Low-Cost Dual Radar System for Collision Prediction at Unregulated Intersections

Presented by:

Dayalan Nair

Prepared for:

Dr. M. Y. Abdul Gaffar

Dr. S. Winberg

May 21, 2023

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for the degree of MSc(Eng) in Electrical Engineering.

Abstract

Automotive collision avoidance has become a key aspect of the automotive design process. The shift in the automotive market towards reducing road accidents has led to the development of many sensor-based collision avoidance systems. The majority of such systems are limited to high-end vehicles. Statistics indicate road intersections are a common location for car accidents. In South Africa, right turns at unregulated intersections require drivers to observe traffic in both directions, which demands substantial attentiveness. This project investigated the suitability of a low-cost bidirectional radar system for determining the safety of turning right at unregulated intersections. The focus was on the detection of four-wheeled vehicles, where frequency-modulated continuous-wave radar facilitated the measurement of the range and speed thereof. An algorithm was designed for detecting multiple targets on a two-lane, bidirectional road using the triangle FMCW waveform. MATLAB simulations ensured selected hardware and processing algorithm met the user requirements. Python allowed for testing real-time processing performance. A roadside test rig consisting of two cameras, two radars, a Raspberry Pi, and a power bank was developed for field testing the proposed system. Performance was validated by comparing radar speed and range estimates to both video footage and GPS estimates. Findings suggested that the final prototype met the project requirements, with a few caveats. The system struggled to detect outer-lane vehicles during peak traffic due to the masking of these vehicles by those travelling on the inner lane. Placing the left radar at an angle reduced the effect of masking and reduced returns from cars travelling away from the host vehicle. The speed estimates had a standard deviation of 8 km/h and 11 km/h for the right side and left side radar, respectively, which was an order of 10 larger than that of GPS estimates.

Acknowledgments

Firstly, a deep thank you to my supervisors Dr Yunus Abdul Gaffar and Dr Simon Winberg for the swift, thoughtful, and consistent feedback and for providing most of the required hardware. Your contributions streamlined the research and development process. I appreciate the time taken to review each section multiple times to ensure the research was correctly presented.

I am profoundly grateful to the many laboratory and work colleagues for lending an ear when needed and for all the assistance, conversation, and motivation. Your input enriched both the project and the investigation process. In particular, I would like to thank Grant Norrie for assisting with experimentation and for setting me up in the laboratory. Thank you to my friends Zaid Karjikar and Tom Robbie for taking the time to drive up and down repeatedly so that I could collect data. A massive thank you to Johan Blaauw and Clifford van Dyk for assistance with the last-minute angle adjustment component and installation thereof. Mark Cammidge, thank you for providing valuable comments and tips regarding the dissertation write-up.

Finally, I would like to thank my parents, friends, and family for the unending love and support received throughout this tumultuous period. This project has made me realise the power of a solid support system, without which it would be significantly more arduous.

Plagiarism Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signed by candidate

Dayalan Nair

May 21, 2023

Contents

List of Figures

List of Tables

Abbreviations

ACAS Automotive Collision Avoidance System 3, 5, 10

ADAS Advanced Driver Assistance System 5

ADC Analogue-to-Digital Converter 36

ADT Automated Driving Toolbox 79

AEB Autonomous Emergency Braking 5, 12

AoA Angle of Arrival 63

CAD Computer-Aided Design 47, 49

CFAR Constant False Alarm Rate 29

FCW Forward Collision Warning 12

FFT Fast Fourier Transform 29, 67

FMCW Frequency-Modulated Continous Wave 29

GIS Geographical Information Systems 11

GPS Global Positioning System 44

GUI Graphical User Interface 54

I/Q In-Phase and Quadrature 46

PAST Phased Array System Toolbox 79

SDK Software Development Kit 52, 55

T2T Time-to-Turn 45, 46

TOA Time of arrival 6, 118

V2I Vehicle-to-Infrastructure 13

V2V Vehicle-to-Vehicle 10, 18

Chapter 1

Introduction

1.1 Introduction

Throughout the world, road vehicles are a common mode of transport for goods and people alike. Road safety is a concern for both drivers and pedestrians. Consequently, it is an area of interest for researchers and car manufacturers alike [1].

Today, automotive collision avoidance systems (ACAS) utilise multiple sensors such as LiDAR [2], cameras [3], sonar [4], and radar [5]. The use of radar for vehicle collision avoidance began in the 1960s and proved highly effective [5]. Since radar can operate in poor lighting and weather conditions, it is especially suited to the road environment. The ability to estimate range, radial velocity, angle of arrival, and reflectivity patterns offers vast possibilities for target observation and classification. Current applications include blind spot detection [6], adaptive cruise control [7], and park assist [8].

Lane-crossing turns at road intersections require sound observations and mental calculations by the driver, as traffic moving in both directions needs to be analysed. Unfortunately, all intersections cannot be regulated by traffic lights due to feasibility, risk, and throughput. An example is the intersection of streets and main roads in urban and suburban areas. In South Africa, where vehicles travel on the left-hand side of the road, right turns from streets onto main roads require lane crossing. The driver must observe vehicles travelling in both directions, including pedestrians, bicycles, motorcycles, and cars on the opposite

street (collectively referred to as actors). Poor weather such as rain or fog, low lighting after sunset in areas of poor street lighting, and visual obstructions further increase the difficulty of safely navigating this scenario. This project focuses on the development of a system to assist in performing a right turn at uncontrolled road-street intersections. Figure 1.1 provides a visualisation of the scenario, showing how the radars are arranged.

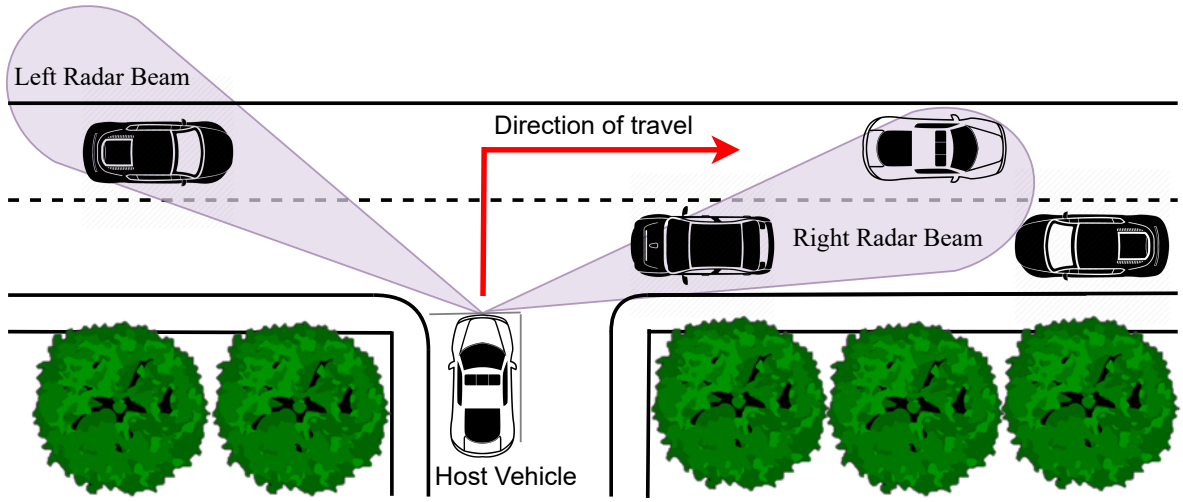


Figure 1.1: Unregulated intersection showing the radar collision prediction system for the right-turn scenario

1.2 Background

According to the World Health Organisation (WHO), low- and middle-income countries account for 93% of road fatalities despite having only 60% of the global number of vehicles [9]. This is due to a multitude of factors, such as road infrastructure and accepted vehicle safety ratings [9]. Significant risk factors include speeding, driving under the influence, distracted driving, non-use of seat belts, unsafe road infrastructure, inadequate law enforcement and traffic laws, and post-crash care [9]. In South Africa, more people die on the roads compared to the global average: 26 compared to 18 per 100 000 globally [10]. This has been described as a national crisis by the Automobile Association of South Africa [11].

The Insurance Institute for Highway Safety (IIHS) in the United States performed several studies into ACAS. In March of 2022, a summary of the benefits of ACAS was released [12], with the most notable statistics being a 78% reduction in reverse crashes, and a 50% reduction in front-to-rear crashes. A compendium [13] compiled by the Highway Loss Data Institute (HLDI) in the United States summarises the reduction in insurance claim frequency. Notably, the front and rear automatic emergency braking (AEB) systems accounted for roughly a 25% and 28% reduction in claim frequency, respectively [13].

Though many road accidents are due to external/uncontrollable factors, some can be mitigated using vehicle collision avoidance and prediction technologies. As of 2016, a five-star rating by the European New Car Assessment Programme (NCAP) requires that at least one advanced driver-assistance system (ADAS) [3] be installed [14]. Collision avoidance systems serve to improve driver perception, reducing collision probability. This is achieved by the optimal placement of sensors and their tolerance to low-visibility conditions.

An unregulated intersection is one where the right-of-way is not dictated by traffic lights, stop signs or road markings. This is regarded as one of the most complex and hazardous driving scenarios faced by road users [15]. According to the U.S. Department of Transportation, over 50% of accidents are intersection-related [16]. Additionally, 50% of serious collisions and 20% of fatal collisions occur at intersections [17]. Ideally, all intersections should be controlled by traffic lights. However, this is not feasible and is only efficient when throughput is sufficiently high in all directions. It is simply more efficient to let the driver identify when it is safe to turn in the low throughput case.

1.3 Problem Statement

Based on the accident statistics presented in Section 1.2, the need for collision avoidance systems is clear. Without considering driver negligence, navigating certain road scenarios is inherently risky. A good perception of the scenario and vehicle proportions is required by drivers to safely navigate complex scenarios without accident. Current technologies and systems are not without flaws. All sensors are affected by weather, interference, missed detection or false alarms due to irregularly shaped vehicles or clutter. Such sensors require

an unobstructed line of sight to work correctly. Generally, radar sensors offer superior tolerance to weather conditions and can penetrate through non-metallic obstructions. A holistic approach to collision avoidance requires all vehicles to be fitted with ACAS and ADAS technologies. In so doing, motorists are assured that they will not be the cause or the recipient of collisions.

In the IIHS summary of ACAS benefits [12] it was noted that such technologies increase the cost of post-accident repair since the sensors are located around the exterior of a vehicle. Generally, advanced ADAS and ACAS technologies are limited to high-end vehicles due to high development costs.

Most intersection accidents occur during the lane-crossing turn scenario [17]. For this scenario to be navigated safely, the driver must perform several mental calculations and observations. Vehicles approaching from all possible directions need to be observed in addition to pedestrians, bicycles, and motorcycles. The difficulty of the task is increased by weather and visibility, driver alertness, and human error. Currently, most cross-traffic alert systems are used at the rear of the vehicle and detect (rather than predict) whether a nearby car is blocking the direction of travel [18] [19].

In the proposed work, a dual radar system was developed to estimate the time of arrival (TOA) of vehicles travelling in each direction along a two-lane, bidirectional main road. The TOA was based on the range and velocity radar estimates of the approaching vehicles relative to the host vehicle. Based on the TOA of vehicles in each lane, the system notified the driver when it was safe to turn. The turn safety should be displayed visually, with an audible warning if the driver attempts to make an unsafe turn. Figure 1.2 shows how the proposed system could be integrated with existing automotive radar systems. Additionally, Fig. 1.3 shows how the radar placement allows for superior visibility compared to the driver's viewing position.

1.4 Objectives

The main research objective was to investigate whether or not a low-cost dual radar system could be used to determine right-turn safety at unregulated intersections based on the TOA

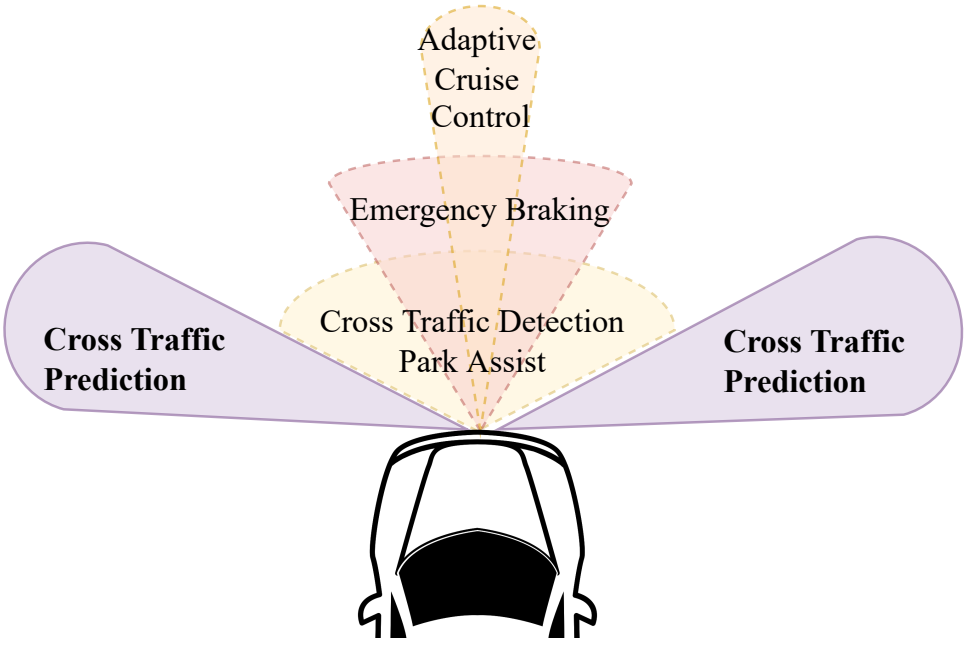


Figure 1.2: Frontal collision avoidance systems incorporating the cross traffic prediction system

of cars approaching from both perpendicular directions. Specific research objectives were as follows:

1. Identify user and system requirements following a literature review. These requirements are used for hardware selection and system validation.

2. Design simulations of the dual radar system in various intersection scenarios. Design a signal processing algorithm (to extract target speed and range parameters) and a data processing algorithm (to determine turn safety), considering the scenario and radar hardware. Assess the algorithms through simulations.

3. Design a prototype for roadside deployment. This allows for experimentation without blocking the flow of traffic and simplifies the prototype design process when compared to a vehicle-mounted system.

4. Collect and process radar data from controlled and uncontrolled experiments offline and online (in real-time) and analyse the results.

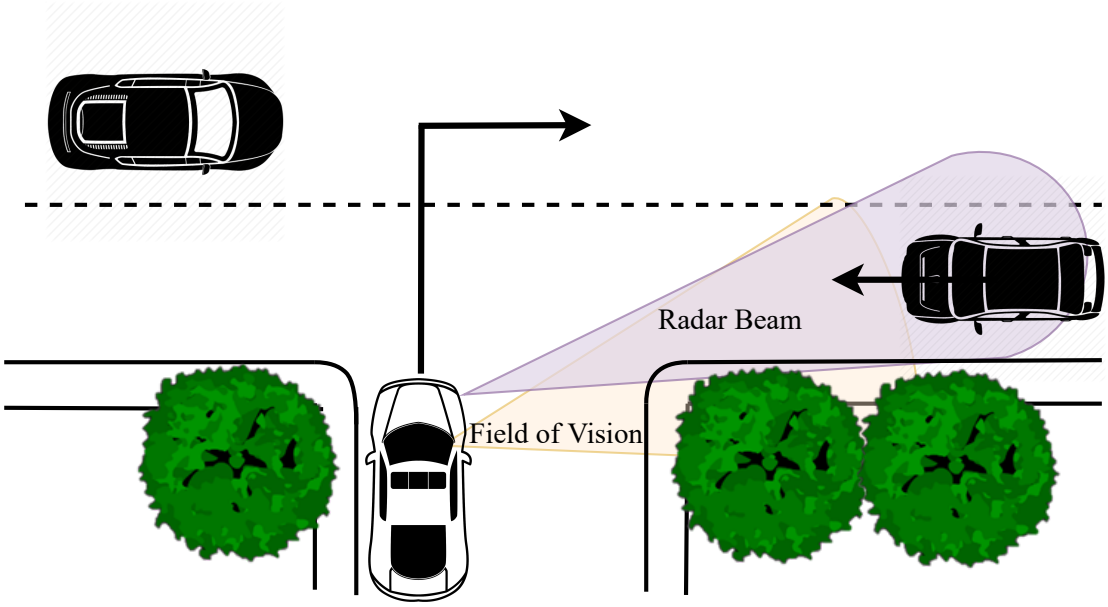


Figure 1.3: Improved perception provided by a perpendicular front-mounted radar

1.5 Scope and Limitations

Limitations of the proposed system were governed by the available time and budget. Specifically, the time equates to 1 200 hours (based on the credit allocation for a partial dissertation at the University of Cape Town) and the budget was set to R20 000. The limited time influenced the project scope, so only core functionality was implemented. The limited budget had little effect on the system since the system is expected to be low-cost (relative to equivalent radar systems). The scope was as follows:

1. Only a single lane was observed in each direction. Since the closest vehicle sets the speed of vehicles behind it, closely spaced vehicles can be considered a single hazard. Consequently, good range resolution was not important.

2. Only four-wheeled motor vehicles travelling towards the host vehicle on a single lane in each direction were to be detected.

3. The design of an application-specific circuit was not considered as this forms part of commercialisation.

4. Processing was implemented using scripting languages as opposed to a compiled software program.

5. A custom radar was not designed. Rather, off-the-shelf hardware was selected. The effect of various weather conditions on performance was not investigated.

6. Both controlled and uncontrolled scenarios were used for testing and data collection. Only a hatchback and compact sedan were used for controlled testing.

1.6 Thesis Outline

The remainder of this thesis is organised as follows:

Chapter 2, Literature Review: This chapter presents road accident statistics and reviews of current ADAS and ACAS methods and technologies. Additionally, a theoretical framework is presented.

Chapter 3, Methodology: This chapter describes the steps, procedures, and reasoning employed while designing the proposed system.

Chapter 4, Design: The design and development of algorithms, software and prototypes are described. Detail is provided on signal processing techniques and prototype design.

Chapter 5, Simulation: The process of simulating the radar hardware for the intersection scenario using MATLAB is presented.

Chapter 6, Implementation: This chapter describes the implementation of the designed system. Python scripts for processing and plotting data in real-time and the test rig construction process is presented.

Chapter 7, Results: Experiment results are presented here, along with discussions and comparisons to GPS estimates and video footage.

Chapter 8, Conclusions: Conclusions are drawn from the results and future work is presented.

Chapter 2

Literature Review

This chapter provides context to the work by outlining the problem and discussing existing solutions. It begins with a brief review of road intersection safety statistics, particularly those relating to the unregulated intersection right-turn scenario. Various automotive collision systems for intersection safety are presented, focusing on radar-based systems. Standalone systems, which do not require the sharing of information, are analysed in further depth, considering the various strengths and shortcomings of the communications-free technology. A theory section is presented, providing a basis for the concepts and algorithms applied during the design stage of this work.

2.1 Background and Current Systems

This section presents research on collision statics, particularly those at road intersections, and the analysis thereof. Current automotive collision avoidance systems (ACAS) are then presented, focusing on the three main categories: vehicle-to-infrastructure (V2I) communication, vehicle-to-vehicle (V2V) communication, and standalone systems.

2.1.1 Automotive Collision Statistics

In South Africa, the Road Traffic Management Corporation (RMTC) provides various traffic reports and statistics, examples of which include the State of Road Safety Report (January

to December of 2021) and the South African Fatal Crashes in Context (December 2021) [20]. The latter contains information on fatal crashes (FC) and speed infringements (SI) based on a variety of vehicle parameters and combinations thereof. Light passenger vehicles make up around 53.8% of fatal crashes and 79.6% of speed infringements. Regarding make and model, low-cost vehicles make up a significant portion of these statistics, though one should consider these parameters in relation to the percentage of the total vehicle population being considered. Public transport minibuses have the highest percentages of FC and SI compared to their percentage of the total vehicle population. These statistics indicate that low-cost ACAS systems could improve the vast majority of accidents in South Africa. Though many useful statistics are presented, an analysis of the road scenarios in which accidents occur has not been presented.

Research by [21] provides a thorough analysis of the accident statistics at road intersections in the Turkish city Antalya between 2009 and 2010. Geographical Information Systems (GIS) refer to all the processes relating to traffic data and analysis. The data originated from traffic accident reports, which contained information such as the accident type and road conditions at the time of the accident. The large data set was used for a deep analysis, providing links between previously unrelated data sets. The benefit of GIS is that it can establish relationships between accident attributes and spatial objects. Essentially, it can identify if the road design or surroundings are conducive to accidents. The study found that 99.12% of intersection accidents result in injury. The four-way intersection accounted for around 60% of all intersection accidents, with around 68% of all intersection accidents resulting in sideswipes (parallel collision). It was found that unregulated intersections accounted for around 73% of the total accidents at road intersections.

A study on the cause of car accidents at intersections [22] formulated five fault trees which represent almost all possible road accident causes. Figure 2.1 is one such example. Fault Tree Analysis is used to determine the relationship of various events within a system [22]. Fault trees allowed for the identification and analysis of the dominant events that lead to accidents at road intersections. Such events reveal where attention should be paid when designing systems for reducing such accidents.

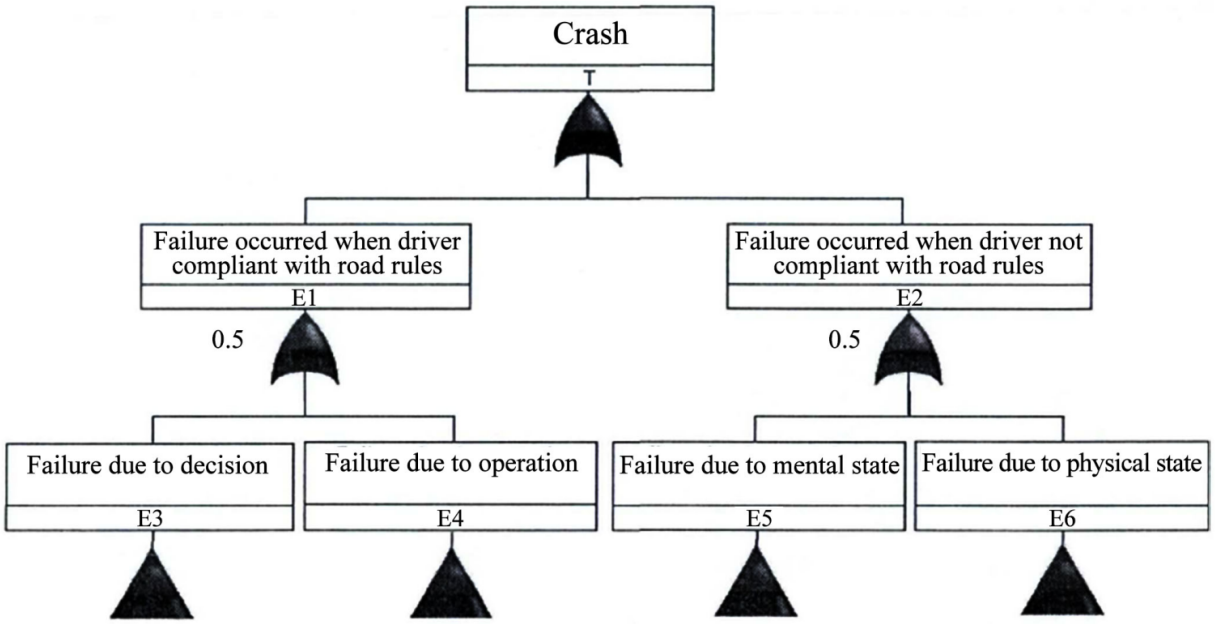


Figure 2.1: Top-level fault tree for road collisions [22]

2.1.2 Current Systems and Technologies

There are two categories of collision avoidance systems: collision avoidance warning and collision avoidance assist systems [23]. Collision avoidance warning systems alert the driver of possible collisions using lights, sound, or vibration. Examples include forward collision warning (FCW) [24], blind-spot detection [6], and cross-traffic warning [25]. Since these technologies have not been perfected, false alarms and other erroneous behaviour cannot be ruled out. This is understandable when one considers the complexity of the scenarios and the restrictions on sensor placement and orientation.

Collision avoidance assist systems are connected to the vehicle control mechanisms and serve to prevent collisions by preventing the driver from performing an action that will result in a collision. Systems include automatic emergency braking (AEB) [24], adaptive cruise control [7], electronic stability control, and park assist [8]. Since these systems are far more complex than warning systems and require rigorous testing, they are often confined to high-end vehicles such as those offered by Lexus [25].

Currently, systems typically use a variety of sensors (LiDAR, radar, cameras) working together to determine various surrounding conditions [26]. The integration of different types

of sensors is referred to as sensor fusion. The efficacy of currently available systems is highly variable and scenario dependent. An example is the list of conditions provided by Mazda for its Front Cross Traffic Alert (FCTA) system [27]. A subset of these are false alarms arising from nearby obstacles, interference from similar systems, and missed detections during bad weather conditions or for irregularly shaped targets, such as sports cars. Some companies offer these as standalone systems, such Bosch, Mobileye, and Continental AG. These companies offer powerful systems, generally combining front collision warning systems with others such as lane departure warning and cameras. At the time of writing, pricing could not be found, and specific intersection collision warning systems were not provided. Another interesting product line is Cohda Wireless' Vehicle-to-everything (V2X) systems. Such systems include roadside units and software applications.

Systems such as V2V [28] and V2I [29] communication are promising. Data generated by the vehicle or roadside infrastructure is shared between nearby vehicles to determine safety in various road scenarios. These systems are highly complex, requiring a great deal of surrounding infrastructure. Given that [28], [29], and [30] have been published recently, these technologies are far from maturity.

2.2 Vehicle-to-Infrastructure Communication

Vehicle-to-Infrastructure (V2I) communication relies on roadside units (RSU) to determine intersection safety [29]. Components of such systems include communication networks and protocols, roadside and onboard sensors, and dedicated processing systems. Such systems can use a combination of communication standards, such as dedicated short-range communications (DSRC), Bluetooth, and WiMAX [31]. Though the reliance on infrastructure is a major challenge for V2I communication systems, they offer a far higher probability of accident mitigation compared to onboard, isolated collision avoidance systems. This is because the sensors and placement thereof are not bound to a host vehicle and can be optimally positioned around an intersection. A variety of sensors can be arranged in various topologies and can be adapted to any road scenario. The flexibility and performance of V2I communication systems make them an ideal solution for automotive collision avoidance.

There are, of course, a notable number of downsides to V2I communication infrastructures. Firstly, the cost of such systems is substantial. Examples of hardware include cloud servers, sensors, processing units, RSUs, and vehicle-installed onboard units (OBU) [31]. For road intersections, link breakage and message collisions may occur due to vehicles moving at relatively high speeds [31]. The maintenance and protection of RSUs and OBUs is yet another factor affecting the suitability of such systems.

2.2.1 Collision Probability at Intersections

Intersection management and collision avoidance algorithms need to be designed to ensure robust safety estimations and efficient data transfer. These algorithms are often based on collision probability models. One such model was developed by [32]. This model required vehicles to share their current position, speed, and turning direction with a roadside unit (RSU). The RSU determined possible collisions based on the connected vehicles' dimensions and predicted trajectories. This was achieved by calculating the time to collision (TTC) based on the intersection of the predicted tracks of all detected vehicles. No distance or speed sensing was required: all relevant information is captured and shared by the surrounding vehicles. Essentially, the RSU behaves as an external observer. Trajectory prediction is handled using the Simulation of Urban MObility (SUMO) traffic simulation package [33]. It was found, based on the correlation between the simulation and a Gaussian curve, that a Gaussian function is well-suited to predicting the trajectory of turning vehicles. Table 2.1 shows the correlation of different functions to various possible intersection manoeuvres.

Table 2.1: Correlation between mathematical functions and intersection manoeuvres [32]

	Straight	**Left turn**	**U-turn**	**Right turn**
Gaussian	0.21	0.98	0.97	0.94
Line	0.99	0.32	0.35	0.44

2.2.2 Communications Architecture

The communications architecture proposed by [34] builds on the V2I communication idea, focusing on the communication protocol used by such systems. The objective was to establish the communication link as soon and as early as possible. A Dedicated Short Range Communication (DSRC) system was proposed, which used seven non-overlapping channels to share information between nearby vehicles. One channel was dedicated to control operations, with the other six used as service channels for non-safety data. The channels were used to create two zones: the service channel zone (SCHZ) and the control channel zone (CCHZ) as shown in Fig. 2.2. An appropriate communication architecture required fast transmission and low packet loss which is crucial in high-traffic scenarios. This was especially important since all information was routed through, and processed by, the RSU. Based on this requirement, three methods were chosen for comparison: shared channel, dedicated channel, and competition-based [34]. The distance of the vehicle from the intersection determined which channel it was placed in, with farther-away vehicles placed in the control channel and nearby vehicles placed in the service channel. In [34], it was found that the dedicated channel method, requiring one RSU for control and another for service, was most attractive when traffic density was high. Unfortunately, the evaluation of these methods was left as future work. At the time of writing, no future work relating to this study [34] could be found.

2.2.3 Wireless Sensor Networks

Wireless sensor networks were used to monitor traffic by [35]. Possible collisions were detected by a set of sensors and relayed to the relevant drivers. The sensor nodes consisted of a magnetic sensor, radio transceiver, microcontroller, and batteries. The magnetic sensor could detect vehicles and measure velocity with high accuracy, using the time difference of arrival (TDOA) between multiple nodes. Multiple sensors relayed measurements to a base station, which computed the differential estimation of the velocity and predicted collisions for vehicles approaching an intersection. It is noted that for such sensor networks to work, the sensors must be time synchronised and should operate in a deterministic manner to

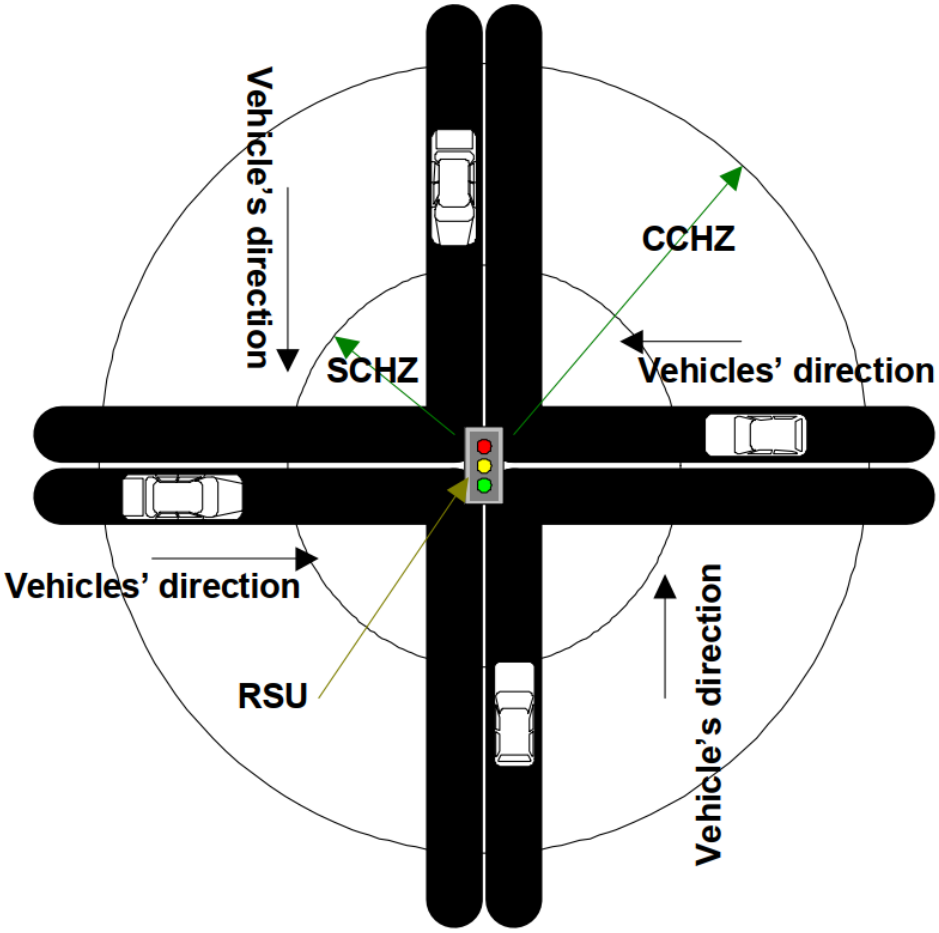


Figure 2.2: Illustration of the communications architecture showing the two zones and RSU placement [34]

optimise accuracy. Figure 2.3 shows the proposed placement of magnetic sensors at road intersections.

Research by [36] investigated the effects of sensor spacing, the distance of sensors from the intersection, acceleration variance of vehicles, and measurement noise on intersection collision avoidance. Only simulations were performed due to the high cost of putting cars on a collision course. It was concluded that five magnetic sensor nodes spaced ten metres apart offered a collision prediction rate of 96%. The research recommended using six nodes to account for sensor malfunction and that performance could be improved by using more sensors. However, it is noted that such a deployment is expensive and that physical limitations exist. Regarding magnetic sensors, a changing temperature was identified as a

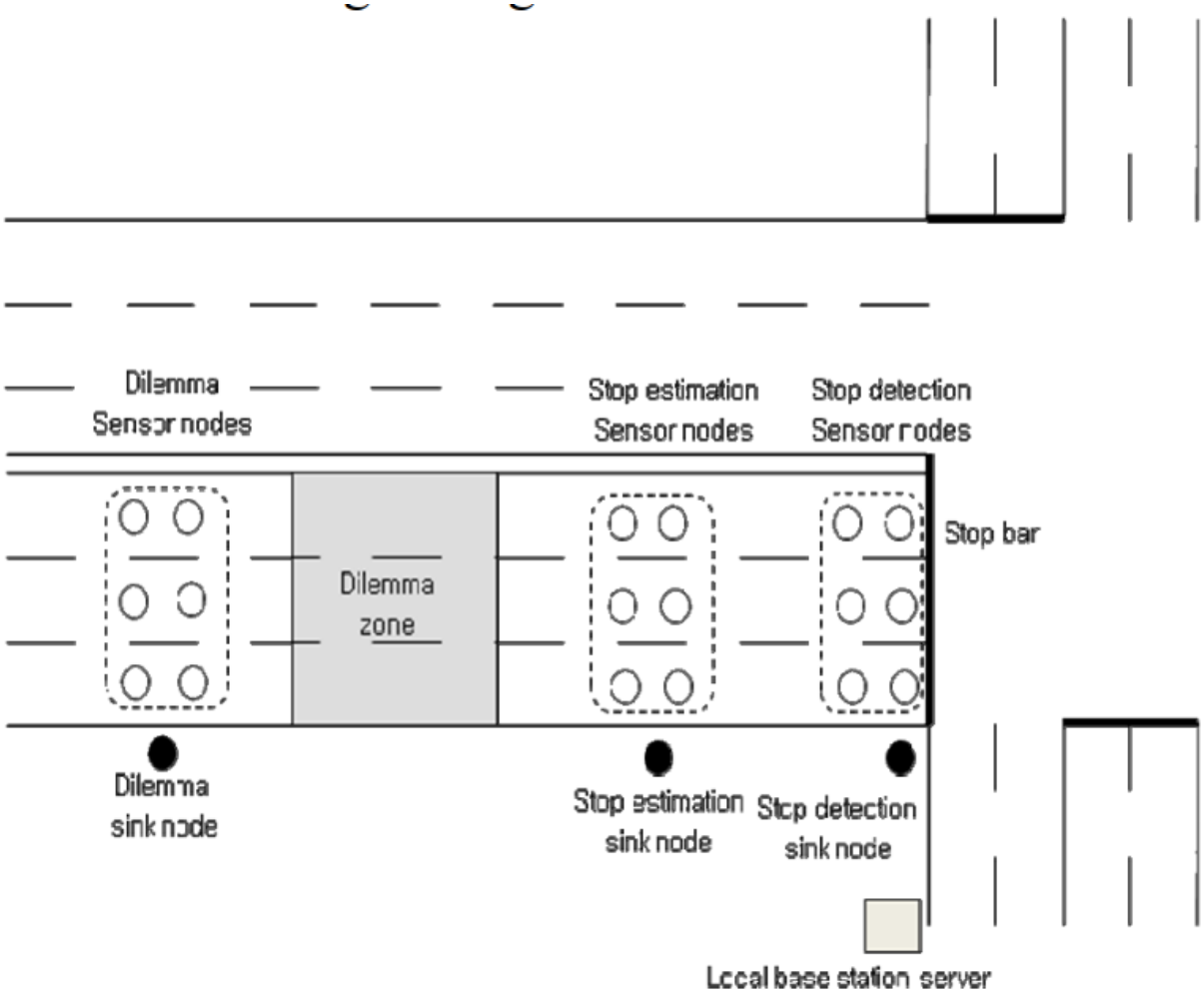


Figure 2.3: Placement of magnetic sensors as investigated by [35]. Sink nodes determine vehicle speed based on time difference between two dilemma node detections.

possible cause of errors within the system. It was noted that the requirement for multiple sensors introduced multiple points of failure within the system.

2.2.4 Automotive Radar with Passive Reflector

An investigation into the use of a passive radar reflector for uncontrolled blind intersections was undertaken by [14]. Firstly, to optimise performance, the shape and angle of such a reflector were carefully considered. It was found that a convex reflector of radius 20 meters with a 43.5°offered the best performance [14]. Results indicated that the system had

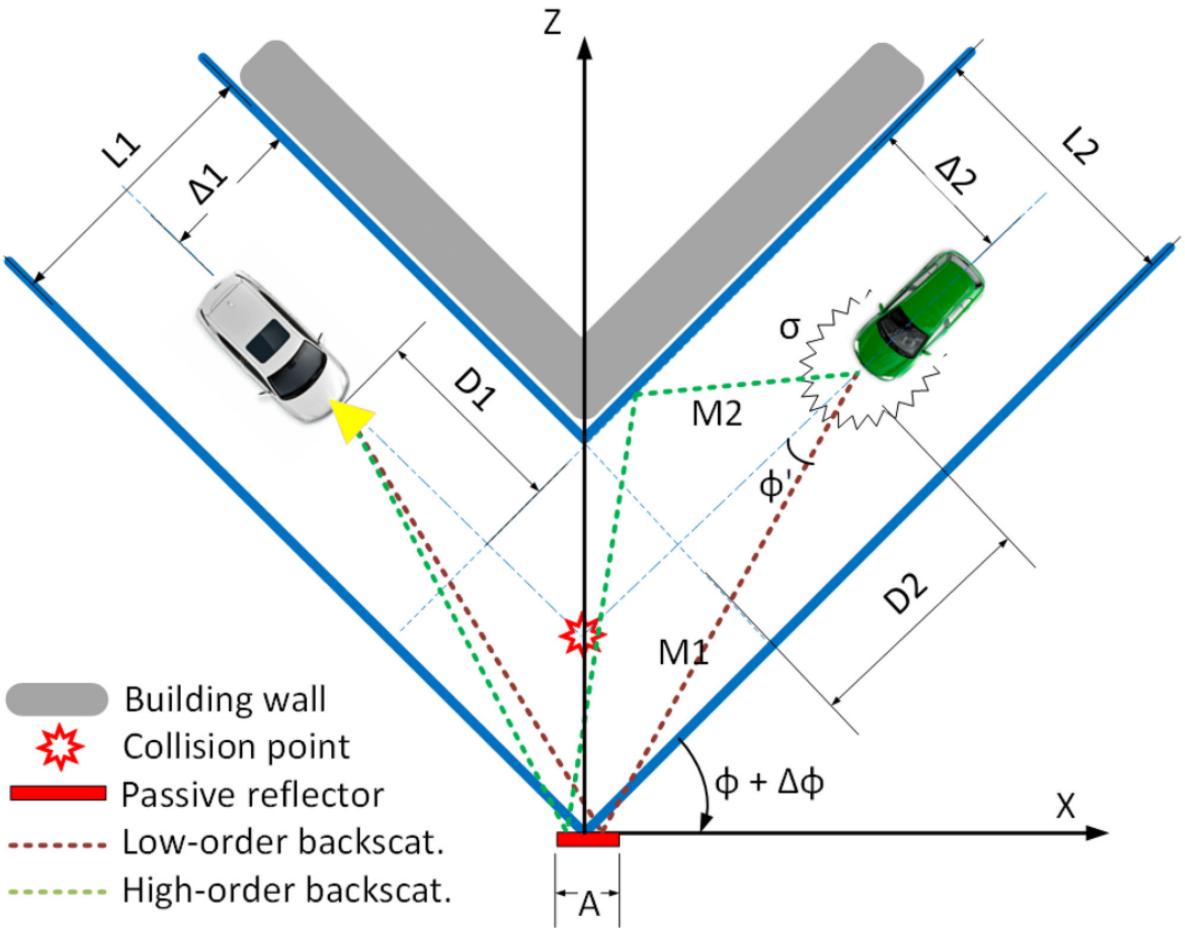


Figure 2.4: Illustration of the use of passive reflectors for avoiding collisions at blind intersections [14]

poor clutter rejection due to the many possible propagation paths introduced. Since the transmitted energy must be reflected three times before it is received, there is an increased propagation and scattering loss. Consequently, a relatively large amount of transmitted power was required. The scenario is presented in Fig. 2.4.

