Platform and Pipeline Development for a FMCW Radar System for Vital Sign Detection

Nicholas Bowden

Dissertation submitted to the University of Cape Town in fulfilment
of the requirements for the degree of
Master of Science in Engineering

Engineering and The Built Environment
University of Cape Town
South Africa
18 January 2024

Declaration

1. 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.

2. This thesis/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."

Name: Nicholas Keith Bowden
Signature: NKB
Date: 18 January 2024

Acknowledgements

I would like to thank both my supervisors, Dr Stephen Paine and Assoc. Prof. Amir Patel, for their unending support, guidance and encouragement. I learned so much from both of them and would not have gotten far without them.

I would then like to thank my partner, in life and love, Carla Coetzee for her tireless emotional support, compassion and understanding. She was always there and remained strong for me through the hardest parts of my work.

I would like to send my love and thanks to my family including my mother - Dr Lyn Horn, my father - Dr David Bowden, my brother - Dr Gregory Bowden and my aunt, uncle and cousins, Jackie, David, Stephen, Joan and Catherine for their love and support.

I would like to give special thanks my dearest and closest friends Jacques Van Wyk, Timothy Young and Liam Berry for always listening to me vent about my dissertation and providing the good times and good company I needed.

I would like to acknowledge the ARU and RRSG labs and their members for their help, guidance, friendship and resources without which the following research would not be possible.

Finally, I would like thank the Harry-Crossley Fellowship and SA College Croll Scholarship for their financial support which enabled me to focus on my research.

Abstract

Frequency Modulated Continuous Wave (FMCW) Radar has become a frequently examined technology for the purposes of ubiquitous vital sign monitoring applications. Vital sign monitoring of heart rate and respiration rate is important because it gives key physiological insights into the health of individual people. Having vital sign technology that allows for constant and ubiquitous monitoring would be of great benefit to people and physicians around the world.

Most vital sign research outputs focus on singular parts of the vital sign monitoring problem, often heavily relying on machine learning and enormous datasets of preprocessed data to detect vital signs and compensate for artefacts introduced by breathing and other motion. This dissertation details the design of a system from the ground up to collect raw and unprocessed data and then goes further to explain the design a processing pipeline to validate the data from the system. This provided maximum versatility and flexibility for future research outputs.

To validate the system and pipeline for vital sign detection, several sets of experiments were done with increasing complexity to identify points of failure within the pipeline. Complexity was added by adding layers of motion. First, the participant was in seated position and recordings were taken while the participant held his breath. Second, again in a seated position, recordings were taken while the participant was asked to inhale and exhale to visual cues. For these sets of experiments the pipeline performed well with accuracy ranging from 80% to over 90%. For the third set of experiments, the participant was asked to walk backwards and forwards during the recording session. Even after compensating for the movement, the accuracy of the system dropped significantly to below 60%. Compensation for large scale motion was achieved using a simple test rig by subtracting the known motion from the signal. However, the human walking motion was too complex to remove with just a simple subtraction. This complexity comes from the fact that walking requires multiple moving parts and these are all measured by the radar whereas with the rig, there is only one part that is moving.

After exhausting traditional filtering and other standard Digital Signal Processing (DSP) techniques, this dissertation concludes that future work should probably adopt a Machine Learning (ML) approach to compensate for complex motions such as walking.

Contents

Abbreviations

ADC Analogue to Digital Converter.

AoA Angle of Arrival.

API Application Programming Interface.

bpm Beats Per Minute.

CFAR Constant False Alarm Rate.

CLI Command Line Interface.

CPU Computer Processing Unit.

CUDA Compute Unified Device Architecture.

CUT Cell Under Test.

DC Direct Current.

DDS Data Distribution Service.

DNN Deep Neural Network.

DSP Digital Signal Processing.

eCAL enhanced Communication Abstraction Layer.

ECG Electrocardiogram.

EMD Empirical Mode Decomposition.

FFT Fast Fourier Transform.

FIR Finite Impulse Response.

FMCW Frequency Modulated Continuous Wave.

FoV Field of View.

FPGA Field Programmable Gate Array.

GPU Graphical Processing Unit.

GUI Graphical User Interface.

HDF5 Hierarchical Data Format v5.

ICEEMDAN Improved Complete Ensemble Empirical Mode Decomposition with Adaptive Noise.

IF Intermediate Frequency.

IIR Infinite Impulse Response.

IMU Inertial Measurement Unit.

IoT Internet of Things.

IQ In-Phase and Quadrature.

IR-UWB Impulse Radar - Ultra Wide Band.

JSON JavaScript Object Notation.

LVDS Low Voltage Differential Signalling.

M2S2 Multi-Modal Sensor-Suite.

MD Mode Decomposition.

MIMO Multiple Input Multiple Output.

ML Machine Learning.

N-RMSE Normalised - Root Mean Square.

NLLS Non-Linear Least Squares.

OS Operating System.

RC Resistor Capacitor.

RCS Radar Cross Section.

RGB-D Red, Green, Blue - Depth.

RLC Resistor Inductor Capacitor.

RMB Random Body Motion.

RMS Root Mean Square Error.

RMSE Root Mean Square Error.

RMW ROS2 Middle Ware.

ROS1 Robot Operating System 1.

ROS2 Robot Operating System 2.

rpm Respirations Per Minute.

RTOS Real Time Operating System.

RTPS Real Time Publish Subscribe.

SAR Synthetic Aperture Radar.

SDK Software Development Kit.

SIMO Single Input Multiple Output.

SNR Signal to Noise Ratio.

SoC System on a Chip.

SOS Second Order Section.

SPI Serial Peripheral Interface.

SSH Secure Shell.

TDM Time Division Multiplexing.

TI Texas Instruments.

UART Universal Asynchronous Receive Transmit.

UDP User Datagram Protocol.

VMD Variable Mode Decomposition.

YAML Yet Another Markup Language.

List of Figures

List of Tables

1 Introduction

This dissertation represents a body of exploratory research into embedded systems, Frequency Modulated Continuous Wave (FMCW) radar processing, vital sign monitoring. The hope for this research is to set a solid foundation for future research using FMCW radar in conjunction with other sensors in a variety of applications. In particular, a vital sign monitoring pipeline was implemented and validated using traditional Digital Signal Processing (DSP) techniques and the effects and compensation of Random Body Motion (RMB) are explored.

1.1 Background

In the world of engineering there are a plethora of applications that require systems to interface with the physical world. Lots of research is being done on combining multiple modes of sensing to provide a complete picture of the world for automated and/or interactive systems. Applications for these systems vary wildly, including automation tasks in industry, self-navigating robots, augmented reality, machine learning and even medicine, all of which often require multiple sensors to be integrated together [1]. As the world we live in becomes more and more integrated with these systems, sensing tasks operating ubiquitously throughout society Internet of Things (IoT) are becoming ever more present as they feed into the ever-growing pool of data we, as researchers, can draw from. So, when working with sensors, it becomes important that they are designed to be able to be integrated together.

In recent years, in the context of biomedical engineering, many researches have explored the use of FMCW radar to provide insights into the physiological state of people within an environment as shown by several papers [2, 3, 4, 5, 6, 7, 8]. FMCW has been looked at in particular, because it could enable the health of individuals to be monitored in unconstrained environments (i.e. environments outside the hospital bed and away from Electrocardiogram (ECG) machines) [9, 10]. However, a roadblock to fully realising this technology is that FMCW radar, in the context of vital sign monitoring, is extremely vulnerable to corruption from any movement of the body that is not cardio-pulmonary in nature [11]. Therefore, much of vital sign research is devoted to detecting, estimating and removing for this motion such as the papers in [4, 5, 6, 7]. There are numerous strategies to do this, all of which can benefit from multi-modal sensing (thus briefly highlighting, again, the importance of the ability to integrate multiple sensors together).

Non-contact vital sign monitoring systems, such as FMCW radar, are often accompanied by machine learning algorithms to aid in extracting heart rate as in the paper written by Zhao *et al.* [10] in 2020. However, machine learning (ML) approaches are data driven and thus need ground truth data to train off of. Therefore, first principle approaches that use traditional DSP techniques that are mathematically driven become desirable in the absence of large datasets. This dissertation seeks to create a traditional, non-ML pipeline to fill that gap in areas where large vital sign datasets are not available. Doing this will, therefore, also

provide a system that could potentially create ground truth data for future ML approaches in situations where traditional ground truth collection is not viable (such as vital signs of wild animals).

1.2 FMCW Vital Sign Monitoring

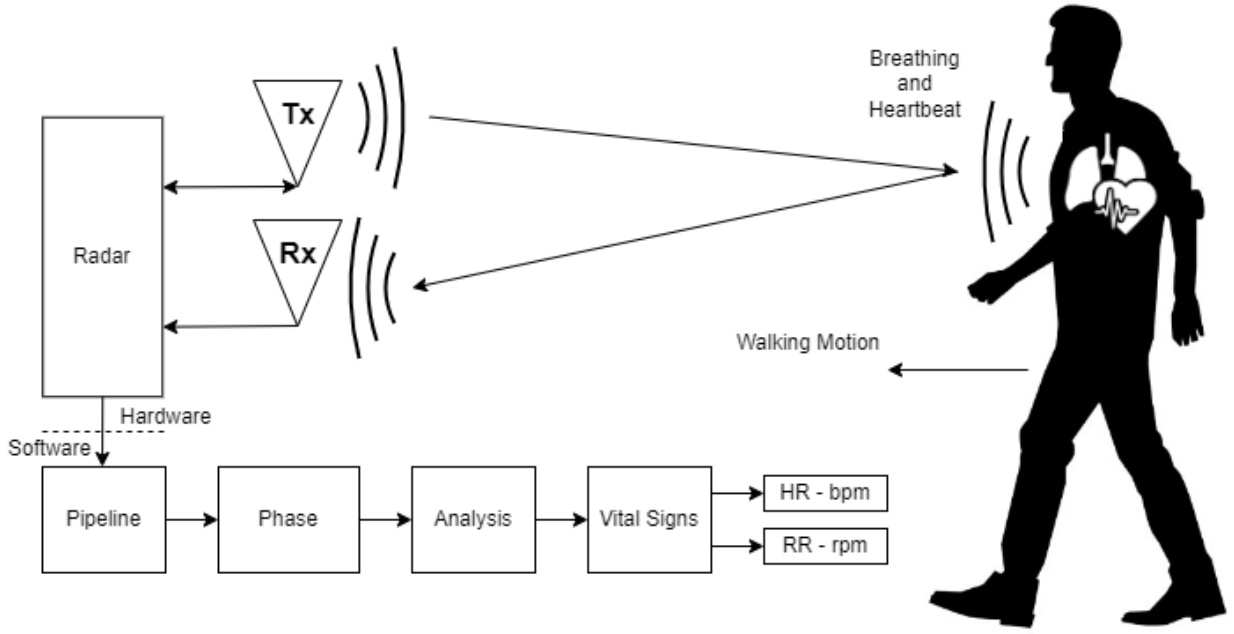

Figure 1: Diagram showing basic principle of FMCW Vital Signs. A human subject's breathing and heart rate are detected as motion relative to the radar by the system.

To perform vital sign monitoring, mm-Wave FMCW radar takes advantage of the extremely small wavelength associated with this class of antenna which, as the name suggests, is in the millimetre region. The radar emits high frequency "chirps" which are reflected off of a target and received by the radar as shown in Figure 1. Every received chirp has a phase associated with it that is directly proportional to the distance to the target and inversely proportional to the wavelength [9]. Therefore, because the wavelength is so small for this class of radar, the phase of the radar chirps is so sensitive to motion, movements of even less than a millimetre cause drastic changes to the phase. This high sensitivity to small motions is is what enables vital sign monitoring to be performed by mm-Wave FMCW radar because any cardio-pulmonary activity causes small movements in the skin which are encoded into the phase of each chirp [9]. Every radar sample is stored as a complex In-Phase and Quadrature (IQ) value where the in-phase value represents the real part of the complex sample and the quadrature component represents the imaginary part [2, 9]. The phase is calculated using trigonometry using the arc-tangent function in a process called arc-tangent demodulation as shown in Figure 2. Because the phase is circular in nature due to the geometry of the complex plane, phase values wrap around when they exceed plus or minus π radians. Therefore, before any processing can be done to extract the vital signs from the phase over time, the phase needs to be unwraps where the jumps caused by phase

wrapping are removed [2]. Analysing the frequencies contained within the final signal allow the heart rate and respiratory rate to be measured.

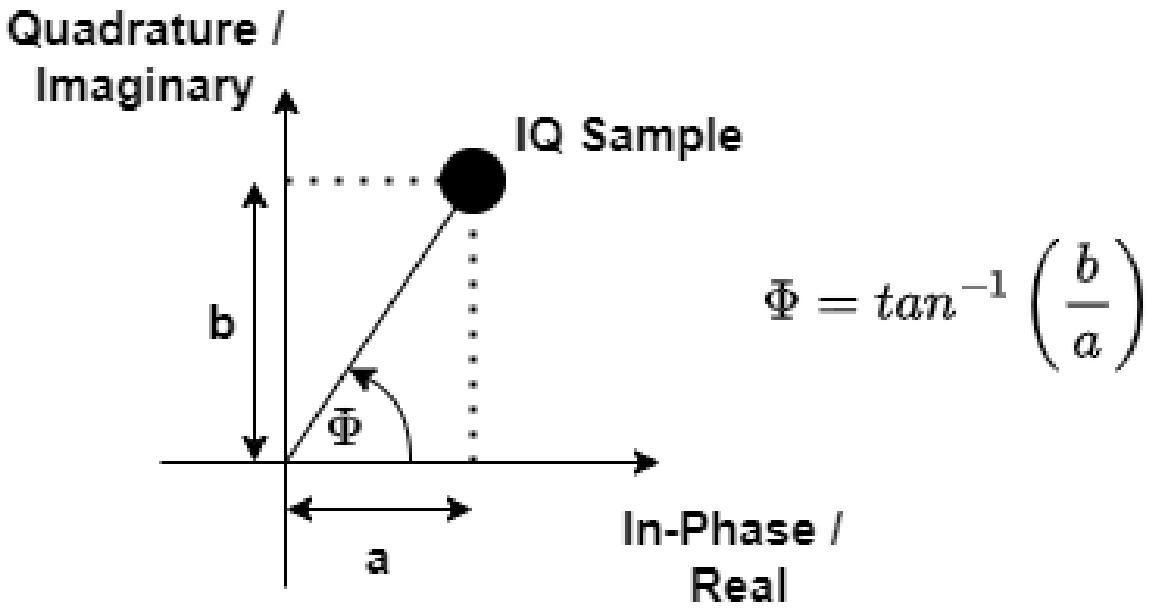

Figure 2: Diagram showing principle of arc-tangent demodulation. The size of the real and imaginary part are used to calculate the angle of the complex vector referred to as the phase.

The motion caused by the human subject in Figure 1, that is recorded by the radar, can be separated into three theoretical layers (this separation is sometimes easier said than done in practice). The first, and largest, layer is the gross motion of the person which are the large scale motions caused by walking or any movement not associated with cardio-pulmonary activity and can range from mere millimetres to multiple metres in size. These gross motions can be compound in nature as the human body is a multi-body system comprised of many different reflectors including the arms, legs and torso. This is often referred to as RMB in literature [9]. The second layer contains any motions associated with respiration (breathing) and are localised to the chest as the lungs expand during inhalation and contract during exhalation and are typically between 1 mm-3 mm in size. The third, and smallest, layer is the due to cardiac activity. These cardiac movements are ubiquitous throughout the body as blood is pumped to all areas and blood vessels expand and contract to accommodate the pressure changes that occur during systole and diastole and can range from hundredths to tenths of millimetres with the largest oscillations in the chest.

However, the high sensitivity that enables vital sign monitoring is also FMCW radar's greatest problem. The first, gross, layer of motion (RMB) often corrupts and overshadows the motions of the next two layers because of the disparity in sizes.

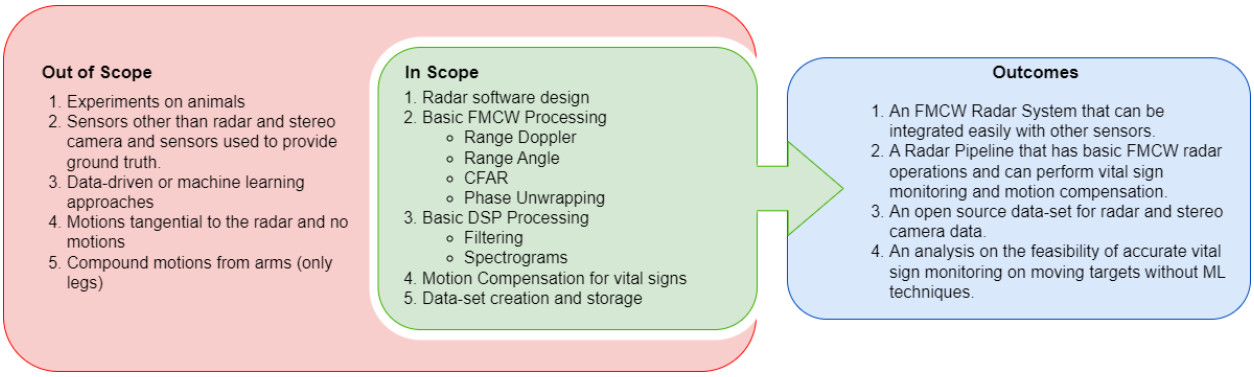

Figure 3: Diagram explaining the scope of the dissertation separated into out-of-scope, in-scope and outcome blocks.

1.3 Scope, Limitations and Outcomes

Vital sign monitoring has a very large project space and can refer to many measurements beyond heart rate and breathing including but not limited to temperature, blood oxygen saturation and blood pressure [12]. There are multiple approaches to extracting vital signs which broadens even further when starting to look at different motion compensation strategies. Therefore, it is important to divide the project space into what is in scope and out of the scope for the project.

In this dissertation, the radar software and pipeline design for vital sign monitoring is in the scope of the project. Particularly, for the vital sign monitoring, only phase reconstruction (representing the reconstruction of chest wall movements) will be examined. Motion compensation using radar and stereo camera sensors will be discussed (no other sensors will be looked at). Additionally, storage of the data for future work in an easily readable format will be part of the scope. Machine learning techniques, animals and complex compound motions will not be featured in the work because of time constraints. Additionally, only FMCW radars will fall in the scope of the dissertation as that is the hardware that is available.

This scope will produce several outcomes, as listed in Figure 3. The dissertation will produce a working radar system that can be integrated seamlessly with other sensors and a pipeline to process the data collected. The dissertation will produce several data-sets that can be used in future projects for machine learning or other applications. Finally, there will be a discussion on the feasibility of motion compensation strategies that are not reliant on machine learning techniques.

1.4 Dissertation Outline

The initial three chapters of this dissertation serve as a prelude to introduce the work done during the course of the research. This is done to give readers a sense of the direction and purpose of the research and what topics are dealt with within the dissertation. The next two chapters represent the majority of the work and give insights into how the results of the

dissertation were achieved. These two chapters represent the design sections of the dissertation. The final two chapters show and discuss the results obtained from the experiments using the work from the design section.

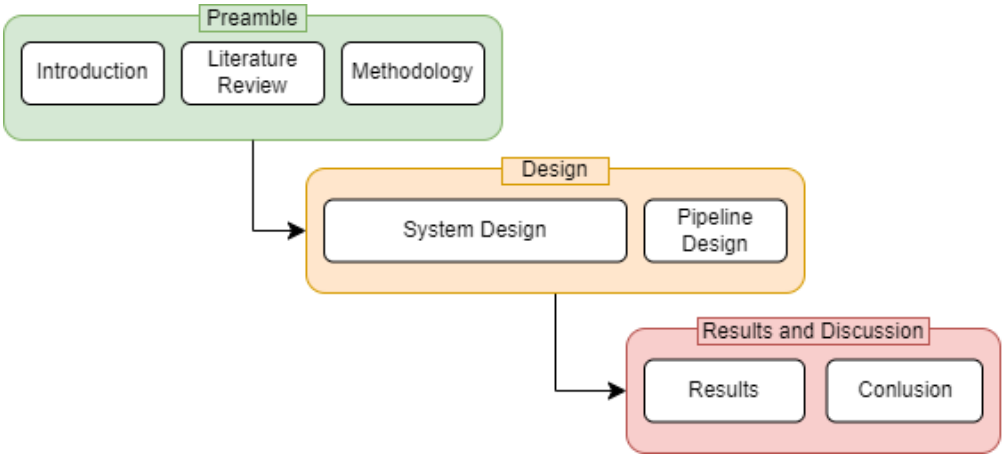

Figure 4: Figure of proposed outline of the dissertation moving the preamble through the system and pipeline design into the results and concluding sections.

The **Literature Review** will be a examination of the current state of the research to identify the foundations upon which to build the research conducted in this dissertation.

The **System Design** Section will go into detail about how the sensor system used to record data for the experiments was created. First it will focus on the hardware used and then move into the software development for the radar.

The **Processing Pipeline Design** Section will explain the basic concepts necessary to understand FMCW radar, stereo camera and digital signals processing concepts used in the pipeline to process the data gathered using the system created in **System Design**.

In the **Testing Methodology** Section, the procedures for conducting experiments, how the data will be validated and how the performance of the system will be characterised (what metrics will be used) will be explained.

In the **Results** Section, the outcomes of the data created with the processing pipeline will be shown. This will include a statistical analysis of the results.

The **Conclusion** will contain a discussion of the results presented before going on to identify areas for exploration in future work.

2 Literature Review

This section of this dissertation will examine the current state of vital sign monitoring research by identifying important knowledge and themes within the current literature. An important step to take, before exploring literature related to FMCW vital signs, is to understand why vital sign monitoring is so important, the traditional methods associated with vital sign monitoring and why a robust implementation of FMCW vital signs is useful.

The most common and important vital signs are temperature, pulse (or heart) rate, respiration (or breathing) rate and blood pressure [12, 13]. These measurements provide a fairly complete picture of physiological health and therefore are important indicators of stresses and/or sickness within the body [12]. However, temperature and blood pressure fall outside the scope of this dissertation.

2.1 Contact Based Vital Sign Monitoring

This section of the literature review is important for highlighting potential methods of developing ground truth data to test the proposed system against. One caveat with these kinds of methods is that they require a willing and still participant to be effective and can often only be used within a clinical or research setting [9, 12].

2.1.1 Traditional Methods

There is not much literature on the traditional methods because they are part of common knowledge and, as the earliest vital sign measurement methods, have been available for a long time. Despite this they are worth briefly mentioning to provide a complete overview.

Traditional methods utilise the human senses to determine vital signs. These include aural, haptic and visual methods. Aural methods include using a stethoscope or similar. Heart beats and respiration can be heard like this and the quality of the heart beat and breathing can be determined. Haptic measurements (i.e. by touch) such as the pulse fingering of arteries all around the body can be used to determine heart rate [13]. Heart and breathing rate can even be determined visually in some cases.

The usage of these types of measurements reveal two things. First, some vital sign signals are large enough to be determined visually. Second, vital sign signals such as the heart rate are ubiquitous throughout the body and cause movements large enough to be felt by the human touch.

(a) (b)

Figure 5: Figure of **a**: Digital oximeter [14]. **b**: Smart watch with pulse detection [15].

2.1.2 Photoplethysmography - PPG

Optical based measurements use a technique known as photoplethysmography [12]. A emitter and receiver pair of light sensors are usually placed close to the skin. The emitter sends out green light and the receiver measures the light reflected back. The amount of light reflected back changes based on which part of the cardiac cycle is currently active [12]. This difference allows photoplethysmography based devices to detect heart rate by measuring the periods between changes. Devices that use this technique include smart-watches and pulse-oximeters as shown in Figure 5. Such devices, such as a MAXREFDES103 Smart Watch, can be used to provide ground truth data for heart rate estimation which was done in the dissertation by Kukhokuhle Tswenga [9].

2.1.3 Electrocardiogram - ECG

ECGs, or electrocardiograms, are the current state of the art for contact based heart rate monitoring [9]. ECG Machines work by measuring potentials, with surface electrodes placed as shown in the left image in Figure 6, as they propagate through out the body [9]. These potentials create waves as shown in Figure 6. ECG machines are known to cause discomfort among patients due to the electrodes, the procedures to place them and the gel used to improve conductivity [9, 12]. The electrode placement is very important and so a good knowledge of anatomy is required to use such machine's correctly and so makes ECGs impractical outside of health-care or clinical settings [9]. Additionally, ECG machines have their own issues with signal quality which is described in a paper examining the effect of observation window sizes on ECG quality assessment ML networks by Tanantong *et al.* [16]. Motion of the patient can cause shifts in the electrodes which can cause changes in voltage leading to motion induced artefacts along with other issues [16]. The paper by Tanantong *et al.* [16] describes a sliding window method to evaluate the quality of the ECG signal. They go on to say the optimal window length varied between 5s and 10s. While describing ECG signal quality is well outside the scope of this dissertation, it does provide a potential

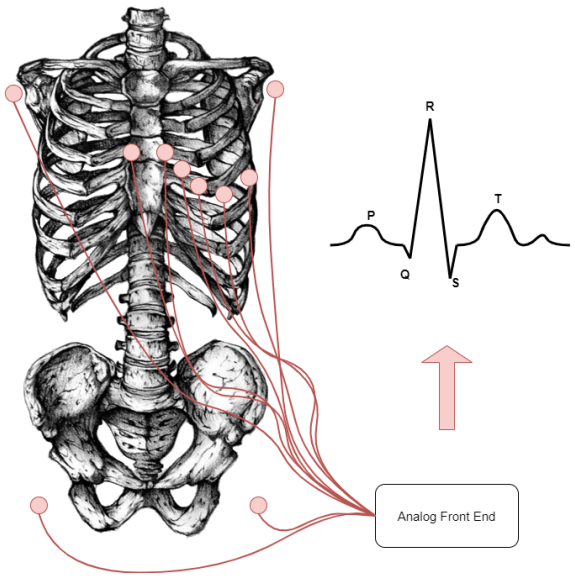

Figure 6: Placement of ECG electrodes on the human body and example of a single ECG wave output highlighting the P-wave, QRS-complex and the T wave. Modified from [9].

starting point for the selection of window sizes for analysis.

Heart rate monitors, such as the Polar H10 which are used in many papers, use a similar approach to traditional ECG machines and are often used in research as a cheaper, more comfortable and readily available alternative to ECG machines [10, 17, 18, 19, 20].

2.1.4 Respiratory Rate Measurement

Fewer methods are readily available for accurately measuring breathing rate. There are a few options discussed in a review paper about the potential of radar for vital sign measurements by Kebe *et al.* [12]. These include CO2, temperature and/or humidity measurements that can help determine breathing rate. These require medical masks and other specialist equipment. A paper about 120 GHz FMCW vital sign measurements by Wenjie *et al.* [21] uses such equipment. Another group of sensors mentioned in the review by Kebe *et al.* [12] refer to sensors that measure chest displacement including capacitive, inductive and piezo-electric sensors among others which again seem to require some kind of specialist equipment. However, the paper goes on to mention that a 3D movement sensor or Inertial Measurement Unit (IMU) on a belt is sometimes used [12]. This essentially describes the Polar H10 monitor mentioned previously as a way to measure breathing rate as it contains an IMU. This seems like an excellent way to validate respiratory rate data provided no other accelerations are present (i.e. the participant remains still). The dissertation written by Kukhokuhle Tswenga [9] uses a blinking LED to inform participants when to breathe in and breathe out. This conforms respiratory rate measurements to a known frequency which provides ground truth data by providing prior knowledge of the breathing frequency.

2.2 Non-Contact Based Vital Sign Monitoring

2.2.1 RGB-D Camera Vital Sign Monitoring

Papers by Zhao *et al.* [10] and Yang *et al.* [17] both make mention of using Red, Green, Blue - Depth (RGB-D) cameras to detect subtle fluctuations in skin tones which strongly correlate to heart rate. This approach is similar to the optically based photoplethysmography devices mentioned previously. Depth provided by a stereo camera can also measure displacements due to breathing [9]. The RGB-D vital sign approach can suffer due to a number of factors including varying lighting conditions and occlusion by clothes and hair.

Non-contact based optical methods are outside the scope of this dissertation and thus will not be explored further. However, it is important to take note of such research as it could be a powerful tool when combined with radar based vital sign monitoring.

2.2.2 Radar Vital Sign Monitoring

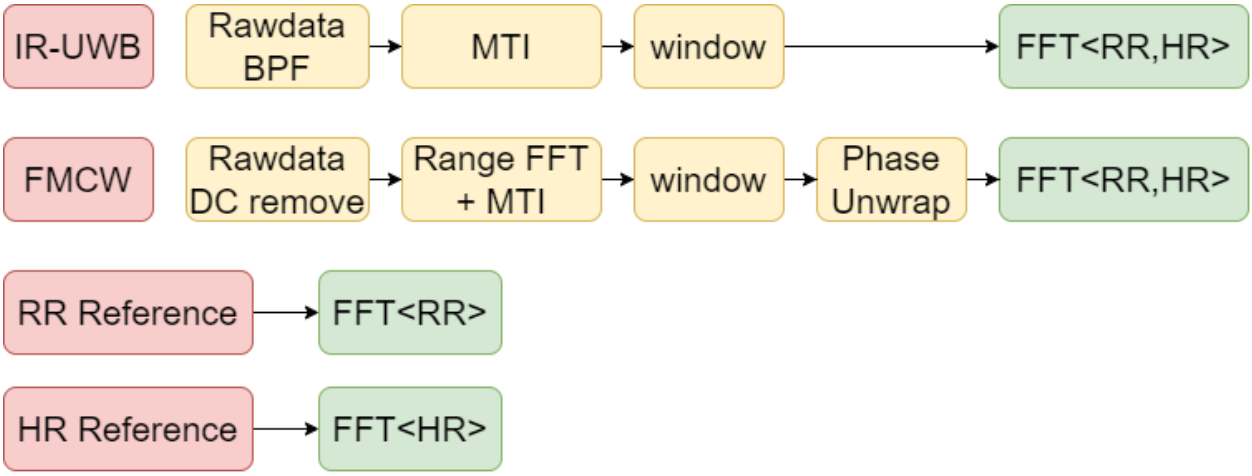

Figure 7: Comparison of basic pipeline between IR-UWB and FMCW pipelines taken from Wange *et al.* [22].

Vital sign monitoring with radar is split into two broad categories: FMCW radar and Impulse Radar - Ultra Wide Band (IR-UWB) radar which are compared for vital sign monitoring in a paper by Wang *et al.* [22]. This study concluded that IR-UWB had slightly higher Signal to Noise Ratio (SNR) and better accuracy but was far more expensive and complex [22]. The two systems work on different principles. As described in an analysis by Lazaro *et al.* [23] and the comparison by Wang *et al.* [22], IR-UWB radar is a type of pulsed radar that has a very large bandwidth (i.e. a very narrow pulse in time) which yields an incredibly small range resolution. This small range-resolution allows the measurement of sub-millimetre displacements associated with vital signs. However, IR-UWB pulse radar is not part of the scope of this dissertation and therefore will not be discussed further. FMCW does not directly measure the displacement as with IR-UWB but instead utilises the sensitivity to motion of the phase difference between successive chirps [9, 21, 22].

In the paper by Wenjie *et al.* [21], not only do the authors describe the expected rates, they go on to state the amplitudes of breathing and heart rate measurements as 2 mm and 0.3 mm respectively. In the same paper, they state the statistically probable maximum rate for breathing rate and heart rate is 0.67 Hz and 3 Hz respectively [21]. This is confirmed in the paper by Alizadeh *et al.* [3] and documentation by Texas Instruments (TI) [24] where they state that the typical range for breathing amplitude is 1 mm to 12 mm and 0.2 Hz to 0.5 Hz for frequency and the typical range amplitude range for heart beats is 0.1 mm to 0.5 mm and 0.8 Hz to 2 Hz for frequency. The authors in [21] go into a fair amount of detail about their hardware which many papers do not. Very few papers also include the chirp and frame parameters used to collect the data. The paper by Iyer *et al.* [25] breaks the trend and provides the chirp parameters and the system they used. However, the authors in [25] like many others, use the default TI pipeline and then apply machine learning to classify data is normal sinus rhythm or as an arrhythmia. They did not develop their own data collection and phase unwrapping and analysis pipeline. A 2022 paper by Upadhyay *et al.* [20] uses a TI DCA1000 raw data capture module to collect raw data using TI's raw data capture tool which is fairly inflexible and relies on third party software. However, this paper does a good job of highlighting the parameters used for the radar, explaining their algorithm and showing the spread of data they collected and the corresponding results. The spread of their data is very small, only containing resting heart rate measurements. Additionally, this paper does not delve into any motion compensation but does look at breathing harmonic rejection as in a paper by Tu *et al.* [26].

Motions due to vital signs are not always purely sinusoidal [2]. This is also stated by Fouladi *et al.* [27] in 2013, where they say some researchers model the breathing rate as a response to an Resistor Capacitor (RC) circuit as described by another by Zhang *et al.* [8]. As the air pressure in the lungs increases, the rate of chest expansion during inhalation slows and likewise for exhalation while the lungs empty, similar to a capacitor charging and discharging. Regardless of what causes the deviation from the ideal sinusoid, the non-sinusoidal nature of breathing causes harmonics in other parts of the spectrum These harmonics can overlap the spectrum peaks due to heart rate. The paper by Zhang *et al.* [8] also goes on to attempt to model the heart beat as the output of a Resistor Inductor Capacitor (RLC) circuit as the authors state that the heart beat is also not purely sinusoidal.

For FMCW platforms a frequent step performed in the radar pipeline is Direct Current (DC) offset compensation [3, 9, 22, 28]. A DC offset in the IQ samples, caused by differences in the In-phase and Quadrature Analogue to Digital Converter (ADC) components of the radar receiver, can cause erroneous unwraps during the phase reconstruction process [9]. Several approaches for removing the offset are suggested by Tswenga but he states that subtracting the mean of all the samples works best [9]. The paper by Alizadeh *et al.* [3] use a Non-Linear Least Squares (NLLS) approach to estimate and remove the offset. The DC compensated samples are then usually subjected to the arc-tangent demodulation algorithm to extract the phase [9, 29].

The paper by Zhao *et al.* [10], shows a comparison of different analysis methods for a robot mounted FMCW radar. They conclude a Deep Neural Network (DNN) operating on

preprocessed phase data (data being collected directly from the radar) provided the best accuracy (over 90%). However, the radar was positioned extremely close to the participant ankle. This provided good SNR by minimising the target range, reduced the chances of other targets being in the same range bin interfering with the measurement and reduced the complexity of the measurement because the respiratory interference mentioned by Alizadeh *et al.* [3] is localised to the chest. Additionally, the authors in [10], do not make any mention of the spread of the data used to train the network. If all the data was collected while the participants were at a near resting heart rate then it is quite possible the machine learning network could have learned to output a narrow range of values that ensure high accuracy for the dataset. A model like this would immediately fail if it encountered a heart rate outside the range of values it was trained on. Therefore, it is important to talk about the spread of data used when evaluating vital sign detection performance. The authors did include a study of how the reflecting surface is angled and how that angle can severely degrade measurement quality if too far away from the perpendicular.

