Karlsruher Forschungsberichte aus dem
Institut für Hochfrequenztechnik und Elektronik

Band
101

Alexandros Ninos

# Multi-User Gesture Recognition with Radar Technology

KIT Scientific Publishing

Alexandros Ninos

# Multi-User Gesture Recognition with Radar Technology

Karlsruher Forschungsberichte
aus dem Institut für Hochfrequenztechnik und Elektronik

*Herausgeber: Prof. Dr.-Ing. Thomas Zwick*
*Prof. Dr. Ing. Ahmet Cagri Ulusoy*

**Band 101**

Eine Übersicht aller bisher in dieser Schriftenreihe erschienenen Bände finden Sie am Ende des Buches.

# Multi-User Gesture Recognition with Radar Technology

by
Alexandros Ninos

KIT Scientific Publishing

Karlsruher Institut für Technologie
Institut für Hochfrequenztechnik und Elektronik

Multi-User Gesture Recognition with Radar Technology

Zur Erlangung des akademischen Grades eines Doktor-Ingenieurs
von der KIT-Fakultät für Elektrotechnik und Informationstechnik des
Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation

von Dipl.-Ing. Alexandros Ninos

Tag der mündlichen Prüfung: 11. April 2022
Hauptreferent: Prof. Dr.-Ing. Thomas Zwick
Korreferent: Prof. Dr.-Ing. Michael Heizmann

**Impressum**

Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

KIT Scientific Publishing is a registered trademark
of Karlsruhe Institute of Technology.
Reprint using the book cover is not allowed.

www.ksp.kit.edu





Print on Demand 2022 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 1868-4696
ISBN 978-3-7315-1225-7
DOI 10.5445/KSP/1000148867

# Preface of the Editor

Radars are not only used in aerospace and defense. Today, we encounter radars in many areas of everyday life without explicitly noticing them. Especially motion detection, e.g. for opening doors or in alarm systems, is widespread. Another important field of application is automotive environment sensing. Thanks to enormous progress in semiconductor technologies as well as in packaging and interconnection techniques, radar systems have been increasingly integrated into vehicles for about 20 years. Although radar systems do not offer the resolution of optical systems by far, they can, in contrast to them, still deliver useful measured values even in bad weather conditions or in a dirty state. In addition, radar sensors have the advantage that they can be mounted invisibly behind plastic surfaces and, unlike cameras, do not permit the identification of persons. This, together with the technological possibilities that now allow a complete radar system to be integrated in a chip, gives radar technology enormous potential for new areas of application in the future. One option that has been the subject of intensive research for several years is gesture recognition for controlling various applications ranging from consumer electronics and household appliances to industrial plants and machinery. The algorithms used for this purpose are based on machine learning methods, with a clever selection of features extracted from radar measurements and the design of the radar system itself playing a particularly important role.

This is exactly where the work of Dr. Alexandros Ninos comes into play. In Corporate Research at Robert Bosch GmbH, he performed scientific research on gesture recognition with radar sensors. In contrast to most previous research on the subject, the author has focused on macro gestures. After building a radar system for macro-gesture detection and he implemented a real-time gesture detection for this purpose. In particular, in addition to Doppler, he has included azimuth angle in his feature list for the neural network, achieving 94.3% accuracy for his 10 classes of gestures. Based on openly available human motion generation software, Dr. Ninos has created an automatic tool for generating

macro gestures including the radar backscattering behavior of those gestures. This allows a large number of gestures to be generated very quickly for training the network. He was able to show that a network trained with this synthetic data still achieved 84.2% accuracy on real test data. In the third part of the work, novel tracking and clustering techniques were implemented to separate and recognize gestures of multiple people by the radar system.

The work of Dr. Alexandros Ninos thus represents an important innovation to the state of the art. I am sure that his innovative concepts and the remarkable real-time demonstrator will draw much attention and find many followers worldwide. For Dr. Ninos, with his creativity and extremely high efficiency, I wish him further much success in his career and economic endeavors.

Prof. Dr.-Ing. Thomas Zwick

- Institute Director -

# Zusammenfassung

Digitalgeräte wie Mobiltelefone, Computer und eingebettete Systeme in Fahrzeugen spielen eine wichtige Rolle im heutigen Alltag. Gleichzeitig steht das Internet der Dinge unmittelbar bevor, was bedeutet, dass bald die meisten Geräte in einem Smart Home Teil eines Systems sein werden, das auf die Befehle eines Benutzers reagiert und dabei optimal auf die Umgebung abgestimmt ist. Für ein solches intelligentes Gebäude wären Informationen über die Anzahl der Personen in den einzelnen Räumen aus verschiedenen Gründen erforderlich, z. B. zur Optimierung der Belüftung. Darüber hinaus werden immer mehr Anwendungen eine berührungslose Mensch-Maschine-Schnittstelle für Unterhaltungs- und Hygienezwecke erfordern.

Die oben genannten Funktionen können mit einem üblichen kamerabasierten System zur ständigen Überwachung der Bewohner umgesetzt werden. Radarsensoren stellen eine bessere Alternative dar, da sie keine Bedenken hinsichtlich des Datenschutzes aufkommen lassen, billiger sind, wenig Strom verbrauchen und hinter Kunststoffstoßstangen versteckt werden können. Auf dem Verbrauchermarkt wurden sie jedoch hauptsächlich für Mikrogestenerkennung eingesetzt (d. h. für kleine Handbewegungen nur wenige Zentimeter vom Gerät entfernt) und nicht als multifunktionales System zur Verfolgung und Erkennung des Benutzerverhaltens. Ziel dieser Doktorarbeit ist die Entwicklung eines solchen Radarsystems zum Einsatz in Verbraucheranwendungen.

Zunächst habe ich mich auf die Gestenerkennung auf Makroebene konzentriert, d. h. der Benutzer befindet sich einige Meter entfernt und kann sich überall im Sichtfeld des Geräts befinden; ein solches Szenario ist bei Alltagsaufgaben in einem Haus sehr häufig. Der Grund warum ich diese Anwendung ins Auge gefasst habe, liegt in zwei Punkten. Erstens sind Makrogesten mit Radar die am wenigsten in der Literatur erforschte Anwendung, obwohl sie bei der berührungslosen Mensch-Maschine-Interaktion in einem Smart-Home-System die wichtigsten sind. Zweitens bieten stufenweise Verbesserungen von Mikrogesten, die in unmittelbarer Nähe des Geräts erfolgen, keine deutliche Verbesserung des Funkti-

onsspektrums oder des Bedienungskomforts im Vergleich zu Touchscreens, die allgegenwärtig sind und wesentlich mehr Funktionen bieten. Nachdem ich die Gesten ausgewählt hatte, konnte ich mit der Datenerhebung anfangen, wobei sich zehn Versuchspersonen beteiligten. Nach gründlicher Literaturrecherche und Evaluierung bekannter Methoden habe ich eine neuartige Signalverarbeitungskette entwickelt, die Informationen aus der Ankunftsrichtung integriert und empirische Merkmalsextraktion verwendet. Die Genauigkeit der Testdaten erreichte 95% bei einer Modellgröße von nur 34kB.

Während der vorgenannten Arbeit stellte ich drei Gründe fest, die die Datenerhebung erschweren. Die Gesten müssen an vielen Stellen innerhalb des Sichtfelds erfasst werden, da die Radialgeschwindigkeit (einer der vier Outputs des Sensors) stark von der Lage der Versuchsperson sowie von der Armbewegung abhängt. Außerdem muss die jeweilige Geste mit unterschiedlicher Geschwindigkeit erfolgen, um möglichst mehrere Szenarien zu erfassen. Schließlich gibt es eine gewisse Variabilität in der Art und Weise, wie Menschen ihre Arme bewegen, und dies muss im Trainingsdatensatz erfasst werden. Die Erfassung von Proben von verschiedenen Personen, an vielen Orten und in unterschiedlichen Geschwindigkeiten erfordert daher einen hohen manuellen Aufwand. Die Erzeugung von synthetischen Daten ist ein im maschinellen Lernen sehr beliebtes Konzept zur Lösung ähnlicher Probleme bei der Bilderkennung. Ich habe ein solches System für mmWave-Anwendungen entwickelt, welches Proben für sieben Gesten erstellen kann, wobei die Bewegungsgeschwindigkeit und die Lage des virtuellen Benutzers unterschiedlich sind. Nachdem ich 600 Proben erstellt hatte, programmierte ich ein Modell mit der gleichen Signalverarbeitungskette wie zuvor und testete es mit dem echten bereits gesammelten Datensatz. Die Genauigkeit erreichte 84,2%, ein recht hoher Wert, bedenkt man, dass beim Training keine echte Probe verwendet wurde.

Im nächsten Teil meiner Arbeit habe ich mich auf Mehrbenutzerszenarien konzentriert. Zunächst implementierte ich zwei häufig verwendete Algorithmen, Group Tracker und Cluster First Track Later, die für die Verfolgung von Personen konzipiert sind. Für jede verfolgte Person im Sichtfeld habe ich die bestehende Pipeline zur Gestenerkennung laufen lassen und so Gestenerkennung und Tracking gleichzeitig erreicht. Nachdem ich mit mehreren nahen beieinander liegenden Objekten experimentiert hatte, stellte ich fest, dass das Tracking eine entscheidende Komponente ist. Wenn z. B. die Abbildung der Punktziele zu bestimmten Strecken für einige wenige Frames nicht exakt ist,

wird die Gestenerkennung fehlschlagen. In bestimmten Szenarien haben sich die bestehenden Methoden nicht bewährt. Deshalb habe ich einen neuen aus beiden Methoden zusammengesetzten Ansatz entwickelt, den ich als „Group Tracker with Clustering“ [Cluster-Tracking (auch Gruppen-Tracking genannt)] bezeichnen möchte.

Der letzte Schritt in meiner Forschung bestand darin, herauszufinden, wie der Benutzer dem System eine kontinuierliche Eingangsgröße bereitstellen kann. Dies würde viel mehr Verbraucheranwendungen ermöglichen, da der Benutzer nicht nur einen Befehl aus einer zulässigen Liste auswählen, sondern auch einen Wert angeben könnte. Unter Berücksichtigung der Einschränkungen der Millimeter-Wellen-Technologie habe ich die intuitivste Methode gefunden, die Hand des Benutzers zu verfolgen. Ich habe eine Verarbeitungskette entwickelt, die, sobald sie ausgelöst wird, die Spur des Benutzers in Körper und Hand aufteilt und dann beide weiterverfolgt. Für die Erfassung der Bodenwirklichkeit habe ich ein kamerabasiertes System mit einer RGB-D-Kamera und einer üblichen Pipeline zur Lagebestimmung verwendet. Ich stellte fest, dass bei schnellen Handbewegungen die Abweichung der geschätzten Lage der Handverfolgung erhöht wurde. Dies war zu erwarten, da das Radar die Lage und nicht den Geschwindigkeitsvektor der Hand erfasst. Daher benötigte der Filter einige Messschritte, bis er zu den richtigen Werten konvergierte. Zur Verbesserung habe ich zwei Radarknoten in einem losen gekoppelten Netz verwendet und ihre Messungen kombiniert. Auf diese Weise schätzte ich den Geschwindigkeitsvektor der Hand in einem einzigen Snapshot und verwendete diese Information in einem adaptiven Kalman-Filter, den ich als „Multi-Model with Velocity Estimation“ [Multimodell mit Geschwindigkeitsabschätzung] bezeichnete.

# Abstract

Digital devices like mobile phones, computers, and embedded systems in vehicles play an important role in our everyday lives. At the same time, the presence of Internet of Things is imminent, which means that soon the majority of the devices in a smart house will be part of a system that senses and reacts to a user's commands optimally, in accordance with the surroundings. Such a smart building would inevitably require information about the number of people in each room for several reasons, such as for optimizing the ventilation. In addition, more and more applications will require a touchless human-machine interface for purposes that range from entertainment to hygiene.

The aforementioned features could be implemented with a state-of-the-art camera-based system that would continually monitor the residents. Radar sensors present a better alternative since they do not raise privacy concerns, offer cheaper, low-power solution and can be hidden behind plastic casings. However, in the consumer market they have been used mostly for micro-gesture recognition (i.e., small hand movements few centimeters from the device) and not as a multi-functional system that can track and recognize user behavior. The aim of this thesis is the development of such a Radar system that could be used for consumer applications.

Initially, I focused on gesture recognition on a macro-level, in which case the user is at a distance of a few meters and could be anywhere in the field of view of the device; such a scenario is very common in everyday tasks in a house. The reason why I targeted this application is twofold. Firstly, macro-gestures with Radar are the least studied application in the literature, even though they are the most prominent ones in touchless human-machine interactions in a smart home system. Secondly, incremental improvements of micro-gestures, which are performed in close proximity to the device, do not provide significant improvement in the range of functionalities or ease of use, in comparison to touch screens, which are ubiquitous and offer much more functionality. Once I selected which gestures would be useful, I started the data collection

that included ten subjects. After a thorough literature research and evaluation of state-of-the-art methods, I developed a novel signal processing chain that integrates information from direction of arrival and uses empirical feature extraction. The accuracy of the test set reached 95% while the model size is only 34kB.

During the aforementioned work, I came across three reasons that make data-collection hard. The gestures have to be captured in many locations within the FoV because the radial velocity information (one of the four outputs of the sensor) heavily depends on the position of the subject as well as the arm motion. Furthermore, each gesture has to be performed with different speed in order to capture more possible scenarios. Finally, there is some variability in the way people move their arms which has to be captured in the training dataset. As a result, recording samples from different people, in many locations and different speeds requires significant manual labor. A synthetic dataset generator is a concept quite popular in machine-learning, that solves similar problems in image recognition tasks. I developed such a system for mmWave applications, that is capable of generating samples for seven gestures, with variable speed of execution and location of the virtual user. After generating 600 samples, I trained a model using the same signal processing chain as before and tested it with the real dataset that I had already collected. The accuracy reached 84.2%, a quite high value considering that no real sample was used during training.

In the next part of my work, I focused on multi-user scenarios. Initially, I implemented two commonly used algorithms, Group Tracker and Cluster First Track Later, that are designed for people tracking. For each tracked person in the FoV I ran the existing gesture recognition pipeline, thus achieving simultaneous gesture recognition and tracking. After experimenting with multiple objects close to each other, I found that the tracking part is a crucial component. For example, in case that the mapping of point-targets to tracks is not accurate for a few frames, the gesture recognition part will fail. In certain scenarios existing methods did not perform well, that is why I developed another approach that I called Group Tracker with Clustering, with elements from both methods.

The final step in my research was to figure out a way so that the user could provide an analog input to the system. This would enable many more consumer applications, since the user would be able not only to select one command among an allowed list, but also to provide a value. The most intuitive way that I found, taking into consideration the limitations of mmWave technology, is to

track the hand of the user. I developed a processing chain that, once triggered, splits the track of the user into body and hand and then continues to track both. For collecting ground-truth, I used a camera-based system with an RGB-D camera and a state-of-the-art pose-estimation pipeline. I found that during fast hand maneuvers, the residual of the estimated position of the hand-track was increasing. This was expected since the Radar measures the position and not the velocity vector of the hand. Thus, the filter required a few measurement frames until it converged to the correct values. In order to improve that, I used two Radar nodes in a loosely coupled network and combined their measurements. This way, I estimated the velocity vector of the hand in a single snapshot, and used that information in an adaptive Kalman Filter that I called “Multi-Model with Velocity Estimation”.

# Acknowledgments

I would like to thank Jürgen Hasch for agreeing to provide the technical supervision of this Thesis. The discussions that we had were invaluable and his guidance helped me reach a good understanding on the interdisciplinary topic of Radar sensors. I would like to thank my Professor Thomas Zwick for accepting me as one of his PhD students and for always being very precise and honest in his feedback. Furthermore, I would like to thank my second supervisor, Professor Michael Heizmann, for his support during the last part of the PhD. Next, I would like to thank the group manager Thomas Binzer for providing all the needed support like hiring students and creating an environment suitable for research. Many colleagues from the department have supported me throughout my work. I would like to thank Gor Hakobyan for providing invaluable assistance in the signal processing topics, Tobias Schmid for helping me with infrastructure topics related to software, Martin Fink for his assistance in the analog-design domain and Robert Korn for the support while developing the Radar demonstrator.

I would not have reached this state without the continuous support of my parents and brother. I would like to thank my father for passing on his out-of-the-box thinking and the winning mentality, my mother for teaching me humility and patience as well as to have faith. Finally, I would like to thank my wife for providing me with unfailing support, continuous encouragement and always guiding me towards the end of the tunnel.

# Contents

# Abbreviations and Acronyms

## Acronyms

| | |
|---|---|
| **2D-FFT** | 2 Dimensional Fast Fourier Transform |
| **ADC** | Analog to Digital Converter |
| **ANOVA** | Analysis of Variance |
| **API** | Application Programming Interface |
| **ASIC** | Application Specific Integrated Circuit |
| **CA CFAR** | Cell Averaging Constant False Alarm Rate |
| **CFAR** | Constant False Alarm Rate |
| **CMKF** | Converted Measurements Kalman Filter |
| **CNN** | Convolutional Neural Network |
| **CUT** | Cell Under Test |
| **DAC** | Digital to Analog Converter |
| **DBF** | Digital Beamforming |
| **DBSCAN** | Density-Based Spatial Clustering of Applications with Noise |
| **DFT** | Discrete Fourier Transform |
| **DML** | Deterministic Maximum Likelihood |
| **DoA** | Direction of Arrival |

| | |
|---|---|
| **FFT** | Fast Fourier Transform |
| **FMCW** | Frequency Modulated Continuous Wave |
| **FoV** | Field of View |
| **FPGA** | Field-Programmable Gate Array |
| **GPU** | Graphics Processing Unit |
| **IC** | Integrated Circuit |
| **IHE** | Institut für Hochfrequenztechnik und Elektronik |
| **KF** | Kalman Filter |
| **KIT** | Karlsruher Institut für Technologie |
| **KPI** | Key Performance Indicator |
| **Lidar** | Light Detection and Ranging |
| **LNA** | Low-Noise Amplifier |
| **LSTM** | Long Short-Term Memory |
| **mD** | micro-Doppler |
| **MIMO** | Multiple Input Multiple Output |
| **ML** | Machine Learning |
| **MLP** | Multi Layer Perceptron |
| **MMVelEst** | Multiple Model with Velocity Estimation |
| **mmWave** | millimeter Wave |
| **NCI** | Non Coherent Integration |
| **NF** | Noise Figure |
| **OPTICS** | Ordering Points To Identify the Clustering Structure |

| | |
|---|---|
| **OS CFAR** | Ordered Statistics Constant False Alarm Rate |
| **OMP** | Orthogonal Matching Pursuit |
| **OOP** | Object Oriented Programming |
| **PC** | Personal Computer |
| **PCB** | Printed Circuit Board |
| **PGA** | Programmable Gain Amplifier |
| **PLL** | Phase-Locked Loop |
| **PTP** | Precision Time Protocol |
| **R-CNN** | Region based Convolutional Neural Network |
| **Radar** | Radio Detection and Ranging |
| **RCS** | Radar Cross-Section |
| **RDM** | Range Doppler Maps |
| **SIMO** | Single Input Multiple Output |
| **SNR** | Signal to Noise Ratio |
| **TDM** | Time Domain Multiplexing |
| **TuT** | Track under Test |
| **ULA** | Uniform Linear Array |
| **UML** | Unified Modeling Language |
| **VCO** | Voltage Controlled Oscillator |

# Constants

| | |
|---|---|
| $\pi$ | Ratio of a circle's circumference to its diameter: 3,14159 . . . |
| $c_0$ | Speed of Light: 2.99792458e8 m/sec |

# Latin Symbols and Variables

## Lowercase Letters

| | |
|---|---|
| $a_k$ | Attenuation of wave |
| $\mathbf{a}_s(\theta)$ | Spatial steering vector |
| $d$ | Distance |
| $d_a$ | Constant distance between antennas in an ULA |
| $f_{beat}$ | Beat frequency |
| $f_c$ | Center frequency |
| $f_D$ | Doppler frequency |
| $f_{min}$ | Minimum frequency of chirp |
| $f_{Rx}$ | Receiving frequency |
| $f_{Tx}$ | Transmitting frequency |
| $k$ | Index of sample |
| $k_\theta$ | Normalized spatial frequency in radians per sample |
| $k_r$ | Slope of chirp |
| $l$ | Index of chirp |
| $m$ | Index of range spectrum |
| $n$ | Index of Doppler spectrum |
| $p_q$ | Position of q-th antenna |
| $t_k^{Rx}$ | Signal reception time |

| | |
|---|---|
| $t_k^{Tx}$ | Signal transmission time |
| $\vec{u}$ | Kalman filter's input vector |
| $u_k^{IF}$ | Baseband time domain signal |
| $u_k^{Rx}$ | Received time domain signal |
| $u_k^{Tx}$ | Transmitted time domain signal |
| $\vec{v}$ | Kalman filter's measurement noise |
| $v_r$ | Radial velocity |
| $\vec{w}$ | Kalman filter's state noise vector |
| $\vec{x}$ | Kalman filter's state vector |
| $\vec{z}$ | Kalman filter's measurement vector |

## Capital Letters

| | |
|---|---|
| $B_{\text{chirp}}$ | Chirp bandwidth |
| $B$ | Kalman filter's input model |
| $F$ | Kalman filter's state transition matrix |
| $G_r$ | Gain of receiving antenna |
| $G_t$ | Gain of transmitting antenna |
| $H$ | Kalman filter's measurement matrix |
| $K$ | Number of samples in each chirp |
| $Kg$ | Kalman filter gain |
| $L$ | Number of chirps in chirp-sequence FMCW |
| $N_\theta$ | Number of angles to estimate during DoA |
| $N_{R_x}$ | Number of receiving antennas |
| $N_{T_x}$ | Number of transmitting antennas |
| $N_{VR_x}$ | Number of virtual receiving antennas |
| $P_r$ | Received power |
| $P_t$ | Transmitted power |

| | |
|---|---|
| $Q$ | Kalman filter's covariance of state noise |
| $R$ | Kalman filter's covariance of measurement noise |
| **D** | Range FFT |
| $Rx$ | Receiving antenna |
| $S$ | Kalman filter's residual covariance matrix |
| $T_{\text{chirp}}$ | Duration of chirp |
| $Tx$ | Transmitting antenna |
| **V** | Velocity FFT |

## Greek Symbols and Variables

| | |
|---|---|
| $\theta$ | Direction of arrival |
| $\lambda$ | Wavelength |
| $\lambda_c$ | Wavelength of the center frequency |
| $\sigma$ | Radar cross section |
| $\tau$ | Delay between transmission and reception |
| $\chi$ | Ambiguity function |

## Mathematical Symbols

| | |
|---|---|
| $\mathbf{a}$ | Complex numbers |
| $\vec{a}$ | Vector |

# 1 Introduction

## 1.1 Motivation

Radio Detection and Ranging (Radar) sensors have been used in the automotive industry for developing safety features in series production. In addition, their low-cost, compact size, relatively low power consumption, and their robustness in any weather made them an indispensable component for any experimental autonomous vehicle. During the last years there have been significant breakthroughs that improved the resolution in all three available measurement dimensions, distance, radial velocity and Direction of Arrival (DoA).

In my thesis, I use this sensing technology for consumer applications, in which the typical Key Performance Indicators (KPI) that characterize the resolution and maximum unambiguous measurements are not useful. The important part lies in the algorithm development that leads to novel use-cases. In some scenarios, calculating the distance with high accuracy could be meaningless, in others the DoA could provide little extra information. Thus, the term Radar does not represent what is of importance here. However, throughout the thesis I will use this term to designate the sensor system that I use.

Existing work has focused on recognizing the so called micro-gestures, those are hand or even finger movements performed few centimeters on top of the device [LGK+16]. I believe that such a feature is not very useful since it offers similar functionality to a touchscreen or buttons; the main benefit is the touchless interaction which is suitable for applications with hygiene requirements. However, I was motivated by this work and decided to improve it. I focused on applications in which the user is positioned few meters away from the sensor and would like to interact with it, i.e., in scenarios where the user needs to control certain devices in the house.

## 1.2 Scientific Contributions

The main contributions of my work are in the field of signal processing of Radar data. These are:

1. Gesture recognition in macro level using empirical feature extraction [NHZ21b].
2. Synthetic dataset generator for macro-gesture recognition with mmWave technology [NHAZ21].
3. Multi-user gesture recognition in macro level [NHZ21a].
4. Improvement in people tracking with Radar sensors [NHHZ22].
5. Hand tracking with Radar sensors [NHHZ22].

## 1.3 Outline

The remainder of this Thesis is structured as follows:

- Chapter 2 introduces the topic of Radar, provides basic theoretical background, explains the modulation, and the baseband signal processing pipeline that is used throughout this Thesis.
- Chapter 3 presents the experimental hardware system that was used, its components, and the calibration process.
- Chapter 4 introduces the concept of macro-gesture recognition with mmWave technology and its advantages in comparison to the more popular micro-gestures. It describes the gestures that were selected, as well as the data collection from ten subjects. Finally, it presents the signal processing pipeline that I developed, the experimental results and comparisons with the state-of-the-art.
- Chapter 5 is dedicated to the description of a synthetic dataset generator that I developed, which is capable of simulating seven gestures and of generating an artificial dataset with 600 samples. In addition, it discusses why such a generator is of importance for gesture recognition tasks.

Finally, synthetic samples are compared with real ones and a machine learning model trained on synthetic samples is tested on real samples.

- Chapter 6 focuses on the use-case of people tracking. Specifically, it presents two state-of-the-art methods for people tracking using mmWave technology. Moreover, it describes a method that I developed which is robust in certain complex scenarios with multiple users.
- Chapter 7 combines many of the aforementioned topics. It introduces the concept of multi-user gesture recognition and shows results from two users performing gestures. Hand-tracking based on mmWave is also introduced; a novel adaptive Kalman Filter is presented that uses information from two Radar nodes to better track hand maneuvers.
- Chapter 8 summarizes the most important contributions of the Thesis and provides suggestions for further improvements.

Fig. 1.1 summarizes the connection between the different chapters of the Thesis.

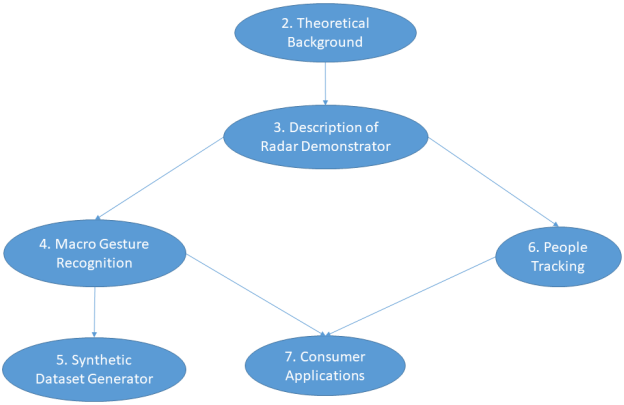



Figure 1.1: Chapters of the Thesis and their connection

Throughout this work the pronoun “we” is used when something was developed or designed with other colleagues in the department. Whereas, when something was developed solely by myself the pronoun “I” is used.

Python programming language in combination with Numpy [HMvdW+20] was used for matrix manipulation and Scipy [SVG+20] for scientific computing. In Appendix A.3 I provide Unified Modeling Language (UML) diagrams with the software architecture that I designed and developed for this work.

# 2 Radar Basics

This chapter provides a basic theoretical background for Radar sensors. To remain as much focused as possible to the contributions of this Thesis, I will present only the basics of the FMCW Radar sensor, focusing on the formulas to calculate distance, radial velocity and DoA. Then, I will present the Multiple Input Multiple Output (MIMO) concept and I will elaborate on the baseband signal processing pipeline.

## 2.1 Radar Principle

Radars' main usage has been the detection of objects (namely Radar targets), as well as the estimation of their parameters, such as the distance of the object, the radial velocity and the DoA. Radars transmit electromagnetic waves which are reflected from the aforementioned Radar targets. Fig. 2.1 shows a typical example of a target in the Field of View (FoV) of a Radar sensor and the coordinate system that will be used throughout this Thesis.

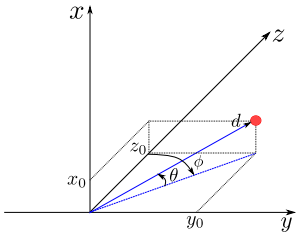



Figure 2.1: Target in the FoV of a Radar sensor and 3D coordinate system, $d$ stands for distance, $\phi$ for azimuth DoA and $\theta$ for elevation DoA.

### 2.1.1 Distance-Velocity measurement

In general, Radars use the time-of-flight principle to measure the distance of an object. In Eq. 2.1 the distance is calculated using the delay $\tau$ between transmission and reception of the signal and the velocity of propagation $c_0$ which is approximately equal to the speed of light in vacuum.

$$d = \frac{c_0 \tau}{2} \tag{2.1}$$

Using the Doppler frequency shift $f_D = f_c - f_{Rx}$, in other words the difference of the transmitted and received frequency, it is possible to calculate an approximation of the radial velocity as in Eq. 2.2.

$$v_r \approx -\frac{c_0 f_D}{2 f_c} = -\frac{\lambda_c f_D}{2} \tag{2.2}$$

where $f_c$ denotes the transmitting frequency and $\lambda_c$ the equivalent wavelength of the signal.

### 2.1.2 Direction of Arrival

In order to calculate the position of a target in 3D space, an estimation of the DoA of the reflected waves is needed. In this work, I use the concept of Digital Beamforming (DBF), which does not require mechanical steering of the antennas. Following, the fundamentals of DBF and its assumptions are introduced.

The signals from all directions are received simultaneously by multiple receiving antennas. Due to different antenna positions, the receiving signals will have a phase difference, which contains the information about DoA. As depicted in Fig. 2.2, the antenna positions can be regarded as sample points of the spatial wave. Therefore, the DoA estimation corresponds to the estimation of spatial wave frequencies, i.e. the phase progression over the antenna elements. If the sample points are sufficiently dense, which is a prerequisite for the unambiguous coherent processing, the ratio of the array aperture and wavelength

determines the angular separability [LY11]; i.e. the capability to resolve two closely located targets.

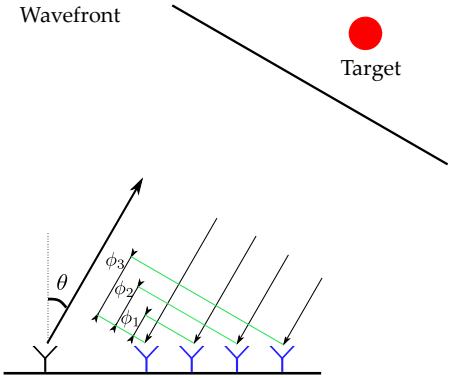



Figure 2.2: DoA estimation using antenna array with one transmitter and four receivers. The signals arrive at each receiving antenna with a phase-shift which depends on the antenna positions.