2.3 Vehicle-to-Vehicle Communication

Vehicle-to-Vehicle (V2V) communication requires vehicles to share information such as speed and position, which is then processed onboard to determine if collisions are imminent. Similar to V2I communication systems, V2V communication does not require line-of-sight

to determine turn safety. The benefit of these systems over V2I is that less infrastructure and maintenance are required.

The downsides of these systems are primarily security related. Hacking and interference pose a significant threat to the safety and privacy of drivers. Dedicated communication protocols and security standards are required to ensure these systems become widely adopted [37]. Examples of possible V2V attacks include message spoofing, replay attack, integrity attack, and impersonation [37]. Several encryption methods are presented in [37], with varying degrees of speed vs. security. For V2V systems to operate in real-time, communication needs to be established quickly as latency could result in collisions, especially during high-speed scenarios. Note that security and communications protocols apply equally to V2I communication systems.

2.3.1 GPS-based Collision Avoidance

Inter-vehicle communication using a DGPS (Differential Global Positioning System) and gyroscope was investigated in the year 2000 [38] and was proven to be a successful first step in preventing collisions at off-sight intersections. It worked as follows: DGPS was used to determine if the host vehicle was approaching an unregulated intersection. If so, a warning signal was sent out. If this signal was received by another vehicle, information such as speed and position was exchanged. Finally, the drivers were issued a warning should they be on a collision course [38]. DGPS allows for increased accuracy in GPS measurements by transmitting correction data via a nearby base station. Correction data is proportional to the difference between the satellite estimation of the base station's location compared to its known coordinates. Radar was deemed ineffective in off-sight detection scenarios due to reflections and attenuation from trees and buildings [38]. The scenario is depicted in Fig. 2.5 which illustrates how a sensor-based system is blinded by a large building. It was found that the gyroscope accuracy had significant drift, which coupled the system's accuracy with the period over which the prediction is made. The system assumes that neither driver will stop or yield at the intersection. This is highly unlikely, as a driver intending to turn must slow down and observe the road. A side-looking radar can provide effective visual assistance when the driver follows the rules of the road.

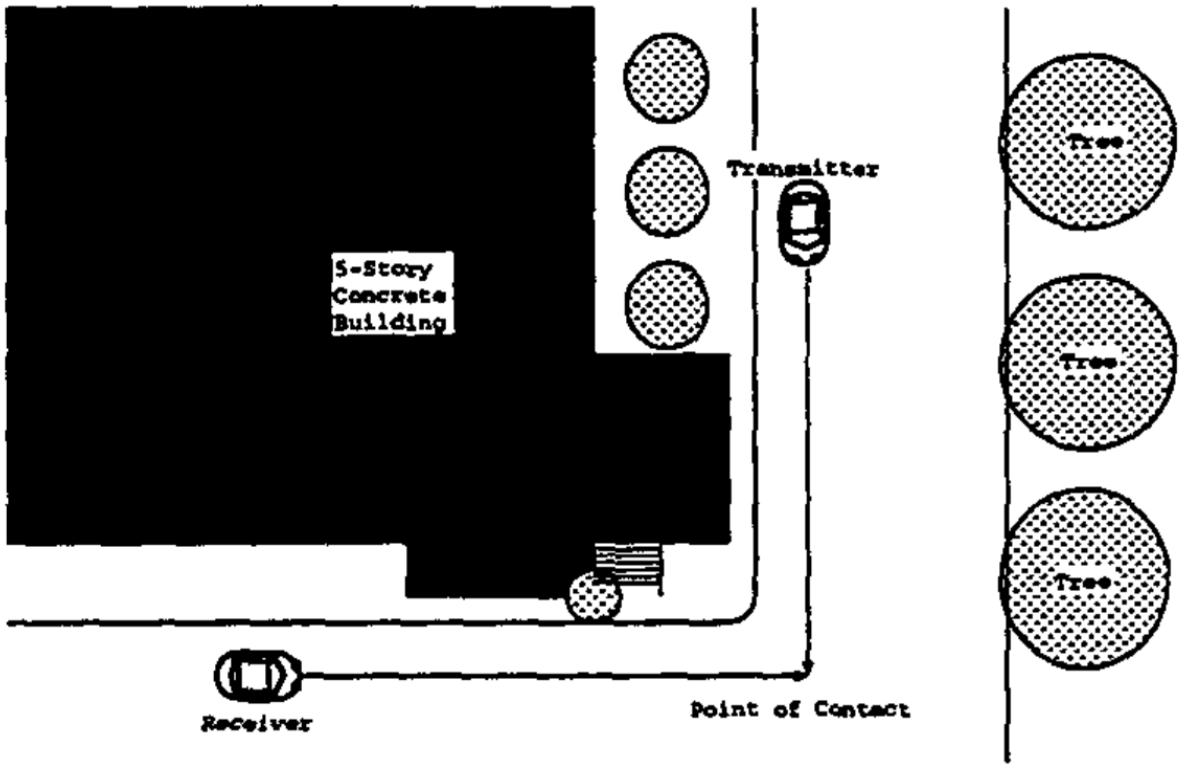


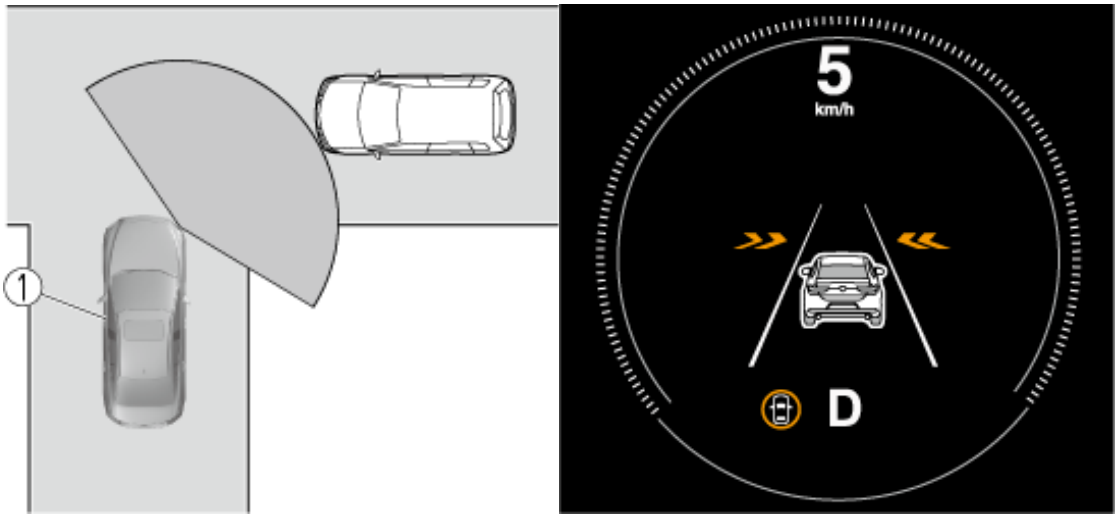
Figure 2.5: Blind intersection where the use of DGPS and gyroscopes for collision prediction was investigated [38]

Another GPS-based system is presented in [39]. A successful warning rate of 81.7% was produced, though only simulations were carried out. Since simulations are often idealistic representations, further work is required to determine if this solution is indeed valid.

2.4 Onboard Intersection Collision Avoidance

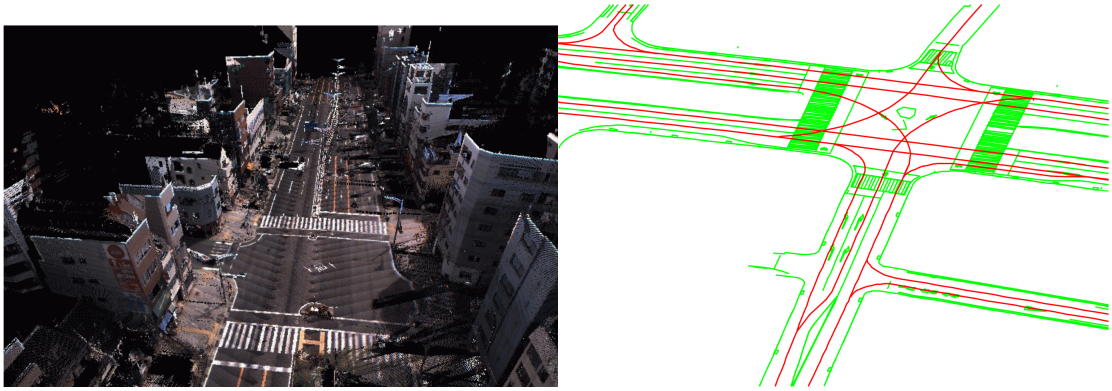
This section describes systems which solely rely on onboard sensing and processing. These systems utilise various sensors to enhance driver perception and reduce the reliance on driver decision-making. Two commercial examples include Lexus' and Mazda's front cross traffic alert (FCTA) systems, presented in Figures 2.6, 2.7a, and 2.7b respectively.

Figure 2.6: Lexus front cross-traffic alert (FCTA) system



(a) Radar beam pattern (b) User interface

Figure 2.7: Figures showcasing the Mazda front cross-traffic alert (FCTA) system



(a) High density point cloud (b) Vector map

Figure 2.8: Methods used for blind area traffic prediction [40]

2.4.1 LiDAR and High Definition Maps

Research presented in [40] tackled the issue of occluded lanes in urban areas. It noted that though simply stopping at each urban intersection seems best, it is not always possible due to the large number of intersections in urban areas. The system used LiDAR sensors to determine lane visibility by generating a 3D point cloud. A noteworthy feature is the use of vector maps. A vector map contains traffic signs, lanes, and signals, which could assist the collision system in better understanding the scenario being observed.

The high-density point cloud maps were generated using laser scanners, RTK-GPS (Real-Time Kinetic), and fibre-optic gyroscopes [40]. These point cloud maps were used in conjunction with vector maps to create a vivid depiction of the scene. The authors [40] referred to this as a 6D localisation. When the sensors were not able to detect obstacles directly, it was assumed that nearby vehicles will obey the social and physical rules of the road, allowing for viable estimations to be made [40]. Such considerations and assumptions are crucial when designing effective automotive collision detection and prediction systems. The system [40] could be integrated with the one proposed in this work to enhance performance at occluded intersections. Figure 2.8 shows an example of a point cloud and a vector map.

2.4.2 Radar-based Traffic Tracking System

An older work from 1998 [41] presented a method of collision avoidance using a Kalman filter [42] [43] for tracking approaching vehicles. According to [42], Kalman filters are used to produce estimates of quantities that cannot be measured directly and can predict the future state of a system. These filters are vital for radar, where measurements suffer from reduced accuracy due to noise and other variables. The fixed track of cars travelling along a road greatly simplified tracking.

The Intersection Collision Avoidance (ICA) system proposed by [41] used the Kalman filter to track vehicles approaching the intersection based on the position and velocity measurements as well as the computed acceleration. The updated state vector is used to compute the time of arrival of the vehicle at the intersection. Additionally, the system used GIS (described in Section 2.1.1) to provide additional information about the road scenario, such as the angle of the adjoining roads. Here, the target arrival time is referred to as the 'gap' time. The radar is then used to determine whether this gap time is sufficient for the host vehicle to pass through or turn safely. Tracks are formed as 'candidate' tracks which are later promoted to actual tracks provided they meet certain requirements. A 'gate' is placed around the predicted positions. If a subsequent detection is within the gate, it is associated with the track. Most gates operate in more than one dimension: speed, position, and acceleration must all fit into a gate for the detection to be likely a target. Targets moving away from the host are excluded from tracking, as well as targets moving too fast or are too far away. Figure 2.9 presents the results of the research [41] and describes how these results are to be interpreted. The format of these results is likely representative of most radar-based vehicle tracking systems. There are substantial similarities between the research performed in [41] and the work proposed in this dissertation. The incorporation of a map database could prove to be a useful future addition to the work. Unfortunately, the radar parameters were not provided in the conference paper.

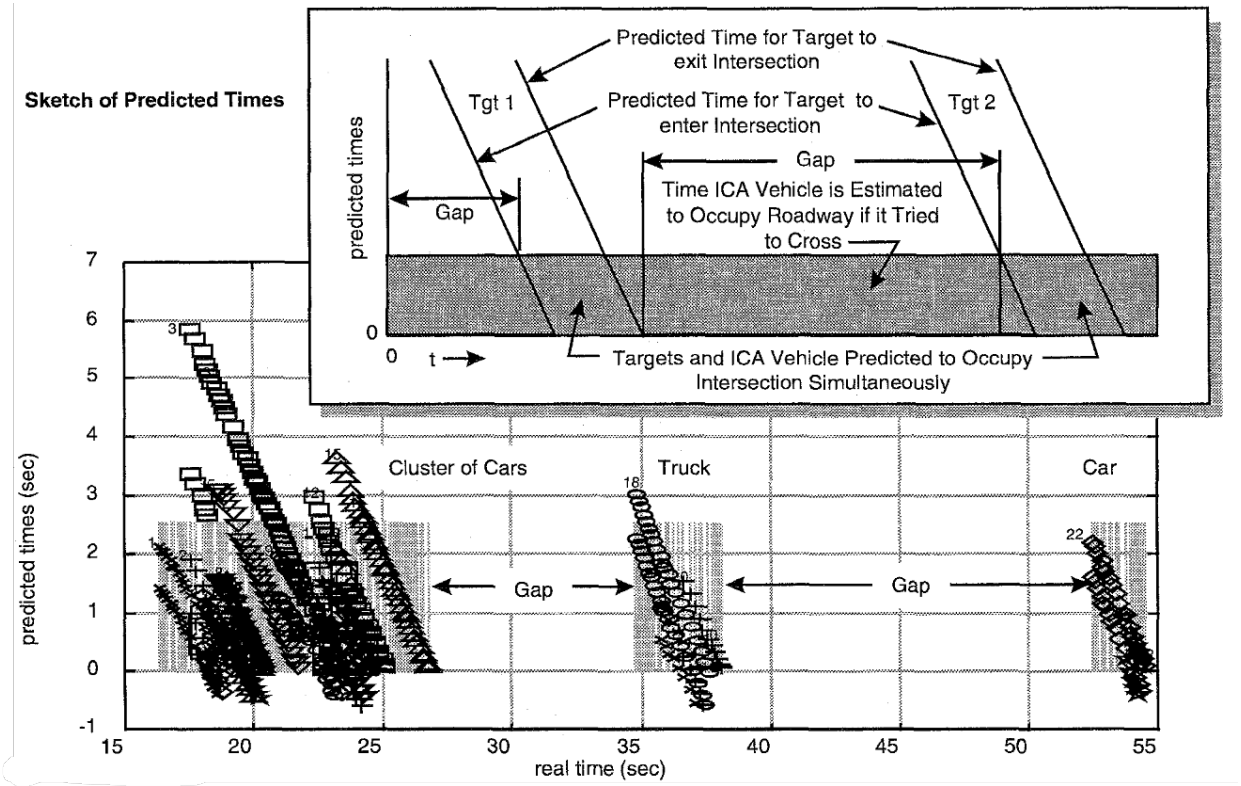


Figure 2.9: Results and description thereof for a radar tracking system for intersection collision avoidance [41]

2.5 Critical Review of Current Systems

Though V2I communication systems are generally more robust than the alternative methods for collision avoidance, these systems require infrastructure (RSUs and OBUs) at both the intersection and installed on all vehicles passing through. The need for secure, high-speed communication links between nearby vehicles and road infrastructure further increases the complexity of such systems. The roadside infrastructure will remain at risk of vandalism, and one cannot expect all vehicles to be fitted with compatible technology. Latency is an inherent downside of such a system, as data has to be transmitted, processed, and re-transmitted to the vehicles well in advance of any probable collisions. The research into communications architecture [34] indicates how vehicles could be prioritised based on their distance from the intersection.

Communication systems appear to be a brute-force solution to the problem of intersection collision prediction. V2V communication systems reduce the amount of infrastructure and communications required, though still share some of the V2I communication drawbacks such as communication latency and security. Additionally, the decryption and computations based on only relative kinematic information between two vehicles will have an associated latency. Though the use of onboard standalone sensors eliminates the latency and security issues and reduces cost, such systems require accurate sensing and processing of the scenario from a single, restricted point of observation. These systems have varying degrees of success. Most commercial systems make use of onboard sensing for driver assistance in primarily basic scenarios, such as lane-changing and blind spot detection. There is no reliance on other vehicles being fitted with similar technology. Ideally, a combination of all the above methods will provide robust mitigation of intersection collisions, and is, in all probability, the future of intersection collision avoidance.

The use of standalone sensor systems mounted on cars was deemed ineffective by some works due to the single point of observation, the probability of false alarm (PFA), the current state of the technology, and because these systems can only reduce collisions by the host driver. The advantages of a standalone system are the independence of surrounding infrastructure, low cost, and low maintenance. If these systems are proven to be effective, they will doubtlessly become the primary method of intersection collision avoidance.

2.6 Automotive Radar Systems

Automotive radar systems are those which use radar technology to identify possible road and driving hazards [5]. Considerations include the selection of a suitable carrier frequency [44], mount angle [45], the mitigation of interference [46], and analysis of any degradation due to vehicle movements and vibrations [47]. These considerations form the cornerstone of the design of any automotive radar system.

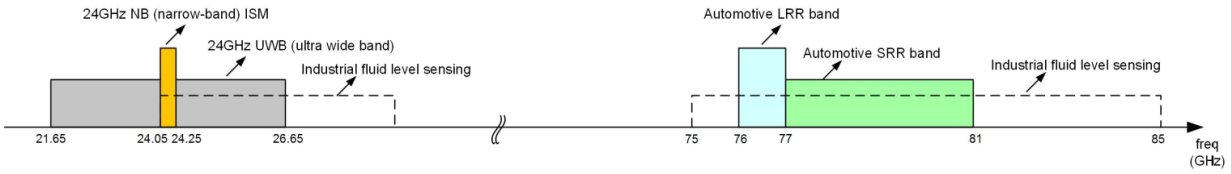


Figure 2.10: Automotive radar frequency bands [48]

2.6.1 24 GHz versus 77 GHz Radar

Texas Instruments, a company which produces various embedded radar systems, provided a paper discussing the move from traditional 24 GHz to 77 GHz radar for automotive systems [48]. The 24 GHz band is referred to as the industrial, scientific, and medical (ISM) band. The ISM narrowband (NB) frequency range is from 24.0 GHz to 24.25 GHz. A 5 GHz ultrawideband (UWB) is currently being phased out, leaving only the ISM narrowband available. The need for higher bandwidth makes the ISM band undesirable for use in modern radar systems. The 76-77 GHz band is suited to long-range applications as it is afforded a high equivalent isotropic radiated power (EIRP). The 77-81 GHz band is used for short-range radar, with the 4 GHz bandwidth allowing for higher range resolution compared to the 250 MHz ISM band. A graphical representation of these frequency bands is presented in Fig. 2.10. The higher 77 GHz carrier frequency allows for the use of approximately three times smaller radar sensors for a given beam width compared to a 24 GHz carrier frequency.

The 24 GHz radar is, however, still relevant. It can obtain a larger maximum range (for a comparative average transmitted power) and is suited to applications which do not require fine range resolution. One such example is the single-lane monitoring scenario. A study on radar target characterisation [44] revealed that the 24 GHz radar sensors were less sensitive to the orientation of the vehicular targets. This should aid in the detection of vehicles, though will not work well for the classification thereof. This is expected as the lower resolution (due to the narrower bandwidth) makes it difficult to identify target features. Though benefits exist for using 24 GHz radar, it is now classified as legacy technology. A system designer should aim to use the newer 77 GHz standard for automotive applications.

2.6.2 Radar Mount Angle Calibration

Research by [45] shows that radar performance can be improved by calibrating the mounting angle. Since the radar results are relative to the host vehicle, the orientation has a direct effect on measurement accuracy. All radar measurements are relative to boresight, i.e. the measured range and velocity are the radial components of the true values, so a change in orientation will change measurement results. The research [45] provided a method of automatically calibrating the mounting angle of automotive radar systems. Figure 2.11 illustrates the kinematic considerations for calibrating the radar mount angle.

The first step was to use an inertial measurement unit (IMU) to determine the radar's position relative to the vehicle's centre axis. The next step was to use triangulation by placing three targets at known relative positions. The method followed to obtain automatic mount angle calibration was to determine the mounting angle at which the known point target occupied the fewest grid cells in an occupancy grid map. This required iterating the mounting angle and selecting the angle at which the detection filled the fewest cells.

2.6.3 Mitigation of Interference

When using the ISM band, mitigation of interference is paramount. In 2018, research by [46] was undertaken to prove the effectiveness of a novel method for interference mitigation. The existing method of altering the operating frequency cannot guarantee that the new operating frequency will be free of interference. Existing techniques focus on reconstructing the original signal through knowledge of the interference. These methods are implemented post-detection. The work by [46] proposed a novel solution for mitigating interference prior to detection. It was achieved by separating the target echo and interference using morphological component analysis (MCA). The target beat frequency was considered sparse in the frequency domain while the interference was not. The sparsity was used by the MCA for the separation of the two components.

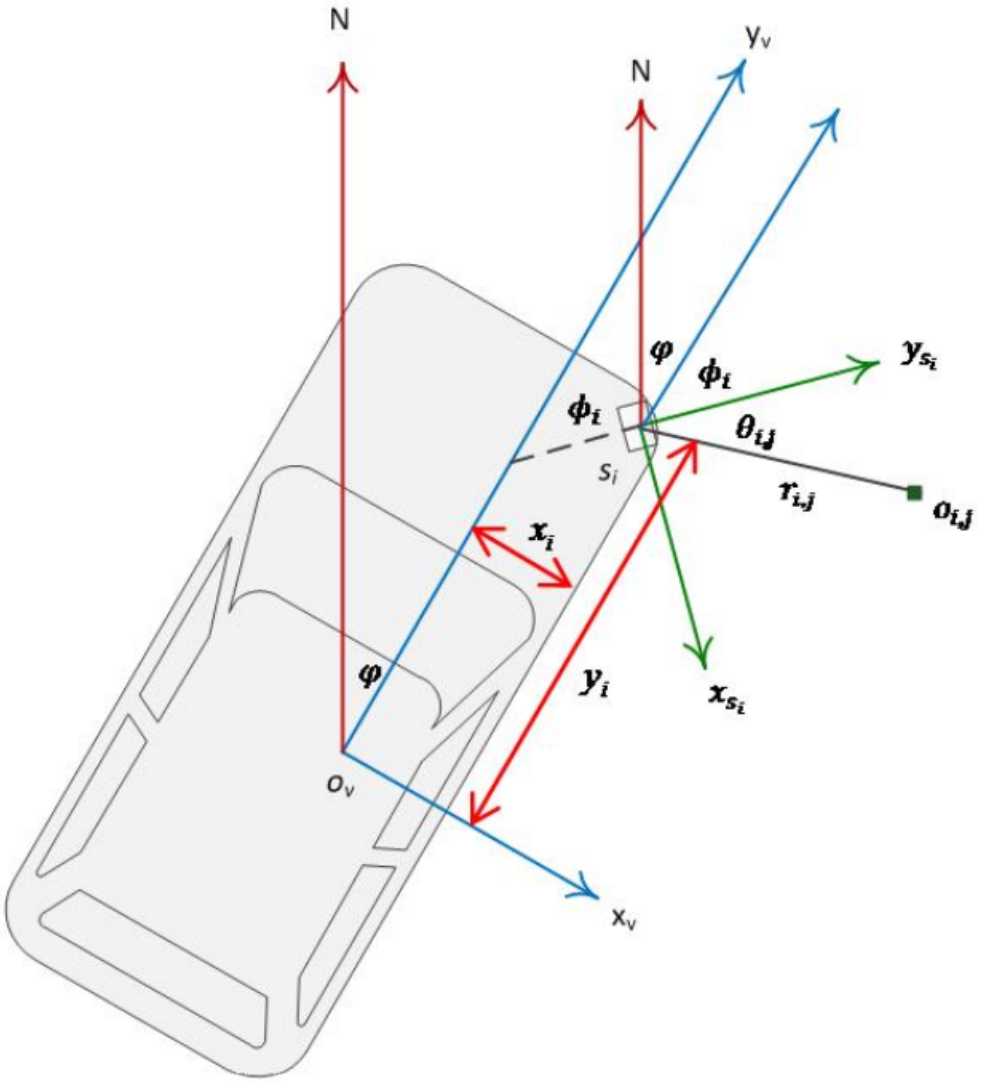


Figure 2.11: Vehicle Kinematics showing radar mount angle [45]

2.6.4 Degradation of Automotive Radar Performance

The effect of motion on radar performance is an important consideration when placing a radar on a motor vehicle. Such motion includes acceleration, braking, and vehicle vibration. Nonlinear movements refer to those which occur within one measurement cycle. These movements influence the PD and accuracy. Vibrations include those generated by the engine, drag, and road texture. Research conducted by [47] identified road texture as the leading cause of vehicle vibration. Several models describing nonlinear movements were

presented [47]. A new equation for noise level estimation was presented, which should prove helpful when designing automotive radar systems. In this dissertation, since the vehicle is stationary, nonlinear movements can be safely ignored.

2.7 Theoretical Framework

The theoretical framework lays the foundation for understanding the various technologies, algorithms, and concepts used in this project. The section begins with an introduction to Frequency-Modulated Continuous-Wave (FMCW) radar, along with the various modulation schemes thereof. The section on FMCW technology is a key concept for this project, as such waveforms allow for the measurement of both position and velocity, which are needed to estimate the time of arrival. The advantages and disadvantages of each type of waveform are presented.

Subsequently, signal processing methods are presented: namely signal windowing, the Fast Fourier Transform (FFT), and Constant False Alarm Rate CFAR detection. These algorithms facilitate the detection and tracking of vehicles. The selection of an optimal window function and CFAR algorithm, as well as the tuning thereof, are crucial for optimising the detection performance of the radars.

2.7.1 Frequency-Modulated Continuous-Wave Radar

Frequency-Modulated Continuous-Wave Radar (FMCW) is a favoured modulation scheme in the automotive industry as it allows for determining both range and velocity estimates [5]. It was not discovered by a single inventor: various corporations and government bodies contributed to its discovery. Applications include automotive radar, vital sign monitoring, concealed weapon detection and imaging radar. FMCW radars transmit a continuous wave (CW) which is periodically modulated. Modulation is typically linear: two common modulation schemes include sawtooth and triangle modulation. The linear change in frequency characterising an FMCW waveform can be described using Eq. 2.1, which represents the frequency sweep gradient in the time-frequency domain. The variable k represents the ratio of the signal bandwidth (BW) to the sweep duration (T_{sweep}). The range resolution [49]

given by Eq. 2.2 is common to all linear FMCW waveforms. Equation 2.3 is used for the conversion of Doppler shift to velocity.

$$k = \frac{BW}{T_{sweep}} \tag{2.1}$$

$$\Delta R = \frac{c}{2 \cdot BW} \tag{2.2}$$

$$v = \frac{\lambda \cdot f_d}{2} \tag{2.3}$$

A typical transmitted period has the time-domain form $s(t) = \cos(2\pi f_0 t + \pi k t^2)$ where k is the time-frequency slope (Eq. 2.1) and f_0 is the carrier frequency. This waveform is referred to as a chirp wave. The received signal is associated with a time delay based on the two-way propagation distance of the signal from the radar to the target and back again. A Doppler shift is imparted onto the signal if the target is moving. Figure 2.15 shows how these frequencies are derived. The time delay is mapped to an intermediate frequency (referred to as a 'beat frequency') coupled with the Doppler shift. An electronic mixer is used to extract the beat frequency. Radio Frequency (RF) mixing involves multiplying two signals together, producing signals corresponding to the sum and difference of the two frequencies. The beat frequency is obtained by low-pass filtering the mixer output so that only the difference between the two inputs is processed further. Figure 2.13 shows a typical example of the demodulation. Additionally, the input spectrum is brought to baseband (bandwidth centred on 0 Hz) since the carrier frequency is cancelled out. This allows for a minimum sampling rate of twice the maximum beat frequency to avoid aliasing. Figure 2.12 shows the time-frequency plot of the transmitted and received echo of a chirp wave. For sawtooth FMCW, the transmitted wave is periodic. The relationship between the time delay (between the transmission and reception of the chirp wave) and the beat frequency is shown.

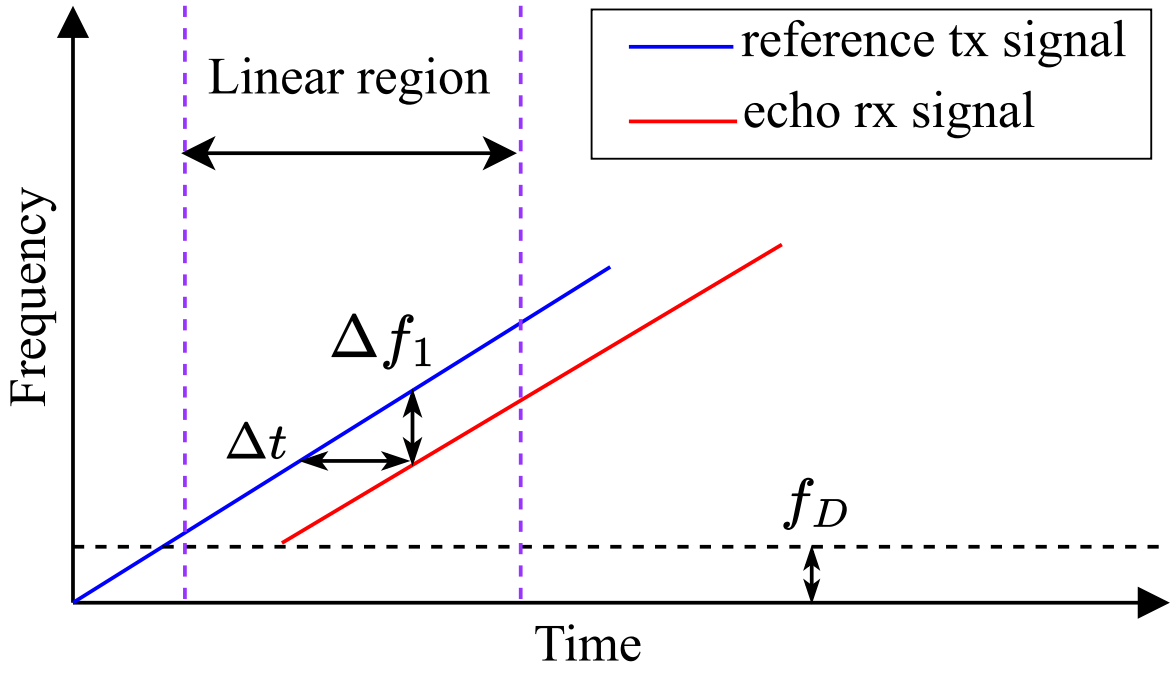


Figure 2.12: Sawtooth FMCW transmitted and received signal showing beat frequency and Doppler shift

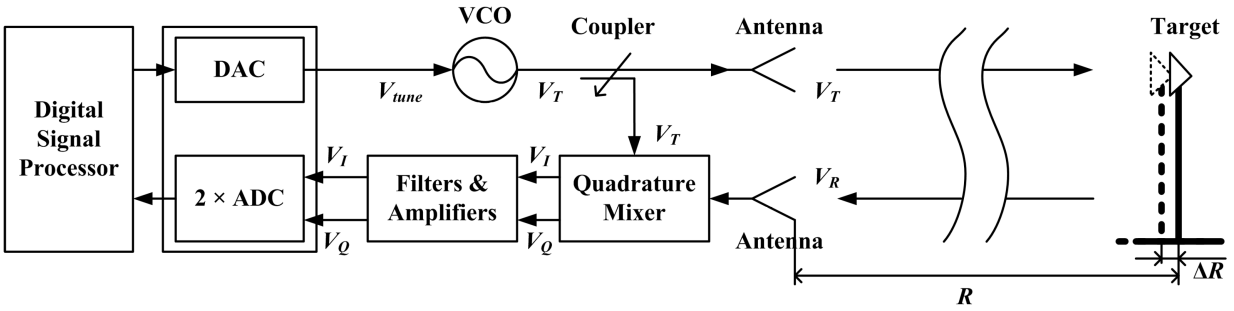


Figure 2.13: Example block diagram of a contemporary radar-on-chip (RoC) solution [50]

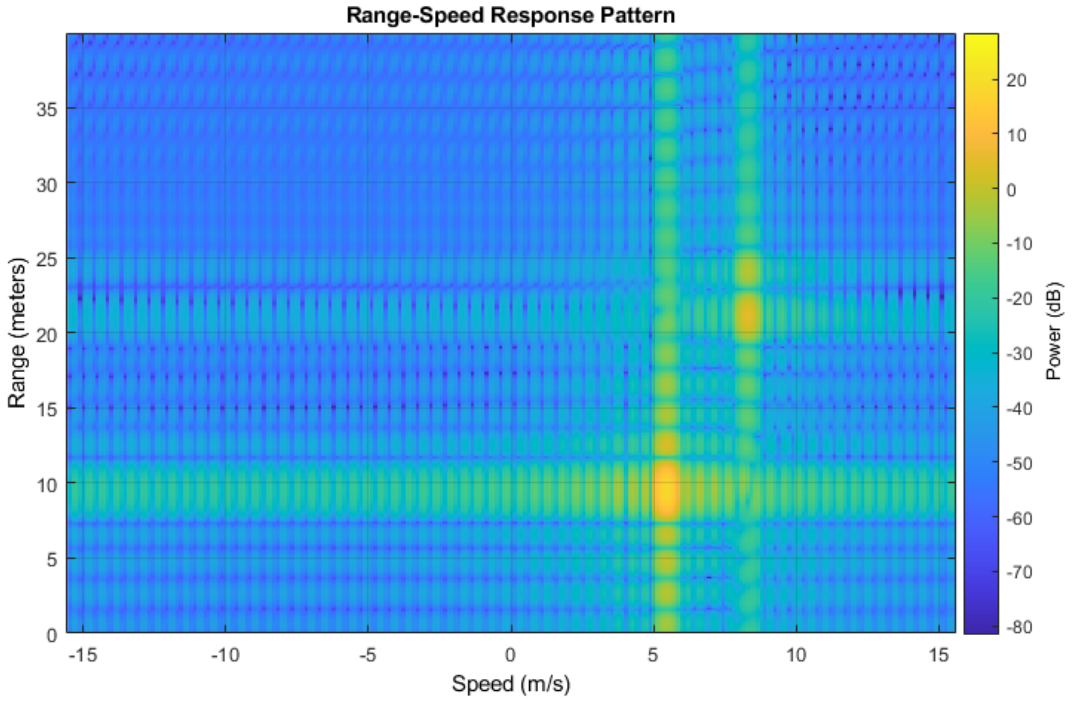


Figure 2.14: Simulated Range-Doppler map with two targets in the received frame

2.7.1.1 Sawtooth Modulation

Sawtooth modulation is common and is preferred due to the use of phase changes for estimating velocity. It is suited to scenarios in which a large detection range is required, with targets moving relatively slowly. A significant amount of processing is required. Firstly, multiple sweeps (periods) must be captured whilst the target occupies a single range bin. A fast-time FFT is then performed on each of these sweeps (rows), after which a slow-time FFT is performed along the frequency bins (columns) of the data matrix. The requirement of a frame of sweeps for target detection as opposed to relying on a single sweep poses a challenge for most entry-level radar systems. Large computations must be performed in short bursts to ensure a suitable output update rate.

Typically, the results of the 2D FFT are presented as a range-Doppler map. The fast and slow time axes are mapped to radial range and velocity, respectively. An example of such a map is presented in Fig. 2.14.

The benefit of sawtooth modulation is the ability to achieve a finer Doppler resolution

compared to triangle modulation (Section 2.7.1.2). The drawback compared to triangle modulation is a reduced maximum unambiguous detectable velocity and range-Doppler coupling (Eq. 2.8). These concepts are explored in this section.

The phase change $\Delta\phi$ of a received echo between consecutive chirps [51] is given by Eq. 2.4, where Δd is the distance traversed by the target and λ_0 is the center wavelength. The target's speed can then be calculated as shown in Eq. 2.5 where T_c is the time taken for the target to move Δd meters at a rate of v meters per second. Since the maximum unambiguous phase shift is $\pm\pi$, the maximum unambiguous velocity can be computed using Eq. 2.6. The slow-time FFT used to resolve the target's speed operates on the samples in the same range bin. By increasing the number of samples in each bin, a higher velocity resolution can be realised. Increasing the frame duration T_f (or coherent processing interval) can therefore improve velocity resolution [49] as shown in Eq. 2.7.

$$\Delta\phi = \frac{4\pi \cdot \Delta d}{\lambda_0} \tag{2.4}$$

$$v = \frac{\lambda_0 \cdot \Delta\phi}{4\pi \cdot T_c} \tag{2.5}$$

$$v_{max} = \frac{\lambda_0}{4 \cdot T_c} \tag{2.6}$$

$$\Delta v = \frac{1}{2 \cdot T_f} \tag{2.7}$$

Equation 2.6 shows that shorter chirps allow for a larger maximum velocity. Since the Doppler shift cannot be separated from the beat frequency, it is treated as a measurement error. The beat frequency after mixing and filtering a positive-ramp chirp is given by Eq. 2.8, where k is given in Eq. 2.1, f_D is the Doppler frequency shift, R is the radial range, c is the speed of light in the medium, and f_b^u is the beat frequency of the received up-chirp wave. The radial range can be measured using Eq. 2.9 which assumes f_D is negligible.

$$f_b^u = \frac{1}{2\pi}\frac{d\phi_{IF}}{dt} = \frac{2kR}{c} - f_D \tag{2.8}$$

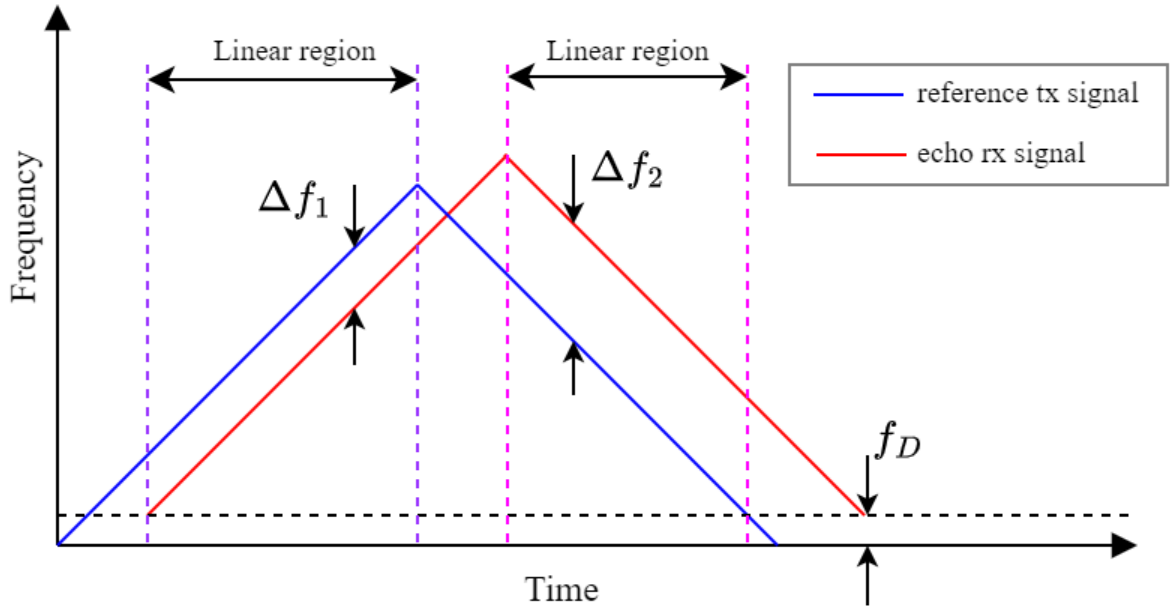


Figure 2.15: Single period of triangle FMCW transmit and echo signals where Δf_1 and Δf_2 are the up and down chirp beat frequencies, respectively

$$R = \frac{c \cdot f_b}{2 \cdot k} \tag{2.9}$$

2.7.1.2 Triangle Modulation

The range-Doppler coupling intrinsic to sawtooth modulation is resolved by triangle modulation. Both Doppler and range estimations can be obtained after two sweeps: an up-chirp followed by a down-chirp. Far less memory and processing is required (compared to sawtooth modulation) since a frame of sweeps is not required and simpler processing methods can be applied. The disadvantage, when compared to sawtooth modulation, is ambiguity in multi-target scenarios and the possibility for 'ghost targets'. The fundamental equations of triangle FMCW are given by Equations 2.10 and 2.11, which is based on Eq. 2.8.

$$f_b^u = \frac{2kR}{c} - f_D \tag{2.10}$$

$$f_b^d = \frac{2kR}{c} + f_D \tag{2.11}$$

$$f_b = \frac{f_b^d - f_b^u}{2} \tag{2.12}$$

$$f_D = \frac{(f_b^d - f_b^u)}{2} \tag{2.13}$$

$$v_{max} = \frac{(f_b^d - f_b^u)_{max} \cdot \lambda}{4} \tag{2.14}$$

Equations 2.10 and 2.11 show how the up-chirp beat frequency (f_b^u) and down-chirp beat frequency (f_b^d) are related to the radial range (R) and Doppler shift (f_D). Subtracting Eq. 2.10 from Eq. 2.11 equates to $2 \cdot f_D$ while addition equates to $2 \cdot \frac{2kR}{c}$. Equations 2.15 and 2.16 show how R and f_D are calculated when using the triangle FMCW waveform. Substituting f_D from Eq. 2.16 into Eq. 2.3 produces a radial velocity estimate. The triangle modulation uses the change in the target's beat frequency between the up and down chirps to determine its velocity and range, respectively. In contrast, the sawtooth modulation uses the change in phase between the target echo and reference signal over a frame of sweeps to determine velocity. Assuming the received echo is sampled at twice the beat frequency (Eq. 2.17), the frequency bins are spaced $\Delta f = \frac{f_s}{N}$ apart, where N is the number of samples per sweep. Velocity resolution can be represented by substituting frequency resolution Δf into Eq. 2.3. Since the frequency resolution Δf is proportional to the velocity resolution, increasing N can be used to better resolve targets separated by velocity. Accuracy can be improved by increasing the length of the time domain signal before performing the frequency domain transformation. A simple method of increasing the length of the time domain sequence is zero-padding, discussed in Section 2.7.5.

$$R = \frac{c(f_b^u + f_b^d)}{4k} \tag{2.15}$$

$$f_D = \frac{f_{beat}^{up} + f^{down}}{2} \tag{2.16}$$

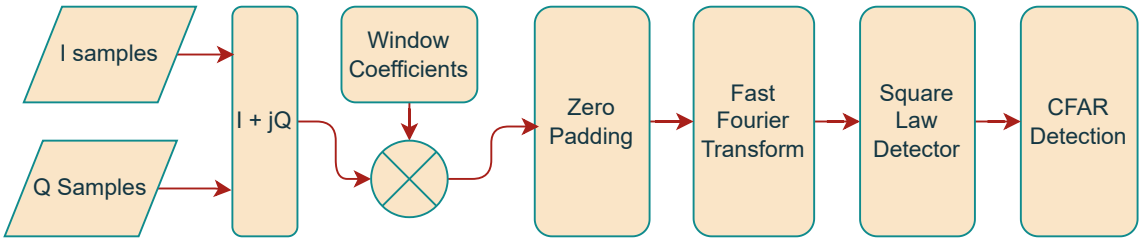


Figure 2.16: Initial radar signal processing stages

2.7.2 Radar Signal Processing

Digital radar signal processing is the process of extracting information from a sampled received signal using some form of a processing system. It refers to specific algorithms and techniques used to optimise processing and results. Several processing techniques which constituted the custom processing algorithm implemented in this work are discussed. These algorithms were optimised for the uncontrolled, single-lane intersection monitoring scenario. Figure 2.16 shows the initial processing stages presented in this section. Concepts explored in this section were obtained from [52] and.