2.3 RMB Compensation Strategies

The most challenging problem with FMCW radar vital sign monitoring is dealing with motion artefacts caused by RMB or any other kind of gross motion not associated with cardio-pulmonary activity [11, 30]. The radar hardware shown in red in Figure 8 conceptually should capture all motions caused by the body including cardiac and respiratory motion. The aim of motion compensation is to use another sensor or some kind filtering to measure the RMB signal to generate a compensation signal that can be removed from the output of the radar. Provided the respiratory and cardiac motion is not captured by the compensating signal, the output after compensation should only contain the vital signs of interest.

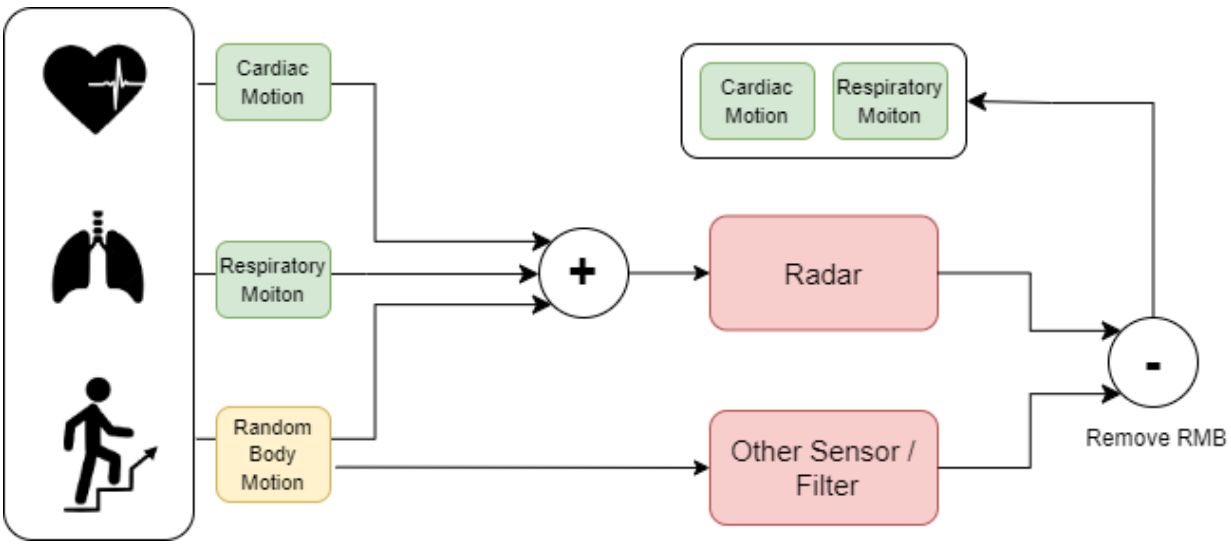

Figure 8: Diagram explaining principle of motion compensation. RMB is measured by a separate sensor and removed to yield only vital sign signals.

2.3.1 Single Radar Compensation

In a paper by Hu *et al.* [11], they use a range-time profile to identify the path taken by a participant. The authors only loosely touched on the system design and processing they used and focused more on how they compensated for walking motion. They did not actually do any compensation per say, but instead used Mode Decomposition (MD) to separate the gross motion from the vital sign motion. A paper by Shuai *et al.* [31] uses similar MD approach. In a paper by Yang *et al.* [7], a single radar is used with an adaptive filtering algorithm to remove random body motion. These motions were in the order of 0.1 m.

2.3.2 Dual Radar Compensation

In a paper by Jang *et al.* [4], researchers use two radars placed in front of and behind the participant. Vital signs present themselves as displacements in the skin that move away from the body centre. Therefore, the forward and back radar record the same oscillation because the movement happens outward. However, gross motion is not bi-directional. This causes the two radars to measure opposite signals. Adding the two measurements together results in the gross motion being cancelled out and the sensitivity to vital sign motions being doubled [4]. However, this method still requires a constrained environment where the participant can be made to stand between the two radars and the extra hardware creates a lot of extra complexity from a system's implementation point of view.

2.3.3 Stereo Camera Compensation

In papers by Wang *et al.* [30] and Yang *et al.* [17], the authors combine cameras to perform better measurements. However, the first paper [30], once again, uses some kind of MD instead of compensation to remove gross motion while the other paper [17] develops a novel "graph segmentation" method for providing robust vital sign monitoring. In Tswenga's dissertation [9], he suggests cancellation of RMB can be achieved by using a depth camera, or any other 3D sensor, to measure the RMB signal to generate a compensation signal.

2.4 Summary

Ground truth data for non-contact vital sign monitoring can be produced in numerous ways. In clinical settings, there is access to gold standard devices such as ECG machines and breathing monitors. However, Polar H10 monitors are commercially available, an excellent way to validate heart rate and are often used in research. Additionally, the Polar H10 monitor is a good way to validate breathing provided participant is stationary. When the participant is not stationary, visual cues can be used to validate breathing rate measurements by conforming breathing to a known respiratory rate.

The literature discussed in this section does provide estimates for amplitude and frequency.

These are 0.2 Hz to around 0.6 Hz for frequency and 2 mm to 12 mm for amplitude for breathing rate. For heart rate, these ranges are 0.8 Hz to 3 Hz and 0.2 mm to 0.5 mm for frequency and amplitude respectively. The amplitudes listed are important for validating the vital signs measured as we can compare them to literature. The frequencies listed are also important and they provide well founded estimates for filter cutoff frequencies.

In terms of compensation and signal analysis, the current state contains a mixture of time and frequency domain techniques. Various MD algorithms are used for time analysis and removing gross motion while spectrograms and FFTs are typically used for frequency analysis. Literature says a window size of 5-12 seconds is applicable for FFT analysis [10]. A large portion of literature uses some kind of deep learning network to improve results as in the papers by Zhao *et al.* [10] and Iyer *et al.* [25]. Traditional DSP approaches to compensation are not often taken but possible and cover gaps in research where there is no opportunity for ground truth data for data driven approaches such as machine learning. Direct compensation by subtraction with additional sensors as suggested by Tswenga [9] has not actually been explored in literature much. Furthermore, many papers use TI radar boards that output preprocessed phase data as in the papers by Zhao *et al.* and Iyer *et al.* [10, 25] or use TI's mm-Wave evaluation tool mmWaveStudio to capture data [20].

There are very few papers, if any, that give a complete picture of the vital sign detection process starting from system design and implementation to the final results. Usually, papers focus on one aspect such as the basic algorithm, machine learning or motion compensation. Almost no papers mention or show the spread of their data and the ones that do have the heart rate in a very narrow band. Even fewer papers detail system design. Therefore, there is a gap in the literature whereby an entire overview of vital sign detection is performed including system design and validation on a wide range of data and motion compensation with external sensors.

3 System Design

During the course of this dissertation, several stages were completed including: hardware and software system design, processing pipeline design, experimental design, recording and processing and finally validation against ground truth data. Section 3 details the hardware and system design. This system was needed in order to record data in a flexible and easy to use way that was not bound by complication introduced by using third party TI software.

3.1 Hardware

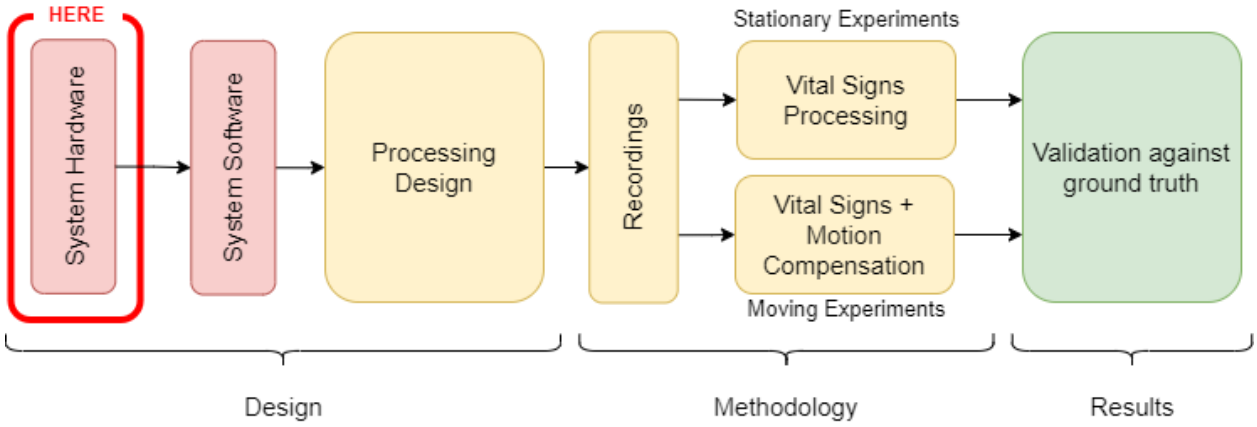

Figure 9: Figure of dissertation layout: Hardware Design.

The following subsections detail what hardware was used for the data capture system and the hardware specifications.

3.1.1 The Multi-Modal Sensor-System (M2S2)

The radar software that was used to collect data in this dissertation was developed alongside several other sensors as part of a larger project called Multi-Modal Sensor-Suite (M2S2) for a paper by Vally *et al.* that is currently under review [32]. The other sensors integrated into M2S2 included: a LiDAR, Event, Thermal, RGB and Stereo cameras and a parabolic audio antenna as shown in Figure 10. The M2S2 project integrated these sensors together to create highly diverse and rich data-sets that could serve many purposes such as pose estimation, vital sign monitoring, drone classification and so on as shown in Figure 11.

While most of the other sensors are outside the scope of this dissertation, it is important to mention them to put the following sections into context. Working within the scope of the M2S2 project created several requirements that guided the development of the radar software. A lot of work was done on the radar software to make it flexible enough to be integrated with the other sensors for future work within the research group.

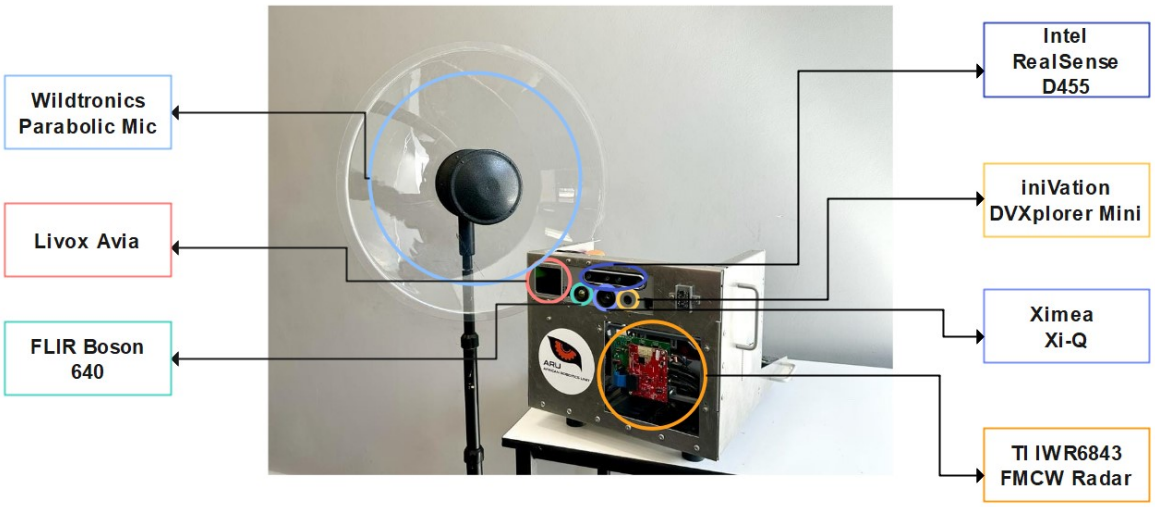

Figure 10: Figure of labelled M2S2 hardware taken from a paper under review by Vally *et al.* [32].

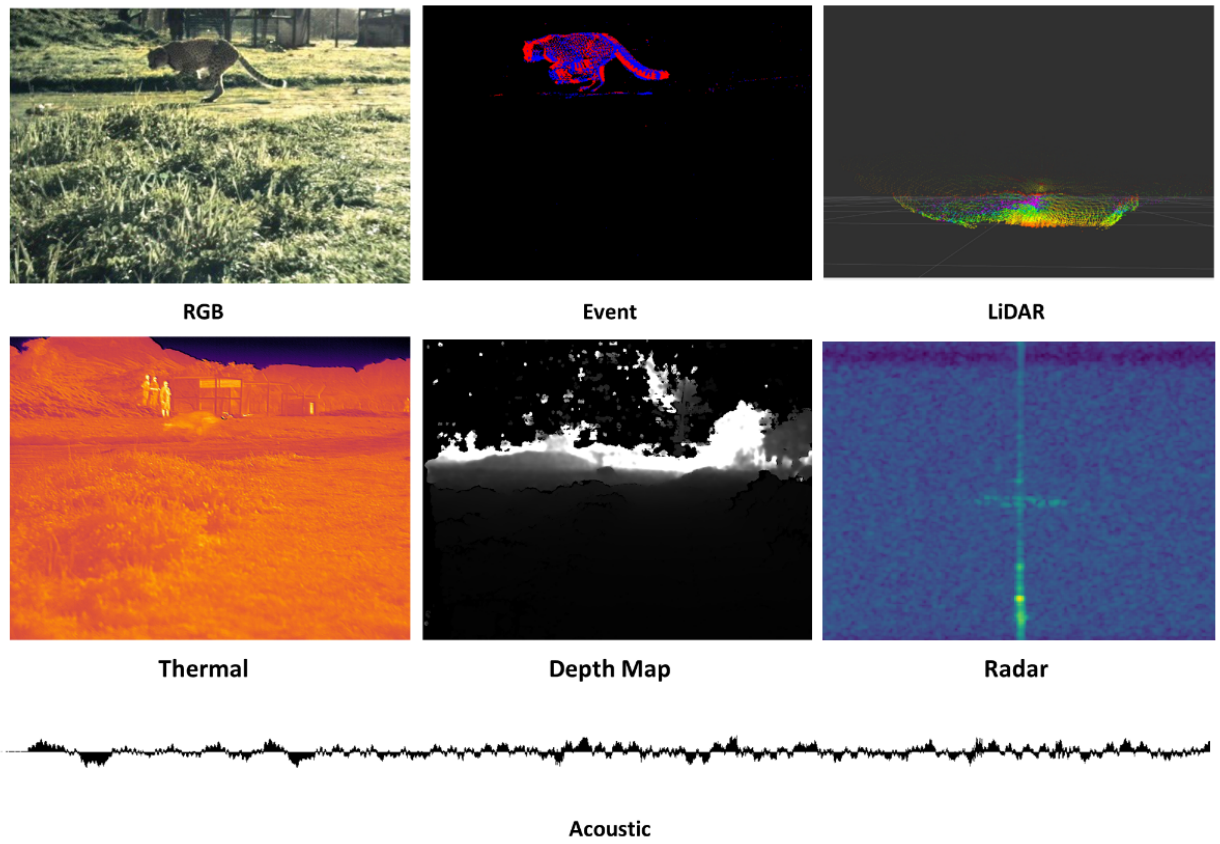

Figure 11: Example of multi-modal data from M2S2 taken from a paper under review by Vally *et al.* [32].

3.1.2 Intel RealSense Camera

Figure 12: Diagram of RealSense D455 Stereo Camera.

The stereo camera mentioned in Section 3.1.1 was an Intel D455 RealSense Stereo Camera. The setup for this was relatively simple as it already had highly performant Robot Operating System 2 (ROS2) drivers that allowed it record data two uint8 infra-red cameras, a 3 channel unint8 RGB camera and a uint16 depth map reliably at 30Hz.

It was used in this dissertation to aid in range bin selection in the instances where conventional radar detection algorithms struggled and to validate the results of the conventional algorithms were correct.

3.1.3 Texas Instruments FMCW Radar

The radar chosen for the system was a TI 77-81 GHz FMCW radar. It was chosen due to its highly configurable design which makes it versatile and well suited to many applications. The radar can be configured to emit almost any chirp design provided the chirp frequency remains within the radar 4 GHz Bandwidth between 77 GHz and 81 GHz. Additionally, the antenna pattern allows for Multiple Input Multiple Output (MIMO) configuration and beam-forming as well as range-azimuth and point cloud processing. The radar can also be made to operate in several modes depending on the firmware flashed to the radar. One of these modes included a raw ADC data capture mode (provided the correct carrier board, the DCA1000, was used) which was perfect for the purposes of this dissertation.

The radar system described above can be broken up into several parts:

1. The radar module (Figure 13) which included:

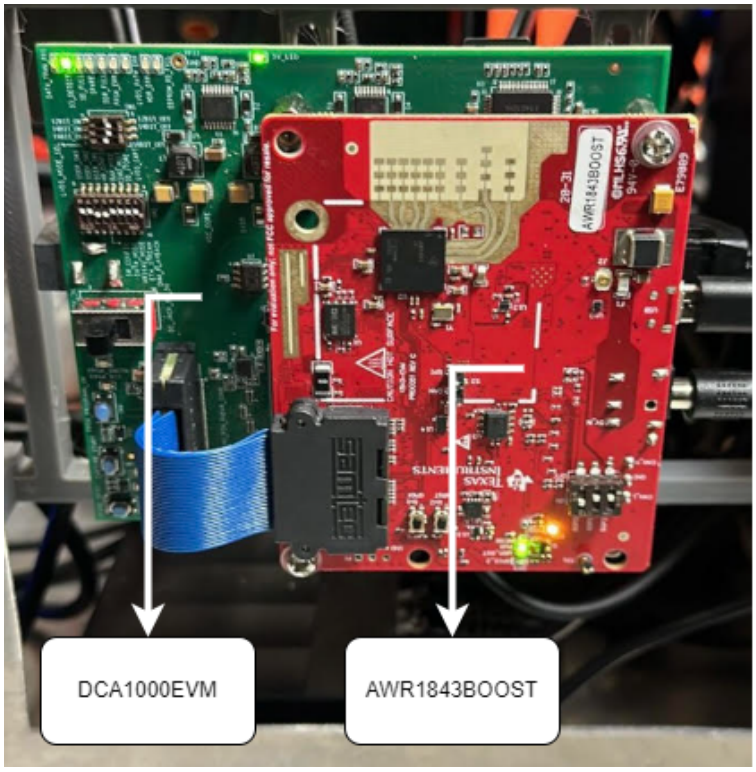

Figure 13: Image of DCA1000EVM (Green Board) and the AWR1843BOOST (Red Board).

(a) One TI DCA1000EVM data capture board (The green board in Figure 13). This board was responsible for capturing raw ADC samples and sending the data over an Ethernet cable to the host system in a particular format.

(b) One TI 77 GHz AWR1843BOOST radar board (The red board in Figure 13). This board contains the radar antennas and a System on a Chip (SoC) responsible for shaping the chirp sent by the antennas based on parameters given to the radar via a Universal Asynchronous Receive Transmit (UART) Interface. This board is "plugged" into the DCA1000.

2. The host processing system: Intel i7 NUC.

3. An Ethernet switch to route any commands and/or data between the NUC and DCA1000 (and the LiDAR). A TP-Link LiteWave 5 Port Gigabit Desktop Switch was used.

The hardware above required the following I/O:

1. One 5V supply capable of supplying up to a max of 1A for the Radar.

2. One 12V supply for the NUC.

3. One Ethernet port for DCA1000, one Ethernet port for the NUC (and one Ethernet port for the LiDAR). The dumb Ethernet switch mentioned above provides all the required ports.

4. One UART serial port for the AWR1843 Radar module (USB).

3.1.4 Polar H10 Heart Rate Monitor

Figure 14: Image of Polar H10 heart rate monitor.

The Polar H10 is a standard heart rate monitor that is often used in research and is readily and commercially available. It uses electrodes and a proprietary algorithm to detect electrical activity in heart. The peaks in electrical activity correspond to peaks in successive QRS complexes (heart beats). It calculates the time between these peaks (the inter-beat-interval or T_{IBI}) to determine the frequency in beats per minutes [Beats Per Minute (bpm)] where:

$$f_{\text{heart}} = 60/T_{\text{IBI}}\ [bpm] \tag{1}$$

An existing GitHub repository was used as a basis to establish Bluetooth communications between the host PC and the Polar H10 [33]. This was re-written within a ROS2[1] node to generate timestamped, ground-truth, heart rate data.

3.1.5 Radar Test Rig

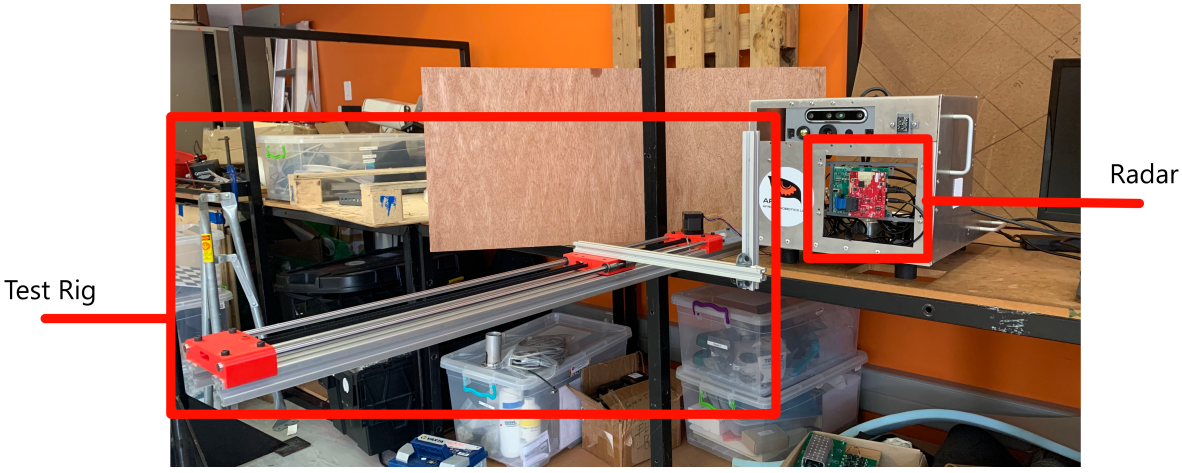

Figure 15: Picture of the designed test rig.

The test rig shown in Figure 15 was used as a validation tool. It was created using 3D printed parts, aluminium extrusions, stainless steel rods and linear bearings and driven by a NEMA17 stepper motor and belt drive with several pulleys.

[1]ROS2 is discussed further in the following Sections

Two perpendicular aluminium extrusions were used to hold up a corner reflector in front of the radar. The corner reflector was actually removed because represented such a massive target that close and the aluminium extrusions provided more than enough Radar Cross Section (RCS). The platform the aluminium extrusions were mounted to was driven by the stepper motor which was controlled by a A4988 stepper driver in full step mode. The pulleys used resulted in the platform moving 4 cm per revolution with 400 steps per revolution in full step mode. This meant that each step moved the platform 0.1 mm.

The A4988 received its step and direction commands from an STM32 micro-controller which was controlled via a USB serial interface handled by a ROS2 node.

3.2 Software

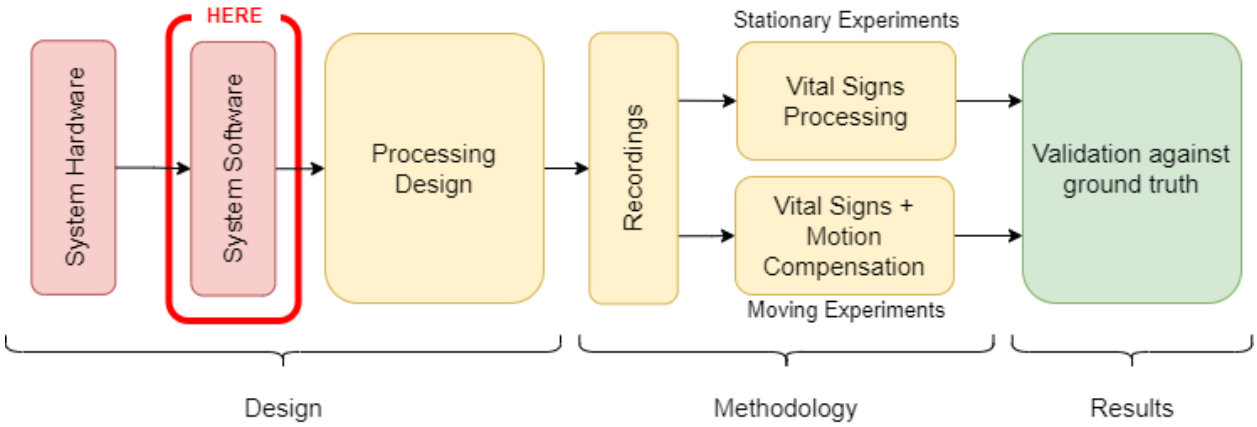

Figure 16: Figure of dissertation layout: Software Design.

The following subsections explain how the software for the system was created and what functionality the software provided, including: ROS2 function and data flow and storage.

3.2.1 Radar Software Requirements

For the data-sets mentioned in Section 3.1.1 to be useful, all the data had to be timestamped so they could be time-synchronised. To achieve this, ROS2 was used. Additionally, as development time was restricted, Python (which is supported by ROS2) was used to prototype all the code. Python code was later replaced by C++ where performance improvements were required.

ROS2 is essentially a collection of client libraries and middle-ware that facilitates a standardised way to develop code within a Real Time Publish Subscribe (RTPS) framework. This enables the seamless integration of sensors and actuators for robotic systems. So, naturally, the data synchronisation is built into ROS2's core functionality which is why it was used. Additionally, ROS2 is quickly becoming widely used with many sensors and actuators already having ROS2 drivers. Therefore, using ROS2 to develop the radar software made it much more useful because any other project or group would be able to integrate the radar into their own ROS2 project.

The use of ROS2 posed certain requirements:

1. The system had to run on Ubuntu 20.04 (the distribution used by M2S2 was ROS2 Foxy-Fitzroy which had the best support on Ubuntu 20.04).
2. The radar was required to have a ROS2/Robot Operating System 1 (ROS1) node to manage its setup and data collection[2].

[2]A ROS1 node would have been acceptable as there is a software package called ROS2 Bridge which

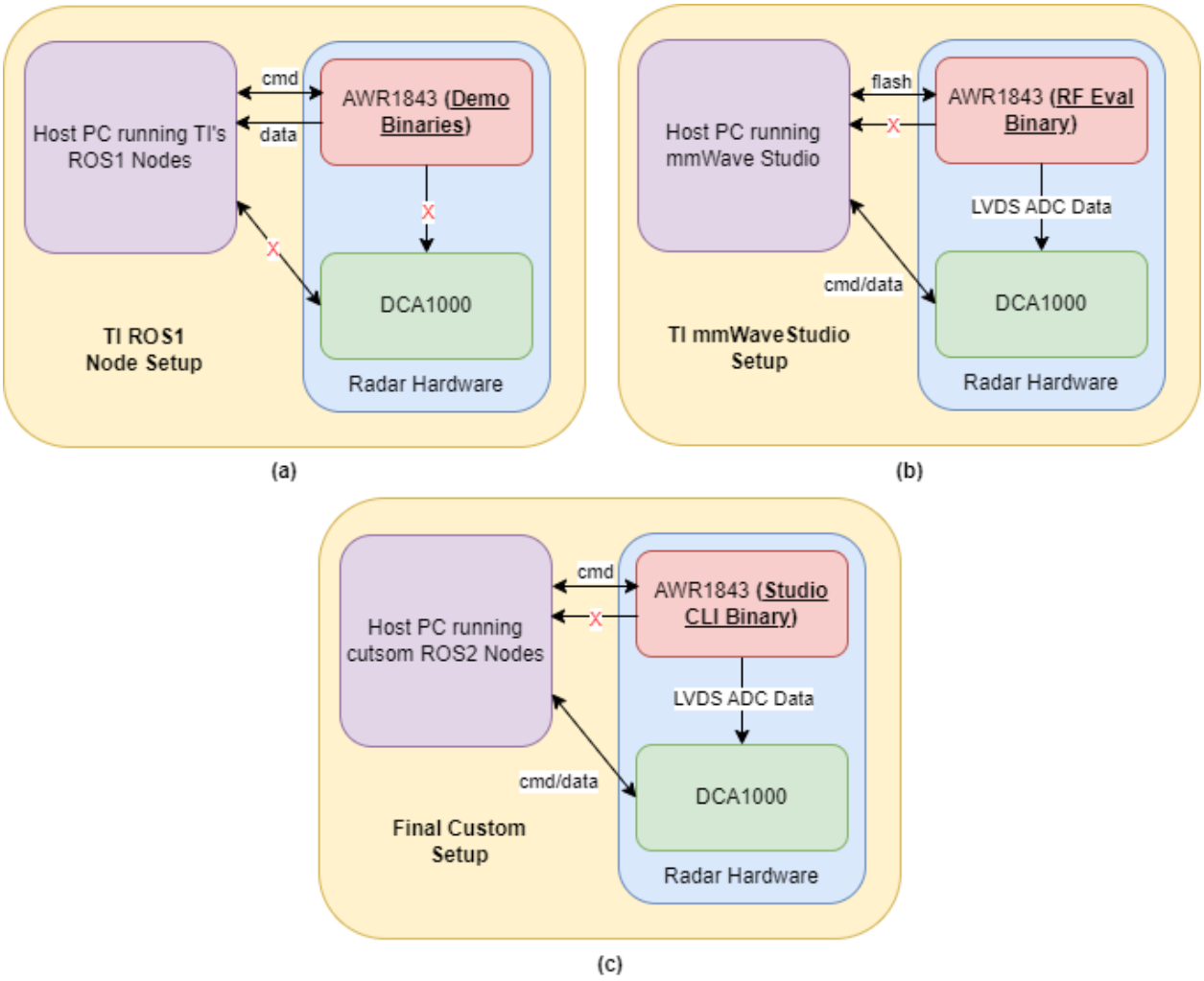

Figure 17: (a) The TI ROS1 Setup which does not enable raw ADC data capture. (b) The TI mmWaveStudio Setup which enables raw ADC data capture but without ROS2. (c) The chosen setup which enables both raw ADC capture with ROS2.

The development of the radar within the M2S2 project posed its own requirements as well:

To maximise the flexibility of the radar and to ensure the richness of the data-sets generated by M2S2, raw, unprocessed ADC data needed to be captured from the radar. As mentioned in Section 3.1.3, the DCA1000 is required for raw ADC data capture. The two sets of requirements above created several conflicts which are discussed below.

The majority of firmware (called binaries by TI) written for the radar board configured it as a standalone device (without the DCA1000). These "demo binaries", as they are often referred to by TI, were required to use TI's ROS1 node. This setup is shown in Figure 17(a). In this configuration, the DCA1000 is not used at all and instead the radar is configured over a UART command port while processed data is sent over a 2nd UART data port. The processing applied to the data is dependent on the application the demo binary was made for. So, in this setup, the flexibility of ROS2 is incorporated but the raw data capture is lost.

The software designed by TI for the DCA1000 is called mmWaveStudio. This software is really only meant for "easily" testing different configurations before settling on a final one or running experiments by hand but it collects raw data as required. This software offers no integration with other sensors and is only available on Windows which makes it extremely inflexible to use. This setup, as shown in Figure 17(b) flashes the rf-eval binary to the radar. This binary comes with mmWaveStudio and configures the radar to be controlled and configured by an Serial Peripheral Interface (SPI) port on the DCA1000. Raw data from the radar is sent over Low Voltage Differential Signalling (LVDS) lanes which is then sent over Ethernet to mmWaveStudio. This configuration satisfies the raw data requirement but completely fails in terms of flexibility.

The binary/firmware flashed to the radar is very important as it determines how it interacts with other hardware and what configuration commands are available to the radar. In the above cases the "demo binaries" allow for standalone control of the radar but does not enable sending data over the LVDS lanes to the DCA1000, while the rf-eval binary allows for raw data capture but makes the radar board impossible to work with outside of mmWaveStudio. The radar SoC runs an Real Time Operating System (RTOS) with the main components of the driver written in C/C++. Technically, a custom binary could have been made and flashed to the radar that allows for both standalone configuration and raw data capture but this required a highly detailed knowledge of TI's mmWave Application Programming Interface (API) and a RTOS skill-set. Fortunately, there is one set of firmware written for the radar provided by Texas Instruments based off of their studio Command Line Interface (CLI) tool which configures the radar board to send raw data to the DCA1000EVM while maintaining the board as a standalone device that can be controlled and configured as such. The studio CLI binaries can be found under under StudioCLI under Tools in the TI mmWave Software Development Kit (SDK). The Studio CLI Binary is the binary that was used for

allows ROS1 and ROS2 middle-ware (ROS2 Middle Ware (RMW)) to communicate with each other and run concurrently. But ROS1 is being phased out, so ROS2 was preferable

the development of the radar software. The one disadvantage of this binary was that it had no ROS2 support, so ROS2 nodes had to be developed.

3.2.2 Radar Interface

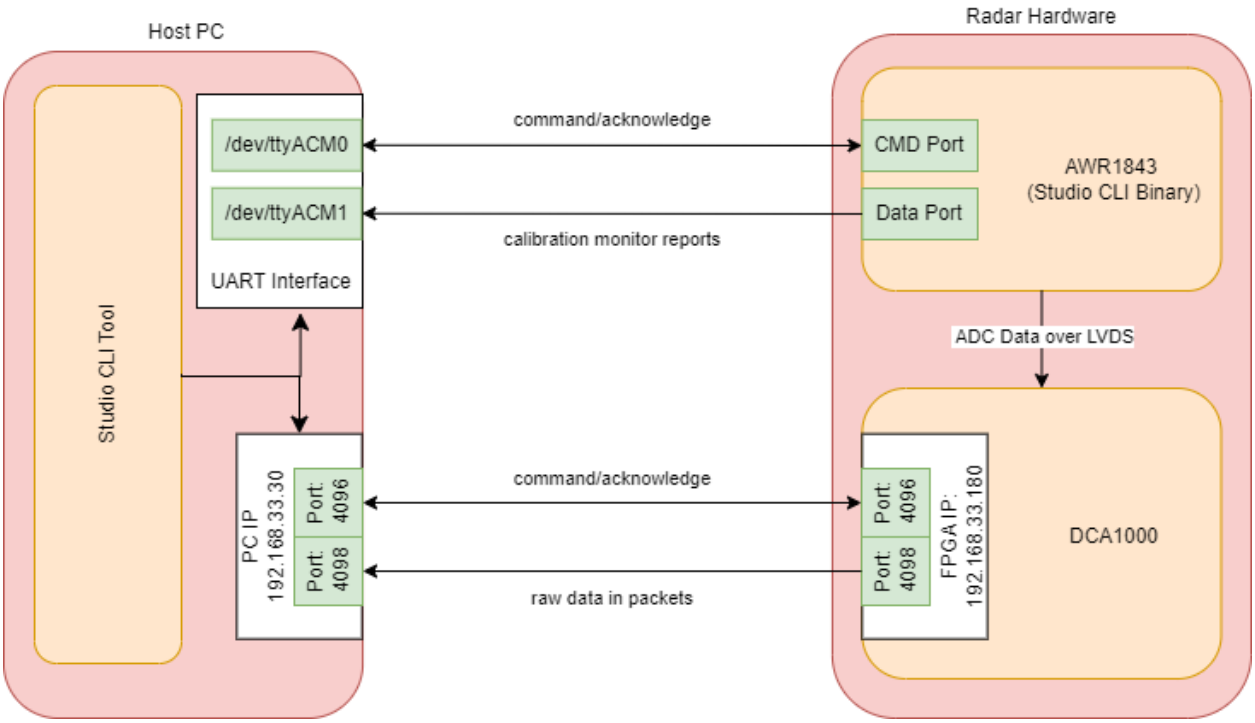

Figure 18: Block diagram of interactions between hardware and software components when using the Studio CLI tool for radar control and setup.