Assembling the input from the receiving antennas $N_{R_x}$ into column vector form, gives the snapshot of a linear array at a fixed time, as in Eq. 2.3.

$$\begin{aligned}\mathbf{Y} &= [\mathbf{y}[0] \quad \mathbf{y}[1] \quad \dots \quad \mathbf{y}[N_{R_x}-1]]^T \\ &= A[1 \quad e^{-j2\pi\frac{d_a \sin(\theta)}{\lambda_c}} \quad \dots \quad e^{-j2\pi\frac{(N_{R_x}-1)d_a \sin(\theta)}{\lambda_c}}]^T \\ &= A[1 \quad e^{-jk_\theta} \quad \dots \quad e^{-j(N_{R_x}-1)k_\theta}]^T \\ &= A\mathbf{a}_s(\theta) \end{aligned} \tag{2.3}$$

where $k_\theta = 2\pi d_a \sin(\theta)/\lambda_c$ is the normalized spatial frequency in radians per sample as projected into the plane of the array face, $\mathbf{a}_s(\theta)$ is the spatial steering vector and $d_a$ is the constant distance between antennas. Thus, there is a one-to-one relationship between the DoA of a plane wave and the spatial frequency across the array face. The range of $\theta$ is $\pm\pi$; equivalently the range of $k_\theta$ is $\pm 2\pi d_a/\lambda_c$. For an excellent review of array signal processing the reader is referred to [KV96, Ric14].

The positions of the antennas and generally the antenna array play an important role since the beam width and the radiation pattern are directly affected. In a nutshell, radiation pattern refers to the angular dependence of the strength of the radio waves from the antenna. Whereas beam width is the angle between the half power points of the main lobe, when referenced to the peak effective radiated power of the main lobe. It is a common practice to use Uniform Linear Array (ULA), in which case the elements have a constant distance. Recent works have used sparse arrays which allow the reduction of antenna elements while keeping the same aperture size [LY11]. The disadvantage of this trade-off is that this sparse setting introduces an increased side lobe level and potentially ambiguities. An important assumption in the DBF is the so-called narrow-band assumption. In other words, the array aperture (i.e. the physical size measured in wavelengths) must be smaller than the inverse relative bandwidth [KV96].

### 2.1.3 Radar Equation

Using the well known Radar equation [Ric14], it is possible to estimate the received power $P_r$ on an antenna, as a function of the transmitted power $P_t$, the gain of transmitting and receiving antennas $G_t$ and $G_r$, the Radar cross-section (RCS) $\sigma$, wavelength $\lambda$ and the distance of a target, according to the formulation given by Eq. 2.4.

$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 r_k^4} \tag{2.4}$$

Thus, the total attenuation $a_k$ from the transmitting antenna to target $k$ and back to the receiving antenna is calculated using Eq. 2.5.

$$a_k = \frac{P_r}{P_t} = \frac{G_t G_r \lambda^2 \sigma}{(4\pi)^3 r_k^4} \tag{2.5}$$

Using the time-delay and attenuation defined above, the voltage referred relationship between the received signal $u_k^{Rx}$ and the transmitted signal $u_k^{Tx}$ is provided by Eq. 2.6.

$$u_k^{Rx}(t) = \sqrt{a_k} u^{Tx}(t - \tau_k) \tag{2.6}$$

## 2.2 Chirp Sequence FMCW Signal Model

Various modulation schemes that provide distance and radial velocity measurements have been developed; we decided to use chirp-sequence FMCW because it is well-studied in the literature and many commercial sensors are already available [HTS+12] [KR14]. A system that uses such modulation transmits a series of chirps (also called ramps) with a constant amplitude, and a frequency that changes linearly. Once the signals are reflected by the targets and received, they are mixed with the transmit chirp. This results in a frequency which depends on the distance and radial velocity, known as beat frequency $f_{beat}$, as shown in Eq. 2.7.

$$f_{beat} = f_{Tx} - f_{Rx} = \tau_k \frac{B_{\text{chirp}}}{T_{chirp}} - f_D \tag{2.7}$$

The $T_{chirp}$, which stands for chirp duration, is relatively short and the slope relatively high. This way, the portion of the $f_{beat}$ induced by the distance of the target is significant. In other words, the Doppler term can be neglected. In order to measure radial velocity, many chirps are sequentially transmitted and processed similar to Doppler processing for pulse-Doppler Radar [Ric14]. Therefore, it is possible to calculate distance and velocity independently, through processing each chirp separately and the phases of consecutive chirps. In this Thesis, the samples within one chirp are referred to as fast-time samples, whereas the chirp is referred to as slow-time.

In detail, consider a linear frequency chirp $f_{Tx}(t)$ as in Eq. 2.8 with a slope $k_r$.

$$f_{Tx}(t) = f_{min} + k_r t \tag{2.8}$$

The modulated signal is calculated by integrating the frequency into the phase [Ric14], as in Eq. 2.9.

$$u^{Tx}(t) = e^{j2\pi \int f_{Tx}(t)dt} = e^{j(\pi k_r t^2 + 2\pi f_{min} t + \theta_0)} \tag{2.9}$$

$\theta_0$ being the phase of $u^{Tx}(t)$ for $t = 0$. Using the above and Eq. 2.5 yields the FMCW received signal as in Eq. 2.10.

$$u_k^{Rx}(t) = \sqrt{a_k} e^{j(\pi k_r (t-\tau_k)^2 + 2\pi f (t-\tau_k) + \theta_0)} \tag{2.10}$$

By mixing the transmit and received signal, the baseband signal $u_k^{IF}$ is calculated. Our Radar system creates a sequence of $L$ chirps, using linear frequency modulation [KR14], with each of them lasting $T_{\text{chirp}}$. Fig. 2.3 shows a sequence with four chirps and the echo signal from a target near the Radar.

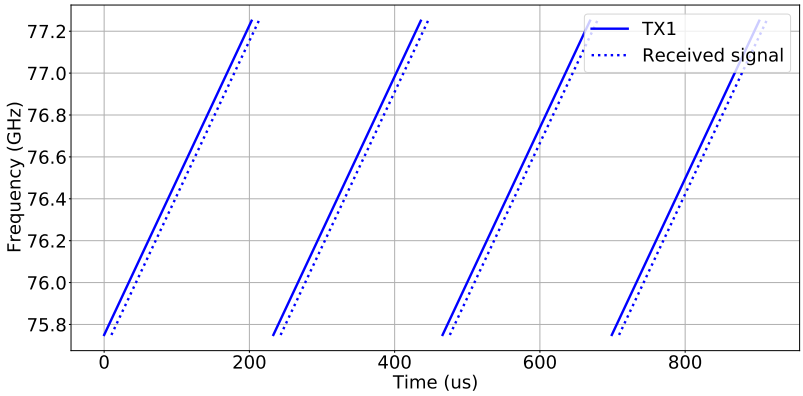



Figure 2.3: Linear FMCW with four chirps and return signal from one target (©IEEE [NHZ21b]).

The following equation describes the baseband signal, which is the output of mixing of transmitted and received signal:

$$\mathbf{s}(t, l) = e^{j2\pi(f_B t - f_D l T_{\text{chirp}} + \phi)} \tag{2.11}$$

where $l$ is the chirp index.

Table 2.1 provides the formulas for calculating the maximum unambiguous distance and radial velocity of a target, as well as their resolution. A more rigorous signal model can be found in [KR14].

| | Distance | Radial Velocity |
|---|---|---|
| Resolution | $\frac{c_0}{2B_{\text{chirp}}}$ | $\frac{\lambda}{2 \cdot L \cdot T_{\text{chirp}}}$ |
| Maximum | $\frac{c_0 \cdot K}{2B_{\text{chirp}}}$ | $\frac{\lambda}{4 \cdot T_{\text{chirp}}}$ |

Table 2.1: Resolution and maximum unambiguous value for distance and radial velocity (©IEEE [NHZ21b]).

## 2.3 MIMO Principle

MIMO Radar is widely used in target detection due to the high angular resolution that it offers. A MIMO system that consists of $N_{Tx}$ transmitting and $N_{Rx}$ receiving elements could be equivalent, as far as the angular resolution is concerned, to a virtual array with $N_{Tx} \cdot N_{Rx}$ receiving elements. For a MIMO Radar to easily separate the signals transmitted by different antennas, the most intuitive and simple way is Time Domain Multiplexing (TDM). In TDM, each transmitter transmits its own waveform alternatively, and there is no overlap between any two transmissions [ZR12, HY19].

A typical example is shown in Fig. 2.4, in which the transmitting duration and pause between chirps is selected so that the total duration of two consecutive chirps is equal to that of Fig. 2.3. This way, the resolution and maximum unambiguous radial velocity will be the same in both cases.

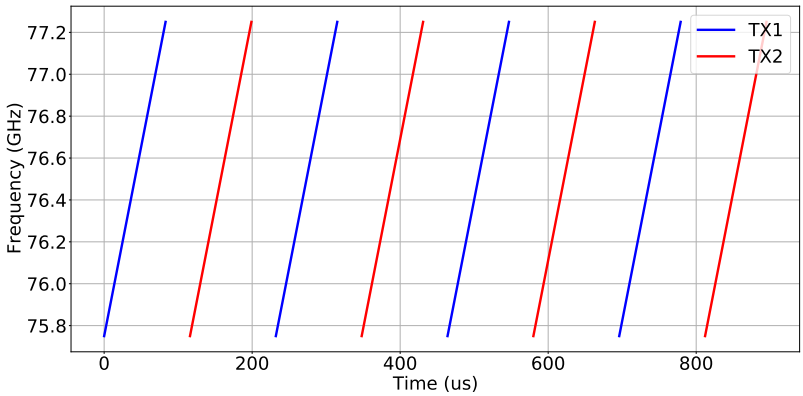



Figure 2.4: Time Division Multiplexing using FMCW (©IEEE [NHZ21b]).

## 2.4 Baseband Processing for Chirp Sequence FMCW

Using Analog to Digital Converters (ADC) for each receiver, the continuous time signal of Eq. 2.11 is sampled and for each chirp, Discrete Fourier Transform (DFT) is applied, as shown in equation 2.12.

$$\mathbf{D}(m,l) = \sum_{k=0}^{K-1} s(k,l) \cdot e^{-j2\pi \frac{k \cdot m}{K}} \tag{2.12}$$

The sample index is denoted with $k$, the number of samples with $K$ and the beat frequency index with $m$. When this procedure is completed for all chirps, a second DFT is applied for each distance gate, in order to estimate the Doppler frequency, as shown in Eq. 2.13.

$$\mathbf{V}(m,n) = \sum_{l=0}^{L-1} \mathbf{D}(m,l) \cdot e^{-j2\pi \frac{l \cdot n}{L}} \tag{2.13}$$

where $n$ is the Doppler frequency index. The DFT can be computed with a Fast Fourier Transform (FFT) algorithm that reduces the complexity to $O(n \log n)$. Therefore, with an efficient 2D-FFT, a distance-velocity estimation is completed. Targets are represented by peaks at locations corresponding to their distances and velocities in a 2D image.

As mentioned in [KR14], due to the two measurements for beat and Doppler frequency, the distance $d$ and radial velocity $v_r$ for each detected target can be calculated as follows:

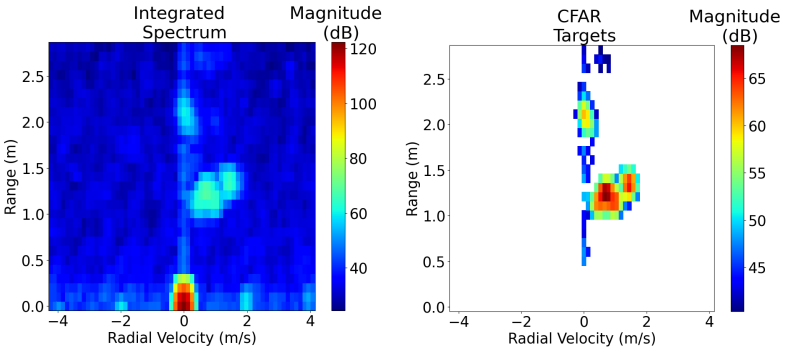
$$d = -(f_B + f_D)\frac{T_{\text{chirp}} \cdot c}{2 \cdot B_{\text{chirp}}} \tag{2.14}$$

$$v_r = -f_D \frac{\lambda}{2} \tag{2.15}$$

where $B_{\text{chirp}}$ is the chirp bandwidth.

In this Thesis, I use the FMCW MIMO Radar. Therefore, for each $Tx$-$Rx$ path one distance-velocity image will be generated. Signal to Noise Ratio (SNR) and thus the detection performance can be improved by integrating these images. I use Non-Coherent Integration (NCI), since it is superior to other integration methods in terms of computational efficiency and offers a reasonable increase in SNR [Hak18]. In NCI, phase information is discarded, instead, the squared magnitudes of the data samples from all $Tx$-$Rx$ combinations are integrated (i.e., summed).

The Constant False Alarm Monitoring (CFAR) family of algorithms determine the power threshold above which any return signal can be considered to originate from a target as opposed to one of the spurious sources. These could either be internal to the Radar receiver or from sources external to the Radar. If the threshold is set too low, more targets will be detected, at the expense of increased numbers of false alarms. On the other hand, if the threshold is set too high, fewer targets will be detected, together with low number of false alarms. The CFAR processor estimates the mean interference power in the Cell Under Test (CUT) by using the measured data in the adjoining cells. The most common approach is called Cell Averaging (CA) CFAR because the threshold is estimated from an average of the power in the cells adjoining the CUT. In this work I used Ordered Statistics (OS) CFAR which orders the reference window data samples to form a new sequence in ascending numerical order. The $k$-th element of the




(a) NCI spectrum

(b) Bins that surpass the noise floor.

Figure 2.5: Example of the NCI spectrum and the bins that surpass the noise floor calculated by CFAR.

ordered list is called $k$-th order statistic. For example, the first order statistic is the minimum and the $N$-th order statistic is the maximum. In OS-CFAR the $k$-th order statistic is selected as representative of the interference level and the threshold is set as a multiple of this value. More detailed information about detection fundamentals and CFAR can be found in [Ric14]. A typical example of an integrated Radar spectrum and the equivalent targets that pass the threshold level can be found in Fig. 2.5. It is important to note that the high values close to zero distance and velocity are due to the antenna coupling; they are suppressed before CFAR is applied.

In order to locate a target in three-dimensional space, the DoA in azimuth and elevation needs to be estimated. I use the Deterministic Maximum Likelihood (DML) approach which is an intuitive and fast method for single-target, single-snapshot scenario. The word deterministic comes from the assumption that the received signal wave-forms are deterministic and unknown. The idea is to "steer" the array in one direction at a time and measure the output power. For each detected target, a matrix multiplication of the complex spectrum $\mathbf{v}$ (a vector with $N_{VRx}$ elements), with the conjugate of the steering matrix takes place, according to Eq. 2.16.

$$Z = |\mathbf{v} * \mathbf{a}_s^*| \tag{2.16}$$

The absolute value of this multiplication will be maximum at the direction of arrival of the target. In Section 3.2, it is explained how to calculate such a steering matrix $\mathbf{a}_s$ by performing an experiment in an anechoic chamber. The number of rows in this matrix equals the number of virtual antennas $N_{VRx}$ and the number of columns equals the angles $N_\theta$ that need to be estimated. Finally, it is important to point out that before the angle estimation it is a common practice to compensate the motion-induced phase errors due to the TDM [BRW17].

At this point, the distance, radial velocity and direction of arrival in azimuth and elevation are available. In addition, an estimation of the SNR can be calculated, since the CFAR algorithm provides an estimation of the noise floor. These attributes are saved in a container for each detected target and passed to the next layers of processing. Fig. 2.6 shows the steps during baseband processing for a chirp sequence FMCW. Last but not least, in the Appendix A.3.1 I provide UML diagrams with the software architecture that I developed for the implementation of baseband processing.

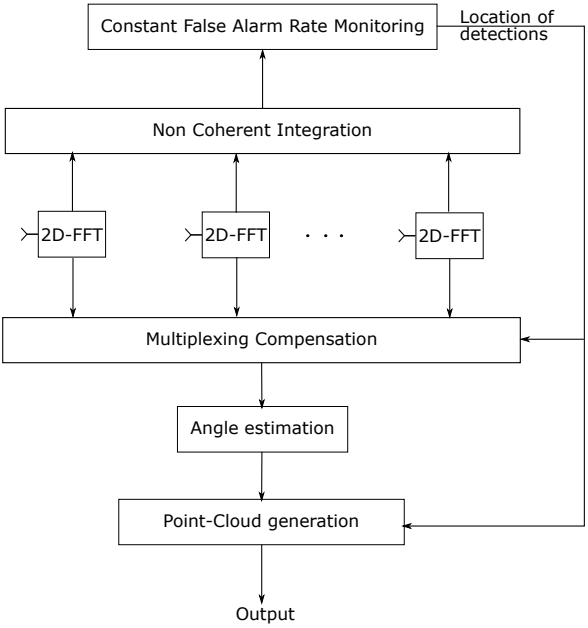



Figure 2.6: Baseband processing steps for chirp sequence FMCW. The time-domain signal will be processed through 2D-FFT and then non-coherently integrated. A CFAR algorithm will detect the target and provide the locations to the angle estimation algorithm. The latter will also use the complex spectrum from 2D-FFT to calculate the direction of arrival. Finally, all the estimated information, distance, radial velocity, DoA and SNR, will be saved in a container and will be given to the next layers of processing.

## 2.5 Alternative technologies

Ultrasound, camera-based and Light Detection and Ranging (Lidar) sensors could be an alternative solution for consumer applications. However, in our opinion Radars have significant advantages, which if exploited properly could lead to significantly better solutions.

Several products can already accomplish similar tasks by utilizing camera-based sensors, but there are certain drawbacks involved with such technology. Accurate detection of gestures is difficult because of varying illumination,

shadows, complex background, and other factors [CSBM18]. The complex articulated shape of the hand makes it hard to model the appearance of both static and dynamic gestures. Variation of gesture parameters due to spatio-temporal variance in hand postures makes the recognition process more difficult [CSBM18]. It is important to point out that processing images, frame to frame, for extracting hand information is usually computationally inefficient for portable devices [GWL19]. In addition, it is not possible to hide a camera behind non-transparent material like cloth or plastic. Finally, people tend to feel uncomfortable when they are in the FoV of a camera, due to privacy concerns [CA09].

Ultrasound sensors use sound waves with frequencies higher than the upper audible limit of human hearing. Ultrasound signals are not different from the "normal" (audible) sound signals in their physical properties, except that humans cannot hear them. A common application that is developed with this technology is the automatic door opener, in which case the sensor detects a person's approach and opens the door. Such sensors can detect intruders, since the ultrasound can cover a wide area from a single point. There are several reasons why this technology is not popular for smart-home applications. To begin with, the maximum range is usually 2-3 meters, which make it unsuitable for outdoor scenarios or large rooms. In addition, if the selected modulation allows Doppler measurement, the maximum unambiguous velocity is significantly lower in comparison to a Radar. Finally, the microphones and the speakers (i.e. receivers and transmitters) are significantly larger in comparison to an antenna optimized for mmWave frequency [ZLW].

Lidar measures the range of targets through light waves from a laser instead of radio or sound waves. The transmitter emits laser light at the target object, and the pulse is reflected if a target object lies in the FoV. The distance is then calculated by using the relationship between constant speed of light in the air, and the time of flight of the signal. The main advantages include very high resolution and accuracy, as well as fast update-rate, which makes it suitable for fast moving objects. Moreover, it provides shorter wavelength compared to Radar or Ultrasound which makes it suitable for creating 3D maps of an object. Finally, as this technology is not passive, it is appropriate both for day and night conditions. However, the beam-width is very narrow, thus the sensor has to be rotated in order to scan the scene in front of it. This requires a bulky mechanical

| Indicator/Technology | Camera | Ultrasound | Lidar | Radar |
|---|---|---|---|---|
| Cost | Low | Low | High | Low |
| Size | Small | Medium | Large | Small |
| Speed detection | No | No | No | Yes |
| Angular Resolution | High | Low | High | Medium |
| Object classification | High | No | Medium | Low |
| Distance estimation | Stereo | Yes | Yes | Yes |
| Deteriorate under poor lighting | Yes | No | No | No |
| Hidden integration | No | Limited | Optical window | Yes |

Table 2.2: Comparison of different technologies.

structure which increases the cost and maintenance requirements, to a value not suitable for consumer market.

An overview of the available technologies and their KPI is available in Table 2.2. A Radar system could be the ideal candidate for sensing applications in a smart-home application since it is low-cost, consumes low-power, can be hidden behind plastic bumps and has enough resolution to understand its surroundings.

# 3 Radar Demonstrator

## 3.1 Hardware Setup

In this Thesis, I use an experimental FMCW Radar sensor with two transmitters and four receivers. Fig. 3.1 shows a block diagram with the high level overview of the hardware. It consists of a high frequency and a baseband part, which are located on separate Printed Circuit Boards (PCB). The high frequency part includes the two-channel transmitter and four-channel receiver, a Voltage Controlled Oscillator (VCO) and a Phase-Locked Loop (PLL) capable of fast linear frequency ramps of up to 2 GHz bandwidth, as well as the transmit and receive antennas. The setup allows MIMO, operating multiple transmitters in TDM.

The baseband board contains the analog interface electronics, ADCs, digital logic, and power supply. The digital logic itself consists of an FPGA part, which controls the real-time operation of the Radar sensor and an ARM microcontroller at 800MHz, which acquires the data and communicates with a host PC connected over Ethernet. Fig. 3.2 shows an image of frontend and baseband boards.

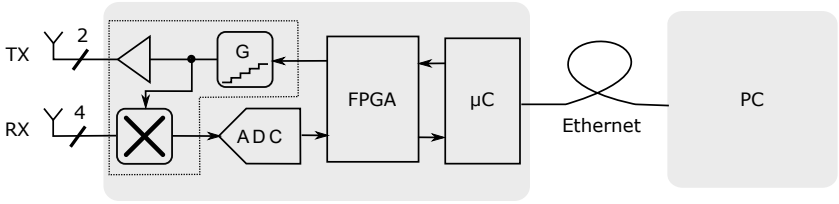



Figure 3.1: Simplified block diagram of the Radar sensor setup (©IEEE [NHZ21b]).

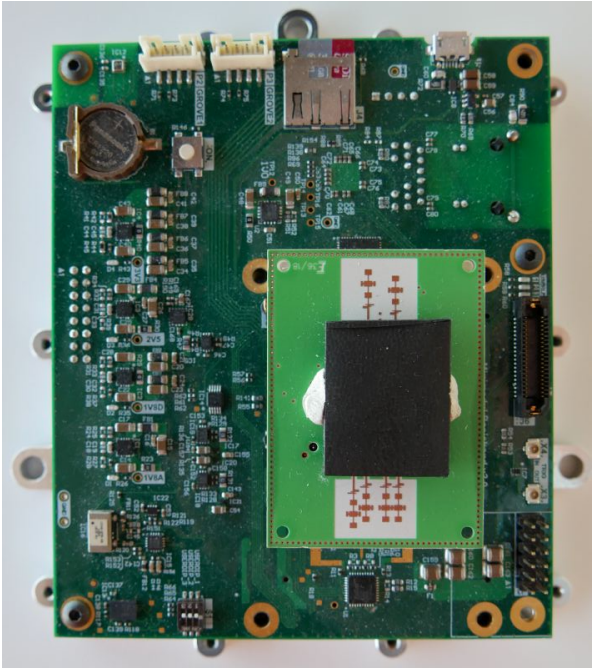


Figure 3.2: Radar hardware with frontend and baseband boards (©IEEE [NHZ21b]).

The center frequency is at 77 GHz, which is suitable for automotive but not for commercial or industrial applications. However, the signal processing pipelines that I developed can be used by a sensor that operates at 60 GHz without modifications. The frequency differs by only 20%. Therefore, no noticeable difference in reflectivity, velocity unambiguity or resolution is expected.

Last but not least, the antenna array consists of elements with three series fed patches and an additional matching structure at the feedline. The antennas are optimally positioned at a distance of approximately half lambda horizontally and the transmitters are placed accordingly at a distance of lambda to generate a MIMO virtual array. Thus, it is possible to calculate the DoA in azimuth and elevation.

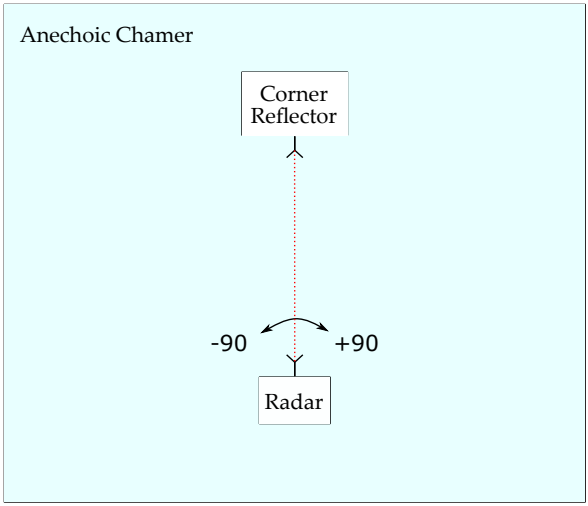



Figure 3.3: Calibration setup in an anechoic chamber with a rotating radar and a fixed corner reflector.

## 3.2 Antenna Array Calibration

Hardware imperfections, like differences in transmission line lengths, manufacturing uncertainties and variances in radio-frequency components, create the necessity for antenna calibration [HHZZ16]. Following the tutorial in [VRD+20], we conducted two calibration measurements using a corner reflector and a rotating Radar in azimuth and elevation respectively, and extracted the steering vector for each direction.

We use a corner reflector that we assume to be a point-target in the far field of the antenna array with a known azimuth angle $\theta = \theta_0$. As it was mentioned in Chapter 2, for a plane wave impinging the antenna array, the phase at the receive antenna is described by the exponential term in the signal model of Eq. 2.3. By applying an angle-dependent calibration measurement, the phase term $\psi(\theta) = 2\pi(d/\lambda)\sin(\theta)$ can be determined. All in all, we used a rotating structure that rotated the Radar around the center of the frontend and a corner reflector in the far field of the antenna array like in Fig. 3.3

Since the antenna array contains elements in two dimensions, we performed two calibration measurements, one for azimuth and one for elevation. The angular step size that we used in both cases was 1°. This way, we created two steering vectors $\mathbf{a}_{s_{az}}(\theta)$ and $\mathbf{a}_{s_{el}}(\theta)$ as in Eq. 2.3. In case that only one of the two dimensions is required then the steering matrix $\mathbf{a}_s$ used in 2.16 will be equal to one of the two steering vectors. For example, when the application layer needs to track people, then it is common practice to calculate the DoA in azimuth only. Whereas, when the application layer requires complex three-dimensional arm gestures, it is needed to calculate the DoA also in elevation.

In order to achieve that, the steering matrix must have as many columns as the azimuth-elevation combinations. Such a steering matrix could be calculated with a separate experiment that would rotate the Radar in all possible angles. However, we decided that this solution is not optimal since it would require almost three days, taking into consideration the restrictions of the setup. That is why, we used the existing two steering vectors and calculated an estimation of a steering vector for two-dimensional angle estimation. We calculate the outer product of the two steering vectors and then divide it with one of the two vectors in initial position, as in Eq. 3.1.

$$\mathbf{a}_{s_{2D}} = \frac{\mathbf{a}_{s_{az}}(\phi) \otimes \mathbf{a}_{s_{el}}(\theta)}{\mathbf{a}_{s_{az}}(0)} \tag{3.1}$$

The output $\mathbf{a}_{s_{2D}}$ is a three dimensional tensor and is not suitable yet for DoA calculation. In the last step, we reshape it so that it has $N_{VRx}$ rows and $N_\theta \cdot N_\phi$ columns; this can be used in Eq. 2.16 for DoA.

An unambiguous estimation of the angular position of the target is one of the fundamental requirements for solving a DoA problem. The term unambiguity describes the ability of the antenna array to uniquely distinguish the DoA of return signals. This is related to the well-established concept of grating lobes [Ric14]; a grating lobe in the receiving array beam-pattern causes an ambiguity in the angular estimation. In [EZO98] the authors formally introduce the ambiguity function as in Eq. 3.2.

$$\chi(\theta_i, \theta_j) = \frac{\mathbf{a}_s^H(\theta_i) \cdot \mathbf{a}_s(\theta_j)}{\|\mathbf{a}_s(\theta_i)\| \|\mathbf{a}_s(\theta_j)\|} \tag{3.2}$$

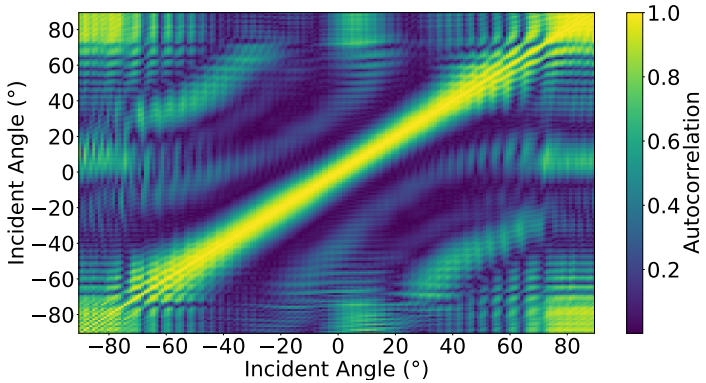



Figure 3.4: Ambiguity function in azimuth dimension, for an antenna array with two transmitters and four receivers (©IEEE [NHZ21b]).

The ambiguity function $\chi(\theta_i, \theta_j)$ refers to the autocorrelation of the receiving array steering vector calculated at the positions $\theta_i$ and $\theta_j$, respectively. An ambiguity takes place when it is not possible to distinguish between two directions. Fig. 3.4 and Fig. 3.5 show the ambiguity function for azimuth and elevation dimension respectively. The highest value of the autocorrelation has been achieved along the main diagonal only when $\theta = [-60°, 60°]$ in azimuth and $\theta = [-30°, 30°]$ in elevation. Therefore, there are no ambiguities for 120° and 60° in azimuth and elevation respectively. This was expected since the antenna array is not an ULA but a sparse array [LY11]. Finally, the 3dB beamwidth [Ric14] was calculated and was found to be 100° and 45° for azimuth and elevation respectively.