After the target echo has excited the radar antenna, it is amplified, filtered, and sampled. A low-pass finite impulse response (FIR) filter, referred to as an anti-aliasing filter in this application, is used to remove unwanted signals from the received signal that would otherwise produce unwanted artefacts post-downsampling [52]. An FIR filter is used since it is inherently stable and has a linear phase characteristic, which is favourable for coherent radar applications. The downside when compared to an infinite impulse response (IIR) filter is that there is a longer delay when data is moved through the FIR filter. This is due to more delay lines (taps) required to provide an equivalent response, since the FIR filter uses only previous inputs, whereas the IIR filter output is based on both previous inputs and outputs.

Sampling is carried out by an analogue-to-digital converter (ADC), which samples signal echoes at a rate corresponding to the Nyquist criterion given by Eq. 2.17, where f_s is the sampling rate and f is the maximum frequency that can be perfectly reconstructed from the sampled signal. Bandpass sampling requires a sampling rate commensurate with the signal bandwidth. Though this appears to violate the Nyquist criterion, the use of both I and Q

data means that two samples effectively represent each time domain sample. The selection of a suitable ADC is determined by many factors, one of which is the level of quantisation noise.

$$f_s > 2\,f \tag{2.17}$$

2.7.3 Signal Windowing

Since the Fourier transform assumes a signal is continuous for all time, the sudden transitions at the start and end points of a received finite-duration signal result in spectral leakage, in the frequency domain. Information presented in this section was sourced from the Principles of Modern Radar [52] textbook. Windowing involves tapering of the received signal to reduce the power level of these frequency domain side-lobes [52]. After applying a window function, the ratio of the mainlobe to the sidelobe power is increased which improves target detection probability for a given PFA. The increase in prominence of the mainlobe comes at the cost of a broader main lobe, which translates to a reduced frequency resolution. Common window functions include the Hamming, Hanning, Blackman, and Taylor windows. A comparison of a subset of window functions is presented in Fig. 4.5. Windowing can reduce the straddling loss, where the true frequency lies between two frequency bins, associated with a discrete frequency spectrum [52].

2.7.4 Fast Fourier Transform

The Fourier Transform [52] is an algorithm used to convert a time domain signal into its corresponding frequency domain representation. This allows for the frequency or spectral content to be analysed and used in further processing. The Discrete Fourier Transform (DFT) is used when the input time domain signal has been digitised and consequently is finite in length. It allows for the computation of the Fourier Transform by a computer on sampled data. The Fourier Transform is a key algorithm in this project, as it allows for the extraction of the beat frequencies generated by the FMCW waveform described in Section 2.7.1. In radar, the Fourier Transform allows for the detection of targets and the

measurement of range, velocity, and angle of arrival.

An FFT refers to any efficient algorithm which implements the DFT using computer hardware. Inputs that are powers of two offer significant performance improvements when using an FFT algorithm. The required number of complex multiplications and additions is $\frac{N}{2} log_2(N)$ and $N log_2(N)$, respectively, when N, the input sequence length, is a power of two. The FFTW (Fastest Fourier Transform in the West) algorithm is a C subroutine library which offers superior performance compared to other publicly available algorithms. This algorithm is used in the MATLAB `fft()` function but not in the NumPy and SciPy Python libraries.

2.7.5 Zero Padding

Time-domain zero padding is a form of frequency-domain interpolation. Adding trailing zeros to a set of time domain samples is defined as zero padding. The primary use is to obtain a power-of-two sample length for improved FFT performance. Additionally, finer frequency accuracy is realised since zero-padding results in finer spacing between frequency bins, allowing for inter-bin frequencies to be better estimated [53] assuming that only a single frequency lies between the bins. Essentially, zero padding interpolates the frequency domain signal, smoothing out the curve fitted to the FFT points. The relationship between frequency bin width and the number of FFT points is given by Eq. 2.18. Note that zero padding does not improve frequency resolution: since no new information is added, multiple frequencies occurring between frequency bins cannot be resolved using zero padding. If it is reasonably certain that only one target is present, the accuracy of the frequency estimation can be significantly improved via interpolation using zero padding. The process of zero-padding is a key aspect of this project since a low-cost radar such as the uRAD USB v1.2 generally offers reduced accuracy compared to higher-priced alternatives.

$$\Delta f = \frac{f_s}{N} \tag{2.18}$$

Similarly, there exist dedicated spectrum interpolation algorithms such as Gaussian and Parabolic interpolations that offer an increased inter-sample resolution. However, zero

padding is the simplest to implement, provides interpolation via the FFT algorithm without requiring an additional algorithm, and is consequently less computationally expensive than the alternative methods.

2.7.6 Constant False Alarm Rate Detection

The need for constant false alarm rate (CFAR) detectors arose from the variability of the noise floor and clutter background in many radar applications [52]. A detailed explanation of CFAR processing is given in Principles of Modern Radar textbook [52]. When using FMCW radar for observing vehicles, CFAR allows for the extraction of the FMCW beat frequencies based on the location of signal amplitudes breaching the adaptive CFAR threshold. The beat frequencies are then used to produce range and velocity estimates. The CFAR detector can detect targets with low SNR by forming a threshold based on the surrounding cells and desired PFA.

A constant threshold produces detections when the received echo exceeds a predetermined value, which is effective when strong target echoes are received against a roughly constant noise and clutter background. The specific value of PFA is user-defined and is typically selected based on the interference power (which may vary spatially and temporally), processing methods implemented post-CFAR, and the specific radar application. The CFAR detector produces detections based on the relative level of one cell, the cell under test (CUT), to those surrounding it. The key CFAR detection equation is given by Eq. 2.19, where the detection threshold D_T is the product of a threshold factor α and an estimate of the noise power P_n^2. The downside of CFAR detection is a reduced probability of detection (PD) compared to a fixed threshold, such as the Neyman-Pearson detector [52]. A larger input SNR is therefore required for a detection to be made.

$$D_T = \alpha \cdot P_n^2 \tag{2.19}$$

Various algorithms exist for estimating P_n^2. The basic cell averaging CFAR (CA-CFAR) uses the average of the cells surrounding each CUT and scales this average by a constant. The constant, α, is calculated based on the desired PFA and the number of training cells

(N_{train}) as shown in Eq. 2.20. Improvements on the generic CA-CFAR algorithm include Smallest-Of (SOCA) and Greatest-Of (GOCA) cell averaging CFAR [52]. These compute the averages on either side of the CUT (the training sets) and use the smaller or larger average, respectively. GOCA CFAR is used to minimise false alarms due to clutter edges, whereas SOCA CFAR is used to suppress the masking of mutual targets [52]. Mutual target masking refers to the presence of one target in the training biasing the noise power estimation [52]. A more sophisticated method of avoiding mutual target masking is the Ordered Statistic (OS) CFAR algorithm.

$$\alpha = N_{train}(PFA^{-\frac{1}{N_{train}}} - 1) \tag{2.20}$$

The ordered statistic CFAR implements a sort operation instead of the averaging operation used by the traditional CA-CFAR variants. According to [54], OS-CFAR is useful in multi-target automotive applications. The OS-CFAR algorithm was first presented by Rohling [55] in 1983. The drawback is the significantly higher computational cost due to the sort operation. Equation 2.21 can be used to calculate the scaling factor α based on the desired PFA, N_{train}, and rank r. The algorithm is capable of rejecting $N_{train} - r$ targets in each training set. Choosing $r > \frac{N_{train}}{2}$ allows for suppression of clutter edge false alarms [52]. Figure 2.17 provides the general CFAR architecture, where the method for estimating clutter power is defined by each algorithm variation, such as CA-, GOCA-, SOCA-, and OS-CFAR. Though only the four fundamental CFAR detection algorithms are discussed, there exist many variations offering varying degrees of success.

$$PFA = r \begin{pmatrix} N_{train} \\ \\ r \end{pmatrix} \frac{(r-1)!(\alpha + N_{train} - r)}{(\alpha + N_{train})!} \tag{2.21}$$

The use of guard cells improves the detection capability of CFAR detectors. Guard cells define cells surrounding the CUT which are excluded from the averaging or sort operation. These cells are often part of the target echo and should be excluded to avoid biasing the

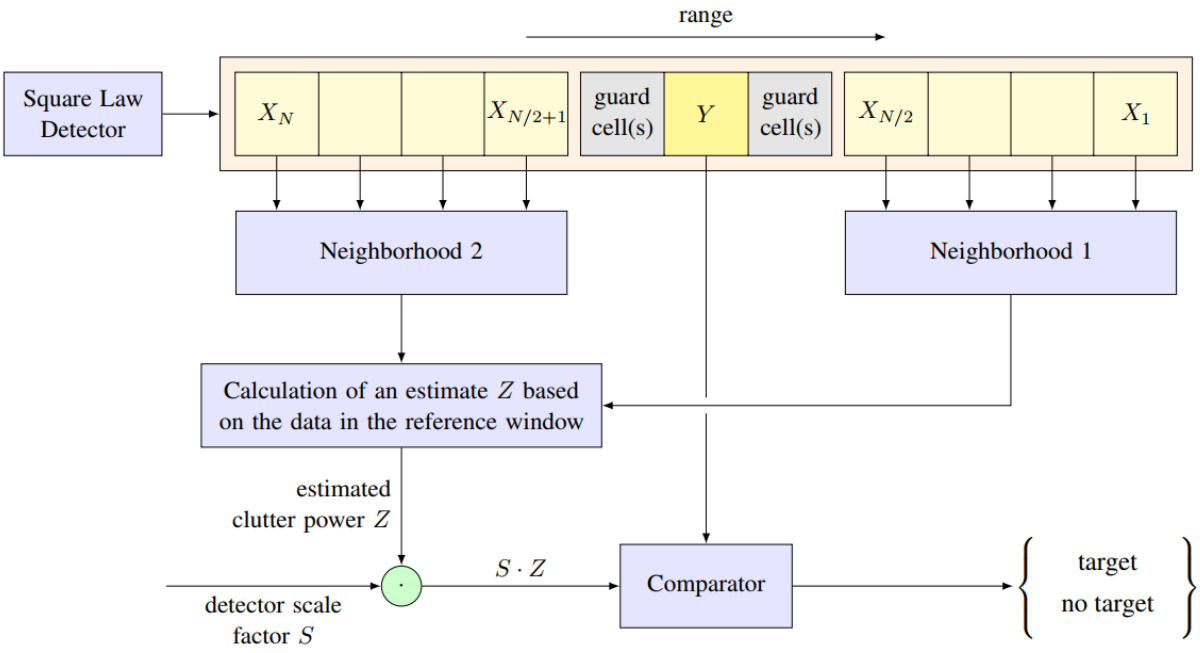


Figure 2.17: Generic one-dimensional CFAR block diagram [56]

noise power estimate. The number of guard cells depends on factors such as the target being observed and the frequency resolution (especially after time-domain windowing). The CFAR algorithms are designed to operate on the output of a square law detector, which is the sum of the squares of the I and Q channels. Fewer computations are required compared to the linear detector since the square-root operation is excluded while performance is maintained [57].

2.8 Summary

The need for ACAS systems is clear: based on the presented collision statistics and ACAS technologies, ACAS systems are effective at reducing road accidents. The road intersection, especially one which is not regulated by traffic lights, was identified as a common location of road accidents. Research into predicting collisions at these locations will significantly reduce the overall number of future road accidents.

Though similar systems to the proposed system are provided as optional extras by several automotive manufacturers, minimal publicly available research into the effectiveness of these

systems—particularly low-cost solutions—exists. Such systems require less infrastructure and dependence on external technologies compared to V2I and V2V communication systems. However, V2I appear to offer better performance than onboard systems as such systems are not restricted to a single vantage point and by the number of sensors and other hardware that can be installed at road intersections. Provided the latency of communicating information is low, V2V and V2I systems can offer effective prediction and mitigation of unregulated intersection collisions. Despite the apparent superiority of the communications systems to onboard systems, the benefit of reduced cost and reduced dependence on infrastructure motivates research into such technologies.

The ability of radar to measure both speed and position of targets makes it a suitable technology for onboard collision prediction. Many considerations are required, such as the radar carrier frequency, mount angle, interference mitigation, and the effects of mounting a radar to a vehicle. The FMCW radar waveform is required for measuring both the range and velocity of a target. The design of the waveform includes factors such as the modulation scheme, ramp bandwidth, and period, which determine the radar metrics. Notable metrics include the maximum detectable unambiguous range and velocity and the range and velocity resolutions.

To correctly estimate the range and velocity of a target, radar signal processing is required. To this end, various signal processing methods are used. Examples include signal windowing, the FFT, and CFAR algorithms. By tuning the parameters of these algorithms, the values of PFA and PD can be improved. Improving these metrics will improve the trustworthiness of the range of velocity estimates produced by the system.

Chapter 3

Requirements and Methodology

The purpose of this section is to describe the research process carried out in this project. The process was divided into five phases. Firstly, an investigation was carried out to assess the current state of the technology. Following this, design tools and requirements were identified. The design and simulation phases were carried out iteratively, followed by the implementation, testing, and performance analysis of the final test rig.

An overview of the approach is presented first, followed by descriptions of the methods used in each project phase. Figure 3.1 shows the five project phases. Additionally, methods for data collection, analysis, and interpretation are presented.

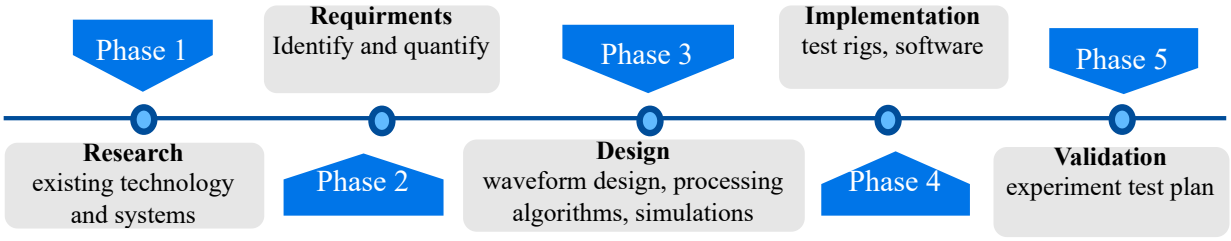


Figure 3.1: Overview of project phases

3.1 Overview of Approach

The first step was to determine if such a system existed commercially or had been previously researched. Research into automotive collision avoidance was undertaken to gain an understanding of the functionality, strengths, and weaknesses of such systems. Suitable hardware and software tools for the proposed system were then identified. Since the design was primarily software-based, the Agile methodology was adopted [58]. The benefits of this methodology include flexibility, speed, and continuous improvement as new information is discovered [58]. Processing algorithms were continuously refined and alternate avenues were pursued before an appropriate solution was found. Once the requirements outlined in Section 3.3 were met, a roadside test rig was designed and built. The test rig was used for controlled and uncontrolled experimentation to evaluate system performance, accuracy, and whether or not the system met the user requirements described in Section 3.3.

3.2 Phase 1: Research and Investigation

To gain an understanding of the capability of automotive collision avoidance systems, research was performed and collated in Chapter 2. Index terms were used to search for any relevant research and commercial systems. Once it was determined that a research gap existed, the research objectives stated in Section 1.4 were devised. These objectives were summarised as follows:

1. Identify and define the user and technical requirements
2. Identify suitable hardware that meets the technical and user requirements
3. Simulate radar hardware and the designed data and signal processing for the intersection scenario
4. Implement processing algorithms and test performance using a suitable test rig
5. Compare results to simulations, Global Positioning System (GPS) data, video recordings, and the project requirements

3.3 Phase 2: Identification of Requirements

The project requirements focused on the required performance of the radar hardware and processing algorithms. The hardware needed to produce results at a rate faster than the average driver reaction time, which is in the order of hundreds of milliseconds [59]. Similarly, the accuracy had to be greater than that of a human estimate [60]. The system needed to have an appropriate size and weight to be installed in the front bumper of a vehicle. The false alarm rate, robustness, and output format influence how the system is perceived by the driver. These high-level requirements are summarised in Tab. 3.1. The 'user' refers to the driver of the vehicle fitted with the proposed system. For UR5 in Tab. 3.1, the scope limited testing to two variations in vehicle type (hatchback and compact sedan) with no consideration of weather conditions.

Table 3.1: User Requirements

ID	Requirement	Description
UR1	Update rate	Faster than the rate of driver decision making in the scenario
UR2	Output format	Intuitive and easy to interpret
UR3	Accuracy	Better than human estimation in the scenario
UR4	False alarm rate	Low enough for the driver to trust the system
UR5	Robustness	Reasonable tolerance to scenario variations

The technical requirements were based on the characteristics of the unregulated intersection monitoring scenario. Characteristics include the maximum expected speed of vehicles, the observation of a single lane in each direction, and the minimum required resolution and accuracy. The maximum required range was determined to be proportional to both the maximum expected velocity of approaching vehicles and the host vehicle's time-to-turn (T2T) using the relationship between time, distance, and speed. A longer T2T and/or faster-approaching vehicles require detection at a further range. Investigation into the average T2T was out of scope. Similarly, the update rate was based on the assumed maximum driver decision-making rate of three estimates per second. These technical requirements are

summarised in Tab. 3.2. The maximum range of 60 m is based on a T2T of 3 seconds and a maximum expected velocity of 70 km/h. The minimum accuracy needed to be equivalent to that of a GPS measurement, or at least greater than that of a human estimate.

Table 3.2: System Technical Requirements

ID	Requirement	Value	Description
TR1	range	0 - 60 m	Based on the T2T of the host vehicle and maximum expected speed of approaching vehicles
TR2	velocity	0 - 70 km/h	Based on the main road speed limit and the T2T of the host vehicle
TR3	range resolution	<4.3 m	Less than the length of a small car
TR4	angular resolution	N/A	Single lane is monitored
TR5	velocity resolution	N/A	Single target is monitored
TR6	range accuracy	$<\pm 1.4$ m	At least as accurate as a GPS measurement
TR7	velocity accuracy	$<\pm 5$ km/h	At least as accurate as a GPS measurement
TR8	hardware output	I/Q samples	To facilitate development of a signal processing algorithm
TR9	update rate	>3 ests/sec	Assumed rate of driver decision-making in estimates per second

3.4 Phase 3: Design and Simulations

Simulations allowed for measuring the performance of the radar hardware in a variety of scenarios. The designed processing algorithm was used to process the simulated data, the

output of which was compared to the ground truth values. The process of designing the processing algorithm and simulation thereof was iterative. MATLAB and various MATLAB toolboxes were used for both simulation and algorithm design. Simulation results were compared to the technical requirements as specified in Tab. 3.2. Three FMCW modulation schemes were investigated: sawtooth, triangle, and dual-rate triangle modulation. Experiments were performed by altering the bandwidth and ramp duration parameters.

Simulation of the scenario involved modelling the road and car trajectories at road intersections. Firstly, one target and one radar were simulated. Following this, scenarios using multiple targets with various motion parameters were tested for the intersection scenario with the dual radar system. Processing algorithms were implemented for each of the three modulation schemes. These were compared to determine which offered the best performance for the given application. Computer-Aided Design (CAD) software was used to design the final test rig. The placement of components, structural integrity and ventilation aspects were considered. Additionally, suitable peripheral hardware was identified.

3.5 Phase 4: Implementation

Various steps were required to adapt the processing algorithm to run in real time. Once a suitable script was designed, it was implemented and tested on an embedded platform. Multi-threaded configurations were implemented to improve performance. Additionally, a real-time plotting script was implemented to allow for viewing processed radar results in real-time, at the cost of reduced performance. This was used for debugging purposes only as the proposed system needed only to report on turn safety.

Three test rigs were designed. The first used a single radar and camera while the second and third used two cameras and two radars. The test rigs allowed for verifying the processing algorithm performance during the development cycle. The dual camera system was implemented for verification of both real-time and offline processing.

The final test rig was designed as a standalone enclosure to be placed on the pavement next to the main road. It was decided that fitting the system to a car was unnecessary for the following reasons:

1. A car-installed test rig would require parking at the intersection for data collection, obstructing other drivers
2. Fitting the system to a car requires additional work, which would increase the project duration
3. The system would have to be designed to fit a single vehicle make and model, or a subset of vehicles with a similar design
4. If the installation is not sound, there will be a risk of damage to the hardware
5. Connecting and supplying power to the hardware will require additional consideration based on the car battery output
6. Since the goal was to provide proof of the concept, a product-level prototype was not required

3.6 Phase 5: Testing and Verification

The final project phase involved assessing system performance and performing controlled and uncontrolled testing. The hardware was tested to verify that the technical requirements outlined in Tab. 3.2 were met. Initially, testing was performed in a laboratory, after which the system was deployed at the side of a road. The initial laboratory experiments used office chairs and a trolley fitted with a corner reflector to replicate cars in the small-scale test environment. Next, a car was driven past the radar at a known speed with GPS data used to verify the radar estimates. Finally, the system was deployed alongside a main road. These experiments were performed for both the single and dual radar test rigs. The experiments were consolidated into an experiment test plan, summarised in Tab. 7.1. Further analysis was performed on the road tests as the laboratory tests were used for iterative system development.

Captured radar data was uploaded to the cloud for both safe storage and access by multiple devices. Data collection took place at several road locations, with controlled experiments performed on residential roads and uncontrolled experiments on main roads. Collected data

included low-resolution video footage, cellular GPS data, radar I/Q samples, and the speed, range, and time of arrival estimates produced by the real-time processing algorithm. The three data sources facilitated both qualitative (video) and quantitative (GPS data) comparisons with the radar estimates. GPS was determined to be the only other technology that could produce equivalent estimates in the proposed scenario. The speedometer measurements were used to maintain the desired target speed but were not used as ground truths since speedometers generally over-report speed measurements [61] and cannot be measured precisely by eye.

3.7 Summary

This chapter presented the methods followed during the five project development stages, where tasks were completed using the agile methodology. The iterative nature of the project was apparent in the simulation and design phases as well as the test rig development and testing stages. The structured approach, software tools, and hardware availability allowed for rapid development and testing.

Substantial research was performed on the topic of automotive collision avoidance systems and FMCW theory. The research process continued through the simulation, design, and implementation processes, ranging from signal processing theory to information on CAD modelling techniques. MATLAB simulations and processing sub-algorithms allowed for the rapid development of the final processing algorithm. The final roadside unit test rig proved to be an exceptional prototype for assessing the performance of the proposed system.

Chapter 4

Design

This chapter provides a structured and detailed description of the major project design stages. The system design is presented first, including a discussion on radar hardware selection and outlining the tools and methods for configuring the demonstration system. Sections on the design of the signal processing chain are presented, including topics on waveform design and descriptions of the various processing stages. Finally, the design of the roadside test rig is presented, which involved CAD modelling and considerations on the optimal layout of the various subsystems.

4.1 System Design

The system design section provides a high-level description of the demonstration system. Various possible radar hardware solutions are identified and discussed. Following this, the various software tools used during system development are presented. Interfacing methods for connecting the various subsystems are then described. Figure 4.1 provides a high-level representation of the implemented prototype, showing the dual radar and dual camera systems connected to a central processing unit. The processing unit routed power from the power supply to the various sensors. Each block required a hardware selection process as described in Section 4.4.2. All interfaces were over USB, with the result streamed using the Secure Shell (SSH) protocol and written to text files for offline processing.

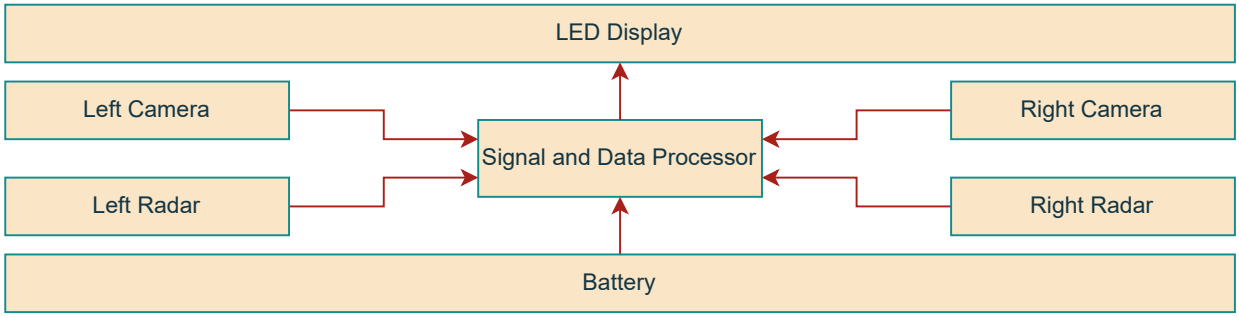


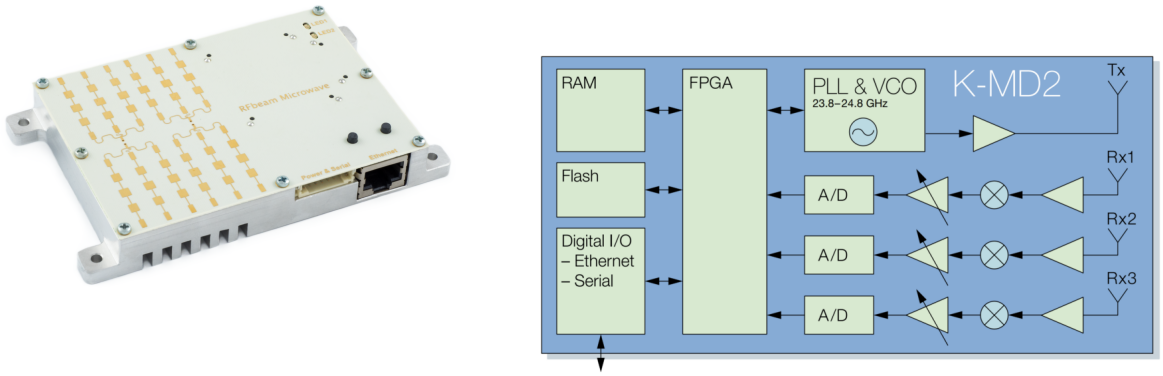
Figure 4.1: High-level system block diagram

4.1.1 Radar Hardware Selection

An investigation into suitable radar hardware was carried out based on the requirements identified in Tables 3.1 and 3.2. Since automotive radar typically operates in the GHz region, the size and weight of the radar antennas are expected to meet the respective requirements. RFbeam Microwave GMBH [62] offered hardware options which were well-suited to the intersection collision avoidance scenario. The suitability was confirmed by inspection of the applications list of each radar system. Applications included intersection surveillance, anti-collision, speed measurement, and counting and classification of vehicles. Some of the hardware options below were identified by narrowing the search of the RFbeam Microwave products to intersection surveillance. It was discovered that RFBeam defines intersection surveillance as an infrastructure-based system. The radar is expected to be installed at intersections and performs traffic flow analysis, queue detection, and intelligent traffic signal control [62].

The K-MD2 radar transceiver [63] is a powerful, high-end FMCW radar system. It offers a maximum detection range of 200 meters for cars, a maximum detectable velocity of 130 km/h with a 1 km/h resolution, and three receiver channels for 3-D position estimation. The system offers integrated firmware processing to aid with fast development since only data processing, as opposed to both signal and data processing, would be required. The downsides were that it was vastly more expensive than the alternatives and was possibly superfluous for the proposed application. Since two radars were required for the project, the cost of the K-MD2 radar transceiver was outside the project budget. The inflexibility of the firmware-based processing reduced the scope for optimising processing for certain

scenarios. Figure 4.1.1 shows the radar module and block diagram.

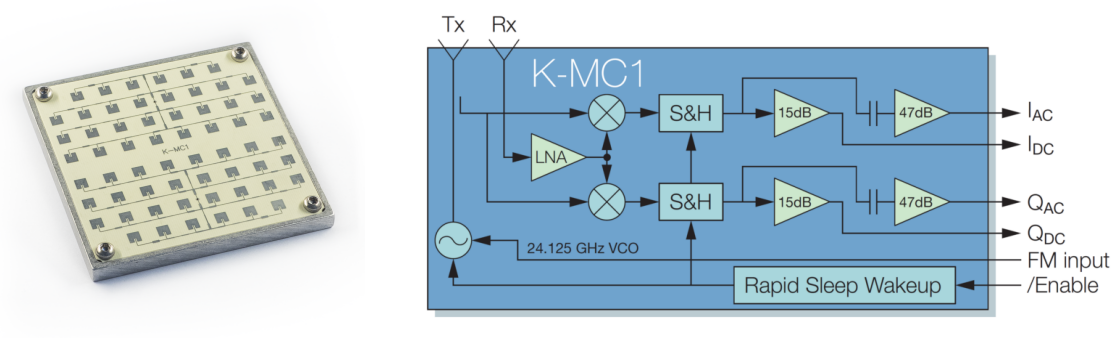


(a) Photograph (b) Block Diagram

Figure 4.2: K-MD2 radar and block diagram

The K-MC1 [64] radar transceiver was another potential solution. Radar parameters such as maximum range and velocity were not presented due to dependence on waveform design by the user. This radar operated at 24 GHz with a bandwidth of 300 MHz. The stated maximum range for the detection of vehicles was 150 m. Though the beamwidth in azimuth of 12° seemed to be too narrow, the radar could be rotated such that the specified elevation beamwidth of 25° could be used in the azimuth plane. The relative sidelobe gain of -40 dB in elevation appeared suboptimal. Unfortunately, this module required the design or integration of additional circuitry to allow it to interface with other hardware. Essentially, it is not a fully-formed, off-the-shelf system. The design of low-level circuitry and embedded software was beyond the scope of this project. Figure 4.1.1 shows the radar module and block diagram.

The uRAD USB v1.2 by Anteral [65] was selected due to its low cost (3497 ZAR excluding shipping, duties, and tax), availability, and ease of programming and deployment. Based on the datasheet, the system appeared to meet the minimum application requirements, making it well-suited to this project. The Python Software Development Kit (SDK) simplified development, as opposed to the C-based SDK offered by most of the alternatives.



(a) Photograph (b) Block Diagram

Figure 4.3: K-MC1 radar and block diagram

For a proof-of-concept, the benefit of a C-based implementation, namely increased performance, remained out of scope. Though uRAD offered an automotive radar solution, this was not used for several reasons. Since the automotive uRAD was capable of 3-D positioning, it was slightly over-specified for the single-lane application. Raw I/Q data could not be produced by the module, which was a project requirement (TR8, Tab. 3.2). Finally, the cost of 4393 ZAR (excluding shipping, duties, and tax) led to the selection of the cheaper uRAD USB v1.2 as it was hypothesised that the lower-specification hardware would work for the project application. Figure 4.4 shows the front, back, and side views of the uRAD radar.

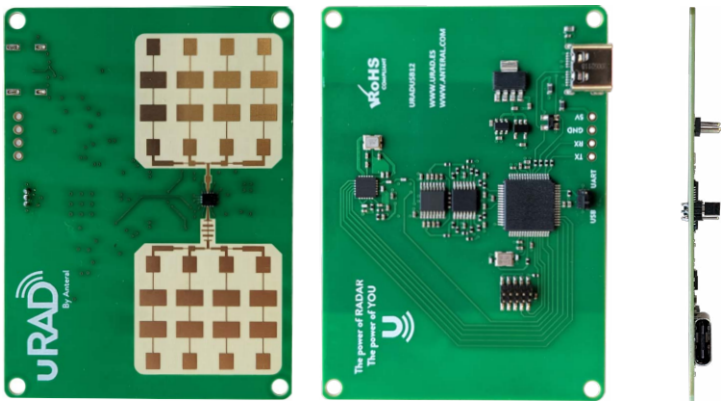


Figure 4.4: uRAD USB v1.2 Radar

4.1.2 The uRAD USB v1.2 Radar

Table 4.1: uRAD USB v1.2 Radar Parameters

Parameter	**Min**	**Typ.**	**Max**	**Unit**
Frequency band	24.005	-	24.245	GHz
Output power	16	18	20	dBm
Antenna gain	-	16.6	-	dB
Field of view	-	30x30	-	deg
Sidelobe levels	-19.8	-	-21.3	dB

The uRAD radar was suited to the project application as it provided FMCW waveforms (Section 2.7.1) which allowed for the estimation of the range and velocity of a target. The device specifications are presented in Tab. 4.1. The device could perform all signal processing onboard, producing range, velocity, and SNR measurements for up to five targets. Only an interface to an external system for retrieval, storage, and data processing was required. The uRAD radar could only estimate both distance and velocity using the triangle and dual-rate waveforms, with only distance produced from the sawtooth waveform (no sawtooth FMCW range-Doppler processing). A Graphical User Interface (GUI) option was also available, where processing was performed by the host PC. Though the uRAD GUI program allowed for some degree of tuning, it could not be customised for the scenario since code for the proprietary processing stages was hidden in pre-compiled shared object (.so) files. The .so files had to be requested from Anteral for the specific Linux OS on which the GUI program was to be executed. Fortunately, the device could be configured to bypass the onboard processing and output I/Q data. The I/Q data could then be processed offline on a computer using custom algorithms. Though the exact hardware schematic could not be acquired since it was proprietary, a generic design structure is presented in Fig. 2.13, which likely resembles that which is implemented on the uRAD USB v1.2.

4.1.3 Software Tools

MATLAB was chosen as the tool to perform simulations and develop the signal processing algorithms since it allowed for the rapid design of simulations, signal processing algorithms, performance testing, and the plotting and analysis of results. Specifically, the Phased Array Toolbox, Automated Driving Toolbox, Radar Toolbox, and Statistics Toolbox were used.

Python was selected as the language for configuring the uRAD USB v1.2 radars and performing real-time signal processing (Section 4.3). The implementation of the processing algorithm in Python is presented in Section 6.2. The radar hardware included a Python Software Development Kit (SDK) for radar configuration and data acquisition. Libraries such as NumPy, SciPy, and Matplotlib simplified the process of converting the processing algorithm from MATLAB to Python. NumPy was used for performing array operations on received radar sweeps, SciPy provided certain processing operations, and Matplotlib was used to plot graphs, such as frequency spectra and camera outputs, in real time.

Git was used for source code version control and to provide a shared code base to be accessed by the relevant computing systems. A link to the GitHub repository can be found in Appendix B. Controlling and transferring data to and from the Raspberry Pi was carried out over SSH using Visual Studio Code. An Android SSH application, Termius, allowed for controlling the test rig without the need for a PC.

4.1.4 Interface Methods

Serial communication (USB) between the radar module and the processing system was used to configure the radar and receive I/Q data. In the Windows operating system (OS), the device ID was found using the Device Manager program, where it appeared under the communication (COM) port menu. On Ubuntu and Raspbian, the USB device ID was prefixed by 'ttyACM'. The term 'ACM' stands for Abstract Control Model and 'tty' stands for teletypewriter. The radars were connected to ports `ttyACM0` and `ttyACM1` by default. Communication between the remote computing platform and a host PC used the SSH protocol. The use of SSH required the remote platform and PC to be connected to the same network using either WiFi or Ethernet. The protocol was also used for transferring

data from the remote platform to the host PC.

The Python OpenCV library was used to record camera data, which took place over a serial communication link. Fortunately, the Raspberry Pi Model 4B had four USB ports, simplifying communication between the processing and sensor subsystems. Alternatively, a USB converter that would connect to the Raspberry PI's 40-pin header would be required.

4.2 Waveform Selection

Since waveform design had a direct impact on detection performance, various waveform modulations were analysed to determine the optimal solution for the single lane-monitoring application. This section describes the investigation into, and analysis of, each suitable waveform available on the uRAD USB v1.2 radar module.

4.2.1 Triangle Modulation

Firstly, the triangle FMCW waveform (Section 2.7.1.2) was evaluated as it offered both radial velocity measurements and radial range measurements within two sweeps. Additionally, this waveform allowed the target's Doppler shift to be decoupled from the beat frequency, which is not possible when using sawtooth modulation (Section 2.7.1.1). Targets with negative or zero Doppler shifts (targets moving away from the radar and stationary targets, respectively) were filtered out by excluding them from further processing since they are of no interest when monitoring turn safety. The first method was to pair the beat frequencies associated with the largest SNR in the up and down chirp spectra. Though only the nearest target was of interest, a larger second target producing a similar or greater SNR resulted in frequent incorrect pairing of beat frequencies. It was hypothesised that the nearest target would always have the largest SNR. It was found that the SNR fluctuated for moving cars, so the devised method of selecting the largest SNR in each sweep was deemed unsuitable for a multi-target scenario: the presence of multiple beat frequencies with fluctuating SNRs resulted in ambiguity in associating the up-chirp beat frequencies with the down-chirp beat frequencies.