Before the ROS2 nodes for the radar module could be developed, the command structure for the DCA1000 and radar module (AWR1843) had to be determined. However, this was made easier due to the fact that the DCA1000 and AWR1843 are two completely separate subsystems when using the StudioCLI tool. They are controlled and configured separately. If one board fails, the other will continue to operate as normal. The DCA1000 observing and logging the raw ADC samples the radar generates is the only interaction between the two boards and is completely uni-directional. The data traffic was observed from two perspectives, one for each subsystem.

Ethernet Traffic with WireShark: WireShark was used to monitor Ethernet traffic between the host PC and the DCA1000. The host PC needed a static IP of 192.168.33.30 and the IP of the DCA1000's Field Programmable Gate Array (FPGA) was hard-coded as 192.168.33.180 as per the documentation in by TI [34]. This helped in singling out specific packets and their direction based on their source and destination which is listed in WireShark as seen in Figure 19. WireShark produced the User Datagram Protocol (UDP) packets shown in Figure 19. Packets highlighted in red represent command and acknowledgement pairs.

Packets highlighted in green represent the raw ADC data captured from the radar. The specific commands (listed on the left in Figure 19) were found in the technical documentation for the DCA1000. The packet contents were compared to op-codes listed in TI's DCA1000 user guide [34] to determine which commands were which. This was important as it showed the correct order that commands needed to be spent. Each command had a very specific and well defined structure which could be seen in the packet contents and confirmed by the documentation [34]. A list of op-codes and the command packet structures were taken from the DCA1000 user guide [34] and are shown in Table 1 and Table 2 respectively.

Table 1: List of op-codes used taken from the DCA1000 user guide [34].

DCA1000 Op-Codes	
RESET FPGA	0x01
CONFIG FPGA GEN	0x03
CONFIG PACKET DATA	0x0b
SYSTEM CONNECT	0x09
RECORD START	0x05
RECORD STOP	0x06
SYSTEM ERROR	0x0A

Table 2: Structure of commands, acknowledgements and data taken from a user guide [34] sent between the DCA1000 and Host PC where N is an Op-Code number listed in Table 1.

DCA1000 Command Structure				
Header (2 bytes)	Op-Code (2 bytes)	Data Size (2 bytes)	Data (Size bytes)	Footer (2 bytes)
0xA55A	0x0N00	variable	variable	0xEEAA

DCA1000 Command Acknowledgement Structure				
Header (2 bytes)	Op-Code (2 bytes)	Status (2 bytes)		Footer (2 bytes)
0xA55A	0x0N00	0x0000 Successful	0x0100 Failed	0xEEAA

DCA1000 Data Packet Structure		
Packet Number (4 bytes)	Total Byte Count (6 bytes)	Data (48-1462bytes)

With this information, the DCA1000 Ethernet interface was fully characterised. A dictionary was created with the op-code name as the key and the full command sequence in bytes as the corresponding value. This made it very easy to reuse, test and validate that individual commands were working. Using the command acknowledgement structure shown in Table 2, the status bytes were extracted to determine if the command was successful.

For the data packets, typically 1456 bytes were retrieved from each data packet except for the last packet which would contain the remaining bytes of the last frame. This can be seen by the length of the incoming data packets in Figure 19 which have a length of 1466 bytes. This corresponds to 1456 data bytes because of the 10 bytes used for the packet number and byte count.

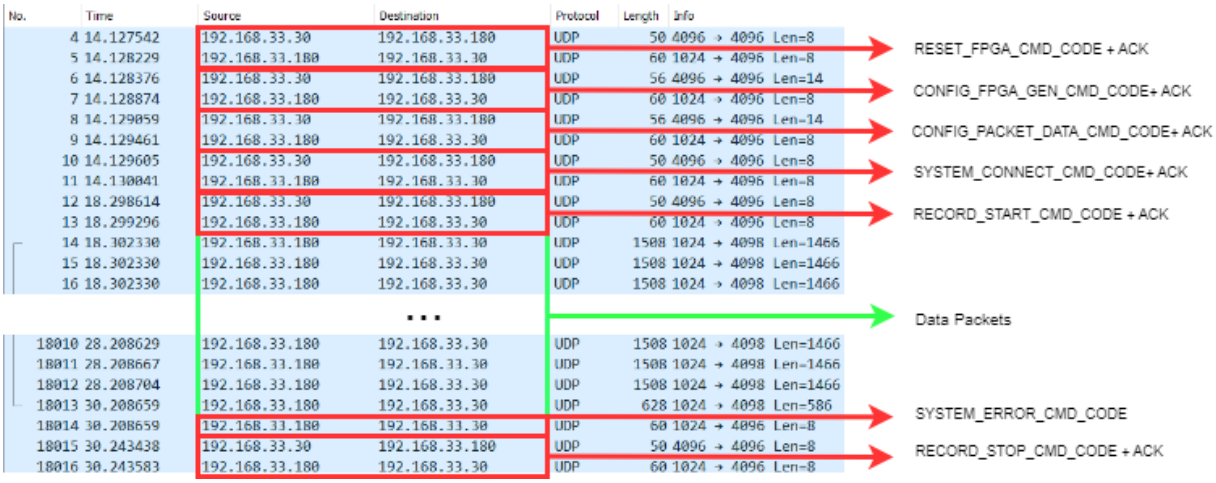

Figure 19: Ethernet packets sent between host PC and DCA1000 during a Studio CLI Session.

UART Traffic with Serial Monitor The same process applied to monitoring Ethernet traffic was applied here except with a Serial Monitor instead of WireShark. This showed that the text file used by Studio CLI (shown in Figure 20(b)) to configure the radar was sent line by line with a new line terminator. Each line corresponded to a type of command such as "profileCfg" (profile configuration) followed by the parameters of the command and a newline terminator. To stop and start radar chirp transmission, the commands "sensorStop" and "sensorStart", respectively, were sent over UART-USB to the command port of the radar, followed by a newline terminator to signal the end of the command sequence.

3.2.3 Radar Config File

The commands shown in Figure 20(b) were difficult to read or change because the parameters for each command were not labelled. This made the config file nearly impossible to work with. This is because TI mmWave users do not typically edit the config file directly. Instead, they use TI's online config generator tool or they use mmWaveStudio which has a GUI to handle config command changes. mmWaveStudio is clunky and inflexible, As discussed previously in Section 3.2.1, and access to internet is not always readily available when field testing equipment, so quick and easy, on-site editing of the config file became a very important feature of the system.

The exact meaning and requirements of all the command parameters were found in the mmWave SDK User Guide [35] under Section 3.4 of the document (*Configuration .cfg File*

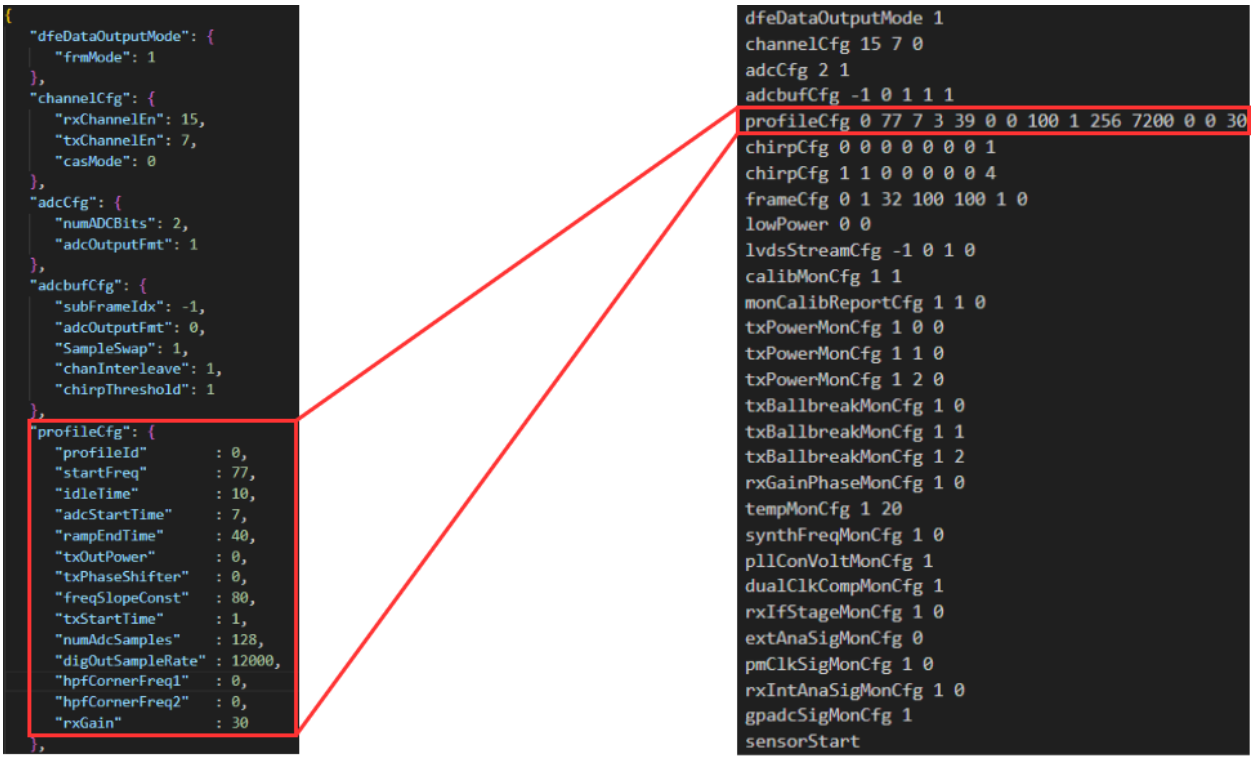

Figure 20: Comparison of improved JavaScript Object Notation (JSON) config file (a) vs original config file (b).

Format). The config file was then stored as a JSON dictionary as shown in Figure 20(a). Each command was a key where the corresponding dictionary value was a new dictionary for each parameter. The name of each parameter as listed in the guide [35] was used as a key in the inner dictionary and the value was the value of the parameter to be sent. This made it much easier to find which parameter needed to be edited and made it easier for new users to find the parameter in the documentation. In hindsight, Yet Another Markup Language (YAML) would have been a better option instead of JSON because comments could have been added to the configuration file.

The majority of the commands listed in the config are fairly arbitrary in that are seldom changed or only need to be changed for very specific use cases. These commands typically deal with the kind of data produced (real or complex) and can usually be left to their default values. However, there are three very important commands which determine the radar's performance metrics such as range resolution, velocity resolution, etc. and the radar's ability to perform MIMO processing. These factors will be discussed in Section 4. These commands include the:

1. profilCfg command
2. chirpCfg command
3. frameCfg command

3.2.4 Initial Software Components

Initially, the radar software was developed outside of ROS2 so individual software components could be easily tested. Separating these components into individual packages made it very easy to debug and to copy them into their ROS2 Nodes. These components included:

1. A Config class to handle the JSON config file.
2. The UART serial interface handling the radar config and control.
3. The Ethernet sockets handling the DCA1000 command and data ports.
4. The ring buffer reading in the data.

Config Handler A Python class was implemented to read in the config parameters and validate the config file (check that values were not out of bounds and the command names and number of parameters per command were correct). The config class also used the JSON file to construct the command to be sent over UART to the radar so it matched the command structure in the original config file (Figure 20(b)).

UART Serial Ports Before anything could be done in Linux with the radar serial interface, the user had to be added to the dialout group to enable use of the serial ports. This is done in Linux using the following command:

```
~ sudo adduser <user-name> dialout
```

The PySerial library was used to create a serial connection (UART) to the radar command port. The radar command port can be found on Linux via the terminal using:

```
~ ll /dev/serial/by-id
```

The above command returned:

```
total 0
drwxr-xr-x 2 root root 80 Aug 14 12:51 ./
drwxr-xr-x 4 root root 80 Aug 14 12:51 ../
lrwxrwxrwx 1 root root 13 Aug 14 12:51 usb-Texas_Instruments_XDS110__02.03.00.07
__Embed_with_CMSIS-DAP_00000000-if00 -> ../../ttyACM0
lrwxrwxrwx 1 root root 13 Aug 14 12:51 usb-Texas_Instruments_XDS110__02.03.00.07
__Embed_with_CMSIS-DAP_00000000-if03 -> ../../ttyACM1
```

From the above output, the relevant ports were /dev/ttyACM0 and /dev/ttyACM1. The number sometimes changed depending on the presence of other ACM devices but the command port was always the lower number of the two (so /dev/ttyACM0 in this case). The baud-rate of the command was found to be 115200baud. The other port, /dev/ttyACM1, is the TI data port. In the case of using the Studio CLI Binary, the data port is used for calibration monitoring reports. It was not necessary to use them and they seemed to interfere with the other port. Disabling the monitoring reports in the config file fixed the issue.

The config file class discussed earlier was used to formulate the commands sent to the radar through the UART ports. The commands were validated by reading in the response from the radar. The response for an example command (e.g. dfeDataOutputMode 1) was as follows:

```
dfeDataOutputMode 1
Done
mmWaveDemo:/>
```

The first line of the response was a repeat of the command sent to the radar, including the new line terminator. The second line of the response was a status. This status was either "Done" for success or one of several error codes listed in Table 3 when a command failed (again followed by an end terminator). The third line was a banner without any kind of end terminator. The concept was that if a user opened a serial session with the radar in an program like putty, the next command would fall on the same line as the banner to make the session look like a terminal. So this line was purely for aesthetic and so was ignored.

Ethernet Sockets Ethernet Sockets were required to handle the transmission of UDP command packets to the DCA1000 and the incoming acknowledgement and data packets from the DC1000. To send the commands, UDP sockets were created in Python using Python's socket library to match the setup shown in Figure 18. One socket was created to send commands to the FPGAs IP (192.168.33.180) over port 4096, one was created to listen for command acknowledgements at PC's IP (192.168.33.30) at port 4096 and another was created at the PC IP listening to port 4098 for data packets. Individual commands were tested to see if the appropriate responses were received in Python from the DCA1000 as shown in Table 2 and were validated using WireShark.

The data packet collection was validated by extracting the packet sequence number from the first 4 bytes of the packet and checking that the number of remaining bytes was as expected. This revealed that there was a significant packet drop problem. Dropped packets were identified by jumps in the packet number. Packet drop is a huge issue as it it affects the coherency and integrity of the data.

The packet drop problem was mostly fixed by re-implementing the data socket in C++ and by increasing the socket receive buffer using the set socket options function. The size of buffer

Table 3: List of common Studio CLI error codes taken from the Studio CLI Developer's Guide [**?**].

Studio CLI Error Codes	
Error as in Command Line	**Explanation**
error -50	Invalid CLI Command sent to mmwave sensor over UART. Command is not supported by the device's application.
error -51	Invalid usage of CLI command, wrong number of parameters in the command.
error -52	Invalid input parameter in the CLI command.
error -54	Sensor already started or stopped and got the same command again.
error -55	LVDS Software session and HSI header is not supported.
error -56	Execution Timeout in Application.
error -57	Error in DataPath Configuration.
error -58	Command is not supported while frame is running.

was tripled from the default value. Doing this improved the performance over a 100 fold over the Python script. The two implementations were compared by collecting zero filled, "dummy", packets over a 5 second window. The Python implementation could only retrieve 300 dummy packets during that window while the C++ implementation could handle close to 87000.

Packets could still be dropped for very high data through-put configurations or during experiments where the Computer Processing Unit (CPU) was heavily loaded with running other sensors. Dropped packets had to be zero filled to prevent packets from future frames being placed in memory that corresponded to earlier frames. Doing this protects future frames from being corrupted.

Ring Buffer Data read in from Ethernet Socket was placed into a ring buffer implemented with a tail (read) and head (write) pointer. The ring buffer uses the parameters in the config file to determine the size of the radar frame in bytes:

$$Bytes = 4N_{\text{chirp}}N_{\text{sample}}N_{\text{Rx}}N_{\text{Tx}}\ [bytes] \tag{2}$$

Where N_{chirp} is the number of chirps per frame. N_{sample} is the number of IQ samples per chirp and N_{Rx} and N_{Tx} are the number of receivers and transmitters respectively. The frame size is multiplied by 4 because the data is complex. So, every sample has a real and imaginary component each of which is two bytes. The ADC can produce either 12, 14 or 16 bit samples all of which required two bytes to be stored. The radar used produces 12 bit data. This data needs is held in 16 bit containers and therefore needed to be shifted accordingly.

The ring buffer was created to store up to 3 frames of data before overwriting old data. Once one frame's worth of data was collected the raw bytes were returned from the ring buffer. This was checked by comparing the current byte count of the ring buffer to the frame size. Upon returning the bytes from the current frame, the byte count was reset and the tail pointer was updated to match the start of the next frame.

Before being implemented in ROS2, the output data returned from the ring buffer was written to binary files (as a temporary substitute for publishing the data to a topic). These binary files containing the raw data collected from the radar were validated by performing basic FMCW processing (Range-Doppler Processing was used and is discussed further in Section 4).

3.2.5 ROS2 Drivers

The core components of the radar software discussed in the previous section (Section 3.2.4), were re-implemented in ROS2. Initially all the code was implemented under one ROS2 node

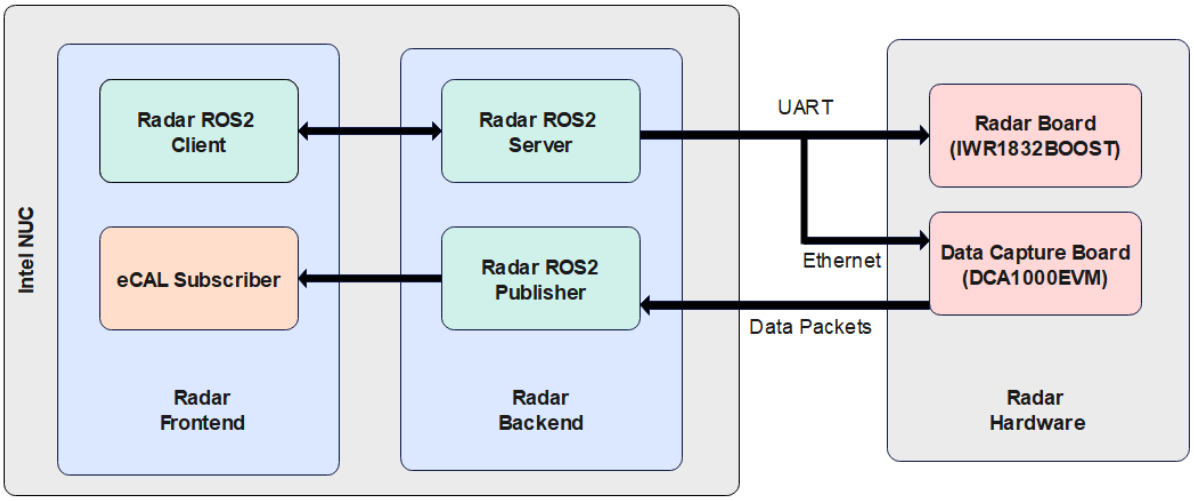

Figure 21: Radar ROS2 Node Network.

but the required functionality was too much to be neatly encapsulated in a single Node. Therefore, the functionality was split into two parts:

1. Control and Configuration
2. Data Collection and Storage

This is shown Figure 21. The top two nodes are responsible for the control and configuration of the radar. As can be seen in Figure 21, this functionality is implemented in a server-client style. ROS2 has built in libraries for creating server-client relationships between nodes. The client (the front-end) tells the server (the back-end) whether to send setup commands stored in the configuration JSON, to start recording or stop recording. The DCA1000 Command Sockets, radar UART and config components from Section 3.2.4 were used to implement the functionality of the ROS2 server node. The server's client is run using one of three commands:

```
ros2 run radar_dca1000_py_pkg radar_clnt SETUP
ros2 run radar_dca1000_py_pkg radar_clnt START
ros2 run radar_dca1000_py_pkg radar_clnt STOP
```

The ability to completely control the radar with bash commands makes this interface highly flexible. The bash commands can be linked to buttons in a Graphical User Interface (GUI) or can be run from an Secure Shell (SSH) session. The only requirement is that the radar ROS2 server is running which can be similarly started with a bash command.

The bottom two nodes in Figure 21 are responsible for data collection and storage. The nodes are implemented in a publisher-subscriber style which ROS2 has libraries for. The publisher node uses the ring buffer, config and DCA1000 data socket software components from Section 3.2.4. The publisher node continuously checks and compares the ring buffer's byte count to the calculated frame size available from the config file. Once the byte count equals or exceeds the frame size, a radar frame is published to a ROS2 topic. The published data follows a specified format defined in a ROS2 message file. For the radar, the message contained a header (which had the frame ID and timestamp split into seconds and nanoseconds) and the actual radar data stored as a byte array. The second of the two data nodes is a ROS2 subscriber which stores the data and represents the front-end of the data collection functionality. This subscriber was initially written in python, leveraging their easy to use Hierarchical Data Format v5 (HDF5) library (h5py). However, this was too slow. It was replaced by third party software called enhanced Communication Abstraction Layer (eCAL) which was much faster and is discussed in Section 3.2.7

Publishing to a topic instead of just writing the data to storage makes the node much more flexible. Other ROS2 nodes can access the data to perform different tasks concurrently. This allows for recording and processing to be done at the same time. Additionally, different processing steps such as Range Fast Fourier Transform (FFT), Range-Doppler, Constant False Alarm Rate (CFAR) or Point Cloud generation could all be implemented on separate nodes each subscribing to a previous stage in the pipeline and then publishing to their own topic for the next stage. These other nodes could even be made to be very performant by being written in C++ with Compute Unified Device Architecture (CUDA) to leverage Graphical Processing Unit (GPU) matrix multiplication to allow for real-time data processing.

An additional advantage of this setup, is the two back-end nodes run completely independently from each other and are mostly separated by the hardware they deal with. This means that if one hardware module or node fails, it can be easily restarted without having to restart the entire node network.

The entire setup was also placed within a ROS2 launch command. This meant that the entire back-end (and even radar processing nodes) could be started with a single command.

3.2.6 Radar Test Rig Software

The test rig software can be split into parts. The ROS2 node responsible for sending movement commands to the STM32 and recording the movement data and the firmware on the STM32 itself.

STM32 Firmware The STM32 firmware defined a set of pre-defined command structures and implementations for handling them. Two basic commands were defined: MOVE and HOME. The HOME command moved the platform towards an end-stop switch close to the radar until the end-stop switch was triggered and then moved the platform a set number

of steps in the opposite direction so the rig always started in the same place. The MOVE command had several parameters including: the number of steps, the direction and the stepping period. The number of steps controlled the overall distance travelled. The direction parameter controlled which direction the rig moved, 1 for forward and -1 for backwards. The stepping period controlled the speed of the motion with smaller periods creating less time between steps creating faster movements. The commands were received from the ROS2 node over USB serial.

Test Rig ROS2 Node The ROS2 node (written in python) used a preset trajectory defined in a numpy array to formulate commands sent to the micro-controller. Numpy was used frequently in this dissertation due its ease of use and extensive functionality [36]. The path to the numpy array was stored in the ROS2 stepper driver launch file. This allowed multiple trajectories to be tested, if desired, by just changing the launch config file. The ROS2 node always started by sending the HOME command to STM32 to start the experiment. The ROS2 node then used the numpy diff function to determine the distance travelled between each sample and the time taken to perform the movement. The number of steps required could be calculated as shown in Equation 3 where n_{req} is the number of steps calculated, n_{rev} is the number of steps per revolution, d_{req} is the distance required to be travelled by the platform and d_{rev} is the distance per revolution.

$$n_{\text{req}} = \frac{n_{\text{rev}} d_{\text{req}}}{d_{\text{rev}}} \tag{3}$$

The stepping period was determined by Equation 4 where T_{step} is the stepping period (or time taken per step), T_{total} is the total time to accomplish all the required steps and is equal to the sample period of the rig trajectory.

$$T_{\text{step}} = \frac{T_{\text{total}}}{n_{\text{req}}} \tag{4}$$

The direction was determined by using the signum function on the sample differences. These parameters were sent to the micro-controller every sample period using the MOVE command.

3.2.7 ROS2 Middle-ware

A brief mention needs to be made concerning ROS2 RMW. Middle-ware is essentially the underlying backbone of ROS2 that manages the topics, publishers, servers, subscribers and all the other components that any ROS2 node is built on. ROS2 itself is a collection of client libraries that connects to a middle-ware and the middle-ware is the code that connects ROS2 nodes to an underlying Data Distribution Service (DDS) (Data Distribution Service) [37].

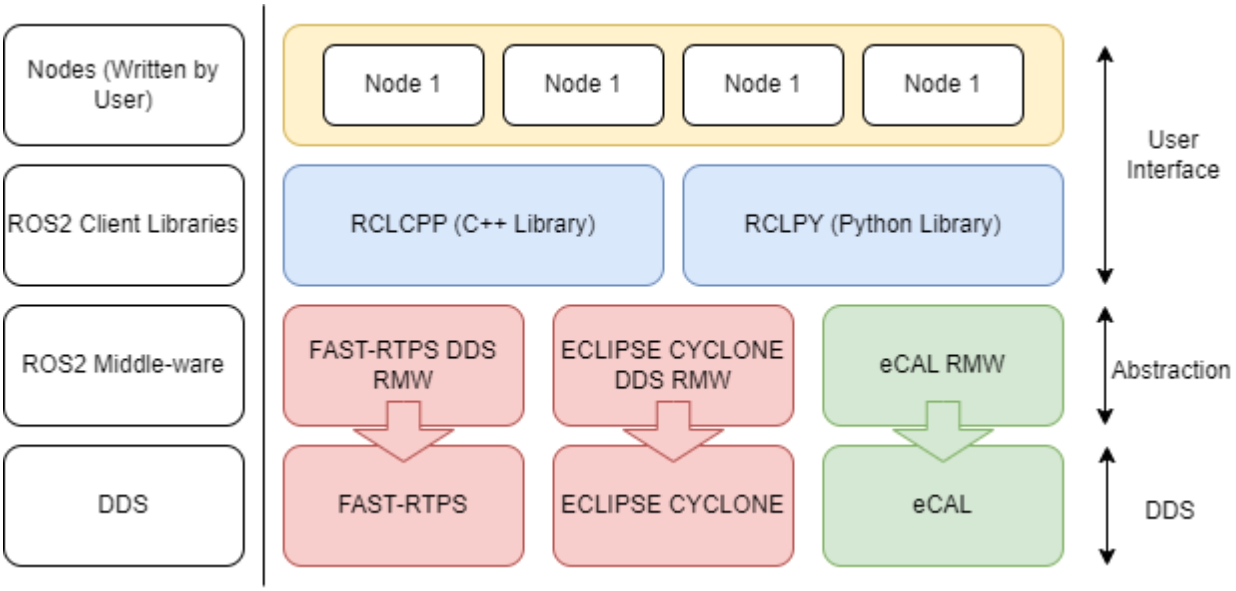

Figure 22: ROS2 Architecture from Node to Underlying DDS.

Different middle-ware supports different DDS implementations which in turn have different advantages or are designed for different things [37]. This layout is shown in Figure 22

The middle-ware used plays a crucial role in the performance of a ROS2 node network as it facilitates the communication between nodes. The ROS2 default RMW is DDS/RTPS. RTPS is a wire protocol that DDS frame works leverage to implement needed communications.

However, the default middle-ware was too slow for the high data rate required when using multiple high throughput sensors. Because of this, the default RMW was replaced by a RMW called eCAL-RMW (enhanced Communication Abstraction Layer) because the underlying DDS (eCAL) was found to be significantly faster [38]. eCAL tries to simplify communications as much as possible to provide as much data throughput as possible while following the same publish-subscribe pattern as other DDS implementations [39].

This RMW came with another advantage which was that it could generate its own subscribers [38]. This negated the need to write any highly performant subscribers which helped with development time and provided a convenient way to record multiple topics from one GUI or CLI interface.

3.2.8 Data Flow Overview

The data produced by the radar setup created the data flow shown in Figure 23. Raw IQ data stored as interleaved bytes are sent via Ethernet into the ring buffer. The ring buffer collects the bytes and pushes the raw bytes to the ROS2 publisher once a single radar frame has been collected in the buffer. The publisher posts the radar frame bytes to the topic which can be subscribed to by other nodes to do different FFTs. The subscribers mentioned in Section

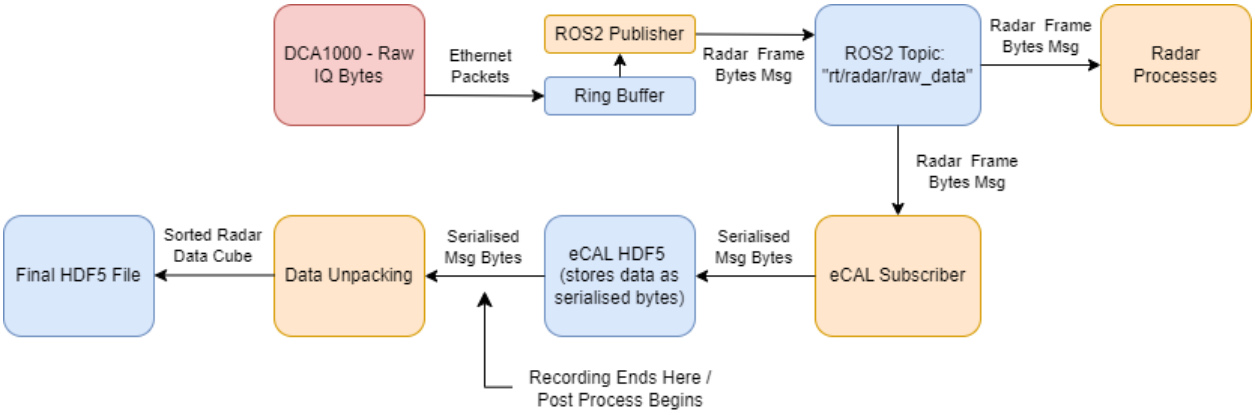

Figure 23: Diagram of data flow through the recording pipeline. Hardware is coloured in red. Data structures are coloured in blue. Processes are coloured in orange.

3.2.7 (the eCAL subscribers) pull data from this topic and store the raw serialised data from the topic into a HDF5 file. Because the data was serialised, entire messages were stored as bytes in these files according to the structure of the message as shown in Figure 24. Every message had its own group named with the number of the message. The group number was determined by the order at which the subscribers received the message. The group contained a header with metadata such as a timestamp.

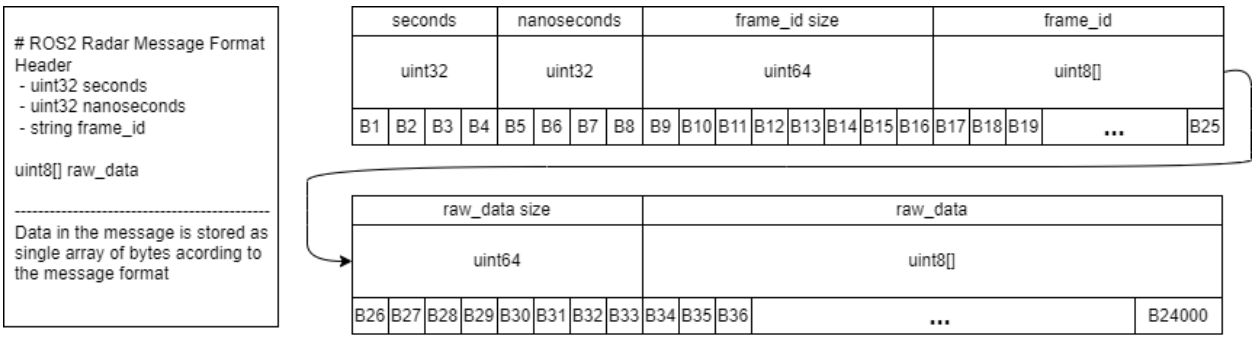

Figure 24: Example of how bytes within an eCAL message entry relate to ROS2 message structure.

Once all the frames were collected and stored in the HDF5 file produced by the eCAL subscriber, the data had to be unpacked in post into a final HDF5 file so the data could be used without any prior knowledge of the message structures required. Unpacking was done by iterating through message entries and extracting the bytes corresponding to different fields. Where possible this process was vectorised to leverage the speed gained by using matrices. This file was created to have meaningful group and data-set names stored alongside any other metadata that needs to be recorded. The structure of the new HDF5 file is discussed in depth in the Section. 3.2.9

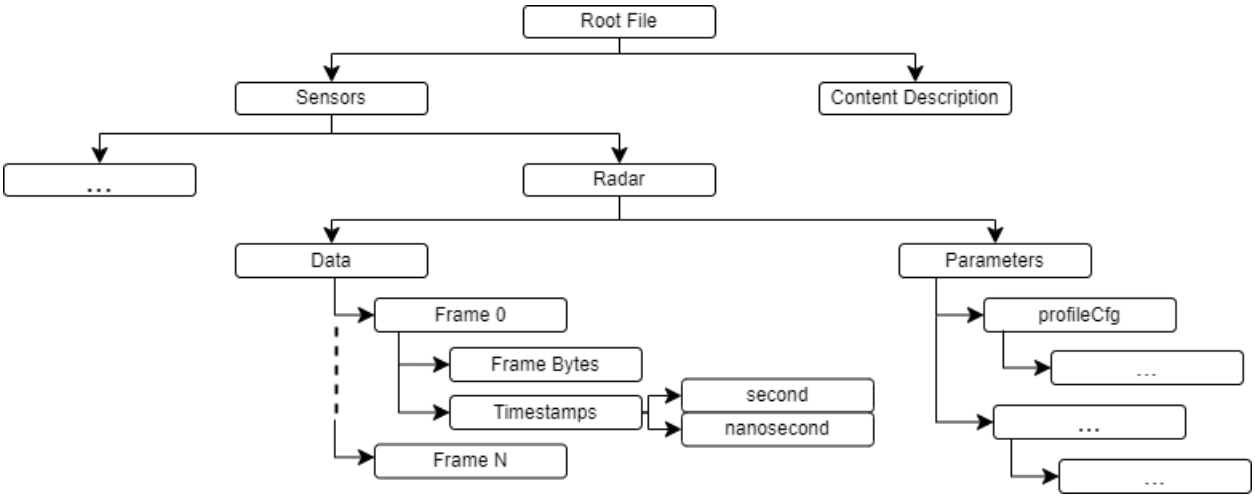

Figure 25: Diagram of final HDF5 structure for the radar.