## 3.3 Modulation Parameters

The modulation that we used has a chirp duration for each transmitter of 66 µs, a bandwidth of 1.5 GHz and 64 transmitted chirps per transmitting antenna. We also added a pause of 50 µs after each chirp in order to satisfy maximum unambiguous velocity and velocity resolution that we believe are suitable for gesture recognition application. Therefore, one complete measurement frame

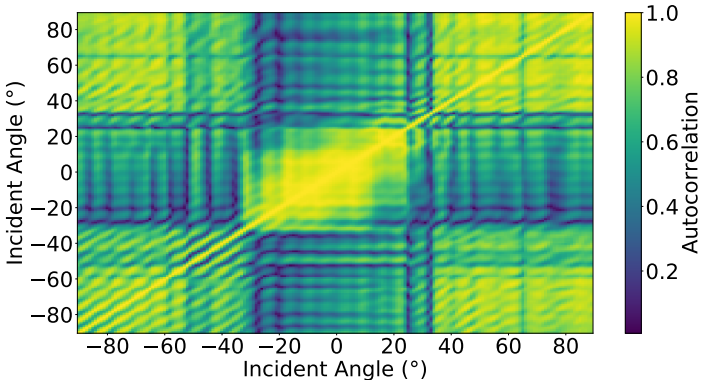



Figure 3.5: Ambiguity function in elevation dimension, for an antenna array with two transmitters and four receivers.

takes almost 15 ms. In order for the signal processing chain to run smoothly we added an extra 5 ms and the total sampling period reached 20 ms. Fig. 2.4 shows the TDM-FMCW that was used throughout this Thesis. The system parameters are summarized in Table 3.1.

## 3.4 Software Development

In the framework of this Thesis, I implemented the software framework that controls the Radar hardware, which was provided by the department. I relied heavily on Object Oriented Programming (OOP) since each hardware component could easily be characterized by certain attributes and methods that can be applied on it. Moreover, I used shallow inheritance interfaces and made more use of composition, in order to create classes that could be re-used in future projects.

The Radar hardware contains many integrated circuits that need to be properly "opened" and "closed". For example a Programmable Gain Amplifier (PGA), a Digital to Analog Converter (DAC), an ADC, etc. I created a class for each of them which inherits from a base class called *Device* and contains abstract

| Name | Value |
|---|---|
| Antenna Gain | 10 dBi |
| Transmit power | 12 dBm |
| FoV (az/el) | 120/60° |
| Antennas (receiving/transmitting) | 4/2 |
| Center Frequency | 76.75 GHz |
| Chirp duration | 66 µs |
| Chirps per Tx Antenna | 64 |
| Break between chirps | 50 µs |
| Bandwidth | 1.5 GHz |
| Measurement duration | 15 ms |
| Break between measurements | 5 ms |
| Total duration | 20 ms |
| Maximum unambiguous distance | 6.4 m |
| Range resolution | 0.1 m |
| Radial velocity resolution | 0.13 m/sec |
| Maximum unambiguous radial velocity | 4.2 m/sec |

Table 3.1: System parameters (©IEEE [NHZ21b]).

methods for opening and closing the device. Each subclass implements the abstract methods and if needed contains extra methods. For example, the *PGA* class contains setter/getter methods for adjusting the gain.

The complete Radar demonstrator consists of three different boards, namely the FPGA, the frontend and the baseband board. I developed a dedicated class for each of them, which is a subclass of the *Component* base class. The *Component* has abstract methods for powering up and down as well as for suspending and resuming operation. The *BasebandBoard* class is connected with the aforementioned ICs using composition and it delegates certain functionality to them (i.e., increase the gain of amplifier). Finally, the class *Radar* uses composition to connect to the aforementioned boards; when the user has a request, the class delegates it to the responsible board class. For example, when the user asks to

power up the Radar, it will delegate the call to all connected *Components*, in a specified sequence. Fig. 3.6 shows a simplified UML diagram of the software architecture of the Radar demonstrator.

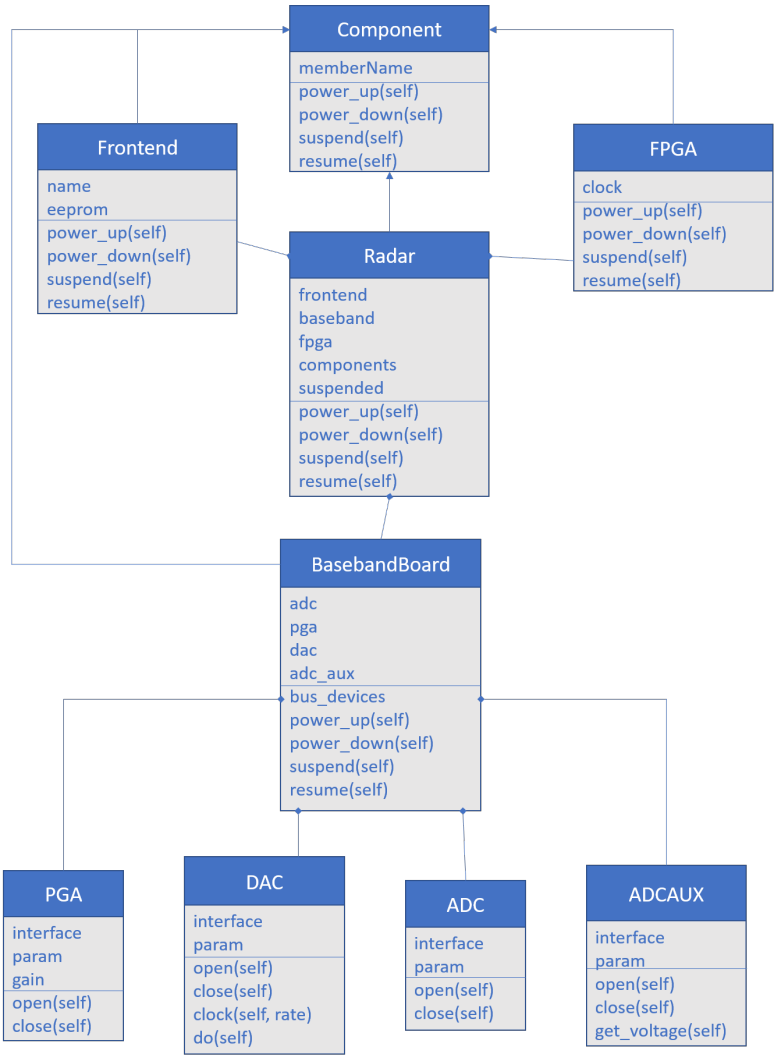



Figure 3.6: Software architecture of the Radar demonstrator. It consists of the *Frontend* class responsible for controlling the Radar chip, the *FPGA* for controlling the chirp sequences and data collection, and the *BasebandBoard* for controlling the analog parts. The classes *PGA*, *DAC*, *ADCAUX*, *ADC* are derived from the base class *Device*, but it is not mentioned in the diagram for easier understanding.

# 4 Macro Gesture Recognition

The analysis and results presented in this Chapter come from the author's publication [NHZ21b], alongside Jürgen Hasch and Thomas Zwick.

## 4.1 Introduction

Interaction with digital devices has been achieved with a variety of practices throughout the years, such as physical manipulation of proxy remote devices (e.g. wireless mouse) or direct physical interaction with the input device (e.g. touch screens) [GWL19]. Despite the ease that we have earned in using these devices through direct contact, there is an increasing need for contactless interaction via gesture recognition. Varying ways have been proposed, including the attachment of sensing devices on the hand. Such an approach is sub-optimal as it adds additional peripherals to the equation [LLC+19]. Recent developments in computer vision allowed the analysis and the identification of hand motion in real time, using cameras [MYG+16]. However, in such setups the system requires a significant amount of power, which often makes it unsuitable for mobile devices or for long-term usage. Since miniaturized low-power Radar sensors became affordable [HTS+12], there was a considerable interest in the development of consumer applications. Google Soli was the first project to demonstrate that hand gesture recognition is possible with such a technology [LGK+16].

Some studies [KT16, AZS18] tackle the above problem by using Continuous Wave (CW) Radars, which can measure the radial-velocity between sensor and moving arm, but not the distance. Others [LGK+16] make use of Frequency Modulated Continuous Wave (FMCW) systems, which additionally take advantage of the distance information.

The micro-Doppler (mD) effect is caused by signals returned from a target that incorporates vibrating or rotating structure [CLHW06]. A moving arm is such

a target and in many studies a mD spectrogram is utilized, since it represents a signature of the movement. In [LGK+16, LHYC18, AZS18, RJ19] empirical features are developed to extract as much information from the spectrogram as possible, including the length of a gesture, the ratio of negative to positive Doppler frequency, the bandwidth of mD frequency and the spectral power distribution. In [LZRG17], the authors make use of the fact that such a spectrogram is sparse and apply the Orthogonal Matching Pursuit (OMP) algorithm to extract features. Convolutional Neural Networks (CNN) are also widely used for the feature extraction of mD, reporting improved results [KT16, CML+19].

Other studies [WSL+16, ZTZ18, WWZ+19] use Range Doppler Maps (RDM), generated from FMCW Radars, which contain only spatial information of the movement. In these cases, many consecutive images need to be processed in order to get the temporal information. In these studies, a CNN is used for the feature extraction and Long short-term memory (LSTM) is utilized for modeling the dynamics of a gesture. This comes at a cost of running a machine learning network in every frame, as well as with a high memory footprint. However, it combines information from the range and the radial velocity of the targets, whereas during the generation of mD images, the range information is discarded.

Using the DoA of the detected targets is crucial [SFL+20]. In [CLFG] the authors developed a multi-static Radar with four receiving antennas that generated four mD images which were used by a novel CNN architecture. Since the antennas are placed a few cm apart, each spectrogram depends on the DoA, which is exploited by the classifier to improve the accuracy. Using DoA from collocated antennas is also possible, as shown by [SAHLE19]. The authors created a new heat-map with the same shape as the mD image but instead of the magnitude, they depict the DoA in each bin. Then, a custom CNN is used for feature extraction on the stacked images. Moreover, [CML+19, WJZ+19] created three feature maps from the Radar data, Range-Time, Doppler-Time and Angle-Time by integrating range, Doppler and DoA in the time domain respectively; these maps were simultaneously sent to a CNN for feature extraction. In [SFGP19] the authors also embedded DoA by creating two extra maps, one for elevation and one for azimuth. Initially, the authors calculated mD signatures by integrating the range dimension using multiple frames. Then, they calculated the phase differences for each measurement cycle, using two receiving antennas for each dimension. Moreover, they created images with the same shape as an mD

map by integrating the range dimension of the phase differences. Finally yet importantly, [SFL+20] combined five attributes of the $K$ most important bins of the Range Doppler map, in $L$ measurement cycles to generate a feature cube $(5, K, L)$. Two of the five attributes contained information from elevation and azimuth of the detected target. Thus, their classifier in the next stage could identify 3D gestures with a high accuracy.

Automatic detection of a gesture is a pivotal part of a real-time recognition system. [ZTZ18, WSL+16] achieved that by using a CNN combined with an LSTM. In [SFGP19] a Faster R-CNN object detection framework was used to identify the Region of Interest in the feature map that could contain a gesture and send it to the classifier. On the other hand, in [SFL+20, LPS+11] the power content of the returning signal in each frame was used to identify if a significant event took place.

The aforementioned work focused on the detection and identification of micro-gestures, which are small hand or finger movements performed only a few centimeters away from the sensing device. On the contrary, I developed a system that is capable of recognizing macro-gestures, so that a user could interact with the device using his/her arm at a distance of a few meters. Fig. 4.1 exhibits such a typical interaction setup. Furthermore, the proposed system does not use a CNN or LSTM, for inferring the gesture type, which usually require hardware accelerators, but relies on empirical feature extraction which can be implemented in an embedded system. Finally yet importantly, I evaluated the effect of certain system parameters in the average accuracy of the classifier.

The remainder of this chapter is structured as follows: Section 4.2 introduces the novel signal processing pipeline for gesture recognition. Section 4.3 explains the macro-gestures and Section 4.4 provides details on how the dataset was recorded. Finally, Section 4.5 presents the results of my method and compares it against the state-of-the-art. An introduction to Machine Learning (ML) is available in Appendix A.2 and detailed pictures of the gestures can be found in Appendix A.1.

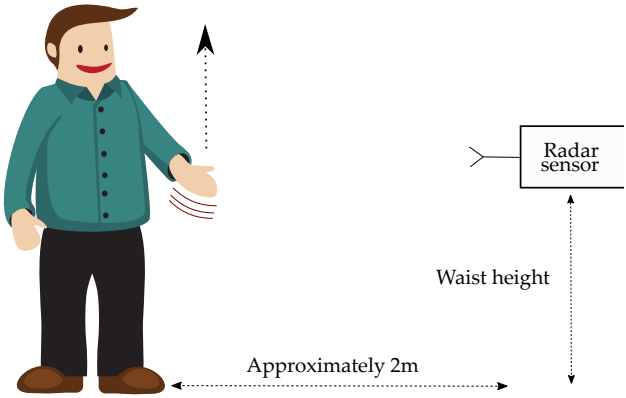



Figure 4.1: Macro-gesture example, “swipe-up” in front of the sensing device (©IEEE [NHZ21b]).

## 4.2 Processing Pipeline

This section describes the processing steps needed to transform time-domain raw Radar data in a form suitable for ML, the feature extraction method and the ML model. After I get the recorded signal from the ADC of receiving channels, I apply a 2D-FFT for calculating the RDM and then I estimate the noise floor using a CFAR algorithm, as explained in Section 2.4. In the next step, I replace the bins of RDM that do not exceed the noise floor, with zeros. Then, I calculate the mD vector by selecting the maximum power bin for every fixed radial velocity value along the range dimension. I chose not to integrate over range, as is usually the case in FMCW Radar, in order to be able to apply DBF in the selected bins. This way, I significantly reduce the processing time for each frame, since DBF is applied only to a few bins of the mD vector which are actually important for gesture recognition. The above procedure is repeated for each measurement frame and the mD vectors are concatenated so as to create the mD image, which is sparse due to the CFAR thresholding. This image constitutes the first out of the three feature maps that will be used in the next stages of the pipeline.

After that, I developed an algorithm which identifies if a significant event took place, in order to initiate the classification process. According to this algorithm,

if the maximum radial velocity in the mD image is higher than a given threshold for a certain amount of time, then an event is taking place. These two hyper-parameters depend on the event type that needs to be detected. For dynamic hand gestures I selected a velocity threshold equal to 0.8 $m/s$ and a minimum duration of 200 $ms$. The machine learning model expects input with a certain shape, that is why I set the total event duration as a third hyper-parameter equal to 50 frames. However, this raises a problem in case the user completes a gesture faster than the prefixed value, because she/he would have to wait until the remaining frames are recorded and sent to the classifier [SFL+20]. Hence the event detector also searches for the ending point of the gesture, by checking if the maximum velocity is below 0.8 $m/s$ for 200 $ms$. If the end of the gesture is detected, then the remaining frames of the event window are filled with zeros and sent to the next stage of the pipeline.

After event detection, I create two more feature-maps with the same shape as the mD image, whose amplitude refers to the DoA in azimuth and elevation respectively. Thus, in total three images transform the arm movement, as it was captured by the sensor, in a compact form that contains valuable information about the radial velocity and the DoA.

Next, I developed an empirical method that generates a one-dimensional signal from each feature-map. For the mD image I use the highest velocity of each frame and for the two DoA maps, I use the median angle of each frame. At this stage, I have three one-dimensional signals which contain information on the radial velocity and the DoA in azimuth and elevation. For simplifying the classification even further, I decided to extract certain features out of the three signals. These are the following:

1. Number of zero-crossings in radial velocity.
2. Arguments of maximum and minimum radial velocity.
3. Maximum and minimum radial velocity.
4. Maximum and minimum angle in azimuth and elevation.
5. Angle in azimuth and elevation when radial velocity reached its maximum and minimum value.

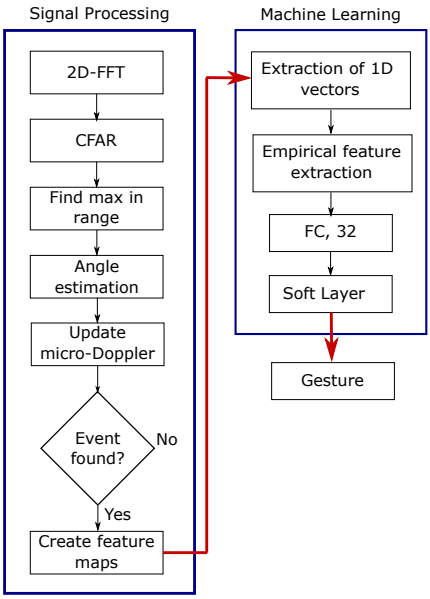



Figure 4.2: Signal processing chain for gesture recognition with mmWave sensor (©IEEE [NHZ21b]).

I use the aforementioned 13 values for training a Multi Layer Perceptron (MLP) with 32 neurons in the hidden layer [GBC16]. Fig. 4.2 shows a diagram with the proposed signal processing pipeline.

## 4.3 Gestures

Previous studies have focused on the detection of hand gestures conducted on top of a mmWave sensor at a distance of a few centimeters. This could be very useful for sensors integrated in a mobile phone, like the Google Pixel 4 [TWJ+21]. However, my goal is to simulate a smart home application in which a device is attached on a wall or on a TV. That is why during data collection, the Radar was facing the room and each subject was located at a distance approximately 2-3 meters from it. I found that a range higher than three

meters would deteriorate the results due to the low RCS of a human arm, the noise characteristics of the sensor and the large FoV in azimuth and elevation which decrease the gain. I selected ten gestures for the subjects to perform:

1. Random movement/walk
2. Pull
3. Push
4. Swipe up
5. Swipe down
6. Swipe right
7. Swipe left
8. Rotate
9. Wave
10. Push-pull

Figures that depict the gestures are available in Appendix A.1. I recruited ten subjects with various heights and ages to perform the above gestures in an intuitive manner. Each subject had to repeat each gesture 15 times; this way I collected $10 \times 10 \times 15 = 1500$ samples. In order to simulate a real scenario as good as possible, each subject conducted the gestures at three different aspect angles in relation to the device. The first location was in front of the sensor, the second around 20° to the left and the third around 20° to the right.

## 4.4 Dataset Collection

For the collection of the dataset, I developed a data logging application using the Bokeh library [Bok18]. When the subject conducts a gesture, the processing pipeline will detect it and plot the three feature maps. The administrator should either discard the gesture (in case that something went wrong) or select the gesture type that took place and save the data. In that case, the application saves

the raw Radar data and the processed feature maps, then the detection process is resumed.

## 4.5 Results

### 4.5.1 Automatic Gesture Detection

Fig. 4.3 shows an mD image with five gestures and the output of the event detection algorithm. The first and second are a “push” and “pull” respectively, for which the algorithm correctly finds the start and end point. The third gesture is a “push-pull”; the algorithm manages to identify that it is a single event. On the other hand, the fourth and fifth gestures have a small break between them and the algorithm managed to correctly separate them.

It is important to point out that I fine-tuned the gesture detection part, so that small movements would be identified and would trigger the classifier. The latter is responsible for discarding random motions by classifying them as "Random movement". If a gesture was not detected, then two main causes could have contributed. Either the user performed it in an unsuitable manner with very low or high velocity, or the SNR was not enough and CFAR failed to separate noise from useful targets. However, even if a poorly performed gesture was detected, the Machine Learning (ML) part would most likely provide inaccurate result, since it was not trained with similar samples.

### 4.5.2 Feature Maps

As already mentioned, feature maps are generated after an event is detected. Fig. 4.4 shows a typical example of the maps from a “pull” gesture, whereas Fig. 4.5 shows the ones from a “push”. Even for a naked eye it is easy to extract important features, like the fact that target velocity is always positive during the former and always negative during the latter. It should be noted that in both cases the DoA in azimuth and elevation does not show a significant change. However, things are quite different in Figures 4.6 and 4.7 which show the results from a “swipe left” and a “swipe right” gesture respectively. In these cases, radial

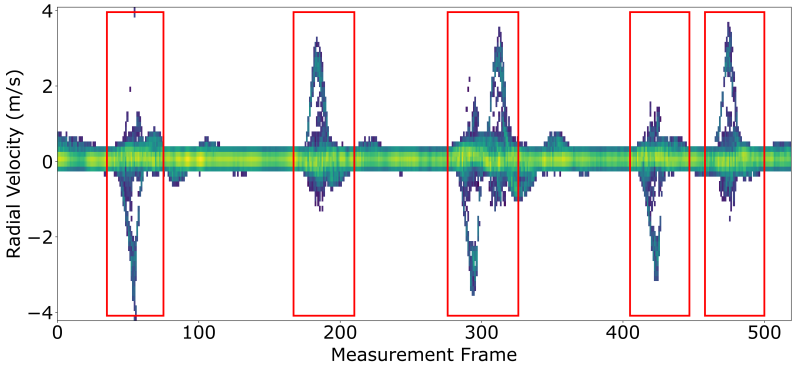



Figure 4.3: Micro Doppler stream with five different gestures, the red bounding box shows the start and end of the gesture as it was estimated by the event detection algorithm (©IEEE [NHZ21b]).

velocity changes sign and DoA in azimuth dimension has a significant change over time.

It is important to point out that when the subject performs a “swipe right” from a different position, for example 20° from bore-sight, then the radial velocity will not change sign and will look like a “pull”. In that case the azimuth feature map will be important for the classification since it will be able to capture positive or negative slope of the DoA, as shown in Fig. 4.8. In a similar fashion, the elevation feature map is important for the “swipe up” and “swipe down” gestures. Therefore, the above heat-maps contain the needed information to correctly classify hand gestures.

Fig. 4.9 shows the one-dimensional signal of the same “swipe right” as in Fig. 4.8, after applying the method explained above. The first subplot contains the envelope of the mD image and the second subplot the median angle of each measurement frame. The increase in azimuth from -20 to 5 is an important feature that can be used at the next stage for classification.

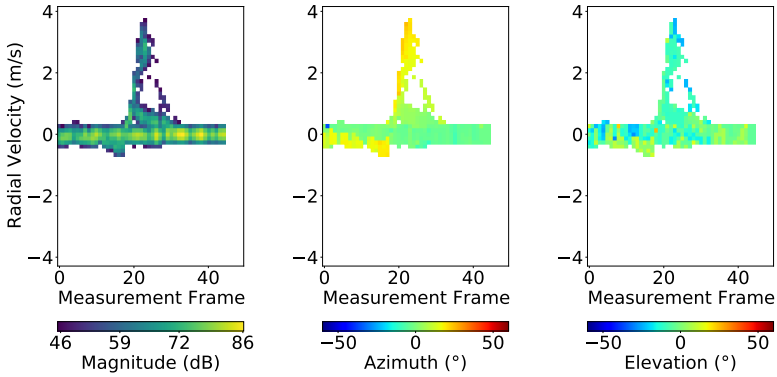



Figure 4.4: Feature maps generated from the DSP chain during a "pull" gesture. The first one refers to Micro-Doppler, the second one to DoA in azimuth and the last one in elevation (©IEEE [NHZ21b]).

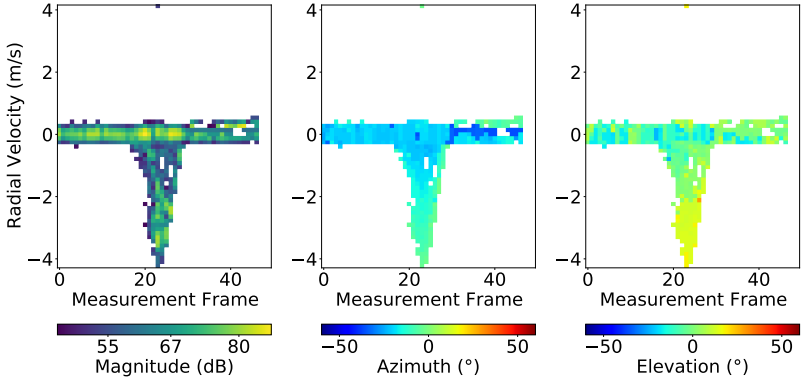



Figure 4.5: Feature maps generated from the DSP chain during a "push" gesture. The first one refers to Micro-Doppler, the second one to DoA in azimuth and the last one in elevation (©IEEE [NHZ21b]).

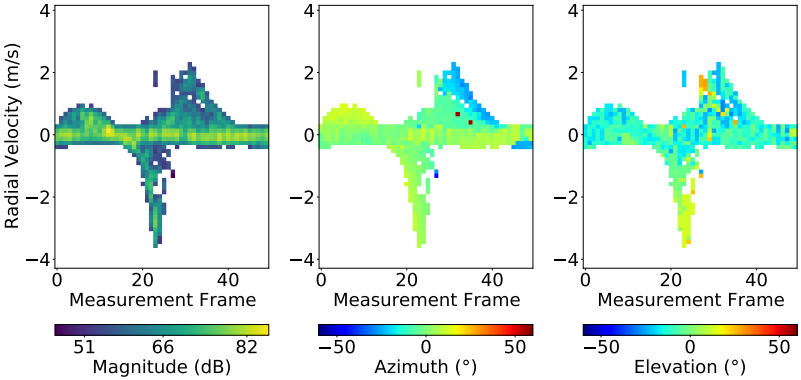



Figure 4.6: Feature maps generated from the DSP chain during a "swipe left" gesture. The first one refers to Micro-Doppler, the second one to DoA in azimuth and the last one in elevation (©IEEE [NHZ21b]).

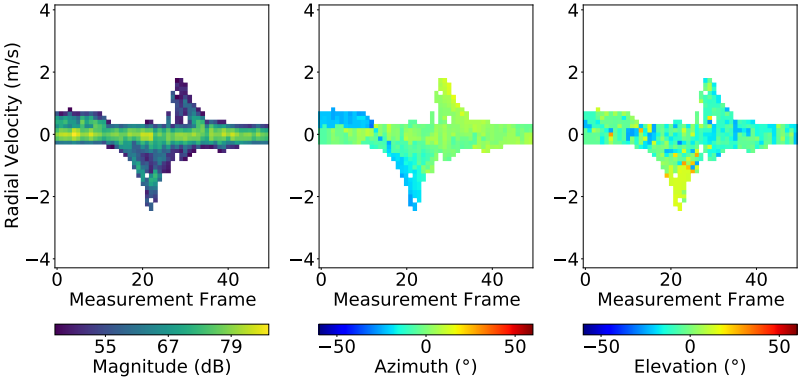



Figure 4.7: Feature maps generated from the DSP chain during a "swipe right" gesture. The first one refers to Micro-Doppler, the second one to DoA in azimuth and the last one in elevation (©IEEE [NHZ21b]).

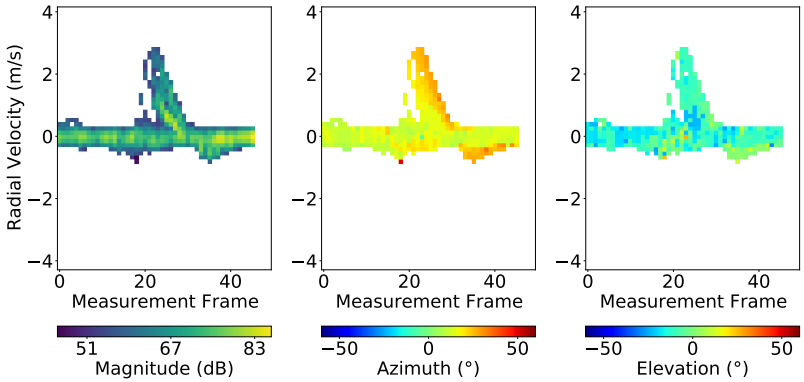



Figure 4.8: Feature maps generated from the DSP chain during a “swipe right” gesture, subject was located 30° from bore-sight. The first one refers to Micro-Doppler, the second one to DoA in azimuth and the last one in elevation (©IEEE [NHZ21b]).

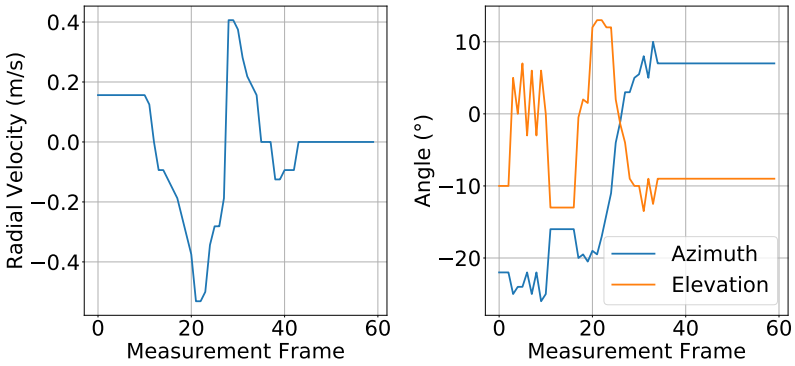



Figure 4.9: Time-series generated from the DSP chain during a “swipe right” gesture. The first subplot refers to Micro-Doppler and the second one to DoA in azimuth and elevation (©IEEE [NHZ21b]).

### 4.5.3 Supervised Learning

During training I split the dataset in two parts, train and test. The train set contains 80% of the samples and the test set 20%. In order to prevent the model from over-fitting, I used weight regularization and dropout [SHK+14, GBC16]. Fig. 4.10 shows the confusion matrix for the test set. The average accuracy is higher than 90% for all classes and 94.3% overall, which shows that the signal processing chain transforms the data coming from the mmWave sensor in a representation which is well-suited for training a MLP.

I also used a CNN architecture for extracting features from the feature-maps but I could not achieve better results in comparison to the empirical feature extraction approach. CNNs are the state-of-the-art approach for image recognition, given enough training samples. In my case, the number of samples is relatively low, but for series production like Google 4, thousands of participants could be used to collect millions of samples [LG20]. Then, a CNN would probably provide higher average accuracy.

In addition, I investigated the effect of certain system parameters in the classification average accuracy. In the first case, I evaluated the significance of high frame rate by using a 2x and 3x down-sampled dataset. Moreover, I assessed the importance of having two transmitting elements in comparison to one, which would reduce the amount of virtual antennas used for angle estimation by half. Last but not least, I combined the above cases and created a dataset that used one transmitting antenna and the frame rate was 2x down-sampled. Fig. 4.11 contains the confusion matrices for the above cases

It is worth pointing out that reducing the number of transmitters or decreasing the frame rate reduces the power consumption and lowers the hardware cost. Results showed that decreasing the frame rate 3x significantly decreased the average accuracy, which is a clear sign that a low frame rate cannot capture the dynamics of the arm movement. However, a system with frame rate of 25 Hz or one transmitting antenna achieves average accuracy above 90%. For certain consumer applications one could compromise with lower accuracy in order to achieve lower costs or lower power consumption. Table 4.1 summarizes the average accuracy that was achieved in the test set under different approaches.

