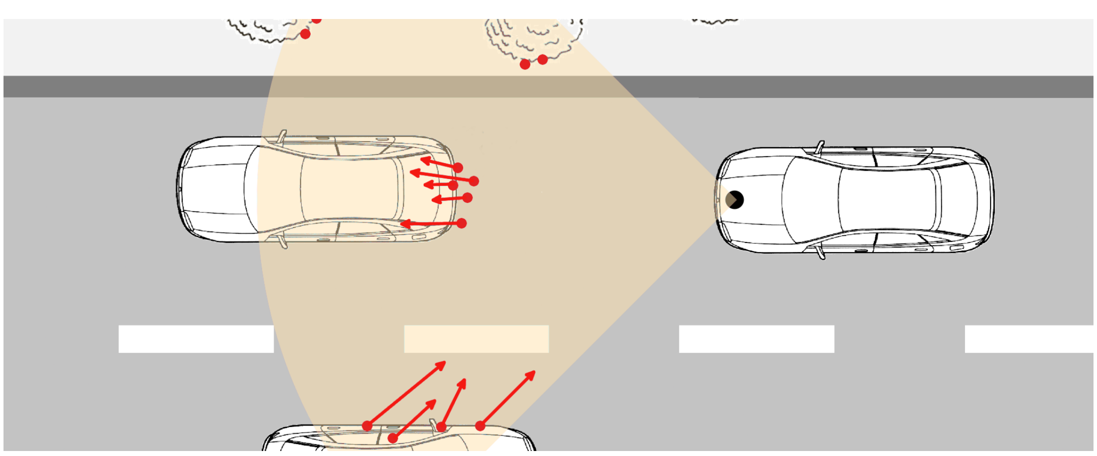


Exploring Millimeter-Wave Radar Algorithms for Obstacle Detection and Tracking

Rafael Pedro Carvalho

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. Alberto Manuel Martinho Vale

Examination Committee

June 2024

Declaration

Acknowledgments

I would like to thank everyone who has supported me throughout these years at university, to all the friends and family who helped me keep going even when things were rough. Without them, this journey would not have been possible.

I would also want to thank Professor Alberto Vale for the guidance during this last stretch and for helping me stay motivated and keep a positive attitude, even when things didn't work out as planned.

Abstract

Robust real-time obstacle detection and tracking remain significant challenges in the automotive industry. Historically, sensors like LiDAR and cameras have dominated this space, but recent technological advancements have enabled the resurgence of radar technology, specifically millimeter-wave (mmWave) radar sensors.

MmWave radar sensors prove to be more robust to adverse weather conditions than other commonly used sensors. This work focuses on developing an obstacle detection and tracking algorithm that relies exclusively on data from these radars. The developed algorithm starts by accumulating recent frames of radar data and then employs DBSCAN for obstacle detection and Kalman Filters for tracking. Results indicate that this approach effectively tracks various types of obstacles. However, it demonstrates superior performance in tracking obstacles moving away from the sensor compared to those moving toward it. This discrepancy arises from the intrinsic properties of radar technology. Nonetheless, with proper radar configuration and processing, this difference can be mitigated.

This approach serves as a method for evaluating the capabilities of mmWave radar systems in obstacle detection and tracking while robust Neural Networks trained specifically for this problem are not yet feasible, mainly due to the extensive data requirements for proper training.

Keywords

ADAS; Radar; Millimeter-Wave; Obstacle; Detection; Tracking.

Resumo

A deteção e o seguimento de obstáculos em tempo real continuam a ser desafios importantes na indústria automóvel. Tradicionalmente, sensores como os LiDARs e as câmaras dominaram este espaço, no entanto, avanços tecnológicos recentes despertaram um novo interesse na tecnologia de radar, mais especificamente nos sensores de radar millimeter-wave (mmWave).

Os sensores de radar mmWave mostram ser mais robustos a condições climatéricas adversas que outros sensores tipicamente usados. Este trabalho foca-se no desenvolvimento de um algoritmo de deteção e seguimento de obstáculos que usa somente dados provenientes destes radares. O algoritmo desenvolvido acumula alguns frames recentes de dados destes radares, de seguida usa DBSCAN para a deteção de obstáculos e Filtros de Kalman para o seguimento. Os resultados obtidos mostram que esta abordagem é eficaz a seguir diferentes tipos de obstáculos. No entanto, mostra existir uma maior capacidade em seguir obstáculos que se afastam do sensor comparativamente a obstáculos que se aproximam do sensor. Esta discrepância é causada por algumas características intrínsecas da tecnologia dos radares. Ainda assim, com uma configuração adequada do radar e com algum processamento dos dados, esta diferença pode ser atenuada.

Esta abordagem serve como uma maneira de avaliar as capacidades dos radares mmWave em sistemas de deteção e seguimento de obstáculos enquanto Redes Neuronais robustas treinadas para este problema em específico ainda não estão prontas, principalmente devido à grande quantidade de dados necessários para um treino eficaz.

Palavras Chave

ADAS; Radar; Millimeter-Wave; Obstáculo; Deteção; Seguimento.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
AoA	Angle of Arrival
CAS	Colision Avoidance System
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
EBA	Emergency Brake Assist
EKF	Extended Kalman Filter
FFT	Fast Fourier Transform
FMCW	Frequency-Modulated Continuous Wave
FoV	Field Of View
GPS	Global Positioning System
HAR	Human Activity Recognition
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LKA	Lane Keeping Assistance
mmWave	Millimeter Wave
RADAR	Radio Detection And Ranging
ROS	Robot Operating System
Rx	Receiving Antennas
SMC	Sequential Monte Carlo
SONAR	Sound Navigation And Ranging
STC	Spacial-Temporal Clustering
UKF	Unscented Kalman Filter

UTM	Universal Transverse Mercator

1 Introduction

Contents

The research and development of autonomous vehicles can be attributed to several key factors. One of the primary driving forces is the still alarmingly high number of deaths and injuries resulting from road traffic accidents, which continue to pose a serious problem globally. The World Health Organization (WHO) estimates there have been 1.19 million road traffic deaths in 2021; this corresponds to a rate of around 15 road traffic deaths per $100,000$ population, and about 66% of these fatalities occur among the working age population, i.e., people aged between 18 and 59 years [1]. These statistics underscore the urgent need for the implementation of safer transportation systems.

Furthermore, the ever-increasing population residing in urban areas contributes to the increasing complexity of existing road networks and infrastructure. The growing number of vehicles on the roads and limited space and resources impose significant challenges on transportation systems. Consequently, finding innovative solutions to alleviate the strain on current infrastructure becomes crucial to ensure the efficient and safe movement of people and goods. Moreover, using automated systems for tasks like driving offers the potential for improved comfort and convenience for the general population, which can lead to better mobility experiences, reduced stress levels, and increased productivity during journeys.

Advancing this field holds promise in terms of mitigating the devastating impact of road traffic accidents and addressing the evolving demands of modern transportation systems.

1.1 Levels of Vehicle Autonomy

Vehicle autonomy refers to the degree to which a vehicle can operate independently without human intervention. The Society of Automotive Engineers (SAE) has established five levels of autonomy [2], representing a continuum of capabilities ranging from fully manual to fully autonomous driving.

Level 0: No Automation At Level 0, there is no automation present in the vehicle, although the vehicle may have some momentary assistance features, such as Automatic Emergency Braking or Blind Spot Warnings;

Level 1: Driver Assistance At Level 1, there is some driver assistance in the vehicle. For example, the vehicle may have a system to assist the driver with either steering or accelerating and decelerating, such as Lane Keeping Assistance (LKA) or Adaptive Cruise Control (ACC). These features are qualified as Level 1 because the human driver is still required to monitor the other aspects of driving, such as steering and braking, and must remain attentive at all times.

Level 2: Partial Driving Automation At Level 2, the vehicle can take control of both steering and accelerating and decelerating in certain situations, such as highway driving or stop-and-go traffic. However, the driver is still responsible for monitoring the vehicle's performance and must be prepared to take control at any time.

Level 3: Conditional Driving Automation At Level 3, the vehicle can operate autonomously under certain conditions, such as clear weather or well-marked roads. Level 3 vehicles have “environmental detection” capabilities and can make informed decisions for themselves, such as accelerating past a slow-moving vehicle. However, the driver must still be present and ready to take control if necessary. In some cases, the driver may be able to engage in other tasks while the vehicle is operating autonomously.

Level 4: High Driving Automation At Level 4, the vehicle can operate autonomously in most situations without human intervention. However, there may be certain situations where the vehicle requires human intervention, such as in extreme weather conditions or on unmapped roads.

Level 5: Full Driving Automation At Level 5, the vehicle can operate autonomously in all situations without human intervention. Level 5 cars won’t even require steering wheels or pedals. They will be free from any virtual limitations, such as the need to drive on a road, and they should be able to go anywhere and do anything that an experienced human driver can do.

In recent years, significant effort has been put into developing Advanced Driver Assistance Systems (ADAS). These systems are targeted at the automotive industry and aim to increase road safety either by automating specific driving tasks such as LKA, Emergency Brake Assist (EBA), and ACC or by providing real-time alerts and warnings to the driver about potential dangers on the road.

While the system developed in this thesis is not able to provide great assistance to the driver by itself, its outputs can serve as the basis to create a Colision Avoidance System (CAS), which is a Level 2 ADAS.

1.2 Sensor Technologies

An essential factor in the safe operation of these vehicles is their ability to perceive and interpret the surrounding environment accurately. Different technologies are already being used in the automotive industry today. These serve as the vehicle’s eyes and ears, enabling it to perceive obstacles and make informed decisions.

1.2.1 Ultrasonic Sensors

Ultrasonic sensors derive their fundamental principles from the early 20th century Sound Navigation And Ranging (SONAR) technology initially developed for naval purposes. This technology has since evolved to suit various applications beyond underwater environments. Unlike SONAR, which excels in aquatic environments for tasks such as submarine navigation and detecting underwater threats, ultrasonic sensors are designed for air and surface environments, making them suitable for ADAS systems.

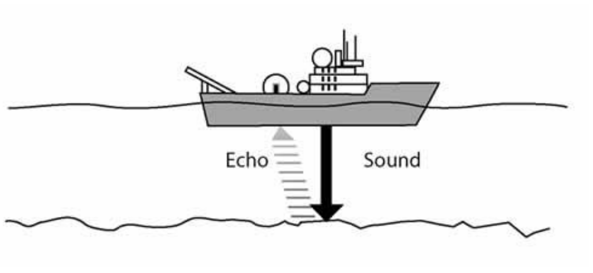


Figure 1.1: Example of a Sonar being used on a boat. [3]

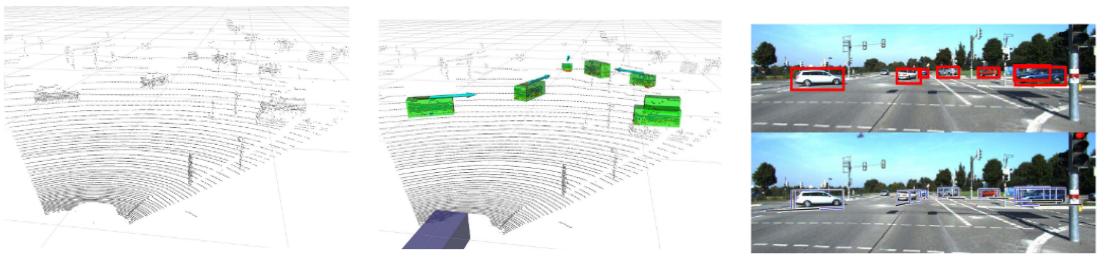
These sensors operate by emitting sound waves and measuring the time it takes for these waves to echo back from nearby objects (Fig. 1.1). Ultrasonic sounds suffer significant attenuation when traveling through the air, resulting in a shorter detection range when compared to other sensors. This technology is commonly used in automotive vehicles to support parking assistance and close-range collision avoidance systems.

The versatility of ultrasonic sensors has allowed them to be adapted for various civilian applications beyond automotive uses, including industrial automation, object sorting, and robotic sensing.

1.2.2 LiDAR

Another sensor commonly used in autonomous vehicles is the Light Detection and Ranging (LiDAR) sensor. LiDAR sensors were initially developed for remote sensing applications in the 1960s, and their use has since expanded to various fields, including the automotive industry.

These sensors work by emitting laser pulses and measuring the time it takes for these pulses to bounce back after hitting an object. By rapidly changing the emitter's position, it is possible to generate a 3D map of the scene, providing highly detailed information about the shape, size, and distance of objects.



(a) LiDAR data of a road scenario. **(b)** Result of the tracking method. **(c)** RGB images with the detected obstacles' bounding boxes.

Figure 1.2: These images illustrate the results of an obstacle tracking method that uses 3D LiDAR point clouds, described in [4].