3.2.9 Final HDF5 File Structure

HDF5 is a standard file format for scientific computing [40]. It is typically used for large, complex and heterogeneous data-sets as the structure's within the file are user defined and the file itself has no file size limits [41]. The two main structures within HDF5 files are:

1. **Groups** which organise data-sets and sub groups within a file. They are equivalent to folders within a directory [41].
2. **Data-sets** which can effectively be any type of data of any size or dimension [41].

Each frame was stored in its own group with a HDF5 data-set for the frame data and sub-group to store the two timestamp data-sets for seconds and nanoseconds. Any additional information that was needed to work with the data was added in separate groups in the same file for ease of use, under the parameters group. Additional information included image heights and widths for cameras or radar parameters used to decode the structure of raw radar data such as the number of channels, chirps and samples used as well as any other parameters needed to assign meaning to values calculated from the radar data. The radar parameters were stored as defined by the JSON config file, where each command formed a group and each parameter for that command were stored as datasets within the group.

3.3 Summary of System Design

This chapter represents the mid-point of the body work presented in this dissertation. In this chapter, the hardware used (including the radar, RealSense, test rig and hear rate monitor) was selected and integrated together via the development of different software components. The hardware and software created thus far allows for the creation of the experiment data as

well as means to validate the data and the processing pipeline to be developed. Additionally, the required flow and storage of data has been established and developed. With these components in place, the processing pipeline was ready for development and then testing and validation.

4 Processing Pipeline Design

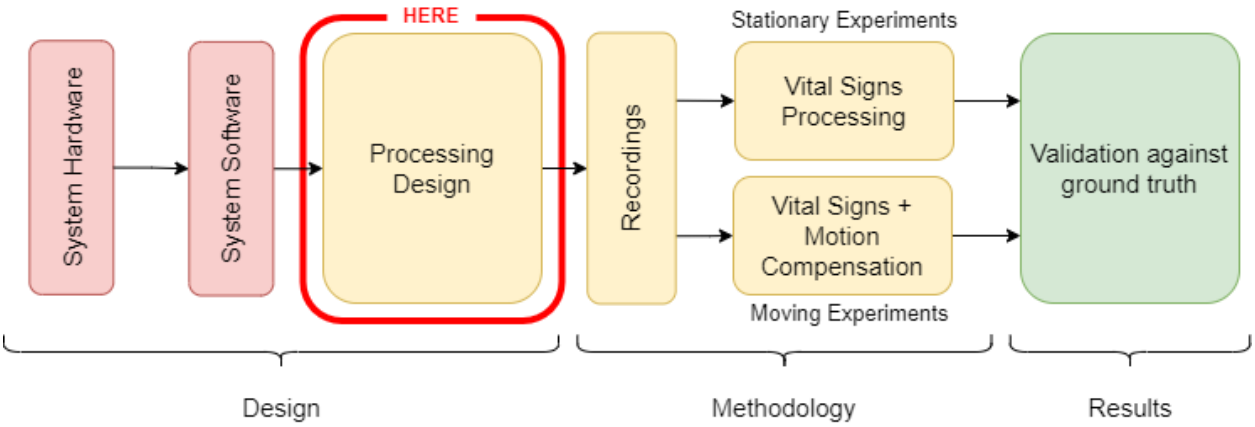

Figure 26: Figure of dissertation layout: Processing Pipeline Design

Following on from the detailing and implementation of the hardware used to conduct the research in this dissertation, the processing pipeline had to be designed to include basic FMCW processing, target tracking and vital sign (phase) processing.

4.1 Fundamentals of FMCW Radar Processing

The following sub-sections detail the theory required to conduct FMCW radar processing. This includes the basic principle of how chirps are transmitted, received and mixed to get the Intermediate Frequency (IF) signal and how that signal is sampled to generate a radar data cube. It then goes on to explain the basic DSP theory behind how multi-dimensional FFTs can be applied to the radar data cube to extract information into various maps and how MIMO is used to improve resolution.

4.1.1 FMCW Theory

FMCW Radar works by continuously transmitting "chirps". Mathematically, these chirps are sinusoidal signals of a single instantaneous frequency that increases over time. This increasing frequency is described in Equation 5 where F_0 is the starting frequency and S is the frequency slope.

$$\omega_c(t) = 2\pi(F_0 + St) \tag{5}$$

Integrating Equation 5 over time to calculate phase and adding the result to an initial phase, ϕ_0, yields Equation 6 taken from the dissertation by Trange [2] where $T_x(t)$ is the transmitted signal.

$$T_x(t) = A\sin(2\pi F_0 t + \pi S t^2 + \phi_0) \tag{6}$$

This results in a wave form that varies over time as shown in Figure 27 [42]. T_{ADC} is the time it takes for the radar to start sampling the received chirp. ΔF_{Total} is the total frequency bandwidth covered by the chirp. This total bandwidth is not necessarily used but cannot exceed the capacity of the radar (in the case of the TI radars, this bandwidth is typically 4 GHz). $\Delta F_{\text{Sampled}}$, is the portion of the total bandwidth utilised by the radar and contained within the output samples of the radar. T_{Ramp} is the amount of the time the radar is actively ramping the frequency while T_{Idle} is the amount of time given to the radar to ramp down the signal back to the starting frequency. The radar never samples during T_{Idle}. The total chirp time is the sum of T_{Ramp} and T_{Idle} because the chirp time is defined as the duration between the start of one chirp and the start of the next chirp.

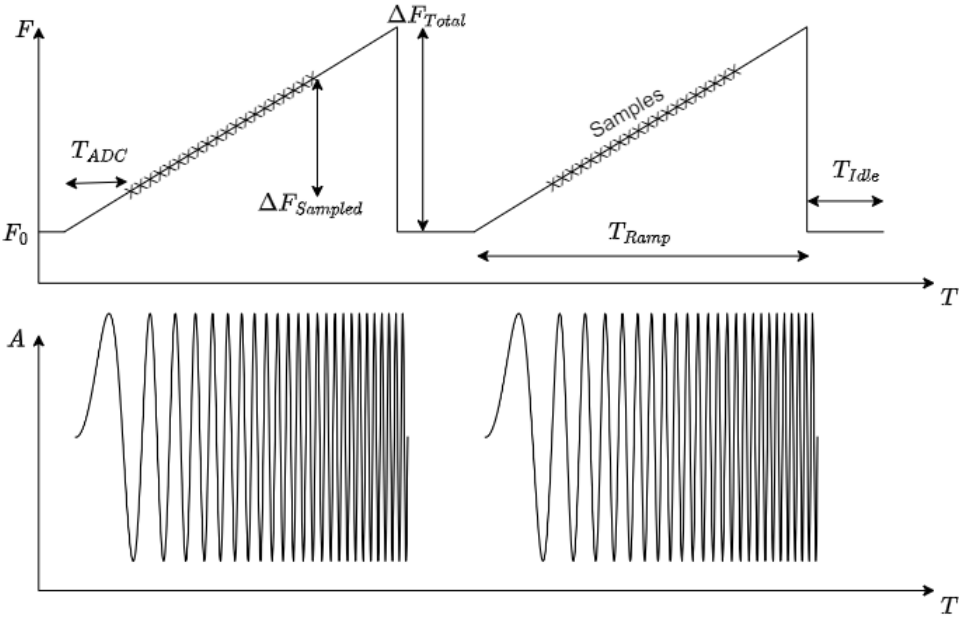

Figure 27: Diagram of chirp waveform and frequency over time

The radar receives an echo of the transmitted chirp which has Doppler shift F_D and a delay T_{Delay} as shown in Figure 28. These shifts in time and frequency cause a difference in frequency between the two waveforms. Through heterodyning, the transmit and receive signals are mixed. This mixing creates a new waveform, often labelled the IF signal, with a frequency F_{Beat} and phase shift Φ_{Beat} as shown in Equation 7 taken from TI's FMCW documentation [43].

$$x_{\text{Beat}}(t) = A_{\text{Tx}} A_{\text{Rx}} \sin(2\pi F_{\text{Beat}}(t)t + \Phi_{\text{Beat}}(t)) \tag{7}$$

$$F_{\text{Beat}}(t) = F_{\text{Tx}}(t) - F_{\text{Rx}}(t) \tag{8}$$

$$\Phi_{\text{Beat}}(t) = \Phi_{\text{Tx}} - \Phi_{\text{Rx}} \tag{9}$$

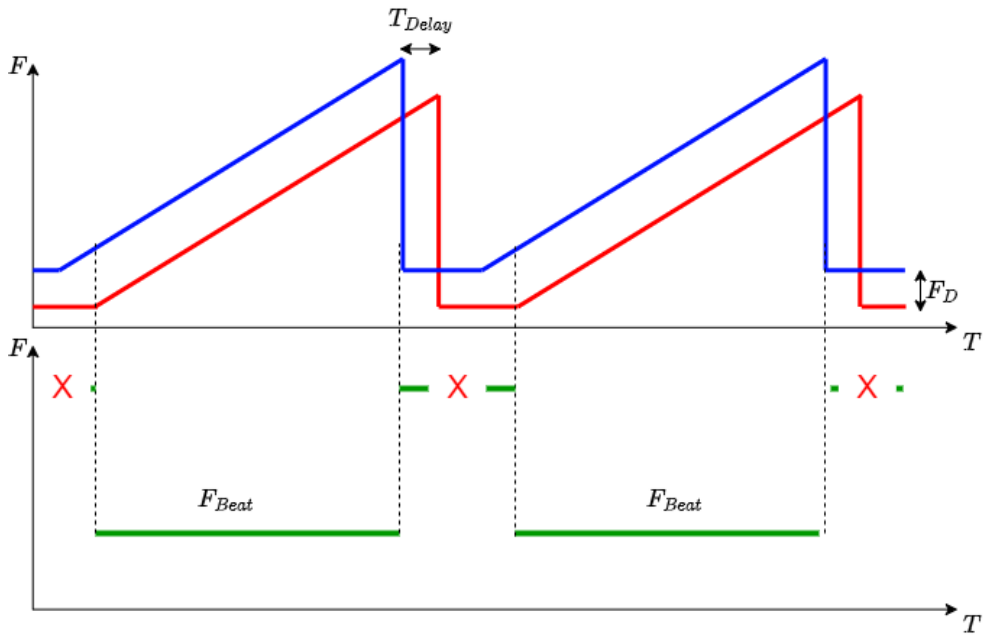

Figure 28: Diagram of received vs transmitted chirp.

This intermediate frequency is sampled by the radar to produce complex IQ (In-Phase and Quadrature) samples. These samples store the information about target range, velocity and angle and are sent out the radar as binary data to be processed. How the samples are placed and ordered is where the magic of FMCW radar processing takes place which is discussed further in the following sections.

4.1.2 FMCW Data Cube

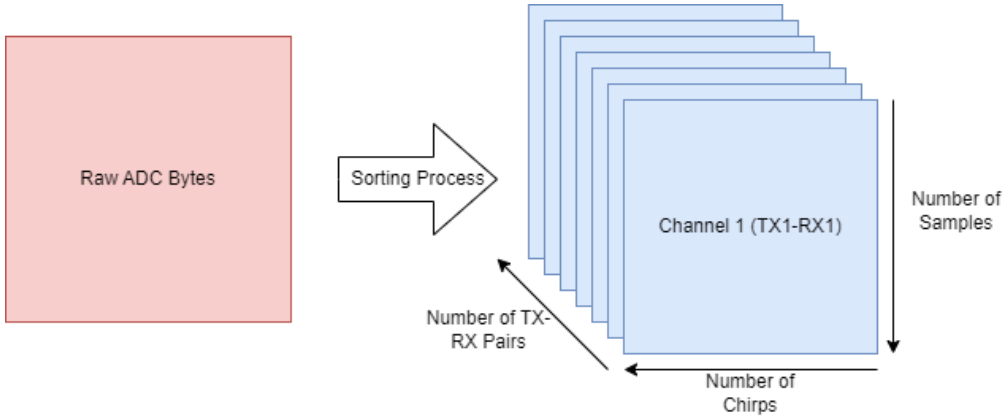

Figure 29: Diagram of conversion from a continuous array of bytes to a radar data cube.

For TI radars, the raw IQ data captured for each frame is stored as a single continuous array of bytes in an interleaved format. This format is determined by the number of LVDS lanes between the radar and data capture board [44].

This data needs to be sorted into complex IQ samples within a data cube. The data cube is an intelligent way of ordering and placing the samples to vectorize processing. The axes of the cube represent the three dimensions that the radar can measure: range, velocity/Doppler and Angle of Arrival (AoA). The length of the range axis is determined by the number of samples used within a single heterodyned chirp to digitise the IF signal. The length of the velocity (or Doppler) axis is determined by the number of the chirps per frame. The velocity axis is often called the slow time axis as it sampled by the chirps in the order of hundreds of microseconds. The range axis is often called the fast time axis by TI and in literature because it is sampled in the order of nanoseconds. In short, the range axis is always sampled faster than the velocity axis (hence the moniker of fast and slow time axis respectively). The length of the angle axis is determined by the configuration (Single Input Multiple Output (SIMO) vs MIMO) and number of transmit and receive antennas.

Taking successive FFTs along different axes of the cube in Figure 29 results in different visualisations of the data such as range-velocity or range-azimuth/elevation maps.

4.1.3 Radar FFTs

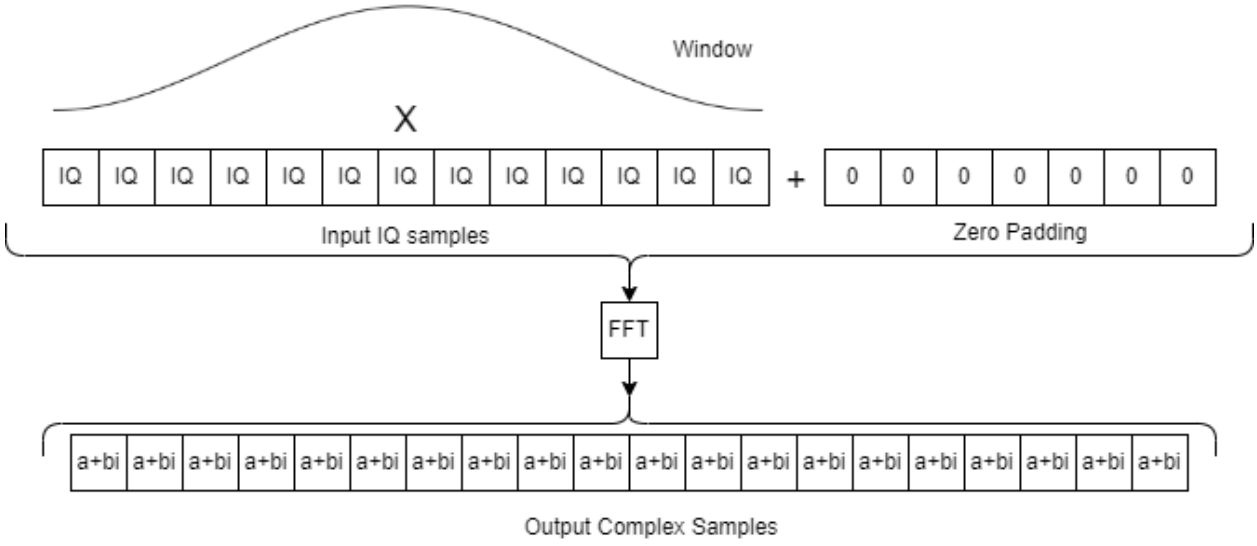

Figure 30: Diagram of FFT input signal conditioning with windowing and zero padding.

The term FFT needs to be discussed in the context of FMCW radar to highlight some additional steps that are performed with every FFT. As mentioned in Section 4.1.2, FFTs are applied to the radar data cube to produce meaningful data. A FFT produces a new set of complex data from its input signal that represents the frequencies contained in the signal. The output of the FFT has the same size as the input. However, this input signal

can be conditioned to try improve the output of the FFT. This is typically done using the two techniques shown in Figure 30:

1. **Windowing** is the processing of doing an element-wise multiplication of the input data with a window. Because the data being taken into the FFT is not infinite in length, it has to be thought of as being multiplied by a rectangular pulse that truncates the data to its current length. The FFT of a rectangular pulse is a *sinc* function as described in Equation 10 where τ is the length of the window. The *sinc* function has a main lobe that appears at every peak in the output frequency spectrum with large side-lobes that add interference to the rest of the signal. These side-lobes are reduced by using a different window because the FFT of the window is designed to have lower side-lobes, at the cost of having a wider main lobe (which can degrade frequency resolution and, consequently, range resolution) [45].

$$\text{rect}_\tau(\frac{t}{\tau}) \xrightarrow{\mathcal{F}} \tau\text{sinc}(\frac{\omega\tau}{2\pi}) \tag{10}$$

   The presence of τ in Equation 10 is important as it highlights that having a longer window improves the frequency resolution (as the main lobe width decreases) and the main lobe height increases.

2. **Zero Padding** is the process of adding zeros the input signal to increase the number of output samples in the FFT. This does not produce any new information but does create a smoother output spectrum at the cost of increased computational complexity. It can make the FFT more efficient by rounding the number of FFT points to a power of two [46].

Figure 31 highlights the effect of and necessity of windowing and padding. The input signal was comprised of two sinusoids, a DC offset and some white noise. The second sinusoid had a much lower amplitude than the first. Consequently, in the comparison of different windows in Figure 31, the peak corresponding to the second sinusoid was hidden beneath the side-lobes of the larger signal. The peak was only revealed when a window was applied. However, in the top plot of Figure 31, you can also see the widening effect of windows on the main lobe. In the bottom plot of Figure 31, the effect of zero padding can be seen. The frequency spectrum gets much smoother and more detail about the side lobes are captured. One can also see that past a certain padding ratio, there are limited gains from increasing the ratio further, as seen by the difference between 4x and 8x padding.

Every FFT discussed in this dissertation uses the techniques shown in Figure 30. A Hann (also called Hanning) window was used. It was picked out of the several windows shown because it had the better side-lobe suppression than the Bartlett and Hamming windows while maintaining a better main lobe width than the Blackman window. Signals were zero padded to a ratio of 4x the length of the input signal and rounded to the nearest power of two to improve computation speeds.

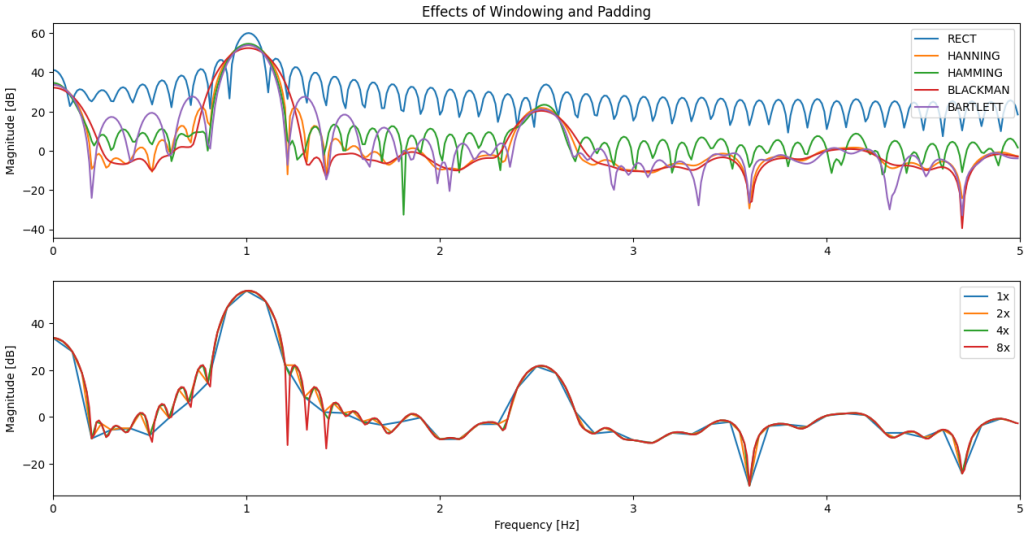

Figure 31: **Top:** comparison of different window types. **Bottom:** Comparison of different pad length using a Hanning window.

4.1.4 MIMO Configuration

The MIMO radar configuration uses Time Division Multiplexing (TDM) to temporally separate chirps from different transmit antennas. This process is explained in TI's MIMO documentation [47]. The FMCW chirps are emitted sequentially from each transmitter so that only one transmitter is firing at a time (this is different to the SIMO configuration where all transmit antennas fire at the same time). This allows receive antennas to distinguish the transmit sources of different chirps. This ability to distinguish the source of the chirp synthesises a virtual antenna array whereby the receive antenna pattern is duplicated spatially based on the separation of the transmit antennas as seen in Figure 32.

Therefore, using the MIMO configuration multiplies the number of receive antennas by the number of transmit antennas which greatly improves the azimuth AoA resolution. The azimuth resolution is the Field of View (FoV) of the radar (which is 120 deg) divided by the number of receive elements in the direction of interest. Therefore, the angular resolution in azimuth improves from 30 deg to 15 deg (120 divided by 8 elements instead of only 4 elements). The MIMO configuration even enables elevation AoA processing due to the vertical separation of the 2^{nd} transmit antenna [47]. The reason for this improvement is discussed in Section 4.2.4. This is based off of a common approximation which is described in Equation11 where 2 is the FoV in radians and N_{Rx} is the number of virtual receiving elements.

$$\theta_{\text{res}} = \frac{2}{N_{\text{Rx}}} \tag{11}$$

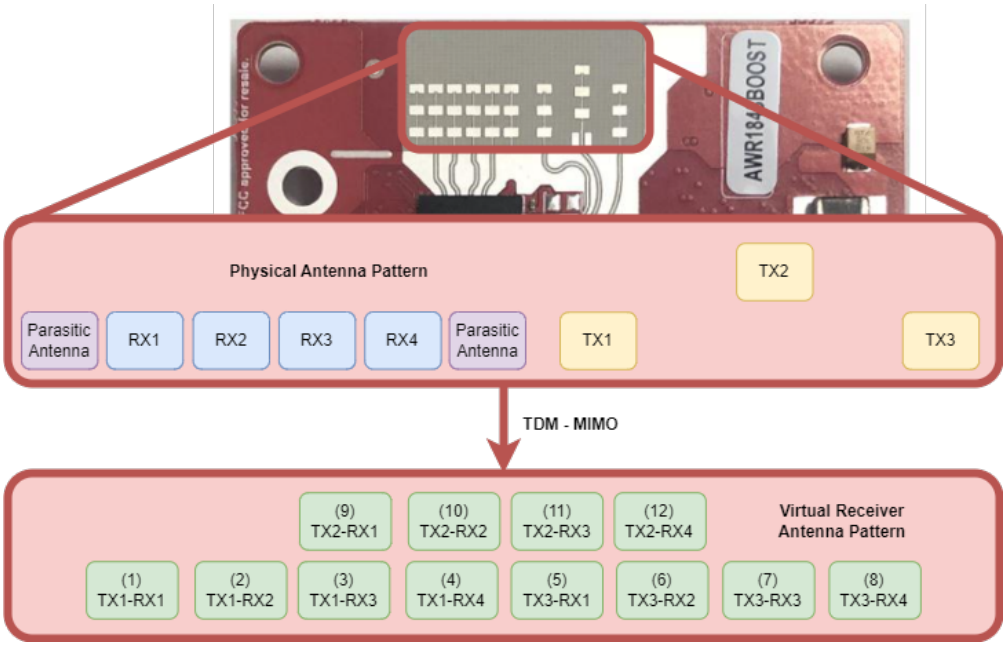

Figure 32: AWR1843BOOST Hardware vs Virtual Antenna Pattern.

4.2 FMCW Radar Target Tracking

4.2.1 Radar Tracking Pipeline Overview

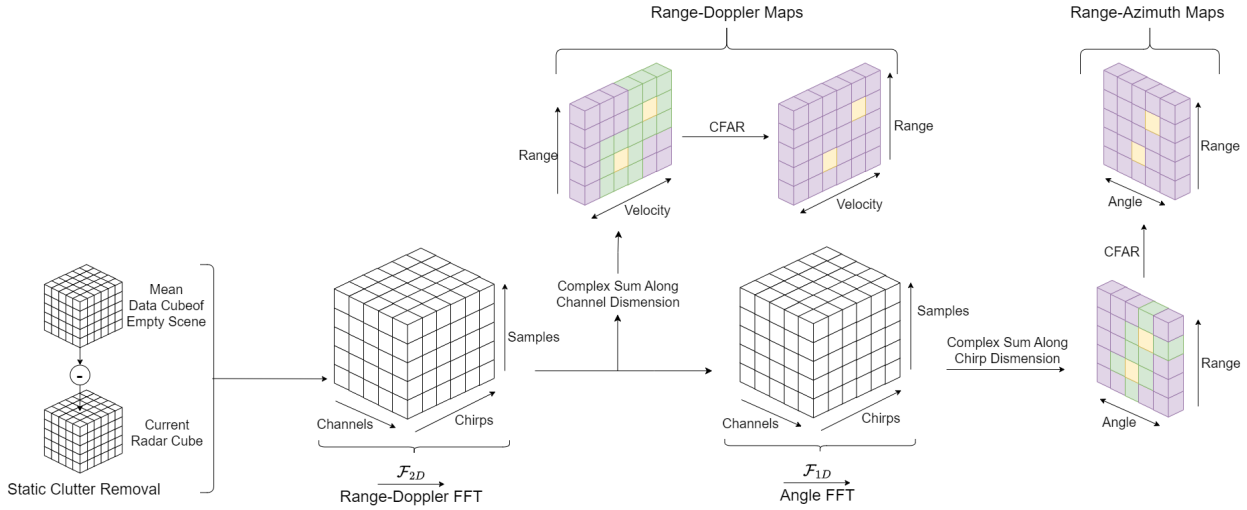

Figure 33: Overview of Radar Tracking Pipeline.

In Section 4.1.2 it is stated that taking successive FFTs of the data stored in the radar cube produces different visualisations of the data [32]. This process is shown in Figure 33. The data undergoes several transforms and operations to produce the CFAR detection maps. The flow of these operations are shown in Figure 33:

1. A cube representing the static clutter in the radar's FoV is generated by taking the

complex average of several cubes over time of an empty scene. This clutter is subtracted from the current data cube to remove any erroneous stationary targets.

2. A 2D FFT is taken along the range (samples) and velocity (chirps) axes of the data cube. This is equivalent to first taking an FFT along the range axis and then another along the chirp axis. This produces a new data cube with a complex Range-Doppler map for each channel.

3. The Range-Doppler Cube under goes a complex matrix summation to produce a two dimensional map such that:

$$Map_{\mathrm{s,c}} = \sum_{\mathrm{n}=1}^{nChannels} Cube_{\mathrm{s,c,n}} \tag{12}$$

   where s corresponds to a sample (row) in the data cube, c corresponds to a chirp (column) in the data cube.

4. The 2D Range-Doppler map under goes a detection algorithm to pull target ranges and velocities out of the map. This is the output of the pipeline.

5. An angle FFT can be applied to the Range-Doppler Cube to produce a Range-Angle data cube.

6. Performing a similar operation to what is done in Equation 12, except along the chirp/velocity dimension, produces a Range-Angle Map.

7. The 2D Range-Angle map also under goes a detection algorithm to pull target ranges and angles out of the map. This is another potential output of the pipeline that can be used to separate signals based on the target's azimuth position.

The steps and processes outlined above are described in detail in the following sections.

4.2.2 Range-Velocity Processing

This section highlights the basic theory of how the 2D Range-Doppler FFT, mentioned in Section 4.2.1, is done.

$$\begin{aligned} r_{\mathrm{Target}} &= \frac{c\, T_{\mathrm{D}}}{2} \\ &= \frac{c\, f_{\mathrm{IF}}}{S2} \end{aligned} \tag{13}$$

As mentioned in Section 4.1.1, the beat frequency is related to the difference in frequency between the transmit and receive frequency at a given point in time. This difference is caused

by the time delay between the two signals. The time delay is directly related to the range of the target by Equation 13 [43].

Therefore, the range to targets is encoded into the frequencies contained within the IF signal with different frequencies corresponding to targets at different ranges [43]. Performing what is known as a range FFT, on the samples captured within the IF signal of a single chirp, creates a frequency spectrum where peaks in frequency relate to ranges of reflective targets. The ability to distinguish between these peaks in frequency is what determines the range resolution in Equation 14. In Section 4.1.3, it is noted that the main-lobe width narrows for longer measurements. Narrow main lobes correspond to better range resolution because it causes peaks to neighbouring peaks to overlap less so they can be resolved. As seen in Equation 14, the range resolution Δr, improves with an increased sampled bandwidth ΔF_{S}. ΔF_{S} is described in Equation 15 and is proportional to the Frequency Slope S, the number of samples N_s and inversely proportional to the IF signal sampling rate IF_{Max}. Increasing S results in a greater difference in frequencies in the IF signal which in turn creates more space between peaks for them to be resolved. Increasing the ratio between N_s and IF_{Max} creates a longer time domain signal for the input to the FFT which narrows the main-lobe width of the FFT, improving FFT frequency resolution and thus, also improving the range resolution.

$$\Delta r = \frac{c}{2\ \Delta F_{\mathrm{S}}} \tag{14}$$

$$\Delta F_{\mathrm{S}} = S\ \frac{N_{\mathrm{s}}}{IF_{\mathrm{Max}}} \tag{15}$$

Intuitively, the maximum range measured is just the range resolution, Δr multiplied by the number of range bins. This is comparative to determining the length of a ruler by multiplying the number of notches by the width of each notch. A range bin is a region of space where two different targets will not be resolved in range and correspond to a single discreet range sample. The number of range bins is therefore equal to the number of samples per chirp. So, substituting Equation 14 and 15 for Δr in Equation 16 yields that the maximum range, r_{max}, is defined by:

$$\begin{aligned} r_{\mathrm{max}} &= N_s\ \Delta r \\ &= \frac{c}{2\ S\ \frac{\cancel{N_s}}{IF_{\mathrm{Max}}}} \cancel{N_s} \\ &= \frac{c\ IF_{\mathrm{Max}}}{2\ S} \end{aligned} \tag{16}$$

The above deals with data contained within a single chirp. However, multiple chirps are transmitted within a radar frame. As a target moves within the radar frame, a small change

in the path length between the radar and target is introduced. This small path length change introduces a phase shift into successive chirps even though the target likely remained in the same range bin. Therefore, velocity information is encoded into the phase of successive chirps [43]. This is shown in Figure 34 where a single IQ sample is shown from two successive chirps. The two samples have the same frequency ω corresponding to the same range but they have two different phase shifts Φ_0 and Φ_1 which reveals that the target has moved in the time taken to receive the two chirps.

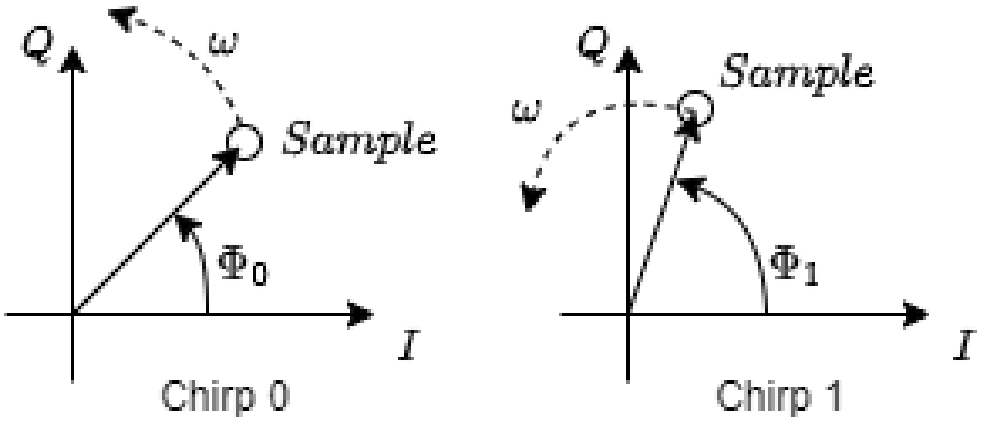

Figure 34: Diagram of the phase shift of an IQ sample between two successive chirps with the same frequency ω.

Frequency in rad/s is, effectively, the rate of change of phase. So integrating the chirp frequency over time will give the received phase of each chirp as the wave travels through the air. To make the integral simpler, it can be assumed that the frequency is a constant equal to the centre frequency of the chirp. Therefore, the phase of the received chirp is equal to the centre frequency multiplied by the time of flights, T_{R1} and T_{R2}, for each chirp as shown in Equation 17 and 18 respectively. Taking the difference between the phases of the two chirps yields the phase change caused by the displacement of the target during the time between the chirps T_c as shown in Equation 19.

$$\begin{aligned}\Phi_{\mathrm{R1}} &= \Phi_0 + (2\pi f_c)\ (T_{\mathrm{R1}}) \\ &= \Phi_0 + (2\pi\frac{c}{\lambda})(\frac{2d_1}{c}) \\ &= \Phi_0 + 2\pi\frac{2d_1}{\lambda}\end{aligned} \tag{17}$$

$$\begin{aligned}\Phi_{\mathrm{R2}} &= \Phi_0 + (2\pi f_c)\ (T_{\mathrm{R2}}) \\ &= \Phi_0 + (2\pi\frac{c}{\lambda})(\frac{2d_2}{c}) \\ &= \Phi_0 + 2\pi\frac{2d_2}{\lambda}\end{aligned} \tag{18}$$

$$\begin{aligned}\Delta\Phi &= \Phi_{\mathrm{R2}} - \Phi_{\mathrm{R1}} \\ &= 2\pi\frac{2[d_2 - d_1]}{\lambda} \\ &= \frac{4\pi\Delta d}{\lambda} \\ &= \frac{4\pi v\ T_c}{\lambda}\end{aligned} \tag{19}$$

Solving for velocity in the final equation in Equation 19 gives the result in Equation 20. However, it is important to note that since this measurement is based on the phase of successive chirps and that phase itself is circular, it possible for the phase to change so much that it aliases (i.e. there is an ambiguous measurement) [43]. Ambiguous measurements only occur if there is a phase change greater or equal to π [43] because the phase will wrap at this point. This occurs when the target moves more than a quarter of the radar wavelength between chirps as can be seen by Equation 19 when $\frac{\lambda}{4}$ is substituted for Δd and the resulting phase change become π.

$$v = \frac{\Delta\Phi\lambda}{4\pi T_c} \tag{20}$$

Setting the phase change $\Delta\Phi$ to a maximum of π in Equation 20 yields the formula for the maximum velocity shown in Equation 21 where T_c is the total time between chirps for a given receiver. This is important as T_c includes the idle time and ramp time for each chirp and includes the time between firings of a specific transmit antenna if the MIMO configuration is used. For MIMO configuration, this does mean that the maximum velocity is reduced when using TDM MIMO for each separated transmitter [48].

$$v_{\max} = \frac{\lambda}{4T_c} \tag{21}$$

Velocity resolution is determined intuitively and similarly to how the maximum range was determined from the range resolution, but in the other direction. The target velocity is consigned to discrete velocity bins according to how many chirps there are per frame as with the range bins. Therefore, the measurable target velocities are divided over the range $-v_{\max}$ to $+v_{\max}$ (which is double the maximum velocity) by the number of chirps, N_c. So, the velocity resolution is determined by substituting Equation 21 into Equation 22.

$$\begin{aligned}\Delta v &= \frac{2v_{\max}}{N_c} \\ &= \frac{\lambda}{2N_cT_c}\end{aligned} \tag{22}$$

Performing what is known as the Doppler FFT along the chirp axis of the radar cube resolves the velocities of different targets where the frequencies are proportional to the phase difference between chirps shown in Equation 20. The Range FFT and the Doppler FFT, together, form the 2D Range-Doppler FFT which produces the map shown in Figure 35. The bright yellow blob at the approximately 2 m range moving at approximately 0.3 m/s towards the radar is a person. The other less intense blobs are static clutter present at all ranges. This is the input to the CFAR algorithm discussed later.