Since velocity was estimated using the change in beat frequency (Eq. 2.14) instead of a

phase change (Eq. 2.5), the maximum unambiguous velocity was much larger than that of the sawtooth modulation. This difference is clear when inspecting Equations 2.6 and 2.14 Though the accuracy of the velocity measurements could be improved with interpolation (Section 4.3.4), the coarse velocity resolution, related to the frequency resolution, could not be improved due to hardware limitations. For triangle modulation, the coherent processing interval is determined by the period of the waveform. A longer period has more samples per sweep, resulting in a finer frequency resolution post-FFT. The range resolution (Eq. 2.2) and frequency spacing (Eq. 2.18) calculations for the uRAD USB v1.2 with $f_s = 200$ kHz are shown below. The minimum Doppler shift was calculated by subtracting Eq. 2.10 from Eq. 2.11. These calculations were then compared to the datasheet values. The stated distance resolution of 1.5 m is believed to be due to signal windowing (Section 4.3.1) applied by the uRAD firmware processing. The velocity resolution of 1.5 m is half of the calculated value. In practice, the velocity resolution was found to be the calculated value of 3.1 m/s without interpolation. This was expected as the accuracy was limited by the frequency spacing Δf_D.

$$\Delta R = \frac{c}{2B} = \frac{3 \cdot 10^8}{2 \cdot 240 \cdot 10^6} = 0.625\,\text{m} \tag{4.1}$$

$$\Delta f = \frac{f_s}{N} = \frac{200 \cdot 10^3}{200} = 1\,\text{kHz} \tag{4.2}$$

$$\Delta f_D = (\frac{f_b^d - f_b^u}{2}) = \frac{\Delta f}{2} = 500\,\text{Hz} \tag{4.3}$$

$$\Delta v = \frac{\Delta f_D \cdot \lambda}{2} = 3.125\,\text{m/s} \tag{4.4}$$

4.2.2 Sawtooth Modulation

Section 2.7.1.1 identified sawtooth modulation as better suited to a multi-target scenario than triangle modulation. Though the range-Doppler coupling was present, it could be safely ignored provided the targets had sufficiently low velocity relative to the chirp rate. If a frame of chirps could be obtained while the target remained in a single range bin,

the changing phase of the return signal could be used to measure its velocity. A two-dimensional FFT and two-dimensional CFAR detector were required, after which detections would need to be clustered. The centroid of the clusters would represent the velocity and range measurement. The waveform had to be designed such that the maximum range and velocity were sufficient for the vehicle-monitoring application.

$$R_{max} = \frac{c}{2 \cdot BW} \cdot \frac{f_s}{2} \cdot T_c \tag{4.5}$$

$$R_{max} = \frac{c}{2 \cdot BW} \cdot \frac{f_s}{2} \cdot T_c \tag{4.6}$$

Starting with the FMCW range equation presented in Eq. 2.8 and assuming (f_D) is negligible, the maximum range is related to the ADC sampling rate following the Nyquist criterion (Eq. 2.17) and is presented in Eq. 4.5. Alternatively, if I/Q sampling is used, the maximum range is given by 4.6. For the uRAD USB v1.2 radar, it was found that the former was used in the provided GUI program. The number of ADC samples in a sweep could be represented as the product of the sampling frequency and the duration of the sweep (Eq. 4.7). Substituting for T_c in Eq. 4.5 allowed for the calculation of the maximum unambiguous range based on the number of ADC samples received (Eq. 4.8). Substituting for T_c in Eq. 2.6 allowed for the calculation of the maximum unambiguous velocity based on the number of ADC samples received (Eq. 4.9).

$$N_s = f_s \cdot T_c \tag{4.7}$$

$$R_{max} = \frac{c \cdot N_s}{4 \cdot BW} \tag{4.8}$$

$$v_{max} = \frac{\lambda \cdot f_s}{4 \cdot N_s} \tag{4.9}$$

It was then found that the uRAD USB v1.2 was not capable of meeting the required waveform parameters: a minimum of 50 samples per sweep was allowable using the provided firmware as stated in the uRAD SDK v1.1 user manual [65]. Additionally, the capture of

multiple back-to-back sweeps was not possible: samples were transferred as single sweeps per request with no option to capture a frame of sweeps. Using Eq. 2.6, 32 samples per sweep would be required to achieve a maximum unambiguous speed of 70 km/h. However, the maximum unambiguous range would become 20 m provided the bandwidth is not reduced (Eq. 4.8). Sawtooth modulation using the uRAD USB v1.2 failed to meet the technical requirements regarding maximum detection range and measurable velocity.

4.2.3 Dual-rate Triangle Modulation

Dual-rate triangle modulation is used to eliminate the incorrect pairing of up and down chirp beat frequencies in multi-target scenarios. The incorrect pairing of beat frequencies resulted in 'ghost targets'. A solution to this was to use two cycles with different slopes—two time-frequency chirp rates. True targets are those found to be in the same position (same beat frequency and Doppler shift) in both the first and second waves [66].

The downsides of this method are as follows: firstly, it takes four sweeps before detection can be made. The processing time increases compared to that of the regular triangle FMCW waveform as four sweeps must be processed simultaneously. Secondly, it requires there to be no missed detections in any of the sweeps for an overall detection to be made. For each period, all possible up and down beat frequency pairings must be made, and each pairing must be compared between the up and down chirps to determine which is present in both [66]. Due to these challenges, this method was not pursued further.

4.2.4 CW and FMCW Hybrid

An interesting approach involving alternating between the continuous wave (CW) and FMCW radar modes was investigated. The CW mode measured only radial velocity, while the FMCW mode was configured to obtain only range estimates using the sawtooth modulation. The CW operated at the base frequency of 24 GHz and the FMCW occupied the rest of the bandwidth. Since the target's Doppler shift would not exceed the kHz range (based on the radar parameters and expected target speed), the CW wave only required a fraction of the available bandwidth. This method could work if one assumes the lead

vehicle in the lane dominates the detected speed measurements: the largest SNR was associated with the closest target. It was determined that using the sawtooth modulation for obtaining only range estimates would not indicate whether the closest detection in range was associated with the speed measured during the CW cycle. Triangle modulation was then used to determine if the target was moving, though the same ghosting issues in the multi-target scenario were observed. This could be resolved by using the CW measurement to identify incorrect beat frequency pairings by the FMCW radar: range and speed estimates, calculated using all possible beat frequency pairings, are compared to the CW speed measurements. A single uRAD USB v1.2 was found to be able to efficiently switch modes. However, the method of switching between CW and FMCW was deemed computationally inefficient, since both waveforms need to be processed separately and range and velocity estimates must be computed for all beat frequency pairings. It was found that the presence of clutter, interference, false alarms, and certain targets producing multiple detections would result in a great deal of unnecessary processing.

4.2.5 Selection, Motivation, and Considerations

The triangle waveform was selected, as it was found that the pairing of beat frequencies between the up and down ramps could be achieved with sufficient accuracy based on the single-lane monitoring scenario. This waveform provided sufficient maximum unambiguous range and velocity and was not as processing-intensive as the alternative methods (sawtooth, dual-rate, and CW/FCMW hybrid). Section 4.3 describes the further processing based on the triangle FMCW waveform and the method presented in Section 4.3.6. Though the waveform parameters must be chosen to provide a suitable maximum unambiguous range (theoretical), the achievable maximum range—the range at which cars can be detected—is limited by factors such as the transmit power (P_t), antenna gain (G_{TX} and G_{RX} for the transmitter and receiver, respectively), and target RCS (σ) at the maximum range. The radar equation, Eq. 4.10, incorporates various radar parameters to provide an estimate of the SNR for a specific target at a specific range [49] [52]. In Eq. 4.10, F is the receiver noise figure, k is the Boltzman constant, T is the receiver temperature in Kelvin, R is the range to the target, λ is the wavelength of the carrier frequency, and t_{meas} is the total measurement

time. For a specific value of PFA, the higher the target's SNR, the greater the desired probability of detection.

$$SNR = \frac{\sigma\, P_t\, G_{TX}\, G_{RX} \lambda^2 t_{meas}}{(4\pi)^3\, R^4\, k\, T\, F} \tag{4.10}$$

4.3 Processing Algorithm Design

This section discusses the design of the processing sub-algorithms, with the final processing method presented in Section 4.3.6. These algorithms were designed to operate on the triangle FMCW waveform discussed in Section 2.7.1.2. The selection of a suitable window function, FFT interpolation length, filtration of DC components, and detection algorithms are presented. The high-level diagram presented in Fig. 2.16 shows the various pre-processing stages. The section concludes with a summary of the key processing aspects and optimisations thereof.

4.3.1 Signal Windowing

The process of signal windowing, discussed in Section 2.7.3, was the first stage of the processing algorithm, following the DC cancellation (subtraction of the mean) as shown in Listing 1. DC cancellation was required to remove any DC offsets and reduce the transmitter feed-through (caused by the proximity and constant transmission of the FMCW waveform). Multiple window functions were tested to determine which offered the best detection probability without degrading the range resolution beyond the minimum requirement. This was narrowed down to the Kaiser, Blackman, Hanning, and Taylor windows. Initially, the Taylor window was selected as it allowed for ensuring the narrowest mainlobe width for a given sidelobe level. Figure 4.5 shows a comparison of several window functions for a real car target, highlighting the information presented in this section. The Taylor window was selected as it allowed for tuning of the maximum sidelobe level (SLL) relative to the mainlobe peak as well as the number of nearly-constant sidelobes $\bar{n}$.

Listing 1 Python I/Q DC cancellation by subtraction of the mean voltage

```
max_voltage = 3.3
ADC_bits = 12
ADC_intervals = 2**ADC_bits
numVoltageLevels = max_voltage/ADC_intervals
i_u_scaled = np.multiply(i_u, numVoltageLevels)
i_u = np.subtract(i_u_scaled, np.mean(i_u_scaled))
```

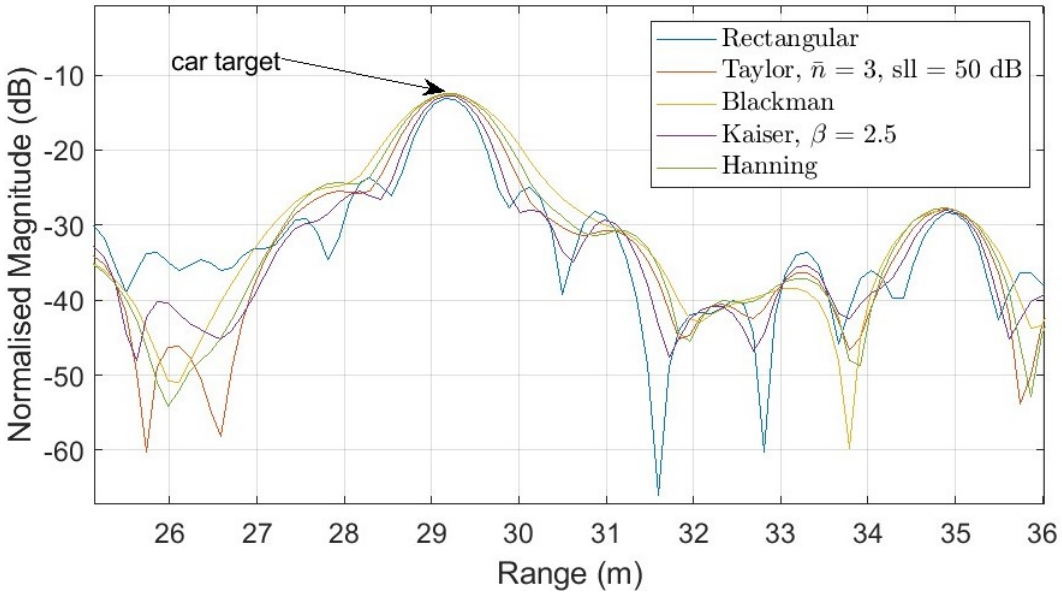


Figure 4.5: Comparison of normalised beat frequency spectra for several window functions on a peak generated by a car target using triangle FMCW modulation

4.3.2 Detection Algorithms

The process of radar detection involves declaring whether a target is present or absent in each range bin. The index of the detection—the sample number at which the detection occurred—was mapped to a frequency value for use in the triangle FMCW range and velocity calculation. The set of frequency values corresponding to each index was generated as an array of multiples of the frequency spacing Δf using Eq. 2.18. Two algorithms for detecting beat frequencies were compared to determine which was most suitable based on detection performance and computational cost.

The MATLAB `findpeaks()` function allowed for identifying the indices of peaks in a

data set based on known characteristics of the peak, such as width and relative height (prominence). Though it could be applied to radar data for identifying beat frequencies, it required the characteristics of the peaks (such as SNR and width) to be constant, and so was not robust to changes in the environment, target fluctuations, and differently-shaped targets. An equivalent function is freely available as part of the Python SciPy library [67], so it could be implemented on the lower-level Raspberry Pi hardware if needed.

It was found that CFAR algorithms were possibly better-suited for ideintifying beat frequencies compared to the MATLAB `findpeaks()` algorithm. CFAR detection was first implemented using the MATLAB `CFARDetector` System Object for triangle and the `CFARDetector2D` System Object for sawtooth range-Doppler processing. Four variations of this algorithm were tested: CA (Cell Averaging), GOCA (Greatest-Of Cell Averaging), SOCA (Smallest-Of Cell Averaging) and OS (Ordered Statistic) CFAR. Initially, OS-CFAR was selected as it was determined by [68] to be effective in automotive applications. Since the PFA and the number of training cells N_{train} are used to derive the CFAR threshold scaling factor (Equations 2.20 and 2.21), both could be tuned based on the target, scenario, surroundings, and processing method. Additionally, the rank parameter r of OS-CFAR is another independent parameter that may be tuned. In addition to tuning PFA, N_{train}, and r, the number of guard cells—cells surrounding the cell under test to be excluded from the training set—could be tuned (literature discussed in Section 2.7.6). Research by [55] was used for selecting the optimal OS CFAR parameters. It was recommended to use a training set of between $N_{train} = 24...32$ cells. The rank r is based on the size of the training window N_{train}, the size of the minimum suppressible clutter L, and the maximum expected target width in range. For N = 32, $\frac{N_{train}}{2} < r < \frac{3\,N_{train}}{4}$.

4.3.3 Angle Correction

Since the radar system estimated radial distance and radial velocity, adjustments to the results based on the angle of arrival (AoA) were needed to improve the radar measurement accuracy. The need for angle correction can be seen in Fig. 5.5, which shows that the velocity estimate deviates from the true speed of the vehicle. Range correction is not required, as nearby targets in the deviation region are classified as a hazard regardless

of the measurement accuracy as long as they are associated with a closing speed. Since the uRAD USB v1.2 radar module was not capable of measuring AoA, an estimate was obtained based on the uniform single-lane monitoring scenario by assuming the main road was of standard width. Figure 4.6 shows the relationship between the radar measurement and the true value. As the target approaches the host vehicle, the AoA (θ) between the true value and the radar measurement increases. The proposed solution assumes a width W of the two-lane road. The side w of the triangle shown in Fig. 4.6a can be estimated as one-quarter of the road width for the right-side oncoming lane. The angle θ was calculated using Eq. 4.13. The true speed could then be estimated since θ is common to both triangles in Fig. 4.6. Equation 4.12 shows how the target's speed ($V_{desired}$) is calculated based on the angle of arrival (θ), which is derived from the range estimate (R_{est}) and the road width (w) as shown in Eq. 4.11.

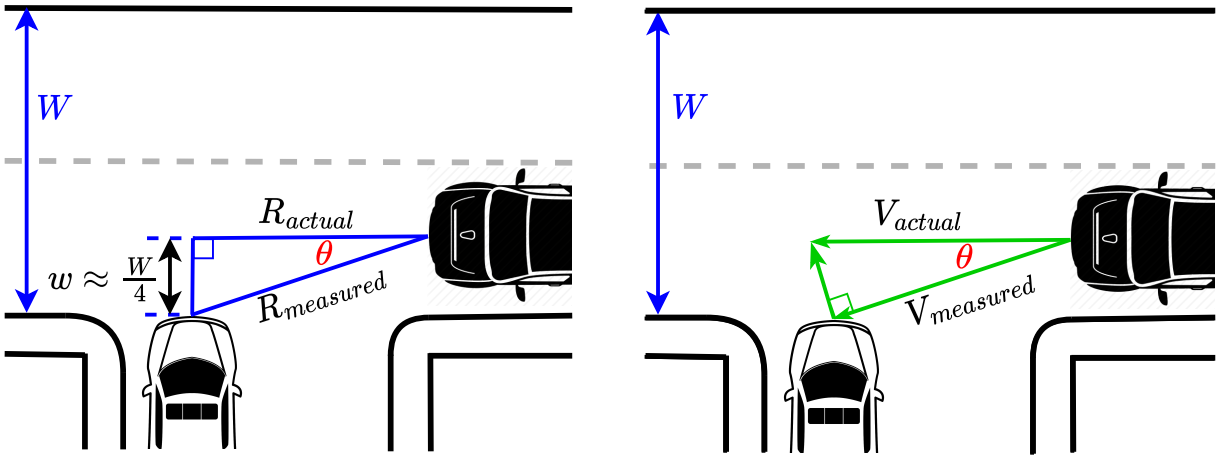
$$\theta = \arcsin\left(\frac{w}{R_{est}}\right) \tag{4.11}$$

$$V_{desired} = \frac{V_{est}}{\cos\left(\theta\right)} \tag{4.12}$$

Angle correction was implemented for the opposite lane by setting $w = \frac{3W}{4}$. The lane width W was assumed to be four meters. Since the car width and the reflections from various points on the front bumper would influence the range measurement, an exact value of w was not required. Additionally, the angle correction assumed the peak SNR would be originate from the midpoint of the front bumper, which may not hold true in practice.

It was found that adjusting the angle (ϕ) of the left-side radar, depicted in Fig. 4.7, allowed for cars on the outer lane to be detected. When the radar is parallel to the road, all of the cars on the inner lane lie on the radar boresight line where the antenna gain is greatest. Rotating the beam towards the opposite lane shifts energy from the inner to the outer lane, reducing the SNR for cars travelling away from the system and increasing the SNR for cars approaching along the outer lane.

After adjusting ϕ to improve outer-lane detection and attenuate the returns of inner-lane vehicles, the following condition was added: if the range estimate was greater than the



(a) Range trigonometry (b) Velocity trigonometry

Figure 4.6: Diagram showing the angle of arrival estimation, where the angle of arrival obtained using the range estimate and assumed road width is used to calculate the target's speed

minimum boresight range $R_{boresight}$, the subtraction of θ from the offset ϕ yielded the AoA (β). This is different to the right-side case, where θ is the AoA with respect to both perpendicular to the host vehicle and to the radar boresight. For the left-side case, β represents the AoA with respect to the radar and θ represents the AoA relative to perpendicular to the host vehicle. The geometry of the scenario is illustrated in Fig. 4.7, with Eq. 4.13 showing the relationship between the various angles. When the range is less than the boresight range, the AoA is relative to the opposite side of boresight. Since this detection range is to be observed by collision warning sensors, this case was not handled.

$$\beta = \phi - \theta \tag{4.13}$$

The angle ϕ was calculated by finding the maximum angle that would allow vehicles to be detected at the maximum range of 60 meters. It was decided that the edge of the beam, -15° from boresight (for the 30° beamwidth), should be used to detect cars at 60 meters while minimising the portion of the beam covering the inner lane. The minimum boresight offset was calculated assuming a road width of 4 meters and summed with half the beam width as shown below:

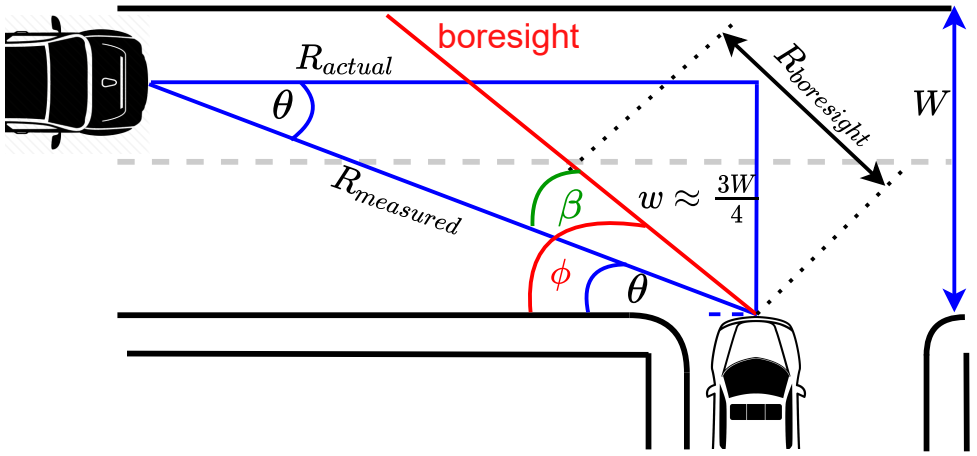


Figure 4.7: Angle correction for left radar with an offset from parallel to the road

$$\phi = \arcsin \frac{\frac{3}{4}4}{60} + 15^\circ \ = 2.9^\circ \ + 15^\circ \ = 17.9^\circ$$

4.3.4 Fourier Transform and Interpolation

Experimentation into improving the velocity measurement accuracy of the uRAD USB v1.2 was undertaken. Since the triangle modulation was used, the Doppler frequency accuracy was equal to half the beat frequency accuracy (Section 4.2.1). Though this accuracy was sufficient for range estimation, it resulted in a course Doppler accuracy. The difference in velocity and range resolutions is presented in Section 4.2.1, where the velocity resolution was calculated as 3.125 m/s. In the case where interpolation was not used, the velocity resolution and accuracy were equal. Interpolation was used to obtain a finer accuracy but since no information is added, multiple targets in a single frequency bin will not be better-resolved post interpolation.

The use of Gaussian and Parabolic interpolation was explored [69], though it was decided not to implement these methods. A finer velocity resolution was not necessary: the system needed only to provide a measurement greater than that of a human estimate. Concerning range resolution, only a single lane was monitored on each side, and closely-spaced vehicles were considered a single hazard. The computational cost and difficulty of implementing such

algorithms provided further grounds for the exclusion thereof. Consequently, zero-padding (Section 2.7.5) the time domain signal was used. The computational burden was minimal and allowed for an efficient power-of-two sample size for optimising the performance of the FFT algorithm. The input I and Q sequences were zero-padded from a length of 200 samples to 1024 samples before performing the FFT. Though 1024 seems large, the uRAD GUI program performed a 4096-point FFT, which was deemed excessive. The resulting value of the frequency bin spacing was calculated as 195.3 Hz:

$$\Delta f = \frac{f_s}{N} = \frac{200 \cdot 10^3}{1024} = 195.3125\,\text{Hz}$$

Range resolution (Eq. 2.2) is based solely on the bandwidth of the transmitted signal and not on the frequency spacing. Increasing the sampling rate results in an increase in the number of samples over a given period, corresponding to an increased maximum beat frequency and consequently an increased maximum unambiguous range. However, the ratio $\frac{f_s}{N}$ remains constant. The uRAD USB v1.2 used a sampling rate of 200 kHz as found in the provided GUI application source code. This was likely chosen based on the maximum range that the radar could detect targets (Eq. 4.10) constrained by parameters such as the transmit power and antenna gain. The maximum chirp duration of 1 ms ($N = 200$) limited the maximum frequency resolution to 1 kHz when Eq. 2.18 is applied. The maximum beat frequency was initially believed to be equal to the sampling rate, since IQ sampling was used. However, the uRAD GUI program used the Nyquist sampling rate of 100 kHz, calculated using Eq. 2.17 instead. After testing, it was found that the maximum beat frequency was indeed 100 kHz.

Lastly, it was found that the FFT output must be divided by N to normalise the total energy in the signal. The effect of this was a scaling of the frequency amplitude spectrum. Since only the beat frequency was of importance (not the target's SNR) in the lane monitoring application, this division was not implemented.

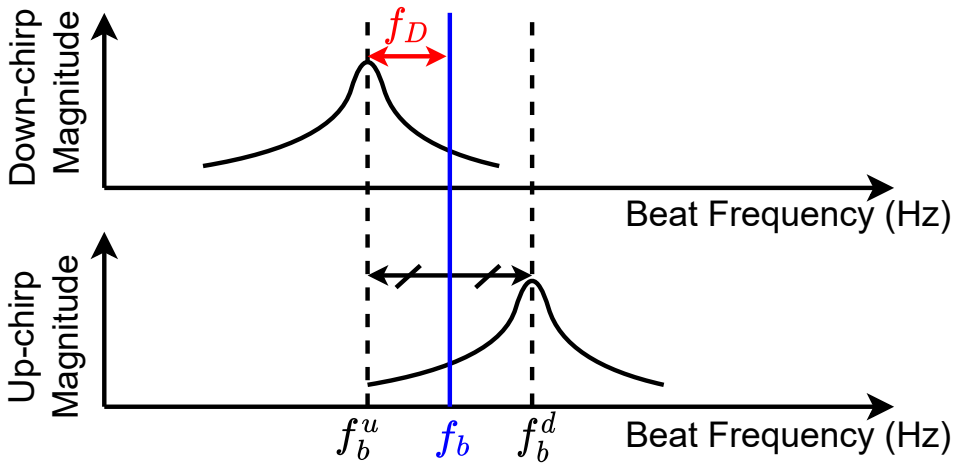


Figure 4.8: Decoupled beat and Doppler frequencies obtained using triangle FMCW modulation from the flipped negative down-chirp spectrum (top) and the positive up-chirp spectrum (bottom)

4.3.5 Spectrum Division Method

The spectrum division method was formulated based on the linear track of vehicles travelling along a single-lane road. Firstly, cars travelling along a single lane have a known trajectory. All cars passing through an intersection can occupy a finite number of positions (or range bins) within a given period. The lead car sets the speed of any cars behind it, so a range resolution of under 10 meters was deemed unnecessary: closely spaced cars can be considered a single target in this application. Additionally, the lead car masks most of the car behind it, as radio frequency (RF) signals cannot penetrate the metal. For a radar facing a single lane, the closest car of interest generally has the largest SNR and a closing velocity. Additionally, vehicles have limited acceleration capabilities. Based on these limitations, a certain maximum frequency shift exists between the up and down ramps when observing cars on a main road.

The proposed method was to split the spectrum into wider bins—bins comprising a group of frequency bins—and compute the local maxima of the detection magnitudes in each bin. This method simplified processing, allowed for multiple target detections at various ranges, and increased the probability of correctly associating beat frequencies of the up and down ramps with the correct target. A 1024-point FFT was performed on each received sweep

(Section 4.3.4).

Based on the triangle FMCW waveform diagram (Fig. 2.15), the difference between the up chirp reference and the echo—which equates to the beat frequency—is always positive. when $f_b > f_D$. Conversely, the difference is always negative for the down chirp. Consequently, negative and positive halves of the up and down chirp spectra were discarded, respectively. For beat frequencies to occur in those regions, the time delay Δt (Fig. 2.15) would have to be negative, which is impossible. The down-sweep negative half was flipped about the y-axis to allow for using a set of only positive frequency values. Since the FFT algorithm placed the negative half-spectrum in the second half of its output, the magnitude of the negative frequencies increases from right to left, opposite to the positive half of the spectrum. Flipping the negative half-spectrum about the y-axis simplified processing by ensuring both the up- and down-chirp spectra were in the same format. Figure 4.8 shows how the decoupled beat frequency and Doppler frequency were obtained based on Equations 2.15, 2.16, and 2.3. The up and down chirp half-spectra were then passed to the CFAR detector, the output of which was divided into $\frac{512}{32} = 16$ bin groups containing 32 samples each. The bin width of 16 samples was selected as follows:

$$v_{max} = 70\,\text{km/h} \tag{4.14}$$

$$f_D^{max} = \frac{2v_{max}}{\lambda} = \frac{2 \cdot \frac{70}{3.6}}{0.0125} = 3.11\,\text{kHz} \tag{4.15}$$

$$(f_b^d - f_b^u)_{max} = 2f_D^{max} = 6.22\,\text{kHz} \tag{4.16}$$

$$\Delta f = \frac{f_s}{N_{FFT}} = \frac{200 \cdot 10^3}{1024} = 195.3125\,\text{Hz} \tag{4.17}$$

$$N_{gate} = \frac{(f_b^d - f_b^u)_{max}}{\Delta f} = \frac{6222.22}{195.31} = 31.86 \approx 32\,\text{samples} \tag{4.18}$$

Using the frequency gate width of 6.22 kHz and the frequency bin spacing of 195.31 Hz, we need a gate width of at least 32 bins. A benefit of this bin width is that it is a power of two, which is favourable for computations using computers. Additionally, this bin width corresponds to a range of 3.9 meters using Eq. 2.9 where $f_b = 32 \cdot \Delta f$, which is roughly the diagonal length of a car (as it would be seen by the radar). Choosing a wider bin group would reduce the range resolution: if two targets are in a bin, the target with the larger SNR

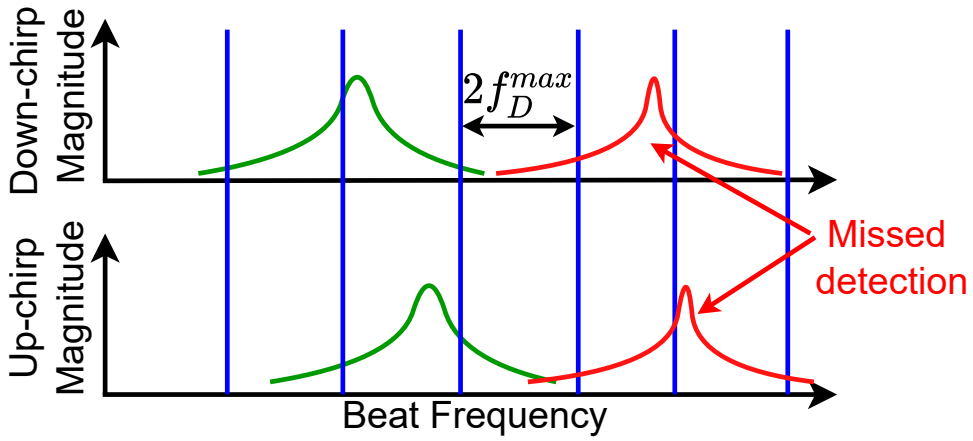


Figure 4.9: Illustration of the spectrum division method for triangle FMCW modulation showing the mirrored negative half of the down-chirp spectrum (top) and the positive half of the up-chirp spectrum (bottom). The case where the beat frequencies do not lie in the same bin (a missed detection) is shown in red. The blue lines indicate the bin boundaries.

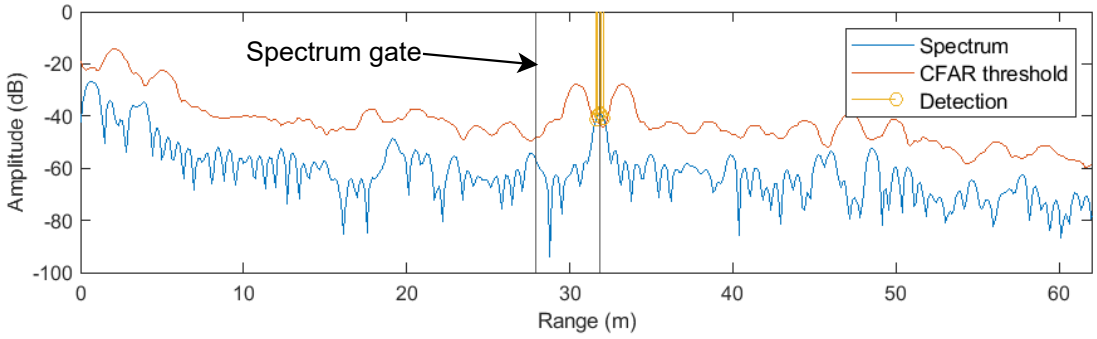
is detected. The current width is the narrowest allowable width that meets the maximum detectable velocity requirement. Note that the maximum speed of 70 km/h refers to the target's radial velocity: the maximum detectable velocity will be slightly below this value and is based on the angle between the car and the radar as discussed in Section 4.3.3.

The largest value in each bin of the up and down chirp was then paired and used to obtain range and velocity estimates using Equations 2.15, 2.16, and 2.3. The largest value in each bin was required as multiple points could breach the CFAR threshold. Only if detections were made in the down chirp would the corresponding bin in the up chirp be examined. If a peak was then found in the up-chirp bin, the two frequencies were assumed to be caused by the same target. This assumed a single target was detected in the bin, which is likely if the radar beam and CFAR threshold are configured to attenuate cars travelling away from the radar (those on the adjacent lane). Figure 4.9 shows the method of spectrum division, including the case of missed detection (shown in red) when the beat frequencies occupy different bins. This occurs periodically as the target moves between the wider bins.

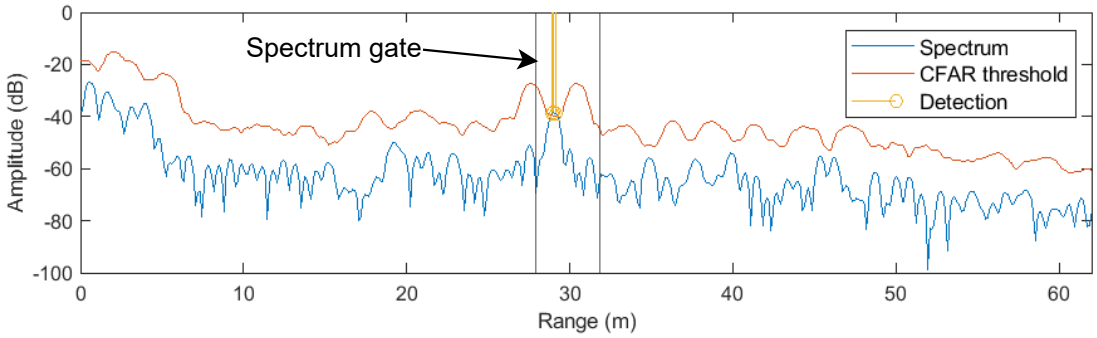
4.3.6 Spectrum Gate Method

The spectrum gate method was conceived to address the periodically missed detections observed when using the spectrum division method. For each detection in one ramp of the triangle modulation, a gate starting at the beat frequency index and extending to the furthest index that the beat frequency could shift to in the second ramp was formed. The gate corresponded to the maximum expected Doppler shift for cars travelling along a main road and was used to limit the search range in the opposite chirp. The gate width was set to the spectrum division bin width calculated in Section 4.3.5. The down chirp was selected for the initial detection since for positive Doppler shifts, the beat frequency of the down chirp is larger than that of the up chirp (Fig. 2.15). The spectrum gate improved robustness since false alarms needed to occur in both sweeps within the specified frequency gate. Additionally, the gate was used to filter out negative Doppler targets (targets moving away from the radar) and stationary targets. The spectrum division method formed the basis of this method, offering the same benefits discussed in Section 4.3.5. The method of spectrum gating addressed the missed detections resulting from bin straddling when using the spectrum division method (Fig. 4.9).

After receiving the triangle FMCW wave, each of the 16 bins of 32 samples each (Section 4.3.5) was iterated over. After a detection in the down-chirp was made, the spectrum gate algorithm was used to limit the search in the up-chirp. A detection is depicted in Fig. 4.10 where a detection in the up-chirp lies in the gate spanning 32 bins from the down-chirp beat frequency. If the beat frequency occurred on a bin edge, it could result in detections in two adjacent bins since the local maxima of each bin would be the result of the main- or sidelobes of the target. Though this could cause detections in two adjacent bins, the measurement of range and velocity would be similar. Additionally, fluctuating static targets could result in detections with very small velocities, which were filtered out by setting a minimum expected velocity. It was found that the paper on tracking technology for onboard collision warning [41] presented a similar concept, referred to as “association” logic. A gate was placed around predicted positions and logic was used to determine if the detection was within the gate. It mentions the possible inaccuracies when multiple targets lie within the same gate. Section 2.4.2 provides further information on this literature. Figure 4.11 shows



(a) Negative half of the down-chirp beat frequency spectrum mirrored about 0 Hz



(b) Positive half of the up-chirp beat frequency spectrum

Figure 4.10: Graphs showing the formation of the spectrum gate (applied to the up-chirp spectrum) extending from the down-chirp detection

a flowchart of the final processing algorithm. A key difference between this work and the work presented by [41] is that a Kalman filter was not implemented. Though a Kalman filter would improve measurement accuracy, it remained out of scope for this project.

4.3.7 Summary

Based on the presented algorithms, the following set of parameters was tuned for the intersection monitoring scenario:

1. Window function type (trade-off between resolution and SNR)
2. CFAR training cells, guard cells, and scaling factor (related to PFA)
3. Interpolation type and length

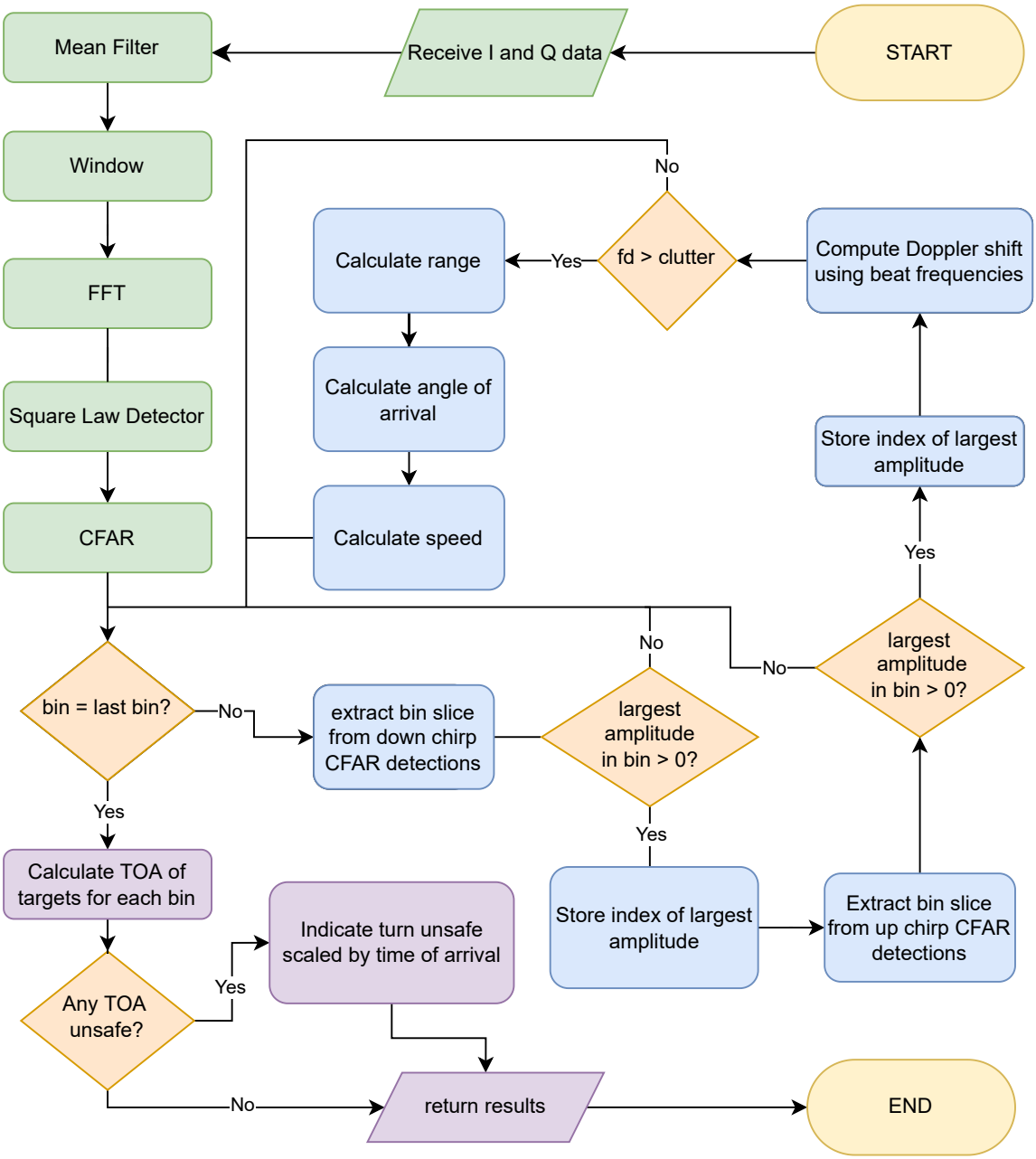


Figure 4.11: Flowchart of the implemented signal processing algorithm. Green blocks indicate preprocessing stages, blue represents processing stages, orange represents conditional statements, and purple indicates data processing stages

4. Gate width and spectrum division width

An effective solution was designed for observing each lane of traffic at a two-lane, unregulated intersection. The true PFA was reduced using the spectrum gate method by limiting the search area for pairing beat frequencies when using the triangle FMCW waveform. Optimisations include processing only the relative spectrum halves of the up and down chirps, spectrum division for multi-vehicle detection, and the spectrum gate method for tracking. These optimisations made it possible to use a low-cost radar module for collision prediction at road intersections.

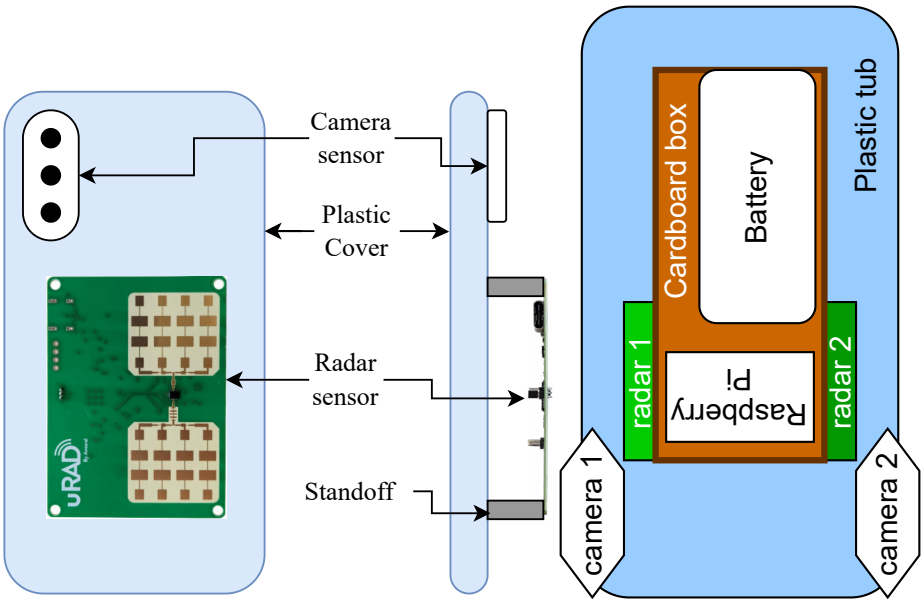
4.4 Test Rig Design

This section describes the various test rigs created during the project development cycle. As the system evolved, more sophisticated test rigs were needed. The intermediate test rigs were used to verify specific radar and processing operations, while the final system was used to perform roadside system testing. The section begins with descriptions of the intermediate test rig designs. The selection of the peripheral hardware used in the final prototype is then presented. Finally, the roadside prototype system design is discussed.