Fig. 1.2(a) shows the data measured by a LiDAR sensor in a road scenario. Figs. 1.2(b) and 1.2(c) show the results of bounding box estimation using LiDAR data, described in [4].

While LiDARs have been widely used in the automotive industry for obstacle detection and mapping, their effectiveness still has some limitations. Besides, it is known that difficult weather conditions, namely rain, can heavily degrade the quality of the readings from LiDAR sensors [5] [6].

1.2.3 Cameras

Cameras are cheap, compact, and offer excellent imaging capabilities. In autonomous vehicles, camera-based perception systems play a vital role in understanding the surrounding environment. Two main camera technologies are being used in today's systems: Red-Green-Blue (RGB) cameras and Depth cameras. These systems can be further categorized into monocular and stereo camera systems.

Monocular systems utilize a single camera, either a RGB or a Depth camera. These systems rely on algorithms to extract useful information from the data provided by the camera, which can then be used to estimate the sizes, positions, and shapes of objects in the scene. Some of the more advanced techniques involve the usage of Machine Learning techniques to segment the data from the camera into different sections, typically separating the road from the obstacles as seen in Fig. 1.3.

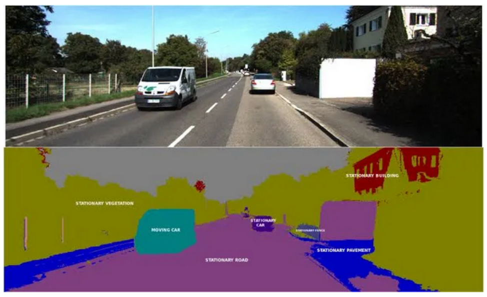


Figure 1.3: Example of the result of segmentation on a typical road scenario.

Stereo camera systems, on the other hand, employ two or more cameras with a known baseline separation. By comparing the images captured from different viewpoints, stereo vision algorithms can estimate the depths of the scene. These systems use the same techniques as monocular systems. However, the additional number of cameras can improve the quality of the results, although it comes at the cost of increasing the complexity of the required algorithms [7].

Camera systems are more sensitive to environmental factors than LiDARs due to their reliance on visible light. In adverse conditions, such as fog or rain, the visibility of objects captured by cameras may be significantly reduced, decreasing the system's image quality and accuracy. Additionally, variations in lighting conditions, such as strong glare or low-light environments, can wash out important visual details from the resulting images.

1.2.4 mmWave Radar Sensor

Radio Detection And Ranging (RADAR)s are another crucial sensor used in autonomous vehicles for perceiving the surrounding environment. Initially developed for military applications during World War II, radar has been widespread in various applications, including the automotive industry.

RADARs operate by emitting electromagnetic waves and analyzing the reflected signals, resulting from the emitted signal being reflected after hitting an object (Figure 1.4). This technology allows the detection of objects and can provide valuable information about their position, velocity, and size.

The name Millimeter Wave (mmWave) comes from the fact that the wavelength of the electromagnetic waves emitted by these sensors ranges typically from 1 to 10 millimeters.

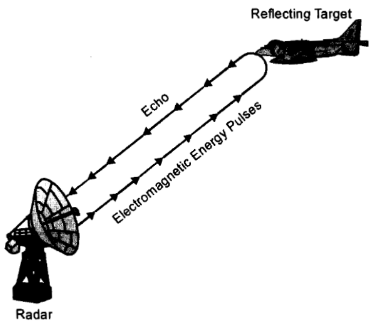


Figure 1.4: Example of a Radar detecting an approaching airplane.

One of the primary advantages of mmWave radars is their ability to function effectively in various weather conditions, including fog, rain, and snow, which can pose challenges to other sensing technologies. Due to their reliance on lower-frequency electromagnetic waves, radar signals can propagate in these challenging environments, enabling detection and perception even in adverse conditions, as demonstrated in [8].

1.2.5 Comparison of Technologies in ADAS

In the context of ADAS, the currently predominant technologies are the cameras, due to their ability to capture detailed visual information, and the LiDAR, due to their ability to quickly and accurately reconstruct a 3D scene. However, despite their long history and extensive research, some fundamental difficulties are associated with each of these technologies.

Meanwhile, mmWave RADARs, which are a relatively new technology, have the potential to improve even more. Further research is still needed to refine the mmWave radar capabilities. Nonetheless, these radars show great potential in complementing or enhancing the capabilities of cameras and LiDARs in developing safe and effective autonomous driving systems.

The research in [9] and [10] provides insights into the different aspects of each sensor, showcasing their major strengths and weaknesses in the context of obstacle detection in autonomous driving. Table 1.1 compares the main characteristics of each sensor.

Table 1.1: Comparison of Sensor Technologies

Sensor Type	Camera Systems	LiDARs	Ultrasonic Sensors	mmWave Radars
Detection Range	Short to Medium	Medium to Long	Very Short	Short to Long*
Detection Accuracy	High	Medium to High	Medium	Medium
Detection Resolution	Medium to High	Medium to High	Very Low	Low to Medium*
Instant Velocity Estimation	No	No	No	Yes
Size	Small	Medium	Very Small	Small
Cost	Low	High	Very Low	Low to Medium
Robustness to Adverse Conditions	Moderate	Moderate to High	Low	Very High

* - The specific frequency used affects the effective range of the radar.

1.3 Importance of Object Classification

Accurately classifying obstacles in real-time is crucial for ensuring safe and reliable autonomous driving. Autonomous vehicles must be able to perceive and understand their surroundings, including identifying and classifying a diverse range of obstacles, both static and dynamic.

For instance, in highway scenarios, a vehicle moving at a constant speed poses a lesser threat to another moving vehicle than a vehicle that is stationary in the middle or on the side of the road. Similarly, in urban environments, near crosswalks or areas with high pedestrian volume, vehicles need to be able to distinguish between pedestrians and stationary obstacles such as traffic signs.

1.4 Research Problem and Objectives

This thesis explores the current challenges of using mmWave RADARs in obstacle detection and tracking applications. While mmWave RADARs offers advantages such as robustness in adverse weather conditions and the ability to detect objects at long ranges, there are several challenges related to the lower resolution of these sensors, the complexity of accurately interpreting RADAR signals, and differentiating between various obstacles in dynamic road environments.

The objective of this thesis is to develop a robust obstacle detection and tracking algorithm that uses exclusively mmWave RADAR data as an input. The algorithm will be designed to handle the specific challenges of low-resolution and noisy data from these radars. After development, its performance will be evaluated through a series of tests in diverse real-world road scenarios.

1.5 Document Outline

This thesis is organized as follows:

- **Chapter 2: Background & State of the Art** provides a brief overview of some of the current State of the Art approaches to the obstacle tracking problem as well as a detailed explanation of the working principles of mmWave RADAR sensors;
- **Chapter 3: Problem Statement** discusses in more detail the problem of obstacle detection and tracking, as well as some important considerations needed to create a robust system;
- **Chapter 4: Developed Algorithm** explains the proposed solution;
- **Chapter 5: Results** contains the results obtained by running the developed algorithm on road scenes from a public dataset, as well as road scenes recorded with our own setup;
- **Chapter 6: Conclusion** provides some overall conclusions about the developed algorithm, its limitations and possible improvements in future work.

2
Background & State of the Art

Contents

This chapter explores the current advancements and methodologies in automotive obstacle detection and tracking, focusing initially on the well-established technologies of cameras and LiDARs followed by an explanation of the working principles of the mmWave RADAR technology along with some of the more promising algorithms that can be applied to the data coming from the RADAR.

2.1 Sensor Applications Today

2.1.1 Cameras

Recent works focused on the usage of cameras resort mainly to machine learning techniques to process data and extract information.

An example of this approach involves using a Neural Network to convert data from an RGB camera into a point cloud, which is subsequently processed by a clustering algorithm to detect obstacles in the scene [11]. This method illustrates the increasing trend of combining traditional imaging with advanced computational techniques to enhance detection accuracy and reliability.

Another different approach involves using multiple depth cameras to provide multi-directional obstacle detection around a vehicle [12]. These cameras create a 3D representation of the surrounding environment. Various techniques are then applied to the point cloud generated by the cameras to filter out noisy points and interpolate regions where information is missing. Finally, the processed point cloud is segmented to identify obstacles in the vehicle's path.

Given that roads are inherently designed with visual cues to guide drivers, it is evident that cameras, as sensors that excel in capturing visual information, will continue to play an indispensable role in automotive safety systems, especially when paired with other types of sensors.

2.1.2 LiDAR

Many different works have been carried out using LiDARs as a basis, from mapping to obstacle detection; which makes LiDARs one of the most adaptable sensors in today's automotive automation sector.

LiDARs generate high-density point clouds, offering great versatility in processing approaches. One common approach involves using derivations of a famous neural network capable of processing point clouds and segmenting the data into different classes, PointNet [13].

One such example involves using a LiDAR sensor placed on a car in an urban scenario to detect obstacles with a custom Neural Network derived from PointNet [14]. This approach can correctly track and estimate the positions and orientations of other vehicles on the road; however, it is unable to do the same for some other obstacles, such as motorcycles.

Another approach developed an algorithm capable of segmenting and tracking data from a LiDAR [15]. The algorithm begins by filtering out road-related data and identifying clusters in the remaining data. It then classifies static obstacles through template matching with previous data frames and moving obstacles using geometric feature matching algorithms. The identified obstacles are tracked with a standard Kalman Filter. This is effective at tracking the positions and yaw angles of obstacles on the road. However, it faces challenges when dealing with obstacles very close to each other, often clustering them into a single large obstacle.

As urban environments grow more congested and road layouts become more complex, the role of LiDAR in automotive safety systems becomes increasingly indispensable. Its precision in recreating detailed road scenes is invaluable for developing reliable ADAS.

2.1.3 Radar

Given that the point clouds produced by mmWave RADARs are sparse and noisy when compared to point clouds produced by Depth cameras and LiDAR sensors, current approaches focus mainly on the usage of mmWave sensors paired with other sensors. This allows for the overcoming of certain limitations of each sensor and the improvement of the quality of the results.

One approach uses both a mmWave radar and a LiDAR to create a multi-obstacle tracking system [16]. The results demonstrate that the fusion of these sensors leads to better tracking performance than methods using only one of the sensors. Improvements are observed across various benchmarks, including obstacle position, velocity, and size estimations.

Another study developed a Neural Network to infer depth in images from an RGB camera, aided by a mmWave radar [17]. The results indicate that the addition of the mmWave radar enhances image segmentation quality, particularly in scenarios where the subjects in the scene contain metallic surfaces that reflect radar signals effectively.

There are approaches that focus solely on the use of radar sensors. For instance, one study uses data from two separate mmWave radars in a Neural Network to estimate bounding boxes of obstacles within the radars' Field Of View (FoV) [18]. While this approach shows promising results, it requires a substantial amount of training data, necessitating the creation of bounding boxes for many hours of data, which can be labor-intensive.

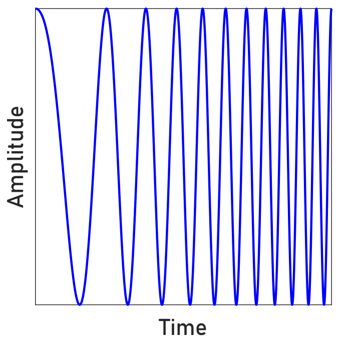
2.1.4 Working Principles of Radar

Radars work by emitting a known signal and detecting the reflections of this signal created by obstacles in the sensor's FoV. The Radar Equation (2.1) provides the mathematical relationship between the transmitted and received signal power in radar systems.

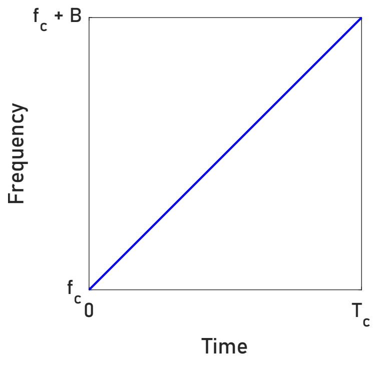
$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma F}{(4\pi)^3 R^4 L} \tag{2.1}$$

Here P_r and P_t are the power of the received signal and the power of the transmitted signal respectively, G_t and G_r, are the gains of the transmitting and receiving antennas respectively; λ is the wavelength of the emitted signal, σ is the cross-section of the obstacle; F is the propagation factor, R is the distance from the emitter to the obstacle, and L is a factor that represents all propagation losses other than free space path loss. This equation encapsulates all the relevant parameters that can influence the relationship between the reflected and the emitted signals, e.g., how the cross-section σ of an obstacle directly influences how much power is reflected to the radar.

In the case of mmWave radars, the emitted signal is a Frequency-Modulated Continuous Wave (FMCW), usually a sinusoidal wave often referred to as chirp (Fig. 2.1). The chirp signal is characterized by a start frequency (f_c), bandwidth (B), duration (T_c), and slope (S) representing the rate of change of frequency.