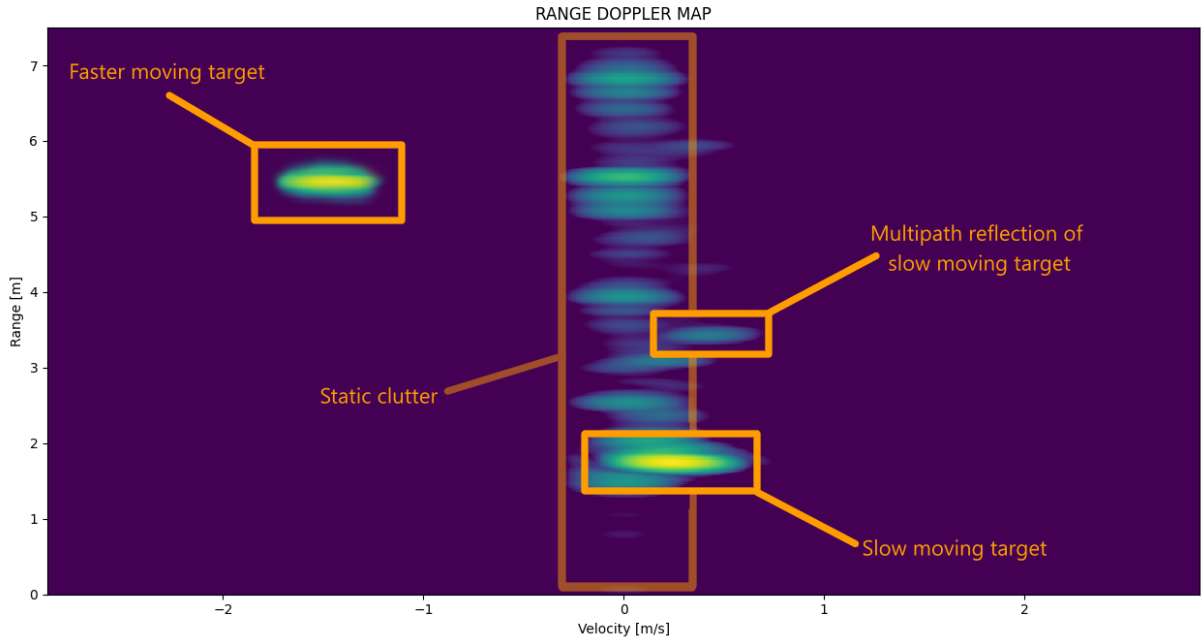

Figure 35: Range Doppler Map of two people walking back and forth at different speeds.

4.2.3 Chirp Parameters Chosen

Table 4: Table of important chirp parameters used.

Radar Parameters	
Parameter Name	**Value**
Number of Samples	96
Sample Rate	2200 [Ksps]
Sweep Rate	70 [MHz/μs]
Starting Frequency	77 [GHz]
Number of Chirps	32
Idle Time	100 [μs]
Ramp Time	57.14 [μs]
Frame Period	15.3 [ms]

Because the phase between consecutive chirps was being used to measure micro displacements, range resolution was not of particular importance. Velocity resolution and maximum

velocity were the important parameters for this study. The walking speeds in this dissertation were constrained to slower speeds of less than 2 m/s. Using Equation 21, the chirp period was determined as 157.14 μs to set the maximum velocity to 2 m/s (Taking into account that the actual chirp time when using MIMO is up to three times longer with three transmit antennas). Making this the ramp time would result in an extremely low sweep rate (so that 4 GHz bandwidth of the radar is not exceeded) which would make the maximum range much longer than it needed to be. The maximum range only needed to be at most 4 m as that was the amount of space easily available in the lab environment. Additionally, the FMCW radar used was relatively low power, so the RCS of anything beyond 4 m would have to have been very large. Therefore, this 157.14 μs chirp time was split into 100 μs of idle time and 57.14 μs of actual chirp ramp time. This allowed the sweep rate to be increased while using the full bandwidth of the radar and keeping the maximum range at a suitable value. The number of samples per chirp in Table 4 was chosen due to constraints on the data throughput of the system. From there, 2200 Ksps was chosen as the IF signal sample rate to maximise the sampled bandwidth of the chirp. The number of chirps was maximised to provide the best velocity resolution while conforming to data throughput constraints. With the number of chirps and the time per chirp known, the total amount of time taken to complete all the chirps was calculated. This was used to select the frame period. The frame period was set to the total amount of time taken to complete all the chirps. While this significantly increased data throughput it ensured there was minimal dead-time between the last chirp of a frame and the starting chirp of the following frame. This was done to minimise the risk of a participant moving more than a quarter wavelength between frames which would have caused ambiguous unwraps. The resulting radar performance metrics from the parameters chosen are shown in Table 5.

Table 5: Table of basic radar performance metrics calculated from Table 4.

Radar Performance	
Metric Name	**Value**
Range Resolution	4.91 [cm]
Maximum Range	4.71 [m]
Doppler Resolution	66.3 [Hz]
Velocity Resolution	0.126 [m/s]
Maximum Velocity	2.01 [m/s]

4.2.4 Range-Angle Processing

Angle processing requires there to be a spatial separation between different receiving elements. This separation is typically $\lambda/2$ where λ is the approximate wavelength of the chirp. The separation causes echoes from a target to take slightly different paths to reach antennas. In Figure 36, the path of l_4 is significantly shorter than l_1 due to the spatial separation of the receive antennas. This causes a difference in the phase of the chirp received at RX1 and RX4. This difference encodes angle information into the difference in phase between channels. This is why performing an FFT across the angle (i.e. the channel) dimension of the radar data cube gives the AoA information of a target.

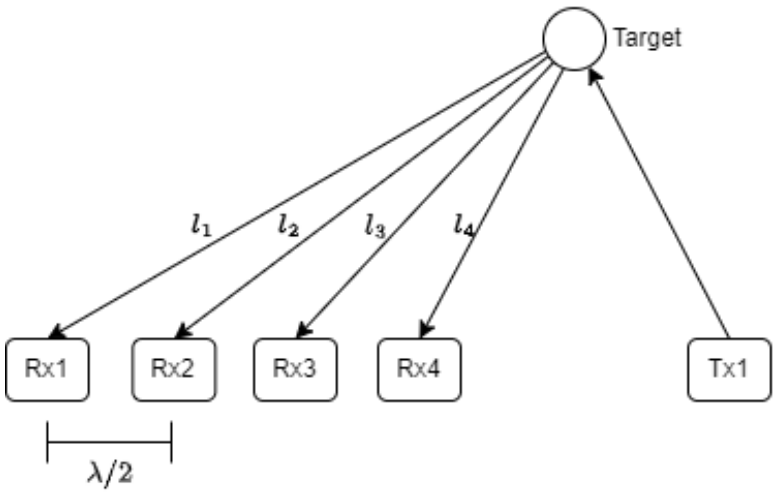

Figure 36: Echo path length change due to separation of antennas.

In Figure 36 only 4 receive antennas are shown whereas the bottom of antennas shown in Figure 32 has 8. Therefore, The maximum path length change of the MIMO configured radar shown in Figure 32 (the difference in path lengths between TX1-RX1 and TX3-RX4) will be much greater than the difference between RX1 and RX4 in Figure 36. This implies that the setup with more antennas (which will obviously have more separation between end antennas) will have a much greater sensitivity to angle information than a setup with fewer antennas. Additionally, there are more antennas which provides more samples in the FFT which creates more azimuth bins. Both of these ways of thinking highlight the fact that larger apertures or arrays provide better angle resolution. This is the same principle upon which Synthetic Aperture Radar (SAR) is built. This is why using the MIMO configuration is better as mentioned in Section 4.1.4. The MIMO configuration provides more antennas which improves the angular resolution by providing more bins [2, 47].

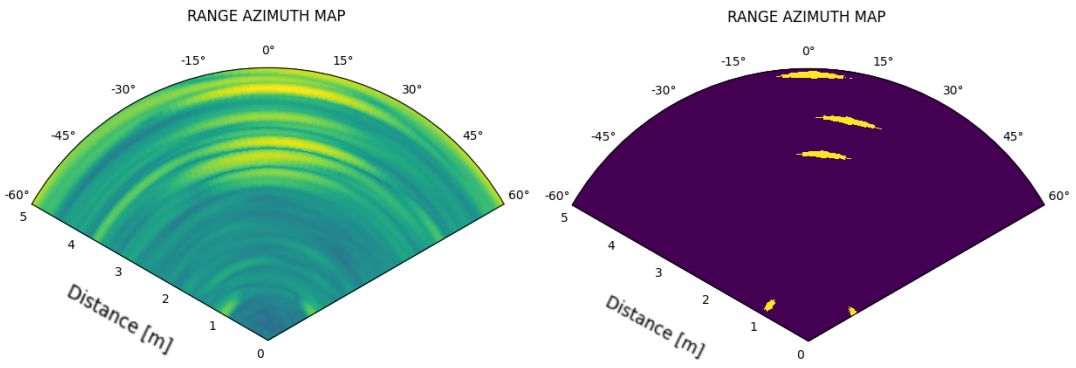

Figure 37: Output of AoA processing and CFAR. Scene contained a target at 3m down the bore-sight of the radar, another target just behind and to the right of the first and a wall behind both targets which are all clearly shown in the CFAR detection map.

In Section 4.1.4, it was also stated that Elevation Processing could be done. The terms of elevation and azimuth are arbitrary in the context of the processing done as it is the same processing for both. The difference arises from the spatial layout of the virtual antenna

pattern. Taking an Angle FFT over channels (1) to (8) provides azimuth while taking an Angle FFT over the coherent summation Channels (3) and (9) or (4) and (10) etc. provides elevation. This is due to the vertical, as opposed to horizontal, separation of the antennas which is only present due to the MIMO Configuration and vertical separation in the transmit antennas.

Taking the 2D FFT of the data cube for a specific chirp across the range and angle axis provides a range-angle map. Limiting the FFT in the angle axis to the channels (1) through (8) in Figure 32 provides the range-azimuth Map shown in Figure 37.

4.2.5 CFAR Detection Algorithm

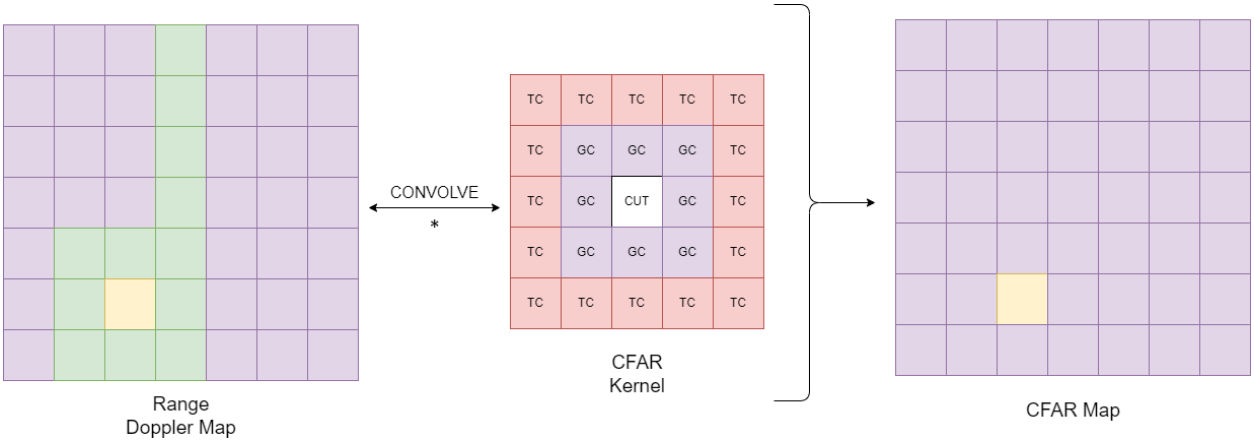

Figure 38: Diagram explaining the CFAR process: purple cells are points of very low power or noise, green cells are cells with higher power but not a target, yellow cells represent the target with high power. Cells labelled TC in the kernel are training cells, cells labelled GC are guard cells and the cell labelled CUT is the test cell.

CFAR is a statistical detection algorithm used for pulling peak locations out of data by comparing the power within a test point to the power of surrounding points. Cells very close to the Cell Under Test (CUT) are often ignored and are called guard cells (GC). This is because the main-lobe of the target can occupy multiple cells. Cells used in the power comparison are called training cells (TC). These cells are placed within a CFAR kernel as shown in Figure 38. The power in all the training cells is summed and this total noise power multiplied by a scaling factor is compared to the power in the CUT If the power exceeds the threshold created by the noise and scaling factor, it is a target. The scaling factor is determined statistically given the number of training cells, N_T, and the desired probability of false alarm, P_{fa}, as shown in Equation 23 taken from the *Fundamentals of Radar Signal Processing* [49].

$$\alpha = N_T(P_{\text{fa}}^{\frac{-1}{N}} - 1) \tag{23}$$

Every cell in the 2D range-Doppler or range-azimuth matrix is subjected to the kernel to generate the CFAR map. This can be done by sliding the kernel through all the rows

and all the columns of the matrix. However, this is very computationally inefficient [49]. Sliding a kernel over space is an already well established mathematical tool defined by the convolution operator. Convolution in the frequency domain is just a simple multiplication [50]. Therefore, CFAR is implemented with the following steps:

1. Zero pad the kernel to match the size of the input matrix (the range-Doppler or range-azimuth matrix).
2. Perform Convolution:
   (a) Perform a 2D FFT on the kernel and power of the input matrix (The square of the magnitude of the complex input matrix).
   (b) Multiply the resulting matrices together.
   (c) Take the inverse 2D FFT of the new matrix to get the Noise Power Matrix.
3. Get the indices of the Input Matrix Power that are greater than Noise Power Matrix multiplied by the scaling factor from Equation 23.
4. The indices of the matrix that return true are labelled as targets within the CFAR detection map.

4.2.6 RealSense Target Tracking

Figure 39: Example output of the RealSense depth data. Image shows the depth of a lab with a person standing in front of the camera.

The depth of a target using the RealSense camera was determined by hand selecting pixels corresponding to the chest. While ideally a machine learning network would be used to select the desired pixels, such content was outside the scope of this dissertation. The results of the selection are shown later in this dissertation.

The person in visible in Figure 39 stood a position marked out as 2 m from the system. The centre of the chest was clicked on during the video to place the cross-hair shown in white in Figure 39. This selected the pixel used to calculate depth. The depth was calculated as 1.85 m which seemed accurate considering any errors in the marking of the position and that the person's chest may not have been exactly over the 2 m marker. Clicking on other points in the scene belonging to static objects confirmed that the data was correct when these measurements were compared to hand measurements performed afterwards.

4.3 Tracking Pipeline Validation

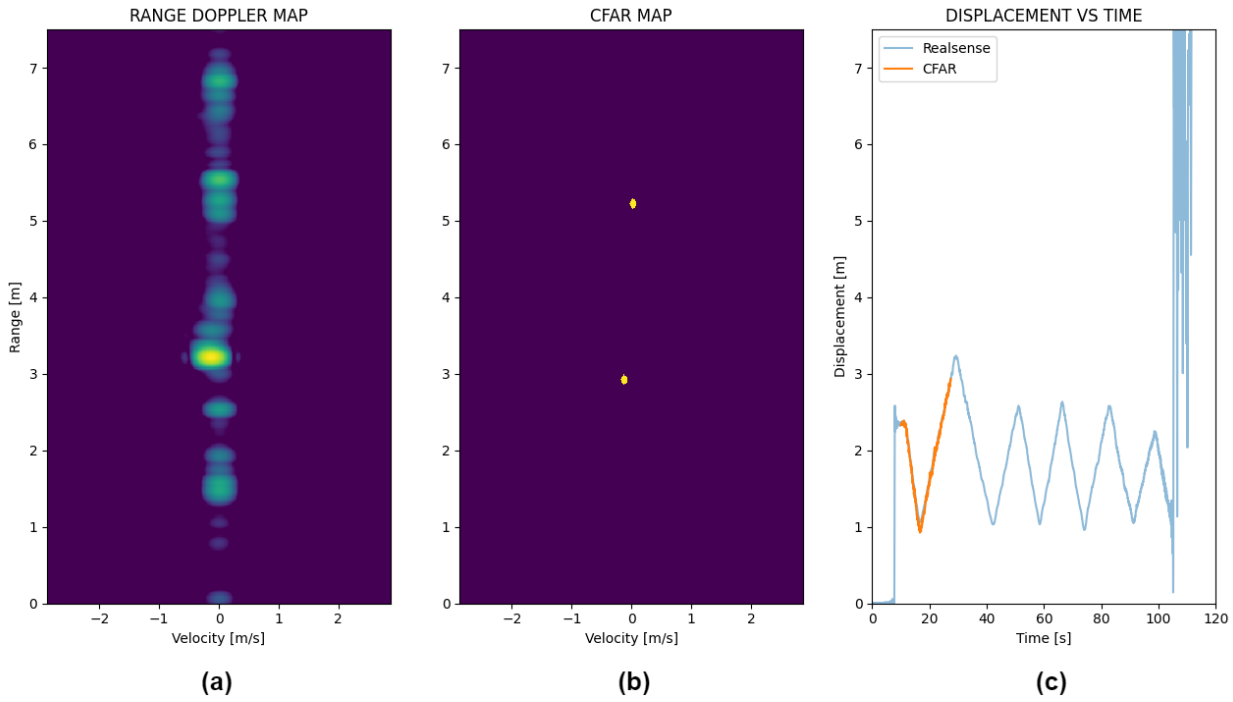

Figure 40: Example output of a frame of the animated tracking pipeline representing the displacement of a person walking back and forth. **a**: Output of Range-Doppler Process. **b**: Output of the CFAR process. **c**: Comparison between CFAR output and RealSense output.

Getting the depth from the stereo camera as discussed in Section 4.2.6 and performing the Range-Doppler and CFAR processing discussed in Sections 4.2.2 and 4.2.5, yields the output of the tracking pipeline shown in Figure 40. The Range-Doppler plot on the left in Figure 40 is as expected with a strong target just beyond 3 m moving away from the radar and some static clutter occupying the zero Doppler bins.

The CFAR performs as expected as shown in the middle plot of Figure 40. It successfully

extracts the target range and velocity. There is an additional target extracted but is ignored because it is stationary and does not change between frames.

The right-hand plot in Figure 40 compares the output of the Range-Doppler and CFAR processes to the output of the stereo camera depth extraction. The two results agree with each other well within an acceptable margin of error, validating that the pipeline is working.

4.4 Filtering

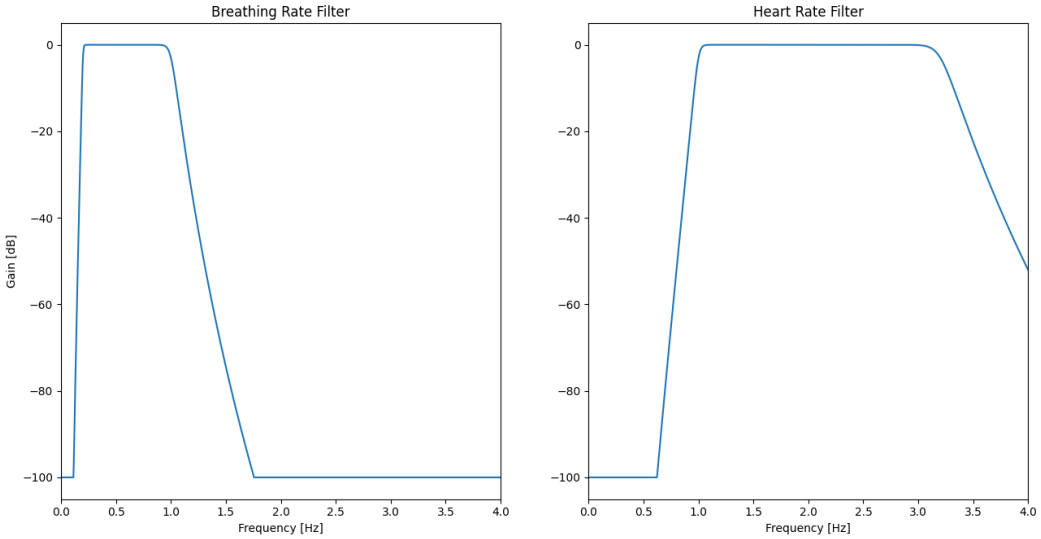

Figure 41: Magnitude response of filters used for vital sign extraction.

4.4.1 Filter implementation

Before discussing phase reconstruction, vital sign measurements and motion compensation, the filters used ubiquitously in these processes need to be mentioned.

The digital filters used in this dissertation were implemented using the SciPy package in python. SciPy has its own functions for Infinite Impulse Response (IIR) filter design. One caveat of IIR filters, is that they have non-linear phase. Even thoughFinite Impulse Response (FIR) filters have linear phase, they were not used due to the extremely high sample rate of the data and slower response time compared to IIR filters [51]. The non-linearity of the IIR phase was removed by running the filter twice in different directions. Using SciPy was acceptable as data only had to processed in post and not in real-time on embedded hardware (as this was outside the scope of the project), so short development time was preferred over stringent filter design. SciPy was used frequently in this dissertation because of its ease of use and extensive functionality [52].

SciPy recommends using the Second Order Section (SOS) implementation for its digital filters [52]. A Second-order Section representation is a representation of a digital IIR filter as a series of Bi-Quad filters. The typical implementation of a Bi-Quad filter is shown in Figure 42. The SOS representation is stored as a N_{sec} by 6 matrix where N_{sec} is the number of second order sections. Each column of the matrix represents a different filter coefficient. The first three coefficients represent the numerator and the second three represent the denominator coefficients. Changing these filter coefficients creates different 2nd-Order high or low pass filters. Applying different combinations of these filters creates higher order low or high pass filters or band-pass or band-stop filters.

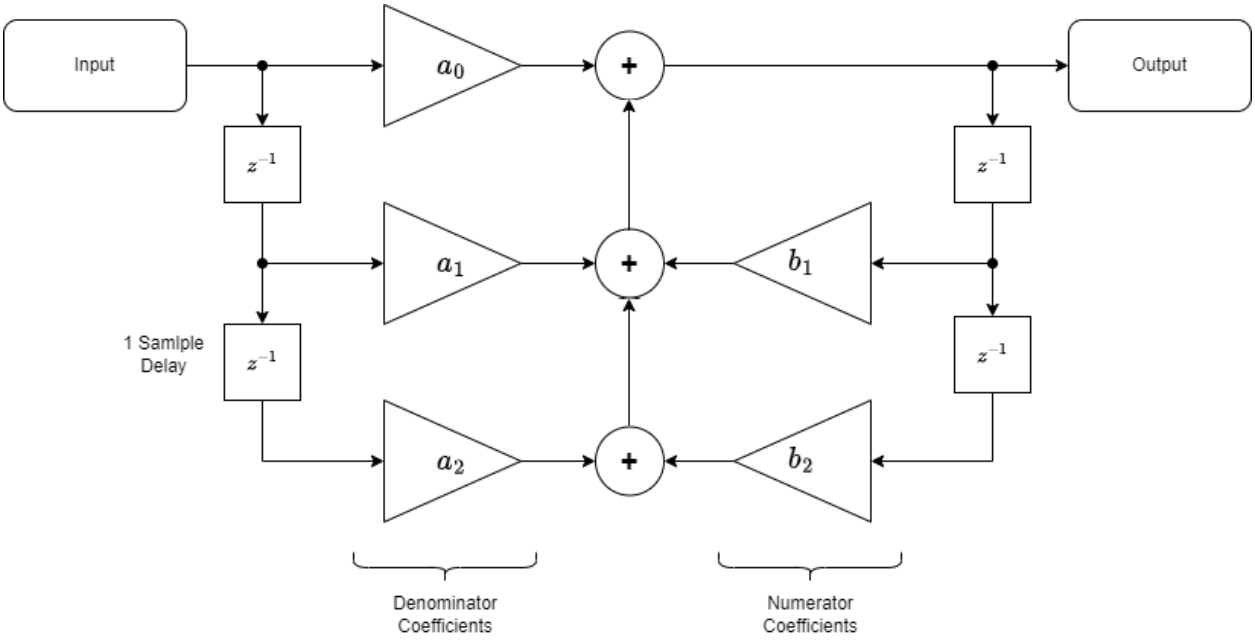

Figure 42: Diagram of digital implementation of a Bi-Quad Filter.

4.4.2 Zero-phase Filtering

To avoid distortions in the pass band, Butterworth filters were used as they have no pass band ripple as seen in Figure 41. Additionally, the filters were required to have linear phase responses to ensure the outputs of the filters were not distorted over time and, to keep various signals synchronised together, no filter delay was permitted. No filter delay can only be achieved using non-causal filters. Running a filter forwards and backwards through the data achieves this effect by creating a filter with a zero (and, therefore, also linear) phase response. This is realised by using SciPy's sosFiltFilt function [52]. Using this implementation doubles the order of the filter because the filter is effectively applied twice, once forwards and once in reverse. The effect of this forward-reverse two pass filter is seen in Figure 43.

The first pass of the filter output shown in Figure 43 has a visible delay. The signal is then reversed, passed through the same filter and then reversed again. The output of the second pass aligns perfectly to the original signal and any phase distortions and or delays have been

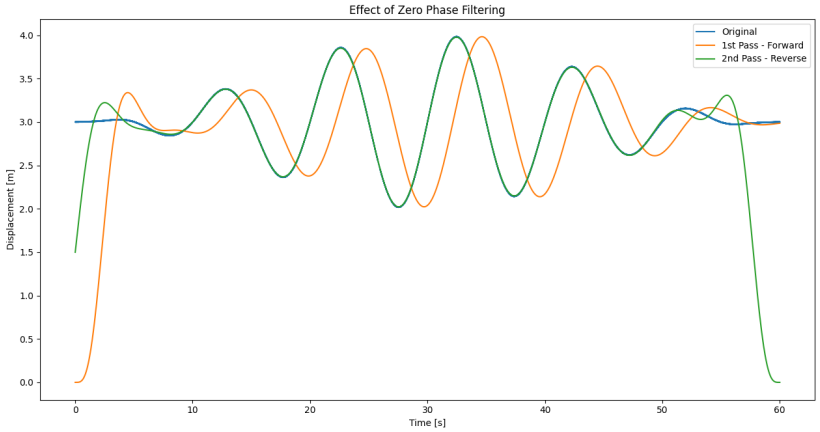

Figure 43: Comparison of a signal as it moves through a low pass, zero phase non-causal filter.

removed. This type of filter would need to process data in chunks when implemented in "Real-Time" systems as the entire signal needs to be known before-hand to do non-causal filtering.

4.4.3 Ringing Elimination

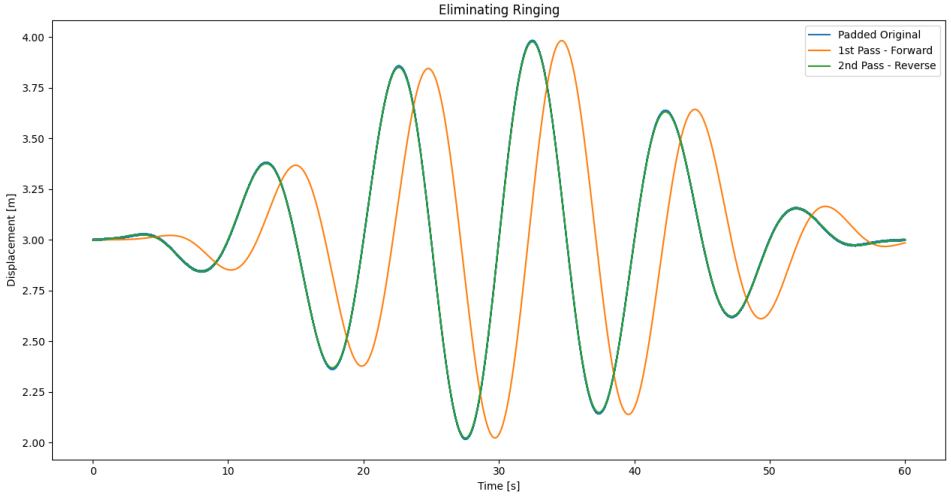

Figure 44: Output of non-causal filter when input signal is edge padded first.

However, in Figure 43, there is a large amount of ringing visible at the ends of the filter output. This was fixed by edge padding the input to the filter (although there are other ways to initialise a filter) and then truncating the output back to the original signal length. The effect of edge padding can been seen when Figure 43 is compared to Figure 44 where the ringing has been removed. The signal needed to be edge padded by about 40 s worth of samples (100000 samples at a sampling rate of 2.5 kHz) to achieve the result shown in

Figure 44. The signal is then truncated back to its original length by removing the amount that was padded on each side.

4.4.4 Filters for Up-Sampling

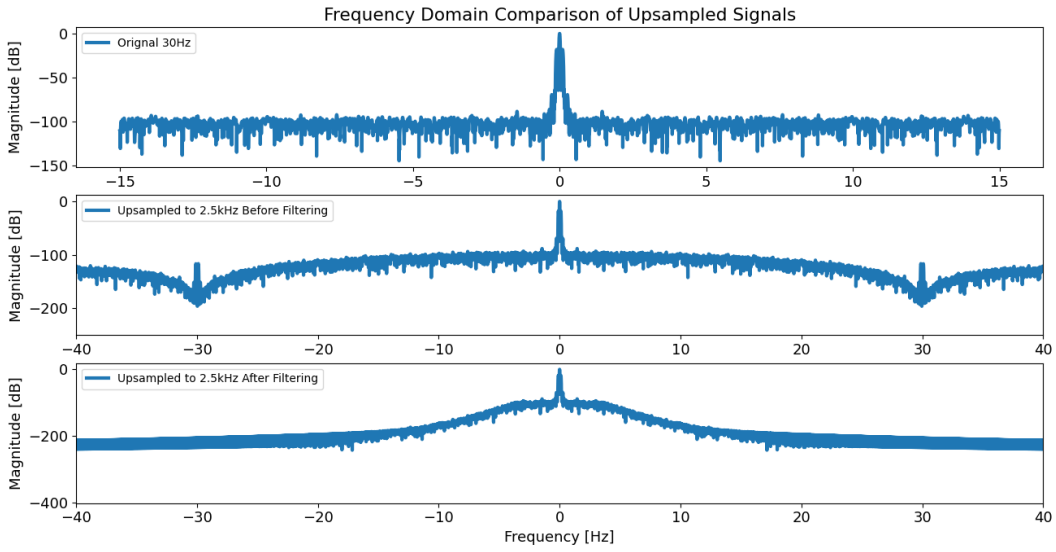

Figure 45: Output of Up-Sampling in Frequency Domain.

Due to varying sample rates (discussed later, in Section 4.5.3) different signals need to be upsampled and synced together so they have the same number of elements corresponding to the same points in time. Doing this causes spectral replicas in the frequency domain. To remove the replicas a low pass filter is used. Thankfully, the region of interest of 0Hz to 4Hz occupies a portion of the spectrum that does not contain these replicas.

4.5 FMCW Vital Sign Processing

4.5.1 Vital Sign Pipeline Overview

The vital sign monitoring pipeline uses the output of the tracking pipeline in Section 4.2.1 in two ways.

1. **Range Bin Selection**: The tracking pipeline is used to get the desired range bin of the target to get the phase from. Using outputs from the tracking pipeline enables dynamic range bin selection to follow a moving target.
2. **Motion Compensation**: The tracking pipeline not only outputs the target range bin but also the target depth. This is used to remove large scale motions from the phase reconstruction of the data.

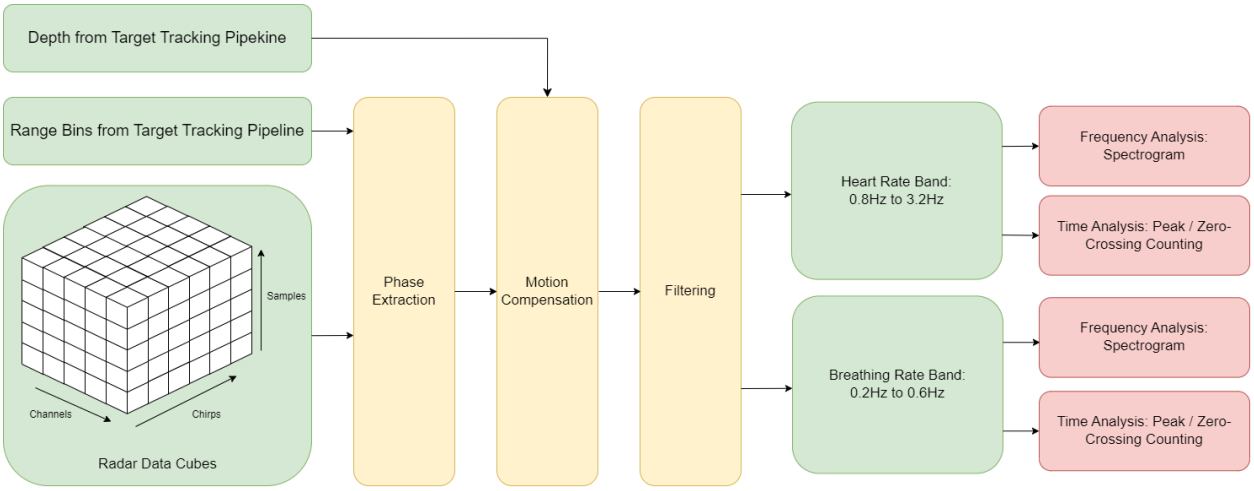

Figure 46: Overview of the vital sign processing pipeline. Inputs are highlighted in the green blocks. Processes are highlighted in the yellow blocks. The analysis processes are highlighted in the red blocks.

The range bins from the target tracking pipeline and the data cubes form the input to the phase extraction algorithm discussed in Section 4.5.2. This process produces a target displacement over time as an output. This displacement is compared to the displacement measured by the radar tracking pipeline and the RealSense Stereo Camera. The two displacements are subtracted from each other to attempt to compensate for the displacement introduced by large scale motions such as walking.

The compensated signal is then passed through two band-pass filters in parallel to separate the components of the compensated signal within the heart rate and breathing rate frequency bands. The design of these filters is discussed in Section 4.4.

The heart rate and breathing rate signals are analysed separately in both the time and frequency domain. Frequency domain analysis is done using a spectrogram to get a view of frequency over time while time domain analysis is done by counting the number of zero crossings within the signal.