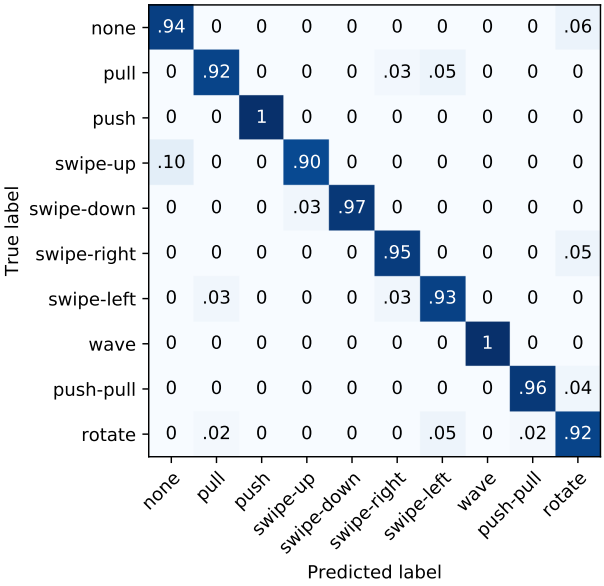
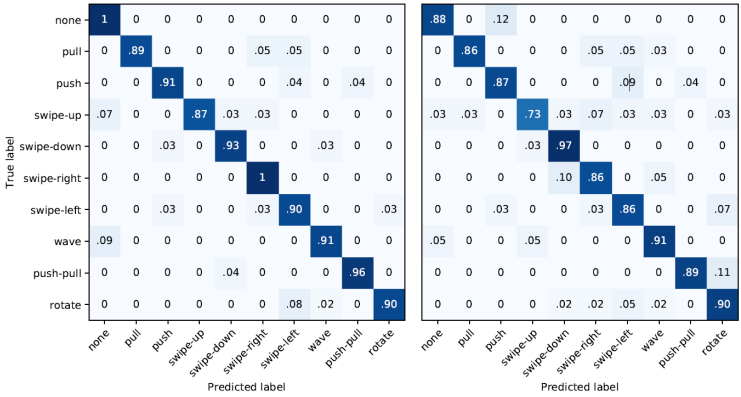


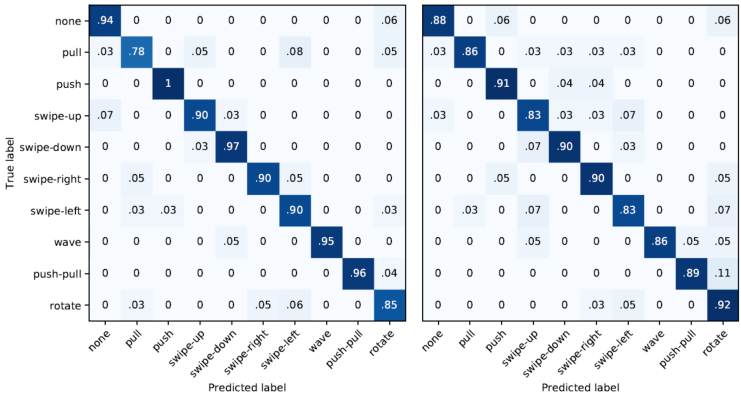

Figure 4.10: Confusion matrix with the original modulation. The accuracy for each class is higher than 90% and the average accuracy is 94.3% (©IEEE [NHZ21b]).

| Frame Rate (Hz) | Transmitters | Average Accuracy (%) |
|---|---|---|
| 50 | 2 | 94.3 |
| 25 | 2 | 92.0 |
| 17 | 2 | 87.6 |
| 50 | 1 | 90.3 |
| 25 | 1 | 88.3 |

Table 4.1: Average accuracy of the test set (©IEEE [NHZ21b]).




(a) On the left side, the frame rate was decreased 2x and the average accuracy reached 92%. Whereas on the right side the frame rate decreased 3x and the average accuracy reached 87.6%.




(b) On the left side, only one transmitting antenna was used and the average accuracy reached 90.3%. Whereas on the right side the frame rate decreased 2x and one transmitting antenna was used, the average accuracy reached 88.3%.

Figure 4.11: Confusion matrices for different modulation parameters and average accuracy of test set (©IEEE [NHZ21b]).

### 4.5.4 Deployment

For the real-time implementation, we used an Intel i7-8650U @1.9GHz. When a gesture is not detected, the median processing runtime is 10.1 ms and it includes 2D-FFT, CFAR, micro-Doppler, DoA estimation, and event detection. In case of a gesture detection runtime reaches 40.3 ms, which is higher than the frame period (i.e. 20 ms). In order for the system to achieve real-time performance, I discard the next frame that is waiting to be processed.

We also selected a Raspberry Pi 4 as target hardware to deploy the application, since it has low cost and capacity to run the Python package that I developed. Results showed that median processing runtime in case that an event is not detected is 31.5 ms. According to Table 4.1, if a frame rate equal to 25 Hz is selected, the average accuracy will drop by 2.3%. On the same time, the allowed processing runtime can be up to 40 ms, more than enough for the Raspberry Pi to handle it. In case that an event is detected and the gesture recognition pipeline is triggered, the median runtime reaches 67.4 ms and like before I discard the next frame, so that real-time performance is achieved.

In conclusion, if the designer is willing to decrease the average accuracy by 2.3%, my suggested method can be deployed in a low cost system. In case that a hardware accelerator is to be used, the low-level signal processing steps, which require more than 30 ms, could be transferred to it. Then a Raspberry Pi, or any similar processing unit, would be responsible only for the gesture recognition part; thus a frame rate of 50 Hz could be possible.

### 4.5.5 Comparison with state-of-the-art approach

In [SFL+20] a thorough comparison can be found between several machine learning approaches for gesture recognition with mmWave sensors. The “Multi-Feature encoder + CNN” yielded high average accuracy and on the same time has as significantly smaller memory footprint in comparison to the rest. That is why I consider this method as state-of-the-art and decided to implement it and train a model with the dataset that I collected.

In a nutshell, the algorithm sorts the bins of each RDM according to their amplitude, selects the first 25 and for each of them saves five attributes, amplitude, range, radial velocity, DoA in azimuth and elevation. If an event is detected, it

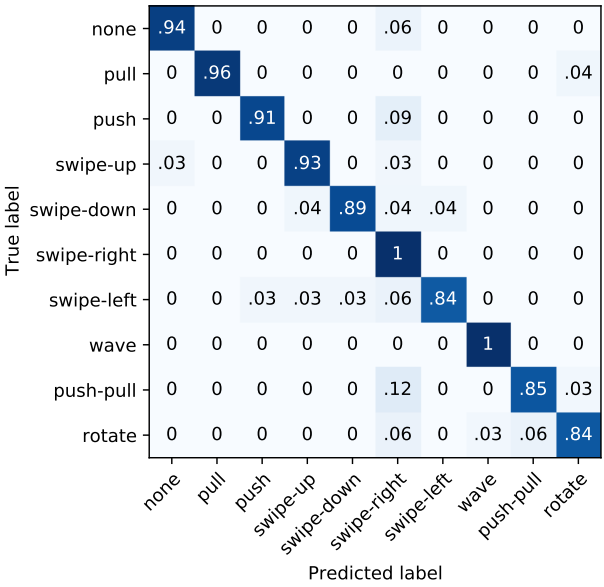



Figure 4.12: Confusion matrix for the feature encoder combined with CNN. The average accuracy in all classes is higher than 84% and 90.5% overall (©IEEE [NHZ21b]).

creates a tensor with the last 40 frames. The three-dimensional tensor is called feature cube and is used to train a CNN and an MLP. In order to make a fair comparison, I applied CFAR thresholding before selecting the bins, like in my pipeline. This way I made sure that only bins above the noise floor will be used for training the machine learning model. Also I used the last 50 frames like in my case and I used the same train and test set as with my DSP chain. Last but not least, I experimented with the number of bins used in each frame and found that 50 instead of 25 achieved the highest average accuracy in the test set, equal to 90.5%. Fig. 4.12 shows the confusion matrix for the different classes. Table 4.2 summarizes the comparison between my method and the one from [SFL+20]. It is important to mention that the number of floating point operations refer to the ML model. In my case, more calculations are needed for extracting the empirical features.

| | My method | Feature Encoder |
|---|---|---|
| Average Accuracy (%) | 94.3 | 90.5 |
| Memory Footprint (MB) | 0.015 | 35.7 |
| Floating point operations in ML model | 1602 | 18.3M |

Table 4.2: Comparison with state-of-the-art (©IEEE [NHZ21b]).

## 4.6 Concluding Remarks

I developed a real-time mmWave based macro-gesture recognition system using the experimental setup with four receiving and two transmitting antennas. I recorded a dataset with nine gestures from ten different subjects, which were positioned approximately two meters from the Radar in three different locations. The proposed signal processing chain and feature extraction method transforms ADC data from the sensor into feature maps, extracts empirical features in a two-step approach, and feeds the result to an MLP for classification. Furthermore, a custom made event detection algorithm detects if a gesture took place and only then activates the ML part. My method achieved 94.3% average accuracy. Following, I evaluated the effect of two system parameters, the frame rate and the antenna number, on the classification accuracy. I found that decreasing the frame rate two times or using one transmitting antenna does not significantly decrease the average accuracy, even though it can lower the cost of the sensor. Moreover, I deployed my method in a low-cost embedded system and found that a real-time performance is possible. Last but not least, I compared my method with the state-of-the-art approach and found out that my model achieved higher average accuracy in the test set, even though my model is significantly smaller.

# 5 Synthetic Dataset Generator

The analysis and results presented in this Chapter come from the author's publication [NHAZ21], alongside Jürgen Hasch, Mario Emilio Pizano Alvarez, and Thomas Zwick.

## 5.1 Introduction

Hand micro-gesture recognition has become available by using miniaturized, low-power Radar sensors, and was first demonstrated by the Soli project [LGK+16]. After that, many research groups focused on reproducing and improving the results, which required the collection of a significant amount of training samples. In [WSL+16], 2750 samples were recorded from 11 subjects, each performing 10 gestures 25 times. Similarly, in [SFL+20] the authors collected 7200 samples from 20 subjects which performed 12 gestures 30 times. In both cases, the authors recorded micro-gestures that were performed a few centimeters above the device. However, as I showed in Chapter 4, the subjects conducted gestures at an approximate distance of around 2 m from the device, not only bore-sight but also in various positions inside the FoV of the sensor. In total, I collected 1500 samples from 10 subjects that carried out 10 different gestures, including random movements which were regarded as noise.

Collecting this amount of data requires manual effort and is time-consuming. Moreover, in case that the sensor hardware (e.g., antenna configuration) or the modulation is modified, the measurements have to be repeated. Therefore, the need for a simulator that is able to generate artificial samples for various experimental cases arises. It is important to point out that in many other machine learning problems several dataset generators have been proposed [ALM11, PPVT19, GSN20, MBS+20].

The contribution of such a generator for the mmWave case could be threefold: it could first and foremost significantly reduce the experimental time for the data recordings, it could enable the testing of a broader spectrum of experimental cases, and finally, it could explore various options for the modulation and hardware parameters [IADW18]. As such, it could increase the variability of samples, especially in the case of macro-gestures during which the subject could be placed at different locations.

Many approaches have been presented on simulating human motion combined with mmWave sensing. In [THCD16] the authors used a kinematic model from [BTT90] to generate synthetic mD [CLHW06] spectrogram and to train a deep learning model. A similar application-specific approach was developed in [SSM+17] for simulating the reflections of cyclists. The authors manually created a model with point targets that represented various parts of the bicycle and the person riding it. In both approaches, only a specific kind of motion could be generated, limiting the number of use-cases that could be simulated.

In [OUY17] the authors surpassed this problem by using Blender [Com20], and its ray tracing capability for graphic simulation. More specifically, after designing a static scene, they used the rendered images "z-pass" and "combined pass", through which they calculated the distance, DoA and amplitude information of the object that was in each pixel of the images. The number of point targets was defined by the number of pixels of the rendered images, which can be set by the user. It is worth pointing out that the scene was static and only the Radar sensor was allowed to move.

In [IADW19], the authors created dynamic gestures by using another rendered image called "speed vector pass", which contains information about pixels moving in two dimensions. The authors were able to generate spectrogram from a synthetically generated human figure that was waving both hands. In [IADW18], Blender was again used to generate an arbitrary animation and created multiple variants via a Python script. Then, they extracted the position of the body joints, used them in a custom FMCW Radar simulator and generated a dataset with 2000 samples per gesture.

Finally yet importantly, in [GRC+19] another method was presented for generating synthetic Radar data for gesture recognition. The authors converted a Kinect dataset into Radar signatures using a simulation framework, then they extracted several features out of the spectrogram and used them to train an ML

model with four gestures. However, they did not test the accuracy of their model with a real Radar dataset; they only compared Radar data with the output of their simulator and found meaningful similarities.

In this chapter, I present a novel system which can generate synthetic Radar data for seven arm gestures. I used it to produce a dataset that contains 600 samples, with varying speed of execution and varying position of the animation relative to the Radar. Using the processing chain presented in Chapter 4 I generated three feature maps for each synthetic sample, extracted empirical features and trained a machine learning model. I tested the trained model using the real dataset that I collected from ten subjects who repeated the same gestures 15 times. The model yielded 84.2% accuracy in the real dataset, indicating a successful combination of the proposed simulator and feature extractor.

## 5.2 Synthetic Dataset Generation

The pipeline for generating a synthetic sample consists of four main parts, as illustrated in Fig. 5.1. First, a human model is selected and the animation is configured so that it moves its arm according to a specified gesture. Then, using Blender the animation is simulated and its point targets are extracted and used in the third step which simulates a Radar sensor. Finally, the pipeline from my previous work is used to transform time-domain data in a form suitable for ML.

### 5.2.1 Generation of Human-Body animations

Blender is an open source 3D creation suite, which supports the entirety of the 3D pipeline: modeling, rigging, animation, simulation, rendering, compositing, and motion tracking [Com20]. It offers a rich API for Python scripting that can be employed to customize the application and write specialized tools.

To create a Blender scene, it is necessary to first generate 3D models out of primitives (cubes, cylinders, spheres, etc.) by joining them or modifying their mesh. Meshes consist of vertices, edges and faces, and define the shape of a 3D model. Fig 5.2 shows an example of a mesh structure, derived from the documentation of Blender. It is also possible to import 3D models from libraries; Fig. 5.3 shows the one that we used, which we obtained from the TurboSquid

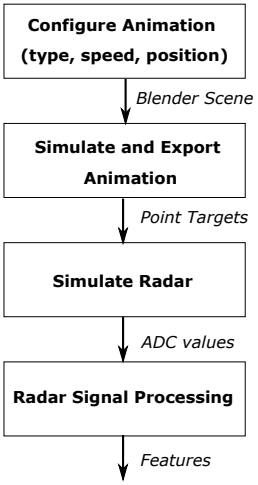



Figure 5.1: High level description of the sample generation process (©IEEE [NHAZ21]).

library. Models, as well as other Blender objects (e.g., camera, lights) can be arranged in the scene by modifying their location, rotation or scale.

In order to generate gestures, an armature has to be assigned to a model. Armatures include bones, which can be rotated to modify the shape of the mesh of a model or a 3D object. This way, the pose of a model can be selected according to the use-case. For defining a gesture, the user has to set key-points in the beginning and ending part of the motion, then Blender will calculate the positions of intermediate frames by solving the inverse kinematics problem. I created synthetic data for the gesture classes “push”, “pull”, “swipe-left”, “swipe-right”, “rotate”, “wave”, “push-pull”, similar to the ones defined in [NHZ21b]. Fig. 5.4 shows the model that was created, together with the armature, performing a “swipe-left”.

### 5.2.2 Extract Point Targets from Animation

After generating the animation that performs a gesture, the point-targets need to be extracted and given to the Radar simulator. These targets are actually the vertices of the body model. I use the API of Blender to access their position

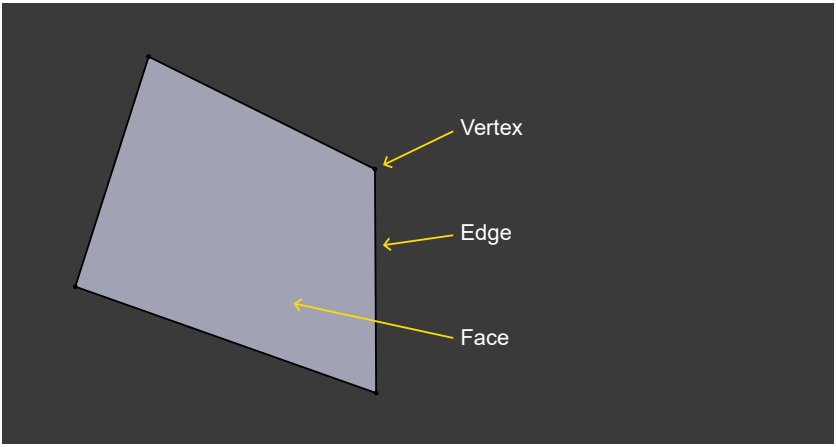



Figure 5.2: Example of mesh structure, Blender

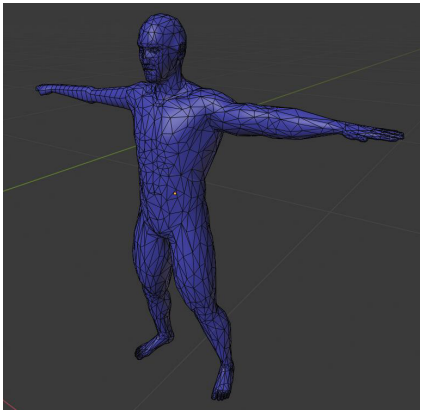


Figure 5.3: Body Model from TurboSquid library

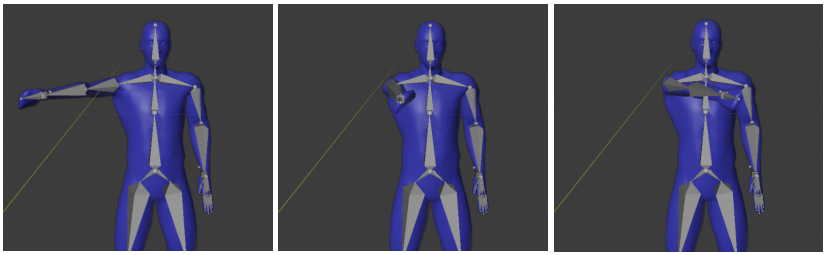


Figure 5.4: Human model with armature performing “swipe-left"gesture (©IEEE [NHAZ21]).

for all simulated frames, then I can easily calculate their velocity. After that, I apply ray casting to find the ones visible by the Radar at each frame. If the vertex is not visible at a particular frame, I set the RCS value to zero, so that it is not taken into consideration during the Radar simulation. With the above steps I managed to extract a tensor with dimensions: number of vertices $\times$ number of frames. Each element of the tensor contains position, velocity and RCS of the vertex. Fig. 5.5 shows an overview of the extraction process.

### 5.2.3 Radar Simulator

The simulator utilizes a monostatic Radar, with multiple $Rx$ and $Tx$ antennas. It generates the output of a Radar sensor, given the location, velocity and RCS of point targets that are provided as input. The first step is to calculate the travel time of the wave from the transmitting antenna to the target and back to the receiving antenna. Then I use Eq. 2.5 to calculate the attenuation of the wave. In the next step, I use Eq. 2.9 and 2.10 to calculate the time domain transmitted and received signal for one chirp. By mixing the signals the baseband signal is calculated, which is what a Radar sensor would record using its ADC. I repeat this procedure for multiple chirps in order to achieve the chirp sequence waveform like in Section 2.2.

The user can configure the simulator in order to approximately mimic the hardware. For example, the user can provide values for the Noise Figure (NF) and gain of the Low Noise Amplifier (LNA). Moreover, an input matrix defines the antenna gain for a discrete set of DoA. This way, the return signals of the

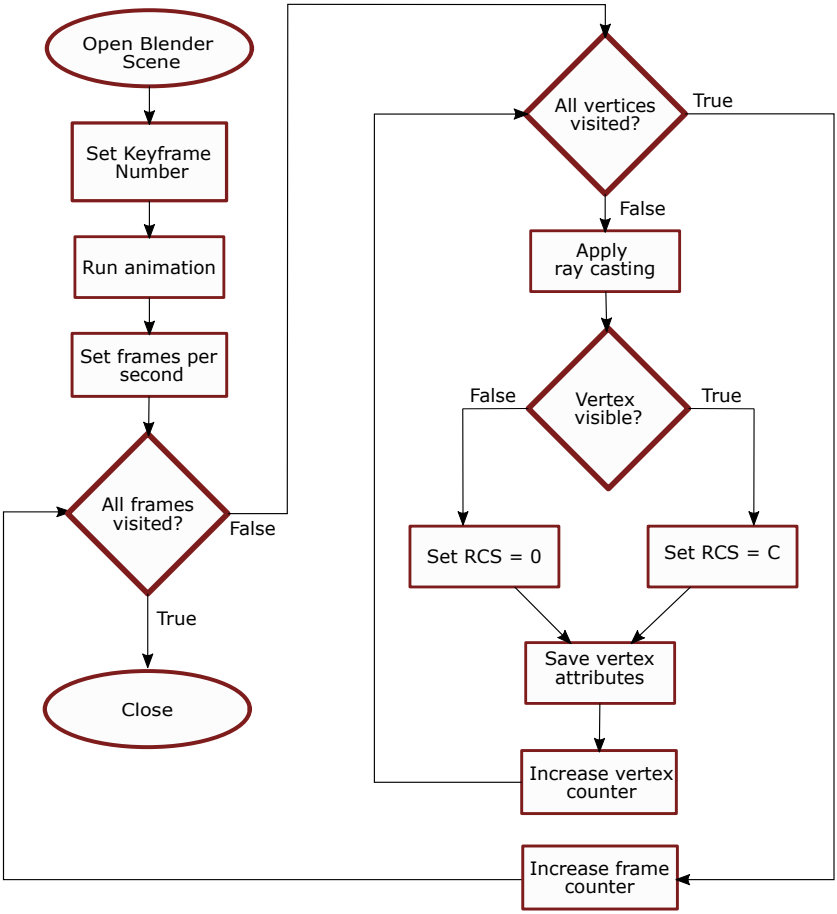



Figure 5.5: Overview of the pipeline for extracting targets from a Blender animation (©IEEE [NHAZ21]).

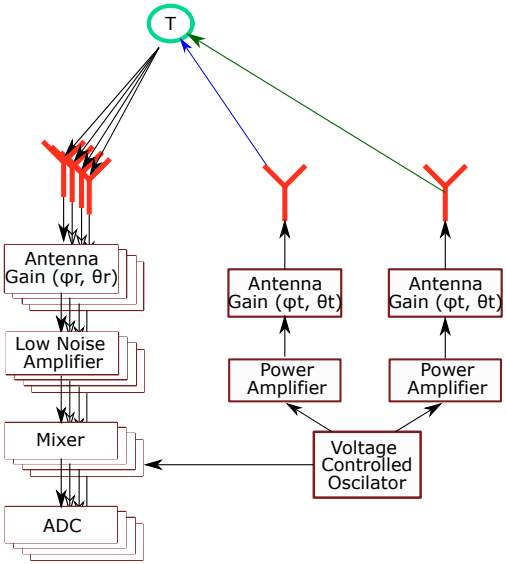



Figure 5.6: Flowchart of the Radar simulator with two transmitting, two receiving antennas and one point-target (depicted by 'T'). The signal source will be amplified and the antenna gain will be calculated based on the target's location. Similarly, in the receiving side, the antenna gain will be calculated for each target, the returned signals will be amplified and sent to the mixer for calculating the baseband signal (©IEEE [NHAZ21]).

animation will depend on its aspect angle. Last but not least, the simulator can also handle TDM-MIMO scheme for improved DoA estimation. Fig. 5.6 shows a flowchart with the basic blocks of the simulator for a 2 $Tx$ - 2 $Rx$ configuration.

## 5.3 Results

Through the synthetic data generation I wanted to create samples that would capture many corner-cases of a real scenario. That is why I created 60 samples for each human animation, using different configurations. The parameters that I can modify are the position of the Radar and the speed of execution. In addition, I found that some subjects performed the "push", "pull" and "rotate" with high variability, that is why I created two different animations for each of the above

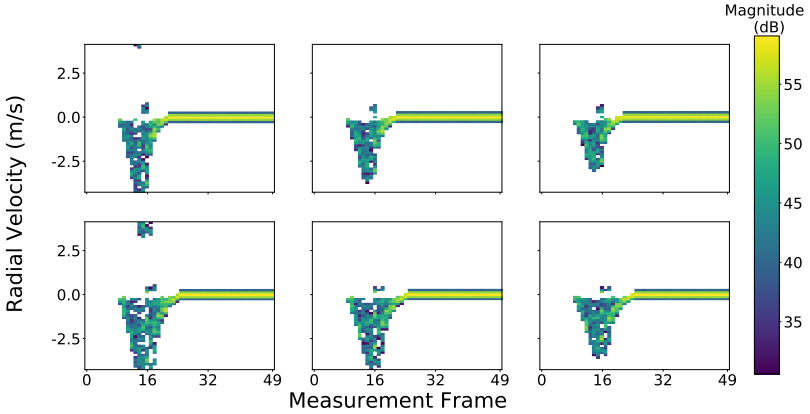



Figure 5.7: Examples of synthetically generated "push" gesture with different motion characteristics. In the first row, the motion has a duration of 13 frames, whereas in the second 16. The columns contain samples with varying maximum velocity (©IEEE [NHAZ21]).

three gestures. In total, I created ten animations, through which I generated 600 samples.

Typical examples of the variability that was achieved are shown in Fig. 5.7. I modified the duration of motion from 13 to 16 frames and I also varied by 40% the speed of the arm gesture.

### 5.3.1 Feature Maps

In this subsection, I present the generated feature maps for a few typical gestures. The first row contains the results using real hardware and the second row using the simulator. Columns correspond to the mD, and the DoA in azimuth and elevation respectively.

The magnitude of the mD for the synthetically generated samples is significantly lower in comparison to the real samples. It is important to point out that on purpose I did not want to fine-tune the simulator's parameters to match the power of the real gestures (e.g., RCS, transmitted power, noise figure), in order to show that the ML pipeline can extract meaningful features without taking into

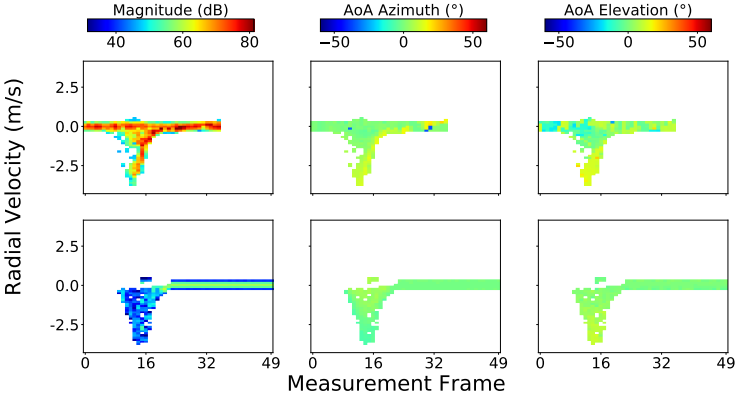



Figure 5.8: Feature maps of a "push" gesture as captured by Radar sensor in the first row and generated by simulator in the second (©IEEE [NHAZ21]).

consideration the absolute power value. In other words, the gesture recognition pipeline that I use remains invariant to the absolute power value, and as such can generalize across different levels of Radar noise and RCS.

In Fig. 5.8 the feature maps correspond to the "push" gesture. Fig. 5.9 depicts the mD maps of a "swipe-left", that is why in the Azimuth Feature Map, the DoA slowly decreases. Similarly, Fig. 5.10 depicts a "swipe-right", consequently the DoA in the Azimuth Feature Map increases.

The result in Fig. 5.11 shows the effect of the subject's position relative to the Radar. The mD map looks similar to a "push" gesture, since in this case the hand of the user always has negative radial velocity. However, the Azimuth Feature Map captures the variation and that information is propagated to the ML model.

### 5.3.2 Supervised Learning

As already mentioned, I used the synthetically generated dataset and the signal processing pipeline to train a classifier and I tested it on the real dataset that I collected. The average accuracy in the test set was 84.2%, Fig 5.12 depicts

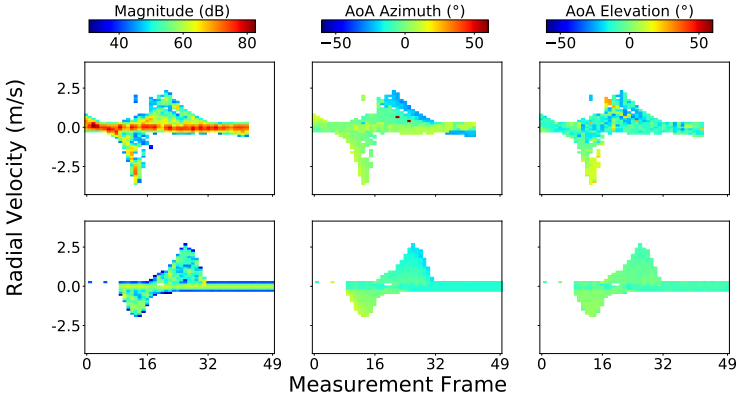



Figure 5.9: Feature maps of a "swipe-left" gesture as captured by Radar sensor in the first row and generated by simulator in the second (©IEEE [NHAZ21]).

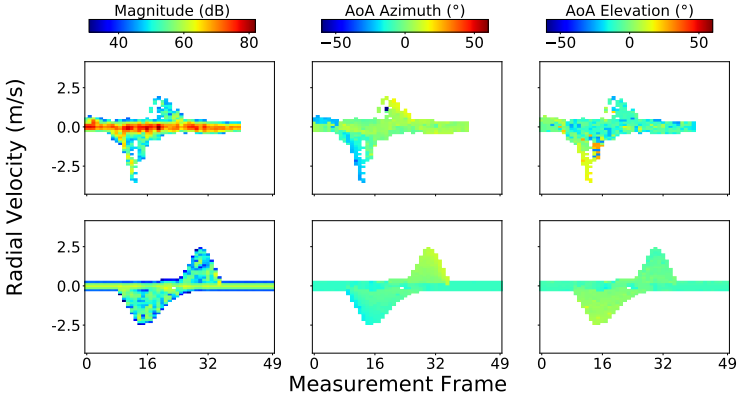



Figure 5.10: Feature maps of a "swipe-right" gesture as captured by Radar sensor in the first row and generated by simulator in the second (©IEEE [NHAZ21]).