(a) Chirp signal, amplitude as a function of time.



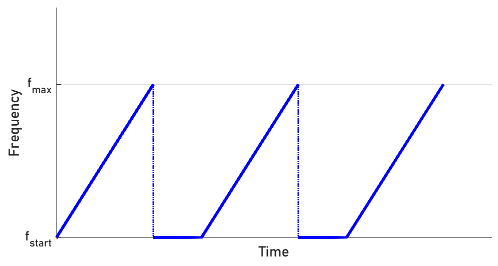
(b) Chirp signal, frequency as a function of time.

Figure 2.1: Signal emitted by a mmWave radar.

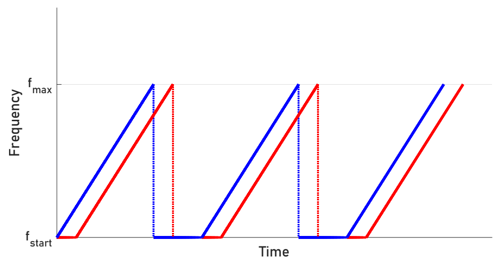
Each of these chirp parameters directly affect the characteristics of mmWave radars such as maximum detection range (R_{max}), range resolution (R_{res}), maximum velocity (v_{max}), and velocity resolution (v_{res}) is described in 2.2.

$$R_{max} \propto T_c\,; \quad R_{res} = \frac{c}{2B}\,; \quad v_{max} = \frac{\lambda}{4T_c}\,; \quad v_{res} = \frac{\lambda}{2T_f} \tag{2.2}$$

Here, T_f corresponds to the time interval of each frame of radar data, which includes multiple chirps, as each frequency sweep occurs in just a few nanoseconds. Fig. 2.2 shows a sequence of chirps in one frame of radar data. Fig. 2.2**(a)** illustrates the emitted chirps, while Fig. 2.2**(b)** shows both the emitted chirps and the received signals.



(a) Sequence of emitted chirp signals with frequency as a function of time.



(b) Sequence of emitted chirp signals (in blue) along with the reflected signal (in red) with frequency as a function of time.

Figure 2.2: Sequence of chirps and reflections.

In this example, there's only one reflected signal, indicating one detected obstacle. However, there are often many unwanted reflections and noise in real-world scenarios. To correctly estimate obstacle distances, it is necessary to know the exact frequency difference between the emitted and the received signals; this difference is also known as beat frequency. To do so, radars mix the emitted signal with the received signals, resulting in an Intermediate Frequency (IF) signal. A Fast Fourier Transform (FFT) is applied to the sequence of IF signals to obtain the actual frequency difference between the two signals.

Let x_1 and x_2 represent the emitted and the received signals, respectively, each represented by a sinusoidal wave with its frequency (ω_1, ω_2) and its phase shift (ϕ_1, ϕ_2):

$$\begin{aligned} x_e &= sin(\omega_1 t + \phi_1) \\ x_r &= sin(\omega_2 t + \phi_2) \end{aligned} \tag{2.3}$$

By mixing these signals, the resulting IF signal is going to be a sinusoidal wave with the form:

$$x_{IF} = sin[(\omega_1 - \omega_2)t + (\phi_1 - \phi_2)] \tag{2.4}$$

For an obstacle at distance d, equation 2.4 can be further simplified to:

$$x_{IF} = Asin(2\pi f_{IF} t + \phi_{IF}) \tag{2.5}$$

Where the values of f_{IF} and ϕ_{IF} can be written in function of the distance to the target (d), the slope of the chirp signal (S), and the wavelength of the emitted wave at the center frequency (λ). The complete mathematical deduction of these variables can be found in [19].

$$f_{IF} = \frac{S2d}{c} \tag{2.6}$$

$$\phi_{IF} = \frac{4\pi d}{\lambda} \tag{2.7}$$

The distance (d) between the obstacle that reflected the wave and the sensor can be easily obtained after estimating the time delay (τ) between the emitted and reflected waves:

$$\tau = \frac{2d}{c} \Leftrightarrow d = \frac{\tau c}{2} \tag{2.8}$$

Having estimated the distance of the obstacle, from 2.7, it is possible to derive the phase difference ($\Delta\phi$) between the emitted and received signal:

$$\Delta\phi = \frac{4\pi v T_c}{\lambda} \tag{2.9}$$

In equation 2.9, the term T_c corresponds to the observation interval, which is equal to the time duration of the modulation of the chirp signal, as seen in Fig. 2.1(b). This equation can be rearranged to estimate the radial velocity (v) of the obstacle that reflected the chirp signal:

$$v = \frac{\Delta\phi\lambda}{4\pi v T_c} \tag{2.10}$$

In order to obtain the correct radial velocity (v) estimation, radars perform a second FFT, known as the Doppler FFT, on the sequence of phase differences over time. This results in a Doppler spectrum, where each frequency corresponds to a different possible velocity of the detected obstacle. The spectrum's peak frequency corresponds to the detected obstacle's correct velocity.

In summary, equations 2.8 and 2.10 show how to estimate the distance and the radial velocity, respectively, of a single target relative to the mmWave sensor. In order to estimate the spatial coordinates of the detected obstacle, a different approach is necessary.

By knowing the angle of a reflected signal along the horizontal plane, called Angle of Arrival (AoA), it is possible to estimate the obstacle's position in this plane. Fig. 2.3 contains a visualization of the AoA.

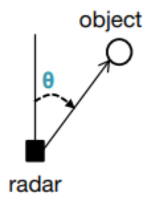


Figure 2.3: Angle of arrival. [19]

To estimate the AoA along a plane, at least two Receiving Antennas (Rx) antennas are required. Given that the two antennas are in slightly different positions, each Rx antenna will measure a different distance d. Fig. 2.4 contains a visualization of the differences in distance measured by each of the Rx antennas.

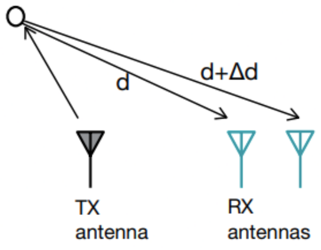


Figure 2.4: Two antennas are required to estimate the AoA. [19]

In the configuration of Fig. 2.4, the phase change between the two antennas can be derived mathematically by equation 2.11.

$$\Delta\phi = \frac{2\pi\Delta d}{\lambda} \tag{2.11}$$

Under the assumption the two Rx antennas are on the same plane, the distance Δd can be expressed as $lsin(\theta)$, where l is the distance between the antennas. By rearranging equation 2.11 with this new value of Δd, it is possible to calculate the AoA of the reflected wave:

$$\theta = sin^{-1}(\frac{\lambda\Delta\theta}{2\pi l}) \tag{2.12}$$

Using multiple Rx antennas makes it possible to estimate the AoA across various planes. In fact, with only three Rx antennas, it is possible to estimate the relative 3D position between the detected obstacle and the radar. The spatial arrangement of these antennas dictates the shape of the radar's FoV and the resolution along each axis.

It is worth noting that, so far, the assumption is that the radar has detected only one object, i.e., there is only one reflection of the emitted chirp signal. In reality, however, multiple reflections of the emitted signal are reaching the Rx antennas simultaneously. To distinguish between different reflections, these radar sensors use FFTs to detect the various beat frequencies present in the IF signals, each corresponding to a detected obstacle. A more in-depth explanation of this process can be found in the official Texas Instruments document where the mmWave radar fundamentals are explained. [19]

In summary, the data generated by a mmWave radar with multiple Rx antennas can then be interpreted as a 3-dimensional point cloud, where each detected point contains additional information about the module of its radial velocity.

2.2 Datasets

As research in ADAS has grown, large datasets featuring recordings from various sensors in diverse road scenarios have emerged. These datasets, containing a wide range of driving conditions, were developed to support and encourage the advancement of ADAS technologies.

One of the first datasets created was the KITTI Dataset, recorded using a stereo camera system consisting of an RGB camera and a monochrome camera, a 3D LiDAR sensor, and vehicle position data provided by Global Positioning System (GPS) [20].

As the demand for more complex datasets increased, new datasets containing more data, recorded with newer sensors, and, importantly, including labeled data, were created. An example is the NuScenes dataset, which comprises 1000 scenes recorded across four different cities [21]. This dataset was recorded using one LiDAR, six cameras providing a 360-degree view around the vehicle, five mmWave radars (three at the front and two at the back of the vehicle), and data from a GPS and Inertial Measurement Unit (IMU) placed inside the vehicle. The inclusion of labeled data is crucial for training neural networks.

In recent years, various radar models have been used across datasets, generally categorized into two main types: the 2+1D radars, which capture range, azimuth, and radial velocity, and the 3+1D radars, which incorporate elevation along with the previous measurements. For example, the NuScenes dataset [21] and the RadarScenes dataset [22] contain data recorded with 2+1D radars. However, more recent datasets tend to use the more advanced 3+1D radars, which is the case of the Pointillism dataset [18] and the View-of-Delft Dataset [23].

2.3 Clustering Algorithms

The analysis of RADAR data presents some challenges due to the presence of significant noise within the generated point cloud and their lower resolution. To identify possible obstacles in the point cloud, it is necessary to apply clustering algorithms. Some algorithms stand out for processing the data generated by RADAR sensors.

2.3.1 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is an algorithm specifically designed to handle sparse and noisy point clouds, making it particularly well-suited for analyzing data from the mmWave radar. This algorithm finds dense areas and expands these recursively, forming individual clusters, separated by low-density zones that exist in the point cloud.

Algorithm 1 Pseudocode of Original Sequential DBSCAN Algorithm

```
Input: DB: Database, $\epsilon$: Radius, minPts: Density threshold, dist: Distance function
Output: Clusters and noise points
  Data: label: Point labels, initially undefined
  for all point $p$ in database DB do
      if label($p$) $\neq$ undefined then
          continue
      end if
      Neighbors $N \leftarrow$ RangeQuery(DB, dist, $p$, $\epsilon$)
      if $|N| <$ minPts then
          label($p$) $\leftarrow$ Noise
          continue
      end if
      $c \leftarrow$ next cluster label
      label($p$) $\leftarrow c$
      Seed set $S \leftarrow N \setminus \{p\}$
      for all $q$ in $S$ do
          if label($q$) = Noise then
              label($q$) $\leftarrow c$
          end if
          if label($q$) $\neq$ undefined then
              continue
          end if
          Neighbors $N \leftarrow$ RangeQuery(DB, dist, $q$, $\epsilon$)
          label($q$) $\leftarrow c$
          if $|N| <$ minPts then
              $S \leftarrow S \cup N$
          end if
      end for
  end for
```

The effectiveness of DBSCAN relies on two main parameters: ϵ (epsilon) and $minPts$. The parameter ϵ defines the maximum distance that determines the neighborhood of a data point, while $minPts$ specifies the minimum number of data points required within that neighborhood for a point to be considered a core point. DBSCAN distinguishes between three types of points: core points, which have a sufficient number of neighboring points within ϵ; border points, which have fewer neighbors than the required minimum but are within the neighborhood of a core point; and noise points, which have neither enough neighbors nor fall within the neighborhood of a core point.

Algorithm 1, as described in [24], explains the iterative process behind the original DBSCAN algorithm. In the worst-case scenario, where each point in the point cloud represents a different obstacle, DBSCAN has a time complexity of $O(n^2)$. However, it is possible to pre-process the point cloud data with spatial indexing structures, such as k-d trees or R-trees, before applying the DBSCAN algorithm. This pre-processing can reduce the time complexity from $O(n^2)$ to $O(\log(n))$ [25].

One study, described in [26], presents a Spacial-Temporal Clustering (STC)-based method built upon the DBSCAN algorithm to work with mmWave radar point clouds. Another one, explained in [27], proposes an inter-frame DBSCAN clustering method for data generated by mmWave radars, utilizing Doppler information.

2.3.2 K-means Clustering

K-means clustering is a widely used algorithm for partitioning data into distinct groups based on similarity. It is a centroid-based algorithm, where each cluster is represented by its centroid, which is equal to the average of all the points in that cluster.

K-means clustering aims to minimize the sum of squared distances between each data point and its assigned centroid. This optimization objective leads to an iterative process that converges to a locally optimal solution.

Algorithm 2 outlines the steps involved in the K-means clustering algorithm. It begins by initializing k centroids randomly within the range of the data points. Then, it iteratively performs two steps: assignment and update. In the assignment step, each data point is assigned to the nearest centroid based on the Euclidean distance in the assignment step. In the update step, each centroid is recalculated as the mean of the assigned data points. These steps are repeated until convergence is achieved, indicated by minimal centroid movement or after a predefined number of iterations. The K-means algorithm has a time complexity of $O(n * k * I * d)$, where n is the number of data points, k is the number of clusters, I is the maximum number of iterations of the algorithm, and d is the dimensionality of the data.