4.5.2 Radar Phase Measurements

Phase measurements with radar, essentially, work off of the same principles discussed in Section 4.2.2 concerning Doppler processing. In Equation 19, phase is highly sensitive to small changes in distance because the wavelength, λ, is in the millimetre range. Even a displacement of 1 mm (assuming a wavelength of 4 mm) results in a phase change of more than π radians which is almost ambiguous as phase wraps for changes greater than π. This enables sub-millimetre measurements of a target displacement by measuring the phase of successive radar chirps. This is different to Micro-Doppler processing which relies on radars with very high Doppler resolution to detect subtle Doppler shifts caused by sub-millimetre measurements and therefore the processing chain is also slightly different.

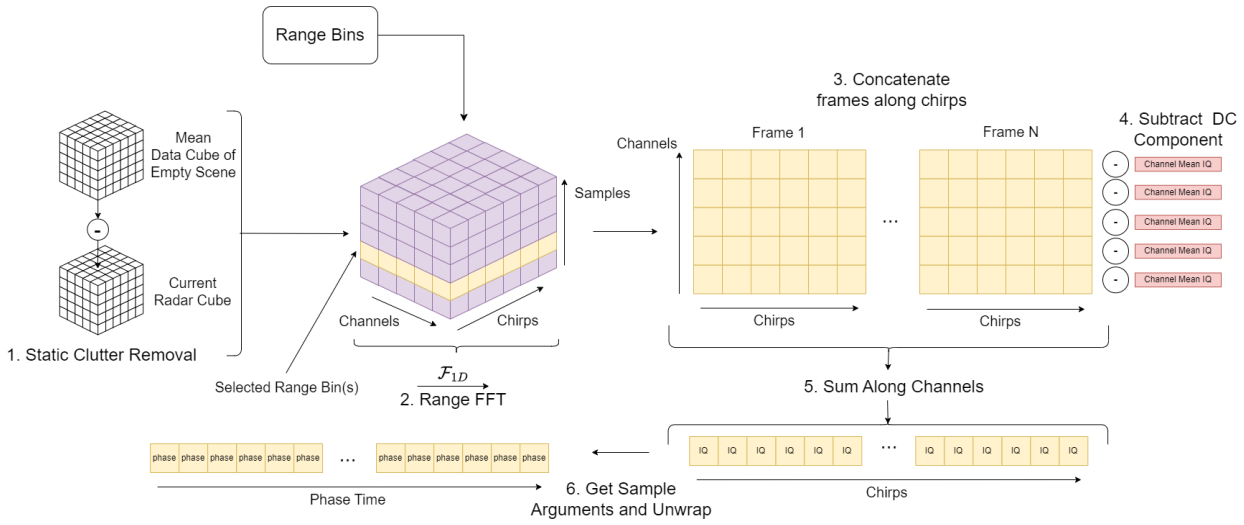

Figure 47: Diagram describing process of how phase is extracted from the radar data. The rang-bins selected over time can be provided by an external sensor (RealSense Camera).

The phase for a particular target is the argument of its complex sample corresponding to the range bin or range bins that the target occupies. So the pipeline for extracting phase starts by doing a range FFT. A 2D matrix is extracted from the data cube. This matrix contains all the samples from each channel (the rows of the matrix) for each chirp (the columns of the matrix). If multiple range-bins are selected, a complex summation takes place similar to Equation 12 along the range dimension. This selection process is performed for every radar frame and the range bin selected is updated dynamically by the tracking pipeline. These selections are concatenated together to produce a matrix with rows equal to the number of channels and columns equal to the number of frames times the number of chirps per frame.

Each row of the matrix undergoes a subtraction, whereby the complex mean of all the chirp samples in the channel is removed as in [9]. This is done to correct for IQ imbalances present in the antenna which cause errors in the phase measurement. Thinking about why this happens in terms of the argument of a sample in the complex plane: a DC offset means that certain samples that would been far from the origin instead become very close to the origin. The closer a sample is to the origin the greater the range of angles the sample can produce as shown by the right hand plots in Figure 48. Noise can be modelled as a normal like distribution of possible points around (blue) a single true value (orange). In the right set of plots in Figure 48 the red lines denote the maximum and minimum possible arguments that can be calculated. The cone these lines form is much narrower for the high SNR case showing less susceptibility to noise. The low SNR case is what leads to erroneous argument calculations using arc-tangent demodulation. SNR is improved by adding all the channels together after DC offset compensation to increase the magnitude of all the IQ samples. The increase in signal power is significantly higher than the increase in noise power because the noise in each channel in uncorrelated.

Taking the argument of each complex sample produces the phase of each chirp. The argument is found using arc-tangent demodulation as described in Equation 24 [9]. However, because

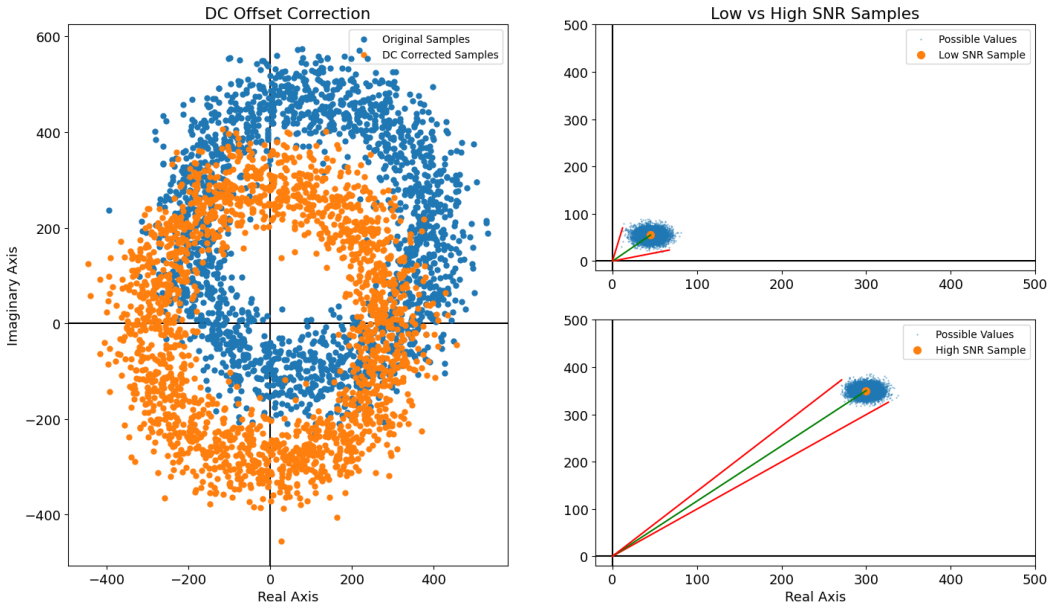

Figure 48: **Left**: Example scatter plot of IQ samples before and after DC correction. **Rigt**: Comparison of High SNR vs Low SNR samples.

phase is circular it wraps for values greater or less than π and $-\pi$ respectively. Therefore, the phase needs to be unwrapped whereby jumps caused by wrapping are determined and every sample after the jump is added or subtracted by 2π depending on the direction of the jump. The output of the unwrapping algorithm is a smooth trajectory of the change in phase over time. Re-arranging for the displacement, Δd, in Equation 19, relates the change in phase chirps to the displacement undergone by the target.

$$\Delta\Phi = tan^{-1}\left(\frac{Q(t)}{I(t)}\right) \tag{24}$$

Multiplying the change in phase over time by $\frac{\lambda}{4\pi}$ gives a final displacement of the target in metres that should be able to contain motions with an amplitude in the sub-millimetre range as shown in Equation 25.

$$\Delta d = \frac{\lambda}{4\pi}\Delta\Phi \tag{25}$$

4.5.3 Motion Compensation

Element-wise operations (such as subtraction) between time-series of different lengths is not possible because those operations need a 1-to-1 mapping of elements.

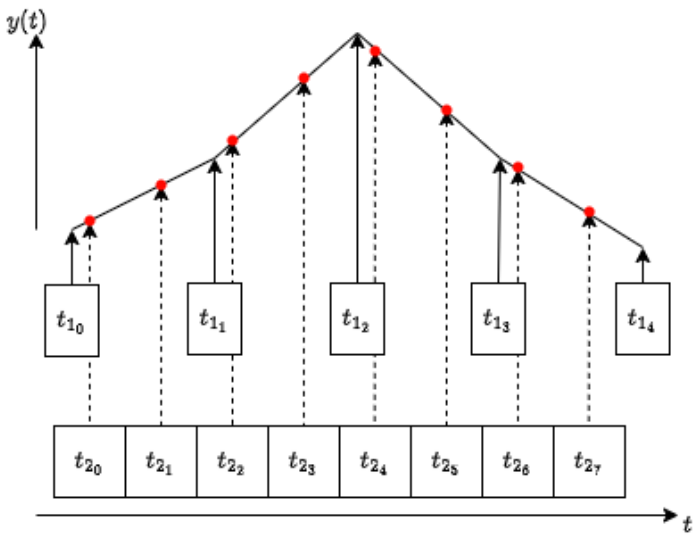

Figure 49: How a time series $y(t)$ corresponding to time t_1 is re-sampled and synced to a time series with time t_2 using linear interpolation. The red point correspond to the newly re-sampled signal.

The phase produced in the extraction algorithm is sampled according to the spacing between chirps. The spacing between chirps is typically in the region of 100's of μs. Therefore, the sampling rate of the phase-time signal is often in the kHz range. However, the signals used for motion compensation are limited to the frame rate of the devices used to capture the compensation signal. These sampling rates are in the order of 10 Hz to 100 Hz. Therefore, for a recording of a given duration, the number of samples present in the phase signal vs the number of samples in compensation signals available is much greater.

Therefore, the motion compensation signals generated by the radar and RealSense Camera need to be up-sampled and time-synchronised to match the phase before the subtraction required to compensate for motion can be done. This is done using linear interpolation as shown in Figure 49. Linear interpolation is specifically used because the time axis is not entirely uniform. Fluctuations in frame rate occur during recording which can lead to distortions in the time domain when using Whittaker-Shannon Sinc Interpolation or FFT-inverse FFT based interpolation to up-sample signals. Using linear interpolation, allows for a custom implementation that ensures exact time syncing of both signals over just making sure there is a 1-to-1 mapping between samples. After the up-sample process is complete, a low-pass filter is applied to remove any spectral replicas as mentioned in Section 4.4.

With all signals time-synced to the phase reconstruction, simple subtraction in different frequency bands can be performed to attempt to compensate for random motion artefacts.

4.5.4 Vital Sign Phase Spectrograms

The spectrogram is probably one of the most basic, yet fundamentally important tool, in analysing the frequency domain and how it changes over time. A huge advantage of the

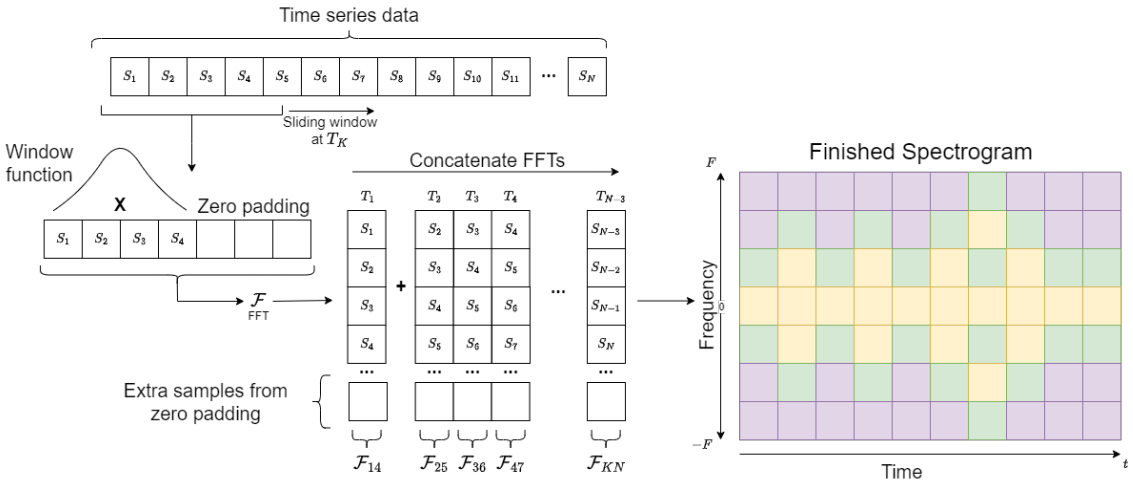

Figure 50: Overview of process to perform a spectrogram.

spectrogram over simply taking the FFT over the entire observation window of a signal is that the spectrogram can show how the frequencies in a signal change over time. Simple FFTs are not robust to non-stationary signals, signals that have changing components, while spectrograms can measure these changes (provided it does not change too much within the spectrograms observation window).

Spectrograms are preformed by breaking a signal into many overlapping observation windows. The percentage overlap can be adjusted to make the process computationally faster but more course in time (less overlap) or slower with smoother transitions (more overlap). This effect is achieved by a sliding window that sequentially selects portions of data. The data is windowed and then zero-padded before being placed into an FFT. Every new selection of samples produced by the sliding window produces a new FFT. Each FFT forms a column which are then all concatenated together to form a 2D matrix. The rows of the matrix correspond to different frequencies and the columns correspond to time. The negative frequencies produced by the FFT are just a mirror of the positive frequencies in magnitude in the context of time series analysis and so are not plotted. The process for performing spectrograms on range-Doppler maps is essentially the same. A set of range-bins are selected and summed across the range axis for each frame to produce a vector where each element is the sum of range bins of a single chirp. Stitching all these vectors from each frame together creates one long 1D array. This 1D array serves as the new time series input to the spectrogram function. In this case, negative frequencies are not mirrored because the frequency of the peak in the FFT will be positive if moving towards the radar and negative if moving away.

Spectrograms also beautifully highlight the trade offs between frequency and time resolution as shown in Figure 51. Longer windows provide better frequency resolution but are not robust to signals which have frequency components that change too quickly. Shorter windows create very wide peaks in the frequency axis that can easily merge into each other but provide much greater temporal resolution and allow us to capture signals that change quickly in their

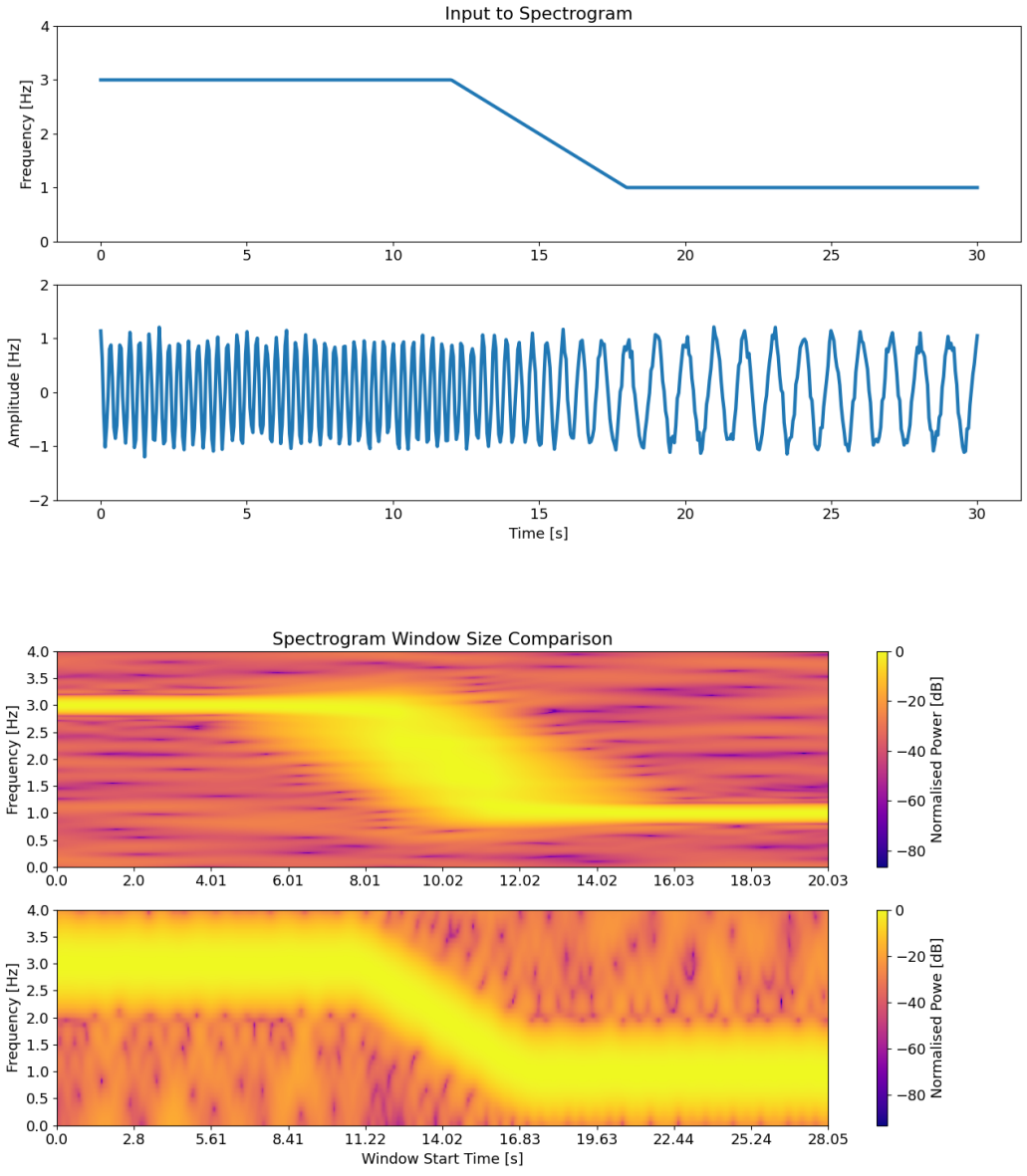

Figure 51: Example of effects of different window sizes on a spectrogram. **Top**: Frequency of the input sinusoid over time followed by the noisy sinusoid with varying frequency in time. **Bottom**: Spectrogram of the sinusoid using a 10s window. followed by a spectrogram of the sinusoid using a 2s window.

nature. The shorter the window, the quicker the change in frequency can be captured. For example, in the context of vital signs, if there is a situation where a participant has a very high heart rate for a long period of time that then suddenly drops to a low a heart rate, a long window will excel in capturing the steady states of the high and low heart rates but will struggle when the window contains a jump. A shorter window will cause wide peaks that might make it hard to work out the exact vital sign but will excel in capturing the jump from low to high frequencies. In the 10s window example, the frequency peaks associated with the steady state are very narrow, but the transition period and around the transition period is very blurry and little information is accurately captured about the transition.

Clearly, choosing the appropriate window size is important for spectrogram analysis. Vital signs can change rapidly in extreme situations but in typical cases (thinking about daily life) if they do change it is a minor fluctuation or is a gradual change as the body adjusts breathing and heart rate to cope with stresses such as exercise. This dissertation focuses more on the detection on vital sign signals at some steady state. Therefore, a longer window is probably preferred. Results for different window sizes on breathing data is shown in Table 52. A window size of 10 seconds was chosen because it was in the middle of plateau.

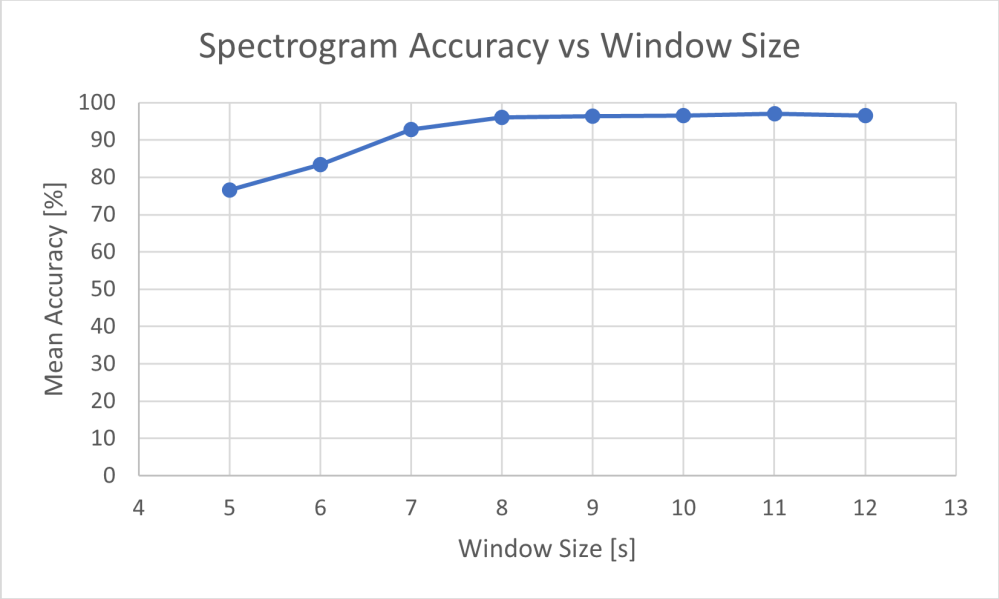

Figure 52: Mean accuracy vs window size for spectrograms of heart rate data.

To confirm the spectrogram algorithm was working it was tested on pseudo time-series data of a known frequency. Two sine waves were created and noise was added. The first sine wave had a randomly varying frequency centred at 0.5 Hz with an amplitude of 2 mm. The second, smaller, wave shared the frequency of the first but was shifted to 2 Hz and had an amplitude of 0.2 mm.

Two spectrograms were created to also check if the filters used were working correctly. The second spectrogram was run using data filtered to the heart rate band. The results are shown in Figure 53

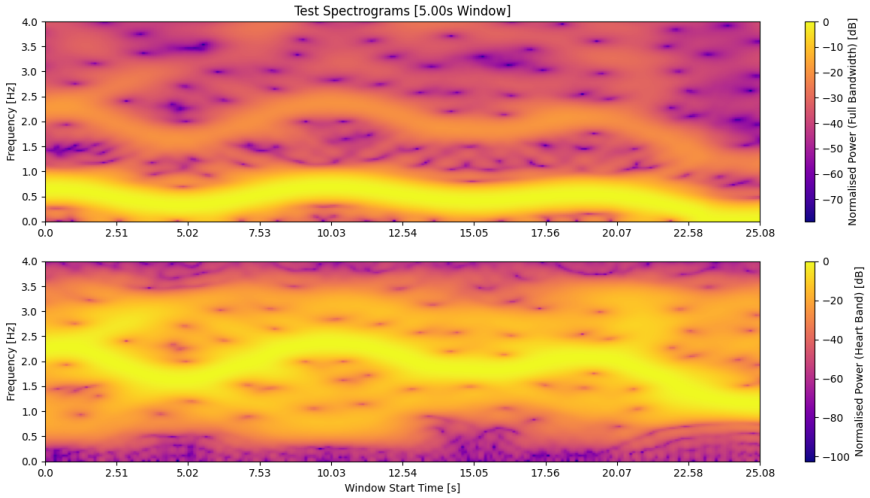

Figure 53: Figure validating the spectrogram algorithm used. **Top**: Spectrogram of the unfiltered signal containing two waves of different amplitudes and slightly varying frequencies at 0.5 Hz and 2 Hz. **Bottom**: Spectrogram of the signal after the larger wave was filtered out.

The top spectrogram clearly shows the correct frequencies and the bottom spectrogram clearly shows that the larger component has been removed.

5 Validation Methodology

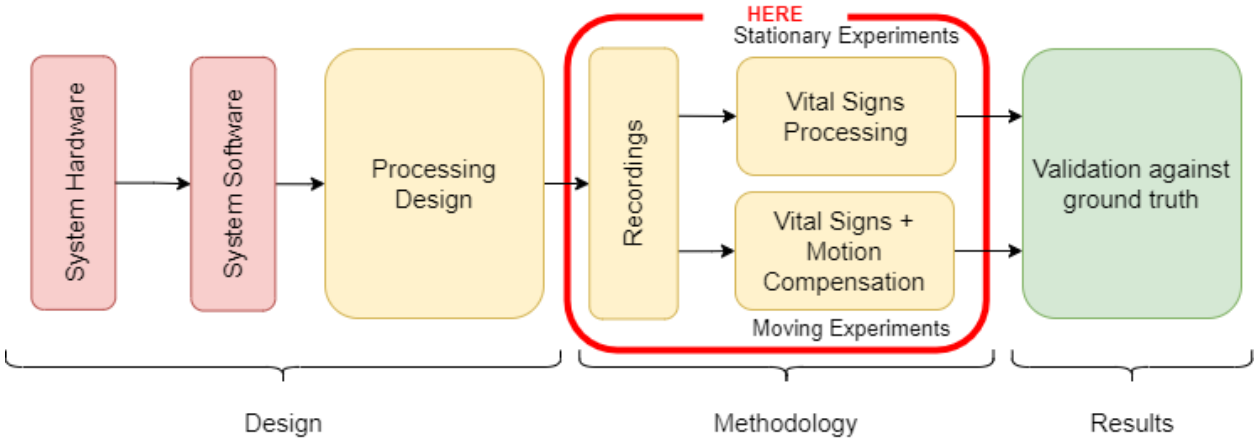

Figure 54: Figure of dissertation layout: Experimental Setup.

This section outlines the procedures used to conduct experiments to test the system and pipeline developed the preceding sections. It also serves as an explanation for how the data was validated and what metrics were used to characterise the performance of the system.

5.1 Experimental Plan

As mentioned in Section 1.2, there are three different layers of motion measured by the radar summarised as: gross motions (RMB), respiratory/breathing motions and cardiac/heart beat motions. Before trying to measure these motions, the outputs of the system and pipeline were validated used a test rig. After the pipeline outputs were validated as correct, to show that the system works as intended at measuring each type of motion, the different layers were added one by one, slowly growing the complexity of the problem. This was done to show exactly where the system pipeline would fail if it did fail. Before attempting complex motion compensation with a walking human participant, vital sign data for moving targets was simulated to get an idea of what to expect and potential issues to look for.

5.1.1 Pipeline and System Validation With Test Rig

The test rig shown in Figure 55 was used to validate the pipeline. As a strong metallic object close to the radar, it had a very high RCS which meant that the likelihood of the radar producing any low magnitude samples was very low. This meant that the phase measurements using this data were unlikely to contain any erroneous unwraps. So it was an excellent way to test the pipeline itself, without worrying about complexities introduced by having low signal power.

The rig, driven by a stepper motor and belt-drive, was made to move according to a known trajectory which was recorded by the rig and timestamped. This allowed various measure-

Figure 55: Linear motion rig used to validate the pipeline.

ments, including the radar phase, RealSense and others, to be compared to a known, ground truth, trajectory. Comparing the measurements to the output of the rig was how the pipeline were validated.

5.1.2 Stationary Vital Signs: Heart Rate Only

Figure 56: Setup of seated stationary experiments for vital sign measurements.

The next set of experiments explored human vital signs by starting with the smallest layer of motion, the cardiac layer. This layer was isolated by removing all other layers. Therefore, to show that heart rate could be accurately measured with the radar, the participant held their breath and remained absolutely still while seated in front of the radar as shown in Figure 56. This was a good starting point in assessing the radar's ability to measure vital signs (by seeing if it could correctly measure the smallest layer).

Several experiments were conducted to validate that the heart rate was correct. These were grouped into two categories: exerted and resting. This was done to ensure the correct heart rate was being measured and was not just coincidentally correct. Furthermore, eight 30 s recordings were attempted for each category of experiment as shown in Table 6. A 30 s interval was chosen because that was the period of time anyone should have been able to hold their breath for and more importantly provided more than enough time for frequency analysis which, as per the literature, usually requires a 5-12 s observation window [10, 16]. The pool size of eight recordings per category was attempted to provide an adequate sample size for statistical analysis. Ground truth heart rate data was provided by a Polar H10 heart rate monitor which has been done in many vital signs research papers (particularly if some kind of ECG is not readily available) [10, 17, 18, 19].

Table 6: Table of proposed experiments for heart rate only vital signs to produce 16 30 s measurements.

Heart Rate Only Experiments		
Recording No.	Resting (50-100 bpm)	Exerted (125-175 bpm)
1-16	8x30s	8x30s

In reality, 30 s recordings were too long for the exerted heart rate experiments because asking the participant to hold their breath for that long was not feasible while their heart rate was exceeding 125 bpm. The recording length for these experiments varied between 6 to 12 s. Additionally, instead of taking short recordings, longer recording sessions were held and then sections of data where the participant could hold their breath were isolated. This was procedurally more efficient and also provided experiment data for the next section. Additionally, this method provided data that nicely showcased the effect of breathing on heart rate measurements as periods of breathing and not breathing could be compared in the same recording.

5.1.3 Stationary Vital Signs: Heart Rate and Breathing Rate

After confirming that heart rate was accurately measurable with the radar, the next layer of motion, the respiratory layer, was added, thereby gradually increasing the complexity of the problem.

To do this, a similar approach to the previous set of experiments was attempted. However, two more categories were introduced, doubling the number of experiments. These two categories represented two different breathing rates (15 Respirations Per Minute (rpm) and 30rpm) as shown in Table 7. The participant was once again asked to sit still in a chair in front of the radar as in Figure 56. However this time, the participant timed their breathing to a GUI element which indicated when to breathe in and when to breathe out in accordance with the desired rate. This served as a ground truth for the respiratory rates calculated by the vital sign pipeline. However, the Polar H10 monitor also provided accelerometer data. Additional validation was provided by looking at the frequencies contained in the

accelerometer data. Since the participant was seated and still, the only data picked up from accelerometer was the respiratory motions which made it easily to analyse. Ground truth for the heart rate was again provided by the Polar H10 module.

Table 7: Table of proposed experiments for simultaneous measurement of heart rate and breathing rate for for two different breathing rates and two different heart rate zones producing 16 30 s measurements.

Breathing and Heart Rate Experiments			
Breathing Rate	Recording No.	Resting (50-100 bpm)	Exerted (125-175 bpm)
15 rpm	1-8	4x30s	4x30s
30 rpm	9-16	4x30s	4x30s

Again, in reality, there were difficulties with exerted experiments. As mentioned in the previous section, exerted breathing data was isolated from longer recordings. This did unfortunately reduce the pool size as it was difficult for the participant to sit still when so exerted and difficult to maintain such a high heart rate. Additionally, it was impossible for the participant to time their breathing with the GUI effectively while in such an exhausted state. This made the validation from the Polar heart rate monitor's accelerometer data invaluable in providing reliable ground truth data.

5.1.4 Simulation Work

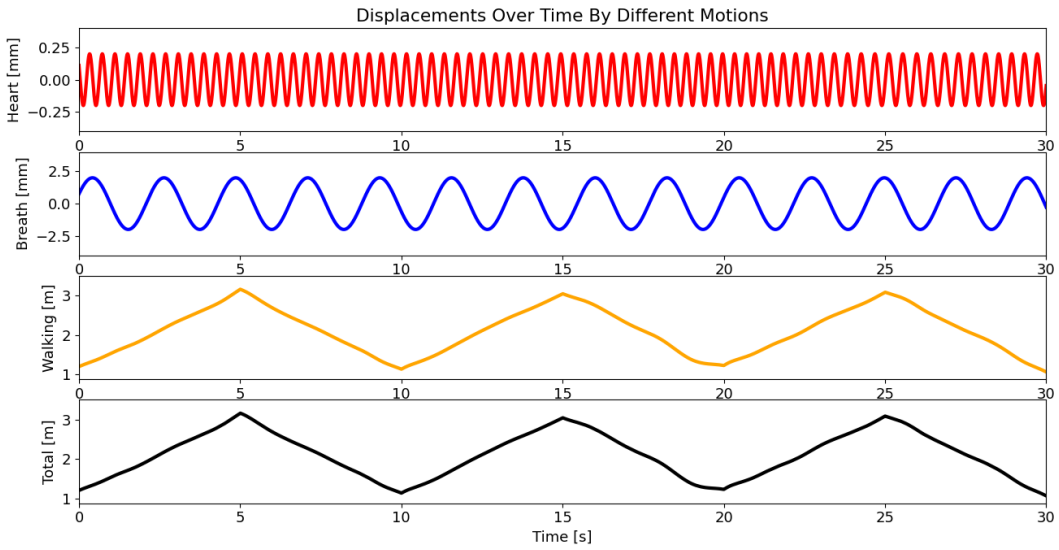

Figure 57: Construction of the simulated signal.

Exploratory simulation work was done before complex motion compensation was attempted to try examine the effects of various factors such as mismatch between the radar phase data and compensation signal, noise, etc. The simulation was created using time-series

data made to mimic a person walking linearly back and forth. This would contain the different motions described in Equation 26, where $d_T(t)$ is the total motion measured by the phase reconstruction obtained from the radar. This is made up of the gross macro motion $d_g(t)$ (RMB), the respiratory motion sinusoid with amplitude A_r and frequency f_r, and the cardiac sinusoid with amplitude A_h and frequency f_h. The variable $n(t)$ represents any noise in measurement. These different layers are shown in Figure 57.

$$d_T(t) = d_g(t) + A_r sin(2\pi f_r t) + A_h sin(2\pi f_h t) + n(t) \tag{26}$$

The vital sign signal amplitudes, A_r and A_h, were set to the values determined by literature (which later were confirmed experimentally but were supposed to be around 0.2mm for cardiac activity and approximately 2mm for respiratory movements). The vital sign frequencies, f_r and f_h, were limited to between 12 and 48 rpm and between 60 and 180 bpm, respectively. These frequencies were initialised to random values between these ranges and were set to vary with time randomly to mimic the non-stationary nature of frequency in real vital sign signals. The gross motion was constructed with a large triangle wave as a base with a IQ offset so that the wave fits between 1 m and 4 m. A triangle wave was chosen to mimic the displacement of a person walking slowly back and forth at a constant velocity. The triangle wave was subjected to large scale, low-frequency noise to break its uniformity to better mimic the randomness present in an experiment and to represent other possible sources of motion such as the body rocking or the legs swinging.

A compensation signal was derived from the signal representing gross motion and an added error. The error was formed using a standard normal distribution. The resulting random error containing white Gaussian noise was put through a low pass filter to make it band-limited and multiplied by a scalar to adjust the size of the errors present. Therefore, the error was controlled by two parameters: magnitude (controlled by scalar multiplier) and bandwidth (controlled by the filter cut-off frequency). This error represented statistical error present in any measurement but also represented any motions not captured by the compensation signal that could corrupt vital signs.

5.1.5 Vital Signs with Motion

The most complex and final layer tested included gross motion introduced by walking. The participant walked backwards and forwards in front of the radar in a straight line in the centre of the radar's field of view. The participant did not turn around at the apexes of the motion and had their hands behind their back during the walking cycles to reduce the complexity of the motion. A similar grouping and categorisation of experiments as shown in Table 7 was used, except that the participant was walking. For the sake of procedural efficiency, recordings of 2 minutes were taken and split into 4 30 s experiments. The participant timed their breathing to the GUI to provide ground truth for breathing and heart rate truth data was provided by the Polar H10 monitor. Unfortunately, due to the extra accelerations

Figure 58: Setup of walking experiments for heart rate and breathing rate measurements.

imposed on the IMU the Polar H10 could not be used for ground truth for respiratory rate (breathing).