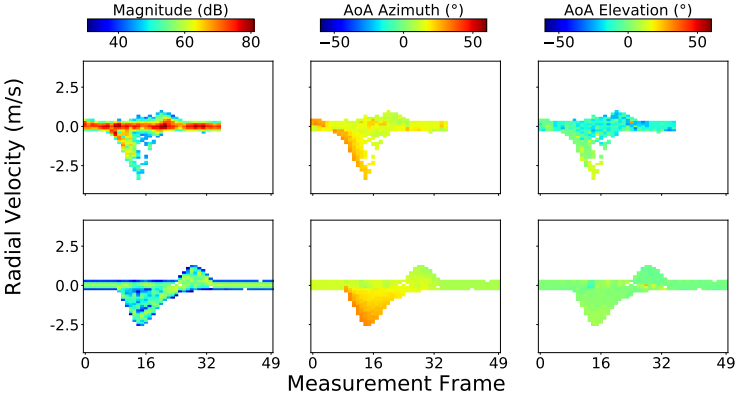



Figure 5.11: Feature maps of a "swipe-left" gesture as captured by Radar sensor in the first row and generated by simulator in the second. Subject was positioned on the right side of the Radar, which had a significant effect on the mD map (©IEEE [NHAZ21]).

the confusion matrix of the test set. In addition, Table 5.1 provides details of the classification results for each class. This indicates that the synthetic dataset generator can be used for pre-training an ML model and could be very helpful for capturing certain known corner cases. Then the model can be further fine-tuned with a dataset collected using the Radar sensor. Table 5.2 summarizes the proportion of the synthetic and real samples as well as the overall accuracy.

| Gesture | # Samples | Precision | Recall |
|---|---|---|---|
| Pull | 120 | 0.9 | 0.88 |
| Push | 120 | 0.83 | 0.69 |
| Swipe-Right | 60 | 0.68 | 0.93 |
| Swipe-Left | 60 | 0.83 | 0.76 |
| Wave | 60 | 0.99 | 0.90 |
| Push-Pull | 60 | 0.72 | 0.99 |
| Rotate | 120 | 0.97 | 0.79 |

Table 5.1: Detailed results of each gesture from supervised learning (©IEEE [NHAZ21]).

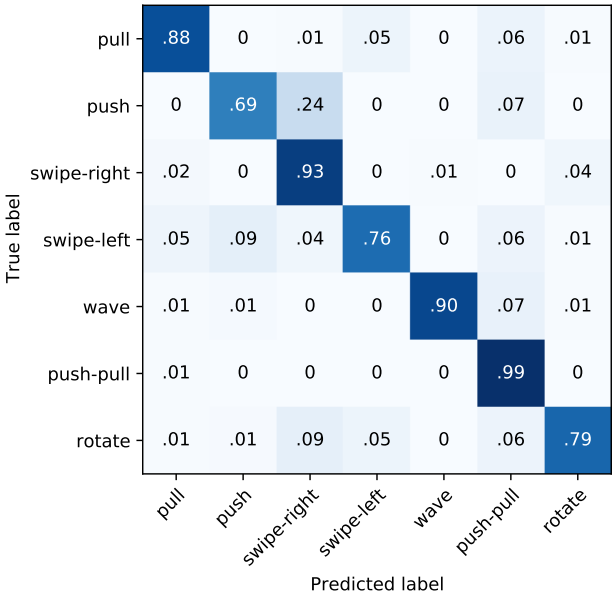



Figure 5.12: Confusion Matrix of test set (©IEEE [NHAZ21]).

| # Gestures | # Synthetically Generated Samples | # Real Samples | Test Set Average Accuracy |
|---|---|---|---|
| 7 | 600 | 1050 | 84.2% |

Table 5.2: Attributes of the datasets and result of supervised learning (©IEEE [NHAZ21]).

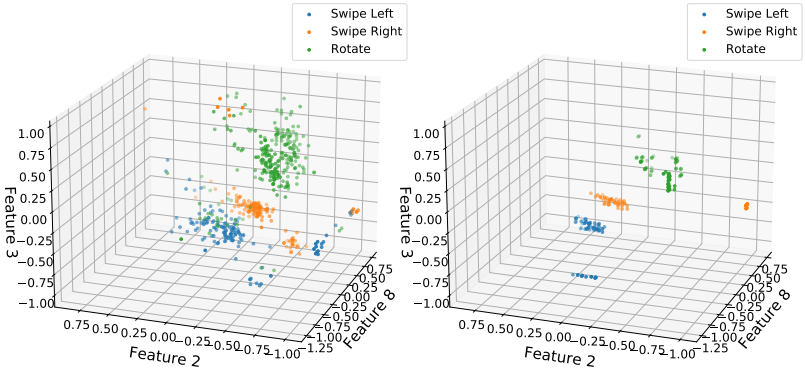



Figure 5.13: Scatter plot with three classes. The left subplot contains results from measurements and right subplot from simulation (©IEEE [NHAZ21]).

### 5.3.3 Empirical Feature Extraction

As a final step for comparing the synthetic and the real dataset, I compared the features that are generated by the processing pipeline in both cases. First, I created three-dimensional scatter plots, where the left subplot shows real, and the right shows synthetic data. Fig. 5.13 shows samples from gestures “swipe-left”, “swipe-right” and “rotate” using the features 2, 3, and 8. Likewise, Fig. 5.14 shows samples from gestures “pull”, “push”, “push-pull” and “wave” using features 1, 4, and 5.

In addition, I performed the Kruskal-Wallis H-test [KW52], to test the null hypothesis that the population median of the synthetic and real data are equal

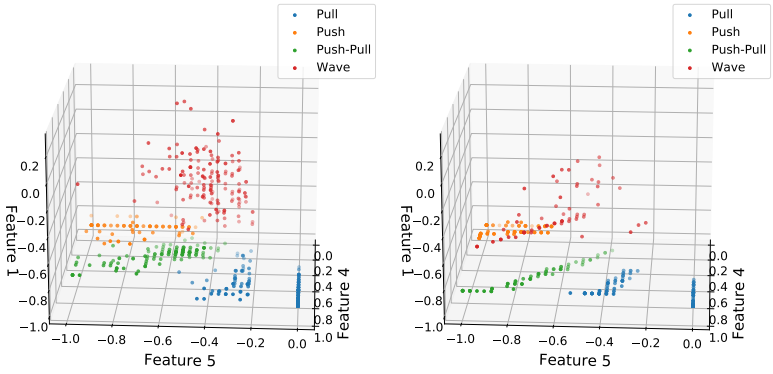



Figure 5.14: Scatter plot with four classes. The left subplot contains results from measurements and right subplot from simulation (©IEEE [NHAZ21]).

for each feature. When the p-value is below the significance level 0.05, then the test rejects the null hypothesis. In other words, there is not enough evidence to suggest that the samples come from the same distribution, therefore it is concluded that they come from different ones. The reason why I selected a Kruskal-Wallis test instead of ANOVA test is that the former is a non-parametric test that does not assume normality (a Shapiro [SW65] test on the data indicated that the features do not come from a normal distribution). Fig. 5.15 shows the result of the Kruskal-Wallis test for all combinations of gestures and features. In this intra-feature level, the majority of features appear to come from different distributions for the real and synthetic case.

Nevertheless, since the classifier uses non-linear combinations of the features, it is not the absolute value of each feature that defines the result, rather the offset among their distributions. To visualize that, Fig. 5.16 shows that the pairs (real - first row, synthetic- bottom row) of distributions of each feature are very close, relative to the others. To also test this beyond the apparent visual inspection, I performed a Wilcoxon non-parametric paired test [Wil45] for the medians of the distribution of each feature, for the two cases. I did so for each of the seven gestures. The test accepted the null hypothesis that the two (synthetic medians and real medians) come from the same distribution in all the gestures (p-value

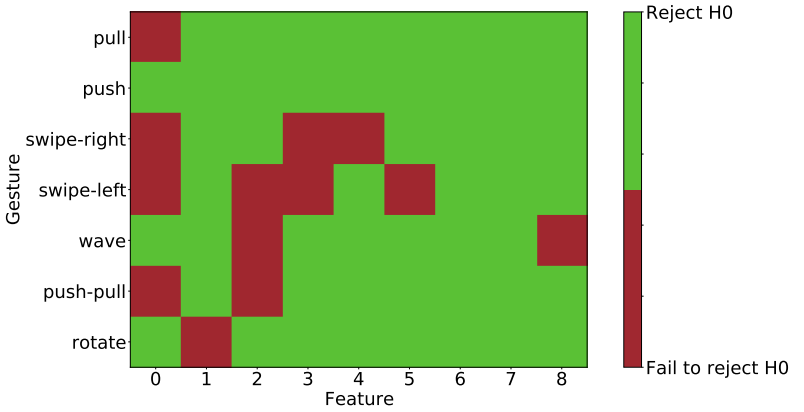



Figure 5.15: Result of Kruskal-Wallis H-test. Green color indicates p-values $< 0.05$ (i.e. real and synthetic data come from different distributions), and red indicates p-values $> 0.05$. The Y-axis shows the different gesture types and the X-axis shows the seven features used in the classifier (©IEEE [NHAZ21]).

| - | Pull | Push | Swipe-Right | Swipe-Left | Wave | Push-Pull | Rotate |
|---|---|---|---|---|---|---|---|
| p-value | 0.30 | 0.35 | 0.83 | 0.35 | 0.67 | 0.87 | 0.36 |

Table 5.3: Result of the Wilcoxon test for the medians of the distribution of each feature. The test accepts the hypothesis that the two feature sets come from the same distribution (©IEEE [NHAZ21]).

$> 0.3$), as it can be seen in Table 5.3, which supports what can be seen in the box-plot figure.

Finally, as a sanity check, I wanted to evaluate whether this difference in the intra-feature distribution of the real and synthetic data that was pointed out by the Kruskall-Wallis test plays a significant role in the supervised learning. To do so, I calculated the average accuracy of the classifier after I excluded the two features (number 6 and 7 at the Fig. 5.16) that had no significant p-value (i.e. whose real and synthetic distribution was completely different) for all gesture types. The classifier yielded 74% compared to the original 84.2%. This again

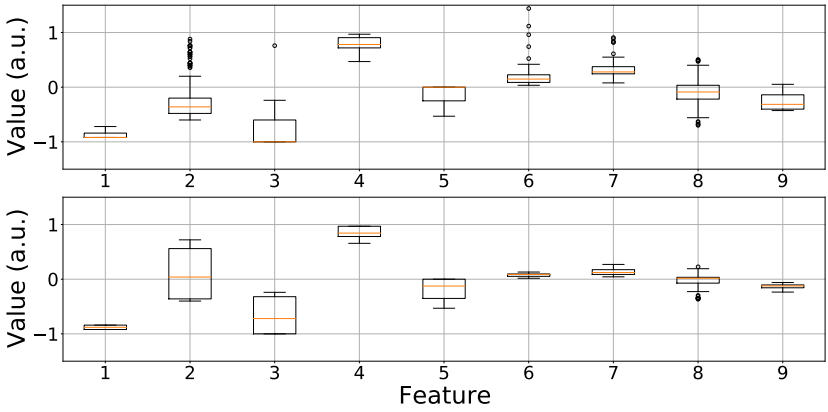



Figure 5.16: Box-plot for each feature of the "pull" gesture. The first row contains samples collected from the Radar and the second row from the simulator (©IEEE [NHAZ21]).

indicates that the classifier uses non-linear combinations of the input features, and as such it remains invariant to absolute variations of the median of each single feature.

### 5.3.4 Distinction from related work

Previous work managed to simulate human motion and combine it with a Radar simulator, but did not test a classifier trained with synthetic samples on real data. As a result, a quantitative comparison with a known benchmark is not possible.

I tried to use methods suggested in literature for exporting point-targets from animations but without success. The authors of [OUY17] published source code, which is only suitable for static scenes. Furthermore, I managed to reproduce the method explained in [IADW19] for exporting moving point-targets from Blender using "speed vector pass". However, the point-target information was accurate enough only for 2D movements but not for complex 3D motions.

## 5.4 Concluding Remarks

In this chapter, I presented a method to generate synthetic datasets for arm movements, and I used it for training an ML model for gesture recognition with mmWave technology. The performance of the model was evaluated on real data, which I had already collected using an experimental sensor, yielding an average accuracy of 84.2%. Therefore, I demonstrated how a novel data-generator like the one I presented here can contribute in the pre-training phase of a model, as well as for capturing corner cases related to the speed of execution and the position of the subject, that are difficult to reproduce during data collection. To the best of my knowledge, this is the first time that a model trained with synthetic Radar data is tested on real Radar data and achieves such high accuracy.

# 6 People Tracking

The analysis and results presented in this Chapter come from the author's publication [NHHZ22], alongside Jürgen Hasch, Michael Heizmann, and Thomas Zwick.

## 6.1 Introduction

People tracking with high resolution Radar sensors has been demonstrated in research as well as in commercial products. There are two major families of algorithms; [Ins18, MCN14] use the "Group Tracker" (the name is defined in [Ins18]) whereas [PMR21, WFS17, ZLW+19] follow another approach that I named "Cluster First Track Later". After thoroughly experimenting with both methods, I found that in certain scenarios they do not offer a robust solution. That is why I defined my own approach which is heavily based on the "Group Tracker" but also uses a clustering part; therefore, I named it "Group Tracker with Clustering". In the next sections, I will thoroughly describe all three methods and compare them under different scenarios. Finally, I will show that my approach outperforms the current state of the art in complex scenarios.

## 6.2 Signal Processing Pipeline

The signal processing can be broken down into two main parts. The first is related to the baseband processing of raw data and has been presented in Section 2.4. The second part is all about tracking and can be broken into the Kalman Filter (KF), track management, and tracking for extended targets.

### 6.2.1 Kalman Filter

In the simple case of a single point target, after each measurement frame is completed, an estimate is available with its distance, radial velocity and DoA, for the local coordinate system of the sensor. However, this estimate contains uncertainty since it is corrupted by noise. By collecting a series of measurements over time and combining these with an a priori known kinematics model, it is possible to improve the accuracy. The KF is a popular and well-studied way to achieve that. It deals effectively with the uncertainty due to noisy sensor data and, to some extent, with random external factors. The filter produces an estimate of the state of the system as a weighted average of the system's predicted state and of the new measurement.

The system to be observed must fit the below model for each time step $k$:

$$\vec{x_{k+1}} = F_k\vec{x_k} + B_k\vec{u_k} + \vec{w_k} \tag{6.1}$$

$$\vec{z_k} = H_k\vec{x_k} + \vec{v_k} \tag{6.2}$$

$F_k$ is known as the state transition matrix, $\vec{x_k}$ is the state vector, $B_k$ describes the input model, $\vec{u_k}$ is in the input vector, and $\vec{w_k}$ is the process noise, assumed to be drawn from zero mean multivariate normal distribution, with covariance $Q_k$. For the measurement equation, $H_k$ transforms filter's state to the observation $\vec{z_k}$, $\vec{v_k}$ represents the measurement noise, assumed to be drawn from zero mean multivariate normal distribution, with covariance $R_k$.

The processing of the filter consists of two steps, namely predict and update. In the former, a state estimate is calculated using the kinematics model and the pre-computed noise covariance. The update step will calculate the residual, using the new measurement and the state estimate. In addition, it will calculate the Kalman gain $Kg$ and finally update the state estimate. A more rigorous explanation can be found in [Ric14].

Designing the KF requires the definition of the state and measurement vector, the corresponding noise covariance matrices $Q$ and $R$, the transition matrix $F$ and the observation matrix $H$. In this work, I use a linear KF like in [Wag18].

The state vector contains the position and velocity in two dimensions whereas the measurement vector contains the position:

$$\vec{x} = [pos_x, pos_y, vel_x, vel_y] \tag{6.3}$$

$$\vec{z} = [pos_x, pos_y] \tag{6.4}$$

Radars do not directly estimate the position in Cartesian space, but with the pre-processing step of Eq. 6.5 it is possible to calculate it using the distance and DoA in azimuth. This technique is called Converted Measurements Kalman Filter (CMKF)

$$\vec{z_k} = [r_k \sin(\theta_k), r_k \cos(\theta_k)] \tag{6.5}$$

### 6.2.2 Tracking Extended Targets

Fig. 6.1 shows a flowchart that explains the two state of the art approaches for tracking people, "Cluster First Track Later" and "Group Tracker". In addition, it presents my own approach, in which I combine characteristics from the existing methods; I named it Group Tracker with Clustering.

For all methods, I will assume that at timestamp $k$ there are already a few tracks available and the system will update their state based on a new Radar measurement. During the first measurement, previous tracks are not available, in that case density based clustering can be used for the initial guess of the clusters.

#### Cluster First Track Later

Initially, the prediction step of the KF is applied. In other words, the kinematics model and the state of the filter up to timestamp $k-1$ are used to predict the location of the existing tracks at timestamp $k$. In the next step, a density-based clustering algorithm (e.g. DBSCAN [EKSX96]) is applied to find clusters with high concentrations of point-targets. A common approach is to use Cartesian two-dimensional space, but some researchers also use the radial-velocity and search for clusters in a three-dimensional space. In addition, some researchers use improved versions of DBSCAN that exploit the fact that the Radar targets

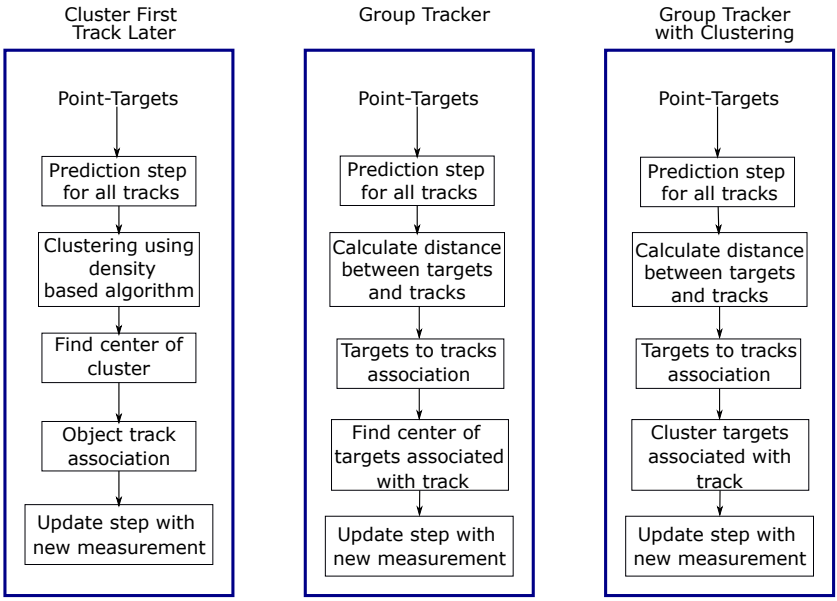



Figure 6.1: Tracking methods for high resolution Radar sensors (©IEEE [NHHZ22]).

can only be in a grid (distance, DoA, radial velocity) [LSF+18]. Finally, some researchers exploit the polar coordinate system of a Radar sensor [WFS15], since the same object generates many point-targets when it is located close to the Radar [WFS15].

When the second step is completed, the large number of point-targets will be mapped to a significantly lower number of clusters. The centroid of each cluster is calculated by a weighted average of the position of the targets belonging to the cluster; the magnitude of the target is used as weight. Thus, targets with strong return signals will have a higher effect in calculating the centroid of the cluster.

During the fourth step, an association algorithm is used (e.g., Hungarian algorithm [Kuh05]) to assign new clusters to existing tracks. If a cluster is not assigned to an existing track, then a new track is initiated; a task handled by the track management system. Finally, the state of the KF for all the tracks is updated using the new information.

A typical problem that rises with this method is when two tracks come very close to each other. For example, when two people shake their hands, hug, or when a person comes close to a piece of furniture. In that case, the density based algorithm will consider that the point-targets in the area of the two objects belong to one cluster and then assign it to the closest track. The other track will not be updated for a number of frames and might be deleted by the track management system. In [WFS18] the authors acknowledged the problem and suggested a solution that uses Ordering Points To Identify the Clustering Structure (OPTICS) [ABKS99]. They provide an algorithm that detects when two tracks have merged in one cluster and then utilizes the hierarchical cluster information to split the cluster into two smaller ones. The authors claim that the OPTICS algorithm has a computational demand slightly higher than DBSCAN.

In my evaluations I found that it actually needs significantly higher runtime and is not optimal for real-time systems. That is why I used the main concept of their pipeline which is to identify when two tracks have merged and in that case use K-Means [Llo82] to find two clusters within the merged cluster. This generated similar results and was faster. However, I believe that the concept of clustering first and then trying to identify "track merge" will not be robust in cases when a large number of tracks come close to each other.

#### Group Tracker

The second popular approach does not rely on a clustering algorithm to find dense regions in the point-targets. Like in the previous method, the prediction step of the KF is applied, for all tracks, using prior information. Then for each existing track $i$ and for all available point-targets $j$ obtained at time $k$, their distances are calculated (using Euclidean or Mahalanobis function). The distance represents the amount of innovation the new measurement adds to an existing track. The amount of innovation that is acceptable depends on the gate size. In other words, point-targets that do not satisfy a distance criterion will not be considered as possible candidates for assignment to the Track under Test (TuT).

After gating and calculating the distances, the point-targets will be assigned to the tracks with the lowest distance. Then, for each TuT a centroid will be calculated using the new point-targets that have been assigned to it. The targets

that are not assigned will be processed by a density based algorithm, in order to find if new objects entered the FoV. In that case, the track management, described in Section 6.2.3, will take over and handle new tracks. Finally, like in the previous method, the state of the KF for all the tracks is updated using the new measurement.

In comparison to "Cluster First Track Later" the main difference is that a density based algorithm is not utilized for updating existing tracks. Thus, the method is much more robust against the track merge issue. A very important parameter is the allowed gate. In case that a high value is set, then during an arm gesture all the point-targets of the arm will be inside the gate. However, noisy targets next to the track could also be assigned. When a low value is used, only the point-targets of the body of a human will be used in the track. The rest of the targets (e.g. arm) could be assigned to neighboring tracks or create a new independent track. This trade-off is one of the most important deficiencies of the method.

#### Group Tracker with Clustering

After implementing both methods, I found that in certain scenarios, which I will show in Section 6.3, tracking is not reliable. In my opinion, the "Group Tracker" is explicitly more robust when it comes to many users getting close to each other. However, I found that during arm gestures it fails to correctly assign the point-targets of the arm to the correct track.

In order to improve that, the first thing to do is to increase the size of the allowed gate. In this case, I allow targets at least one meter from a track to be associated with it. Of course, this modification on its own would lead to significant problems, since targets from other users or random noise would be assigned to the TuT. That is why I modified the fourth step; instead of simply finding the center of assigned targets, I apply a density-based clustering algorithm on the associated targets and keep the ones that belong in the generated cluster. In case that more than one cluster is generated, I select the targets that belong to the larger cluster and assign them to the TuT. I allow the targets that belonged to other clusters to be used by other tracks.

This way I observed the following two benefits: First, the clustering algorithm will discard noisy targets that satisfied the criteria of CFAR, since they will not

form a dense region. Second, if targets from other users are initially associated with the TuT at step three, the density-based clustering algorithm will discard them because they will form a new smaller cluster, separated from the main one.

### 6.2.3 Track Management

Track management is responsible for generating new tracks when new objects appear in the FoV and for deleting tracks which have not been updated. In case that a cluster is not assigned to a track, a new temporary track is created. If this track is matched with a cluster for ten consecutive iterations, it is marked as a stable track. In other words, a new track is not created any time a cluster is not assigned to existing tracks, since it may not be robust due to noise or ghost targets.

Using a similar approach, when a track is not updated with a cluster for 50 frames, it is removed from the track list. I had to keep in mind two things when I decided this value. On the one hand, the system needs to delete tracks that were initiated by mistake, so a rather small number is needed. On the other hand, when an another object blocks the TuT for a few frames, a waiting period is needed until the object moves and the TuT is visible again.

## 6.3 Results

In this section I present results that I collected with the Radar hardware described in Chapter 3.

### 6.3.1 People Tracking

In the first set of experiments I show a typical scenario of two users coming close to each other. Fig. 6.2 shows the three phases of the experiment and compares the aforementioned methods. As explained in 6.2.2, the Cluster First Track Later merges the tracks when the users come close, since the density

based algorithm considers the dense point-cloud in the region as one object. On the other hand, the other two methods by design avoid that.

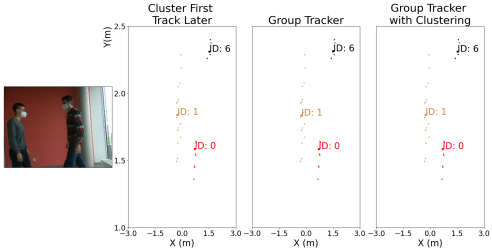
### 6.3.2 Arm Movement

In the second set of experiments, I show results from multi-user gesture recognition experiment; the two users were intentionally placed close to each other. User-0 performed swipe-up and swipe-down, whereas User-1 performed only swipe-left and swipe-right gestures. They simulated a scenario during which the system would need to separate the return signals from two users.

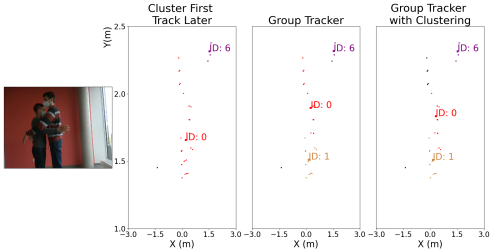
In Fig. 6.3 I show two typical examples when Group Tracker fails to assign targets to the correct users, when they are close to each other. In the first case, point-targets that belong to User-1 are assigned to User-0 because they are closer to the track's centroid. However, Group Tracker with Clustering correctly found that the point-targets initially assigned to User-0 do not form a dense cluster and rejected them. These point-targets were assigned to the next track that was close and could form dense cluster, the one from User-1. In the second case, I show a typical example of noisy point-targets being assigned in the TuT. In case of Group Tracker with Clustering, the clustering method identified that the noisy targets do not form a dense cluster and rejected them.

## 6.4 Concluding Remarks

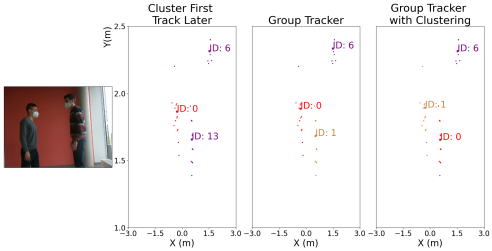
In this chapter, I presented two common approaches for tracking people using a Radar sensor and I introduced my own method which contains elements from both. I evaluated them in complex scenarios and showed that reliable people tracking is a prerequisite for consumer applications.




(a) The two users are approaching each other. Each user has a different track assigned to him, 0 and 1 respectively.

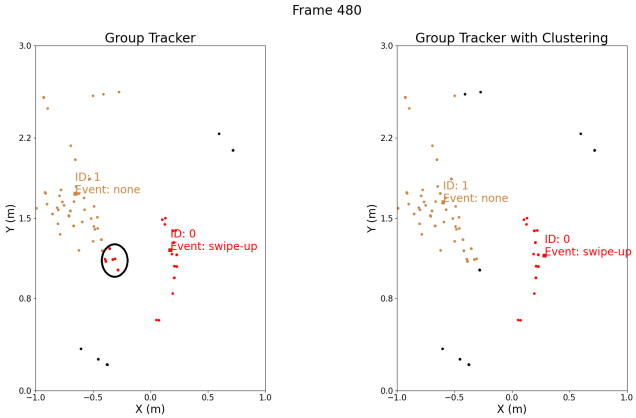



(b) The two users are very close to each other. In the case of Cluster First Track Later, the tracks of the users have merged.

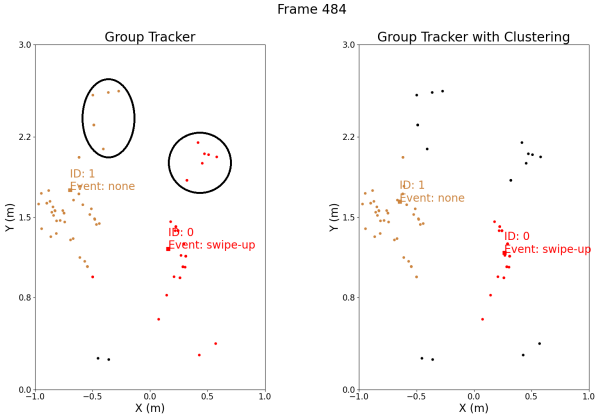



(c) The two users go back to their original position. In the case of Cluster First Track Later a new track has been generated with ID 13 and the history of the old track is lost. For the other two methods, track merge did not take place and both tracks continue to exist.

Figure 6.2: Example of people tracking with two users approaching each other that shows the track merge problem in the Cluster First Track Later method. The color of the targets represents the assigned cluster (©IEEE [NHHZ22]).




(a) In this frame Group Tracker method assigns some of the point-targets of User-1 to User-0 due to distance criterion. On the other hand, Group Tracker with Clustering used DBSCAN and identified that the body of User-0 and the point-targets form two separate clusters so it did not assign them to User-0.




(b) In this frame Group Tracker method assigns the noisy point-targets close to User-1 to him. On the other hand, Group Tracker with Clustering used DBSCAN and identified that these point-targets do not create a dense cluster and regards them as noise. Similar is the situation for the User-0 who remains static.

Figure 6.3: In this scenario the two users perform arm gestures sequentially. I provide two corner cases in which the Group Tracker provides non-satisfactory results (©IEEE [NHHZ22]).

# 7 Extensions of Gesture Recognition

The problem presented in this Chapter is tackled in the author's publications [NHHZ22, NHZ21a] alongside Jürgen Hasch, Michael Heizmann, and Thomas Zwick.

## 7.1 Introduction

In the previous chapters, I have presented two use-cases that could be useful for a smart-home application, gesture recognition with a single user and tracking people. I also showed that reliable people tracking is a prerequisite for any application that involves multiple users, since it maps hundreds point-targets to actual objects. For example, in case that a person is performing a swipe gesture and another person is walking nearby, the system needs to separate them and recognize both activities. Otherwise, the combined mD input from both extended targets will be provided to the ML for classification and the estimation will not be accurate. In a similar fashion, in [RHK] the authors tracked all extended Radar targets and managed to distinguish living from non-living objects in a complex scene. In this chapter, I focus on multi-user macro gesture recognition and hand tracking, but the list goes on and is only limited by the imagination of the researcher. For example, object recognition like in [YMR+18] could play an important role. Fig. 7.1 shows an overview of the signal processing blocks with four feasible applications, either developed in this work or available in literature.

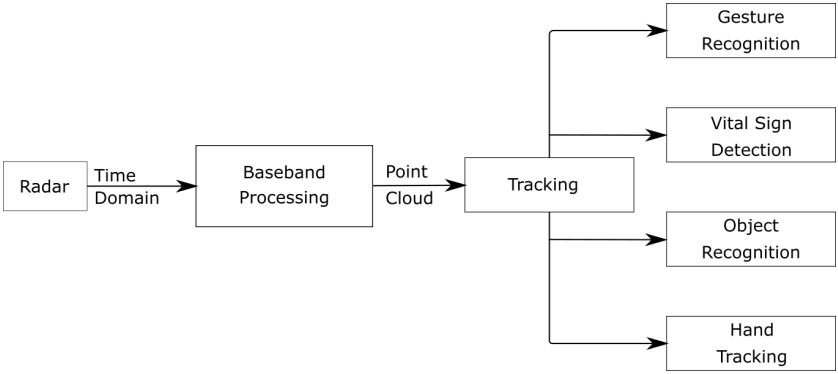



Figure 7.1: Overview of a Radar system with signal processing blocks and possible applications in smart-home scenarios (©IEEE [NHHZ22]).