Algorithm 2 The K-means clustering algorithm

```
Input: $D$: Point cloud data, $k$: Number of clusters
Output: Cluster assignments and centroids
  Initialize $k$ centroids randomly within the data range
  while Convergence or $maxIter$ do
      Assign each data point to the nearest centroid
      Update each centroid as the mean of the assigned data points
  end while
```

The choice of the number of clusters, k, is crucial in K-means clustering, as it can significantly impact the resulting clusters and their interpretability. In obstacle detection, it is impossible to know the number of obstacles (clusters) *a priori*. However, there are techniques that can be used to estimate the optimal number of clusters for a given dataset [28] [29].

The Doppler information can also be used to add one degree of dimensionality to the data, which may increase the effectiveness of the clustering algorithm.

A mmWave radar-based algorithm using the K-means clustering technique is presented in [30]. Here the radar data is used to estimate the positions of pedestrians in a situation containing both human and non-human obstacles.

[31].

2.4 Tracking Algorithms

This section presents some tracking algorithms that are typically used in object-tracking situations. These algorithms estimate and predict the motion of detected objects over time by combining successive information relative to the object being tracked.

2.4.1 Kalman Filter

The Kalman Filter is a widely used tracking algorithm that provides an optimal estimation of state variables by iteratively combining measurements with predictions based on a linear dynamic model. The Kalman Filter assumes that the system being tracked can be described by a linear dynamic model and that the noise present in the measurements follows a Gaussian distribution.

The Kalman Filter basic equations are presented in (2.13) and (2.14). x_k corresponds to the vector of the system's state variables in the instant k, and z_k corresponds to the measurements of the state variables in instant k.

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{2.13}$$

$$z_k = Hx_k + v_k \tag{2.14}$$

From (2.13), it is possible to confirm that x_k results from a linear combination of its previous value, a control signal u_k, which can be null at times, and a process noise w_{k-1}, which captures the uncertainty or randomness in the state evolution. (2.14) shows that the measurements z_k also result from a linear combination of the state plus the measurement noise v_k.

The values A, B, and H correspond to the state transition matrix, the control input matrix, and the measurement matrix, respectively. In linear problems, these matrices contain just numeric values, although these values may change between states.

The process noise w_{k-1} and the measurement noise v_k are assumed to be Gaussian. However, even if the noises' mean and standard deviation are poorly estimated, the Kalman Filter can still converge to correct estimations.

The algorithm operates in two main steps: the prediction step and the update step. In the prediction step, the time update equations estimate the values of the current state and the error covariance for the next time step. In the update step, the measurement update equations adjust the previously estimated values based on the actual measurements at that time step.

Fig. 2.5 showcases the equations used in the Kalman Filter and also how each iteration is processed [31].

In the time update equations, Q is the process noise covariance, $\hat{x}_k^-$ is the *a priori* state estimate obtained from previous state estimations ($\hat{x}_{k-1}^-$), and P_k^- is the error covariance estimate at iteration k.

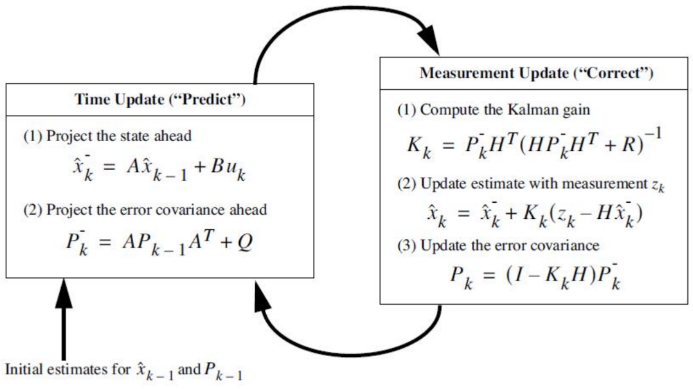

Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Figure 2.5: Kalman Filter equations and iterations. [31]

In the measurement update equations, R is the measurement noise covariance, $\hat{x}_k$ is the *a posteriori* state estimate given by the actual measurements z_k, P_k is the *a posteriori* error covariance estimate, and K_k is the Kalman gain at iteration k.

By iteratively repeating the prediction and update steps, the Kalman Filter can continuously refine its estimation of the system's state over time, resulting in a robust and accurate tracking solution.

The Kalman Filter assumes linear system dynamics and measurement models. However, in many real-world systems, the underlying system dynamics and measurements are non-linear. The Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) are two widely used variations to address this issue.

The EKF [32] addresses this issue by linearizing the system dynamics and measurement functions using first-order Taylor series approximations. It provides a way to estimate states in non-linear systems by incorporating the linearized models into the prediction and update steps. However, the EKF relies on the assumption that the system's characteristics are approximately linear within the estimation region, and its performance can be affected by significant non-linearities, resulting in large estimation errors.

To address highly non-linear systems, the UKF [33] directly captures the mean and covariance of a set of carefully selected sample points called sigma points. By propagating these sigma points through the non-linear functions, the UKF provides a more accurate approximation of the actual state distribution compared to the EKF.

An experiment was conducted using a mmWave radar to capture data in an agricultural field with some moving obstacles [34]. The EKF algorithm was applied to the radar data to track the obstacles. The results showed that the estimations from the EKF had an average error that was 51.6% lower than the average observations from the radar, demonstrating it as a viable approach to the problem.

2.4.2 Particle Filter

The Particle Filter, also known as the Sequential Monte Carlo (SMC) method, is another widely used tracking algorithm in object-tracking scenarios. Unlike the Kalman Filter, which relies on assumptions of linearity and Gaussian distributions, the Particle Filter is a non-parametric and non-linear filtering technique. It is particularly effective in handling non-linear and non-Gaussian systems, making it suitable for a wide range of tracking applications.

The Particle Filter estimates the system's state by representing it with a set of particles, each representing a possible hypothesis of the system's state. These particles are randomly generated and propagated over time based on the system dynamics and measurements.

The Particle Filter's basic functioning is described in five steps: initialization, prediction, update, resampling, and estimation.

The initialization consists of generating an initial set of particles x_i, where $i = 1, 2, ..., N$. Each x_i corresponds to a hypothetical system state and each of these particles has a weight associated with it, which is initially set to $\frac{1}{N}$.

In the prediction step, each particle is propagated forward in time based on the system's dynamics. (2.15) represents the forward propagation of a single particle x_i.

$$x'_i = f(x_i, u) + w_i \tag{2.15}$$

In (2.15), x'_i is the propagated state, $f(.)$ is the non-linear motion model, u is the control input, and w_i is the process noise.

In the update step, the importance (or likelihood) of each particle is evaluated based on the measurements of the system. The measurement model is represented by Equation 2.16.

$$z = h(x_i) + v_i \tag{2.16}$$

In (2.16), z corresponds to the measured value of the system, $h(.)$ to the non-linear measurement function, and v_i to the measurement noise. The weight update process is described in (2.17).

$$w'_i = w_i \cdot P(z|x_i) \tag{2.17}$$

In (2.17), w'_i corresponds to the updated weight, and $P(z|x_i)$ to the likelihood or probability of the measurement given the particle's state.

In the resampling step, the particles with lower associated weights are eliminated. This aims to concentrate particles in regions with higher probability, ideally converging to a single high-density region. There are different techniques to perform the resampling, such as systematic resampling or resampling with replacement.

In the final step, an estimation of the system is obtained by computing the weighted average of the particle states, as described in (2.18).

$$\hat{x} = \sum_{i=1}^{N} w_i \cdot x_i \tag{2.18}$$

By iteratively repeating the prediction, update, resampling, and estimation steps, the Particle Filter can refine its estimation of the system's state over time, providing a robust and accurate tracking solution.

The work developed in [35] presents a method based on the Particle Filter to create a mmWave-based beam tracking algorithm. This approach highlights the potential application of a Particle Filter in our system.

The described algorithms, along with their variations and extensions, demonstrate to be viable approaches for processing the data generated by mmWave radar sensors. Among these, DBSCAN and Kalman Filter appear to be simpler to implement in the early stages and are therefore the initial choices for the system.

3 Problem Statement

Contents

Let's imagine the hypothetical situation in Fig. 3.1. This image presents a road with two opposing lanes, where multiple vehicles drive simultaneously. In one of the lanes, there are two vehicles, denoted Vehicle A and Vehicle B. The opposing lane contains a single vehicle, referred to as Vehicle C. Adjacent to the road is a row of trees, representing possible stationary obstacles.

Let us assume the perspective of Vehicle B in this scenario. The presence of multiple obstacles, including other vehicles and stationary obstacles such as trees, introduces potential dangers that Vehicle B must proactively detect and account for in its decision-making processes. To ensure safe navigation, Vehicle B must accurately perceive the neighboring vehicles' positions, velocities, and trajectories and acknowledge the presence of stationary obstacles and their positions.

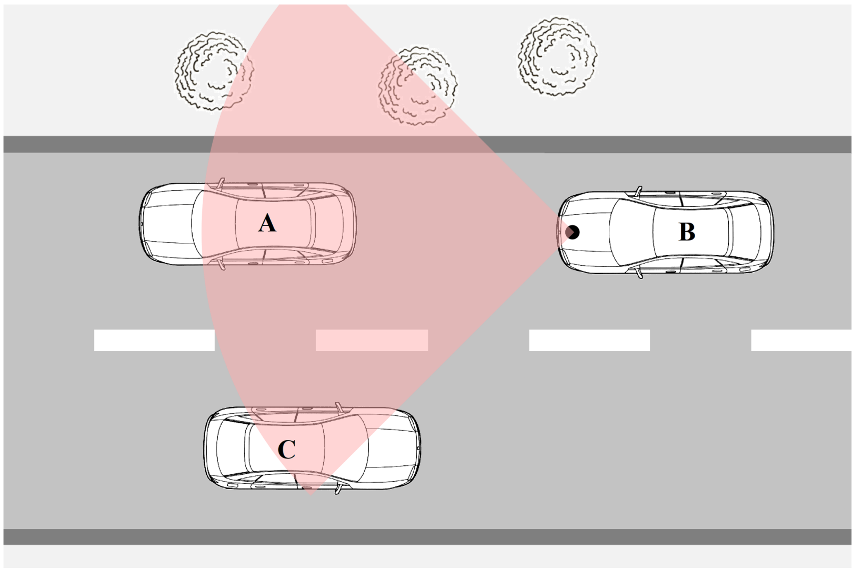


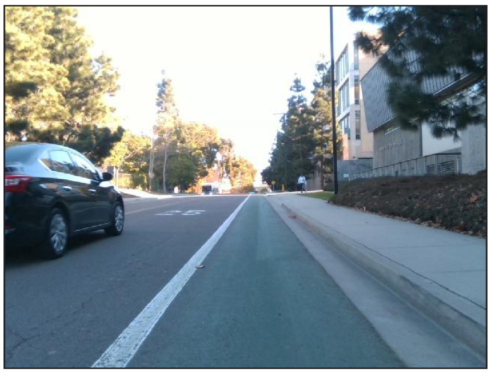
Figure 3.1: Hypothetical road situation.

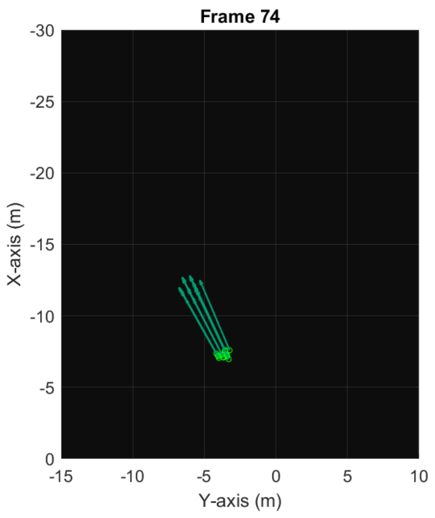
3.1 Thesis Objective

This work aims to develop a real-time detection and tracking system based on a set of mmWave radars. This system should be able to detect both stationary and moving obstacles and should be able to correctly predict the motion of the latter, once they leave the radars' FoV.

3.2 Radar Data Interpretation and Processing

As mentioned in Chapter 2, radar-generated data contains the positions and the velocities of some points in its field of view. Fig. 3.2 presents an example of the data measured by the radar.


(a) RGB image of the radar's FoV



(b) Frame of radar data.

Figure 3.2: Example of a radar data frame.

In (3.1), the generic structure of the data that will be handled is presented.

$$p_{ir}(n) = [x_{ir}(n), y_{ir}(n), z_{ir}(n), v_{ir}(n)] \tag{3.1}$$

Where i corresponds to the index of the point and ranges from 0 to N (number of detected points) and r corresponds to the index of the radar and ranges from 0 to R (number of used radars).