4.4.1 Intermediate Test Rig Designs

Two intermediate test rigs were designed: the cellphone camera and uRAD USB v1.1 test rig and the system-in-a-tub test rig. The former was used to develop the processing algorithm and the latter was used to verify that the dual radar and dual camera system could work. Additionally, the tub system was used to develop the Python real-time plotting scripts in a laboratory.

The phone camera and uRAD USB v1.2 required fixing the uRAD to the back of a cellphone. The phone camera would be set to record continuously while the uRAD recording process was handled by a PC. The burst raw data captures were compared to the video, where voice indicators were used to indicate the portion of the video during which radar data was captured. The design is presented in Fig. 4.12a which shows how the uRAD was



(a) phone camera test rig (b) System-in-a-tub test rig

Figure 4.12: Diagrams of the two intermediate test rigs

mounted to the phone via standoffs and a plastic phone cover. This system was then placed in a rectangular cardboard box which would hold the radar and camera in place.

The system-in-a-tub test rig required fitting all the components into an existing enclosure to be tested in a laboratory. This rig was used for optimising the real-time plotting update rate (Section 6.3) and to ensure the system could operate with four sensors connected. Additionally, it was used to prove that a 3 ampere power supply was required: the Pi reported an under-voltage warning when the system was run on a standard cellphone power adapter. Figure 4.12b shows how the various components were arranged in the plastic tub. The radars were fixed to a cardboard box while the cameras were attached to the lip of the plastic tub. A photograph of the implemented system is shown in Fig. 6.6.

4.4.2 Peripheral Hardware Selection

Since this was an initial investigation into the problem, a custom low-level embedded radar system was not designed. Rather, a general-purpose computing system and radar module were used (Section 4.1.1). A personal computer was used for real-time processing at first,

with the final prototype running on a Raspberry Pi Model 4B. The Raspberry Pi computer simplified deployment and proved that the system could operate on lower-level hardware by meeting the update rate requirement (UR1, Tab. 3.1). It was selected as it allowed for quick deployment and for interfacing with multiple sensors via USB.

Considerations regarding camera selection included frame rate, the communication interface, and the form factor. Initially, the Raspberry Pi Camera was considered. It was found that only a single Pi Camera could be connected at a time using the dedicated connector. USB cameras were then chosen, which required the processing platform to have a minimum of four USB ports (for each radar and camera). Two Logitech C270 HD Webcams were available at the University of Cape Town and these met the required camera specifications: a minimum update rate of 30 frames per second, a USB interface, and a minimum resolution of 320x240 pixels.

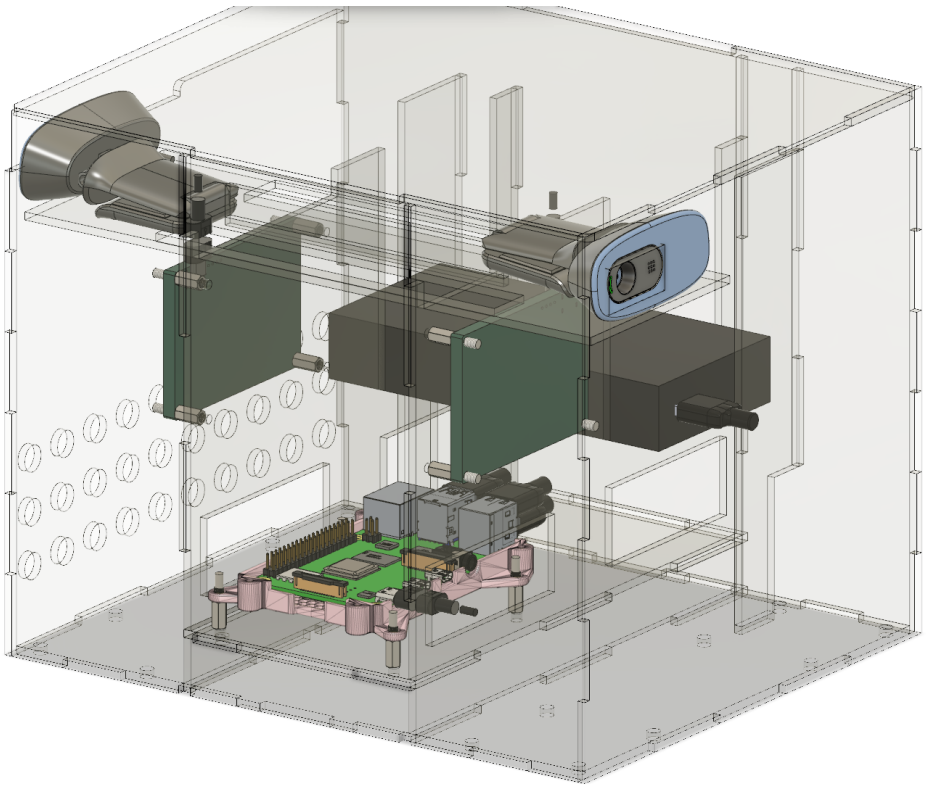
The power supply was chosen to match the Raspberry Pi's 5.1 V/3 A requirement. Initially, a Samsung Super Fast Charge power adaptor providing 5 V/3 A was used to prevent the under-voltage warning caused by running the full system. A Samsung Super Fast Charge power bank was used to power the test rig for roadside deployment. At the time of research, no other power banks could be found that could supply the required 5 V/ 3 A. This was attributed to the high current of 3 A which required sophisticated hardware and thicker USB cables. An alternative was to use a mini-UPS (Uninterruptible Power Supply) and a custom voltage regulator. This would require additional design time and the design of power electronics was outside the scope of the project. The implemented system block diagram is shown in Figure 6.4, showing the various interfaces between the peripherals and the Raspberry Pi processing system. The various peripherals can be seen in the final test rig pictures presented in Fig. 6.7.

4.4.3 Roadside Unit Test Rig

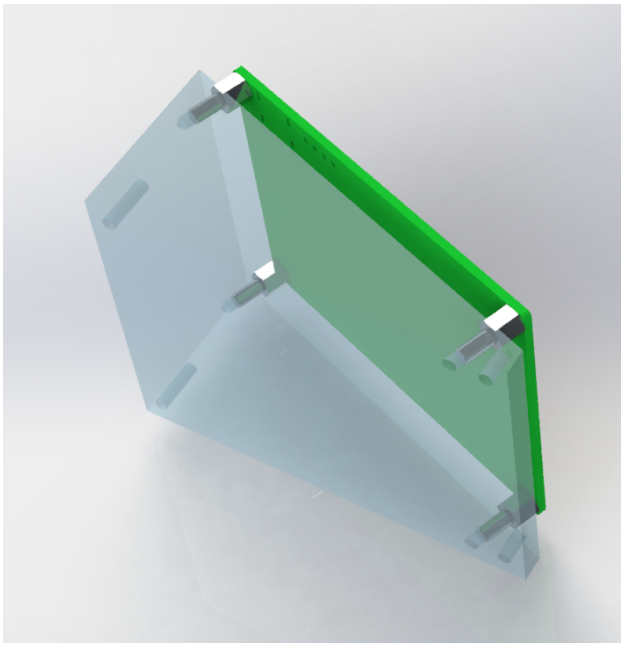
Though multiple intermediate test rigs were used, only the final roadside unit test rig followed a design process. A 3D-printed enclosure was deemed to be unsuitable: aside from not having access to a large enough printer, it was deemed too costly and would take many hours, if not days, to print. Consequently, a custom acrylic enclosure was designed. The

enclosure was modelled in CAD using Autodesk Fusion 360. Modelling involved designing internal structural components for housing the various system peripherals, with CAD renderings of the various peripherals obtained online. An interlocking structure using box joints was used to produce a rigid system. The final CAD model is presented in Fig. 4.13a.

The radars and cameras were placed as close to the top front corners as possible, to ensure an optimal view of the scene. The camera holder was designed to be removable so that the radar hardware could be easily removed. Additionally, wing nuts were chosen to simplify the installation and removal of the uRAD devices. Four legs were added to ensure the radar was at the correct height (as it would be when installed on a vehicle) which had adjustable feet to account for the pavement unevenness. The Raspberry Pi was placed on a raised platform to ensure the nuts did not protrude out the bottom of the device. Lock nuts, standoffs and screws were used to fasten components to the enclosure. A circular pattern was cut into the sides for ventilation and aesthetics. Additional cooling from fans was not necessary as the Pi CPU temperature did not exceed 60° C. This temperature was below the processor's maximum tolerance of 85° C [70]. Finally, an angle adjustment wedge was designed to shift the left radar main beam to focus on the outer lane (Section 4.3.3). Figure 4.13b shows the CAD model of the angle adjustment component. Section 6.4.3 describes the construction of the test rig, with photos from various angles presented in Fig. 6.7.



(a) Oblique view of the system



(b) Radar angle adjustment component

Figure 4.13: CAD models for the roadside unit test rig

Chapter 5

Simulation

Simulations were performed to understand how the system would perform in various scenarios and determine whether the hardware met the minimum requirements. The section begins with descriptions of the various simulated scenarios. Following this, the various MATLAB simulation tools are described. Finally, the simulation results are presented and discussed.

5.1 Simulation Configurations

This section describes how the simulations were implemented in MATLAB. The simulated scenarios are discussed first to provide a high-level understanding of how the system was expected to work. Then, two MATLAB radar simulation tools—the Automated Driving Toolbox (ADT) and the Phased Array System Toolbox (PAST)—are presented along with the use thereof to simulate the radar hardware and scenarios.

5.1.1 Simulation Models and Scenarios

A simulation of the radar system in the road intersection scenario was required to assess whether the uRAD USB v1.2 system could meet the minimum user and technical requirements (Tables 3.1 and 3.2). The uRAD simulation model was tested in the scenarios described in Tab. 5.1. Scenario D was of most interest as it was the most complex scenario

to correctly resolve. Figure 5.4 illustrates scenario D, showing how only the inner-lane vehicle is detected despite the presence of an outer-lane vehicle in the radar beam.

The ADT was used for the initial simulations. Though it provided little control over the radar modelling and signal processing, the GUI allowed for rapid simulation configuration. Following this, the PAST was used for modelling the radar hardware. The radar model was then tested in the defined scenarios using the proposed processing algorithm discussed in Section 4.3.6.

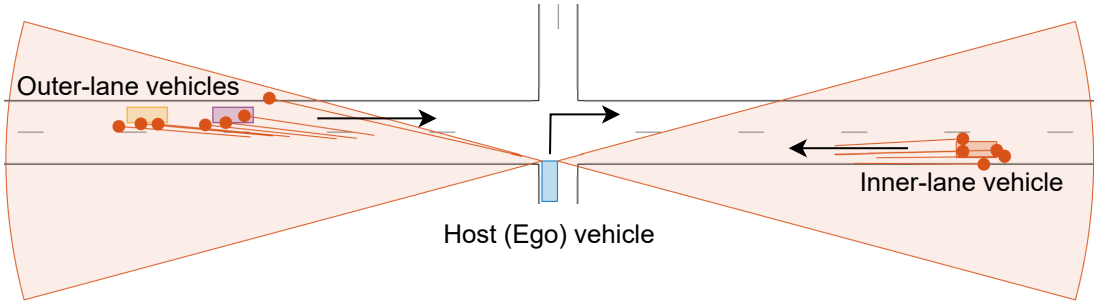
Table 5.1: Simulation Scenarios

Scenario	**Description**
A	One or more cars approaching from the right side of the host vehicle
B	One or more cars approaching from the left side of the host vehicle
C	One or more cars approaching from each direction
D	One car approaching from the left side being masked (obscured) by one or more cars travelling on the inner lane

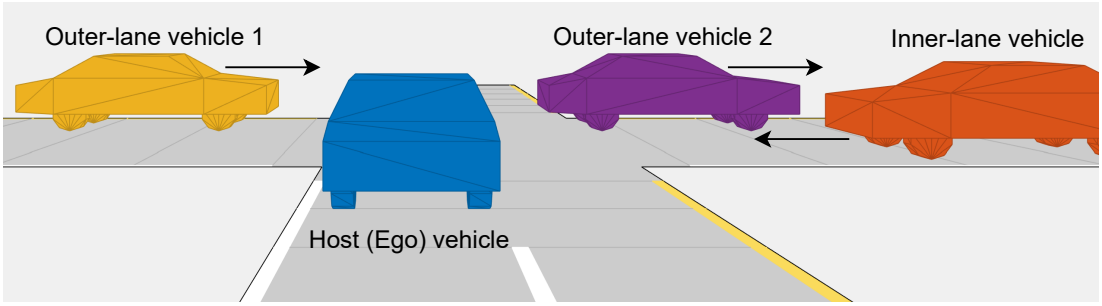
5.1.2 Automated Driving Toolbox Simulation

The ADT was first experimented with as it was a logical starting point for a MATLAB-based simulation. The toolbox allowed for simulating complex road scenarios for cars fitted with numerous sensors such as cameras, LiDAR, and radars. The simulation could be designed using a GUI application, the Driving Scenario Designer, or a MATLAB script. A script was created to extract the range and velocity detection data from each sensor, correctly assign the detections to specific targets (when a target passes the host vehicle, data from that target is present on the other sensor), and compare them to the actual position and velocity of the simulated targets. The data was then saved and exported to an Excel spreadsheet for analysis, which is presented in Section 5.2.1. Figure 5.1 shows the graphical outputs provided by the MATLAB ADT application. The ADT was of limited use as did not produce I/Q data—the measurements were based on the radar datasheet metrics such as accuracy, resolution, and PFA. Consequently, the PAST was used for further simulation,

including the simulation of the custom processing algorithm.



(a) Birdseye Point Cloud



(b) Egocentric mesh models

Figure 5.1: Automated Driving Toolbox Simulation Visualisations

5.1.3 Phased Array System Toolbox Simulation

The PAST was used to build the simulation and generate graphical representations of the scenario. This process required a lower-level design compared to the ADT simulation and provided greater control and flexibility. Additionally, custom processing could be added to the simulation loop. Figure 5.2 shows how the simulation loop was designed in MATLAB. The targets' positions were updated (based on a time increment and the targets' defined speed), after which the wave was transmitted, propagated through space, and reflected off of the targets before the resultant I/Q samples were stored as rows in a matrix. Signal processing was applied post-capture to each captured sweep. The signal processing block (Fig. 4.11) represents the proposed processing algorithm, presented in Section 4.3, which produced the range and speed measurements of the simulated targets. Figure 5.3 presents a snapshot of a scene rendered using the MATLAB `phased.ScenarioViewer` System Object.

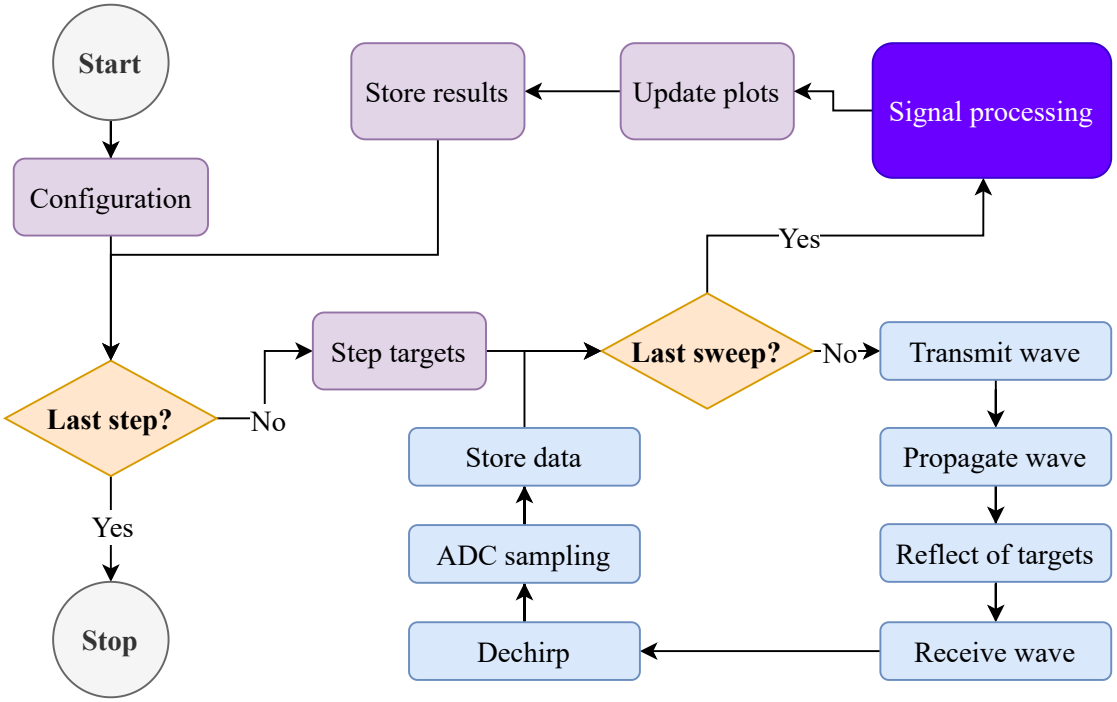


Figure 5.2: Simulation flow diagram of MATLAB Phased Array System Toolbox simulation where data is processed as a matrix as opposed to sweep-by-sweep

The trails behind each target were used to indicate the distance the target has moved from the starting point.

The radar hardware was modelled using the data sheet parameters presented in Tab. 4.1. These parameters were used to instantiate PAST transmitter and receiver objects from the MATLAB PAST [71]. The desired waveform was created using the `FMCWWaveform` System Object using the specified bandwidth, ramp period, and sample rate. The uRAD ADC sampling rate of 200 kHz set the maximum unambiguous range according to the Nyquist criterion (Eq. 2.17): $f = 100$ kHz is the maximum detectable beat frequency. This frequency corresponds to a maximum unambiguous range of 62.5 m (Eq. 2.9). The antenna was configured using the antenna gain presented in Tab. 4.1. The transmitter and receiver preamplifier were then modelled based on peak power and gain parameters. The noise figure was set to 10 dB as is typical for a modern receiver [72].

Initially, the simulation assumed isotropic transmit and receive antennas and the same gain was applied to both the transmitter and receiver [73]. This approach allowed for

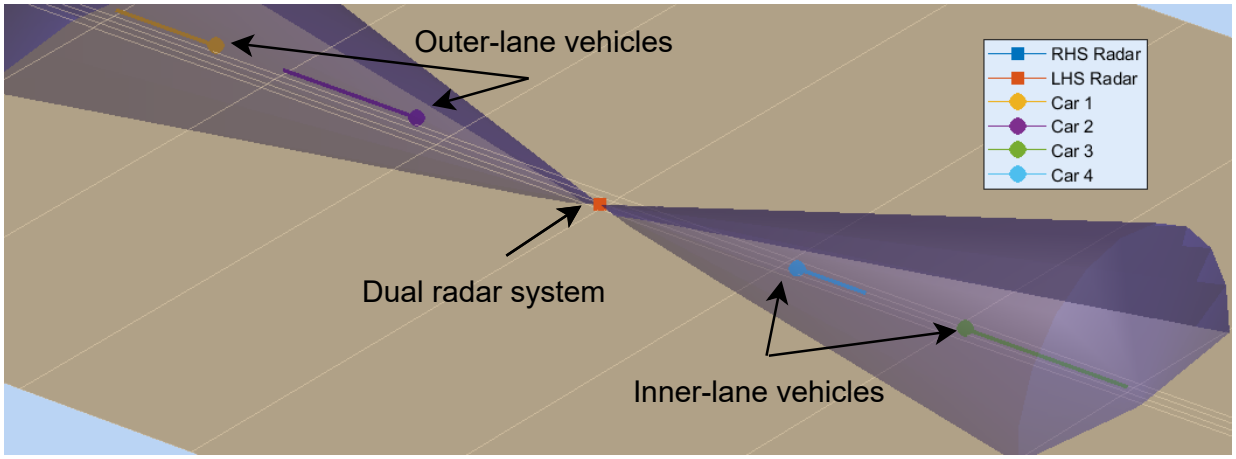


Figure 5.3: Phased Array System Toolbox graphical representation of the intersection scenario

the simulation and testing of the processing algorithm. However, it was not effective for simulating the dual radar scenario since targets on both sides of the antenna were present in the received signal of each antenna. The simulation was adapted to incorporate the beam pattern of the uRAD USB v1.2 patch antenna.

Though the PAST monostatic radar transceiver object abstracted most of the simulation processes, debugging the erroneous output spectra became challenging. The difficulty was due to the complexity of low-level MATLAB, MATLAB being a closed-source package, and a lack of antenna simulation knowledge. It was decided that testing the processing algorithm on real radar data would be more time-efficient than getting a working patch antenna simulation since the hardware was already available at the University of Cape Town. Further efforts to improve simulations were deemed unnecessary. The effect of not simulating the antenna correctly would not affect the processing algorithm design, as it would only affect the SNR measurement at the edges of the radar beam. The SNR did not drop off as the target moved out of the radar beam. As shown in Fig. 5.1a, only the nearest ranges are not illuminated by the beam. Since this area is to be monitored by short-range, collision warning sensors, correct radar simulation in the region is not required.

5.2 Simulation Outputs and Analysis

This section presents and analyses the results obtained from the ADT and PAST MATLAB simulations. The range and velocity measurements were compared to ground truth values. The average deviation between the ground truths and radar measurements was used to assess the radar measurement accuracy. Both the ADT and PAST simulation measurements were plotted using scatter plots with lines representing the ground truth values. The presented results focus on the outer-lane masking scenario—Scenario D in Tab. 5.1. In addition, scenes rendered using both toolboxes are presented.

5.2.1 Automated Driving Toolbox Results

For scenarios A, B, and C in Tab. 5.1, the simulation outputs were as expected and met the project requirements described in Tab. 3.2. Figures 5.5 and 5.6 show the effect of an inner-lane vehicle (Car 1) masking the detection of a vehicle travelling along the outer lane (Car 2). A birdseye plot of the scenario is shown in Fig. 5.4. Masking is due to the inability of the radar wave to penetrate a metal car target. The straight lines are the ground truth values. The scatter points are due to cars being distributed targets—as opposed to the ideal point target—and the radar measurement accuracy. It is believed that multiple points per time instance were due to the use of the theoretical range resolution of $\Delta R = \frac{3 \cdot 10^8}{2 \cdot 240 \cdot 10^6} = 0.625$ meters being less than the length of a car.

As shown in Fig. 5.5, the velocity measurement deviated from the ground truth as the range to the target decreased. This is due to an increase in the angle of arrival (the angle between radar boresight and the target) since the radar is offset from the centre of the road (Fig. 4.6). A larger angle between the actual and radial velocities results in a larger difference between the two values. The proposed trigonometrical correction to this measurement error was presented in Section 4.3.3.

The average absolute difference, referred to as the average deviation, provided a measure of the accuracy in range and velocity. The range accuracy of 0.609 m was higher than the datasheet minimum accuracy of 0.04 m. The velocity accuracy improved after correction based on the angle of arrival (Section 4.3.3). The uncorrected measurement gave an average

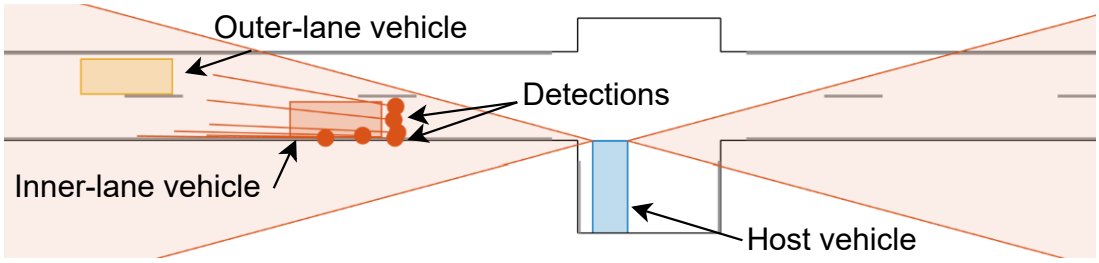


Figure 5.4: Birdseye view of the inner-lane car masking the outer-lane car

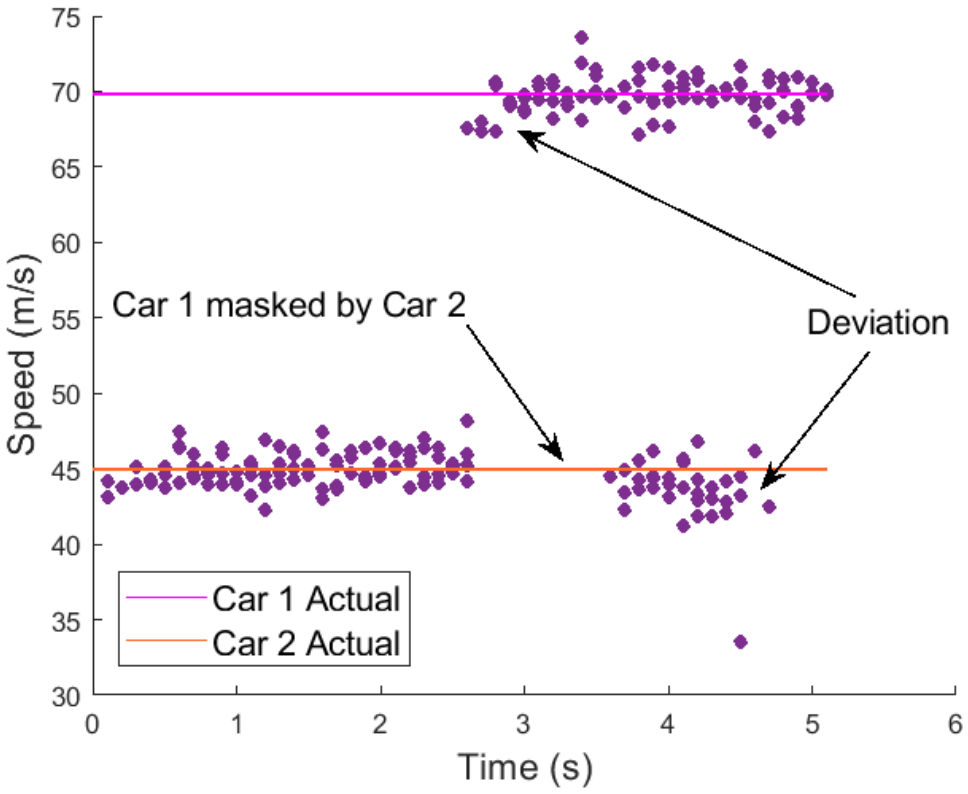


Figure 5.5: ADT simulation results showing radial velocity measurements vs. time for the scenario where the inner-lane vehicle (Car 1) masks the outer-lane vehicle (Car 2)

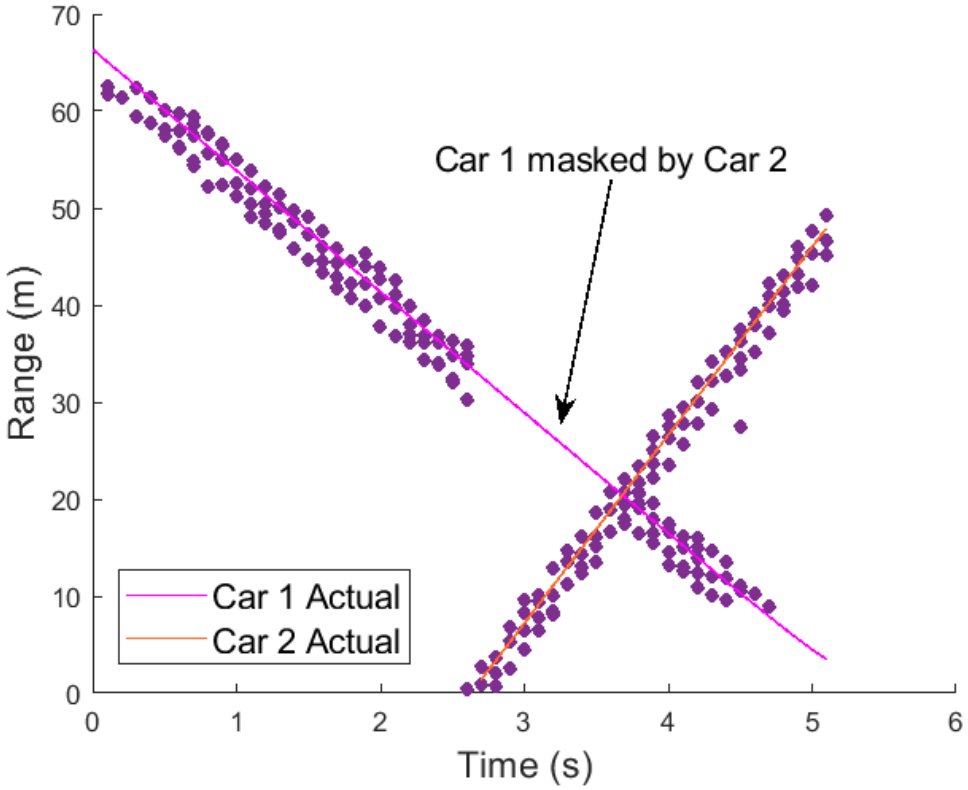


Figure 5.6: ADT simulation results showing radial range measurements vs. time for the scenario where the inner-lane vehicle (Car 1) masks the outer-lane vehicle (Car 2). The deviation due to angle-of-arrival is indicated

deviation of 0.26 m/s which was close to the datasheet value of 0.25 m/s. These values meet the respective requirements defined in Tab. 3.2. Unfortunately, the MATLAB processing algorithms used to obtain these results were not publicly available.

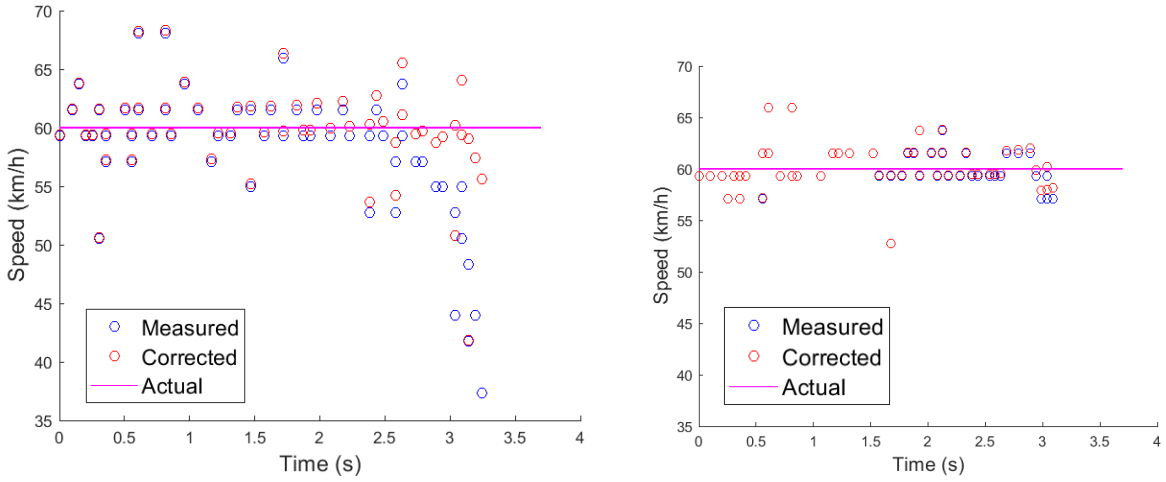
5.2.2 Phased Array System Toolbox Results

The proposed processing algorithm, presented in Section 4.3.6, was used to process the simulated data. The scenarios listed in Tab. 5.1 were simulated using vehicles travelling on either lane, the outer (left to right) and inner (right to left) lanes, at various speeds. Results showed that cars could be detected at the maximum expected range and when travelling at the maximum expected velocity. The accuracy of the measurements for various speeds

and ranges was found to be within the technical requirements (Tab. 3.2) of the project. Scenario D in Tab. 5.1 was explored further as it was found to be the scenario in which system performance would be most impeded.

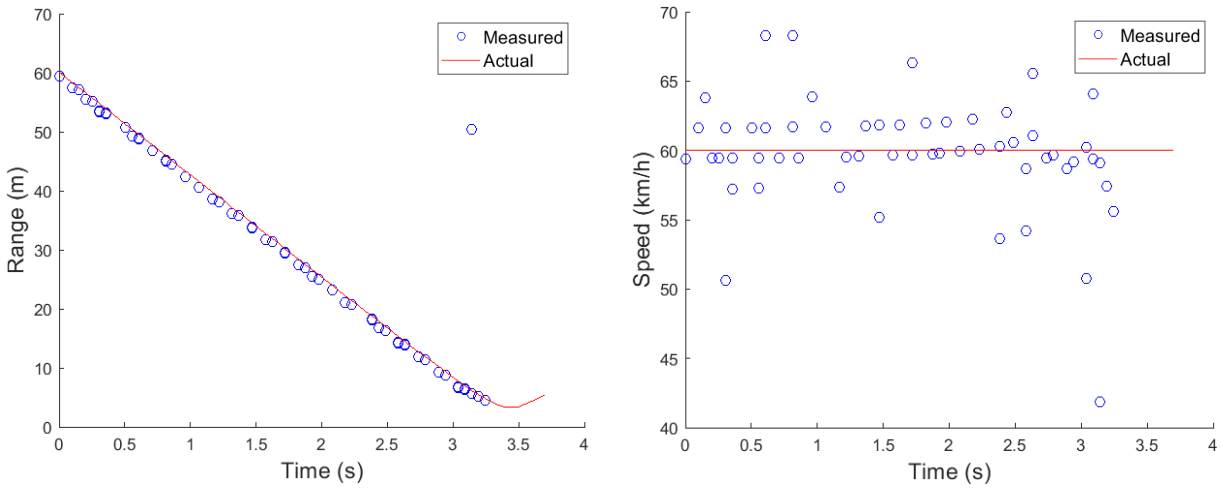
Figure 5.8 provides a reference point for the detection of the outer-lane vehicle when no inner-lane vehicles were present. Comparing Figures 5.7a and 5.7b, it is clear that a smaller offset of the target from radar boresight results in a lower deviation in the velocity measurement. Additionally, the increase in angle of arrival as the target's range decreases has less of an effect on the resulting measurement. Figure 5.9 shows the case where three inner-lane cars mask the outer-lane vehicle. The results indicate that the inner-lane vehicles have been filtered out as expected. Though there are more missed detections of the outer-lane vehicle compared to when there was no masking (Fig. 5.8), there remain sufficient detections for producing a stable time of arrival estimation of said vehicle.

Figure 5.10 is the same case as Fig. 5.9 but two of the inner-lane vehicles are travelling such that they lie within the range gate generated by the processing algorithm (Section 4.3.6). This resulted in a false track as shown in Fig. 5.10a. Since the false track has an increasing range (target moving away), it can simply be filtered out. The number of detections were sufficient for tracking the outer-lane vehicle.



(a) Left radar, target offset 3.3 m from boresight (b) Right radar, target offset 1.1 m from boresight

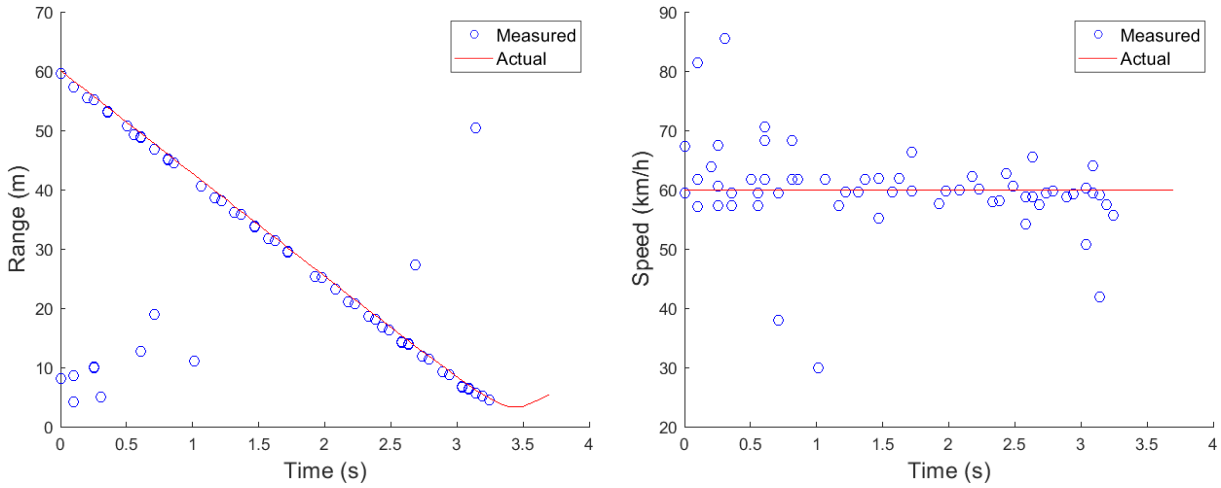
Figure 5.7: Plots showing the actual, measured, and angle-corrected simulation speed measurements vs. time obtained by the left and right radars for a target approaching at 60 km/h parallel to radar boresight



(a) Actual and measured range vs. time

(b) Actual and measured speed vs. time

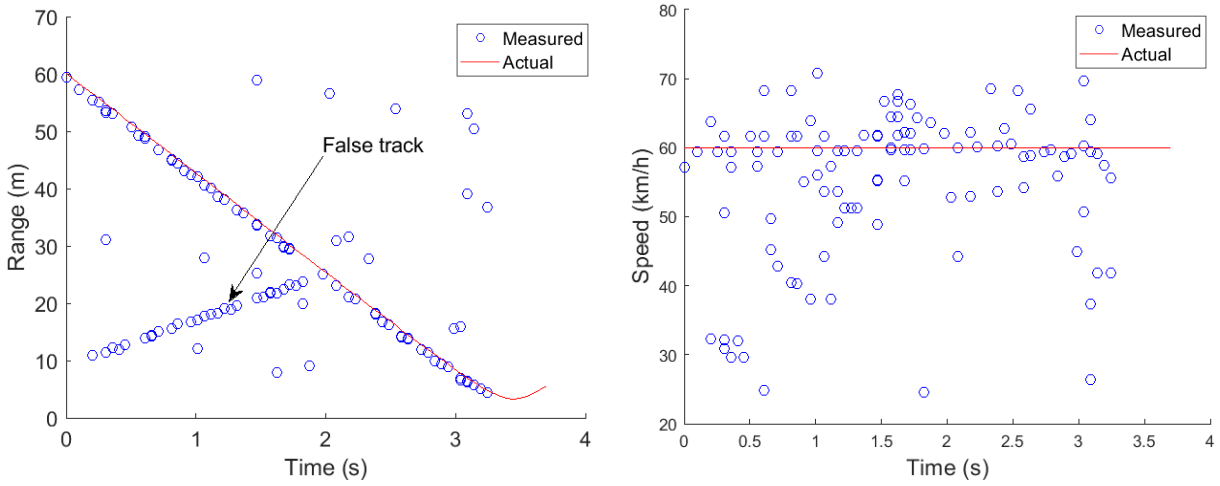
Figure 5.8: Plots showing radar measurements by the left radar for an outer-lane target approaching at 60 km/h offset 3.3 m from radar boresight



(a) Actual and measured range vs. time

(b) Actual and measured speed vs. time

Figure 5.9: Plots showing radar measurements by the left radar for an outer-lane target approaching at 60 km/h offset 3.3 m from radar boresight where three inner-lane cars inhibit detection thereof



(a) Actual and measured range vs. time

(b) Actual and measured speed vs. time

Figure 5.10: Plots showing radar measurements by the left radar for an outer-lane target approaching at 60 km/h offset 3.3 m from radar boresight where three inner-lane cars inhibit detection thereof and two of the inner-lane cars lie within the same range gate

The average deviation between the measured and actual range using the PAST simulation was 0.8 m, meeting the accuracy requirement. The expected accuracy, based on the FFT spacing $\Delta f = 195.3$ Hz, was 0.12 m. Since the car target was not an ideal point target, a larger average deviation was expected compared to the theoretical value. A slight improvement in velocity estimation was obtained when accounting for the angle of arrival, with the average deviation being 1.87 m/s as opposed to the 2.40 m/s before angle correction. The datasheet accuracy values are far off from the simulated ones, which more closely resemble the resolution values instead. To date, no explanation was provided by uRAD as to how the datasheet accuracy values were obtained.

5.2.3 Summary of Simulation Results

The simulation results indicate that the system was effective for intersection collision prediction. The uRAD USB v1.2 radar model was capable of detecting cars at a sufficient range, with an acceptable level of accuracy in measuring both range and velocity. Based on the plotted results, the PFA was low enough that false tracks were only produced when

targets were closely spaced. As shown in Fig. 5.10, the track has an opening range and can therefore be filtered out. Though the masking scenario hampered the system's ability to determine turn safety, the PAST simulations indicate that this scenario did not have a significant impact on the detection of outer-lane vehicles. The simulations provided a solid basis for further work using a dual uRAD USB v1.2 radar system for low-cost intersection collision prediction.

