to the origin. This reflects the influence of the blind angle in the performance of the filters.

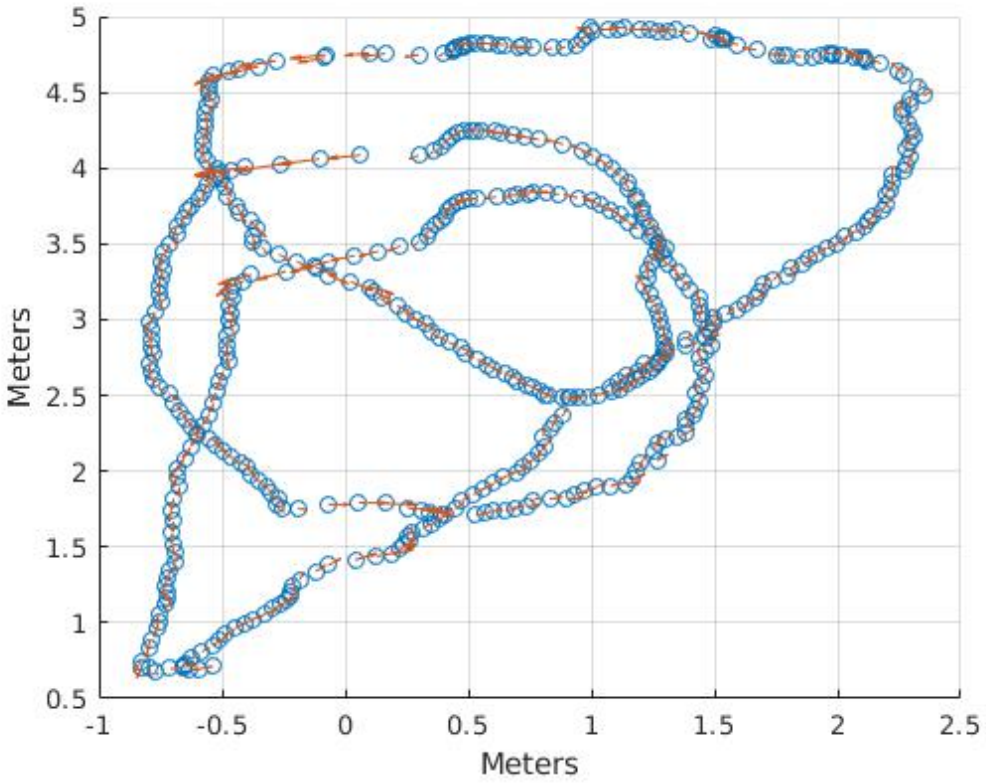


Figura 1: Reconstruction of the travelled trajectory.

The code developed to implement the second version of the Kalman Filter is available at appendix A. In the appendix it is possible to see the matrix that translates the evolution of the system and the error matrix, as well as the method used to update the covariance matrix and the state prediction. The name of each one of the variables used in the filter implementation are the ones which are traditionally used in the literature.

Referências

[1] http://www.ti.com/tool/AWR1642BOOST

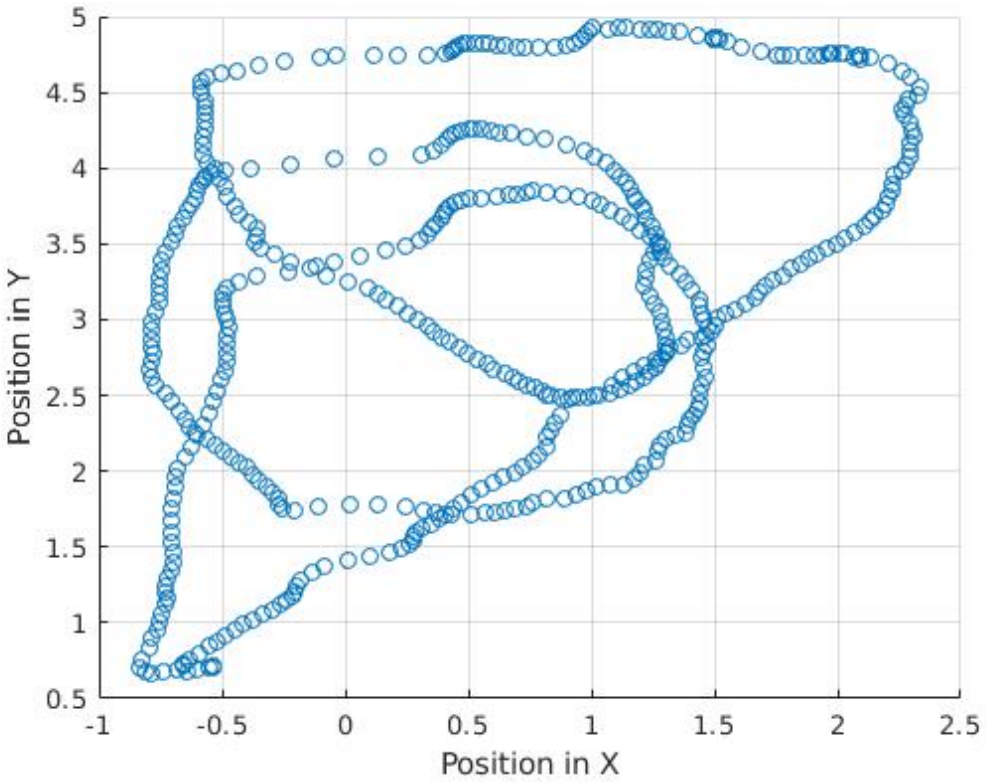


Figura 2: Kalman Filter prediction of the travelled trajectory, using the velocity as a measurement.