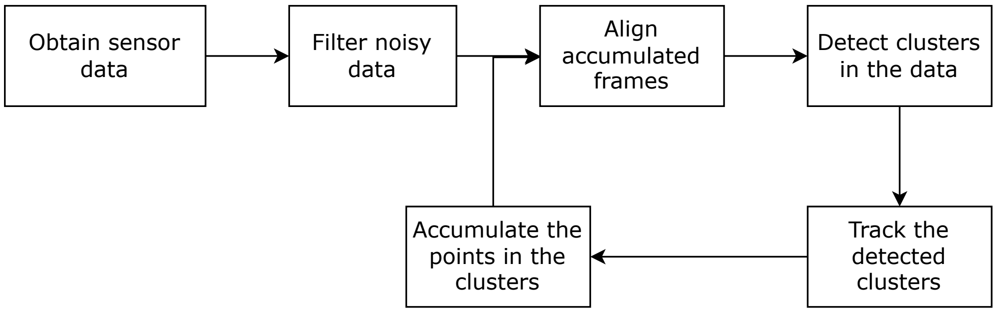
Each frame of radar data will inevitably contain points that correspond to reflections from the road surface or unwanted noisy measurements. These points are not relevant to the problem at hand and must be removed, taking into account the radar spatial position in the scene.

The points obtained from multiple successive radar measurements will be grouped into clusters based on their location and velocity attributes. In (3.2) is presented a set of clusters: the result of the clustering of the data from radar r at instant n. M corresponds to the total number of clusters resulting from this process. Each cluster $c_{mr}(n)$ has a set of radar points associated with it, as seen in (3.3).

$$C_r(n) = \{c_{1r}(n), c_{2r}(n), ..., c_{Mr}(n)\} \tag{3.2}$$

$$c_{mr}(n) = \{p_{ir}(n), p_{i'r'}(n), ..., p_{i''r''}(n)\} \tag{3.3}$$

Finally, eachcluster will be tracked individually and updated as new clusters are detected. The lifespan of each point within the cluster must be carefully selected because the velocities of obstacles can change rapidly and the presence of outdated points in a cluster can affect its analysis. The sequential steps involved in the proposed solution to the problem are illustrated in Figure 3.3.

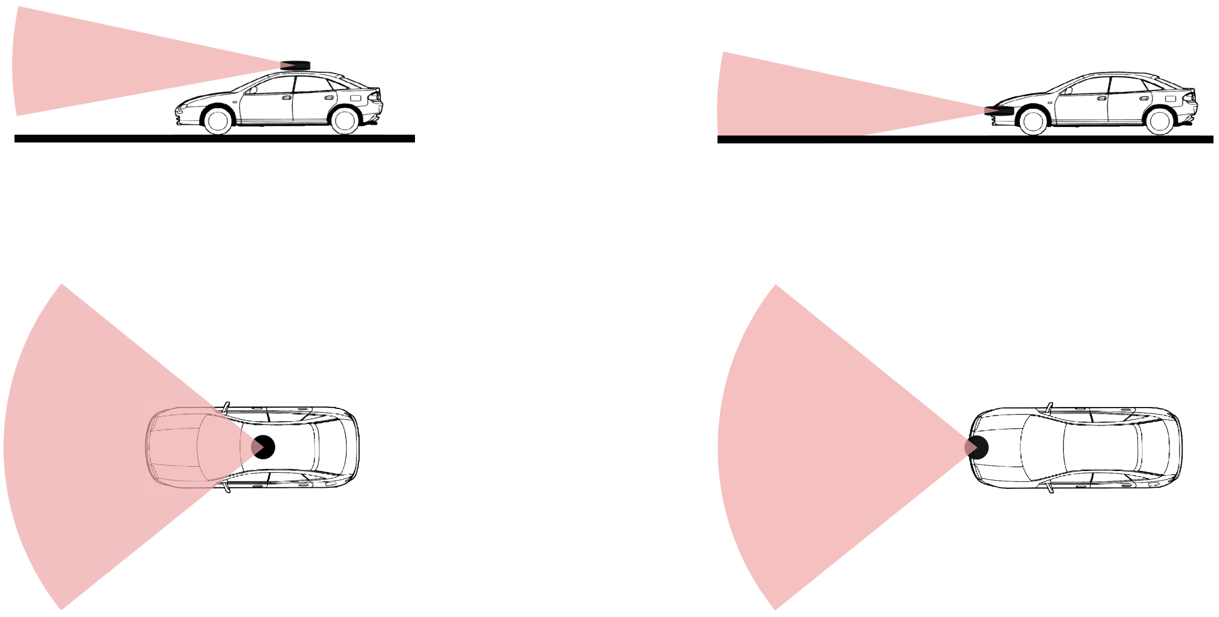


Obtain sensor data
Filter noisy data
Align accumulated frames
Detect clusters in the data
Track the detected clusters
Accumulate the points in the clusters

Figure 3.3: Solution pipeline.

3.3 Placement of the Radar

The accurate placement of the radar system is crucial to ensure effective obstacle detection and situational awareness. Fig. 3.4 visually demonstrates the impact of radar placement on the FoV, represented by the red cone. Each sub-figure contains two images: the upper image depicts the view from the XY plane, while the lower one represents a view from the Z-axis.



(a) Radar FoV, when placed at roof height.

(b) Radar FoV, when placed at headlight height.

Figure 3.4: Effect of radar placement on the FoV.

From these images, it is possible to conclude that placing the radar too high introduces blind spots near the vehicle, compromising its ability to detect obstacles in close proximity, while placing the radar too low leads to the intersection of the road with the radar FoV, reducing its effective area for obstacle detection.

Another challenge involves the placement of multiple radars in the same vehicle. Fig. 3.5 represents the fields of view of two radars placed on the same vehicle.

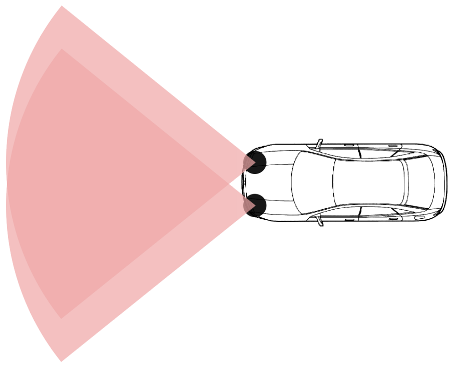


Figure 3.5: Placement of two radars on the same vehicle.

It is possible to see that there are areas where both FoVs overlap, which can potentially improve the quality of detections within these zones. However, there are also areas covered by only one radar, which can be increased by reducing the overlap of the two FoV, increasing the covered area. The balance between maximizing coverage and enhancing the quality of detections remains an ongoing challenge in the automotive context. An experiment was conducted using mmWave radars for Human Activity Recognition (HAR) [36]. The tests demonstrated that fusing data from multiple mmWave radars significantly improves the precision of detections compared to using a single radar.

3.4 Coordinate Frames

It is also important to define the appropriate coordinate frames to deal with the positions of the obstacles and the vehicle.

We assume that a road can be considered a two-dimensional plane for simplicity and ease of analysis. Figure 3.6 illustrates the coordinate frames of the radar (R), the car (C), and the world (W).

Within this diagram, it is possible to observe the coordinate frame associated with the car, typically referred to as the ego frame or vehicle frame. This frame serves as a reference when measuring the positions and orientations of objects relative to the vehicle.

The radar's coordinate frame is the reference to all of its measurements. These measurements need to be converted into the car's coordinate frame to ensure consistency and ease of integration with other vehicle systems.

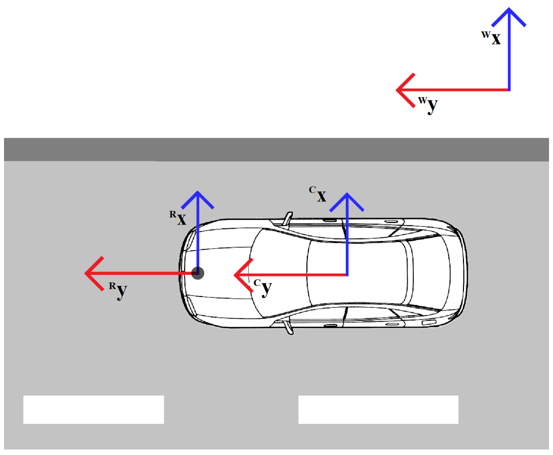


Figure 3.6: Important coordinate frames.

The world coordinate frame is essential to accurately locate the vehicle within its surrounding environment. Using this frame, it is possible to define the vehicle's position using global coordinate systems such as the Universal Transverse Mercator (UTM) coordinates.

3.5 Vehicle Kinematics

When tracking vehicles, it is advantageous to use appropriate kinematic models to predict their motion rather than assuming free movement. This constrains the movement predictions to a smaller and more probable set of values.

A simplified car-like vehicle kinematic model, described in [37], is a potential consideration for future implementation in this work.

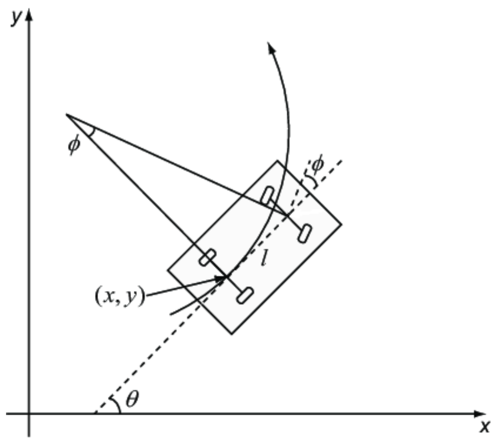


Figure 3.7: The kinematic model of a car-like vehicle. (x, y) are the coordinates of the rear axle midpoint, θ is the orientation of car, ϕ is the steering angle and l is the wheelbase. [37]

The state variables of this kinematic model are the position of the vehicle's rear axle midpoint

(X, Y, Z), the orientation of the car and its variation (θ, Θ), and the steering angle and its variation (ϕ, Φ), which can be seen in (3.4).

The system inputs are the car's linear velocity (v) and the car's steering velocity (ω), as shown in (3.5).

$$s(n) = [x(n), y(n), z(n), \theta(n), \Theta(n), \phi(n), \Phi(n)] \tag{3.4}$$

$$u(n) = [v(n), \omega(n)] \tag{3.5}$$

Finally, the kinematic equations of the vehicle in this model can be expressed in matrix form as shown in (3.6).

$$\begin{bmatrix} x(n+1) \\ y(n+1) \\ z(n+1) \\ \theta(n+1) \\ \Theta(n+1) \\ \phi(n+1) \\ \Phi(n+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(n) \\ y(n) \\ z(n) \\ \theta(n) \\ \Theta(n) \\ \phi(n) \\ \Phi(n) \end{bmatrix} + \begin{bmatrix} \Delta t \cdot cos(\theta(n)) & 0 \\ \Delta t \cdot sin(\theta(n)) & 0 \\ 0 & 0 \\ 0 & 0 \\ tan\frac{\phi(n)}{l} & 0 \\ 0 & 0 \\ 0 & K \end{bmatrix} \begin{bmatrix} v(n) \\ \omega(n) \end{bmatrix} \tag{3.6}$$

4 Developed Algorithm

Contents

This chapter explains in detail the algorithm developed, as well as the considerations and choices made while developing it. The developed algorithm can be divided into a few different steps:

4.1 Acquiring Radar Data

This algorithm utilizes the Robot Operating System (ROS) framework for communication with the radar, by subscribing to the relevant ROS topics. The official mmWave ROS Driver from Texas Instruments [38] was used to publish the radar data onto ROS topics.

4.2 Filter Unwanted Data

Following radar data acquisition, an initial filtration step is executed to mitigate false detections and points irrelevant to the problem. In this stage, data points with aberrant height values, when considering the radar placement, are filtered. Additionally, points within a small radius of the radar are also filtered, as most of these detections result from interference.

4.3 Align Accumulated Data

The data generated by the radar is sparse and noisy, which presents a challenge when trying to cluster the data based on point density because there are instances where an obstacle will only produce a single point in space. To overcome this, a strategy to utilize multiple frames of radar data was employed. This consists of accumulating the last L frames of radar data and aligning the $L-1$ previous frames to the most recent one.