Chapter 6

Implementation

This chapter discusses the implementation of the roadside unit test rig and the Python scripts for data capture, real-time processing, and prototype configuration and control. It begins with a discussion of the acceptance tests and the results thereof. Scripts for acquiring and processing data are then described, followed by the integration of the camera system for verification of radar measurements post-capture. The implemented real-time processing script is presented along with verification by way of real-time plotting. Finally, the construction and implementation of the three test rig iterations are discussed.

6.1 Acceptance Testing

Acceptance testing was performed to assess the system's ability to meet the respective requirements summarised in Tab. 3.2. The hardware test plan, summarised in Tab. 6.1, was used to assess whether the hardware could meet the specifications provided in the uRAD USB v1.2 datasheet [65]. Additionally, the processing algorithm was assessed to ensure the user requirements shown in Tab. 3.1 could be met.

6.1.1 Hardware Acceptance Testing

This section presents the outcomes of the hardware test procedures presented in Tab. 6.1. For testing the maximum detectable range (HT1, Tab. 6.1), it was found that cars were

Table 6.1: Hardware Test Plan

ID	Requirement	Description
HT1	maximum range	Place target at the maximum range and verify radar detection
HT2	maximum velocity	Drive a car at the maximum velocity rating and compare to radar estimate
HT3	update rate	Measure rate at which radar produces data
HT4	range accuracy	Place target at known ranges and compare to radar estimate
HT5	velocity accuracy	Move target at known speeds and compare to radar estimate

detected at the expected range of 62.5 meters (based on the FMCW waveform design). In the uRAD USB v1.2 datasheet, the maximum range for the detection of cars is 75 m, which suggests the power output is sufficient for maintaining a high PD at 62.5 m. Unfortunately, the datasheet did not provide values for PFA and PD. Since these probabilities depend on the target's RCS, they would vary for cars of different shapes and sizes. As a result, these values were not calculated. Testing maximum detectable velocity (HT2, Tab. 6.1) was performed by driving a vehicle at 70 km/h—the maximum speed expected by the processing algorithm (Section 4.3.6) past the radar placed parallel to the road. Results from the controlled experiment, discussed in Section 7.1.4, indicate that requirement HT2 was met.

For test HT3 in Tab. 6.1, it was discovered that raw data was captured at an average rate of 93.4 sweeps (both up and down chirps) per second for the triangle modulation mode. Section 6.2.1 describes the code used to obtain the update rate. The required update rate (UR1, Tab. 3.1) was calculated by requiring a detection to be made in each frequency division, following the processing described in Section 4.3.5. Using the maximum range of 62.5 m and the maximum velocity of 70 km/h, the time of arrival of such a target would be around 3.2 seconds. Since there are 16 bins, the required update rate is $\frac{16}{3.2} = 5$ sweeps per second.

Given that the period of the triangle FMCW was 2 ms and the update period was roughly 10 ms, the radar hardware effectively decimated the number of received sweeps by a factor of five. It was hypothesised that the reduced raw data update rate was due to delays in the front-end circuitry such as the FIR filter N-tap delay, and operations by the onboard microcontroller such as capturing and forwarding data from the radar module to the host PC. Since hardware schematics were not available due to being company property, a holistic analysis was not possible.

For test HT4 (Tab. 6.1) a calibration experiment was performed by placing a sheet of aluminium 3 meters from the radar. The details of this experiment are presented in Section 7.1.1 along with experimentation using increasing amounts of zero padding. Increasing the amount of zero padding allowed for an improved range measurement accuracy of a single target at the cost of an increased FFT processing latency. Thereafter, a calibration factor was calculated as shown in Eq. 7.1 after performing a range calibration experiment, discussed in Section 7.2. The calibration factor was used to adjust all beat frequency measurements, based on the direct proportionality of the beat frequency and the range given by Eq. 2.9. Based on the results of the controlled experiments (Section 7.2.2) the accuracy met the respective user and technical requirements—TR6 and UR3 from Tables 3.2 and 3.1, respectively.

6.1.2 System Acceptance Testing

System acceptance tests relate to assessing if the proposed system met the technical requirements (Tab. 3.2). The various resolutions (range, velocity, azimuth, elevation) were not tested directly, as these varied for different targets and scenarios.

It was initially thought that a range resolution worse than 4 m would not adversely affect the results, as cars travelling close together travel at the same speed and cannot pass the 'lead' car when travelling along a single lane. If an overtake scenario occurred, the overtaking vehicle became the target of interest. It was found that the spectral division algorithm, described in Section 4.3.5, benefited from a finer resolution than 3.9 m as the narrower return echo would not bleed into adjacent frequency bins. Results from uncontrolled testing, presented in Section 7.2.4, show that relatively closely-spaced vehicles—vehicles that are

roughly 3.9 meters apart as required by the processing algorithm (Section 4.3.5)—were resolved in range (Section 7.2.1) by the proposed processing algorithm to meet the defined requirement. Range accuracy (TR6, Tab. 3.2) is discussed in Section 6.1.1.

The velocity accuracy (TR7, Tab. 3.2) was set by the frequency spacing based on the amount of zero-padding applied (Section 2.7.5). Using the frequency spacing of $\Delta f = 195.3125$ Hz and substituting into Eq. 2.16 for $f_{beat}^{up} + f^{down}$, we obtain the minimum detectable Doppler shift $f_D = 97.67$ Hz. Using Eqn. 2.3, the velocity measurement accuracy is $v_{acc} = 0.61$ m/s. The velocity resolution (TR5, Tab. 3.2) and angular resolution (TR4, Tab. 3.2) were not assessed as these do not apply to the monitoring of a single lane of traffic. Since the uRAD USB v1.2 radar was not capable of beamforming, no angular resolutions are presented in the datasheet.

6.2 Software Integration

This section begins with a description of the various scripts developed for experimentation and testing. Several methods for integrating Python data acquisition and MATLAB signal processing were explored. Thereafter, a Python-based solution was implemented, which provided independence from the MATLAB closed-source algorithms contained in the Phased Array System Toolbox. Dual cameras and dual radars were integrated following the implementation of signal processing in Python. Finally, processing optimisations such as multi-threading were added and real-time plotting scripts were developed to provide a visual means for system debugging.

6.2.1 Raw Data Acquisition

Both the provided SDK and the example raw data capture script were optimised to realise the maximum update rate of the radar module. In this context, the update rate refers to the sweeps captured per second. Portions of the SDK were removed such that only I/Q data could be produced and only the triangle FMCW waveform was available. This reduced the number of evaluated conditional statements when data was requested from the hardware. It was found that this optimisation did not affect the update rate. This was due

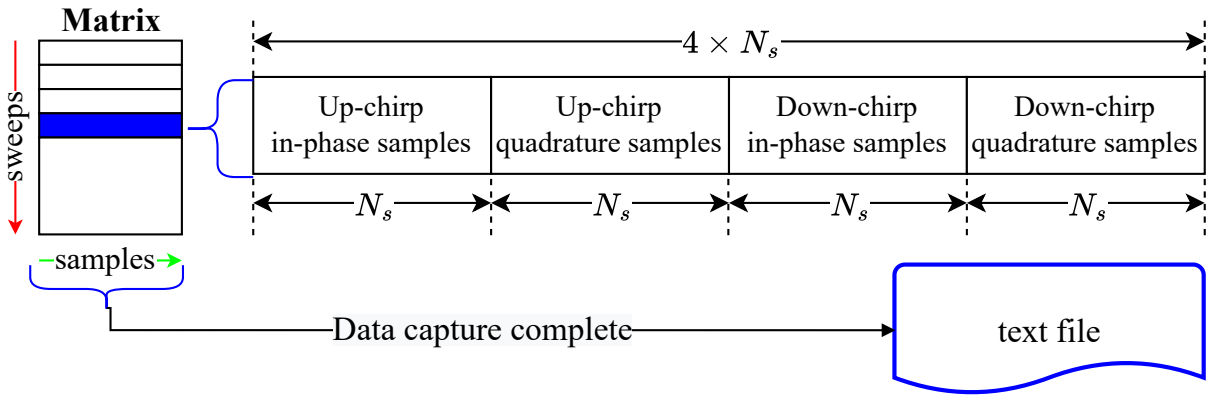


Figure 6.1: Raw data packet structure

to the hardware capture rate being the limiting factor when no processing was performed in software.

The raw data capture script was used to write uRAD I/Q data to a text file. Received sweeps were appended to a Python list, after which the list was written to a text file and transferred to a computer over SSH for analysis and offline processing. This offered a higher average update rate of 92.4 sweeps/second compared to the 85 sweeps/second obtained using the example script, which wrote sweeps as strings to a text file as soon as they were received. For the triangle FMCW waveform capture, the data packet structure followed Fig. 6.1. Listing 5 shows the code used for obtaining the average update rate.

6.2.2 MATLAB-Python Integration

The MATLAB Support Package for Raspberry Pi Hardware [74] allowed for the execution of MATLAB code on the Raspberry Pi. Effectively, the Pi became a sensor node connected to a PC running MATLAB by installing certain packages or converting to the MathWorks Raspbian operating system (OS). Unfortunately, the OS could not be configured to connect to the internet which prevented the installation of the required processing packages. In addition, this option was suboptimal as it made the system dependent on MATLAB and therefore a PC. Consequently, no further work on this approach was undertaken.

The MATLAB API Engine for Python allowed for calling MATLAB functions inside the Python script used for interfacing with the radar module. Naturally, this required a

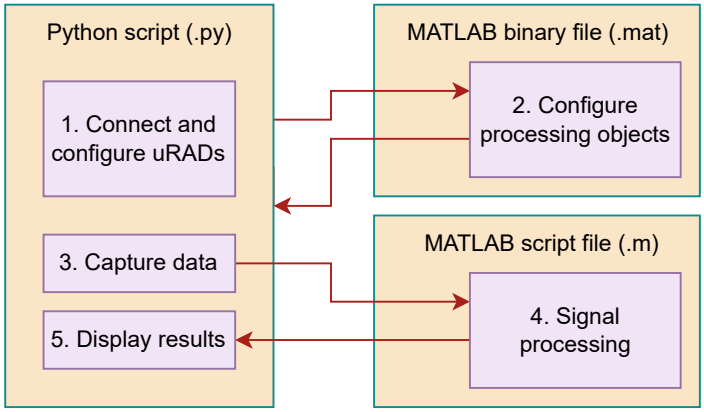


Figure 6.2: Block diagram of MATLAB Python API Engine usage for real-time processing

PC with MATLAB installed or the MATLAB Online application to be accessed via an internet browser. Initially, a Python script was designed to pass parameters to MATLAB functions which would generate the required objects, such as the CFAR object. The radar data was inserted into the MATLAB workspace, in the form of a Python dictionary, before a MATLAB processing script was run on this data. This method was improved by creating a MATLAB script to generate all the required processing variables and objects, which were then stored as a .mat file. This shortened the initialisation time, though parameters and objects were not configurable from within the Python script. Figure 6.2 shows how the MATLAB and Python scripts were integrated: Python was used for interfacing with the radar hardware as well as plotting or storing results while MATLAB was used for processing. After the radars were configured, the pre-instantiated MATLAB objects, such as the CFAR detector, were loaded into the MATLAB workspace. Radar data was then captured and passed to the MATLAB processing script. The results were passed back to Python for display.

For the final system, it was decided that combining MATLAB and Python was not suitable for several reasons. Firstly, the dependence on MATLAB prevented the system from being freely usable and forced the processing to be executed on a PC. Though MATLAB is generally faster than Python, Python is superior at parallel processing [75]. Since MATLAB required a PC for data acquisition, a laptop with four USB ports was needed to capture the data from both radars and cameras used in the prototype system (Section 4.4.3). Most

importantly, MATLAB is an expensive tool that would hamper the commercialisation of the system. In contrast, Python is a free-to-use language with many free-to-use libraries. A Python implementation allowed for the use of a Raspberry Pi Model 4 operating independently of a PC.

6.2.3 Pure Python Implementation

Fortunately, most of the signal processing sub-algorithms were found in existing Python libraries. Specifically, matrix and array operations were carried out using the NumPy library and the SciPy library was used for generating the Taylor window function coefficients. Unfortunately, the four CFAR algorithms available in MATLAB could not be sourced in Python, so a custom Python library was developed.

The custom CFAR library was placed in a public GitHub repository (Appendix B). Considerable effort was expended in handling edge samples—samples near the beginning and end of a sweep with insufficient training cells on a particular side. Initially, only samples with sufficient training and guard cells on either side were considered, allowing for verification of the CFAR algorithms. Subsequently, the edge case was handled. Listing 6 shows how this was achieved. Whenever insufficient training cells were available on a particular side of the cell under test (CUT), cells adjacent to the training set on the opposite side were used to fill the remaining positions. Figure 6.3 shows how the left-hand side edge cases were handled. As the CUT is moved from left to right, fewer samples to the right of the right-hand side training set were used to fill the left-hand side training set.

Code for the CA-, GOCA-, SOCA-, and OS-CFAR algorithm was developed. A performance comparison of these algorithms is presented in Section 7.1.2). Finally, the remainder of the MATLAB processing algorithm was converted to Python code. predominantly required converting matrix operations to operations on single sweeps, allowing computations to be performed in real time.

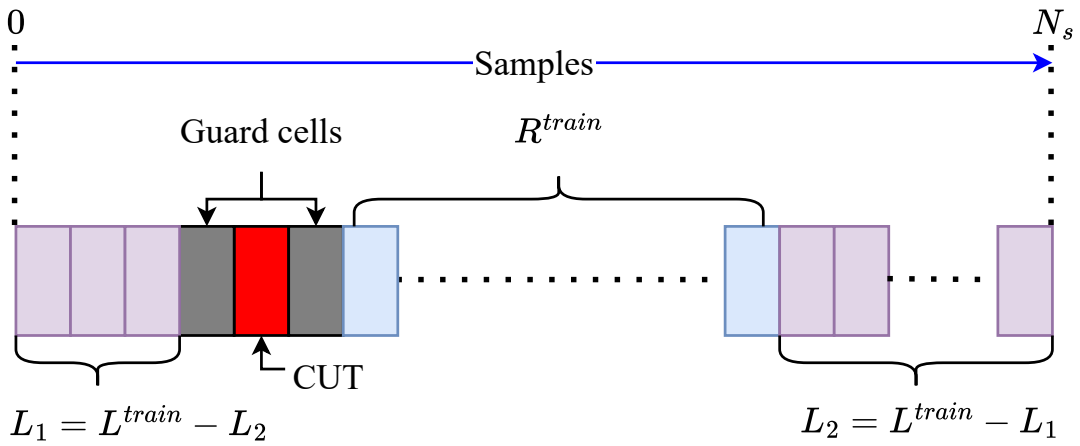


Figure 6.3: Edge case handling for CFAR detection: when the number of training cells on the left side is insufficient, cells are selected from the right of the right side training cells

6.2.4 Software Thread Variations

The frame synchronous version allowed for storing synchronised frames from each of the four sensors. It involved polling each sensor sequentially and storing data in arrays. Once the desired number of sweeps was captured, the data was written to text files. Since an equal number of frames was captured from each device, offline plotting was straightforward: a frame from each sensor was plotted in a grid layout for a specified duration before moving to the next frame. Initially, the update rate was limited to that of the slowest sensor—the uRAD Raspberry Pi v1.1—which averaged 50.4 I/Q sweeps per second. It was hypothesised that a multi-threaded approach, where one or more sensors were run on a separate thread, would improve the update rate. The difficulty in synchronising the offline plotting of the different sensor types and models when multi-threading made the single-threaded approach preferred for performance validation. It was later uncovered that when running the multi-threaded program with two uRAD USB v1.2 radars, the radars acquired approximately three times as many sweeps per video frame. The video frames and radar sweeps were then synchronised by plotting three radar sweeps per video frame.

The four-thread configuration was believed to offer the best performance for raw data capture when running all sensors simultaneously on a single Raspberry Pi 4B. The Pi 4B has a quad-core processor, so each of the four sensors was expected to run on a separate core.

However, it was found that Python was not capable of such parallelism [76]. Nevertheless, there were gains to be made by implementing threaded operations that involved communication with sensors [76]. It was found that running each thread for a specific duration—as opposed to specifying a set number of loop iterations—prevented certain threads from finishing before others. Rather, certain sensors would obtain more data within the specified period. Implementation involved a `while loop` in each thread gated by the maximum time the loop should run. It was found that the real-time plotting of the video from each camera could not be threaded, so a three-thread variation was created. The cameras were run on a single thread when real-time plotting was required.

6.3 Real-Time Plotting

Real-time plotting offered a visual representation of the data and allowed for easy identification of any erroneous behaviour. Specifically, the computed frequency spectra and CFAR threshold were plotted and compared to the observed scene. Real-time plotting had the disadvantage of a severely reduced update rate, which was caused by the added computational burden of rendering and streaming the graphs. The final version was stripped of all graphical elements and simply printed the real-time results to the CLI.

Python offered two plotting libraries: Matplotlib and Pyqtgraph. The former was simpler to use, whereas the latter offered increased performance. After difficulties in plotting data from multiple sensors in a suitable layout using PyQtGraph, it was decided that Matplotlib was sufficient. Since real-time plotting was used for qualitative verification and debugging of the Python-implemented processing algorithm, optimising performance was not crucial. However, certain optimisations were required to make the Matplotlib plotting program usable for verification purposes. Listing 2 shows the code used to speed up the Matplotlib real-time plotting of data from multiple sensors. The use of blit (Block Transfer) allowed for further optimisations in plotting. All graphic elements that did not change from the previous frame were rendered into a background image, reducing the amount of data being drawn. Only elements that changed were updated [77].

The frequency spectra of the up and down chirps of each radar were plotted in a grid

Listing 2 Matplotlib optimisations for faster plotting

```
import matplotlib as mpl
mpl.rcParams['path.simplify'] = True
mpl.rcParams['path.simplify_threshold'] = 1.0
mpl.rcParams['toolbar'] = 'None'
import matplotlib.style as mplstyle
mplstyle.use(['dark_background', 'ggplot', 'fast'])
```

layout. In a separate figure, video footage from each camera was displayed, which allowed for a comparison to the radar data in real time. Many options for streaming generated plots to a PC were attempted: firstly, streaming using SSH X forwarding was implemented, which streamed only the plot window. This was found to severely reduce the plot update rate and was deemed unusable—the update rate was below one sweep per second, failing to meet the requirement of 5 sweeps per second (calculated in Section 6.1.1). The next method involved saving data to image files (.jpeg) after each capture which were viewed over SSH through Microsoft Visual Studio Code by clicking on the .jpeg file and observing the changing image. This method offered much better performance, though the update rate of under two sweeps per second remained suboptimal. The final method was to use a remote desktop connection between the Pi and the PC. This was achieved using a VNC (Virtual Network Computing) viewer, RealVNC, which streamed data between two network devices via the Remote Frame Buffer protocol [78]. TightVNC was used to further improve the update rate, as it allowed for efficient encoding of streamed data and for scaling the streamed desktop size [79]. Combining the TightVNC software with the plotting optimisations mentioned above resulted in a sufficient update rate for real-time visualisation of the CFAR operation and received signal spectra.

6.4 Test Rig Implementation

This section describes the implementation of the three test rigs proposed in Section 4.4. Firstly, the integration of the various peripherals is discussed. The implementation of the intermediate test rigs is then presented. The roadside prototype test rig is presented last,

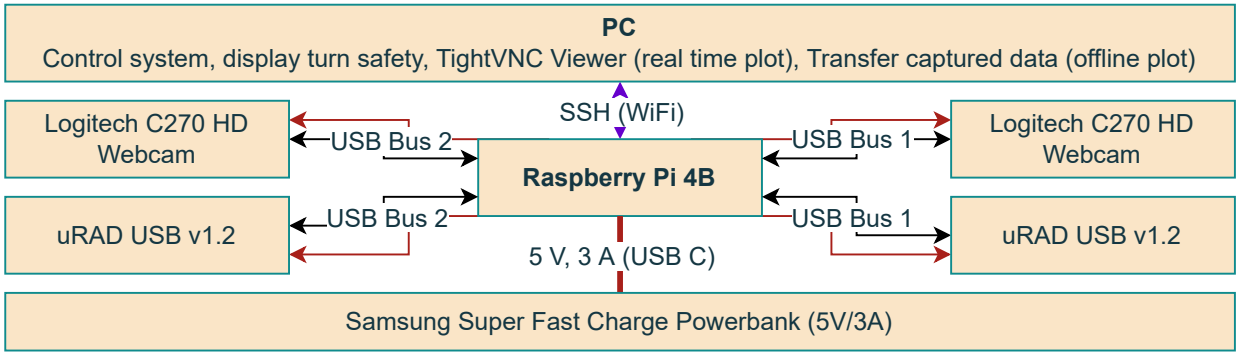


Figure 6.4: Block diagram of the implemented system

including a bill of materials (BOM) shown in Tab. 6.2.

6.4.1 Peripheral Hardware Integration

The OpenCV Python library was used to record video footage from two Logitech C270 webcams. The frame dimensions were selected to minimise processing and storage requirements while ensuring video quality was sufficient for visual inspection. An investigation into the correct combination of codec and container formats was conducted. It was found that certain hardware combinations required specific codec-container pairs. After testing many combinations, it was found that the H264 encoding and the AVI (Audio Video Interleave) container worked for the Raspberry Pi 4B and Logitech C270 webcams. Another finding was that the second camera was referenced by index two despite the first camera being referenced by index zero. This was noteworthy since the error message simply stated that the camera could not be detected. Listing 3 presents the code used to configure the cameras for video capture.

Initially, only a uRAD Raspberry Pi v1.1 and uRAD USB v1.2 was made available by the University of Cape Town. The older uRAD Pi model required the use of the older uRAD SDK and communication took place over SPI. It was found that the uRAD Raspberry Pi v1.1 had a drastically slower update rate of approximately 30 sweeps/second, around three times slower than the 93 sweeps/second produced by the uRAD USB v1.2. The reduced performance was attributed to the older hardware and SDK. Though uRAD did not provide the list of components due to intellectual property, the improved hardware can be identified

Listing 3 Code for configuring video capture for two cameras using the Python OpenCV library

```
# Open capture device for each camera
cap1 = cv2.VideoCapture(0)
cap2 = cv2.VideoCapture(2)

# set image dimensions
cap1.set(3, 320)
cap1.set(4, 240)
cap2.set(3, 320)
cap2.set(4, 240)

# create VideoWriter object with file name, codec,
# Frame rate and frame dimensions
fourcc = cv2.VideoWriter_fourcc(*'X264')
out1 = cv2.VideoWriter('rhs_vid_'+now+'.avi',fourcc,20.0,(320,240))
out2 = cv2.VideoWriter('lhs_vid_'+now+'.avi',fourcc,20.0,(320,240))
```

by inspection of the circuitry: the larger microcontroller present on the board likely operated at a higher clock frequency. The newer SDK performed fundamentally differently, as can be seen in Listing 4 which contrasts the methods for acquiring data using uRAD SDK v1.0 and v1.1. The reduced performance along with the lack of mounting holes on the uRAD Raspberry Pi v1.1 provided strong motivation for the purchase of a second uRAD USB v1.2. All of the scripts were then modified to initiate two serial communication links, one for each uRAD, and were updated from SDK v1.0 to SDK v1.1 for the second radar. Configuration of the serial links between the uRADs and the various processing systems is presented in Section 4.1.4. Each uRAD radar was then run using the script variations outlined in Section 6.2.4. A 'turn unsafe' result from either radar would be conveyed to the driver as per requirement UR2 (Tab. 3.1). The associated circuitry (LEDs, buzzers, etc.) was not implemented since the test rigs were controlled via CLI (command-line interface), which was used to display outputs. Since the user interface falls under product design as opposed to research and development, it was not implemented.

Listing 4 Code for data acquisition using uRAD SDK v1.1 compared to SDK v1.0

```
# SDK 11 data acquisition:
return_code, results, raw_results = uRAD_USB_SDK11.detection(ser)
I_sdk11 = raw_results[0]
Q_sdk11 = raw_results[1]
# SDK 10 data acquisition:
uRAD_RP_SDK10.detection(0, 0, 0, I_sdk10, Q_sdk10, 0)
```

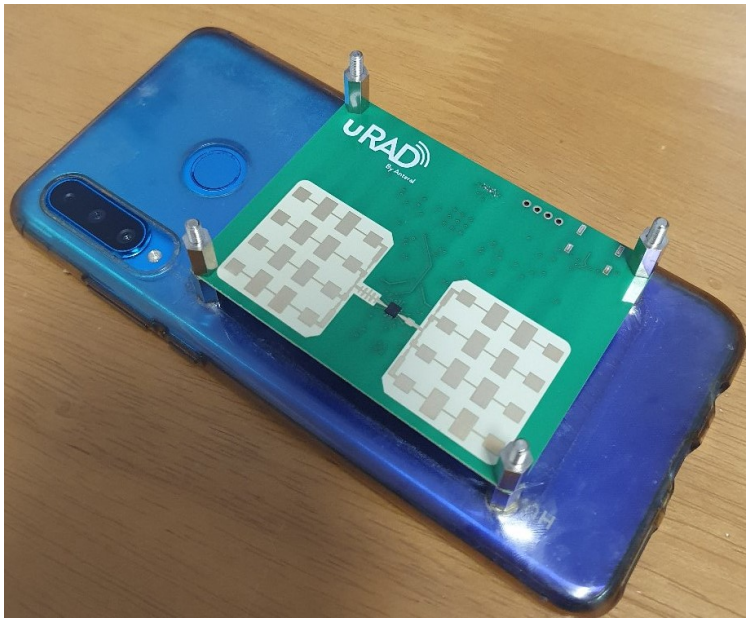


Figure 6.5: Photograph of the uRAD USB v1.2 and phone camera test rig

6.4.2 Intermediate Test Rigs

The phone camera and uRAD USB v1.2 rig involved fastening the radar to a plastic phone cover. Figure 4.12a shows a diagram of the phone and uRAD test rig, where the uRAD was fixed to the plastic phone cover as shown in Fig. 6.5.

The radars were installed on the sides of a cardboard box, which was then placed inside a plastic tub. For the uRAD Raspberry Pi v1.0, both the Pi and the radar had to be mounted to the side of the inner cardboard container. The cameras were placed on the left and right rims of the tub and the battery was placed inside. A birds-eye photograph of the intermediate test rig is shown in Fig. 6.6.

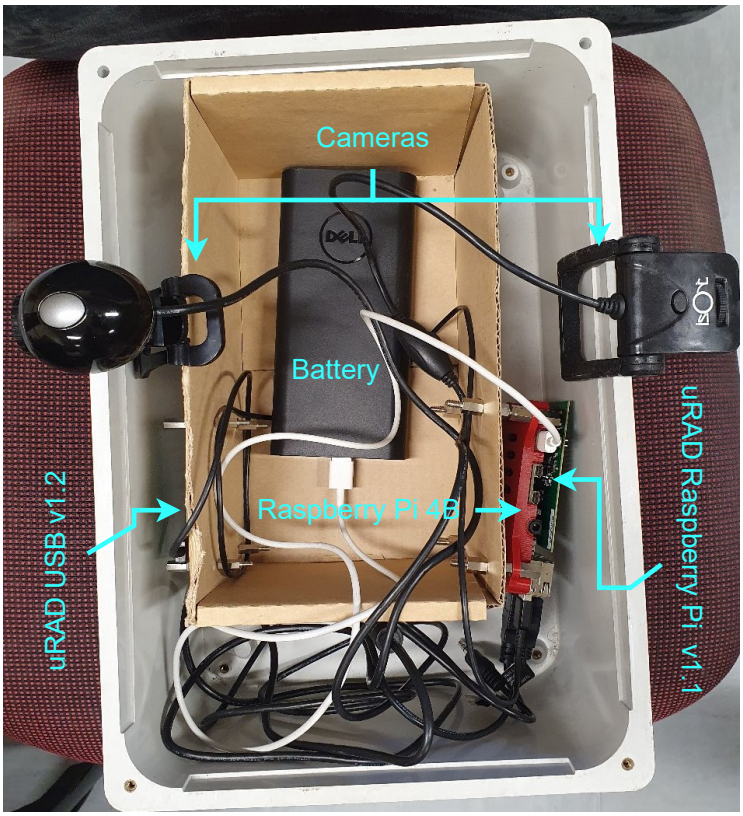


Figure 6.6: System-in-a-tub laboratory test rig using the uRAD Raspberry Pi v1.0

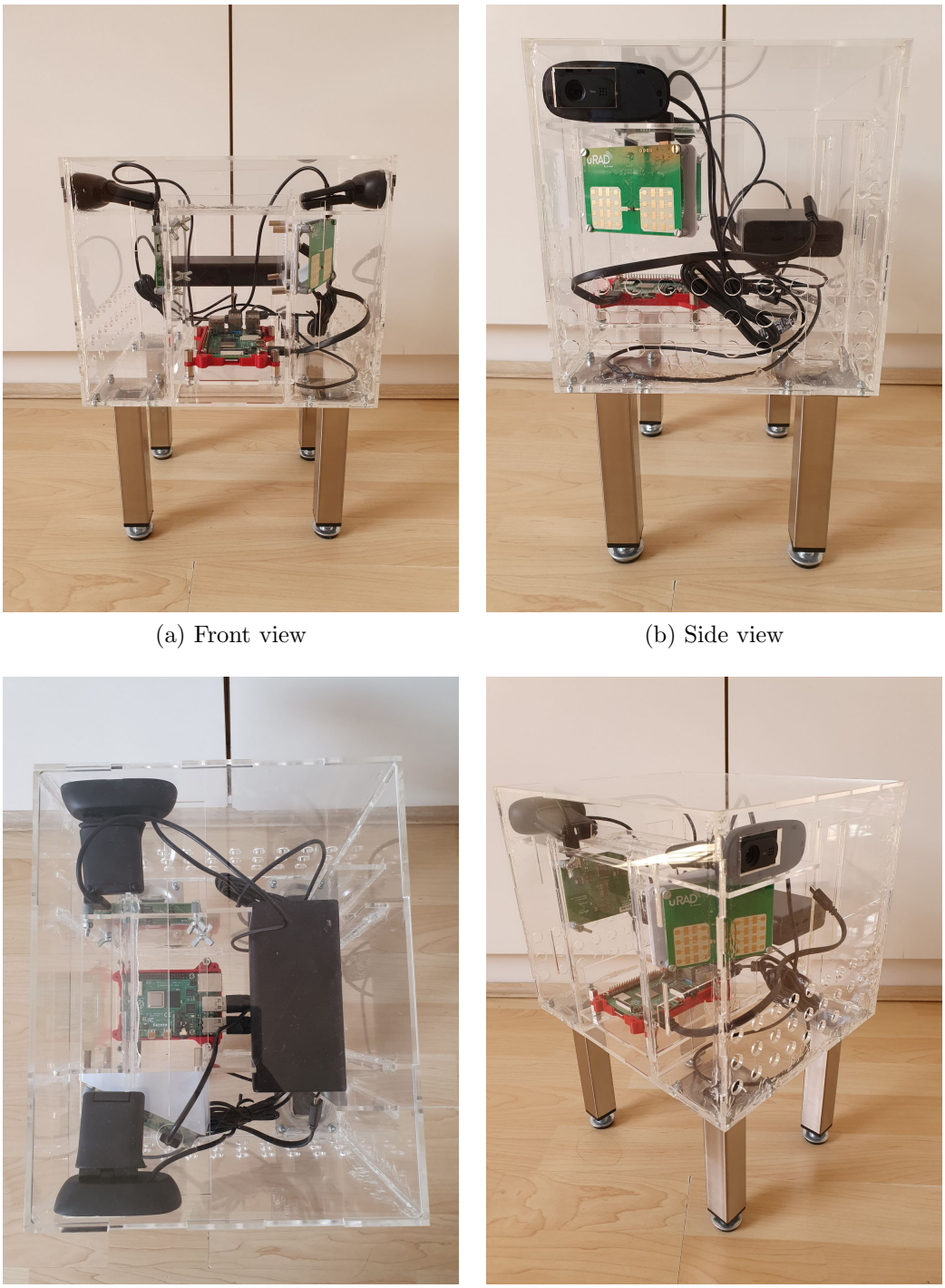
6.4.3 Roadside Unit Test Rig

Once the design was completed, PDF files of each part were generated and used to laser cut perspex sheets. The process of laser cutting required formatting and placing of components optimally such that the minimum amount of acrylic was needed. The laser speed and power had to be configured to ensure a clean cut without melting the acrylic. Symmetry allowed for time-saving by printing duplicates of certain parts. Clear glue was used to secure all the box joints. The legs were attached using lock nuts, with standoffs connecting the 3-D printed Raspberry Pi base enclosure to the inner acrylic plate. The radars were then installed using long screws, plastic spacers, nuts, and wingnuts. The cameras were placed on the holder and the central support beam was fitted and glued. A camera support beam was used to ensure the cameras were held in place while allowing for easy removal. For the left radar, the 3-D printed angle adjustment was installed. The battery pack and the

Table 6.2: Bill of materials for the roadside unit test rig

ITEM	PART	PRICE (R)	QTY	COST (R)
1	uRAD USB v1.2	3 496.93	2	6 993.86
2	Logitech C270 Webcam	497.00	2	994.00
3	Samsung Battery Pack	699.00	1	699.00
4	Raspberry Pi Model 4B	1 039.00	1	1 039.00
5	Plexiglass 3x500x375 mm	179.00	4	716.00
6	Legs 150x25 mm	55.00	4	220.00
7	Screws, nuts, bolts	7.00	3	21.00
8	USB cable	59.00	3	177.00
TOTAL				**10 859.86**

peripherals were connected to the Raspberry Pi via USB cables and placed in the enclosure. Finally, a layer of foil was installed between the two radars to prevent them from interfering with each other. The Pi was configured to connect to a mobile hotspot on boot, enabling access via the Termius Android application. Figure 6.7 shows three photographic views of the roadside unit test rig.



(a) Front view

(b) Side view

(c) Top view

(d) Oblique view

Figure 6.7: Roadside unit test rig photographs, left radar offset 25° from parallel

Chapter 7

Results

This chapter describes the experiments used to assess the system's performance. The experiments were performed according to the experiment test plan outlined in Section 3.6 and can be divided into three categories: processing performance, controlled scenario radar performance, and uncontrolled scenario radar performance. The results are used to assess if the technical requirements of the project were met. Comparisons are drawn between range and speed estimates of moving vehicle(s) using GPS data, video footage, and radar estimates. Both controlled and uncontrolled experiments were performed.

Table 7.1 outlines the experiments performed. For the controlled experiments, only ET5 is discussed. The laboratory and single radar experiments (ET1 to ET4) were performed during the development process and are not representative of the final system configuration and processing algorithm. Similarly, only ET6 is presented for uncontrolled testing.

7.1 Experiments

Experiments were carried out to quantify system performance and to determine if the project requirements were met. After the acceptance testing described in section 6.1 was performed, it was found that the hardware and processing algorithm satisfied the respective requirements. The radar calibration experiment for ensuring accurate range estimation is presented first. An analysis of the effect of zero-padding on accuracy was performed on the

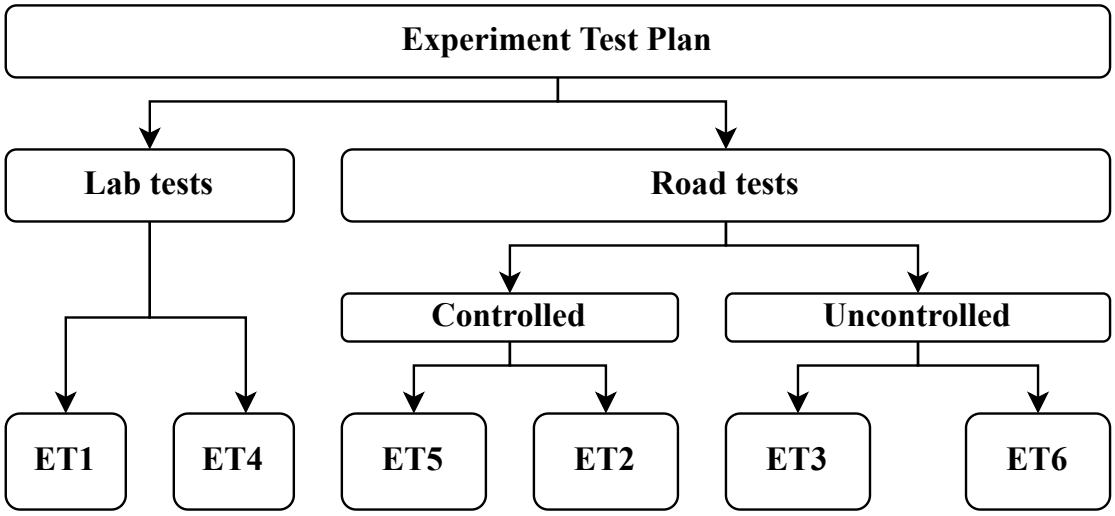


Figure 7.1: Experiment test plan hierarchy, experiments defined in Tab. 7.1

results of this experiment. Next, the four CFAR algorithms were applied to the radar data and their performance was compared. The performance of multithreaded software variations was quantified, followed by a discussion on the controlled and uncontrolled roadside experiments.

Experiments requiring a configuration process were collated into the experiment test plan, summarised in Tab. 7.1. Only ET5 and ET6 from Tab. 7.1 are presented, as experiments ET1 to ET4 were used for iterative project development. Offline processing refers to the capture of raw data and processing at a later stage. Conversely, real-time processing refers to processing data as soon as it is received from the radar sensor. Figure 7.1 illustrates the experiment process where the numbers indicate the order in which the experiments were performed.

7.1.1 Radar Range Calibration

Calibration was required to ensure the radar's range estimates were accurate. This was achieved by comparing the uncalibrated radar estimates to ground truth values. Since the speed estimate was based on the difference in beat frequencies of the up and down chirps, only the beat frequency estimate required calibration. A flat metal plate target was used so that the distance between the plate and the radar could be accurately measured

Table 7.1: Experiment tests for the single and dual radar test rigs

ID	Requirement	Description
ET1	Single radar: laboratory test	Verify processing algorithm using a trolley and office chairs
ET2	Single radar: controlled test	Vehicle driven past system at pre-determined speeds
ET3	Single radar: uncontrolled test	Place system at the main road and record vehicles passing by
ET4	Dual radar: laboratory test	Same setup as ET1, offline and real-time processing
ET5	Dual radar: controlled test	Same setup as ET3, offline processing only
ET6	Dual radar: uncontrolled test	Same setup as ET2, real-time processing only

using a tape measure. The radar I/Q data capture script was run for 30-second bursts. Thereafter, the radar estimates were processed offline and a comparison was made with the actual distance. The ratio of the true distance obtained using a tape measure and the radar's range estimate was used to calibrate the radar by multiplying the ratio with the beat frequency of each sweep. This calibration approach was valid since the beat frequency is directly proportional to the range estimate (Eq. 2.9). The final test rig (Section 6.4.3) was used during the calibration process to ensure the estimates were based on the prototype used in further testing. Figure 7.3 shows the calibration setup, with the aluminium sheet placed parallel to the radar at a distance of 3 meters. The indoor multipath effect did not affect the estimate: the flat plate had a large SNR and removing it from the scene caused the beat frequency at the 3 m range to disappear. Though it would be beneficial to test calibration for further ranges, it was decided that the comparison to GPS range estimates during the final experimentation would highlight whether further calibration was needed.