## 7.2 Multi-User Gesture Recognition

In case that two users perform a gesture simultaneously, the processing pipeline of Chapter 4 will not be able to separate the return signals. Thus, the generated feature maps will contain concatenated information from both arm movements. In order to fix that problem, I utilize the tracking algorithm from Chapter 6 and build on top of that the gesture recognition pipeline from Chapter 4.

From the baseband processing and people tracking steps, I have extracted the detections and their attributes (e.g. distance, radial velocity, DoA in azimuth and elevation) that belong to each track. The next step is to calculate the mD vector and the equivalent DoA vector, like in Chapter 4, for each track. I achieve that by doing the following: For each Doppler gate I collect the corresponding detections, select the one with the highest amplitude and save its attributes. In case that no detection was available, the mD would have a zero value. Thus, for all Doppler gates I have selected detections with a certain amplitude and DoA in azimuth and elevation. This way, it is possible to generate Feature Maps like in [NHZ21b], with using point-cloud instead of using RDMs.

The rest of the processing chain is similar to [NHZ21b]. For each track the event detection algorithm will search for significant motion. If a positive result is given, an empirical method will generate a one-dimensional signal from each

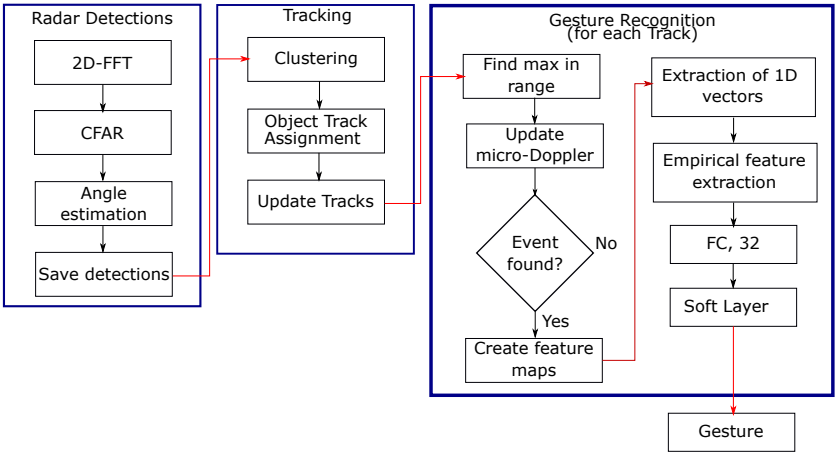



Figure 7.2: Signal processing pipeline for multi-user tracking and gesture recognition (©IEEE [NHHZ22]).

feature-map; for the mD image I use the highest velocity of each frame, for the two DoA maps I use the median angle of each frame. For simplifying the classification even further, another feature extraction step provides 13 features. I use the same ML model that I trained in [NHZ21b] that is capable of recognizing ten classes. Fig. 7.2 shows a diagram with the proposed signal processing pipeline.

## 7.3 Hand Tracking

Gesture recognition, either on micro or on macro scale, allows the user to have a level of control in a digital system with a set of known commands. However, in many cases a system requires a value within an allowed range, for example when controlling the volume of a TV. In my opinion, an intuitive way to achieve that in a touchless manner is via tracking the hand of the user. Camera-based systems and recent achievements in ML made that possible [BGR+20]. However, such a solution raises privacy concerns [VKC19] and consumes significant amount of power not only due to the sensor but also due to the processing that is involved.

More information and comparison with alternative technologies can be found in Section 2.5. I tackled this problem using only mmWave technology and to the best of my knowledge, I am the first to introduce the concept.

Once people tracking is completed, the system knows the location of people or large objects in the FoV. When one of the users starts performing a gesture, the tracking algorithm will assign the point-targets to the user's track. This information accumulated over time is useful for gesture recognition as explained in Section 7.2. However, since the tracking algorithm assigns the point-targets of the arm to the track that belongs to the user, it is not possible to separately track the hand movement.

That is why during the first step of hand tracking, the system needs to split the existing user track into body and hand. The user can inform the system to do that using a triggering mechanism (e.g., voice, button, gesture, etc.). For example in my work, I achieved that with the arm gesture called "rotate". In order to split the user track into two separate tracks for body and hand I use the K-means algorithm [Llo82] and configure it to find two different clusters using the original. I assign the cluster with less number of point-targets to the hand and the other one to the body.

Once the above is completed, I can use the existing tracking method and track the location of the hand for any number of frames. However, I found that using a single Radar is not a robust solution for the following reasons. The FoV of a single sensor might not be enough to capture a complete arm movement from left to right (or vice-versa), in case that the user is close to the sensor. To make matters worse, when the user is a few degrees on the left or right side of the Radar, this situation is more critical. In addition, the number of point-targets assigned to the hand is in many cases not enough for a reliable clustering. Last but not least, I already mentioned that the measurement vector of the tracking filter is the position of the cluster. Thus, during a maneuver of the hand, the filter requires few updates until it converges.

In order to overcome these issues, I improved the hardware setup and created a loosely coupled network using two identical Radar nodes similar to [CD21]. The measurement frames of the two Radars are triggered with a few microseconds time difference, in order to avoid interference between the Radar signals. The two nodes are positioned at a distance of a few centimeters (e.g., 30cm) in

x-axis. In Section 7.3.2, dedicated on the experimental Radar system, I provide more information on the topic.

### 7.3.1 Signal Processing Chain

In the following, I will describe the signal processing chain. After applying the baseband processing for each Radar individually, the two sets of point-targets are concatenated. Therefore, the FoV is now significantly larger since it combines the FoV from both sensors. In addition, when the hand of the user is in the overlapping FoV of both sensors, there are two benefits. Firstly, the number of point-targets assigned to the hand track will be higher. Secondly, if the hand is considered as a point target, it is possible to estimate its velocity vector in a single snapshot as I will explain in subsection 7.3.1. This information can be used in the KF and reduce the residual during a hand maneuver.

Fig. 7.3 shows the geometric setup of a two-node Radar network and a moving target. The goal is to estimate the velocity vector of a small target, which moves in a two-dimensional Cartesian space. The Radars measure in each snapshot the radial velocity $v1_r$ and $v2_r$ respectively. In addition, they measure the azimuth angle $\phi 1$ and $\phi 2$. Using basic trigonometric transformations it is possible to derive Eq. 7.1.

$$\begin{bmatrix} v1_r \\ v2_r \end{bmatrix} = \begin{bmatrix} \cos(\phi 1), \sin(\phi 1) \\ \cos(\phi 2), \sin(\phi 2) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{7.1}$$

This can be reformulated as in Eq. 7.2.

$$\vec{v_r} = A\vec{v} \tag{7.2}$$

Using high resolution Radar nodes, the baseband processing will generate many targets even for small objects. In that case, the aforementioned equations can be extended by adding one row for each detected target for each sensor as in Eq. 7.3.

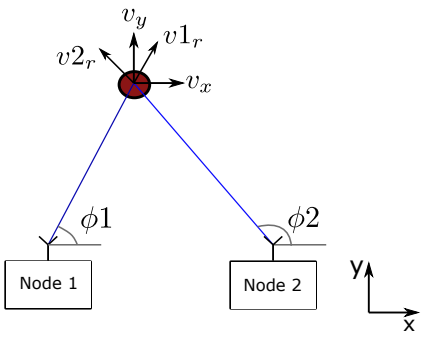



Figure 7.3: Example geometry of a two-node Radar network and one moving target (©IEEE [NHHZ22]).

$$\begin{bmatrix} v1_r \\ v2_r \\ \vdots \\ vN_r \end{bmatrix} = \begin{bmatrix} \cos(\phi 1_1), \sin(\phi 1_1) \\ \cos(\phi 2_1), \sin(\phi 2_1) \\ \vdots \\ \cos(\phi N), \sin(\phi N) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{7.3}$$

This can be reformulated as in Eq. 7.4.

$$\vec{v_r} = H\vec{v} \tag{7.4}$$

The above system is overdetermined; using the pseudo-inverse the velocity vector can be estimated as in Eq. 7.5.

$$\vec{v} = (H^T H)^{-1} H^T \vec{v_r} \tag{7.5}$$

Naturally, the above system could be solved with numerical methods as in Eq. 7.6, but this is out of the scope of this work.

$$\vec{v} = \underset{v}{\operatorname{argmin}} ||\vec{v_r} - H\vec{v}||^2 \tag{7.6}$$

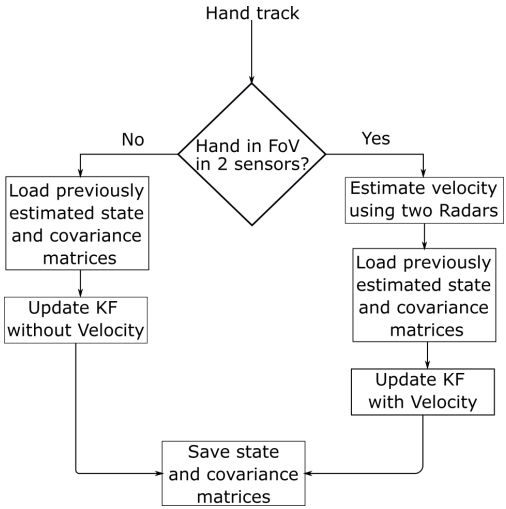



Figure 7.4: Multi Model with Velocity Estimation. Used for hand tracking with two Radar nodes (©IEEE [NHHZ22]).

### Multiple Model with Velocity Estimation

The estimated velocity can be used in the measurement vector of KF. However, the hand of the user is not always in the FoV of both sensors, thus it is not a good idea to rely only on the single snapshot estimated velocity.

To surpass this issue, I developed the Multiple Model with Velocity Estimator (MMVelEst), that utilizes two KFs; I was influenced by the Multiple Model Adaptive Estimator (MMAE) [ZM15] and Interacting Multiple Models (IMM) [BSLK01]. The first filter that I use is similar to the people-tracking case, whereas the second includes the estimated velocity in Cartesian space in the measurement vector. When the single snapshot estimated velocity is not available, I use the first KF to update the state of the track and when it is available I use the second KF. In either case, since the state vector is the same, I can use the previous estimation of the state and covariance matrix. Fig. 7.4 shows the diagram for MMVelEst.

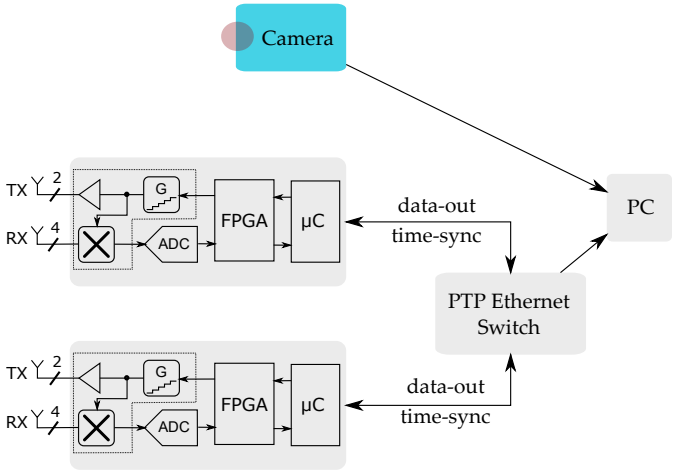



Figure 7.5: Diagram of a loosely coupled Radar sensor network, used for hand tracking including a camera for verification (©IEEE [NHHZ22]).

### 7.3.2 Experimental Radar System

For the hand tracking application, I modified the setup in order to facilitate a second Radar node. I use Precision Time Protocol (PTP), which generates a master slave relationship among the clocks in the system. All clocks ultimately derive their time from a clock known as the grand-master clock [noa08]. This way, each sensor node is synchronized on the shared system time of the network.

Fig. 7.5 shows a diagram that describes the system and Fig. 7.6 depicts the Radar setup.

## 7.4 Results

### 7.4.1 Multi-User Gesture Recognition

In the following examples the combination of tracking and gesture sensing capabilities will be shown. Two users were performing gestures and the system

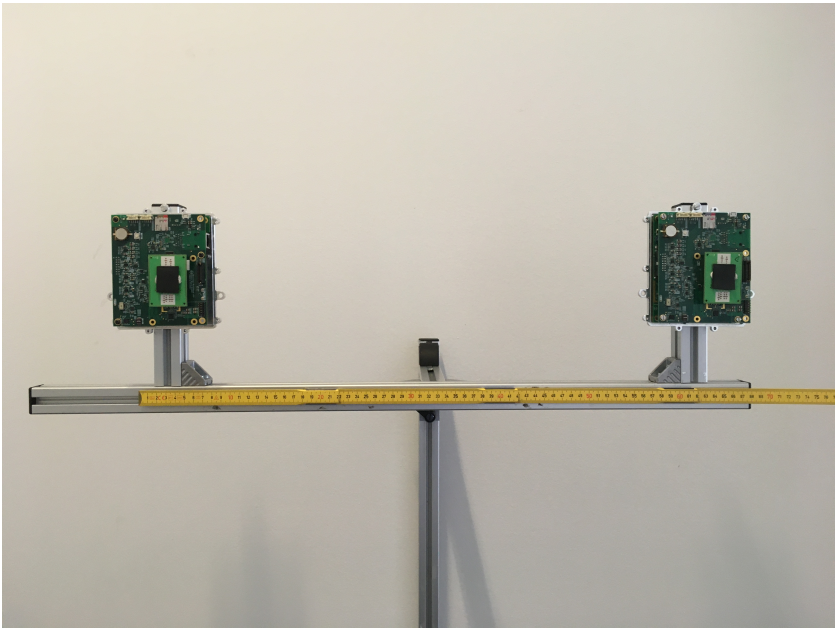


Figure 7.6: Hardware setup with two Radar nodes, approximately 60cm apart (©IEEE [NHHZ22]).

could simultaneously track them and recognize their movements. One of the users is performing swipe-left/right and the other one swipe-up/down. This demonstrates that the framework can achieve multi-user gesture sensing and that the ML model can provide accurate results for different ranges. Fig. 7.7 and 7.8 show the result of multi-user gesture sensing.

### 7.4.2 Hand-Tracking

In order to evaluate the performance of the system, I collected ground truth using one RGB-D camera, the Intel RealSense, and utilized the Mediapipe package [BGR+20] to extract the pose of the user. Fig. 7.9 shows two frames that I recorded during hand-tracking, the output of the pose-estimation and finally the tracks that are generated by my framework.

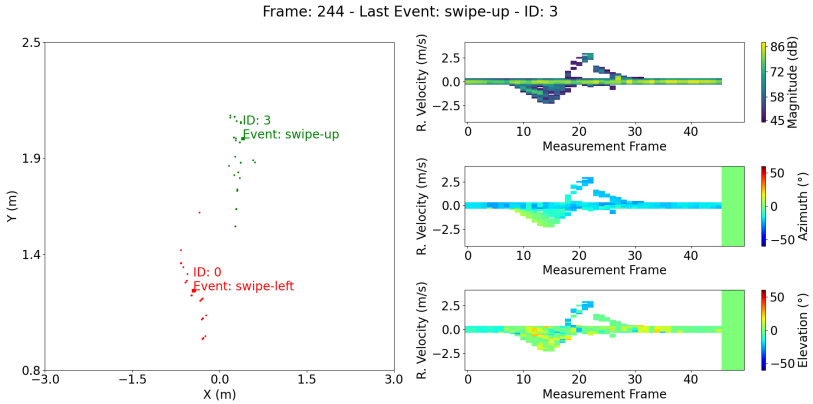



Figure 7.7: Example with two users performing gestures. One of them performs swipe-left whereas the other one swipe-up. The framework can track the users and then recognize the gestures.

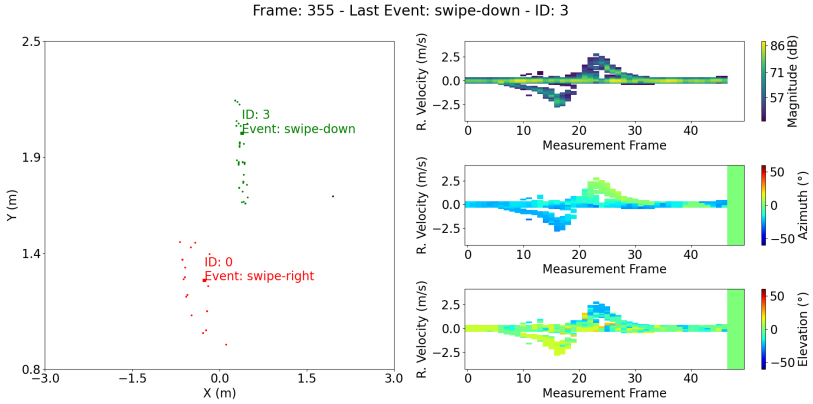
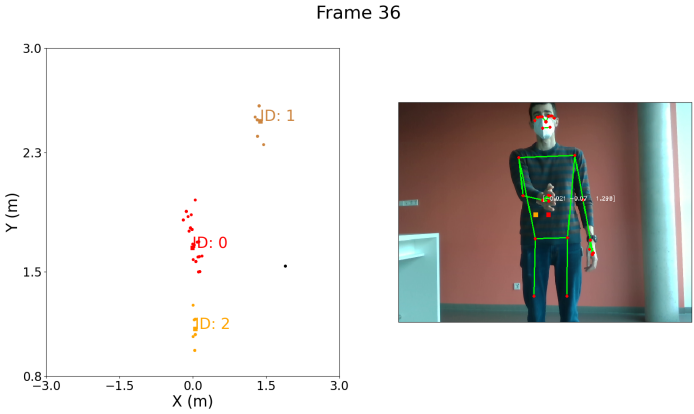
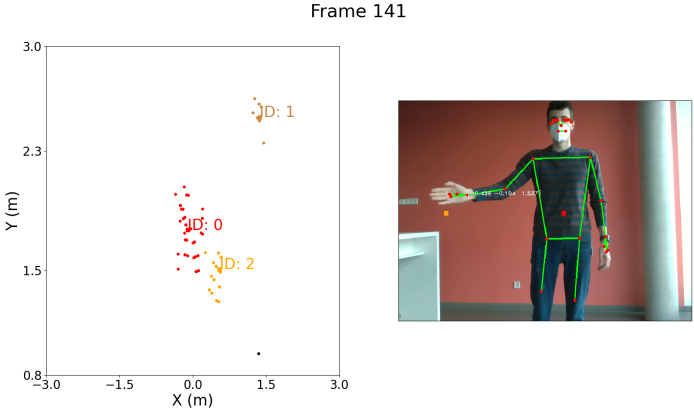



Figure 7.8: Example with two users performing gestures. One of them performs swipe-right whereas the other one swipe-down. The framework can track the users and then recognize the gestures.




(a) Initialization of the new track (ID: 2) which refers to user's hand.




(b) User moved the hand and the track (ID: 2) accurately followed it.

Figure 7.9: Example of hand-tracking. The image on the right side shows the projected Radar tracks with a rectangle symbol and the estimated pose from Mediapipe with circles and lines (©IEEE [NHHZ22]).

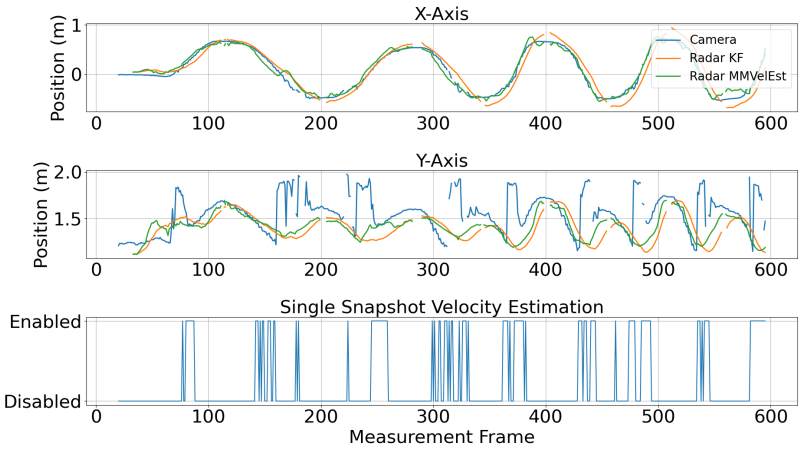



Figure 7.10: Hand Tracking: Comparison between MMVelEst and single KF. For Ground Truth an RGB-D sensor was used together with Mediapipe for pose estimation (©IEEE [NHHZ22]).

Fig. 7.10 shows the estimation of position from the ground truth and the two estimation methods that I used. In addition, Fig. 7.11 shows the residual of the two parameter estimators in comparison to the ground truth. Both filters provide similar results until frame 300; up to this part the hand of the user was moving relatively slow. After that the residual of the KF increased and reached 0.4 meters. However, the MMVelEst appears to be unaffected, since it can use the single snapshot velocity estimation in the kinematics model.

### 7.4.3 Comparison with camera-based solution

It is worth pointing out that I found it hard to collect accurate ground-truth when the color of the clothes of the user was similar to the background, or when there were reflections from the sun. In addition, in many cases the ground truth from the camera-based system was corrupted when the hand was moving. On the other hand, it was very accurate and robust with static poses. Therefore, I believe that a high-accuracy system should integrate both camera and mmWave

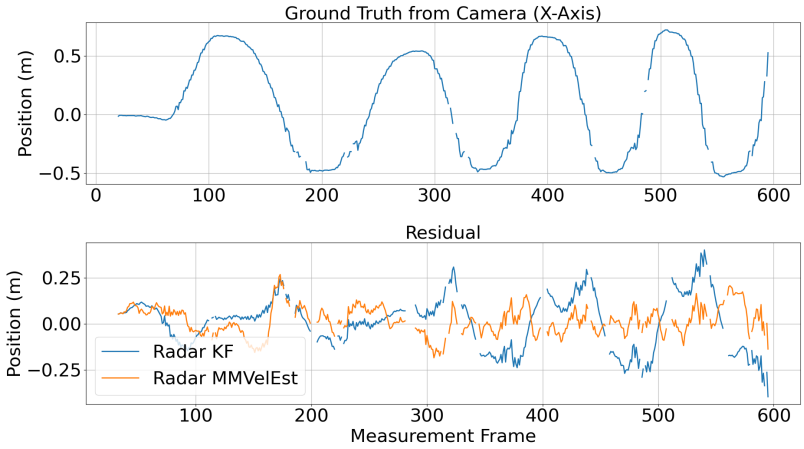



Figure 7.11: Hand Tracking: The first subplot shows the reference signal from the camera-based system whereas the second subplot compares the residual of the two parameter estimators (KF and MMVelEst) (©IEEE [NHHZ22]).

based sensors in order to get the best of two worlds. A possible example could be the Microsoft HoloLens which already has a built-in hand tracking feature.

## 7.5 Concluding Remarks

In this chapter, I presented two novel applications that could be realized with mmWave technology. Both of them are suitable for a smart-home and could improve the human machine interaction systems. With the multi-user gesture recognition, I extended my work on gesture recognition and combined it with people tracking for the first time. Similarly, the hand-tracking feature is a novel application of mmWave technology and to my knowledge I am the first to introduce it.

# 8 Conclusion

## 8.1 Summary of Key Contributions

Transitioning Radar technology from the automotive to the consumer market was the main motivation of this Thesis. The vast majority of similar scientific works focused on micro-gestures performed a few centimeters above a Radar sensor. The popularity of that approach increased even more when Google launched Pixel 4 with an integrated Radar sensor. Such a solution utilizes only one of the important attributes of Radar technology, that of mD.

My decision to focus on macro-gesture recognition and tracking was based on the fact that they constitute the basis of most of the contactless HMI, and they render possible a larger range of applications. At macro scale, the mD is not anymore the important feature for predicting the gesture type, since the DoA plays an equally crucial role. Integrating this information in a form suitable for machine learning, using the Feature Maps was a significant step. On top of that, using empirical features instead of a CNN provided two more benefits. Firstly, the sample size of the training dataset did not need to be tens of thousands as is usually expected; roughly one thousand samples were enough. Secondly, the inference part needs much lower processing power since the used model has only 32 nodes. The average accuracy in the test set for ten classes, including random motion, was 94.3% at 50Hz frame rate and 92% at 25Hz, comparable to research works that focus on micro-gestures only.

In my next work, I created an automated tool capable of creating synthetic samples for gesture recognition with a Radar sensor. Such synthetic dataset generators are very useful in the computer vision and machine learning community. However, initial efforts to bring them in the mmWave world were either unsuccessful or lacking important features. Blender, the free and open source 3D creation suite was utilized for generating human-like animations that perform gestures. Using its Python API, I extracted point-targets from all frames

of the animations and fed them in a Radar simulator that I developed. The latter provides time domain Radar data in a similar form as the experimental sensor and can be used by the aforementioned processing chain, designed for gesture recognition. I used those samples to train a machine learning model and I evaluated its performance on the real dataset that I had already collected, yielding an average accuracy of 84.2%. This shows that the data-generator can contribute in the pre-training phase of a model, as well as for capturing corner cases related to the speed of execution and the position of the subject, that are difficult to reproduce during data collection.

After having completed an in-depth analysis on the topic of gesture recognition for single user in the FoV, I evaluated existing people tracking algorithms. Such algorithms are the corner-stone for applications in multi-user scenarios, since they allow to split the point-cloud into multiple point-clouds, one for each person, and process its history. I selected two commonly used, namely the "Group Tracker" and the "Cluster First Track Later", which performed with accuracy that agreed with the ones reported by their authors, in simple scenarios. However, in more complex cases with more than two people or when users are close to each other and move their arm, the performance of the above two methods deteriorated. My approach called "Group Tracker with Clustering", combined ideas from both methods without inheriting their drawbacks, was able to perform better in the aforementioned scenarios.

Finally, I developed two features that would be useful for a smart-home application. The first one tackled the problem of multi-user gesture recognition for the first time with mmWave technology. To achieve that, I combined my people tracking method with my gesture recognition pipeline. The second feature focused on hand-tracking, solely with a Radar system; improved accuracy was accomplished by using a second Radar node and an adaptive Kalman filter that made use of an estimated velocity vector of the hand.

## 8.2 Outlook

In the aforementioned results, the users were always placed at a distance lower than three meters. At higher range, the results are deteriorated due to the low RCS of a human arm, the noise characteristics of the sensor and the large FoV in azimuth and elevation which decrease the gain. Similarly, during hand-tracking

at a higher distance, the number of targets generated by the arm is too low, and consequently the hand-track is deleted by the track maintenance algorithm. A modern fully integrated Radar System on a Chip [RGG+21] provides much better SNR characteristics and a bandwidth up to 4GHz. Thus, a dense point-cloud would be available at longer range.

The number and position of antennas also play a crucial role. By optimizing the antenna array in azimuth dimension and increasing the angle resolution, the system would be able to identify more objects in the scene. Furthermore, in case that more antennas are used to improve the elevation resolution, it would be possible to apply hand-tracking in three dimensions. It is important to note that adding more antennas in a consumer application is significantly easier in comparison to the automotive domain, in which the requirement for maximum unambiguous velocity leads to very short chirps.

During the experiments I found that the maximum radial velocity during fast arm motions is around 4 m/sec and is achieved during the push or pull gestures. However, in case that gestures with only lateral movement are selected (i.e., swipe-right/left/up/down, etc.) then the required maximum unambiguous velocity will be significantly decreased, like in [HLG+21]. This means that the chirp duration can be higher and thus even more antennas could be used with TDM.

In future work, a system needs to be developed with SW-HW co-design and take into consideration performance, power consumption, cost and flexibility. An architect should choose wisely the target hardware for the different processing blocks. For a block that no major modifications are expected in the future, it is highly suggested to sacrifice flexibility and optimize the rest as much as possible by implementing it in an Application Specific Integrated Circuit (ASIC) or FPGA. The baseband processing constitutes such a block.

On the other hand, algorithm development for tracking people with mmWave sensors is an up-and-coming topic and the designer should expect frequent improvements in the future. That is why my suggestion is to use C++ for this part, since it is commonly used in Robotics, offers very high levels of code re-use and an efficiency very similar to C programming language. An ARM processor should be capable of performing the processing in real-time, since the point-cloud that is generated from baseband processing usually has only a few hundred targets. Finally, for the gesture recognition part, flexibility is of importance, so that new empirical features could be easily integrated. In case

that a large data collection takes place, with aim of collecting tens of thousands of samples, most likely a CNN will outperform the empirical feature extraction solutions. Thus, a GPU will be needed for a real-time inference.

Regarding the frontend, during my work I used an experimental one which was developed within our laboratory. Currently, several semiconductor manufacturing companies offer $2T_x - 4R_x$ or even $3T_x - 4R_x$ with antenna on package. Some of them even provide software development kits which include baseband processing and tracking algorithms. As a result, all the RF/Antenna design and low-level signal processing topics are abstracted in a single chip. This could reduce the development time significantly and bring the product to the market faster.

Finally, more applications could be developed using mmWave technology. Recent studies have shown that it is possible to identify if an object is living or not and measure the respiration rate of humans [RHK]. Another interesting feature would be to identify if an object is an animal or a human. For that task, recent methods developed for Lidar technology, like the PointNet [CSKG17], could be easily transferred in the mmWave domain and classify objects.

# A Appendix

## A.1 Gestures

In the following figures, the nine gestures used in Chapter 4 are presented.

(a) First part of gesture

(b) Intermediate part of gesture

(c) Last part of gesture

Figure A.1: Pull

(a) First part of gesture

(b) Intermediate part of gesture

(c) Last part of gesture

Figure A.2: Push

(a) First part of gesture

(b) Intermediate part of gesture

(c) Last part of gesture

Figure A.3: Swipe-Up

(a) First part of gesture

(b) Intermediate part of gesture

(c) Last part of gesture

Figure A.4: Swipe-Down

(a) First part of gesture

(b) Intermediate part of gesture

(c) Last part of gesture

Figure A.5: Swipe-Right

(a) First part of gesture

(b) Intermediate part of gesture

(c) Last part of gesture

Figure A.6: Swipe-Left

(a) First part of gesture

(b) Second part of gesture

(c) Third part of gesture

(d) Last part of gesture

Figure A.7: Rotate

(a) First part of gesture

(b) Intermediate part of gesture

(c) Last part of gesture

Figure A.8: Wave. The three parts need to be repeated 2-3 times.

(a) First part of gesture

(b) Intermediate part of gesture

(c) Last part of gesture

Figure A.9: Push-Pull

## A.2 Machine Learning

In this section I will provide a gentle introduction to the topic of ML and more specifically in Multilayer Perceptron and Convolution Neural Networks. Those who are interested in a deeper level are encouraged to consider textbooks with a comprehensive coverage of the fundamentals such as [GBC16] and [Bis06].

### A.2.1 Basic Definitions

Computers can easily solve problems described by a list of formal, mathematical rules. However, it is very hard to program them to perform intuitive tasks for any human, like speech and image recognition. Instead of programming them, the state-of-the-art solution is to allow them to learn from examples and understand the world in terms of a hierarchy of concepts. An ML algorithm is an algorithm that is able to learn from data. A more formal definition found in [GBC16] is the following: "A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$ improves with experience $E$". The above sounds more complicated than it actually is and I will try to break it down.