[2] http://dev.ti.com/tirex/content/mmwave_industrial_toolbox_2_5_2/labs/lab0011-pplcount/docs/pplcount_user_guide.html

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* In Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96).

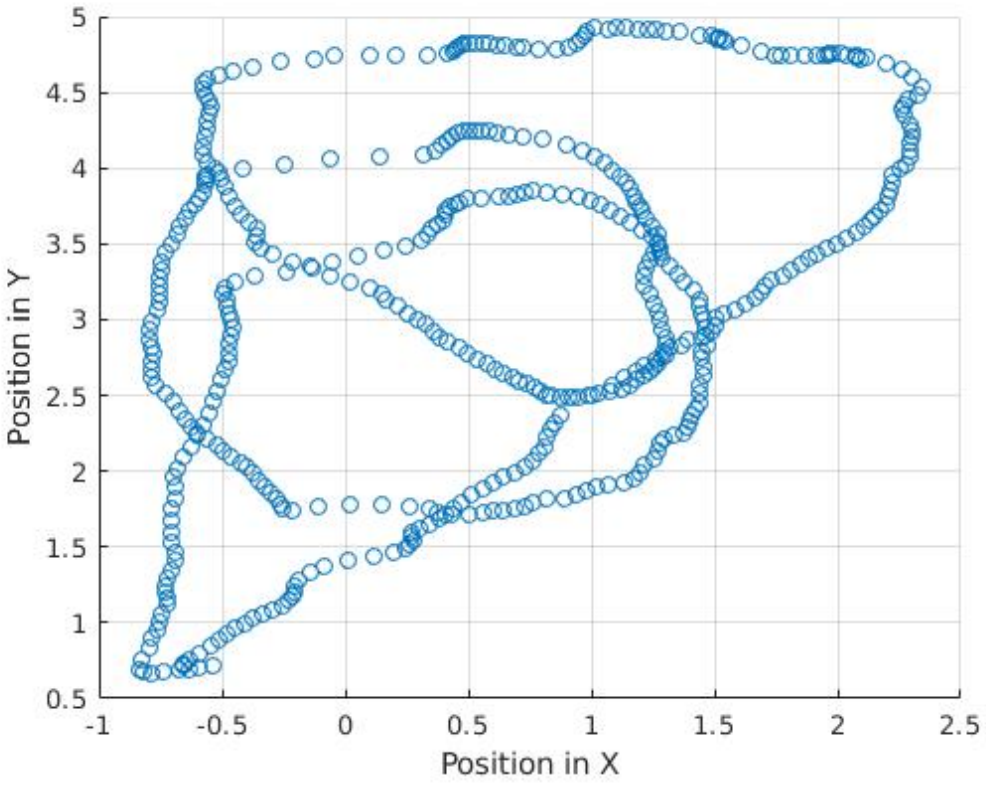


Figura 3: Kalman Filter prediction of the travlled trajectory, using the veloicty as an input

A

Matlab implementation of the Kalman Filter that uses the velocity as an input

```
delta_t = 0.05; % Time interval between measurements

samples = 575; % Number o measurements

I = eye(2); % Identity matrix

```

```
7  A = [1, 0;
8       0, 1]; % Evolution of the model
9
10 B = [delta_t, 0;
11      0, delta_t];
12
13 u = [velx; vely];
14
15 H = [0.45, 0; 0, 0.31];
16
17 Q = [0.05, 0;
18      0, 0.05];
19
20 R = [0.05,0;
21      0,0.05];  % Noise of the sensor (RADAR)
22
23 error_X = zeros(1, samples); % Vector to store errors along
       X dimension
24 error_Y = zeros(1, samples); % Vector to store errors along
       Y dimension
25
26 state_apriori = zeros(2, samples);
27 state_aposteriori = zeros(2, samples);
28 P_apriori = ones(2, 2);
29 P_aposteriori = ones(2, 2);
30 K = zeros(2, 2); % Kalman Gain
31
32 % Initial condition on the state, X and Y.
33 z = [posx; posy];
34 % Prediction equations
35 state_apriori(:,1) = [posx(1); posy(1)];
36 P_apriori = A*I*A.' + Q;
37 % Update equations
38 K = P_apriori/(P_apriori + R);
```

```
39 P_aposteriori = (I - K)*P_apriori;
40 state_aposteriori(:,1) = state_apriori(:,1)+K*(z(:,1) -
       state_apriori(:,1));
41
42 % Calculate the errors
43 error_X(1) = abs(posx(1)-state_aposteriori(1,1));
44 error_Y(1) = abs(posy(2)-state_aposteriori(2,1));
45
46 for j=2:samples
47     % Prediction equations
48     state_apriori(:,j) = A*state_aposteriori(:,j-1) + B*u
           (:,j-1);
49     P_apriori = A*P_aposteriori*A.' + Q;
50     % Update equations
51     K = P_apriori/(P_apriori+R);
52     P_aposteriori = (I - K)*P_apriori;
53     state_aposteriori(:,j) = state_apriori(:,j)+K*(z(:,j) -
           state_apriori(:,j));
54     error_X(j) = abs(posx(j)-state_aposteriori(1,j));
55     error_Y(j) = abs(posy(j)-state_aposteriori(2,j));
56 end
```