Figure 7.2 shows the radar accuracy results for several zero-padding lengths (Section 2.7.5) for the metal plate target placed 3 meters away. By increasing the amount of zero-padding,

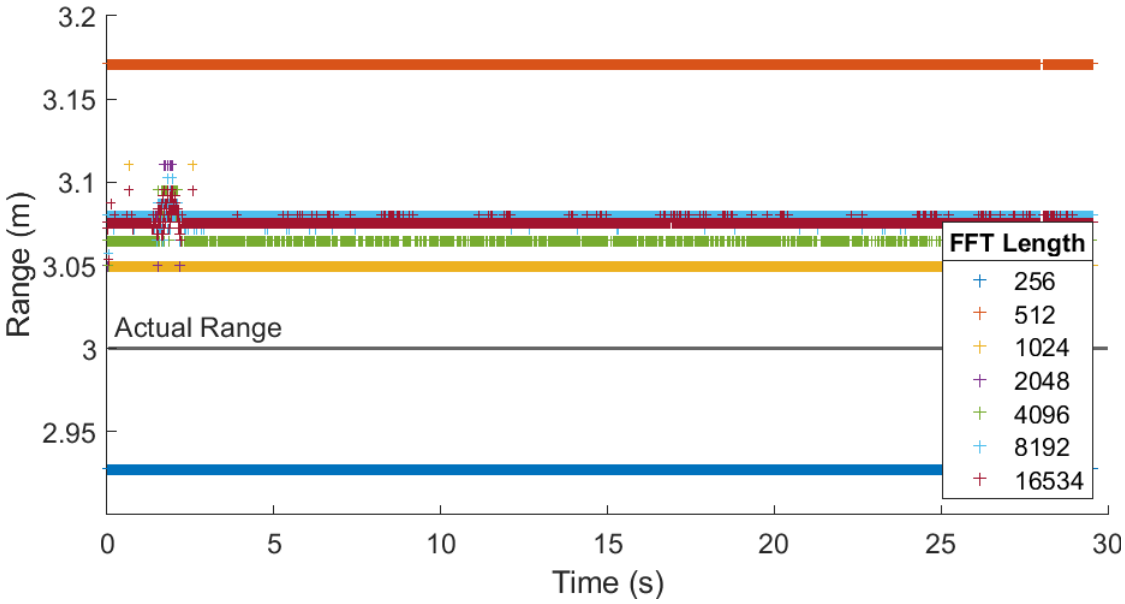


Figure 7.2: Calibration test of the relationship between the length of zero-padding for 200 input samples and the range estimate accuracy for a metal plate target placed three meters from the radar

the estimate accuracy improved up to a certain point. The lengths were chosen as powers-of-two only since these lengths result in optimum FFT performance (Section 2.7.4). A sequence of 256 samples underestimated the target's position while a length of 512 resulted in overestimation. The 1024-point FFT was selected as it provided sufficient accuracy and did not notably increase the processing latency. Though the deviation for larger FFT lengths could have resulted from an incorrect tape measure estimate, this cause was deemed unlikely based on the calculated tape measure uncertainty of $\frac{1}{2\sqrt{6}} = 0.2\,\text{mm}$. Nevertheless, the range estimate accuracy met the requirement of <1.4 m (TR6, Tab. 3.2). The calibration factor ζ was calculated as shown in Eq. 7.1 and implemented as discussed in Section 6.1.1 for the range accuracy acceptance test.

$$\frac{\text{tape measurement}}{\text{radar estimate}} = \frac{3}{3.0496} = 0.9837 \tag{7.1}$$

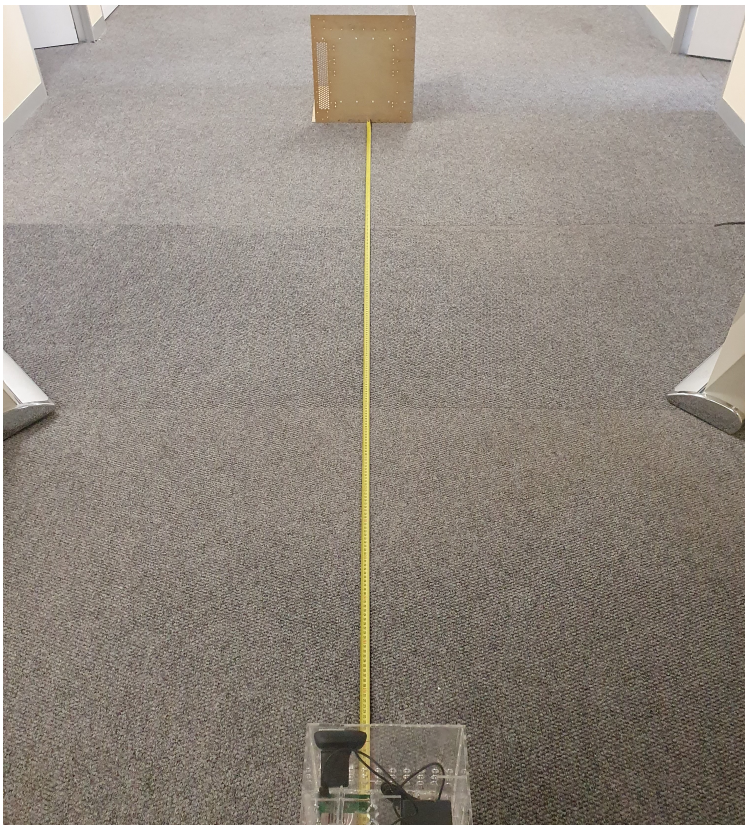


Figure 7.3: Calibration test setup

7.1.2 CFAR Algorithm Comparison

Single and multi-target scenarios were recorded using the radar hardware. Both the CFAR threshold and received spectrum were overlaid and plotted sweep by sweep. The threshold was then observed as the target moved along the spectrum. When the target's beat frequency amplitude resulted in a threshold breach, a detection was made. Thereafter, the detection results of the SOCA-, GOCA-, CA-, and OS-CFAR algorithms were compared.

After testing each algorithm using the MATLAB simulation presented in Section 5.2.2, OS-CFAR produced the fewest missed detections. The superior performance of OS-CFAR was expected based on the literature [55]. It was later found that larger vehicles produced two peaks as opposed to a single peak in the beat frequency spectra. It was observed that the sharp drop in the OS-CFAR threshold fell between the peaks, resulting in a missed detection. Next, the SOCA-CFAR was then evaluated in more depth since it was hypothesised

that the use of the smaller average of the two training sets would allow for the detection of wider, multi-peak targets. After inspecting simultaneous video and frequency spectra playback, and the final detection results, SOCA-CFAR was found, on average, to offer a higher probability of detection when real radar data was used. Additionally, the processing requirements could be reduced as the sort operation used in OS-CFAR was not needed. After testing GOCA and CA-CFAR, it was found the CA-CFAR offered the most consistent performance of the four options. The other three were found to perform better in certain scenarios than others, whereas CA-CFAR offered similar detection performance over a wide range of scenarios.

Table 7.2: Table of the implemented CA-CFAR parameter values

Parameter	**Value**
Number of guard cells	14
Number of training cells	16
Probability of false alarm	0.004

7.1.3 Software Performance Experiments

One aspect of quantifying system performance required measuring the update rate of the multithreaded variations of the Python processing pipeline. The raw data update rate was found to average around 92.4 sweeps/second as discussed in Section 6.2.1. The update rate of the multithreaded variations described in Section 6.2.4 was then measured. This involved recording the processing period for each sweep before calculating the average period. This experiment was run on the Raspberry Pi 4B computer.

7.1.4 Controlled Scenario Experiments

Experiments involving a single controlled car target were performed. Initially, the car travelled at various speeds between 20 and 70 km/h. The test rig was placed as close to the curb as possible. The scenario resembled the simulated scenario depicted in Fig. 5.1a. Raw

I/Q data from the uRAD USB v1.2 radar was recorded and saved on a laptop connected to the test rig. The radar estimates were compared to the speed of the vehicle as seen on the speedometer, despite the inaccuracy thereof [61], to provide a qualitative gauge of the radar's performance. Synchronisation of the video with the radar data required using voice indicators, recorded by the cellphone's microphone, marking the start and stop times of the radar recording. The video sections were then extracted and paired with the correct radar data set.

Experimentation using the roadside unit test rig pictured in Fig. 6.7 involved placing the unit close to the road edge and driving a vehicle past the system. Both the inner and outer lanes were used. I/Q data was recorded on the Raspberry Pi and transferred to a PC for processing. Real-time processing was not tested in this experiment since the purpose was to assess the range and speed metrics such as accuracy and maximum detectable values. Three speeds were used: 45 km/h, 60 km/h and 70 km/h, as these cover the expected range of speeds for vehicles travelling on a main road. The radar was placed at 'car bumper level'—28 cm above the ground. A compact sedan and a hatchback were used for controlled experimentation. GPS data such as speed, coordinates, and accuracy were collected using a cellphone placed inside the test vehicle. The GPS coordinates of the radar and test vehicle allowed for calculating the distance between them. Using Google Earth, the road width (for estimating the angle of arrival) was found to be 7 m. The GPS data and radar estimates were plotted and comparisons were drawn as presented in Section 7.2.2. Since the road was a flat surface, the accuracy of the Google Earth measurement was sufficient. The *inner lane masking the outer lane* scenario and the effect of closely-spaced vehicles on detection performance were investigated in these experiments. The right-side radar was tested using a Hyundai i20, while a Toyota Corolla (2005 model) was used to test the left-side radar.

7.1.5 Uncontrolled Scenario Experiments

This experiment involved placing the test rig on the pavement next to a main road and obtaining 30-second burst captures. The scenario resembled the simulated scenario depicted in Fig. 5.1a. Only real-time processed radar data (processed as soon as the radar sensor captured a sweep of I/Q data) was captured. Accuracy and maximum range metrics were

assessed using the I/Q data obtained during the controlled experiments and were consequently not evaluated in the uncontrolled experiment. Using Google Earth, the road width (for estimating the angle of arrival) was found to be 9 m.

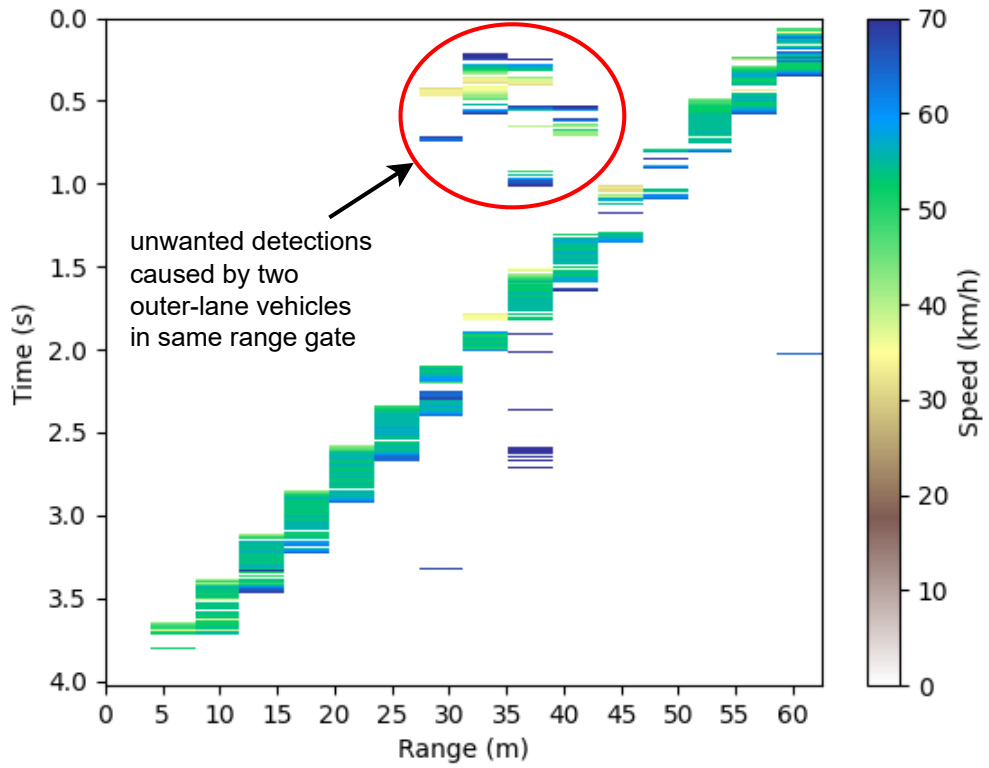
Two uncontrolled experiments are presented in this section. Experiment A involved adjusting the left radar by 25° and placing the radar at two heights. First, the radar was placed at 'car bumper level'. Next, the radar was placed at 'car roof height'—approximately 133 cm above the ground—and 1 m from the road edge. The goal of Experiment A was to assess whether raising the radar would reduce the effect of the outer lane being masked by the inner lane and to assess if the outer-lane SNR could be improved by changing the angle of the left-side radar. Experiment B used the calculated 18° angle, presented in Section 4.3.3, for maximising the left side range and detection of the outer-lane vehicles.

7.2 Analysis and Discussion

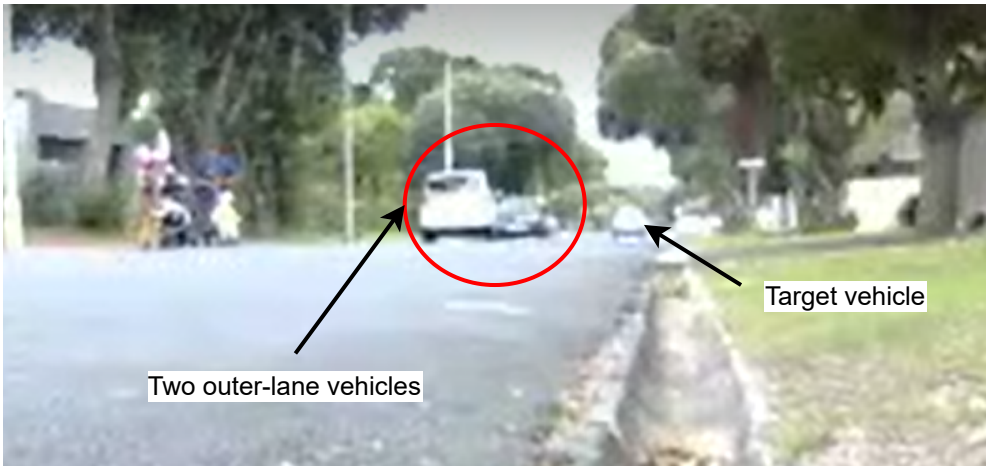
This section reviews the results obtained from the various experiments and presents an analysis thereof. The system performance results are analysed first, followed by the results of the controlled experiments. Finally, the uncontrolled scenario results are described.

7.2.1 Software Performance Results

Table 7.3 compares the update rate for various threaded implementations. Based on requirement TR9 in Tab. 3.2, all of the variations produced a qualitatively sufficient update rate. The update rates in Tab. 7.3 were obtained by averaging the difference between time stamps of the estimates (times at which estimates were produced by the processing algorithm) over 30-second capture intervals obtained using a Raspberry Pi 4B. It was found that using a preallocated array for recording time stamps caused the occasional failure of one of the cameras. Consequently, a Python list was used instead.



(a) Speed vs. Time vs. Range



(b) Video frame

Figure 7.4: Waterfall plot and video frame obtained by the right-side radar for a hatchback car driven past the system at 60 km/h on the inner lane

Table 7.3: Performance comparison of thread configurations for raw and real-time processed data capture on a Raspberry Pi 4B

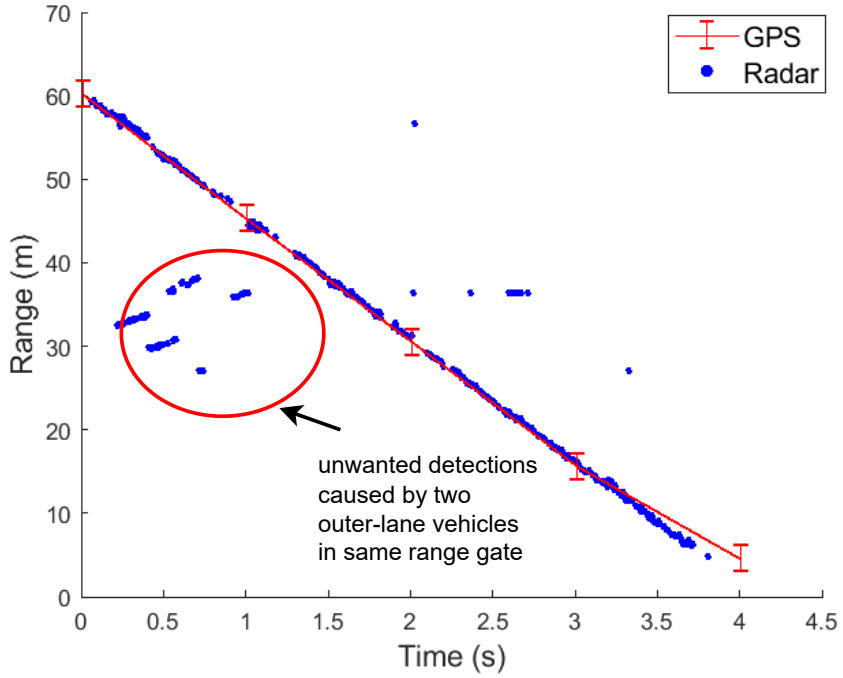
Type	**Threads**	**Average update rate**	**Description**
Raw data acquisition (sweeps/sec)	2	30	single camera and uRAD in same thread
	4	92	Each peripheral on separate thread
Real-time processing (estimates/sec)	1	8.3	All peripherals in single thread
	2	8.4	single camera and uRAD in same thread
	4	7.5	Each peripheral on separate thread

7.2.2 Controlled Scenario Results

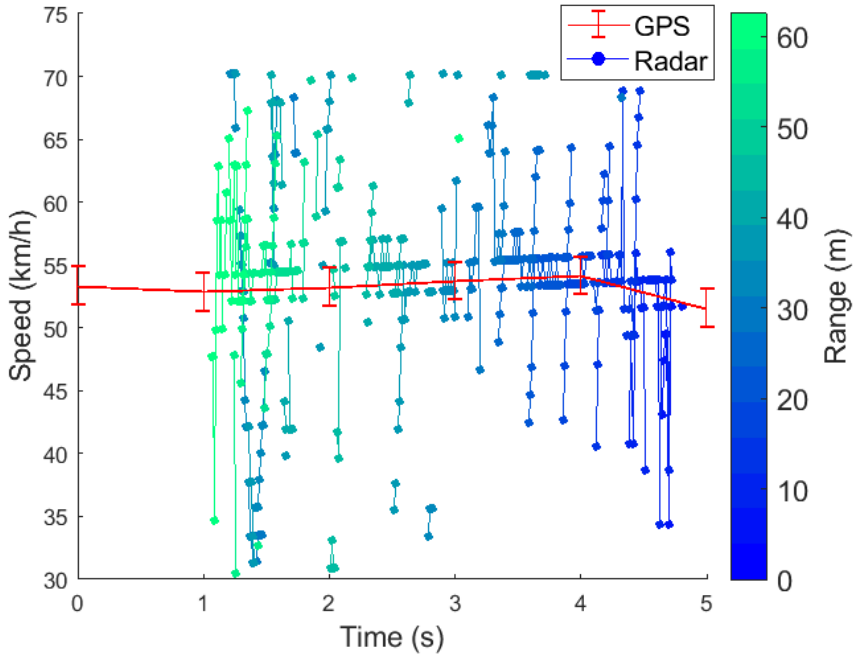
The radar's speed and range estimates satisfied the accuracy and maximum detectable value requirements when tested for speeds 45 km/h, 60 km/h, and 70 km/h. In this section, only the 60 km/h test is analysed. Results of the other tests are presented in Appendix A. Figure 7.4a shows that targets could be detected when at the furthest required range of 60 m. Similarly, the maximum expected speed was tested and is presented in Fig. A.1 in Appendix A.

The unwanted detections caused by the outer-lane vehicles, pictured in Fig. 7.4b, are highlighted. This scenario was found to be uncommon. The main road width of 9 m compared to the 7 m width of the road used in the controlled experiment eliminated the unwanted detection of outer-lane targets by the right-side radar.

As shown in Fig. 7.5a, the radar range estimates corresponded to the GPS range estimates. The GPS range estimates had a maximum accuracy of 1.4 m and are indicated by the error bars in Figures 7.5a and 7.5b. Each GPS estimate, obtained using the Android GPS Logger application, included an accuracy value associated with that estimate. The radar estimates met the required range accuracy as defined in Tab. 3.2. Figure 7.5b shows that the speed estimates lacked precision: the estimates appear to ramp up as the target moves through the range divisions. Despite this, the estimates cluster around the



(a) Range vs. time plot of radar and GPS estimates



(b) Speed vs. time plot of radar and GPS estimates

Figure 7.5: Range and speed estimates, obtained by the right-side radar, compared to cellular GPS estimates for a hatchback car driven past the system at 60 km/h on the inner lane

Table 7.4: Statistics of the right-side radar controlled speed tests

Experiment	**45 km/h**		**60 km/h**		**70 km/h**	
Source	**Radar**	**GPS**	**Radar**	**GPS**	**Radar**	**GPS**
Mean (km/h)	43.27	42.49	53.84	53.12	58.02	59.66
Median (km/h)	41.23	42.48	54.45	53.28	59.7	59.85
σ **(km/h)**	7.9	0.95	8.33	0.89	7.47	1.8

true value and lie within the accuracy error bars of the GPS speed estimates. Essentially, the average speed estimate was within the required accuracy. Despite this, Fig. 7.6 shows that the time of arrival (TOA) calculation—the ratio of distance and speed—was accurate when compared to the GPS estimate. This was due to computing the TOA for each of the 16 large bins and using the smallest value to represent turn safety. This method effectively rejected incorrect estimates of range and/or speed. However, false high-speed detections at far ranges can result in an 'unsafe turn' being reported since the TOA of such a detection is small. Using the data tip in Fig. 7.6, summing the value of X and Y gives a TOA estimate of approximately 5 seconds. Observing the x-axis time of the final estimate in Fig. 7.6, it was observed that the target arrived around the 5-second mark. This calculation can be applied to any TOA estimate and used to determine its validity.

Additionally, it was determined that the closer the target was, the less dependent the safety estimate was on the speed estimate compared to the distance estimate. When targets were further away, an accurate speed estimate ensured a better TOA prediction. When a target was nearby, a wider range of speeds resulted in an unsafe prediction.

Table 7.4 presents the statistics of the speed estimates obtained when monitoring an inner-lane vehicle during the right-side radar, controlled speed experiment. Based on the mean and median of each experiment, the radar and GPS speed estimates are highly correlated. As expected, the speedometer measurements deviated substantially from both the radar and GPS estimates [61], with the deviation increasing for higher speeds. The poor standard deviation (σ) of around 8 km/h for the radar estimates, which can be seen in Fig. 7.5b, indicates that either the processing algorithm or the radar hardware did not

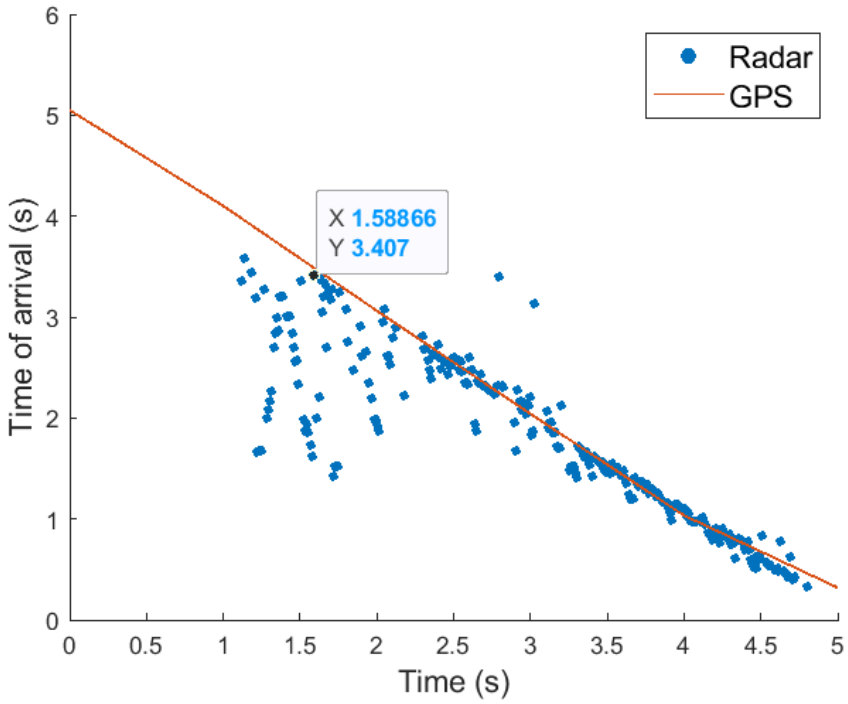
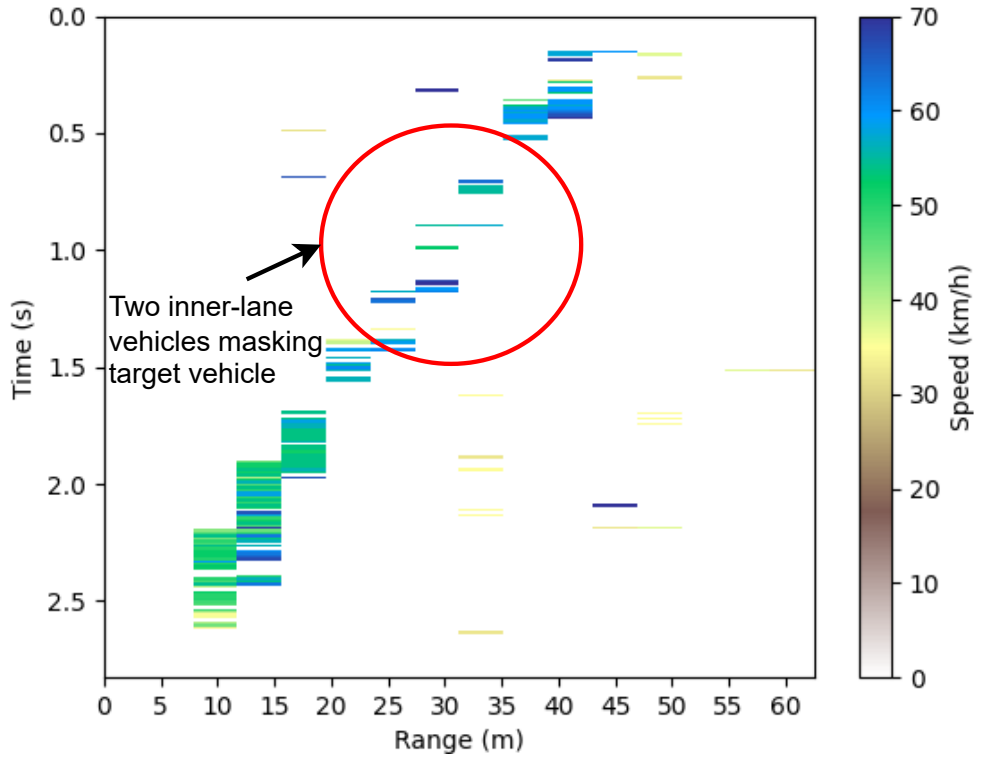


Figure 7.6: Time of arrival vs. time plot of radar and GPS estimates obtained by the right-side radar for a hatchback car driven past the system at 60 km/h on the inner lane. A data tip of the first reliable detection is shown.

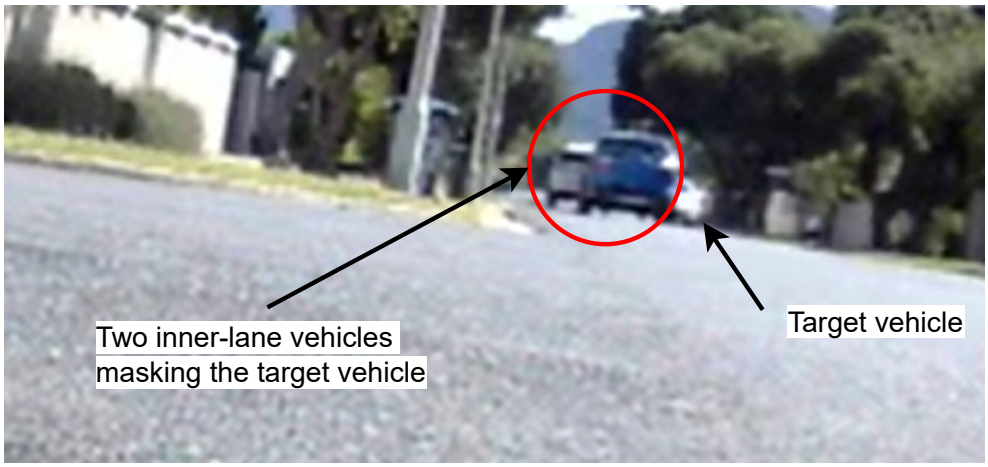
perform as expected.

For targets travelling on the outer lane observed by the left-side radar, similar results were obtained as those from the right-side radar. However, the maximum achievable range for a compact sedan was 45 m as seen in Fig. 7.7a, failing to meet the required detection range of 60 m. This was due to manual steering of the radar beam by 18° for detection of vehicles on the outer lane. The gain of the radar beam at 15° from radar boresight was insufficient for detecting cars at 60 m. It was noted that the 18° offset was based on a 4 m road width. Using the equations presented in Section 4.3.3, a 21.7° would be needed for the 7 m-wide road. However, by offsetting the radar by 18°, targets at the furthest range would be closer to boresight, hence it is believed that increasing the offset angle would be detrimental to the maximum detection range.

Figures 7.7a and 7.8a show the reduced range for a target travelling at 60 km/h along the outer lane of a 7 m wide road. Additionally, the masking of the outer-lane vehicle, pictured



(a) Speed vs. Time vs. Range



(b) Video frame

Figure 7.7: Waterfall plot and video frame obtained by the left-side radar (offset 18° from parallel to the road) for a hatchback car driven past the system at 60 km/h on the outer lane with two inner-lane cars masking the detection thereof

Table 7.5: Statistics of the left-side radar controlled speed tests

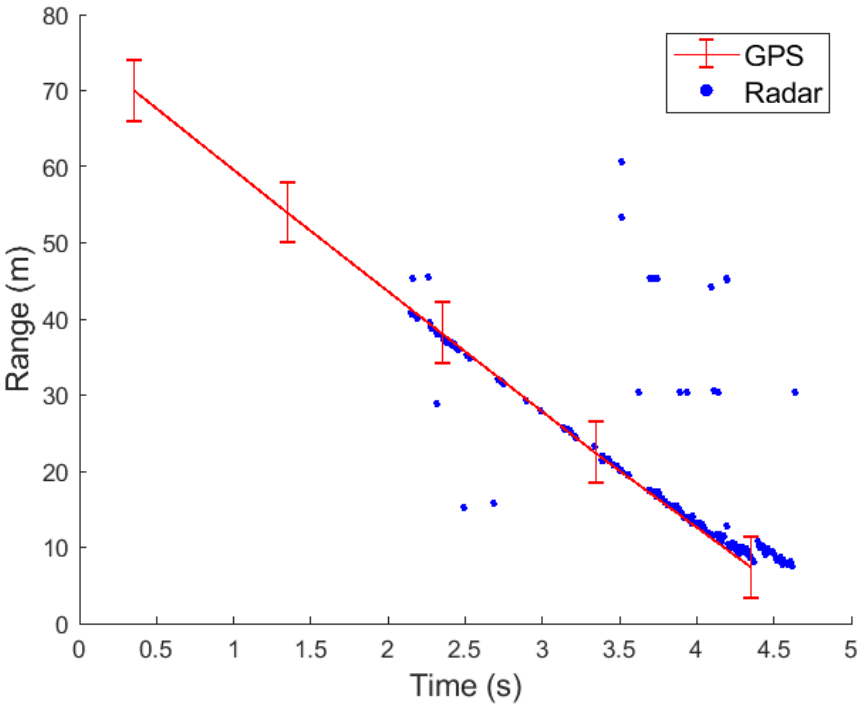
Experiment	**45 km/h**		**60 km/h**		**70 km/h**	
Source	**Radar**	**GPS**	**Radar**	**GPS**	**Radar**	**GPS**
Mean (km/h)	46.91	47.54	52.54	57.12	54.34	64.99
Median (km/h)	45.09	47.56	53.76	57.08	58.07	64.86
σ **(km/h)**	9.47	0.84	9.25	0.38	11.02	2.6

in Fig. 7.7b, did not completely prevent the detection of the outer-lane target vehicle: Fig. 7.7a shows that some detections were made when the target vehicle was obscured by the two inner-lane vehicles. However, some of the speed estimates appear to deviate from the actual values in this region.

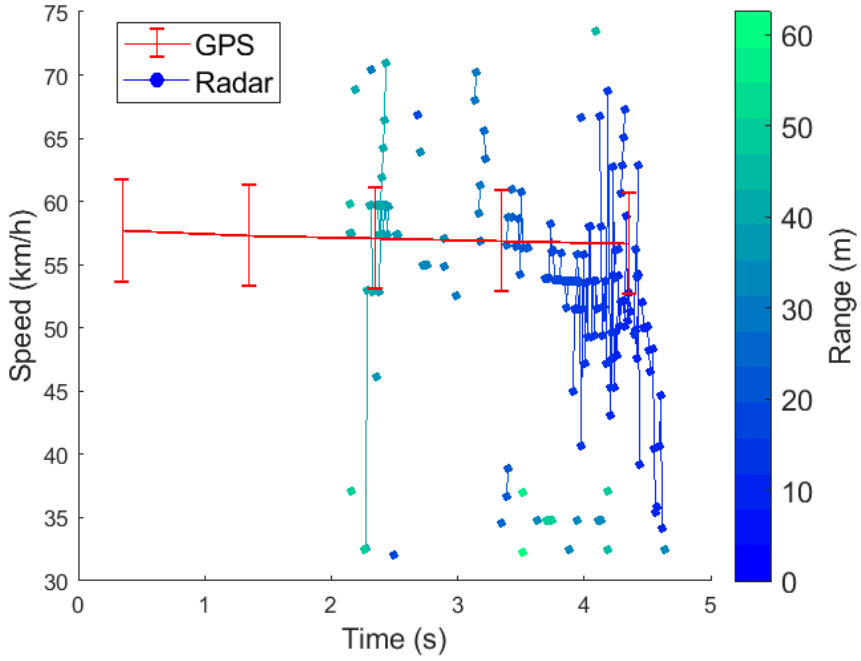
The radar speed estimate deviated from the GPS estimate for close ranges despite the application of angle correction, though the estimates clustered within the GPS accuracy of 3 m. The deviation was anticipated based on the simulation results shown in Fig. 5.7a, where the angle correction improved but did not solve the deviation. The assumption of the largest SNR being returned by the midpoint of the car's front bumper was likely incorrect.

The difference in GPS accuracy compared to the right-side experiment was attributed to the use of a different satellite. The TOA vs. time shown in Fig. 7.9 produced accurate estimates, with gaps due to the vehicles in the inner lane masking the target vehicle in the outer lane. Summing the X and Y value of the data tip in Fig. 7.9 gives approximately 4.7 seconds, corresponding to the x-axis time of the final radar estimate. This test can be applied to any point that is not an outlier in Fig. 7.9.

Table 7.5 presents the statistics of the speed estimates obtained when monitoring an outer-lane vehicle during the left-side radar, controlled speed experiment. Based on the mean and median of each experiment, the radar and GPS speed estimates are correlated, though not as well as those of the right-side radar experiments presented in Tab. 7.4. The deviation due to the change in angle of arrival as the target's range reduced, despite angle correction, was deemed the cause of the decreased correlation. The poor standard deviation (σ) of around 10 km/h the radar estimates was worse than the 8 km/h for the right side



(a) Range vs. time plot of radar and GPS estimates



(b) Speed vs. time plot of radar and GPS estimates

Figure 7.8: Plots comparing estimates by the left-side radar (offset 18° from parallel to the road) and by cellular GPS for a Hyundai i20 travelling at 60 km/h on the outer lane

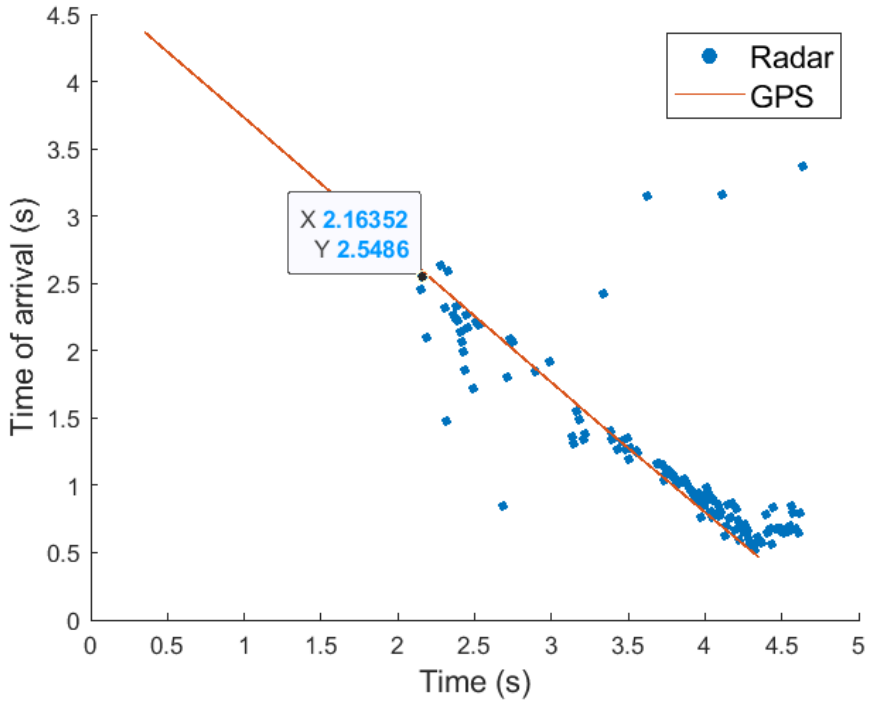
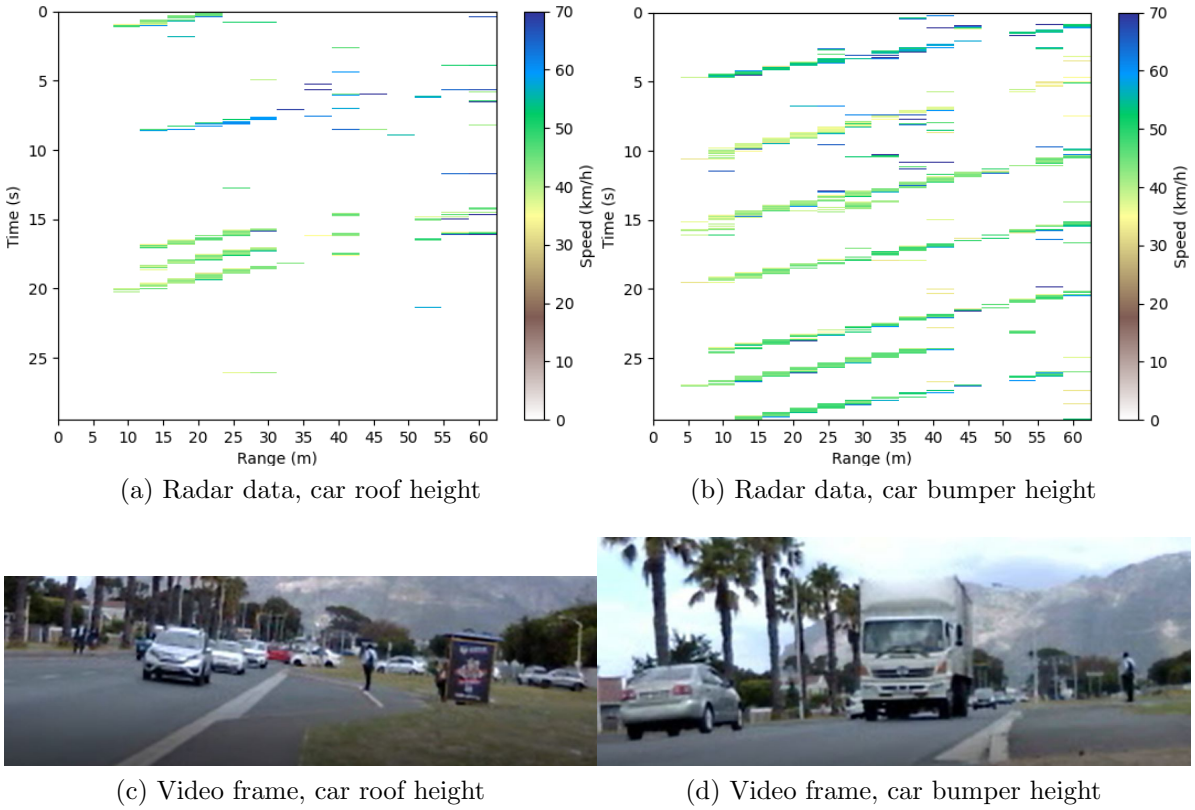


Figure 7.9: Time of arrival vs. time plot of radar and GPS estimates obtained by the left-side radar (offset 18° from parallel to the road) for a hatchback car driven past the system at 60 km/h on the inner lane with two inner-lane cars masking the detection thereof. A data tip of the first reliable detection is shown.

radar. The poor standard deviation was attributed to measurement and process noise [42], which could be drastically improved by implementing a tracking algorithm such as the Kalman filter [42] [43].

7.2.3 Uncontrolled Scenario: Experiment A

Comparing Figures 7.10a and 7.10b, placing the radar on the roof of a vehicle resulted in more detection discontinuities (gaps in the detection range-time lines). This reduction was primarily due to the offset of 1 m from the road edge. This offset resulted in cars being observed at an angle further from radar boresight compared to placement at the edge of the road. Figures 7.10c shows the frame where three vehicles are spaced in range and correspond to the three uniformly spaced detection sets in Figures 7.10a in the 15-20 second



(a) Radar data, car roof height

(b) Radar data, car bumper height

(c) Video frame, car roof height

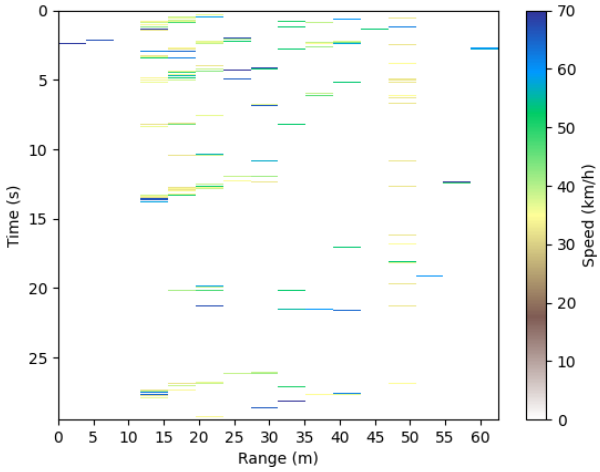
(d) Video frame, car bumper height

Figure 7.10: Right-side radar speed vs. time vs. range plots and video frames captured at a 9 m wide main road

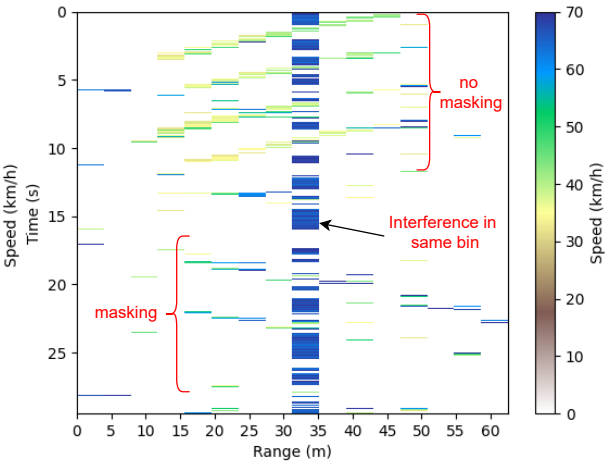
interval. Figure 7.10d shows the truck that generated the set of detections ending at the 15-second mark of Fig. 7.10b.