Tasks are usually described in terms of how the ML system processes a sample (also called example). A sample is a collection of features that have been measured, and is usually represented as a vector $\chi \in \mathbb{R}^n$ where each entry $\chi_i$ is another feature. For example, the features of an image are the values of the pixels. Many tasks can be solved with machine learning (e.g., classification, regression,

machine translation, anomaly detection, imputation of missing values, etc.), in this work only classification tasks are needed. In this case, the computer is asked to specify which of $k$ predetermined categories a new sample belongs to. To solve that, the algorithm needs to produce a function $f : \mathbb{R}^n \Rightarrow \{1, \ldots, k\}$.

A quantitative measure of the performance $P$ is used to evaluate the ML algorithm. For classification tasks the most common approach is to use the accuracy of the model, which is the proportion of samples for which the model produces the correct output. It is important to evaluate a model with samples that it has not seen before, since this determines how well it will work when deployed in the real world. Therefore, a separation of the test set and of the training set, which is used for training the model, needs to take place.

ML algorithms can be categorized as unsupervised or supervised by what kind of experience $E$ they are allowed to have during training. The term supervised learning originates from the view of the label being provided by an instructor or teacher who shows the ML system what to do. In this work, I used supervised learning so that each sample in the dataset contains features and is associated with a label (also called target). One common way of describing a dataset is with a design matrix X that contains a different sample in each row. Each column corresponds to a different feature.

### A.2.2 Multilayer Perceptron

Multilayer Perceptron, also called feed-forward neural network, is commonly used to approximate some function $f^*$. For example, a classifier $y = f^*(\chi)$ maps an input $\chi$ to a category $y$. Such a network defines a mapping $y = f(\chi; \theta)$ and learns the values of the parameters $\theta$ that result in the best function approximation. The name feed-forward originates from the fact that information flows through the function being evaluated from $\chi$, through the computations used in $f$, and finally to the output $y$. There are no feedback connections that connect the output to the input of the model. Such models are of great importance in many ML problems and they are used extensively in commercial applications.

The model is associated with a directed acyclic graph describing how the functions are composed together. In case of three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$, the first, second and third layer respectively. They will form $f(\chi) =$

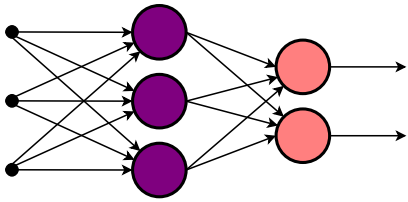


Figure A.10: Simple example of an MLP. It consists of three input nodes, one hidden layer with three nodes, and an output layer with two nodes.

$f^{(3)}(f^{(2)}(f^{(1)}(\chi)))$. The name "deep learning" arose from such models with a large number of layers, since the length of the chain is equivalent to the depth of the model. During training, each sample $\chi$ specifies what the last layer (also called output) must do. The learning algorithm must decide how to use all the layers, intermediate and output, to best implement an approximation of $f^*$. The intermediate layers are sometimes called hidden layers because they are not directly observable from the systems inputs and outputs.

This model is called neural because it is inspired by neuroscience and each element of each layer may be interpreted as a neuron. Rather than thinking of each layer as representing a single function, it can be considered as many units that act in parallel, each representing a vector to scalar function. Each unit resembles a neuron in the sense that it receives input from many other units and computes its own activation value. However, modern neural network research is guided by mathematical and engineering disciplines and its goal is not to mimic the behavior of the brain.

Learning occurs in the MLP by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. It is possible to represent the degree of error in an output node $j$ in the $n$-th sample point by $e_j(n) = d_j(n) - y_j(n)$, where $d$ is the target value. The node weights can then be adjusted based on corrections that minimize the error in the entire output, given by $E(n) = \frac{1}{2}\sum e_j^2(n)$. Using gradient descent and back-propagation, the change in each weight is calculated. The user needs to set the hyper-parameter learning rate which defines how quickly the weights will converge to a response. Fig. A.10 shows a simple example of an MLP with one hidden layer that consists of three neurons.

### A.2.3 Convolutional Neural Networks

They are a specialized kind of neural network for processing data that has a known grid-like topology. Typical examples include time-series and image data. The former can be thought of as a 1-D grid taking samples at regular time intervals, whereas the latter as a 2-D grid of pixels. CNNs have been successful in practical applications and are considered state-of-the-art for image recognition and image segmentation. The name convolutional indicates that the model employs a mathematical operation called convolution instead of matrix manipulation in at least one of the layers.

In its most general form, convolution is a specialized linear operation on two functions ($f$ and $g$) that produces a third function ($f * g$) that expresses how the shape of one is modified by the other. In that case $f$ is considered as the input and $g$ as kernel, which is similar to a weighting function. Convolution leverages three important ideas that can help improve an ML system: sparse interactions, parameter sharing and equivariant representations.

In case of MLP every output unit interacts with every input unit. However, when an image is processed the input could contain thousands of pixels, but certain features like edges occupy only tens or hundreds of pixels. As a result, only sparse interactions are needed which also has the benefit of fewer parameters and memory requirement reduction. The improvement in efficiency is quite important; if there are $m$ inputs and $n$ outputs, then matrix multiplication requires $m \times n$ parameters and the algorithms have $O(m \times n)$ runtime. If the number of connections for each output is limited to $k$, then the sparsely connected approach requires only $k \times n$ parameters and $O(k \times n)$ runtime.

Parameter sharing refers to using the same parameters for more than one function in a model. In an MLP, each node is used only once when computing the output of a layer. In other words, it is multiplied by one element of the input and then never revisited. In a CNN, each member of the kernel is used at a every position of the input. Thus, rather than learning a separate set of parameters for every location, only one set is learned. This does not have an effect on the runtime of the forward propagation but it reduces the storage requirements.

Equivariant representation means that if the input changes then the output changes in the same way. In case of images, the convolution creates a 2-D map of where certain features appear in the input. If the objects in the input are

moved, its representation will move the same amount in the output. This is useful when processing images and it is important to detect edges in the first layer of a CNN. The same edges appear more or less everywhere in the image, so it is practical to share parameters across the entire image.

A typical layer of a CNN consists of three stages. Initially, the layer performs several convolutions in parallel to produce a set of linear activation. Then each linear activation is run through a nonlinear activation function, such as the Rectified Linear Unit. Finally, a pooling function down-samples feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively. In any case, pooling helps to make the representation approximately invariant to small translations of the input.

Two popular CNN architectures that include aforementioned layers can be found in Fig. A.11.

## A.3 Software Architecture

In the following section, I provide UML diagrams for the most important software packages that I developed. The first part is about baseband processing of time-domain Radar data; theoretical background can be found in Section 2. The second part provides diagrams for the tracking package, more info about people tracking can be found in Section 6. Finally, the third presents diagrams for the synthetic dataset generator; detailed information and results are available in Section 5.

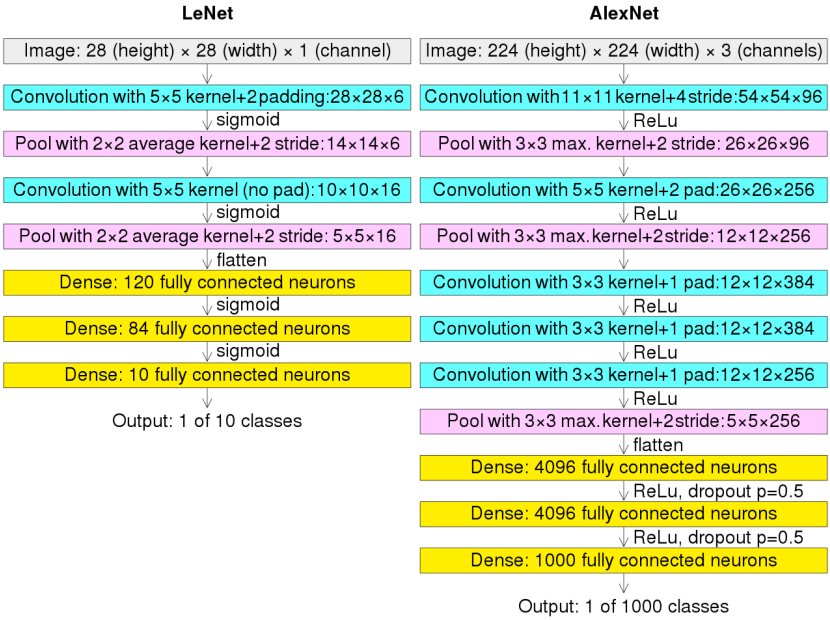



Figure A.11: Comparison of the architectures of LeNet and AlexNet by CMG Lee using data from `http://d2l.ai/chapter_convolutional-neural-networks/lenet.html` and `http://d2l.ai/chapter_convolutional-modern/alexnet.html`.

### A.3.1 Baseband Processing

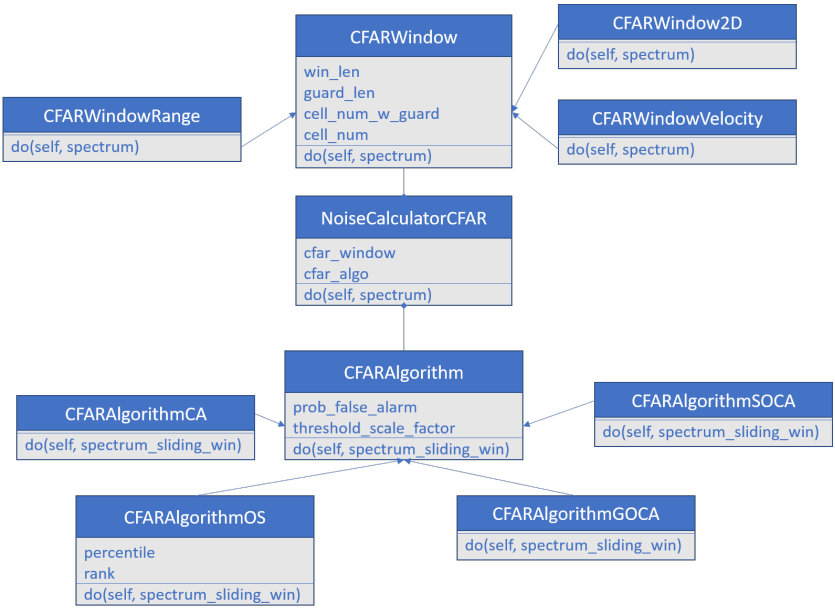



Figure A.12: Software architecture of the noise calculator that uses CFAR. It can use four different algorithms and apply them in range, velocity as well as in 2D window functions. The $NoiseCalculatorCFAR$ has two attributes, one for applying the window function and one for applying the CFAR algorithm, this way these two orthogonal parts of the algorithm are decoupled. The strategy pattern is used for the $CFARWindow$ and $CFARAlgorithm$ interfaces.

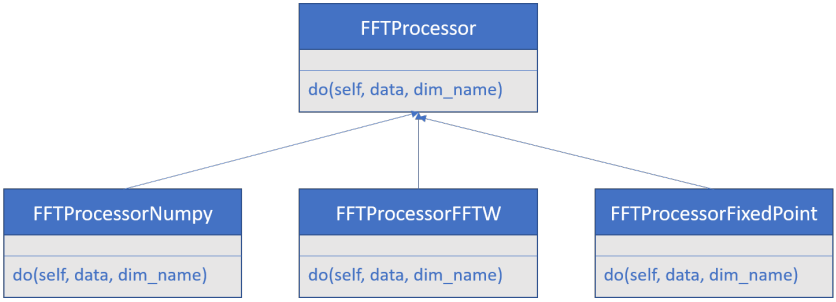



Figure A.13: Software architecture of the FFT processing part. Strategy pattern is used to implement three back-ends, one with Numpy, one with FFTW [FJ05] and one with a fixed-point implementation in C.

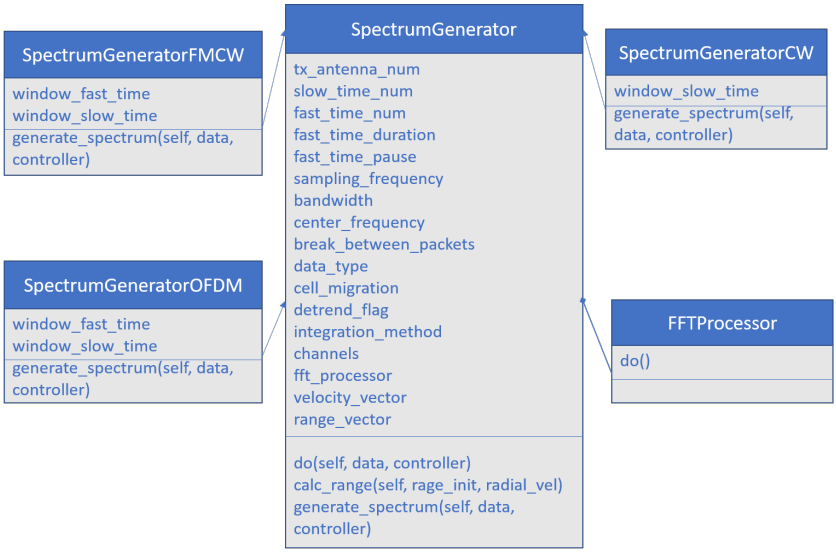



Figure A.14: Software architecture of the part that generates the spectrum from the time domain data, with concrete implementations for FMCW, OFDM and CW. In this case, the template pattern is used, $do$ being the template method in the abstract class $SpectrumGenerator$. The sub-classes provide a concrete implementation of the $generate\_spectrum$ method. The $SpectrumGenerator$ "has-a" (composition) $FFTProcessor$ through which it implements the FFT.

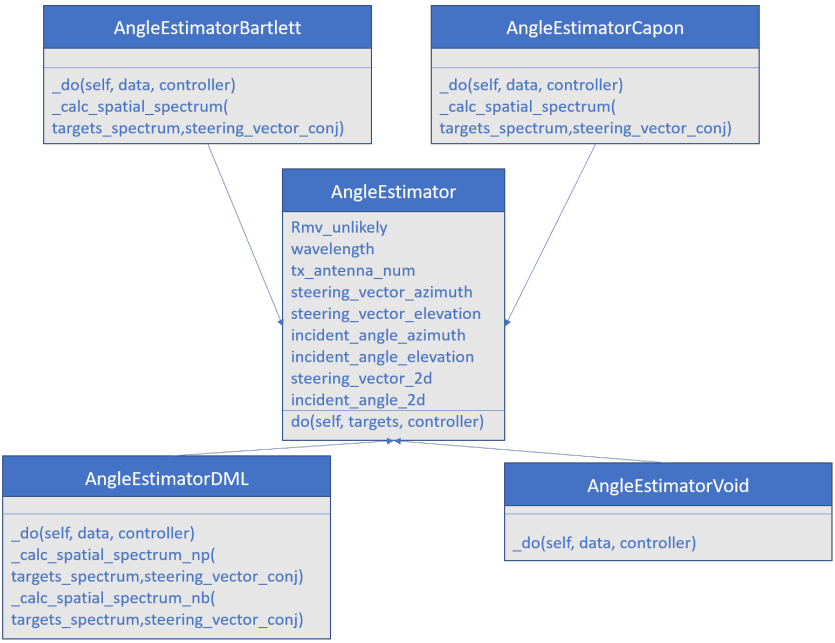



Figure A.15: Software architecture of the part that performs DoA. The template pattern is used, *do* being the template method in the abstract class *AngleEstimator*. The subclasses provide a concrete implementation of the *_do* method. In case of DML, the spatial spectrum can be calculated with two different backends, Numpy or Numba. *AngleEstimatorVoid* is used when the user did not request DoA calculation, in that case the template method *do* is an empty call.

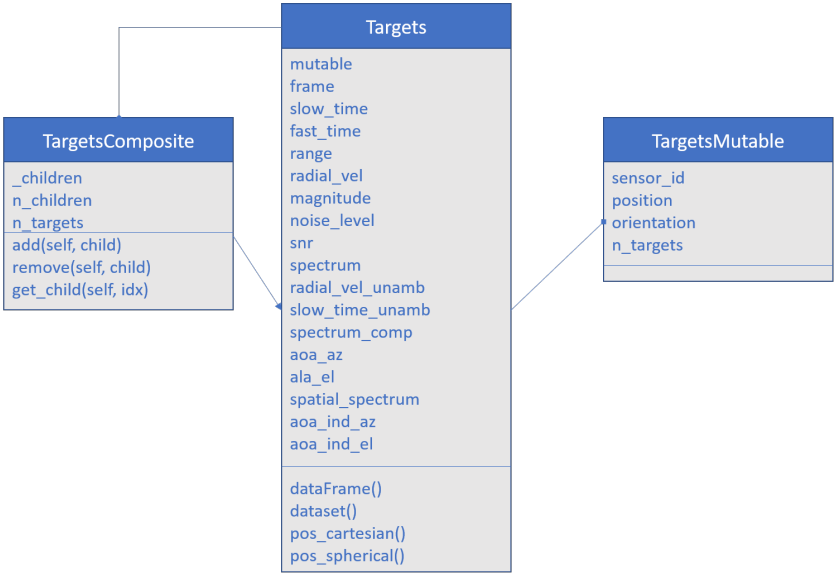



Figure A.16: Software architecture of the container of the point-cloud. $Targets$ is a Python Dataclass and is also "frozen", which means that once it is instantiated the values of the attributes cannot be modified. That is useful in case that a function tries to modify it by mistake. For example, the clustering part will use the point-cloud to generate clusters, but it must not modify it. Some information of the point-cloud is still mutable (i.e. non-frozen), this is available in the $TargetsMutable$ class. This architectural choice was made because during the baseband processing, the location of the sensor is not known, this is set later. For concatenating point-clouds from multiple sensors, I used the composite pattern, implemented by $TargetsComposite$. The benefit is that the user of $TargetsComposite$ class is not aware if only one sensor or multiple sensors are used, since the interface is exactly the same.

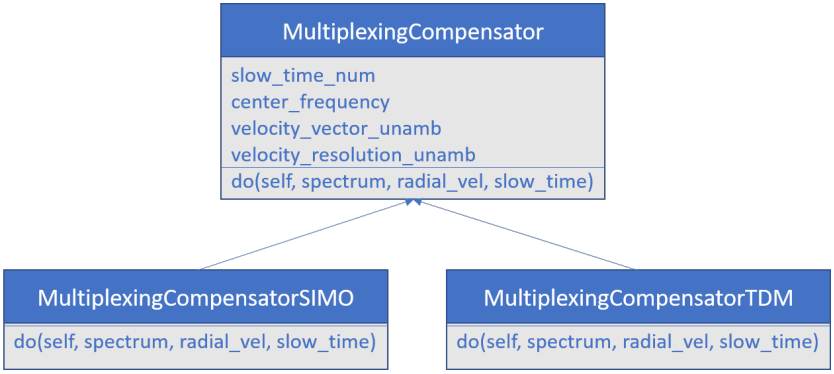



Figure A.17: Software architecture of the part that compensates the effect of multiplexing, like in TDM-MIMO. The strategy pattern is used and two cases have been implemented.

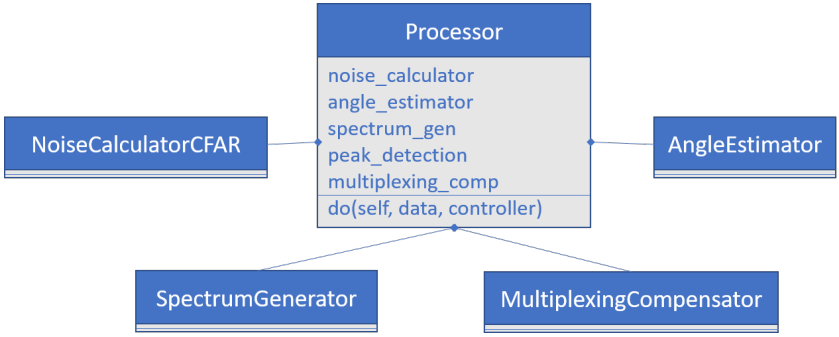



Figure A.18: Software architecture of the complete baseband processing chain. The facade pattern is used so that the interaction of complex components is masked within one class.

### A.3.2 Tracking

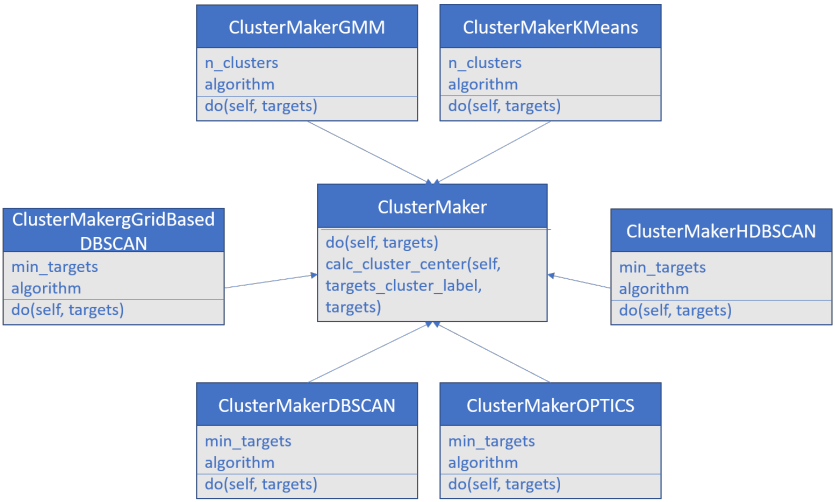



Figure A.19: Software architecture of the part that clusters the point-cloud into objects. The strategy pattern has been used to implement several algorithms, while keeping the same interface.

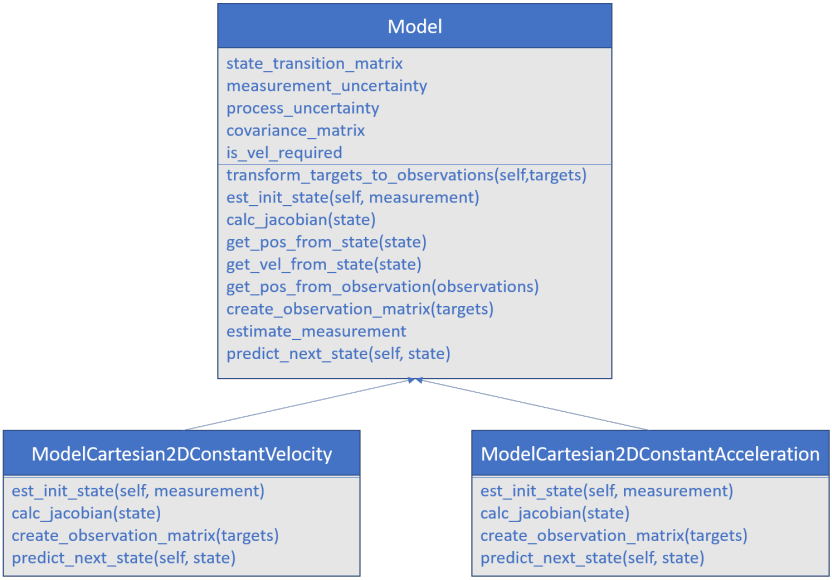



Figure A.20: Software architecture of the different kinematics models that I have tried.

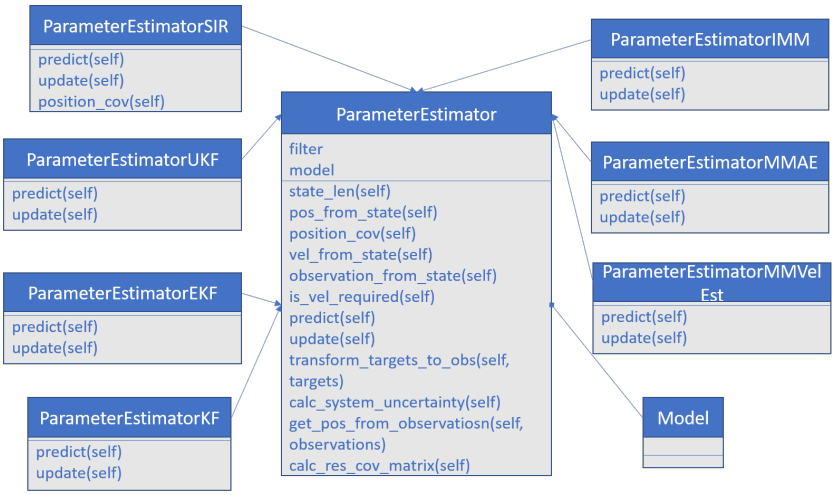



Figure A.21: Software architecture for the available parameter estimation approaches. It can handle different KF like linear, non-linear (EKF, UKF) as well as adaptive (IMM, MMAE). The MMVelEst is also available, developed for hand-tracking. The estimator can use any Model instance shown in A.20.

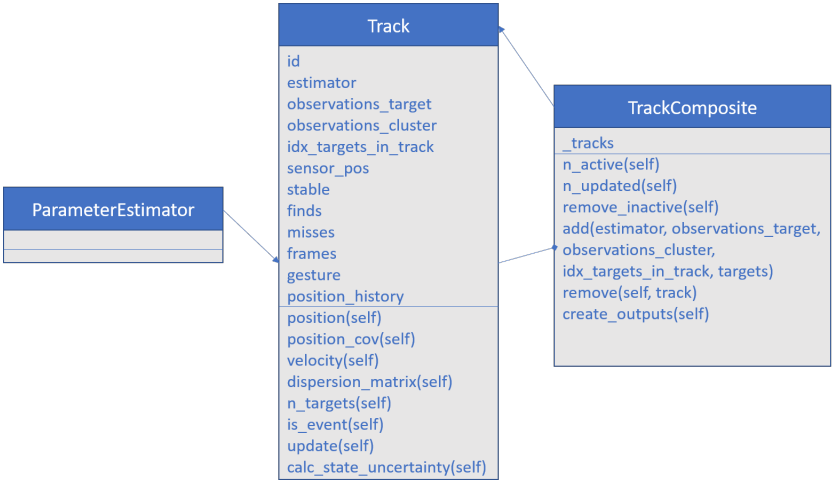



Figure A.22: Software architecture for the container of the detected tracks. $Track$ is a Python Dataclass that holds all attributes of a track. For concatenating multiple tracks, I used the composite pattern, implemented by $TrackComposite$.

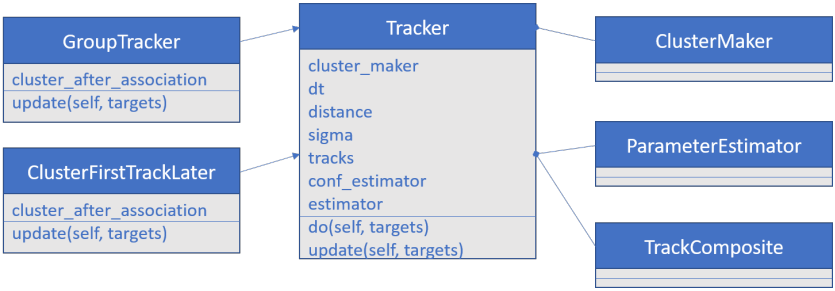



Figure A.23: Software architecture of the complete tracking processing chain. It combines the facade pattern so that interaction with multiple modules is possible and the template pattern. The template method $update$ is implemented by the two available subclasses.

### A.3.3 Radar Simulator

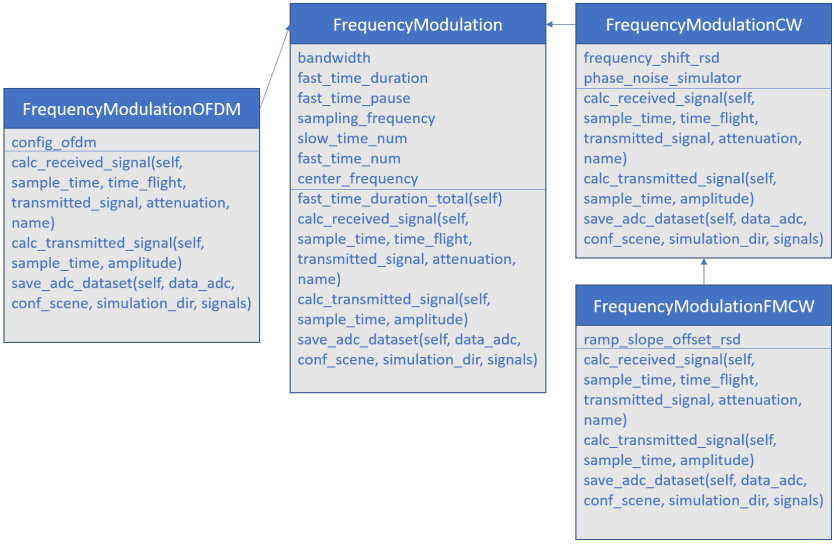



Figure A.24: Software architecture of the frequency modulation scheme, CW, FMCW and OFDM are available.

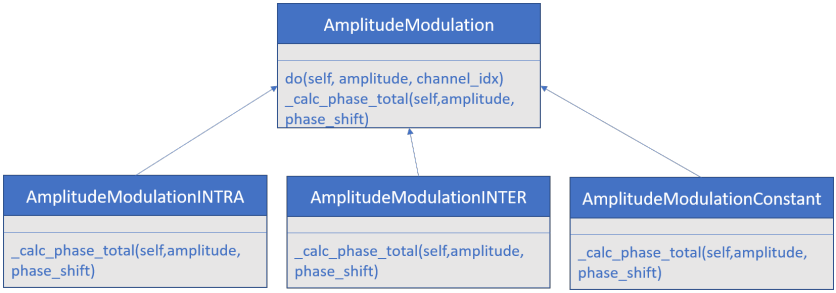



Figure A.25: Software architecture of the amplitude modulation scheme. User can select between constant amplitude for all chirps, different amplitude during chirp transmission (INTRA) or different amplitude for each chirp (INTER).

