IMPLEMENTATION OF A LOW-COST PASSIVE BISTATIC RADAR

by

Joshua Leigh Sendall

Submitted in partial fulfilment of the requirements for the degree
Master of Engineering (Electronic Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

September 2016

SUMMARY

IMPLEMNTATION OF A LOW-COST PASSIVE BISTATIC RADAR

by

Joshua Leigh Sendall

Supervisor:	Prof. W.P. du Plessis
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Electronic Engineering)
Keywords:	Radar, passive radar, low-cost, adaptive filtering, clutter cancellation, direct-path interference cancellation, matched filtering, tracking filter, broadcast radio

Passive radar detects and ranges targets by receiving signals which are reflected off targets. Communication transmissions are generally used, however, theoretically any signal with a suitable ambiguity function may be used. The exploitation of an existing transmitter and the removal of emissions allow passive radars to act as a complementary sensor which is useful in environments where conventional active radar is not well suited. Such environments are in covert operations and in situations where a low cost or spectrally efficient solution is required.

Most developed passive radars employ intensive signal processing and use application specific equipment to achieve detection. The high-end processors and receiver equipment, however, detract from some of the inherent advantages in the passive radar architecture. These include the lower cost and power requirements achieved by removing transmitter hardware.

This study investigates the challenges faced when removing application-specific and high end components from the system and replacing them with low-cost alternatives. Solutions to these challenges are presented and validated by designing and evaluating a radar using these principles.

It was found that the major limitation in passive radar is the dynamic range of the receiver. While processing the signals was, and is, a significant challenge, be implemented on a low-cost, low-power embedded processor. This was achieved by asserting a few limitations to the configuration, exploiting the subsequently generated redundancy, and taking advantage of the parallelism by using general purpose graphics processing.. Even on this processor, the system was able to run in real time and able to detect targets up to 91 km (bistatic range of 195 km) from the radar.

OPSOMMING

IMPLEMENTERING VAN 'N LAEKOSTE- PASSIEWE BISTATIESE RADAR

deur

Joshua Leigh Sendall

Studieleier:	Prof. W.P. du Plessis
Departement:	Elektriese, Elektroniese en Rekenaaringenieurswese
Universiteit:	Universiteit van Pretoria
Graad:	Magister in Ingenieurswese (Elektroniese Ingenieurswese)
Sleutelwoorde:	Radar, passiewe radar, laekoste, aanpasbare filter, sluierkansellasie, direkte baan-versteuringskansellasie, aangepaste filter, volgfilter, uitsendingsradio

Passiewe radar spoor teikens op en meet die afstand na hulle toe deur seine te ontvang wat van die teikens af gereflekteer word. Uitgesaaide kommunikasieseine word meestal gebruik, maar teoreties kan enige sein met ʼn geskikte dubbelsinnigheidsfunksie gebruik word. In ʼn passiewe radar word senderhardeware en beheerde versendings nie benodig nie en daarom kan dit as ʼn aanvullende sensor gebruik word in omgewings waar konvensionele aktiewe radar nie geskik is nie. Voorbeelde van sodanige omgewings is koverte operasies en situasies waar ʼn goedkoper oplossing of effektiewer spektrumgebruik verlang word.

Meeste bestaande passiewe radars maak gebruik van intensiewe seinverwerking en benut toegewyde hardeware om opsporing moontlik te maak. Die gebruik van gesofistikeerde prosesseerders en ontvangstoerusting doen egter afbreuk aan die inherente voordele wat passiewe radars sou kon inhou, byvoorbeeld die laer koste en laer kragvereistes wat verkry word deur sendinghardeware te verwyder.

In hierdie studie is die uitdagings ondersoek wat vorendag kom wanneer toegewyde hardeware van die stelsel verwyder word en vervang word deur alternatiewe, laekoste-komponente. Oplossings vir hierdie uitdagings is gevolglik voorgestel en getoets deur ŉ radar wat van hierdie beginsel gebruik maak, te ontwerp en te evalueer. Die resultate van die praktiese radartoetse dui daarop dat die voorgestelde oplossings wel werkbaar is.

Daar is in hierdie studie bevind dat die grootse beperking van passiewe radar die dinamiese reikwydte van die ontvanger is. Seinprosessering was ŉ groot uitdaging en het ŉ paar beperkinge op die konfigurasie geplaas. Daar was egter steeds ŉ mate van prosesseringsoortolligheid wat benut kon word en parallelle prosesseringstegnieke is gebruik om die stelsel op ŉ laekoste- en laekrag- ingebedde prosesseerder te implementeer. Sodoende was dit moontlik om ŉ eenvoudiger prosesseerder suksesvol te implementeer. Die prosesseerder het die stelsel in staat gestel om intyds te werk en om teikens tot en met 91 km (193 km bistaties afstand) van die ontvanger af op te spoor.

ACKNOWLEGEMENTS

I would like to thank the following persons for their support.

My supervisor, Warren Paul du Plessis: for the guidance, support and inspiration during my studies. Also for reading and correcting all of my reports and papers, and teaching me so much.

Francois Maasdorp, Craig Tong, Christo Cloete, Rossouw van der Merwe and the CSIR: for supporting my research and "showing me the ropes".

My loving girlfriend, Melissa Reed: for being a foundation in my life.

My parents: for their love and support, which is always there unconditionally.