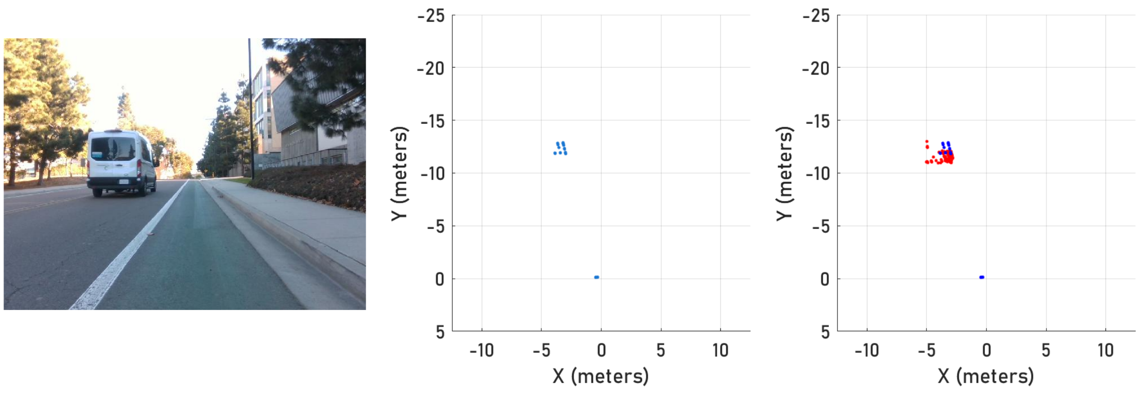
This is possible because of the radial velocity values measured by the radar, which allows the prediction of future positions of the points. The initial prediction is done with the radial velocity value v_r. Once the same point has been associated with two different clusters, it is possible to estimate the velocity of this point by measuring the displacement of the clusters between frames, from where v_a results. The velocity vector v_a is more accurate than the radial velocity measured by the radar v_r. This process is explained in Algorithm 3.

Each time a point position is aligned to the most recent frame, its uncertainty increases. This means that the number of frames to accumulate, L, should not be too high, as this is also the number of updates done to each point.

This strategy results in an increase of effective data, allowing the clustering algorithm used, DBSCAN, to have a denser point cloud as the input. Fig. 4.1 shows how much difference accumulating and aligning the recent frames of data (with $L = 5$) can make, even in a scene where only one obstacle is present.

Algorithm 3 Temporal Alignment Function

```
Input: Frame of radar data
Output: Aligned frame of radar data
  function TemporalAlignment(frame $f$)
      Compute $\Delta t$ between frame $f$ and last frame
      for point in $f$ do
          Compute radial velocity vector $v_r$ using Doppler value
          if point has 2+ associated clusters then
              Estimate more accurate velocity $v_a$ of the point
          end if
          if point has $v_a$ associated then
              Update point: $point \leftarrow point + v_a \cdot \Delta t$
          else
              Update point: $point \leftarrow point + v_r \cdot \Delta t$
          end if
      end for
      return frame $f$ aligned
  end function
```



(a) RGB image of a road scene.

(b) Single frame of radar data (XY plane projection).

(c) Accumulated and aligned frames of radar data (XY plane projection).

Figure 4.1: Comparison of a single frame of radar data and the result of accumulation and aligning of recent frames radar data.

4.4 Cluster Detection

The chosen clustering algorithm to implement in the algorithm was DBSCAN. While, initially, K-means clustering was considered, it proved to be hard to estimate the number of obstacles in the scene. The number of obstacles could be estimated by having a RGB camera monitoring the area corresponding to the FoV of the radar sensor and running an external methodology to estimate the number of obstacles in the scene, however, since DBSCAN performed well in early tests, it was chosen instead.

Depending on the radar configuration used, the ranges and densities of the point clouds generated can vary significantly, meaning that, for each configuration, the values of ϵ and $minPts$ in a cluster must be tuned.

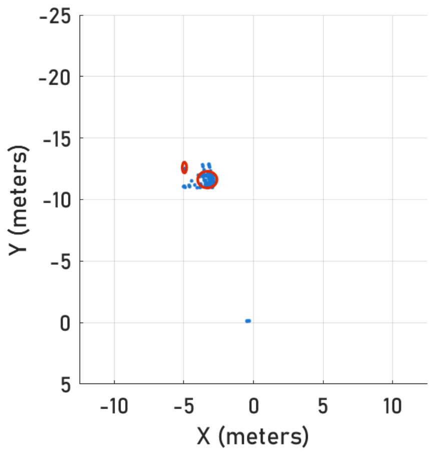


Figure 4.2: Clusters resulting from DBSCAN (XY plane projection).

Fig. 4.2 illustrates the result of the clustering algorithm in the scene described in Fig. 4.1. After each cluster is obtained, its centroids and the average values of v_r and v_a can be estimated by averaging all the associated points. The covariance matrices of each cluster can also be obtained, which will be used to update and initialize the associated Kalman filters. In Fig. 4.2, the calculated variance of the points in each cluster is represented by the red ellipse.

4.5 Cluster Tracking

Once each cluster properties are obtained, it is possible to initialize a Kalman filter to track each cluster individually. The chosen properties to track correspond to the centroid position (x, y, z) and velocity direction $(\dot{x}, \dot{y}, \dot{z})$, which can be collectively represented as the state vector $\mathbf{x}_n$, described in (4.1):

$$\mathbf{x}_n = [x_n, y_n, z_n, \dot{x}_n, \dot{y}_n, \dot{z}_n]^T \tag{4.1}$$

With this state representation, the system dynamics can be described using the state transition matrix A, described in (4.2):

$$\mathbf{x}_{n+1} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \cdot \mathbf{x}_n \tag{4.2}$$

We can now define the observation vector $\mathbf{y}_n$, which comprises two sub-vectors: y_1 representing the position information ($[x, y, z]$) and y_2 representing the velocity information ($[\dot{x}, \dot{y}, \dot{z}]$). This separation allows for the positions and velocities of the centroids to be updated separately. The observation vector $\mathbf{y}_n$ is presented in (4.3):

$$\mathbf{y}_n = [y_{1n}, y_{2n}]^T \tag{4.3}$$

Given that there are two velocity values (v_r and v_a) associated with each cluster, it may be possible to update the Kalman Ffilter velocity with two different measurements, resulting in a better estimation. The Kalman Filter implementation allows the estimation of the positions and velocities of each centroid over time, creating robust tracking for each cluster, even with noisy property estimations. Fig. 4.3 illustrates the result of the Kalman filter tracking for each cluster detected in Fig. 4.2. Here, the magenta ellipses represent the tracking uncertainty of the centroid position.

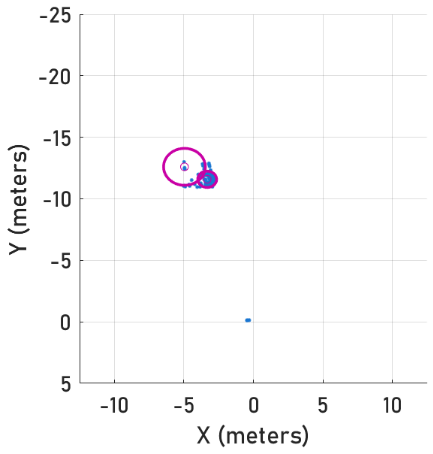


Figure 4.3: Kalman filter tracking results (XY plane projection).

4.6 Data Publishing

Finally, all the calculated parameters are published into ROS topics, meaning they can be visualized with the built in ROS visualizer, rviz, and that they can also be recorded into a rosbag file for posterior analysis.

4.7 Algorithm Structure

Algorithm 4 outlines the steps of the developed solution: it starts by accumulating and aligning the data to the most recent frame ($TemporalAlignment$ function is described in Algorithm 3), followed by detecting clusters within the data, and finally updating current tracks and creating new ones when necessary. In every iteration, each point may be associated with a cluster.

Algorithm 4 Developed Algorithm

```
Input: Sparse data from a mmWave radar
Output: Array of detected obstacles, along with their
        properties: [x, y, z, $\dot{x}$, $\dot{y}$, $\dot{z}$]
  while Data received do
      Accumulate the last $L$ frames
      for $i = 1{:}L - 1$ do
          $TemporalAlignment$(frame $i$)
      end for
      DBSCAN(Aligned Data, eps, MinPts)
      Update list of associations
      if no Kalman Filters then
          Create a Kalman Filter track for each detected cluster
      else
          Predict the next state of existing Kalman Filters
          Update Kalman Filters with closest valid cluster
          Create a Kalman Filter track for new clusters
      end if
  end while
```

Initially, a particle filter approach was also considered for testing. However, due to time constraints, this approach was not pursued.

5

Results

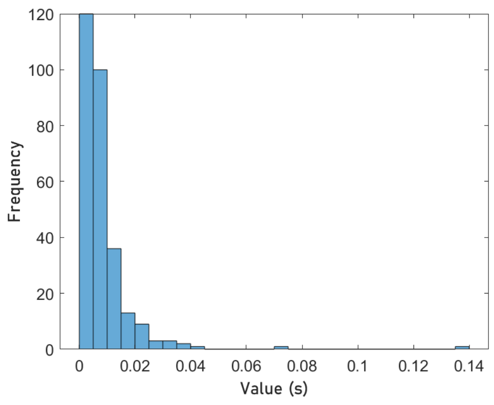
Contents

To evaluate the performance of the developed algorithm, tests were conducted using the Pointillism dataset [18], a public dataset containing recordings of various road scenes captured with a LiDAR, two mmWave radars, and an RGB camera. This specific dataset was selected because the mmWave sensors used for recording are very similar to the one available in our lab. After verifying that the algorithm was fully operational, additional scenes were recorded using our current setup for further analysis.

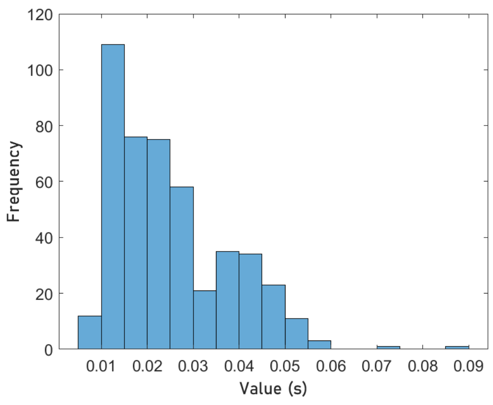
5.1 Real Time Analysis

This algorithm was developed to operate in real-time. To assess its performance, the execution time of each iteration of the algorithm was measured in both a scene from the Pointillism dataset [18] and a scene recorded with our setup. These tests were run on a quad-core laptop running Ubuntu 20.04.

Fig. 5.1(a) shows the execution times of the algorithm running with the Pointillism dataset, while Fig. 5.1(b) presents the execution times for our recorded scene. Some iterations exhibit abnormally long execution times, likely due to sudden drops in performance or momentary synchronization issues. However, the majority of iterations execute well within the radar publishing frame rate, which was set to 30 frames per second in the Pointillism dataset and 15 frames per second in our experiments. The difference in execution times between the Pointillism dataset and our recordings is attributable to the different radar configurations used; the configuration used in our recordings generates more points per frame than the one used in the dataset, requiring more time to process.



(a) Histogram of the iteration execution times in a scene from the Pointillism dataset.



(b) Histogram of the iteration execution times in a scene recorded with our setup.

Figure 5.1: Algorithm execution time analysis.

5.2 Results with a public dataset

In all of the following figures containing ellipses, the ellipses correspond to a confidence level of 3σ, which represents approximately 99.7% certainty that the points will fall within the corresponding bounding ellipse.

The parameters of the radar configuration used for recording the scenes in the Pointillism dataset [18] are presented in Table 5.1, along with the parameters used in the algorithm, shown in Table 5.2.

Table 5.1: Radar Configuration Parameters

Parameter	Value
Start Frequency	77 GHz
Frame Rate	30 fps
Bandwidth	2240 MHz
Range Resolution	0.067 m
ADC Rate	7500 ksps
Chirp Duration	40 μs
Velocity Resolution	2.59 m/s
Maximum Velocity	20.74 m/s

Table 5.2: Algorithm Parameters

DBSCAN Parameters	Value
ϵ	1.5
MinPts	3
Generic Parameters	**Value**
Num Accumulated Frames	6
Kalman Filter Lifespan	10