5.2 Performance Metrics

5.2.1 Percentage Accuracy

Percentage Accuracy, P_{Acc}, is defined as shown in Equation 27 where X_e is the expected (or true) value and X_p is the predicted (or measured) value.

$$P_{\text{Acc}} = 100\Big(1 - \frac{X_e - X_p}{X_e}\Big)\% \tag{27}$$

However, this metric does have its flaws. It suffers when the range of possible values is very small compared to the size of X_e and when there is an imbalance in sample sizes. As an example if nine out of ten vital experiments have heart rate measurements in range 60 to 80 beats per minute and a single experiment with a heart rate of over 150 bpm but the radar only ever determines the rate to be within the 60 to 80 bpm range, then the accuracy of the radar will close to 90% even though it got one of the measurements horribly wrong. With this distribution of data, it is difficult to say that the system is accurate because the data is imbalanced. If more data with higher heart rates were included the accuracy would lower to more a reasonable value. Therefore, to use the accuracy metric successfully, the data is

required to have a full and balanced set of possible values. On the other hand, if the range of possible values is very small, then the accuracy metric will only return good results and so it is not a good metric for characterising the performance of systems with small output ranges. An example of this could be respiratory rate that occupies a very narrow portion of the frequency spectrum.

5.2.2 Root Mean Square Error (RMSE)

The other common metric used is Root Mean Square Error (RMS) error or Root Mean Square Error (RMSE) and is calculated as shown in Equation 28. RMSE is a good metric due too its sensitivity to outliers. The square term in Equation 28 heavily penalises large errors.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}||d_i - f_i||^2} \tag{28}$$

5.2.3 Normalised Root Mean Square Error (N-RMSE)

RMSE is a great metric but it does suffer from a similar issue as percentage accuracy. It does not take the possible range of values into account. As an example, if a vital sign signal is filtered to 0.2 Hz to 0.6 Hz for breathing and 1 Hz to 4 Hz for heart rate, the range of possible values for heart rate is significantly larger. Therefore, a RMSE error of 5 bpm is very impressive for heart rate but terrible for breathing rate. Therefore normalising the RMSE to the range by dividing by the measurement range and quoting the output as a percentage gives a more standardised way of measuring between the two signals. In the above example the Normalised - Root Mean Square (N-RMSE) for heart rate would 2.8% while the N-RMSE for the respiratory rate would be 20%. This clearly shows that the breathing rate measurement is terrible compared to the heart rate measurement in this scenario because the percentage of the range the N-RMSE error occupies is so much larger.

6 Results and Discussion

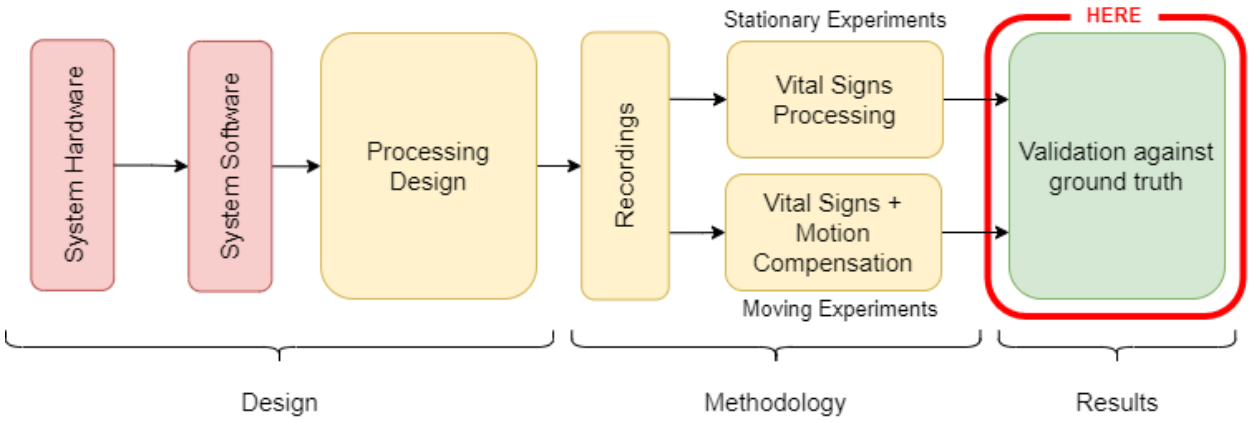

Figure 59: Figure of dissertation layout: Results

The main body of work of this dissertation ends with the results that follow. They validate the system and pipeline developed work for vital sign measurement. The results begin with testing the phase pipeline within a highly controlled environment using a test rig with a large RCS. This is done to confirm that the phase measurement and tracking pipeline works and that the underlying principle used to perform motion compensation in this dissertation is feasible. Referring back to Figure 8, the underlying principle is that the compensation algorithm has access to two identical signals except that one signal has an added micro motion. Subtraction of the two signals should yield only the micro motion.

This section then goes on to attempt real vital sign measurements on a stationary participant. These measurements are split into two categories: breathing and not-breathing with the former being the more complex case due the respiratory harmonics present [3]. After the vital sign measurements were completed for the stationary case, simulation work was done to highlight any difficulties with motion compensation for a moving participant. Finally, this dissertation attempted to perform vital sign detection using motion compensation for the most complex case: a walking participant.

6.1 Controlled Test Rig Experiment Results

Controlled experiments with a rig were performed to test both the system designed and pipeline developed against accurately measured movement data. This was an excellent way to validate the radar system and pipeline because the rig was made of metal and represented a huge target for the radar with very high RCS and the stepper motor in the rig very accurately controlled and recorded the target's movement. Because the rig motion was created using a stepper motor and belt-drive, the rig produced a large amount of mechanical vibration. In the zoomed in section of the top plot in Figure 60, the mechanical vibrations of the rig can be seen very clearly in the phase measurement. This demonstrates the radar's capability to measure displacements in the millimetre to sub-millimetre range.

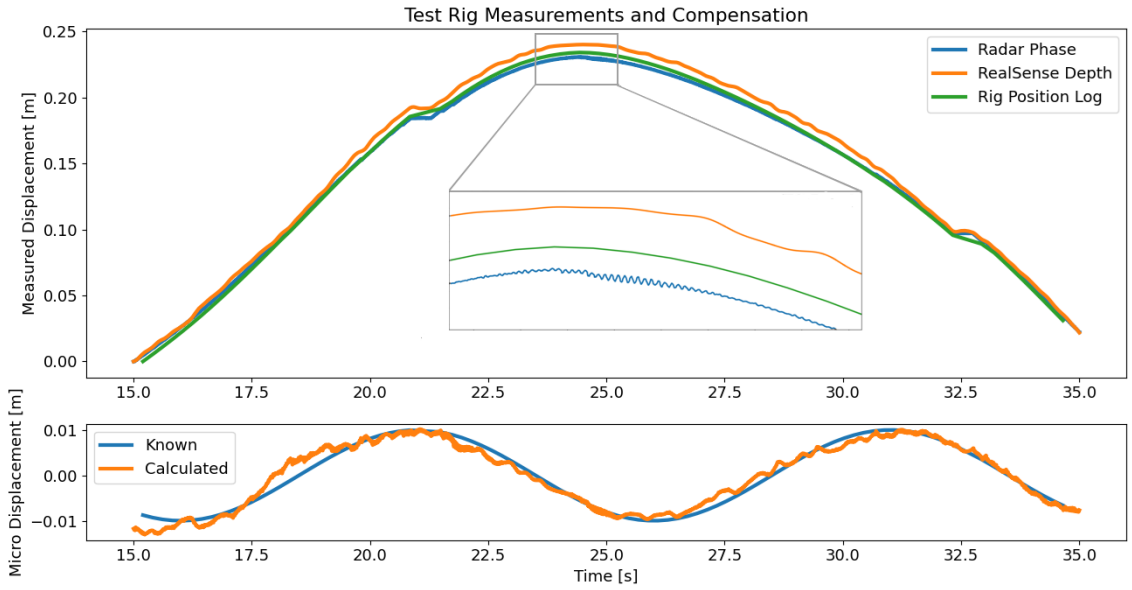

Figure 60: Figure of measured test rig displacements. **Top**: Comparison of displacements of the rig measured by the radar, rig and RealSense camera. This plot contains a zoomed in section of the measured displacements that shows the radar's capability to measure micro-motions over the other sensors. **Bottom:** Comparison of the micro-motion extracted after compensation and the ground truth.

The test rig moved towards a position of 25 cm and back over a course of 20 s. The rig contained three sets of motions. Two controlled motions: one large sinusoidal arc with a frequency at 0.01 Hz and height of 25 cm and a micro motion sinusoid with a height of 1 cm and frequency of 0.1 Hz and one uncontrollable motion: the mechanical vibrations due to the stepper motor. The rig position was measured with 3 instruments. The test rig's position logger, the radar and the stereo camera. As can be seen in Figure 60, all the measurements aligned nearly perfectly. Ideally the stereo camera was only supposed to pick up the larger motion. However, the compensation signal derived from the RealSense camera actually contained the known signal at 0.1 Hz with an amplitude of 1 cm because a 1 cm was too large and fell within the camera's depth resolution. Using a narrow-band band-stop filter to remove the known micro motion yields the correct gross motion from the RealSense data to allow for compensation. Obviously this is not possible for real applications because the frequency of the micro-motion would not be known. However, the motions we are trying to detect are typically much less than 1 cm, putting them outside the stereo camera's resolution.

The resulting micro-motion show in the bottom of Figure 60 matches the expected motion perfectly with extra noise introduced by the stereo camera. The noise introduced by the stereo camera destroys the high frequency vibrations seen in the phase measurement. This shows that the stereo camera needs to be filtered to only contain frequencies outside the vital sign band that is being measured as the stereo camera contains noise that has an amplitude higher than what is being measured. This does limit the sensor as a compensation tool because it can only compensate for motions that lie outside the vital sign frequency band.

This experiment shows that the phase is measured accurately in a controlled environment and

that motion compensation with external sensors is possible although it does have nuances and constraints that could make it difficult to do. Essentially, this kind of motion compensation will work provided you can accurately measure the gross motion without measuring the motion that one is trying to detect.

6.2 Stationary Heart Rate

As stated in Section 5.1, complexity of real data would be built up over several series of experiments. The simplest case for heart beat detection is when there is no other movement, including respiratory motion. To achieve this, the participant was asked to hold their breath for a 30 s recording period while remaining as still as possible. The plots shown in Figure 61 and Table 8 display the results of those experiments.

6.2.1 Results

Three methods, were used to evaluate the performance of the radar during these sets of experiments including: spectrograms, zero-crossings and peak-counting (by hand). As per the methodology, 16 data-sets were processed, split between resting and exerted heart rates. Three metrics were used where possible to describe the performance including: percentage accuracy, (RMSE) and (N-RMSE). Taking the average of each of these metrics over all experiments produced the statistics in Table 8. Because the peak counting was done by hand, over the entire experiment duration, the heart rate obtained is effectively only one measurement. Therefore, the RMSE is just the error in the measurement which is why it is not stated in Table 8.

6.2.2 Discussion

The raw phase shown in the top left plot of Figure 61 shows a clear periodic signal with some low frequency fluctuation imposed on top of it. This low frequency drift could be a result of ambiguous unwraps during the reconstruction process, but is so small it could just as likely be subtle motions made by the participant. An excellent example of this is shown in Figure 62 which shows an experiment where the participant had to hold their breath while at a very high heart rate. Due to the level of exertion, holding their breath for the full 30 s was impossible. As a result, the recording shows periods of heavy breathing which are clearly visible and periods of held breath where heart beats can clearly be seen as shown by the zoomed in section. The top plot shows the phase reconstruction on its own. Only looking at that reconstruction shows what could easily be an ambiguous unwrap between 20 s and 30 s. However, when compared to the RealSense camera data, it is seen that the phase reconstruction is perfectly accurate.

The experiment shown in Figure 62 is a great experiment that highlights several issues. First, the RealSense camera contains a lot of noise where the participant held their breath compared

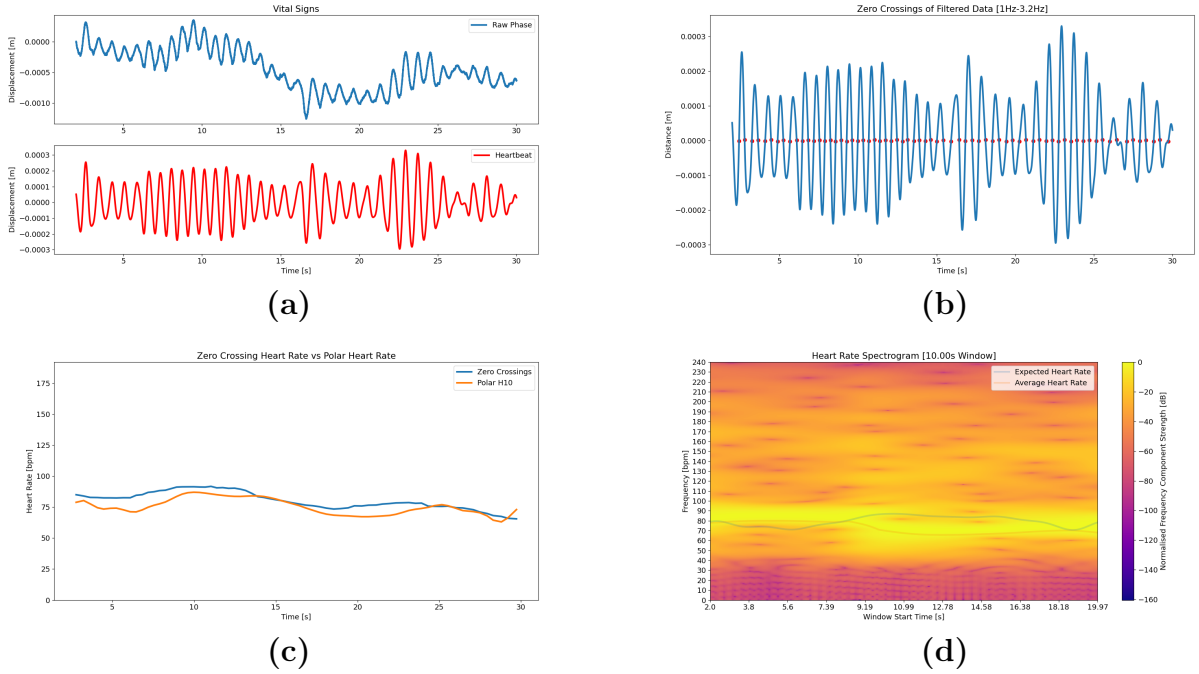

Figure 61: Results for a 30 s experiment where the participant sat still while holding their breath. **(a)**: The raw phase before and after filtering. **(b)**: Selection of points from the filtered used for zero crossing analysis. **(c)**: The comparison of the heart rate determined by the zero crossings and the polar heart rate monitor. **(d)**: Spectrogram of the filtered phase.

Table 8: Statistical results of heart rate only measurements for three different methods of analysis (10 second windows of 300 samples).

Heart Rate Only Results			
Zero Crossings			
Performance Metric	Resting (50-100 bpm)	Exerted (125-175 bpm)	Overall
Accuracy [%]	87.1 ± 6.01	81.5 ± 4.78	84.5 ± 6.15
RMSE [bpm]	11.3 ± 5.24	30.9 ± 9.21	20.4 ± 12.26
N-RMSE [%]	8.54 ± 3.97	23.4 ± 6.98	15.5 ± 9.29
FFT Spectrogram			
Performance Metric	Resting (50-100 bpm)	Exerted (125-175 bpm)	Overall
Accuracy [%]	89.9 ± 3.37	81.8 ± 7.34	85.7 ± 7.24
RMSE [bpm]	9.19 ± 3.42	34.8 ± 12.43	21.9 ± 15.48
N-RMSE [%]	6.97 ± 2.59	26.4 ± 9.42	16.6 ± 12.11
Peak Counting			
Performance Metric	Resting (50-100 bpm)	Exerted (125-175 bpm)	Overall
Accuracy [%]	96.2 ± 2.63	95.8 ± 3.17	96.0 ± 2.90

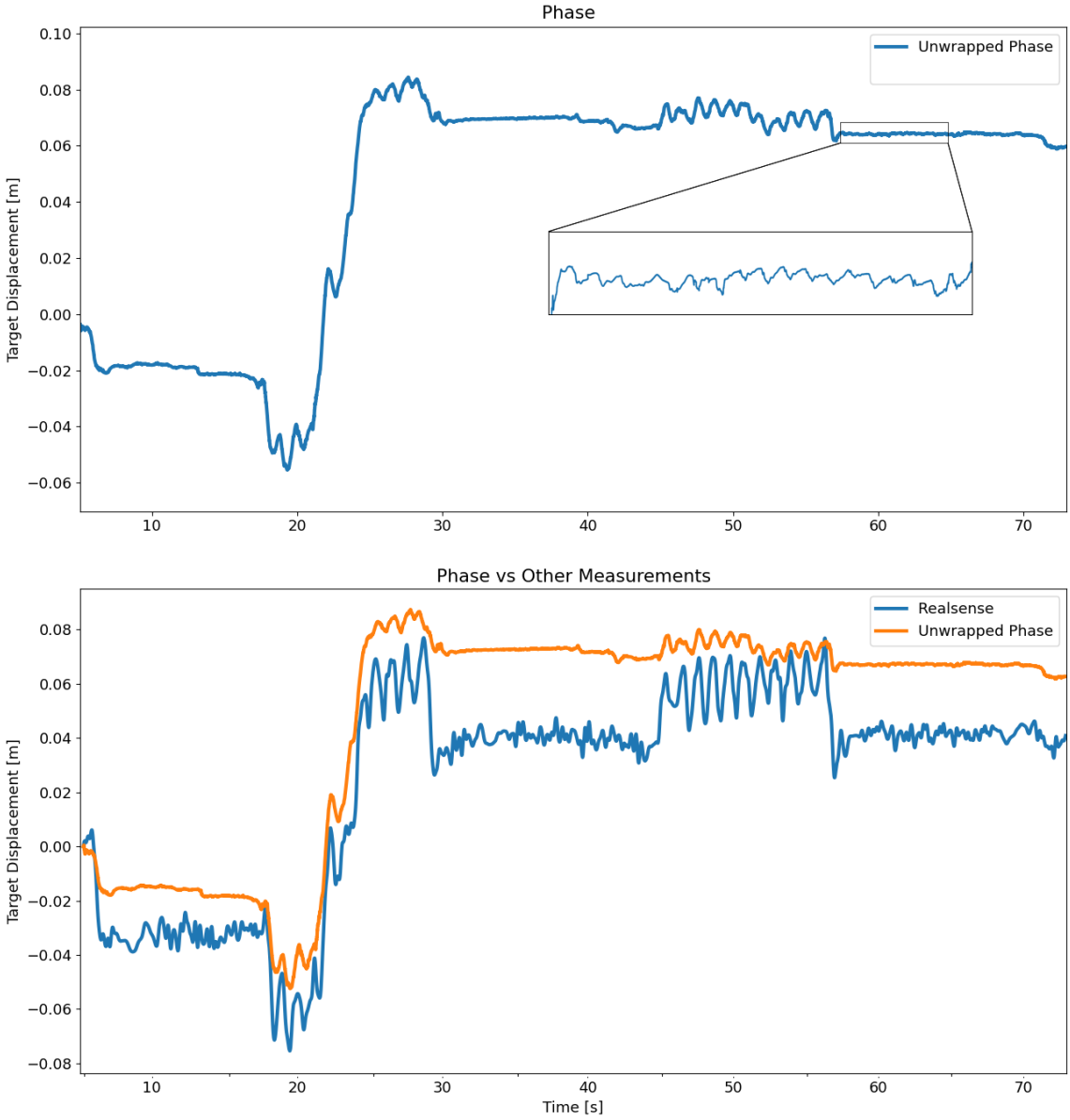

Figure 62: Experiment where participant was asked to hold their breath while at a very high heart rate. **Top**: Raw phase. **Bottom:** Comparison of raw phase and RealSense data.

to the phase signal which is very clean. Therefore, trying to do any motion compensation in the heart rate band with the RealSense will ultimately cause severe corruption of the cardiac signal.

Second, the RealSense data contains the breathing motion which ideally means that upon doing compensation in the time domain, the breathing would be removed which is something that may or may not be desirable depending on the situation. It might be a useful tool to remove breathing artefacts from heart rate measurements but it might also remove breathing rate components from breathing measurements. However, both of these facts are moot considering the amplitude of the signals do not match. There is a clear correlation between the signals but the mismatch will cause corruption if they are subtracted from each other in the time domain.

6.3 Stationary Heart and Breathing Rate

To build on the work done in the previous set of experiments, chest wall motions due to respiration were added. Again multiple sets of experiments were conducted as per the methodology to generate the statistics shown in Table 9 and the results shown in Figure 63.

6.3.1 Results

The same three methods as in the previous section were used for evaluating the performance of the pipeline. However, because of difficulties maintaining breathing and heart rate during exerted experiments (where the participants heart rate was pushed as high possible), only three sets of data were able to be collected at unspecified breathing rates. So the pool of data for exerted breathing experiments was smaller.

Table 9: Statistical results of stationary breathing (10 second windows of 300 samples).

Breathing Rate Results			
FFT Spectrogram			
Performance Metric	Resting (50-100 bpm)	Exerted (125-175 bpm)	Overall
Accuracy [%]	93.4 ± 2.34	93.4 ± 0.86	93.4 ± 2.04
RMSE [rpm]	1.78 ± 1.00	3 ± 1.18	2.1 ± 1.19
N-RMSE [%]	3.70 ± 2.08	6.26 ± 2.46	4.4 ± 2.47

6.3.2 Discussion

Comparing the results shown in Table 10 to Table 8, there is a notable degradation in performance when breathing is introduced. While, respiratory rate is measured effortlessly with very high accuracy and a clean phase reconstruction that shows the expected amplitude

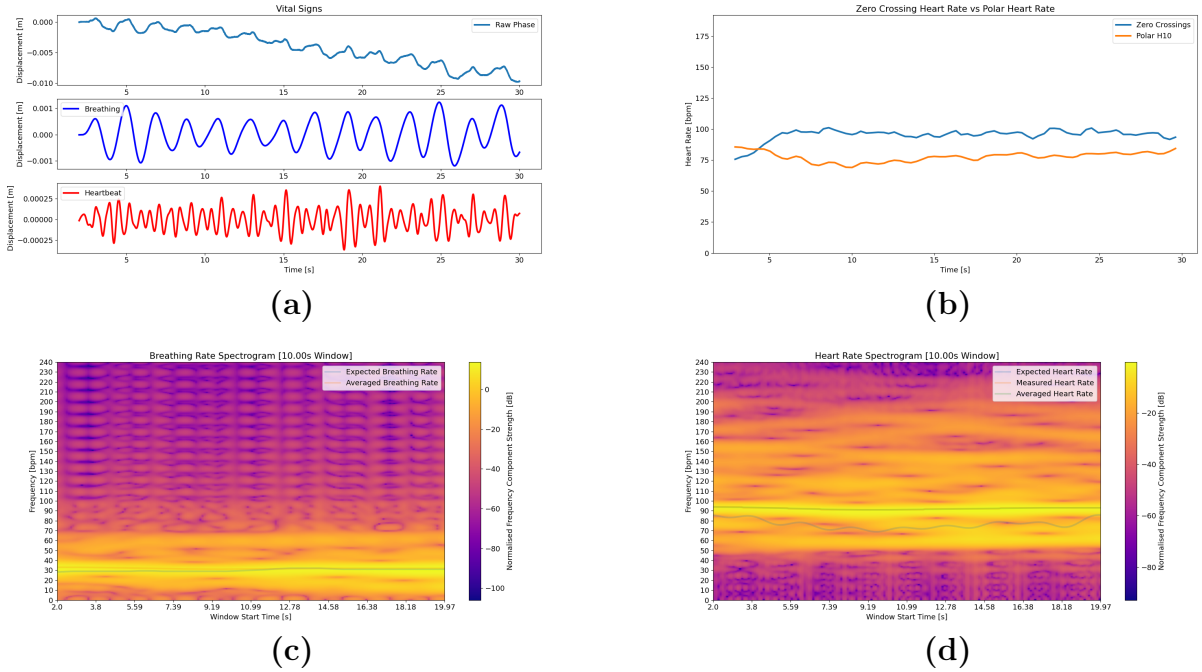

Figure 63: Results for a 30 s experiment where the participant sat still while breathing at 30 breaths per minute. **a**: The raw phase before and after filtering. **b**: The comparison of the heart rate determined by the zero crossings and the polar heart rate monitor. **c**: Spectrogram of the breathing rate band between 0.2 Hz and 1 Hz. **d**: Spectrogram of the heart rate band between 1 Hz and 3.2 Hz.

Table 10: Statistical results of stationary vital signs methods (10 second windows of 300 samples).

Heart Rate Results While Breathing			
Zero Crossings			
Performance Metric	Resting (50-100 bpm)	Exerted (125-175 bpm)	Overall
Accuracy [%]	74.4 ± 7.15	60.3 ± 9.37	70.5 ± 10.0
RMSE [bpm]	21.5 ± 4.40	64.4 ± 17.4	33.2 ± 21.5
N-RMSE [%]	16.3 ± 3.33	48.7 ± 13.2	25.1 ± 16.2
FFT Spectrogram			
Performance Metric	Resting (50-100 bpm)	Exerted (125-175 bpm)	Overall
Accuracy [%]	84.1 ± 6.81	48.6 ± 8.91	74.4 ± 17.5
RMSE [bpm]	11.1 ± 4.40	84.5 ± 14.34	31.1 ± 33.75
N-RMSE [%]	9.98 ± 3.33	64.0 ± 10.9	24.7 ± 24.89
Peak Counting			
Performance Metric	Resting (50-100 bpm)	Exerted (125-175 bpm)	Overall
Accuracy [%]	94.5 ± 2.69	75.6 ± 12.8	90.6 ± 11.62

(2 mm) as seen in Table 9 and Figure 63, the heart rate measurements have suffered. Looking at Figure 64, you can see the effects that breathing has on heart rate measurements. Spectral components of breathing can spill over into the heart rate band and hide the heart rate. These additional components, separate from the fundamental frequency, are present because breathing is not perfectly sinusoidal [8, 27]. There are distortions and other irregularities that introduce power into different parts of the spectrum. These can be seen in the filter output for breathing in the top left plot of Figure 63. The effects of these distortions can be seen in the time domain by the higher amplitude signal that dominates the filter output (shown in the bottom plot of Figure 64 while the participant is breathing compared to the expected amplitude of the heart rate while the participant holds their breath. If breathing were perfectly sinusoidal with no other spectral components, the effects of breathing on the heart rate band would not be visible in Figure 64.

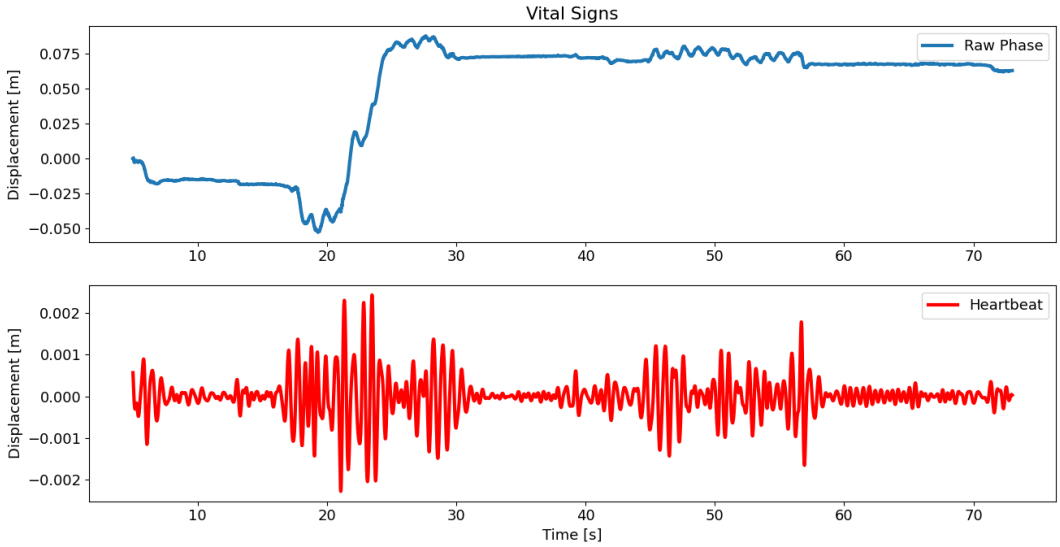

Figure 64: Experiment where participant was asked to hold their breath while at a very high heart rate. **Top**: Raw phase. **Bottom:** Raw phase after filtering for cardiac activity.

However, degradation of the quality of the heart rate measurement is particularly present in experiments of the exerted heart rate category. The measurements for resting heart rate are considerably better with a N-RMSE of approximately 16% compared to that of the exerted N-RMSE of almost 49%. An important question to be asked is: why does the heart rate measurement suffer so much in the exerted case compared to that of the resting heart? Additionally, because the pipeline outputted similar values for both the resting and exerted case, does this mean that the pipeline just coincidentally generated a heart rate that was correct? This does not seem to be the case because the amplitude measured for the heart rate in the resting case is in the correct range as shown in Figure 63. However, the amplitude for the exerted case is outside the expected range. This makes it more likely that another kind of motion is present in the exerted case and that is what is being detected by the pipeline. With such a high heart rate (150 bpm-170 bpm, as measured by the polar heart rate monitor), it is very likely that the body's response to exhaustion caused additional micro

movements that could not be controlled by the participant.

The experiment in Figure 64 also brings to light an additional problem. Literature commonly says that the breathing band is between 0.2 Hz and 0.6 Hz. However, the respiratory motions seen Figure 64 have a frequency greater than 0.6 Hz which falls outside these typical ranges used for filtering for breathing. The correct breathing rate was only obtained when the filter bounds were increased to 1.3 Hz. Therefore, constant filter bounds are a potential limitation. The incorporation of other measurements, respiratory and cardiac models, and even machine learning could be used to inform the selection of adaptive filter bounds.

6.4 Simulation Results

6.4.1 Ideal Compensation

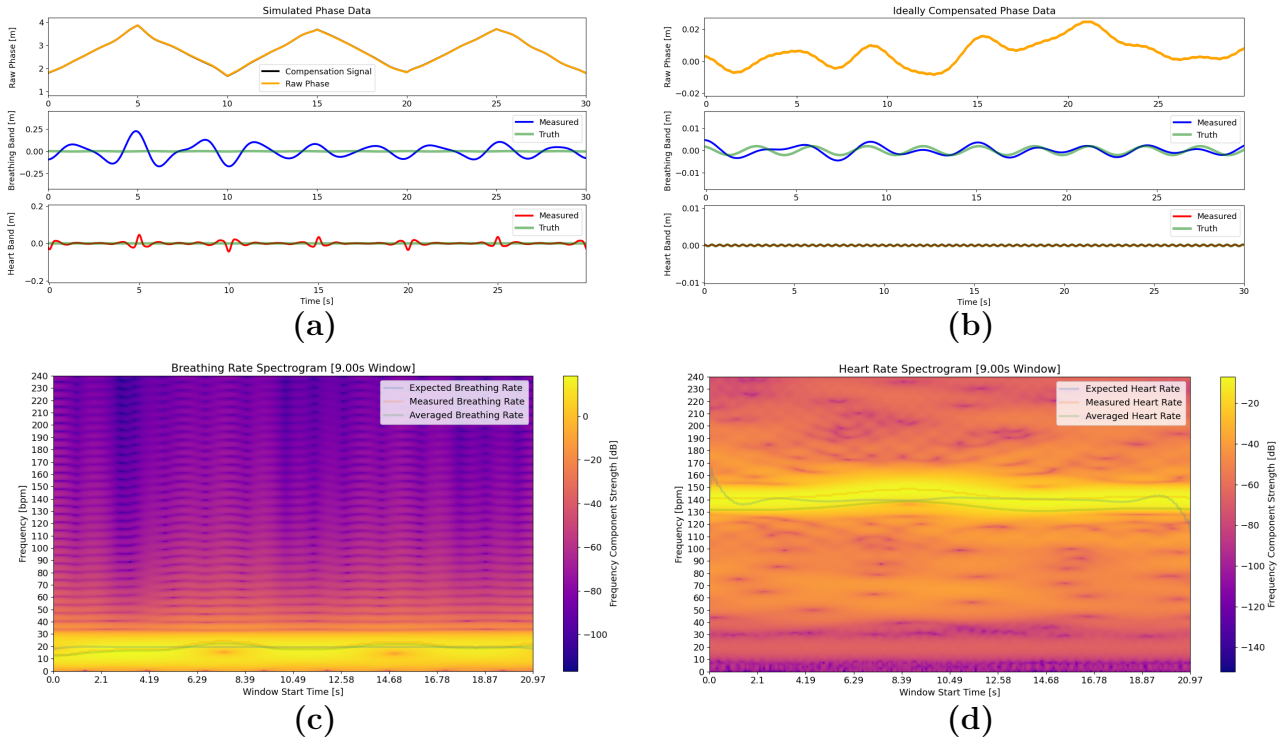

Figure 65: Example figure of an output of simulation for the ideal compensation case. **a**: Simulated data and results of cardiac and respiratory filtering without prior compensation. **Tb**: Compensated simulated data and results of cardiac and respiratory filtering after ideal compensation. **c**: Spectrogram of the respiratory frequency band after ideal compensation. **d**: Spectrogram of the cardiac frequency band after ideal compensation.

Before, exploring more complex cases, the ideal scenario for moving vital signs was simulated. The results of this simulation are shown in Figure 65. The spectrograms shown in the bottom two plots show a very strong correlation between the measured and expected frequencies present for heart rate and breathing rate and the frequencies are clearly visible. In this case, the bandwidth of the error present in the compensation signal was limited to frequencies outside the heart-rate and breathing-rate frequency bands. As can be seen in the the orange

plot in the top-right set of plots in Figure 65, there is a large amount of low frequency noise after compensation but this is filtered out and the breathing and heart rate components are resolved.

This highlights that errors, such as gradual phase drift, which are present in the lower frequency bands are ultimately negligible because they are filtered out.