The effect of placing the radar at car roof height severely reduced the detection of outer-lane vehicles. This is shown in Fig. 7.11a where the detection sets are vague compared to the detection sets shown in Fig. 7.11b. The video frames pictured in 7.11c and 7.11d show the difference in the observed scene between the car bumper height and car roof height placements. This experiment proved that raising the height of the system would not reduce masking.

Figure 7.11b shows that when there is no masking, the outer-lane vehicles are sufficiently detected, aside from the reduced maximum range of 50 m. In the event of masking (lower half of Fig. 7.11b), no detections are made aside from two 'faint' false detection sets. Interference of unknown origin is shown in Fig. 7.11b. The interference speed estimates ranged



(a) Radar data, car roof height



(b) Radar data, car bumper height

(c) Video frame, car roof height

(d) Video frame, car bumper height, no masking

(e) Video frame, car bumper height, masking

Figure 7.11: Left-side radar speed vs. time vs. range plots and video frames captured at a 9 m wide main road with the radar at 25° from parallel to the road

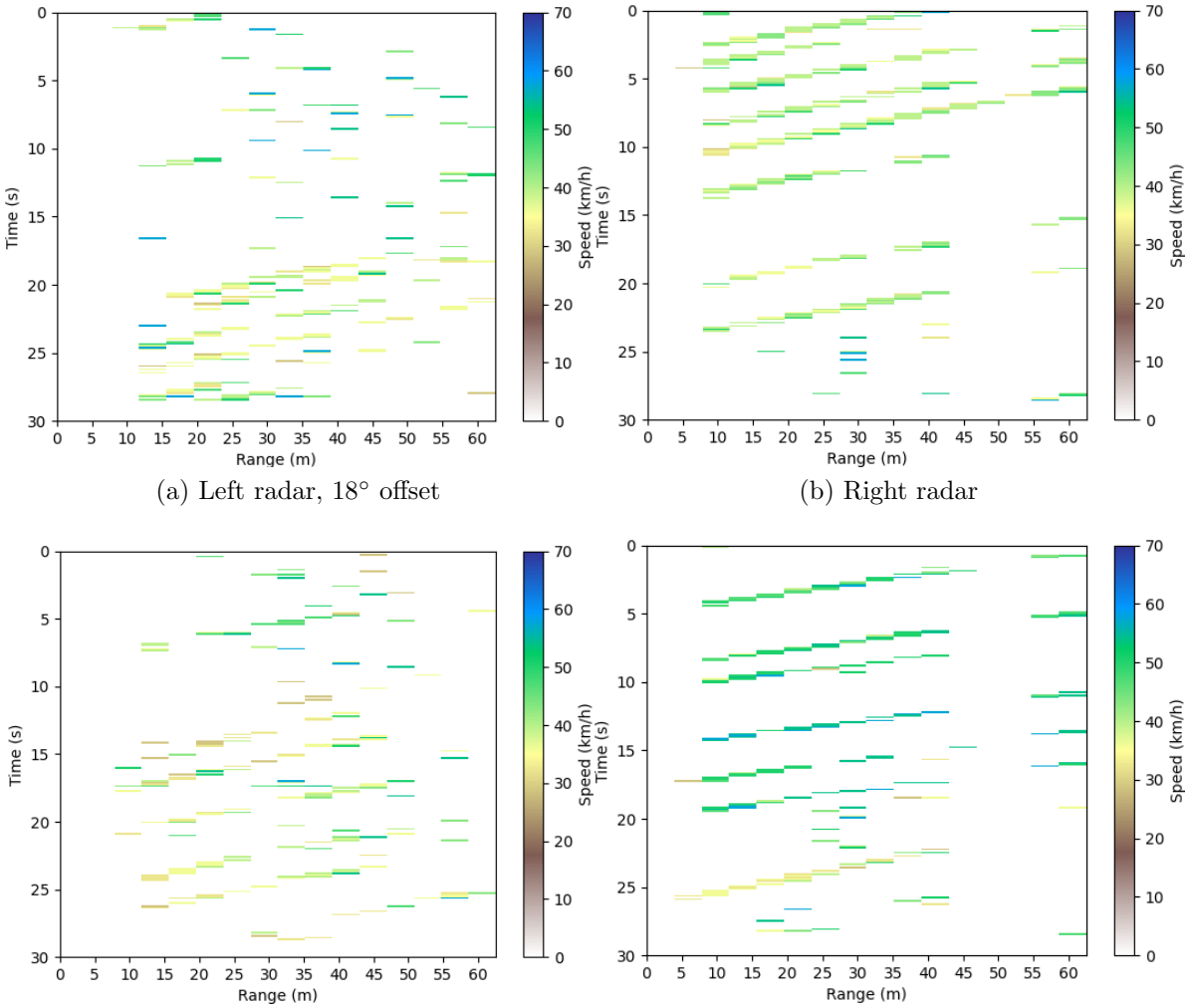
from 66 km/h to 70 km/h. This interference could be filtered out with additional processing due to its regular nature. Alternatively, a more rigorous approach could be followed such as the approach presented in Section 2.6.3. The video frame in Fig. 7.11e shows the severity of the masking from the radar's point of view. Both the plots and video frames prove that the radar system cannot function in high-traffic scenarios.

7.2.4 Uncontrolled Scenario: Experiment B

Figures 7.12 and 7.13 show the results obtained by the roadside unit test rig pictured in Fig. 6.7. The detected speeds were limited to a maximum of 60 km/h due to false alarms being reported for speeds greater than 60 km/h (dark blue bars in the figure), predominantly for the left-side radar. Interference was attributed as the cause of these false alarms. Figure 7.13a shows that the left radar is capable of detecting vehicles at 50 m consistently and 55 m occasionally. Observing the video footage, it was found that the detection at 55 m was of a Toyota Quantum minibus.

The estimates indicate that the left radar obtained far more false alarms than the right radar. This was due to the significant amount of masking that occurred when the road was busy. The periods where no masking occurred are clear: detection sets can be seen in the left radar results. Looking at the right-side radar results in Fig. 7.12 and 7.13, the system performs optimally: there are minimal false alarms and only one region of discontinuity, attributed to interference in those range bins. The results shown in Fig. 7.12 and 7.13, particularly those for the right side radar, indicate that $PFA = 0.004$ (used to set the CFAR threshold) was tuned sufficiently.

The reduced update rate of 8.4 sweeps/second due to real-time processing in Experiment B compared to the offline processing update rate of 93 sweeps/second in Experiment A did not have a notable effect on the system performance. Comparing Figures 7.10b and 7.12d, no visible differences between the processing methods are discernible.



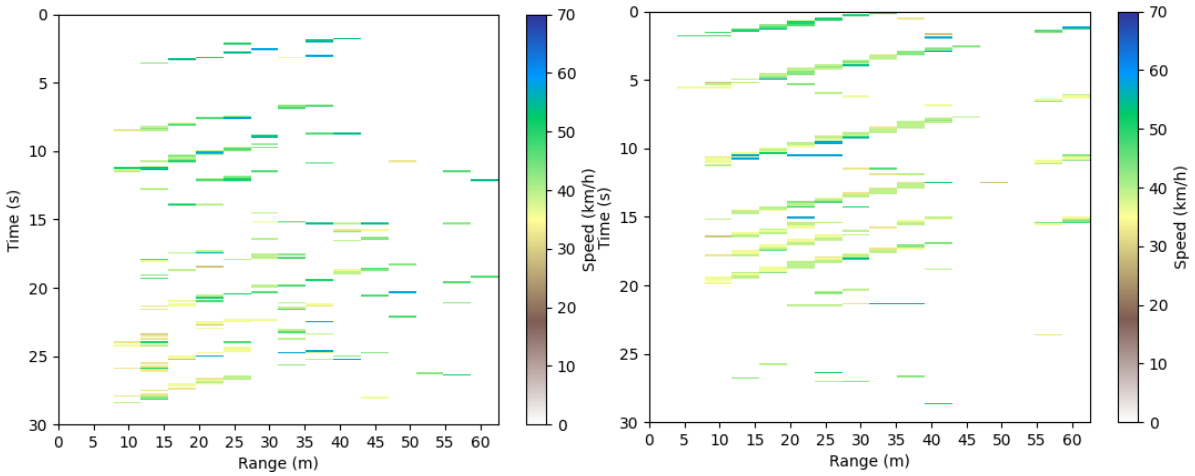
(a) Left radar, 18° offset

(b) Right radar

(c) Left radar, 18° offset

(d) Right radar

Figure 7.12: Two captures represented as Speed vs. Time vs. Range plots where the system was placed at a 9 m wide main road at car bumper height and data was processed in real time



(a) Left radar, 18° offset

(b) Right radar

Figure 7.13: Additional capture represented as Speed vs. Time vs. Range plots where the system was placed at a 9 m wide main road at car bumper height and data was processed in real time

7.3 Summary

Thorough experimentation was performed to assess the suitability of the low-cost dual radar system for predicting turn safety at unregulated intersections. The performance of the processing algorithm, processing hardware, and radar hardware met the technical requirements summarised in Tab. 3.2. Though the update rates of all multithreaded variations met the minimum requirement, the two-threaded implementation was selected as it offered the highest update rate of around 8.4 estimates/second when real-time processing was performed. As shown by the controlled experiments, the system was able to predict the TOA of an approaching vehicle accurately when compared to GPS estimates—the measurements were within the GPS accuracy range of ±1.5 m. The speed estimates had a standard deviation of around five to ten times the GPS estimates, reducing the reliability of the system. Presently, the cause of this deviation is unknown. Despite the deviation, the mean and median values of the radar estimates correlated closely with the GPS estimates as shown in Tab. 7.4. In addition, it was found that angle correction only improved the speed estimate but did not fully resolve the deviation due to the angle of arrival of targets observed by the left-side radar. For the uncontrolled tests, the processing algorithm was able to detect

multiple vehicles on a single lane using the triangle FMCW waveform. These tests highlighted the weakness of onboard collision prediction: poor detection of outer-lane vehicles when inner-lane vehicles are present.

The suboptimal performance of the left radar in detecting outer-lane vehicles indicated that the system failed to meet the goal of accurately determining turn safety at unregulated intersections. Comparing the left radar estimates to those of the right radar presented in Fig. 7.12 and 7.13 shows that the left radar had an increased number of false alarms due to the masking of the outer lane targets by the inner lane cars. In addition, the detection sets (lines of constant gradient) for each vehicle had more fluctuations in the speed estimates—a standard deviation above 8 km/h—and more discontinuities which can be seen in Figures 7.12a, 7.12c, and 7.13a. It was hypothesized that a tracking algorithm such as the Kalman filter [42] [43] would improve both the masking and poor precision of the speed estimates.

Chapter 8

Conclusions

8.1 Conclusion

This project investigated the suitability of a dual, low-cost radar system for determining turn safety at uncontrolled road intersections. The proposed system warns the driver of potential collisions if a turn is made. Specifically, the right-turn scenario was investigated as this turn requires the driver to cross one or more lanes of traffic (in regions where vehicles travel on the left side of the road). Observation of vehicles travelling in both directions is required to determine right-turn safety. A processing algorithm was designed based on the two-lane scenario. An elegant solution for estimating the angle of arrival of an approaching vehicle, based on a given road width, was conceived to improve the accuracy of the speed estimates. A roadside prototype was then designed using CAD, constructed, and used for experimentation.

The average raw data update rate of 92.4 sweeps/second provided enough headroom for real-time processing. When processing data in real time, a maximum average update rate of 8.4 estimates/second was realised, meeting the requirement of being higher than the average rate of driver decision-making in the intersection scenario.

A comparison of the radar estimates with video recordings and GPS estimates showed that the system met the project requirements regarding maximum range, maximum speed, accuracy, and update rate. The accuracy of range estimates compared to GPS estimates was

approximately 1 m, surpassing the requirement thereof. Statistical analysis was performed on the speed estimates, showing that the mean value of the GPS and radar speed estimates were highly correlated (differing by no more than 2 km/h).

Several limitations were uncovered. High traffic scenarios resulted in the masking of vehicles in the outer lane by vehicles in the inner lane. Raising the radar to car roof height, as opposed to car bumper height, did not improve the outer-lane detection in the masking scenario. Additionally, the detection range was reduced from 60 m to 50 m: since the radar hardware was incapable of beamforming, the beam had to be steered by adjusting the angle between the radar boresight and the edge of the road. The reduced power of the radar beam off of boresight reduced the maximum detection range.

The precision of the radar's speed estimates was suboptimal: the standard deviation of 8 km/h for the right-side radar and 10 km/h for the left-side radar was approximately ten times that of the GPS estimates. Since it did not markedly affect the time of arrival estimation, further investigation was not undertaken. A tracking algorithm, such as the Kalman filter [42] [43], would potentially improve the radar speed estimate precision and reduce the effect of detection gaps due to masking.

Despite the poor detection of the outer-lane vehicle and suboptimal speed estimate precision, positive research outputs were identified. Inner-lane vehicles were detected consistently at the required maximum range. Furthermore, estimates were produced at a sufficient rate and with sufficient accuracy. The roadside unit test rig allows for further research avenues to be pursued using the uRAD USB v1.2 radar. Overall, the project was a successful demonstration of the capabilities of a low-cost onboard radar solution for intersection collision prediction.

8.2 Future Work

Based on the identified shortcomings, a hybrid system consisting of a roadside unit and an onboard system would be more effective than the proposed system. The roadside unit monitors the outer lane vehicles and communicates with the host vehicle (vehicle-to-infrastructure communication [29]). The inner lane should be monitored using the proposed

system installed in the bumper of the host vehicle. This solution reduces the infrastructure cost and ensures the host vehicle can, at minimum, predict collisions from a single side at road intersections.

Though the uRAD USB v1.2 was selected, it is recommended that the offerings by RFbeam Microwave be considered as these radar modules offer superior performance and flexibility. The limitation uncovered by this project is the missed detection of outer-lane vehicles when they are masked by those on the inner lane. Emphasis should be placed on acquiring a radar capable of beamforming a sufficiently narrow beamwidth. An alternative to using a new radar system is to request a custom uRAD radar based on the required improvements. The benefit of this approach is that the developed Python software and scripts will not require modification.

An interesting avenue is to use the radars and cameras of the roadside test rig for target classification. The machine-learning application may prove to be an interesting study, as the classification could be based on both radar and camera data. Finally, a vehicle-to-infrastructure communication system could be developed. A roadside test rig could be installed on each side of the road, observing each lane independently. Alternatively, a single unit could be installed on the opposite side of the road for observing the outer-lane vehicles and providing a visual warning to the driver. The inner lane could then be observed by the onboard system, reducing the required infrastructure and meeting the goal of low-cost collision prediction at unregulated intersections.

Bibliography

[1] O. Ararat and B. Aksun-Guvenc, "A survey of recent developments in collision avoidance, collision warning and inter-vehicle communication systems," 2020. [Online]. Available: https://arxiv.org/abs/2012.12441

[2] M. E. Warren, "Automotive lidar technology," in *2019 Symposium on VLSI Circuits*, 2019, pp. C254–C255. [Online]. Available: https://ieeexplore.ieee.org/document/8777993

[3] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, "Advanced driver-assistance systems: A path toward autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 7, no. 5, pp. 18–25, 2018.

[4] A. Carullo and M. Parvis, "An ultrasonic sensor for distance measurement in automotive applications," *IEEE Sensors Journal*, vol. 1, no. 2, pp. 143–, 2001.

[5] C. Waldschmidt, J. Hasch, and W. Menzel, "Automotive radar — from first efforts to future systems," *IEEE Journal of Microwaves*, vol. 1, pp. 135–148, 01 2021.

[6] G. Liu, L. Wang, and S. Zou, "A radar-based blind spot detection and warning system for driver assistance," in *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2017, pp. 2204–2208.

[7] W. Pananurak, S. Thanok, and M. Parnichkun, "Adaptive cruise control for an intelligent vehicle," in *2008 IEEE International Conference on Robotics and Biomimetics*, 2009, pp. 1794–1799.

[8] H. G. Jung, Y. H. Cho, P. J. Yoon, and J. Kim, "Scanning laser radar-based target position designation for parking aid system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 406–424, 2008.

[9] WHO, "Road traffic injuries," *World Health Organization: WHO*, Jun. 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

[10] J. The University of the Witwatersrand, "2021-04 - South Africa bottom of the class for road safety? Here's why this isn't true - Wits University," Aug. 2022, [Online; accessed 26. Aug. 2022]. [Online]. Available: https://www.wits.ac.za/news/latest-news/opinion/2021/2021-04/south-africa-bottom-of-the-class-for-road-safety-heres-why-this-isnt-true.html

[11] L. Rondganger, "South Africa's roads deaths are a 'national crisis'," *IOL - News that Connects South Africans*, May 2021. [Online]. Available: https://www.iol.co.za/news/south-africa/kwazulu-natal/south-africas-roads-deaths-are-a-national-crisis-cefc54fe-bafe-45c6-b0d9-f7c4b1ce7ea8

[12] IIHS, "Real-world benefits of crash avoidance technologies," Report, Insurance Institute for Highway Safety, Highway Loss Data Institute, Tech. Rep., March 2022. [Online]. Available: https://www.iihs.org/topics/advanced-driver-assistance#overview

[13] HLDI, "Compendium of hldi collision avoidance research," Bulletin, Highway Loss Data Institute, Tech. Rep., December 2020. [Online]. Available: https://www.itskrs.its.dot.gov/node/209273

[14] D. Solomitckii, C. B. Barneto, M. Turunen, M. Allén, G. P. Zhabko, S. V. Zavjalov, S. V. Volvenko, and M. Valkama, "Millimeter-wave radar scheme with passive reflector for uncontrolled blind urban intersection," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 7335–7346, 2021.

[15] "Intersections and safe driving." [Online]. Available: https://www.arrivealive.co.za/Intersections-and-Safe-Driving

[16] "Intersection Safety | FHWA," Sep. 2022, [Online; accessed 4. Sep. 2022]. [Online]. Available: https://highways.dot.gov/research/research-programs/safety/intersection-safety

[17] "Statistics on Intersection Accidents," Aug. 2022, [Online; accessed 27. Aug. 2022]. [Online]. Available: https://www.autoaccident.com/statistics-on-intersection-accidents.html

[18] "What is a Rear Cross Traffic Alert? | Kia British Dominica," Sep. 2022, [Online; accessed 4. Sep. 2022]. [Online]. Available: https://www.kia.com/dm/discover-kia/ask/what-is-a-rear-cross-traffic-alert.html

[19] "Rear Cross Traffic Alert - Nissan Motor Corporation Global Website," Sep. 2022. [Online]. Available: https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/RCTA

[20] Road Traffic Management Corporation, *South African Fatal Crashes in Context*. RTMC, December 2021. [Online]. Available: https://www.rtmc.co.za/index.php/publications/reports/research-development-reports

[21] Ertunc, Ela and Cay, Tayfun and Mutluoğlu, Ömer, "Intersection road accident analysis using geographical information systems: Antalya (turkey) example," in *2013 7th International Conference on Application of Information and Communication Technologies (AICT)*, 10 2013, pp. 1–5.

[22] F. Fan, "Study on the cause of car accidents at intersections," *Open Access Library Journal*, vol. 5, no. 5, pp. 1–11, 2018. [Online]. Available: http://dx.doi.org/10.4236/oalib.1104578

[23] "What is a Collision Avoidance System?" Aug. 2022. [Online]. Available: https://www.nauto.com/glossary/what-is-a-collision-avoidance-system

[24] J. B. Cicchino, "Effectiveness of forward collision warning and autonomous emergency braking systems in reducing front-to-rear crash rates," *Accident*

Analysis & Prevention, vol. 99, pp. 142–152, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457516304006

[25] "Front Cross Traffic Alert - Technology | Lexus Europe." [Online]. Available: https://www.lexus.eu/discover-lexus/safety/front-cross-traffic-alert

[26] T. Herpel, C. Lauer, R. German, and J. Salzberger, "Multi-sensor data fusion in automotive applications," in *2008 3rd International Conference on Sensing Technology*, 2008, pp. 206–211.

[27] Mazda Motor Corporation, *Front Cross Traffic Alert (FCTA)*, Oct. 2019. [Online]. Available: https://owners-manual.mazda.com/gen/en/mazda3/mazda3_8hs5ee19i/contents/05283401.html

[28] S.-Z. Liu and S.-H. Hwang, "Vehicle anti-collision warning system based on v2v communication technology," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, 2021, pp. 1348–1350.

[29] D. Kanthavel, S. Sangeetha, and K. Keerthana, "An empirical study of vehicle to infrastructure communications - an intense learning of smart infrastructure for safety and mobility," *International Journal of Intelligent Networks*, vol. 2, pp. 77–82, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666603021000105

[30] M. Malinverno, G. Avino, C. Casetti, C. F. Chiasserini, F. Malandrino, and S. Scarpina, "Performance analysis of c-v2i-based automotive collision avoidance," in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2018, pp. 1–9.

[31] R. Q. Malik, K. N. Ramli, Z. H. Kareem, M. I. Habelalmatee, and H. Abbas, "A review on vehicle-to-infrastructure communication system: Requirement and applications," in *2020 3rd International Conference on Engineering Technology and its Applications (IICETA)*, 2020, pp. 159–163.

[32] F. Basma, Y. Tachwali, and H. H. Refai, "Intersection collision avoidance system using infrastructure communication," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 422–427.

[33] "Eclipse SUMO - Simulation of Urban MObility," Nov. 2022, [Online; accessed 22. Nov. 2022]. [Online]. Available: https://www.eclipse.org/sumo

[34] Z. Y. Rawashdeh and S. M. Mahmud, "Intersection collision avoidance system architecture," in *2008 5th IEEE Consumer Communications and Networking Conference*, 2008, pp. 493–494.

[35] J. Kim and J. Kim, "Intersection collision avoidance using wireless sensor network," in *2009 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2009, pp. 68–73.

[36] W. J. Barnes, T. I. King, H. H. Refai, and J. E. Fagan, "A wireless sensor network simulation for highway intersection collision prevention," in *2007 IEEE Intelligent Transportation Systems Conference*, 2007, pp. 173–177.

[37] A. Demba and D. P. F. Möller, "Vehicle-to-vehicle communication technology," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, 2018, pp. 0459–0464.

[38] Y. Morioka, T. Sota, and M. Nakagawa, "An anti-car collision system using gps and 5.8 ghz inter-vehicle communication at an off-sight intersection," in *Vehicular Technology Conference Fall 2000. IEEE VTS Fall VTC2000. 52nd Vehicular Technology Conference (Cat. No.00CH37152)*, vol. 5, 2000, pp. 2019–2024 vol.5.

[39] P. Wang, S. Fang, L. Zhang, and J. Wang, "A vehicle collision detection algorithm at t-shaped intersections based on location-based service," in *2015 International Symposium on Frontiers of Road and Airport Engineering*, 10 2015, pp. 308–317.

[40] E. Takeuchi, Y. Yoshihara, and N. Yoshiki, "Blind area traffic prediction using high definition maps and lidar for safe driving assist," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 2311–2316.

[41] E. Jocoy and J. Knight, "Adapting radar and tracking technology to an on-board automotive collision warning system," in *17th DASC. AIAA/IEEE/SAE. Digital Avionics Systems Conference. Proceedings (Cat. No.98CH36267)*, vol. 2, 1998, pp. I24/1–I24/8 vol.2.

[42] A. Becker, "Online Kalman Filter Tutorial," Nov. 2022, [Online; accessed 22. Nov. 2022]. [Online]. Available: https://www.kalmanfilter.net/default.aspx

[43] T. Basar, *A New Approach to Linear Filtering and Prediction Problems*. Wiley-IEEE Press, 2001, pp. 167–179.

[44] K. Geary, J. S. Colburn, A. Bekaryan, S. Zeng, B. Litkouhi, and M. Murad, "Automotive radar target characterization from 22 to 29 ghz and 76 to 81 ghz," in *2013 IEEE Radar Conference (RadarCon13)*, 2013, pp. 1–6.

[45] M. Z. Ikram and A. Ahmad, "Automated radar mount-angle calibration in automotive applications," in *2019 IEEE Radar Conference (RadarConf)*, 2019, pp. 1–5.

[46] F. Uysal and S. Sanka, "Mitigation of automotive radar interference," in *2018 IEEE Radar Conference (RadarConf18)*, 2018, pp. 0405–0410.

[47] F. Hau, F. Baumgärtner, and M. Vossiek, "The degradation of automotive radar sensor signals caused by vehicle vibrations and other nonlinear movements," *Sensors*, vol. 20, no. 21, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/21/6195

[48] K. Ramasubramanian and K. Ramaiah, "Moving from Legacy 24 GHz to State-of-the-Art 77-GHz Radar," *ATZelektronik worldwide*, vol. 13, no. 3, pp. 46–49, Jun 2018. [Online]. Available: https://doi.org/10.1007/s38314-018-0029-6

[49] S. Rao, "Introduction to mmwave Sensing: FMCW Radars," April 2017. [Online]. Available: https://www.ti.com/video/series/mmwave-training-series.html

[50] C. Li, W. Chen, G. Liu, R. Yan, H. Xu, and Y. Qi, "A noncontact fmcw radar sensor for displacement measurement in structural health monitoring," *Sensors*, vol. 15, no. 4, pp. 7412–7433, 2015. [Online]. Available: https://www.mdpi.com/1424-8220/15/4/7412

[51] V. Milovanović, "On fundamental operating principles and range-doppler estimation in monolithic frequency-modulated continuous-wave radar sensors," *Facta universitatis - series: Electronics and Energetics*, vol. 31, pp. 547–570, 01 2018.

[52] M. Richards, J. Scheer, J. Scheer, and W. Holm, *Principles of Modern Radar: Basic Principles*. Institution of Engineering and Technology, 2010.

[53] "AmplitudeEstimationAndZeroPaddingExample," Mar. 2023, [Online; accessed 11. Mar. 2023]. [Online]. Available: https://www.mathworks.com/help/signal/ug/amplitude-estimation-and-zero-padding.html

[54] E. Hyun and J.-H. Lee, "A new os-cfar detector design," in *2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, 2011, pp. 133–136.

[55] H. Rohling, "Radar cfar thresholding in clutter and multiple target situations," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-19, no. 4, pp. 608–621, 1983.

[56] ——, "Ordered statistic cfar technique - an overview," in *2011 12th International Radar Symposium (IRS)*, 2011, pp. 631–638.

[57] A. Melebari, A. K. Mishra, and M. Y. Abdul Gaffar, "Comparison of square law, linear and bessel detectors for ca and os cfar algorithms," in *2015 IEEE Radar Conference*, 2015, pp. 383–388.

[58] D. Trivedi, "Agile methodologies," *International Journal of Computer Science & Communication*, vol. 12, pp. 91–100, 04 2021.

[59] M. Zhuk, V. Kovalyshyn, Y. Royko, and K. Barvinska, "Research on drivers' reaction time in different conditions," *EasternEuropean Journal of Enterprise Technologies*, vol. 2, pp. 24–31, 04 2017.

[60] C. Wu, D. Yu, A. Doherty, T. Zhang, L. Kust, and G. Luo, "An investigation of perceived vehicle speed from a driver's perspective," *PLOS ONE*, vol. 12, p. e0185347, 10 2017.

[61] "Are car speedometers accurate? How accurate is my speedometer? | startrescue.co.uk," Jan. 2020, [Online; accessed 15. Jan. 2023]. [Online]. Available: https://www.startrescue.co.uk/breakdown-cover/motoring-advice/safety-and-security/how-accurate-is-my-speedometer

[62] "Rfbeam – leading supplier of planar radar sensors, k-band measuring equipment and engineering." https://www.rfbeam.ch/, (Accessed on 11/14/2022).

[63] "K-MD2 Engineering sample – RFbeam," Mar. 2023, [Online; accessed 5. Mar. 2023]. [Online]. Available: https://rfbeam.ch/product/k-md2-engineering-sample

[64] "K-MC1 Radar transceiver – RFbeam," Mar. 2023, [Online; accessed 5. Mar. 2023]. [Online]. Available: https://rfbeam.ch/product/k-mc1-radar-transceiver

[65] "uRAD Radar for USB - uRAD - Universal Radar," Jan. 2023. [Online]. Available: https://urad.es/en/product/urad-radar-usb

[66] Dipl.-Ing. f. C. Wolff, "Radartutorial," Nov. 2022, [Online; accessed 17. Nov. 2022]. [Online]. Available: https://www.radartutorial.eu/02.basics/Frequency%20Modulated%20Continuous%20Wave%20Radar.en.html

[67] "scipy.signal.find_peaks — SciPy v1.9.1 Manual," Aug. 2022, [Online; accessed 11. Sep. 2022]. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

[68] C. Katzlberger, "Object detection with automotive radar sensors using cfar algorithms," 2018.

[69] M. Gasior and J. Gonzalez, "Improving fft frequency measurement resolution by parabolic and gaussian spectrum interpolation," 11 2004.

[70] *Raspberry Pi 4 Model B Datasheet*, 1st ed., 2019.

[71] "Phased Array System Toolbox," 2022, The MathWorks, Inc. Natick, MA, USA. [Online]. Available: https://www.mathworks.com/help/phased/

[72] Dipl.-Ing. f. C. Wolff, "Noise, Noise Figure, Noise Temperature - Radartutorial," Aug. 2022, [Online; accessed 5. Mar. 2023]. [Online]. Available: https://www.radartutorial.eu/18.explanations/ex08.en.html

[73] "FMCWExample," Sep. 2022, [Online; accessed 10. Sep. 2022]. [Online]. Available: https://www.mathworks.com/help/radar/ug/automotive-adaptive-cruise-control-using-fmcw-technology.html

[74] "MATLAB Support Package for Raspberry Pi Hardware Documentation," Sep. 2022, [Online; accessed 15. Sep. 2022]. [Online]. Available: https://www.mathworks.com/help/supportpkg/raspberrypiio/index.html?s_tid=CRUX_lftnav

[75] "Matlab vs Python: 9 Comparisons For Which Language is Best for You," Oct. 2021, [Online; accessed 21. Sep. 2022]. [Online]. Available: https://blog.boot.dev/python/matlab-vs-python

[76] "Parallelising Python with Threading and Multiprocessing | QuantStart," Jan. 2023, [Online; accessed 14. Jan. 2023]. [Online]. Available: https://www.quantstart.com/articles/Parallelising-Python-with-Threading-and-Multiprocessing

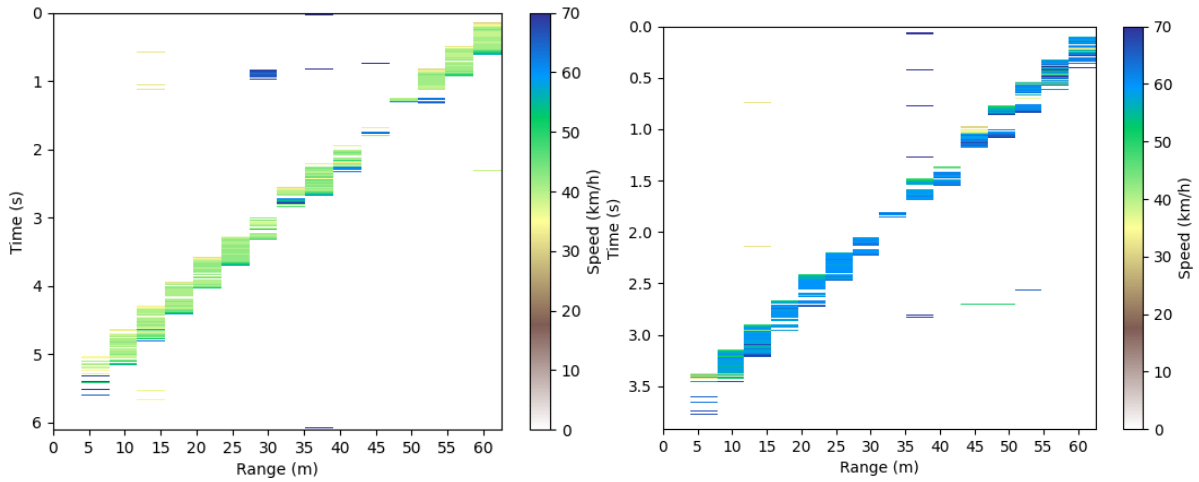
[77] "Faster rendering by using blitting — Matplotlib 3.6.0 documentation," Sep. 2022, [Online; accessed 23. Oct. 2022]. [Online]. Available: https://matplotlib.org/stable/tutorials/advanced/blitting.html

[78] "Chapter 2. The Remote Frame Buffer protocol," [Accessed 16. Mar. 2023]. [Online]. Available: https://docs.kde.org/trunk5/en/krdc/krdc/what-is-RFB.html

[79] C. Kaplinsky, "Introduction to TightVNC," Jan. 2023, [Online; accessed 20. Jan. 2023]. [Online]. Available: https://www.tightvnc.com/intro.php

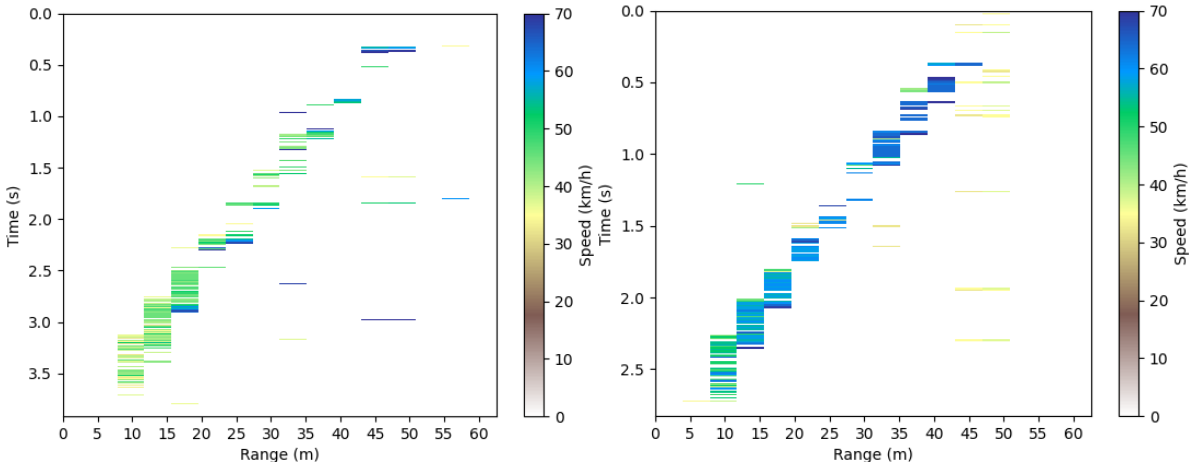
Appendix A

Additional Results



(a) Target travelling at 45 km/h, $P_{FA} = 0.004$ (b) Target travelling at 70 km/h, $P_{FA} = 0.004$

Figure A.1: Speed vs. Time vs. Range plots obtained by the right-side radar during controlled testing on a 7 m wide road using a Hyundai i20

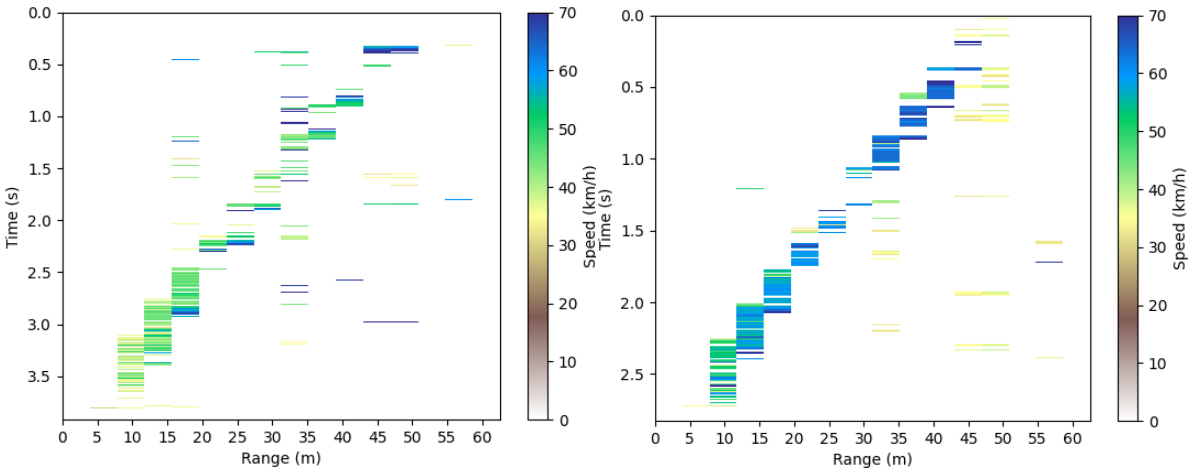


(a) Target travelling at 48 km/h, $P_{FA} = 0.004$, masked by bakkie

(b) Target travelling at 70 km/h, $P_{FA} = 0.004$, no masking

Figure A.2: Speed vs. Time vs. Range plots obtained by the left-side radar observing the outer lane during controlled testing on a 7 m wide road using a Hyundai i20

Figure A.3: Video frame of the target masking during the 48 km/h controlled test



(a) Target travelling at 48 km/h, $P_{FA} = 0.012$, masked by bakkie

(b) Target travelling at 70 km/h, $P_{FA} = 0.012$, no masking

Figure A.4: Speed vs. Time vs. Range plots obtained by the left-side radar observing the outer lane during controlled testing on a 7 m wide road using a Hyundai i20 where a higher false alarm probability was used to set the CA-CFAR threshold

Appendix B

Code

B.1 Repository links

The code repository is available on GitHub here:

https://github.com/dayalannair/RCWS

The Python implementation of the custom processing algorithm can be found at this location: RCWS/python_dsp/custom_modules/pyDSPv2.py

The final Python and MATLAB scripts for running the system and plotting the data and results can be found in this directory: RCWS/python_dsp/dual_urad_usb

MATLAB simulations can be found in this directory: RCWS/matlab_sim

B.2 Python Code Snippets

Listing 5 Code for obtaining the maximum update rate of the uRAD USB v1.2 radar module

```
for i in range(sweeps):
        return_code, raw_results = uRAD_USB_SDK11_v2.detection(ser)
        t_i.append(time())
        periods.append(t_i[i]-t_i[i-1])

elapsed = t_i[len(t_i)-1] - t_0
time_arr = np.array(t_i)
print("Elapsed time: ", str(elapsed))
print("Average period: ", str(np.average(periods)))
print("Average update rate: ", str(1/np.average(periods)))
uRAD_USB_SDK11_v2.turnOFF(ser)
```

Listing 6 Handling of the edge cells for CFAR detection

```
# if no training cells on the left of the CUT
if (cutidx <= half_guard):
    rhs_train = data[cutidx+half_guard:cutidx+lead]
    lhs_train = data[cutidx+lead:cutidx+lead+half_train]
# If some LHS cells, use these and take the remainder from RHS
elif (cutidx < lead):
    rhs_train = data[cutidx+half_guard:cutidx+lead]
    lhs_train = data[0:cutidx-half_guard]
    lhs_fill  = half_train-len(lhs_train)
    lhs_train = np.append(lhs_train, \
    data[cutidx+lead:cutidx+lead+lhs_fill])
# If enough train cells on either side of CUT
elif (lead < cutidx < lag):
    lhs_train = data[cutidx-lead:cutidx-half_guard]
    rhs_train = data[cutidx+half_guard:cutidx+lead]
# If some RHS cells, use these and take the remainder from LHS
elif (cutidx >= (ns-lead)):
    lhs_train = data[cutidx-lead:cutidx-half_guard]
    rhs_train = data[cutidx+half_guard:]
    rhs_fill  = half_train-len(rhs_train)
    rhs_train = np.append(rhs_train, \
    data[cutidx-lead-rhs_fill:cutidx-lead])
# if no training cells on the right of the CUT
elif (cutidx >= (ns-half_guard)):
    lhs_train = data[cutidx-lead:cutidx-half_guard]
    rhs_train = data[cutidx-lead-half_train:cutidx-lead]
```