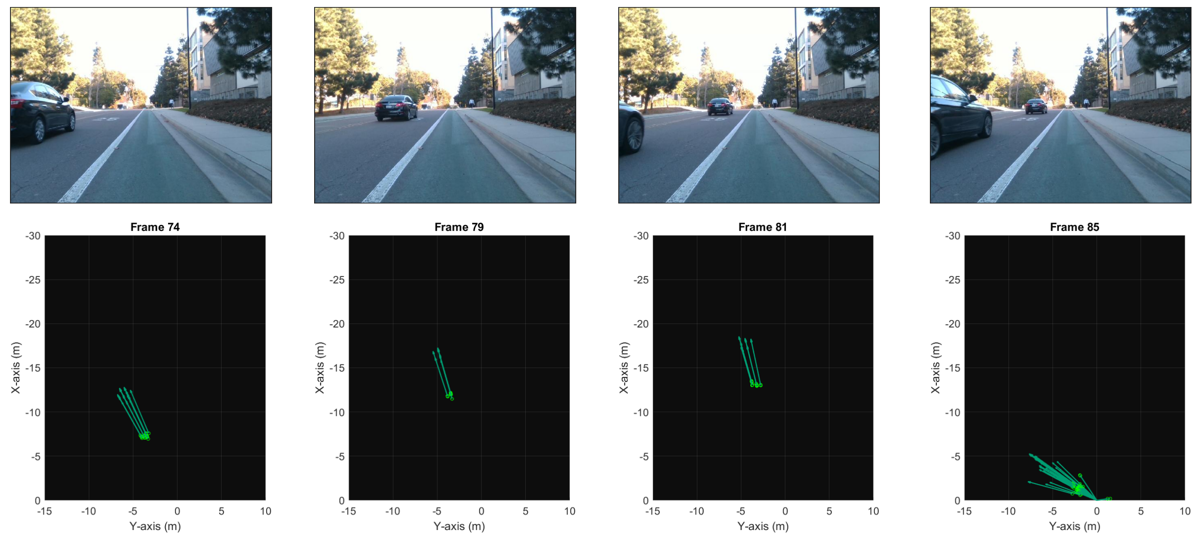
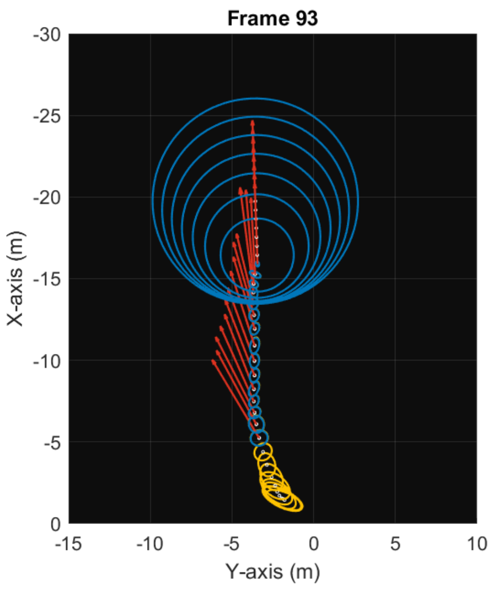
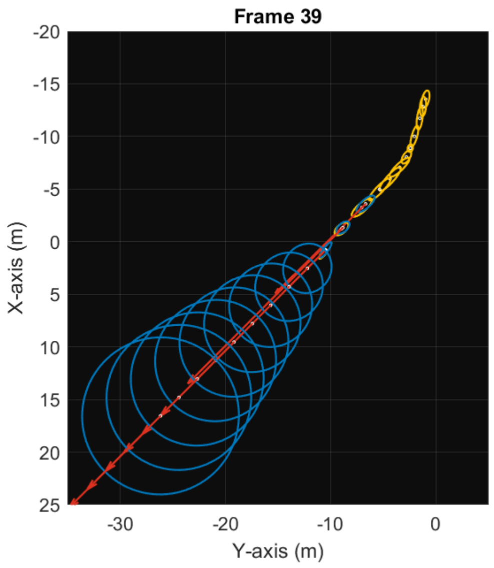


Figure 5.2: Excerpt of scene14, where a car moves away from the sensor. The top row of images contains RGB images of frames 74, 79, 81, and 85, respectively, and the bottom row contains the radar measurements of the same frames.

Fig. 5.2 presents a sequence of frames taken from “scene 14” of the dataset. In this sequence, a vehicle is moving away from the radar sensor. The top row of images contains the RGB image of the scene, while the bottom row contains snapshots of an aerial view of the radar measurements, along with the associated radial velocities represented by green arrows. In these snapshots, the sparse nature of the radar data is evident. Additionally, the last frame of radar data in Fig. 5.2 highlights the limited range of the radar configuration used, where the distant vehicle returns no measurements while the closer one does.



(a) Tracking of the vehicle described in Fig. 5.2.



(b) Unsuccessful tracking example.

Figure 5.3: Tracking examples on a scene 14 from the dataset.

Fig. 5.3(a) shows the result of the algorithm tracking the front vehicle from Fig. 5.2 over time. The centroids are represented by small white circles, the estimated confidence ellipses by the yellow and the blue ellipses, and the velocity vectors estimated by the algorithm by red arrows. The yellow ellipses correspond to the confidence ellipses associated with the clustering algorithm and are calculated using the covariance of the points within the cluster. The blue ellipses, on the other hand, originate from the Kalman Filter tracking this vehicle and reflect the uncertainty of the system's state estimation over time. As expected, after each successful update, the uncertainty of the system diminishes. However, when the tracked vehicle exits the FoV of the radar, the tracking stops receiving new updates, relying solely on its predictions to estimate the system's state, resulting in progressively higher uncertainty.

It is apparent that the algorithm's performance improves as the distance from the sensor increases. This is because the sensor was placed on the side of the road, causing vehicles to enter the radar's FoV at a 90-degree angle. As mentioned previously, when the target moves perpendicular to the radius of the FoV, the velocity measurements of the radar do not correspond to the actual velocity of the target, causing the initial predictions to exhibit jittery movement. However, as the distance from the sensor increases, the radial velocity measured becomes closer to the actual movement direction of the target, leading to smoother tracking.

Fig. 5.3(b) depicts the movement of another vehicle in ”scene 14” of the dataset, this time moving towards the radar. The algorithm fails to track the vehicle correctly, which can be explained by the fact that the Kalman Filter did not have enough updates to estimate the vehicle's movement accurately. Additionally, the final updates were performed when the vehicle was moving perpendicular to the radius of the radar's FoV, resulting in poorer v_r predictions. We can conclude that tracking vehicles moving toward the radar is less reliable than tracking vehicles moving away from the radar with this method.

5.3 Results with current setup

Initially, the radar configuration used in our experiments was intended to match that of the Pointillism dataset (Table 5.1); however, issues arose when parsing this configuration into our radar sensor. This led to the experimentation with different configurations in road scenarios. The analysis of each configuration was conducted qualitatively, relying on visual inspection rather than quantitative data. The most promising configuration is presented in Table 5.3, and the algorithm parameters used are shown in Table 5.4.

Table 5.3: Radar Configuration Parameters

Parameter	Value
Start Frequency	77 GHz
Frame Rate	15 fps
Bandwidth	5000 MHz
Range Resolution	0.03 m
ADC Rate	5000 ksps
Chirp Duration	192 μs
Velocity Resolution	0.029 m/s
Maximum Velocity	19.8 m/s

Table 5.4: Algorithm Parameters

DBSCAN Parameters	Value
ϵ	1.0
MinPts	6
Generic Parameters	**Value**
Num Accumulated Frames	4
Kalman Filter Lifespan	10

Fig. 5.4 shows the setup used for our experiments. It comprises an Intel RealSense camera [39] to record RGB images of the scene and an AWR1443 mmWave radar from Texas Instruments [40]. The camera is placed directly above the radar sensor's antennas, providing a better perspective of the obstacles within the radar's FoV.

The scenes recorded with our setup focused on situations where vehicles on the road move toward or away from the sensor, as these are the most common scenarios while driving.

Fig. 5.5 illustrates a successful tracking example of a vehicle moving toward the sensor. The top row displays RGB images of the tracked vehicle, while the bottom row shows the trail of detections of this vehicle over time. This scene also contains some parked vehicles, which serve as stationary obstacles in this situation. Each snapshot of the tracked obstacles contains two clearly defined obstacles, validating the ability of the developed algorithm to also handle static obstacles. This visual representation effectively illustrates the algorithm's ability to accurately track the vehicle as it approaches the sensor, even

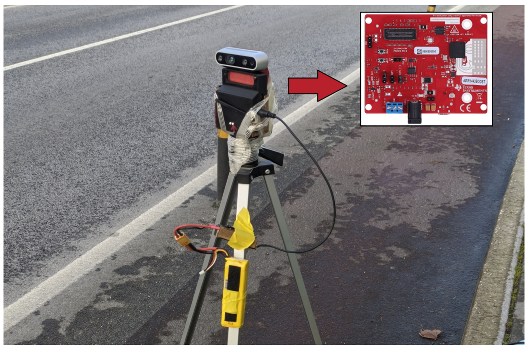


Figure 5.4: Sensor setup containing a TI mmWave radar [40] and a RealSense camera [39].

during periods when the tracking system receives no updates and must rely solely on its predictions. The blue ellipses indicate iterations where the Kalman Filter has not been updated, leading to increased uncertainty in the tracking.

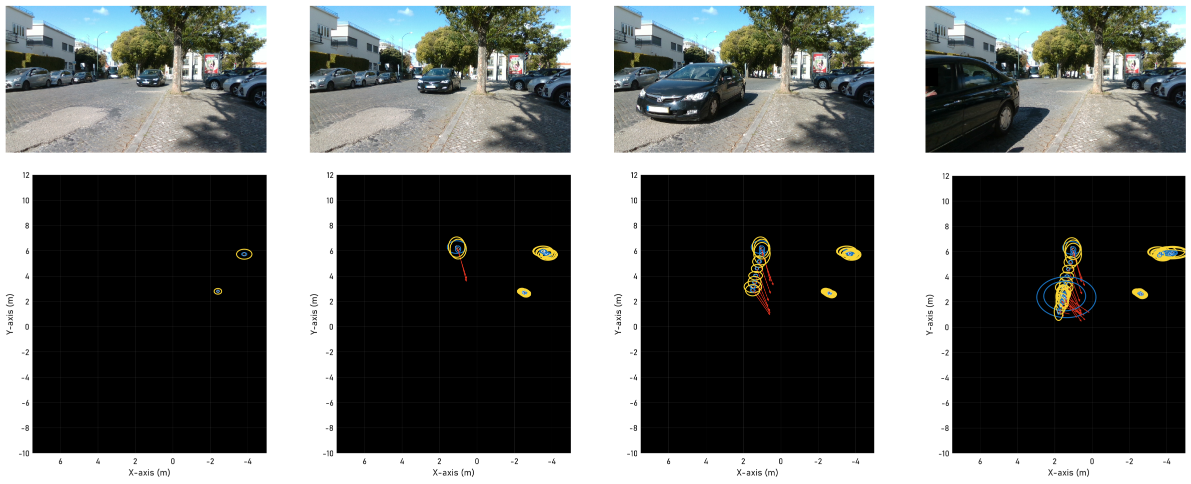


Figure 5.5: RGB snapshots of the scenario where a vehicle is moving toward the radar sensor.

Fig. 5.6 presents an example of a vehicle moving away from the sensor. From the snapshots of the tracked obstacles, we conclude that, with this radar and algorithm configuration, the maximum detection range is around 13m when the obstacle is moving away from the sensor. However, in the situation described in Fig. 5.5, where a vehicle is moving toward the sensor, the maximum detection range is between 6m and 8m. This discrepancy highlights the impact of accumulating recent frames over iterations while applying the time alignment method described in Algorithm 3, as each point's position is updated based on its estimated velocity, slightly increasing the number of detections at the limit of the radar's FoV.

Fig. 5.7 contains a larger version of the complete tracking of the vehicles mentioned above. As in Fig. 5.3, the centroids of the Kalman tracking are represented by small white circles, and the estimated confidence ellipses of the clustering algorithm results and the Kalman filter are represented by the yellow

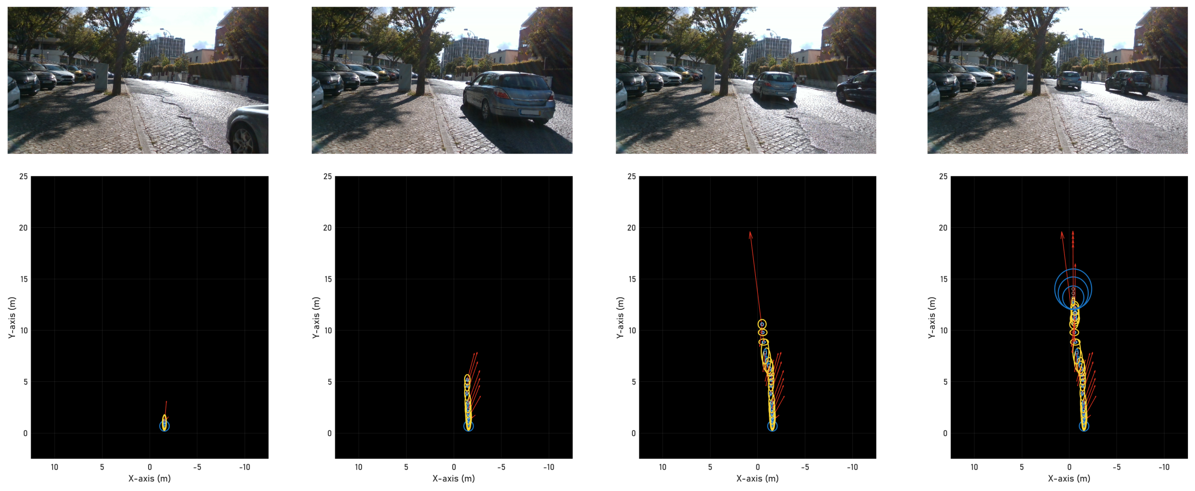
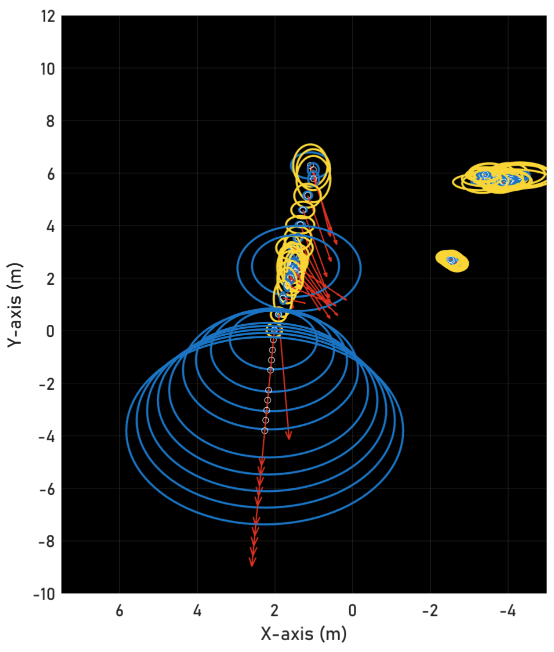