6.4.2 Non-Ideal Compensation

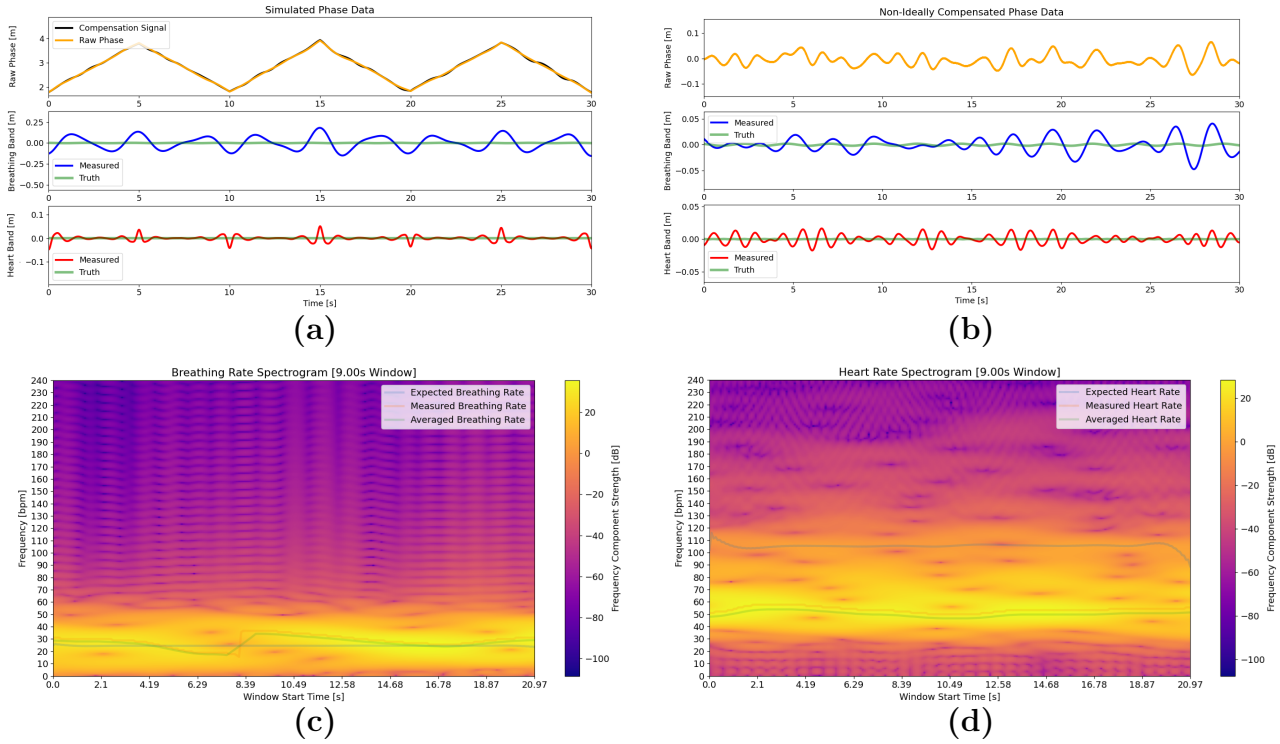

Figure 66: Example figure of an output of simulation for the non-ideal compensation case. **a**: Simulated data and results of cardiac and respiratory filtering without prior compensation. **b**: Compensated simulated data and results of cardiac and respiratory filtering after non-ideal compensation. **c**: Spectrogram of the respiratory frequency band after non-ideal compensation. **d**: Spectrogram of the cardiac frequency band after non-ideal compensation.

Following the successful compensation in the close-to-ideal case discussed previously, more complex cases were examined. The bandwidth of the error was increased to include the entire vital sign spectrum up to 4 Hz. It was found that even small amounts of error were enough to completely corrupt vital sign signals. This is because the vital sign signals are so small in magnitude. Breathing rate was much more robust to noise than heart rate because the simulated signal for respiratory activity was an order of magnitude bigger than the cardiac activity signal.

However, some simulated experiments showed a high accuracy for both breathing and heart rate after spectrogram analysis. This was co-incidental accuracy where the frequencies present in the error happened to coincide with the correct breathing and heart rate. This

highlighted a pit-fall of only looking at the frequency domain. By examining the time domain signals, it could be seen clearly that the inputs into the spectrogram were orders of magnitude larger than the ground truth vital sign signals generated by the simulation. Therefore, it was found to be important to examine both domains when performing analysis.

6.5 Moving Human Heart and Breathing Rate

Finally, exploring the most complex case (within the scope of the thesis), gross motion was added. Multiple sets of experiments were conducted as in the methodology to generate the statistics shown in Table 11, Table 12 and the results shown in Figure 67.

6.5.1 Results

The same metrics: Accuracy, RMSE and N-RMSE were used in this section as in the prior sections. However, only the FFT Spectrogram method was used and evaluated.

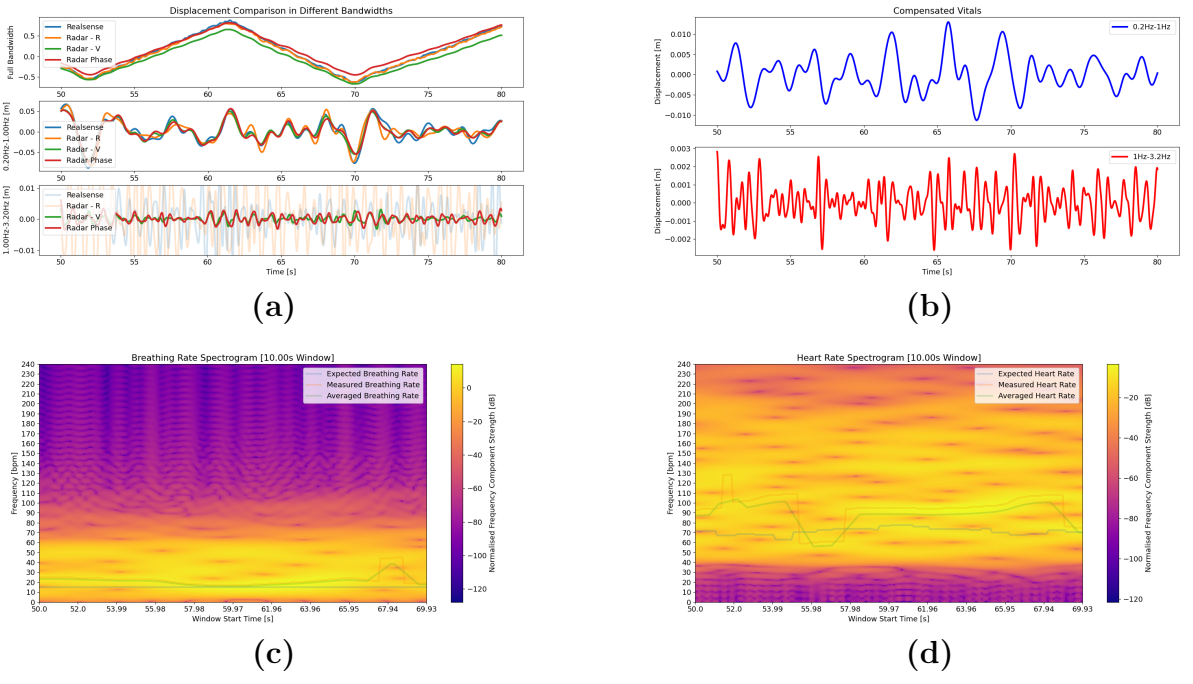

Figure 67: Example results for a 30 s experiment where the participant walked breathing at 15 rpm while at a resting heart rate. **a**: The raw phase, RealSense, radar range (Radar-R) and radar integrated velocity (Radar-V) before and after filtering. **b**: The resulting waveforms after doing compensation with the radar velocity (Radar-V) signal. **c**: Spectrogram of the breathing rate band between 0.2 Hz and 1 Hz after compensation. **d**: Spectrogram of the heart rate band between 1 Hz and 3.2 Hz after compensation.

6.5.2 Discussion

The top right set of figures in Figure 67 represent the components of different measurements in different bandwidths. The top plot of the three represents the signals at their full band-

Table 11: Statistical results of respiratory rate measurements while walking.

Breathing Rate Results While Walking			
FFT Spectrogram			
Performance Metric	Resting (50-100 bpm)	Exerted (125-175 bpm)	Overall
Accuracy [%]	61.9 ± 9.97	65.4 ± 8.15	63.8 ± 9.95
RMSE [rpm]	10.5 ± 3.09	9.4 ± 3.18	9.9 ± 3.36
N-RMSE [%]	21.9 ± 6.44	19.6 ± 6.62	20.7 ± 6.99

Table 12: Statistical results of heart rate measurements while walking

Heart Rate Results While Walking			
FFT Spectrogram			
Performance Metric	Resting (50-100 bpm)	Exerted (125-175 bpm)	Overall
Accuracy [%]	83.8 ± 5.69	75.7 ± 18.9	65 ± 18.53
RMSE [bpm]	14.5 ± 4.94	57.3 ± 14.3	35.9 ± 25.6
N-RMSE [%]	11.0 ± 3.74	48.9 ± 6.83	65.0 ± 18.5

width while the following two shows the results of filtering each signal to the respiratory and cardiac bandwidths. Radar-R and Radar-V stand for Range-Range and Radar-Velocity. Radar-R is the target depth as determined by the tracking pipeline and Radar-Velocity is the velocity of the target as determined by the tracking pipeline but integrated over time to put it into the corrected units. All the signals are very similar and agree with each other in terms of the gross motion detected and the contents of the breathing band. However, the measurements from the RealSense and radar range are extremely noisy and have amplitudes orders of magnitudes bigger than the phase and radar velocity measurements. Therefore, they cannot used to compensate for motion artefacts in the heart rate band. The strong correlation between the all the signals as shown in the top left plots of Figure 67, show that the measurements are at least relatively accurate. There is some amount of mismatch between the signals. They do not all measure the exact same displacement.

It was hoped that the mismatch between the signals would be the vital sign signal for each respective band. However, as can be seen by the top right set of plots in Figure 67, the amplitudes of the compensated vital signs are significantly higher than what is expected. This means that there is some other motion that is being measured by the radar phase reconstruction that is overshadowing the vital signs. This is further confirmed by the spectrograms which contain a large amount of background noise to the point where no clear signal can be seen. The lack of any clear fundamental frequency is particularly evident when you compare the results shown in Figure 67 to the earlier results in Figure 63 from the breathing and heart rate experiments.

The statistics shown in Table 11 and Table 12 aptly show the degrading effect of motion on the vital sign measurements even after compensation. The errors alone easily take more than half of the possible error range and the standard deviation between experiments is significantly higher. This wild variation in results indicates that any amount of accuracy calculated from the current pipeline is purely co-incidental.

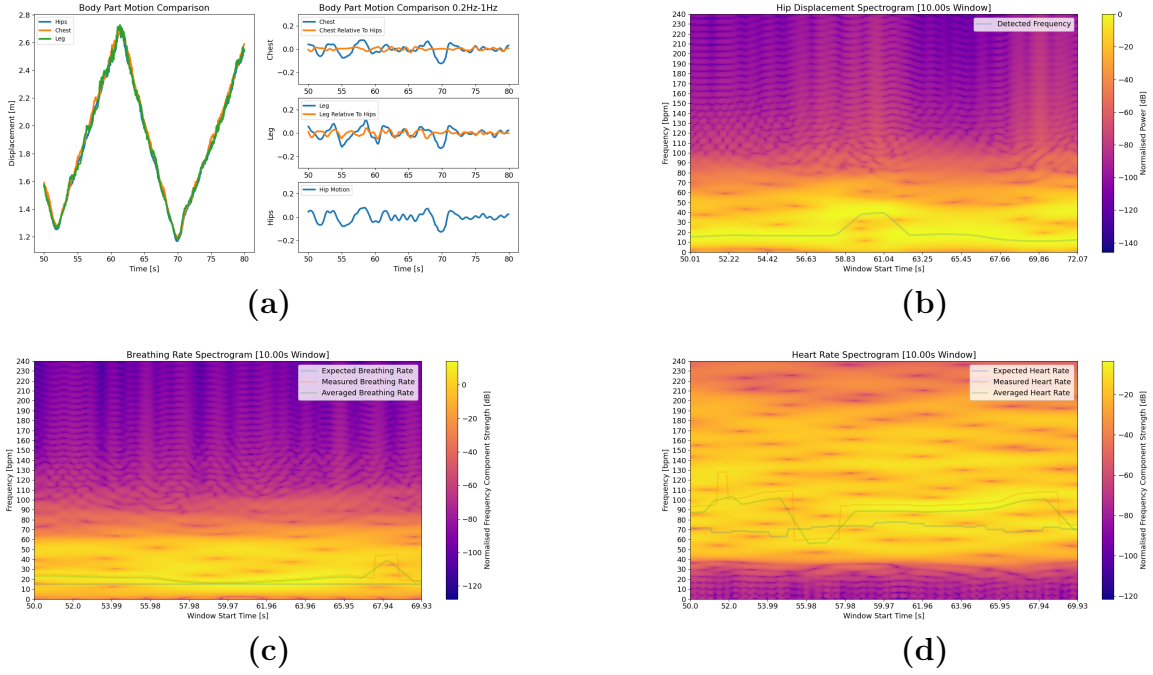

Figure 68: Example results for a 30 s experiment where the participant walked breathing at 15 rpm while at a resting heart rate. **a**: The raw phase, RealSense, radar range (Radar-R) and radar integrated velocity (Radar-V) before and after filtering. **b**: The resulting waveforms after doing compensation with the radar velocity (Radar-V) signal. **c**: Spectrogram of the breathing rate band between 0.2 Hz and 1 Hz after compensation. **d**: Spectrogram of the heart rate band between 1 Hz and 3.2 Hz after compensation.

However, the question needs to asked what is causing the extra motions measured. Thinking of the chest as a simple cube subject to the micro motions associated with vital signs and then translated linearly towards and away from the radar without any kind rotation or random oscillatory motion is too simplistic. Motions such as walking contain myriads of compound and complex three dimensional movements that could all be measured by the phase. These could include the swing associated with each leg, the rocking and rotation of the torso introduced by the gait. All of these motions are potential contributors to extra spectral components that could present themselves in the frequency domain and over-shadow vital sign measurements. Using the RealSense to measure the displacement of the hips, torso and legs, the contributions each body part can be approximated. The arms were held behind the participant's back and so were treated as part of the torso. The results are shown in Figure 68 where the time domain plot shows the overall motions side-by-side on the left and the motions of different body parts filtered to the breathing band on the right. These motions serve as the inputs to the spectrograms. The two bottom spectrograms show the frequency content of the torso and leg relative to the hips (the hips were treated as the body's reference frame). To achieve this, the measured hip displacement was removed from the measured torso and

leg displacement. Both the torso and the leg swing have a frequency around 0.3 Hz. This is similar to the incorrect frequency calculated from the breathing measurement shown in Figure 67. The hip displacement spectrogram shown in Figure 68 probably represents the rhythm of the participant's gait. A paper by Tahmoush *et al.* [53] uses radar to classify the gait of walking humans. This clearly suggests the components of one's gait, such as how the legs swing, how the body rocks to provide balance and how the hips rhythmically accelerate and decelerate during the walk cycle, are measurable and present within the radar signal. This is seen in the data presented in Figure 68. A strong 0.25 Hz signal is seen here. While the participant was breathing at that frequency, this cannot be taken as the correct rate from this signal as the amplitude is well outside the expected range. This indicates that motions captured from the body's gait and legs lie in the same frequency band as respiratory vital signs and therefore cannot be filtered by traditional means. They need to be separately measured and accounted for.

Performing a spectrogram on the phase data shown in the top left plot of Figure 67, filtered to the breathing band without any compensation, yields the same result as the hip spectrogram shown in Figure 68. Compensating the phase data causes it to share the frequency shown in the body rock and leg swing spectrograms in Figure 68. Therefore, the extra motions detected even after compensation must be due to some combination of the body rock and leg swing.

7 Conclusion

The final section of this dissertation has a summary of the result and discusses the overall successes and failures within the dissertation before going on to describe possible avenues of future work to correct those failures. Ultimately, the dissertation was a success with a flexible well designed system and pipeline implemented capable of vital sign detection while also able to be used for other applications. While motion compensation was not entirely successful for walking participants, it was achieved for highly controlled cases.

7.1 Summary of Results

Table 13: Table of requirements that have been attempted and/or completed.

Validated Requirements			
No.	**Requirement**	**Completed**	**Validation**
R1	ROS2 Integration	Yes	Data is correctly published
R2	Raw Data Collection and Storage	Yes	Data was processed and outputs appeared correct
R3	Basic Radar Tracking Pipeline	Yes	Validated against a test rig
R4	Phase Unwrapping Algorithm	Yes	Validated against a test rig
R5	Basic Vital Sign Measurements	Yes	Validated on stationary human with ground truth measurements
R6	Constrained Motion Compensation with Test Rig	Yes	The test rig showed that compensation was possible for a simple constrained case with no compound motion.
R7	Unconstrained Motion Compensation with a human participant	No	Validated on moving human with ground truth. Results showed errors caused by added complexity in walking motions.

From the system design perspective there were many successes over the course of the project as shown in Table 13. Typically, for raw data capture using the desired TI radar system, third party software that was only available on Windows had to be used. The work done on this project circumvents that requirement and allows the radar to be used on any Operating System (OS) capable of running ROS2. Storage and collection of the data was completed and validated.

From the signal processing perspective, a complete pipeline was created for post-processing the data that employed basic radar and DSP techniques to generate meaningful informa-

tion such as target motion and basic vital sign measurements of stationary targets using spectrogram analysis of radar phase data.

Table 14: Table of average statistics of heart rate measurements across experiments.

Final Statistical Results – Heart Rate			
Metrics	**Heart Rate Only**	**Breathing and Heart rate**	**Moving Vital Signs**
Accuracy [%]	88.8 ± 5.4	78.5 ± 13.0	65.0 ± 18.5
RMSE [bpm]	21.2 ± 13.9	32.2 ± 27.6	47.4 ± 33.8
Normalised RMSE [%]	16.0 ± 10.7	24.9 ± 20.6	35.9 ± 25.6

Table 15: Table of average statistics of respiratory rate measurements across experiments.

Final Statistical Results – Respiratory Rate			
Metrics	**Heart Rate Only**	**Breathing and Heart rate**	**Moving Vital Signs**
Accuracy [%]	NA	93.4 ± 2.04	63.8 ± 9.95
RMSE [bpm]	NA	2.10 ± 1.19	9.9 ± 3.36
Normalised RMSE [%]	NA	4.40 ± 2.47	20.7 ± 6.99

However, vital sign estimation on moving targets still remains a challenge as seen by the results shown in Table 14 and 15 for heart rate and respiratory rate respectively. While simple stationary measurements show good accuracy (particularly for stationary breathing), the performance of the vital sign pipeline degraded significantly for experiments including some kind of gross motion (walking). Both heart rate and respiratory rate measurements had poor accuracy around 65%. The range normalised RMSE was over 20% for respiratory measurements and over 35% for heart rate which is an enormous error. Additionally, the standard deviation for the metrics (included by the plus-minus for each measurement) is extremely high for the heart rate measurements showing that the performance of the pipeline varies wildly between experiments. This highlights the co-incidental accuracy mentioned where some experiments show better metrics just because the heart rate happens to agree with the ground truth data.

Vital sign monitoring for moving targets failed because first principle compensation by subtraction is simple in theory but it has been shown in the results that there are numerous additional complexities including error in measurements, mismatches between measurement devices. Radar measurements can receive echoes from many places on the body at once, so it becomes difficult to account for all of them by hand even if they are actually measured by the RealSense and radar. Measuring these displacements by hand is tedious and also probably prone to some errors. The results in Figure 62 highlights the possibility for mismatches between what the phase measures and what other sensors measure. In short, doing compensation by first principle subtraction is challenging, prone to error and ultimately leads to spurious results. There needs to be much better signal conditioning before the subtraction

or another method needs to be used. This leads into the discussion of the future work that should follow this project.

7.2 Future Work

7.2.1 Radar Software Design

First, while the radar software worked effectively there are some improvements to be made. The radar software should have a better GUI to handle configuration of the radar, evaluation of the parameters and an efficient quick look functionality to ensure the radar is recording clean data.

Additionally, the radar data collection node could be improved to handle more data throughput from the radar to reduce the limitations placed on using high sample counts, high chirp counts or high frame rates.

Finally, the radar configuration storage and how the radar configuration is sent to the ROS2 server can be made to be much more efficient and flexible. Having the configuration stored as a JSON or YAML for easy editing is perfect but the configuration parameters should also be published so the eCAL recorder stores them in the same HDF5 file instead of adding the config file to the finished HDF5 in post. This will ensure that incorrect config files are not introduced in the conversion process stipulated in Section 3.2.9.

7.2.2 Processing Pipeline Design

There are number of features that need to be added to the radar pipeline and a number of tasks that should be completed before the recording of data. Before data is recorded for future work, experiments need to be done to determine the best parameters for vital sign measurements of moving targets. Evaluating the performance of parameters on different layers of motion as done in thesis is a good starting point. Introducing some close to real-time construction of the phase signal used could be an easy way to conduct these experiments. This could be implemented on the CPU, but could benefit from using GPU based functions to speed up large matrix operations for systems with a dedicated NVIDIA GPU. The CUDA software library available from NVIDIA would be required [54]. NVIDIA would be preferred due to the usage of CUDA as well as the NVIDIA Jetson platforms embedded platforms commercially available.

Finally, there are a number of avenues to explore in improving the phase reconstruction of the chest wall movement, the compensation for motion and the analysis. For the improvement of the phase measurement, beam forming could be used to exclude areas of the body that contain irrelevant motion that corrupts the vital sign signal [55]. Furthermore, beam forming and stricter range bin selection could be used to select parts of the body where no breathing is present. This could make it easier to extract heart rate with no corruption from breathing

artefacts as breathing is localised to one place whereas heart beats oscillations are ubiquitous throughout the body as shown in the paper by Zhao *et al.* [10] where they measure heart rate with the radar at the ankle.

In terms of the post-processing pipeline many researchers use more advanced techniques and approaches such as ML, Adaptive Filters, Sensor Fusion and more recent analysis techniques such Empirical Mode Decomposition (EMD), Variable Mode Decomposition (VMD), Improved Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (ICEEMDAN) and the Hilbert-Huang transform [11, 56]. The use of adaptive filters could be explored. Such filters could try use the expected amplitude of the signal to find frequencies that produce such amplitudes or could use prior information derived from other sensors to evaluate the likelihood of vital signs being in certain frequency ranges over others. For instance, a camera could tell if a person or animal is breathing heavily and adjust the frequency of the vital sign filters accordingly.

For ML, in cases where ground truth data is not available for heart rate and respiratory rate, perhaps a better approach is to train the machine learning network to find the gross motion rather than find vital signs and then remove the gross motion using the motion compensation technique discussed in this dissertation or to train the network on simulated data. As can be seen by comparing the results shown in Figure 66 and 67 for simulated and gross motion experiments, simulated data can fairly accurately reflect real life data. More advanced test rigs can also be created to try create simulated data to train networks on. These test rigs could be made to simulate vital sign data with speakers or even mechanically using some kind of cam gear for breathing. Exploratory research into how complex motions affect vital sign monitoring could also be done with such a test rig. I believe this would be beneficial as the exact effect of the complex motion of different body has not been well quantified in this research.

References

[1] L. Reddy Cenkeramaddi, J. Bhatia, A. Jha, S. Kumar Vishkarma, and J. Soumya, "A survey on sensors for autonomous systems," in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, pp. 1182–1187. [Online]. Available: https://ieeexplore.ieee.org/document/9248282/

[2] A. Trange, "FMCW mm-wave radar for detection of pulse, breathing and fall within home care," Master's thesis, KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science, 2021.

[3] M. Alizadeh, G. Shaker, J. C. M. D. Almeida, P. P. Morita, and S. Safavi-Naeini, "Remote monitoring of human vital signs using mm-wave FMCW radar," *IEEE Access*, vol. 7, pp. 54 958–54 968. [Online]. Available: https://ieeexplore.ieee.org/document/8695699/

[4] A.-J. Jang, I.-S. Lee, and J.-R. Yang, "Vital signal detection using multi-radar for reductions in body movement effects," *Sensors*, vol. 21, no. 21, p. 7398. [Online]. Available: https://www.mdpi.com/1424-8220/21/21/7398

[5] Y.-K. Yoo and H.-C. Shin, "Movement compensated driver's respiratory rate extraction," *Applied Sciences*, vol. 12, no. 5, p. 2695. [Online]. Available: https://www.mdpi.com/2076-3417/12/5/2695

[6] F. Wang, X. Zeng, C. Wu, B. Wang, and K. J. R. Liu, "Driver vital signs monitoring using millimeter wave radio," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 283–11 298. [Online]. Available: https://ieeexplore.ieee.org/document/9615374/

[7] Z.-K. Yang, H. Shi, S. Zhao, and X.-D. Huang, "Vital sign detection during large-scale and fast body movements based on an adaptive noise cancellation algorithm using a single doppler radar sensor," *Sensors*, vol. 20, no. 15, p. 4183. [Online]. Available: https://www.mdpi.com/1424-8220/20/15/4183

[8] B. Zhang, B. Jiang, R. Zheng, X. Zhang, J. Li, and Q. Xu, "Pi-ViMo: Physiology-inspired robust vital sign monitoring using mmwave radars." [Online]. Available: http://arxiv.org/abs/2303.13816

[9] K. Tswenga, "Low-cost non-contact monitoring of small motions in the human body," 2022.

[10] P. Zhao, C. X. Lu, B. Wang, C. Chen, L. Xie, M. Wang, N. Trigoni, and A. Markham, "Heart rate sensing with a robot mounted mmwave radar," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2812–2818. [Online]. Available: https://ieeexplore.ieee.org/document/9197437/

[11] Y. Hu and T. Toda, "Remote vital signs measurement of indoor walking persons using mm-wave FMCW radar," *IEEE Access*, vol. 10, pp. 78 219–78 230. [Online]. Available: https://ieeexplore.ieee.org/document/9839588/

[12] M. Kebe, R. Gadhafi, B. Mohammad, M. Sanduleanu, H. Saleh, and M. Al-Qutayri, "Human vital signs detection methods and potential using radars: A review," *Sensors*, vol. 20, no. 5, p. 1454. [Online]. Available: https://www.mdpi.com/1424-8220/20/5/1454

[13] J. H. Medicine, "Vital signs (Body Temperature, Pulse Rate, Respiration Rate, Blood Pressure)." [Online]. Available: https://www.hopkinsmedicine.org/health/conditions-and-diseases/vital-signs-body-temperature-pulse-rate-respiration-rate-blood-pressure

[14] K. PPE, "Digital finger pulse oximeter." [Online]. Available: https://kingppe.co.za/product/digital-finger-pulse-oximeter/

[15] H. Rehabilitation and Health, "Can you rely on your smartwatch to monitor your heart rate?" [Online]. Available: https://www.hunterrehab.com.au/can-you-rely-on-your-smartwatch-to-monitor-your-heart-rate/

[16] T. Tanantong, "A study on the effects of window size on electrocardiogram signal quality classification."

[17] X. Yang, Z. Zhang, X. Li, Y. Zheng, and Y. Shen, "Remote radar-camera vital sign monitoring system using a graph-based extraction algorithm," in *2021 46th International Conference on Infrared, Millimeter and Terahertz Waves (IRMMW-THz)*. IEEE, pp. 1–2. [Online]. Available: https://ieeexplore.ieee.org/document/9566987/

[18] M. Marcato, J. Kenny, R. O'Riordan, C. O'Mahony, B. O'Flynn, and P. Galvin, "Assistance dog selection and performance assessment methods using behavioural and physiological tools and devices," *Applied Animal Behaviour Science*, vol. 254, p. 105691. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168159122001496

[19] P. Vermunicht, K. Makayed, P. Meysman, K. Laukens, L. Knaepen, Y. Vervoort, E. De Bliek, W. Hens, E. Van Craenenbroeck, L. Desteghe, and H. Heidbuchel, "Validation of polar H10 chest strap and fitbit inspire 2 tracker for measuring continuous heart rate in cardiac patients: impact of artefact removal algorithm," *Europace*, vol. 25, p. euad122.550. [Online]. Available: https://academic.oup.com/europace/article/doi/10.1093/europace/euad122.550/7176633

[20] B. R. Upadhyay, A. B. Baral, and M. Torlak, "Vital sign detection via angular and range measurements with mmWave MIMO radars: Algorithms and trials," *IEEE Access*, vol. 10, pp. 106 017–106 032. [Online]. Available: https://ieeexplore.ieee.org/document/9908541/

[21] W. Lv, W. He, X. Lin, and J. Miao, "Non-contact monitoring of human vital signs using FMCW millimeter wave radar in the 120 GHz band," *Sensors*, vol. 21, no. 8, p. 2732. [Online]. Available: https://www.mdpi.com/1424-8220/21/8/2732

[22] D. Wang, S. Yoo, and S. H. Cho, "Experimental comparison of IR-UWB radar and FMCW radar for vital signs," *Sensors*, vol. 20, no. 22, p. 6695. [Online]. Available: https://www.mdpi.com/1424-8220/20/22/6695

[23] A. Lazaro, D. Girbau, and R. Villarino, "Analysis of vital signs monitoring using an IR-UWB radar," *Progress In Electromagnetics Research*, vol. 100, pp. 265–284. [Online]. Available: http://www.jpier.org/PIER/pier.php?paper=09120302

[24] T. Instruments, "TI mmwave labs vital signs measurement," https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/1023/vitalSigns_5F00_lab_5F00_user_5F00_guide_5F00_v1.2UPDATE.pdf, 2017.

[25] S. Iyer, L. Zhao, M. P. Mohan, J. Jimeno, M. Y. Siyal, A. Alphones, and M. F. Karim, "mm-Wave radar-based vital signs monitoring and arrhythmia detection using machine learning," *Sensors*, vol. 22, no. 9, p. 3106. [Online]. Available: https://www.mdpi.com/1424-8220/22/9/3106

[26] J. Tu and J. Lin, "Respiration harmonics cancellation for accurate heart rate measurement in non-contact vital sign detection," in *2013 IEEE MTT-S International Microwave Symposium Digest (MTT)*. IEEE, pp. 1–3. [Online]. Available: http://ieeexplore.ieee.org/document/6697732/

[27] R. F. Fouladi and A. Oncu, "Vital signs modeling for doppler radar cardiorespiratory monitoring," in *2013 36th International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, pp. 363–366. [Online]. Available: http://ieeexplore.ieee.org/document/6613953/

[28] M. Xiang, W. Ren, W. Li, Z. Xue, and X. Jiang, "High-precision vital signs monitoring method using a FMCW millimeter-wave sensor," *Sensors*, vol. 22, no. 19, p. 7543. [Online]. Available: https://www.mdpi.com/1424-8220/22/19/7543

[29] A. Ahmad, J. C. Roh, D. Wang, and A. Dubey, "Vital signs monitoring of multiple people using a FMCW millimeter-wave sensor," in *2018 IEEE Radar Conference (RadarConf18)*. IEEE, pp. 1450–1455. [Online]. Available: https://ieeexplore.ieee.org/document/8378778/

[30] Y. Wang, Z. Wang, J. A. Zhang, H. Zhang, and M. Xu, "Vital sign monitoring in dynamic environment via mmWave radar and camera fusion." [Online]. Available: http://arxiv.org/abs/2304.11057

[31] S. Fu, M. Ling, Z. Li, and L. Pan, "A new method for vital sign detection using fmcw radar based on random body motion cancellation," *Biomedical Engineering / Biomedizinische Technik*, vol. 68, no. 6, pp. 617–632, 2023. [Online]. Available: https://doi.org/10.1515/bmt-2023-0068

[32] A. Vally, N. Bowden, L. Mbatha, G. Maswoswere, S. Paine, P. Amayo, A. Markham, and A. Patel, "M2S2: A multi-modal sensor system for remote animal motion capture in the wild."

[33] K. Brennan, "Don't hold your breath – breathing analysis with the polar H10 heart rate monitor," https://github.com/kbre93/dont-hold-your-breath, 2023.

[34] T. Instruments, "DCA1000EVM data capture card user guide," https://www.ti.com/lit/ug/spruij4a/spruij4a.pdf?ts=1695777509115, 2019.

[35] ——, "mm-wave SDK user guide," https://dr-download.ti.com/software-development/software-development-kit-sdk/MD-PIrUeCYr3X/03.06.00.00-LTS/mmwave_sdk_user_guide.pdf, 2021.

[36] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[37] O. Robotics, "ROS2 - about different middleware vendors," https://docs.ros.org/en/galactic/Concepts/About-Different-Middleware-Vendors.html, 2023, accessed: 04-10-2023.

[38] C. Corporation, "eCAL ROS middleware wrapper (eCAL RMW)," https://github.com/eclipse-ecal/rmw_ecal, 2023, accessed:04-10-2023.

[39] "eCAL - enhanced communication abstraction layer," ://eclipse-ecal.github.io/ecal/index.html.

[40] T. H. Group, "Hierarchical data format version 5," https://www.hdfgroup.org/wp-content/uploads/2017/12/HDF512-17.pdf, 2023, accessed:04-10-2023.

[41] L. A. Wasser, "Hierarchical data formats - what is HDF5?" https://www.neonscience.org/resources/learning-hub/tutorials/about-hdf5, 2020, accessed:04-10-2023.

[42] V. Dham, "Programming chirp parameters in TI radar devices," https://www.ti.com/lit/an/swra553a/swra553a.pdf?ts=1695778781352&ref_url=https%253A%252F%252Fwww.ti.com.%252Fsitesearch%252Fen-us%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US%2526searchTerm%253DProgramming%2BChip%2BParameters%2Bin%2BTI%2BRadar%2BDevicesDB%252FOL%2526nr%253D5.

[43] C. Iovescu and S. Rao, "The fundamentals of millimeter wave radar sensors," https://www.ti.com/lit/wp/spyy005a/spyy005a.pdf?ts=1697080925522&ref_url=https%253A%252F%252Fwww.google.com%252F.

[44] T. Instruments, "mm-Wave radar device ADC raw data capture," https://www.ti.com/lit/an/swra581b/swra581b.pdf?ts=1694696371668#:~:text=The%20DCA1000%20captured%20data%20samples,LVDS%20lane%20will%20be%20stored., 2018.

[45] S. Rapuano and f. harris, "An introduction to FFT and time domain windows," *Instrumentation Measurement Magazine, IEEE*, vol. 10, pp. 32 – 44, 01 2008.

[46] A. Gasmi, "What is the fast fourier transform?" 08 2022.

[47] T. Instruments, "MIMO radar," https://www.tij.co.jp/jp/lit/an/swra554a/swra554a.pdf?HQS=ti-null-null-ted-appn-ds-faq_sns-jp, 2018.

[48] H. A. Gonzalez, C. Liu, B. Vogginger, P. Kumaraveeran, and C. G. Mayr, "Doppler disambiguation in MIMO FMCW radars with binary phase modulation," *IET Radar, Sonar & Navigation*, vol. 15, no. 8, pp. 884–901, 2021. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/rsn2.12063

[49] M. A. Richards, *Fundamentals of Radar Signal Processing*. ¡country¿US¡/country¿: McGraw-Hill Professional, 2005. [Online]. Available: https://mhebooklibrary.com/doi/book/10.1036/0071444742

[50] V. Podlozhnyuk, "FFT-based 2D convolution," 07 2007.

[51] L. Litwin, "FIR and IIR digital filters," *Potentials, IEEE*, vol. 19, pp. 28 – 31, 11 2000.

[52] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[53] D. Tahmoush and J. Silvious, "Radar micro-doppler for long range front-view gait recognition," *IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems, BTAS*, pp. 1 – 6, 10 2009.

[54] NVIDIA, P. Vingelmann, and F. H. Fitzek, "Cuda, release: 10.2.89," 2020. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[55] S. Ahmed, J. Park, and S. H. Cho, "Effects of receiver beamforming for vital sign measurements using FMCW radar at various distances and angles," *Sensors*, vol. 22, no. 18, p. 6877. [Online]. Available: https://www.mdpi.com/1424-8220/22/18/6877

[56] Z. L. Xia, X. Huai Wang, H. B. Wei, and Y. Xu, "Detection of vital signs based on variational mode decomposition using fmcw radar," in *2021 International Conference on Microwave and Millimeter Wave Technology (ICMMT)*, 2021, pp. 1–3.