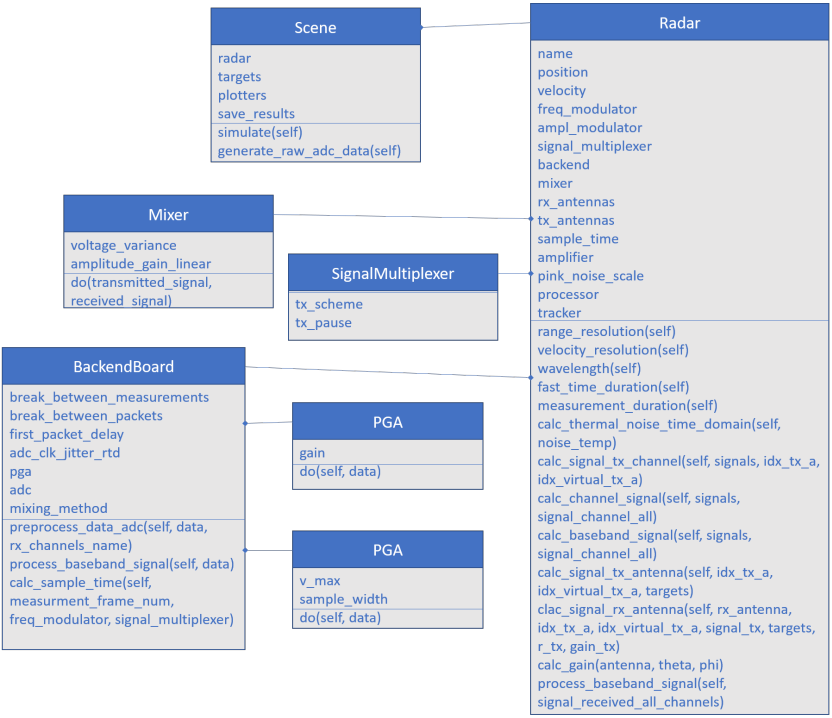



Figure A.26: Software architecture of the simulator platform. The *Scene* class contains all the attributes that the simulation needs, including the Radar and the point-targets. The *Radar* class consists of the *AmplitudeModulator*, the *FrequencyModulator*, the *BackendBoard*, *SignalMultiplexer* and the *Mixer*.

# Bibliography


[ABKS99] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[ALM11] Georgia Albuquerque, Thomas Lowe, and Marcus Magnor. Synthetic Generation of High-Dimensional Datasets. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2317–2324, 2011.

[AZS18] Moeness G. Amin, Zhengxin Zeng, and Tao Shan. Hand Gesture Recognition based on Radar Micro-Doppler Signature Envelopes. *arXiv preprint arXiv:1811.12467*, 2018.

[BGR+20] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. BlazePose: On-device Real-time Body Pose tracking. *arXiv preprint arXiv:2006.10204*, 2020.

[Bis06] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006.

[Bok18] Bokeh Development Team. *Bokeh: Python library for interactive visualization*. 2018.

[BRW17] Jonathan Bechter, Fabian Roos, and Christian Waldschmidt. Compensation of Motion-Induced Phase Errors in TDM MIMO Radars. *IEEE Microwave and Wireless Components Letters*, 27(12):1164–1166, December 2017.

[BSLK01] Yaakov Bar-Shalom, X.-Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, USA, 2001. DOI: 10.1002/0471221279.

[BTT90] Ronan Boulic, Nadia Magnenat Thalmann, and Daniel Thalmann. A global human walking model with real-time kinematic

personification. *The Visual Computer*, 6(6):344–358, November 1990.

[CA09] Ray Crozier and Lynn Alden. *Coping with shyness and social phobias: A guide to understanding and overcoming social anxiety*. Simon and Schuster, 2009.

[CD21] Han Cui and Naim Dahnoun. High Precision Human Detection and Tracking Using Millimeter-Wave Radars. *IEEE Aerospace and Electronic Systems Magazine*, 36(1):22–32, January 2021.

[CLFG] Zhaoxi Chen, Gang Li, Francesco Fioranelli, and Hugh Griffiths. Dynamic Hand Gesture Classification Based on Multistatic Radar Micro-Doppler Signatures Using Convolutional Neural Network. In *2019 IEEE Radar Conference (RadarConf)*, pages 1–5.

[CLHW06] Victor C. Chen, Fayin Li, S.-S. Ho, and Harry Wechsler. Micro-Doppler effect in radar: phenomenon, model, and simulation study. *IEEE Transactions on Aerospace and electronic systems*, 42(1):2–21, 2006.

[CML+19] Xiaodong Cai, Jingyi Ma, Wei Liu, Hemin Han, and Lili Ma. Efficient Convolutional Neural Network for FMCW Radar Based Hand Gesture Recognition. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, UbiComp/ISWC '19 Adjunct, pages 17–20, London, United Kingdom, 2019. Association for Computing Machinery.

[Com20] Blender Online Community. Blender - a 3d modelling and rendering package, 2020.

[CSBM18] Biplab Ketan Chakraborty, Debajit Sarma, Manas Kamal Bhuyan, and Karl F. MacDorman. Review of constraints on vision-based gesture recognition for human–computer interaction. *IET Computer Vision*, 12(1):3–15, 2018.

[CSKG17] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3d Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[EGA19] Baris Erol, Sevgi Z. Gurbuz, and Moeness G. Amin. GAN-based Synthetic Radar Micro-Doppler Augmentations for Improved Human Activity Recognition. In *2019 IEEE Radar Conference (RadarConf)*, pages 1–5. IEEE, 2019.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[EZO98] Miljko Eric, Aleksa Zejak, and Milorad ObradoviC. Ambiguity characterization of arbitrary antenna array: Type I ambiguity. In *1998 IEEE 5th International Symposium on Spread Spectrum Techniques and Applications-Proceedings. Spread Technology to Africa (Cat. No. 98TH8333)*, volume 2, pages 399–403. IEEE, 1998.

[FJ05] Matteo Frigo and Steven G. Johnson. The Design and Implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, 2016.

[GRC+19] Andrew Gigie, Smriti Rani, Arijit Chowdhury, Tapas Chakravarty, and Arpan Pal. An agile approach for human gesture detection using synthetic radar data. pages 558–564. ACM Press, 2019.

[GSN20] Qian Gao, Xukun Shen, and Wensheng Niu. Large-Scale Synthetic Urban Dataset for Aerial Scene Understanding. *IEEE Access*, 8:42131–42140, 2020.

[GWL19] Changzhan Gu, Jian Wang, and Jaime Lien. Motion Sensing Using Radar: Gesture Interaction and Beyond. *IEEE Microwave Magazine*, 20(8):44–57, August 2019.

[Hak18] Gor Hakobyan. *Orthogonal frequency division multiplexing multiple-input multiple-output automotive radar with novel signal processing algorithms*. PhD thesis, 2018.

[HHZZ16] Marlene Harter, Jurgen Hildebrandt, Andreas Ziroff, and Thomas Zwick. Self-Calibration of a 3-D-Digital Beamforming Radar System for Automotive Applications With Installation

Behind Automotive Covers. *IEEE Transactions on Microwave Theory and Techniques*, 64(9):2994–3000, September 2016.

[HLG+21] Eiji Hayashi, Jaime Lien, Nicholas Gillian, Leonardo Giusti, Dave Weber, Jin Yamanaka, Lauren Bedal, and Ivan Poupyrev. RadarNet: Efficient Gesture Recognition Technique Utilizing a Miniature Radar Sensor. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, Yokohama, Japan, 2021. Association for Computing Machinery.

[HMvdW+20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[HTS+12] Jürgen Hasch, Eray Topak, Raik Schnabel, Thomas Zwick, Robert Weigel, and Christian Waldschmidt. Millimeter-Wave Technology for Automotive Radar Sensors in the 77 GHz Frequency Band. *IEEE Transactions on Microwave Theory and Techniques*, 60(3):845–860, March 2012.

[HY19] Gor Hakobyan and Bin Yang. High-Performance Automotive Radar: A Review of Signal Processing Algorithms and Modulation Schemes. *IEEE Signal Processing Magazine*, 36(5):32–44, September 2019.

[IADW18] Karim Ishak, Nils Appenrodt, Jrgen Dickmann, and Christian Waldschmidt. Human motion training data generation for radar based deep learning applications. In *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4. IEEE, 2018.

[IADW19] Karim Ishak, Nils Appenrodt, Jürgen Dickmann, and Christian Waldschmidt. Advanced Radar Micro-Doppler Simulation Environment for Human Motion Applications. In *2019 IEEE Radar Conference (RadarConf)*, pages 1–6, 2019.

[Ins18] Texas Instuments. Tracking radar targets with multiple reflections, 2018.

[IS12] Edin Insanic and Paul R. Siqueira. A Maximum Likelihood Approach to Estimation of Vector Velocity in Doppler Radar Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 50(2):553–567, February 2012.

[KR14] Matthias Kronauge and Hermann Rohling. New chirp sequence radar waveform. *IEEE Transactions on Aerospace and Electronic Systems*, 50(4):2870–2877, October 2014.

[KT16] Youngwook Kim and Brian Toomajian. Hand Gesture Recognition Using Micro-Doppler Signatures With Convolutional Neural Network. *IEEE Access*, 4:7125–7130, 2016.

[Kuh05] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52(1):7–21, 2005.

[KUS03] Pavlina Konstantinova, Alexander Udvarev, and Tzvetan Semerdjiev. A study of a target tracking algorithm using global nearest neighbor approach. In *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech'03)*, pages 290–295, 2003.

[KV96] Hamid Krim and Mats Viberg. Two decades of array signal processing research: the parametric approach. *IEEE signal processing magazine*, 13(4):67–94, 1996.

[KW52] William H. Kruskal and W. Allen Wallis. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[LG20] Jaime Lien and Nicholas Gillian. Soli Radar-Based Perception and Interaction in Pixel 4, 2020.

[LGK+16] Jaime Lien, Nicholas Gillian, M. Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics (TOG)*, 35(4):142, 2016.

[LHYC18] Shengchang Lan, Zonglong He, Kai Yao, and Weichu Chen. Hand Gesture Recognition using a Three-dimensional 24 GHz Radar Array. In *IEEE/MTT-S International Microwave Symposium-IMS*, pages 138–140. IEEE, 2018.

[LLC+19] Bor-Shing Lin, I-Jung Lee, Pei-Ying Chiang, Shih-Yuan Huang, and Chih-Wei Peng. A Modular Data Glove System for Finger and Hand Motion Capture Based on Inertial Sensors. *Journal of Medical and Biological Engineering*, 39(4):532–540, August 2019.

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[LPS+11] Liang Liu, Mihail Popescu, Marjorie Skubic, Marilyn Rantz, Tarik Yardibi, and Paul Cuddihy. Automatic fall detection based on Doppler radar motion signature. In *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 222–225, 2011.

[LSF+18] Mingkang Li, Martin Stolz, Zhaofei Feng, Martin Kunert, Roman Henze, and Ferit Küçükay. An adaptive 3d grid-based clustering algorithm for automotive high resolution radar sensor. In *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 1–7. IEEE, 2018.

[LY11] Oliver Lange and Bin Yang. Antenna geometry optimization for 2d direction-of-arrival estimation for radar imaging. In *2011 International ITG Workshop on Smart Antennas*, pages 1–8. IEEE, 2011.

[LZRG17] Gang Li, Rui Zhang, Matthew Ritchie, and Hugh Griffiths. Sparsity-based dynamic hand gesture recognition using micro-Doppler signatures. In *IEEE Radar Conference (RadarConf)*, pages 0928–0931. IEEE, 2017.

[MBS+20] Sandro De Paula Mendonca, Yvan Pereira Dos Santos Brito, Carlos Gustavo Resque Dos Santos, Rodrigo Do Amor Divino Lima, Tiago Davi Oliveira De Araujo, and Bianchi Serique Meiguins. Synthetic Datasets Generator for Testing Information Visualization and Machine Learning Techniques and Tools. *IEEE Access*, 8:82917–82928, 2020.

[MCN14] Adrian Macaveiu, Andrei Campeanu, and Ioan Nafornita. Kalman-based tracker for multiple radar targets. In *10th International Conference on Communications (COMM)*, pages 1–4. IEEE, 2014.

[MYG+16] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3d Convolutional Neural Networks. pages 4207–4215. IEEE, June 2016.

[NHAZ21] Alexandros Ninos, Jürgen Hasch, Mario Emilio Pizano Alvarez, and Thomas Zwick. Synthetic Radar Dataset Generator for Macro-Gesture Recognition. *IEEE Access*, 9:76576–76584, 2021.

[NHHZ22] Alexandros Ninos, Jurgen Hasch, Michael Heizmann, and Thomas Zwick. Radar-Based Robust People Tracking and Consumer Applications. *IEEE Sensors Journal*, 22(4):3726–3735, February 2022.

[NHZ21a] Alexandros Ninos, Jürgen Hasch, and Thomas Zwick. Multi-User Macro Gesture Recognition using mmWave Technology. In *EuRAD*, 2021.

[NHZ21b] Alexandros Ninos, Jürgen Hasch, and Thomas Zwick. Real-Time Macro Gesture Recognition using Efficient Empirical Feature Extraction with Millimeter-Wave Technology. *IEEE Sensors Journal*, 2021.

[noa08] *1588-2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.* 2008. OCLC: 958709955.

[OUY17] Monika Ouza, Michael Ulrich, and Bin Yang. A simple radar simulation tool for 3d objects based on blender. In *2017 18th International Radar Symposium (IRS)*, pages 1–10. IEEE, 2017.

[PMR21] Jacopo Pegoraro, Francesca Meneghello, and Michele Rossi. Multiperson Continuous Tracking and Identification From mm-Wave Micro-Doppler Signatures. *IEEE Transactions on Geoscience and Remote Sensing*, 59(4):2994–3009, April 2021.

[PPVT19] Srdan Popic, Bogdan Pavkovic, Ivan Velikic, and Nikola Teslic. Data generators: a short survey of techniques and use cases with focus on testing. pages 189–194. IEEE, September 2019.

[RGG+21] Philipp Ritter, Michael Geyer, Tilman Gloekler, Xiaolei Gai, Thomas Schwarzenberger, Gregor Tretter, Yikun Yu, and Guenter Vogel. A Fully Integrated 78 GHz Automotive Radar System-

an-Chip in 22nm FD-SOI CMOS. In *2020 17th European Radar Conference (EuRAD)*, pages 57–60. IEEE, 2021.

[RHK] Manjunath Thindlu Rudrappa, Reinhold Herschel, and Peter Knott. Distinguishing living and non living subjects in a scene based on vital parameter estimation. In *2020 17th European Radar Conference (EuRAD)*, pages 53–56.

[Ric14] M. A. Richards. *Fundamentals of radar signal processing*. McGraw-Hill Education, New York, second edition, 2014.

[RJ19] Matthew Ritchie and Aaron M Jones. Micro-Doppler Gesture Recognition using Doppler, Time and Range Based Features. pages 1–6. IEEE, April 2019.

[SAHLE19] Sruthy Skaria, Akram Al-Hourani, Margaret Lech, and Robin J. Evans. Hand-Gesture Recognition Using Two-Antenna Doppler Radar With Deep Convolutional Neural Networks. *IEEE Sensors Journal*, 19(8):3041–3048, April 2019.

[SBL14] Hongbo Sun, Frederic Brigui, and Marc Lesturgie. Analysis and comparison of MIMO radar waveforms. In *International Radar Conference*, pages 1–6. IEEE, 2014.

[SFGP19] Yuliang Sun, Tai Fei, Shangyin Gao, and Nils Pohl. Automatic Radar-based Gesture Detection and Classification via a Region-based Deep Convolutional Neural Network. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4300–4304. IEEE, 2019.

[SFL+20] Yuliang Sun, Tai Fei, Xibo Li, Alexander Warnecke, Ernst Warsitz, and Nils Pohl. Real-Time Radar-Based Gesture Detection and Recognition Built in an Edge-Computing Platform. *IEEE Sensors Journal*, pages 1–1, 2020.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[SSM+17] Martin Stolz, Eugen Schubert, Frank Meinl, Martin Kunert, and Wolfgang Menzel. Multi-target reflection point model of cyclists for automotive radar. In *European Radar Conference (EURAD)*, pages 94–97. IEEE, 2017.

[SVG+20] SciPy 1.0 Contributors, Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020.

[SW65] S. S. Shapiro and M. B. Wilk. An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, 52(3/4):591, December 1965.

[THCD16] R. P. Trommel, R. I. A. Harmanny, L. Cifola, and J. N. Driessen. Multi-target human gait classification using deep convolutional neural networks on micro-doppler spectrograms. In *European Radar Conference (EuRAD)*, pages 81–84. IEEE, 2016.

[TWJ+21] Saverio Trotta, Dave Weber, Reinhard W. Jungmaier, Ashutosh Baheti, Jaime Lien, Dennis Noppeney, Maryam Tabesh, Christoph Rumpler, Michael Aichner, Siegfried Albel, Jagjit Singh Bal, and Ivan Poupyrev. 2.3 SOLI: A Tiny Device for a New Human Machine Interface. pages 42–44. IEEE, February 2021.

[VKC19] Junia Valente, Keerthi Koneru, and Alvaro Cardenas. Privacy and Security in Internet-Connected Cameras. pages 173–180. IEEE, July 2019.

[VRD+20] Claudia Vasanelli, Fabian Roos, Andre Durr, Johannes Schlichenmaier, Philipp Hugler, Benedikt Meinecke, Maximilian Steiner, and Christian Waldschmidt. Calibration and Direction-of-Arrival Estimation of Millimeter-Wave Radars: A Practical Introduction. *IEEE Antennas and Propagation Magazine*, 2020.

[Wag18] Thomas Wagner. *Tracking and Feature Extraction using a Short-Range Radar with Focus on Pedestrian Observation and Computational Efficiency*. PhD thesis, 2018.

[WFS15] Thomas Wagner, Reinhard Feger, and Andreas Stelzer. Modification of DBSCAN and application to range/Doppler/DoA measurements for pedestrian recognition with an automotive radar system. pages 269–272. IEEE, September 2015.

[WFS16] Thomas Wagner, Reinhard Feger, and Andreas Stelzer. A fast grid-based clustering algorithm for range/Doppler/DoA measurements. In *2016 European Radar Conference (EuRAD)*, pages 105–108. IEEE, 2016.

[WFS17] Thomas Wagner, Reinhard Feger, and Andreas Stelzer. Radar Signal Processing for Jointly Estimating Tracks and Micro-Doppler Signatures. *IEEE Access*, 5:1220–1238, 2017.

[WFS18] Thomas Wagner, Reinhard Feger, and Andreas Stelzer. Modifications of the OPTICS Clustering Algorithm for Short-Range Radar Tracking Applications. In *2018 15th European Radar Conference (EuRAD)*, pages 91–94. IEEE, 2018.

[WHM21] Christian Waldschmidt, Juergen Hasch, and Wolfgang Menzel. Automotive Radar — From First Efforts to Future Systems. *IEEE Journal of Microwaves*, 1(1):135–148, 2021.

[Wil45] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80, December 1945.

[Win07] Volker Winkler. Range Doppler detection for automotive FMCW radars. In *European Radar Conference*, pages 166–169. IEEE, 2007.

[WJZ+19] Yong Wang, Xiuqian Jia, Mu Zhou, Xiaolong Yang, and Zengshan Tian. Rammar: RAM Assisted Mask R-CNN for FMCW Sensor Based HGD System. pages 1–6. IEEE, May 2019.

[WSL+16] Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev, and Otmar Hilliges. Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum. pages 851–860. ACM Press, 2016.

[WWZ+19] Yong Wang, Shasha Wang, Mu Zhou, Qing Jiang, and Zengshan Tian. TS-I3d Based Hand Gesture Recognition Method With Radar Sensor. *IEEE Access*, 7:22902–22913, 2019.

[YMR+18] Hui-Shyong Yeo, Ryosuke Minami, Kirill Rodriguez, George Shaker, and Aaron Quigley. Exploring Tangible Interactions

with Radar Sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(4):1–25, December 2018.

[ZLW] Feifei Zhou, Xiangyu Li, and Zhihua Wang. Efficiently User-Independent Ultrasonic-Based Gesture Recognition Algorithm. In *2019 IEEE SENSORS*, pages 1–4. IEEE.

[ZLW+19] Peijun Zhao, Chris Xiaoxuan Lu, Jianan Wang, Changhao Chen, Wei Wang, Niki Trigoni, and Andrew Markham. mid: Tracking and identifying people with millimeter wave radar. In *15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 33–40. IEEE, 2019.

[ZM15] Paul Zarchan and Howard Musoff, editors. *Fundamentals of Kalman Filtering: A Practical Approach, Fourth Edition*. American Institute of Aeronautics and Astronautics, Inc., Reston, VA, January 2015. DOI: 10.2514/4.102776.

[ZR12] Alex Zwanetski and Hermann Rohling. Continuous wave MIMO radar based on time division multiplexing. In *13th International Radar Symposium*, pages 119–121. IEEE, 2012.

[ZTZ18] Zhenyuan Zhang, Zengshan Tian, and Mu Zhou. Latern: Dynamic Continuous Hand Gesture Recognition Using FMCW Radar Sensor. *IEEE Sensors Journal*, 18(8):3278–3289, April 2018.

# List of Own Publications

## Journals

[1] Alexandros Ninos, Jürgen Hasch, Mario Emilio Pizano Alvarez, and Thomas Zwick. Synthetic Radar Dataset Generator for Macro-Gesture Recognition. *IEEE Access*, 9:76576–76584, 2021.

[2] Alexandros Ninos, Jürgen Hasch, and Thomas Zwick. Real-Time Macro Gesture Recognition using Efficient Empirical Feature Extraction with Millimeter-Wave Technology. *IEEE Sensors Journal*, pages 1–1, 2021.

[3] Alexandros Ninos, Jurgen Hasch, Michael Heizmann, and Thomas Zwick. Radar-Based Robust People Tracking and Consumer Applications. *IEEE Sensors Journal*, 22(4):3726–3735, February 2022.

## Conferences

[4] Alexandros Ninos, Jürgen Hasch, and Thomas Zwick. Multi-User Macro Gesture Recognition using mmWave Technology. In *EuRAD*, 2021.

**Forschungsberichte aus dem Institut für Höchstfrequenztechnik und Elektronik (IHE) der Universität Karlsruhe (TH)**
(ISSN 0942-2935)

---

Die Bände 1 (1992) bis 55 (2008) der Schriftenreihe können über das Institut Hochfrequenztechnik und Elektronik bestellt werden (https://www.ihe.kit.edu).

Fortführung als:

**Karlsruher Forschungsberichte aus dem Institut für Hochfrequenztechnik und Elektronik**
(ISSN 1868-4696)

---

Band 55 Sandra Knörzer
Funkkanalmodellierung für OFDM-Kommunikationssysteme bei Hochgeschwindigkeitszügen
ISBN 978-3-86644-361-7

Band 56 Thomas Fügen
Richtungsaufgelöste Kanalmodellierung und Systemstudien für Mehrantennensysteme in urbanen Gebieten
ISBN 978-3-86644-420-1

Band 57 Elena Pancera
Strategies for Time Domain Characterization of UWB Components and Systems
ISBN 978-3-86644-417-1

Band 58 Jens Timmermann
Systemanalyse und Optimierung der Ultrabreitband-Übertragung
ISBN 978-3-86644-460-7

---

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 59 Juan Pontes
**Analysis and Design of Multiple Element Antennas for Urban Communication**
ISBN 978-3-86644-513-0

Band 60 Andreas Lambrecht
**True-Time-Delay Beamforming für ultrabreitbandige Systeme hoher Leistung**
ISBN 978-3-86644-522-2

Band 61 Grzegorz Adamiuk
**Methoden zur Realisierung von dual-orthogonal, linear polarisierten Antennen für die UWB-Technik**
ISBN 978-3-86644-573-4

Band 62 Jutta Kühn
**AlGaN/GaN-HEMT Power Amplifiers with Optimized Power-Added Efficiency for X-Band Applications**
ISBN 978-3-86644-615-1

Band 63 Małgorzata Janson
**Hybride Funkkanalmodellierung für ultrabreitbandige MIMO-Systeme**
ISBN 978-3-86644-639-7

Band 64 Mario Pauli
**Dekontaminierung verseuchter Böden durch Mikrowellenheizung**
ISBN 978-3-86644-696-0

Band 65 Thorsten Kayser
**Feldtheoretische Modellierung der Materialprozessierung mit Mikrowellen im Durchlaufbetrieb**
ISBN 978-3-86644-719-6

Band 66 Christian Andreas Sturm
**Gemeinsame Realisierung von Radar-Sensorik und Funkkommunikation mit OFDM-Signalen**
ISBN 978-3-86644-879-7

Band 67 Huaming Wu
**Motion Compensation for Near-Range Synthetic Aperture Radar Applications**
ISBN 978-3-86644-906-0

Band 68 Friederike Brendel
**Millimeter-Wave Radio-over-Fiber Links based on Mode-Locked Laser Diodes**
ISBN 978-3-86644-986-2

Band 69 Lars Reichardt
**Methodik für den Entwurf von kapazitätsoptimierten Mehrantennensystemen am Fahrzeug**
ISBN 978-3-7315-0047-6

Band 70 Stefan Beer
**Methoden und Techniken zur Integration von 122 GHz Antennen in miniaturisierte Radarsensoren**
ISBN 978-3-7315-0051-3

Band 71 Łukasz Zwirełło
**Realization Limits of Impulse-Radio UWB Indoor Localization Systems**
ISBN 978-3-7315-0114-5

Band 72 Xuyang Li
**Body Matched Antennas for Microwave Medical Applications**
ISBN 978-3-7315-0147-3

Band 73 Sebastian Diebold
**Transistor- und Leitungsmodellierung zum Entwurf von monolithisch integrierten Leistungsverstärkern für den hohen Millimeterwellen-Frequenzbereich**
ISBN 978-3-7315-0161-9

Band 74 Christian Rusch
**Integrierte, planare Leckwellenantennen für 3D-Millimeterwellen-Radarsysteme basierend auf dem holografischen Prinzip**
ISBN 978-3-7315-0234-0

---

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 75 Marlene Harter
**Dreidimensional bildgebendes Radarsystem mit digitaler Strahlformung für industrielle Anwendungen**
ISBN 978-3-7315-0249-4

Band 76 Michael A. Baldauf
**Abhängigkeit der Exposition von der Zellgröße beim Mobilfunk unter Gewährleistung der Versorgung**
ISBN 978-3-7315-0308-8

Band 77 Alicja Ossowska
**Highly Resolved Synthetic Aperture Radar with Beam Steering**
ISBN 978-3-7315-0315-6

Band 78 Małgorzata Dominika Brzeska
**RF Modelling and Characterization of Tyre Pressure Sensors and Vehicle Access Systems**
ISBN 978-3-7315-0348-4

Band 79 Ulrich Lewark
**Aktive Frequenzvervielfacher zur Signalerzeugung im Millimeter- und Submillimeterwellen Frequenzbereich**
ISBN 978-3-7315-0354-5

Band 80 Kai-Philipp Walter Pahl
**Distributed Transformers for Broadband Monolithic Millimeter-Wave Integrated Power Amplifiers**
ISBN 978-3-7315-0409-2

Band 81 Serdal Ayhan
**Hochgenaue radarbasierte Abstandsmessung mit geführter Wellenausbreitung**
ISBN 978-3-7315-0433-7

Band 82 Yoke Leen Sit
**MIMO OFDM Radar-Communication System with Mutual Interference Cancellation**
ISBN 978-3-7315-0599-0

Band 83 Steffen Scherr
**FMCW-Radarsignalverarbeitung zur Entfernungsmessung mit hoher Genauigkeit**
ISBN 978-3-7315-0607-2

Band 84 Tom Schipper
**Modellbasierte Analyse des Interferenzverhaltens von Kfz-Radaren**
ISBN 978-3-7315-0639-3

Band 85 Malyhe Jalilvand
**Application-Specific Broadband Antennas for Microwave Medical Imaging**
ISBN 978-3-7315-0664-5

Band 86 Benjamin Göttel
**Millimeterwellen On-Chip Antennensysteme für die Integration in SoC Applikationen**
ISBN 978-3-7315-0667-6

Band 87 Christian Arnold
**Im Orbit einstellbare Ausgangsfilter und -multiplexer**
ISBN 978-3-7315-0722-2

Band 88 Tobias Mahler
**Synthese kapazitätsoptimierter Antennensysteme mit messtechnischer Verifikation**
ISBN 978-3-7315-0737-6

Band 89 Daniel Müller
**RF Probe-Induced On-Wafer Measurement Errors in the Millimeter-Wave Frequency Range**
ISBN 978-3-7315-0822-9

Band 90 Tristan Visentin
**Polarimetric Radar for Automotive Applications**
ISBN 978-3-7315-0888-5

Band 91 Christian von Vangerow
**Entwurf und Modellierung von Breitbandverstärkern mit variablem Gewinn in SiGe BiCMOS Technologien**
ISBN 978-3-7315-0910-3

Band 92 Mekdes Girma
**Concepts for Short Range Millimeter-wave Miniaturized Radar Systems with Built-in Self-Test**
ISBN 978-3-7315-0938-7

Band 93 Akanksha Bhutani
**Low Temperature Co-fired Ceramics for System-in-Package Applications at 122 GHz**
ISBN 978-3-7315-0945-5

Band 94 Jochen Schäfer
**Oberflächenwellenerzeuger für Millimeterwellen-Leckwellenantennen**
ISBN 978-3-7315-0962-2

Band 95 Jerzy Kowalewski
**Capacity Enhancement by Pattern-Reconfigurable Multiple Antenna Systems in Vehicular Applications**
ISBN 978-3-7315-0997-4

Band 96 Christian Weber
**Verfahren zur automatischen Spektralanalyse für die Optimierung drahtloser Kommunikation und Sensorik**
ISBN 978-3-7315-1014-7

Band 97 Florian Boes
**Breitbandige Frequenzweichen für die Parallelisierung von Millimeterwellen-Messtechnik**
ISBN 978-3-7315-1078-9

Band 98 Jonathan Mayer
**Kapazitiv gekoppelte Streifenleitungsantennen für Millimeterwellenanwendungen**
ISBN 978-3-7315-1111-3

Band 99 Benjamin Nuß
**Frequenzkamm-basiertes breitbandiges MIMO-OFDM-Radar**
ISBN 978-3-7315-1203-5

Band 100 Joerg Andreas Eisenbeis
**Hybride Beamformingsysteme niedriger Komplexität für den Mobilfunk**
ISBN 978-3-7315-1184-7

Band 101 Alexandros Ninos
**Multi-User Gesture Recognition with Radar Technology**
ISBN 978-3-7315-1225-7

**Karlsruher Forschungsberichte aus dem Institut für Hochfrequenztechnik und Elektronik**

*Herausgeber: Prof. Dr.-Ing. Thomas Zwick*
*Prof. Dr. Ing. Ahmet Cagri Ulusoy*

The presence of the Internet of Things is imminent, which means that soon the majority of the devices in a smart home will be part of a system that senses and reacts to a user's commands. Such a system would require information about the number of people in the room and a touchless human-machine interface for purposes that range from entertainment to hygiene. Millimeter wave Radar sensors provide an ideal solution since they do not raise privacy concerns, are relatively cheap, consume low power for detection and processing, and can be hidden behind plastic casings. The aim of this work is the development of such a Radar system for consumer applications.

Alexandros Ninos received his Diploma in Electrical and Computer Engineering from the National Technical University of Athens in 2014. His research focused on millimeter wave Radars for the consumer market in the Corporate Sector Research and Advance Engineering of Robert Bosch GmbH in Stuttgart, Germany. Simultaneously, he pursued his doctoral degree at Karlsruhe Institute of Technology, Germany.

ISSN 1868-4696
ISBN 978-3-7315-1225-7