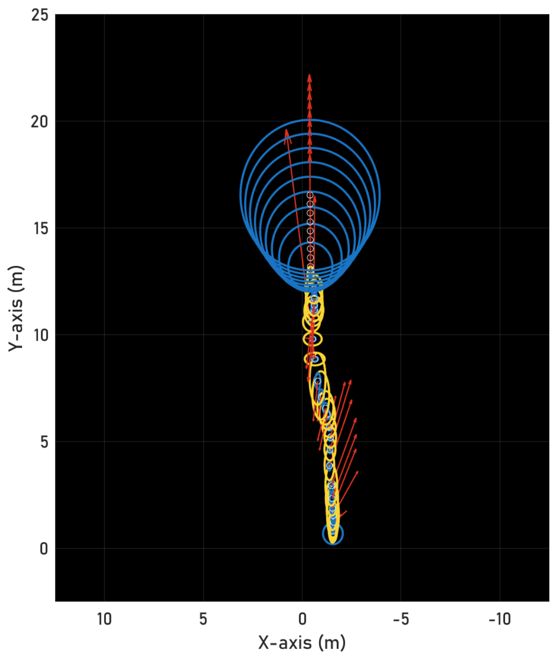
Figure 5.6: RGB snapshots of the scenario where a vehicle is moving away from the radar sensor.

and blue ellipses, respectively. Finally, the estimated velocity direction and intensity of the tracking are represented by red arrows.

Some tests were performed with the radar setup held by hand, creating a less stable scenario compared to using a static tripod. This introduced small oscillations around all axes. These tests simulate the oscillations that occur when a vehicle is moving. Although we don't have direct evidence, the tracking performance was not affected by these small oscillations. This suggests that in a scenario where the setup is mounted on a vehicle moving in a straight line, the developed algorithm can function effectively without the need of any tuning.



(a) Successful tracking of a vehicle moving towards the radar sensor.



(b) Successful tracking of a vehicle moving away from the radar sensor.

Figure 5.7: Tracking examples on scenes recorded with our setup.

6 Conclusion

Contents

6.1 Achievements

The objective of this thesis was to study the development of radar perception algorithms for obstacle tracking and classification. This objective was successfully achieved by developing a detection and tracking system based on mmWave radar technology. The system's characteristics, advantages, and limitations were thoroughly analyzed and discussed.

It was concluded that, while it is feasible to rely solely on the data generated by mmWave radars to create a simple, low computational complexity detection and tracking algorithm, its performance depends heavily on the specific radar configuration used. With this in mind, for some applications, this technology is most effective when combined with other sensors to compensate for the low resolution of the radar data.

An article [41] explaining the research conducted in this thesis was published and presented at the 2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). This publication underscores the current interest and relevance of this research topic.

6.2 Conclusions

The developed algorithm is capable of successfully detecting and tracking vehicles crossing the radar's FoV. It performs better at tracking vehicles moving away from the sensor compared to those moving toward it. This is due to the nature of the radar measurements, where the radial velocity is more accurate and useful when an obstacle is not too close to the sensor and not moving perpendicular to the radar's FoV.

6.3 Future Work

The performance of the algorithm is highly dependent on the quality of the radar data measurements. The radar configuration used could be further optimized, and finding the ideal configuration is something that should be done.

Further improvements to the algorithm itself could include estimating a minimum distance from the radar at which measurements cease to be useful, refining the Kalman Filter matrices, exploring the use of different state variables, and investigating alternative clustering criteria. Additionally, extrapolating bounding boxes for each obstacle based on the Kalman Filter state uncertainties would be beneficial. Integrating an IMU into the algorithm could improve the temporal alignment of data, particularly in dynamic scenarios, by compensating for small movements of the sensor.

The current model assumes that every obstacle can move in any direction and change direction at any time. Future work should incorporate more realistic kinematic models for each obstacle, as this can better constrain the possible movements, leading to reduced uncertainty when predicting vehicle motion. Finally, this algorithm should be evaluated with multiple radar sensors and in a non-static setup to assess its performance under varied and realistic conditions.

Bibliography

[1] World Health Organization, “Global status report on road safety 2023,” https://www.who.int/publications/i/item/9789240086517, Accessed: March, 2024.

[2] SAE International, “Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,” https://www.sae.org/standards/content/j3016_202104/, Accessed: May, 2023.

[3] G. Tuna, O. Arkoc, S. POTIRAKIS, and B. Nefzi, “Analyzing the water budgets of reservoirs by using autonomous mini boats,” *Key Engineering Materials*, vol. 605, pp. 51–54, 04 2014.

[4] V. Vaquero, I. del Pino, F. Moreno-Noguer, J. Solà, A. Sanfeliu, and J. Andrade-Cetto, “Dual-branch cnns for vehicle detection and tracking on lidar data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 6942–6953, 2021.

[5] C. Zhang, Z. Huang, M. H. Ang, and D. Rus, “Lidar degradation quantification for autonomous driving in rain,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3458–3464.

[6] M. Hahner, C. Sakaridis, D. Dai, and L. Van Gool, “Fog simulation on real lidar point clouds for 3d object detection in adverse weather,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 15 283–15 292.

[7] E. Harvey, M. Shortis, M. Stadler, and M. Cappo, “A comparison of the accuracy and precision of measurements from single and stereo-video systems,” *Marine Technology Society Journal*, vol. 36, no. 2, pp. 38–49, 2002.

[8] C. X. Lu, S. Rosa, P. Zhao, B. Wang, C. Chen, J. A. Stankovic, N. Trigoni, and A. Markham, “See through smoke: robust indoor mapping with low-cost mmwave radar,” in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, 2020, pp. 14–27.

[9] Z. Wei, F. Zhang, S. Chang, Y. Liu, H. Wu, and Z. Feng, “Mmwave radar and vision fusion for object detection in autonomous driving: A review,” *Sensors*, vol. 22, no. 7, p. 2542, 2022.

[10] A. Pandharipande, C.-H. Cheng, J. Dauwels, S. Z. Gurbuz, J. Ibanez-Guzman, G. Li, A. Piazzoni, P. Wang, and A. Santra, "Sensing and machine learning for automotive perception: A review," *IEEE Sensors Journal*, vol. 23, no. 11, pp. 11 097–11 115, 2023.

[11] T. Hachaj, "Potential obstacle detection using rgb to depth image encoder–decoder network: Application to unmanned aerial vehicles," *Sensors*, vol. 22, 9 2022.

[12] X. Zhao, H. Wu, Z. Xu, and H. Min, "Omni-directional obstacle detection for vehicles based on depth camera," *IEEE Access*, vol. 8, pp. 93 733–93 748, 2020.

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[14] H. Lee, H. Lee, D. Shin, and K. Yi, "Moving objects tracking based on geometric model-free approach with particle filter using automotive lidar," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 863–17 872, 2022.

[15] D. Xie, Y. Xu, and R. Wang, "Obstacle detection and tracking method for autonomous vehicle based on three-dimensional lidar," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419831587, 2019.

[16] J. Shi, Y. Tang, J. Gao, C. Piao, and Z. Wang, "Multitarget-Tracking Method Based on the Fusion of Millimeter-Wave Radar and LiDAR Sensor Information for Autonomous Vehicles," *Sensors*, vol. 23, no. 15, p. 6920, Jan. 2023.

[17] A. D. Singh, Y. Ba, A. Sarker, H. Zhang, A. Kadambi, S. Soatto, M. Srivastava, and A. Wong, "Depth Estimation From Camera Image and mmWave Radar Point Cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9275–9285.

[18] K. Bansal, K. Rungta, S. Zhu, and D. Bharadia, "Pointillism: Accurate 3D bounding box estimation with multi-radars," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, ser. SenSys '20. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 340–353.

[19] Texas Instruments, "The fundamentals of millimeter wave radar sensors," https://www.ti.com/lit/wp/spyy005a/spyy005a.pdf Accessed: May 2024.

[20] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[21] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[22] O. Schumann, M. Hahn, N. Scheiner, F. Weishaupt, J. F. Tilly, J. Dickmann, and C. Wöhler, "Radarscenes: A real-world radar point cloud data set for automotive applications," in *2021 IEEE 24th International Conference on Information Fusion (FUSION)*. IEEE, 2021, pp. 1–8.

[23] A. Palffy, E. Pool, S. Baratam, J. F. P. Kooij, and D. M. Gavrila, "Multi-Class Road User Detection With 3+1D Radar in the View-of-Delft Dataset," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4961–4968, Apr. 2022.

[24] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: Why and how you should (still) use dbscan," *ACM Trans. Database Syst.*, vol. 42, no. 3, jul 2017. [Online]. Available: https://doi.org/10.1145/3068335

[25] M. Götz, C. Bodenstein, and M. Riedel, "Hpdbscan: Highly parallel dbscan," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, ser. MLHPC '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2834892.2834894

[26] Z. Bi, Y. Gao, C. Wang, Z. Liu, Y. Wan, and X. Yang, "Recursive spatial-temporal clustering-based target detection with millimeter-wave radar point cloud," *Measurement Science and Technology*, vol. 34, 7 2023.

[27] S. Xie, C. Wang, X. Yang, Y. Wan, T. Zeng, and Z. Liu, "Millimeter-wave radar target detection based on inter-frame dbscan clustering," in *2022 IEEE 22nd International Conference on Communication Technology (ICCT)*, 2022, pp. 1703–1708.

[28] F. Liu and Y. Deng, "Determine the number of unknown targets in open world based on elbow method," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 5, pp. 986–995, 2021.

[29] C. Shi, B. Wei, S. Wei, W. Wang, H. Liu, and J. Liu, "A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm," *Eurasip Journal on Wireless Communications and Networking*, vol. 2021, 12 2021.

[30] A. Worasutr, D. Worasawate, T. Pongthavornkamol, and K. Fukawa, "Improved human detection algorithm by indoor w-band fmcw radar using k-means technique," in *2021 9th International Electrical Engineering Congress (iEECON)*, 2021, pp. 571–574.

[31] H. AbdEl-Gawad and V. K. Sood, “Kalman filter-based maximum power point tracking for pv energy resources supplying dc microgrid,” in *2017 IEEE Electrical Power and Energy Conference (EPEC)*, 2017, pp. 1–8.

[32] S. Yang and M. Baum, “Extended kalman filter for extended object tracking,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4386–4390.

[33] E. Wan and R. Van Der Merwe, “The unscented kalman filter for nonlinear estimation,” in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158.

[34] S. Pan, Y. Xie, K. Chen, Y. Zhang, and J. Mo, “Obstacle tracking based on the extended kalman filter and millimeter-wave radar,” in *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2022, pp. 1770–1775.

[35] Q. Zhang, K. Ji, Z. Feng, Z. Han, and H. Gao, “Vehicle behavior-cognition-based particle-filter-enabled mmwave beam tracking for connected automated vehicles,” *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21 292–21 304, 2022.

[36] H. Cui and N. Dahnoun, “High precision human detection and tracking using millimeter-wave radars,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 36, no. 1, pp. 22–32, 2021.

[37] W. Xu, W. ZhaYao, H. Zhao, and H. Zha, “A vehicle model for micro-traffic simulation in dynamic urban scenarios,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2267–2274.

[38] Texas Instruments, “mmWave ROS Driver,” 2024, https://dev.ti.com/tirex/explore/node?node=A__AFIEVaaBoBCgo.VCtJenZQ__radar_toolbox__1AslXXD__LATEST Accessed: May 2024.

[39] “Intel RealSense Depth Camera D435.” Website: https://www.intelrealsense.com/depth-camera-d435/ Accessed: May 2024.

[40] “Texas Instruments IWR1443 mmWave Sensor.” Website: https://www.ti.com/product/IWR1443 Accessed: May 2024.

[41] R. Carvalho and A. Vale, “Exploring millimeter-wave radar algorithms for obstacle detection and tracking,” in *2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2024, pp. 82–87.