My Lord and saviour Christ Jesus: whom is my strength and shield, and through whom all thing are possible.

LIST OF ABBREVIATIONS

ADC	Analogue to Digital Converter
ADS-B	Automated Dependent Surveillance Broadcast
AGC	Automatic Gain Control
ALU	Arithmetic-logic Unit
ARD	Amplitude-Range-Doppler
ATC	Air Traffic Control
AWGN	Additive White Gaussian Noise
BLAS	Basic Linear Algebra Subroutines
CFAR	Constant False Alarm Rate
CGLS	Conjugate Gradient Least Squares
CISC	Complex Instruction Set Computing
CPI	Coherent Processing Interval
CPR	Compact Position Report
CPU	Central Processing Unit
CUT	Cell Under Test
CW	Continuous Wave
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DPI	Direct Path Interference
EM	Electromagnetic
FAR	False Alarm Rate
FDC	Frequency-domain Correlation
FDTC	Frequency-domain Time Correlation
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FLOP	Floating Point Operation
FM	Frequency Modulation
GAL	Gradient Adaptive Lattice

GPGPU	General Purpose Graphics Processing Unit
GPU	Graphics Processing Unit
LNA	Low-noise Amplifier
LPF	Low-pass Filter
LSB	Least Significant Bit
MAC	Multiply-Accumulate
MIMD	Multiple Instruction Multiple Data
MIMO	Multiple Input Multiple Output
MKL	Math Kernel Library
NaN	Not a Number
NLMS	Normalized Least Mean Squared
OEM	Original Equipment Manufacturer
OTS	Off-the-Shelf
PC	Personal Computer
PFA	Probability of False Alarm
PLL	Phase-locked Loop
PPS	Pulse per Second
RAM	Random Access Memory
RCS	Radar Cross-section
RF	Radio Frequency
RISC	Reduced Instruction Set Computing
RMS	Root Mean Square
RMSE	Root Mean Square Error
SDR	Software Defined Radio
SFDR	Spurious Free Dynamic Range
SIMD	Single Instruction Multiple Data
SIMT	Single Instruction Multiple Thread
SISD	Single Instruction Single Data
SLL	Sidelobe Level

SM	Streaming Multiprocessor
SNR	Signal to Noise Ratio
TDR	Target to Direct-path Ratio
TDTC	Time-domain Time Correlation

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context of the problem

Passive coherent radars have been an area of recent interest as they have the potential to meet needs which conventional active radars cannot. The primary advantage of a passive radar is that it does not transmit a signal. Instead of transmitting a signal, passive radars use transmitters of opportunity, such as television, radio, or cellular transmissions [1]. Removing the transmission of a signal provides several benefits including being difficult to detect in covert scenarios [2, 3], reduced vulnerability to anti-radiation armaments [4], requiring potentially less power (as no power needs to be transmitted), and not requiring a new frequency band allocation [2] (i.e. reduced electromagnetic pollution [3]). A practical motivation for a low-cost passive radar has been identified [5], whereby developing countries, such as those found in Africa, are unable to afford and maintain primary air traffic control (ATC) radar systems [5].

A passive radar may use any transmitter of opportunity that the system deems suitable. However, various signal properties such as bandwidth, transmission power, and ambiguity functions may impose limits and place unacceptable demands on a system, in order for that signal to be used.

The absence of transmitter hardware has the potential to reduce capital and operational costs of implementing a passive radar system [3]. However, the increased demand for processing [6] depreciates the economic benefit of removing the transmitting components from the system by requiring more expensive and power hungry processors.

Replacing conventional custom receivers [7] with low-cost off-the-shelf (OTS) digital receivers reduces the cost of the system, but sacrifices performance for cost. As such, this trade off dramatically affects the system’s ability to recover target reflections from the received signals. The main factors which contribute to the decreased target detection capability are:

- Non-coherency between receiver channels, which degrades the performance of direct-path interference (DPI) removal and the matched filter,
- decreased SFDR of the reciever,
- analogue to digital converters (ADCs) with lower bit depth, which increases the systems minimum signal required for detection [8], and
- wider analogue filters, which reduce the effective bit depth of the receiver.

These factors introduce challenges into the system by making DPI mitigation before, and after digital sampling a high priority. By understanding the performance critical factors of the receiver with regard to a passive radar, intelligent selection of components can allow for the performance/cost ratio to be maximised.

In order to remove DPI and clutter, which is required before targets can be detected, digital adaptive filters are employed [9]. The adaptive filters, which typically implement least-squares algorithms, are computationally intensive and traditionally require the majority of available processing power [6, 9]. Compounding the challenges incurred when implementing a DPI cancellation filter is the high dynamic range of the signals (with the direct path signal being in excess of 100 dB larger than typical target signals) [1].

Reducing the signal processing requirements allows low-cost processing components to be considered. While contributing to a reduction in capital cost, low-cost processors can also exhibit lower power consumption, thereby further reducing the operational costs of the system.

1.1.2 Research gap

Many passive radar systems exist, but due to the general market requiring high specification and high performance systems, they often rely on specialized and high-end hardware to realize real-time processing. However, there is little research into implementing a low-cost system which can be deployed in financially sensitive scenarios. Previous attempts have either resulted in poor performance [10], or the need for custom hardware [7] [11]. Furthermore, the reduction in cost of any system increases the resistance to competitor market entry.

The single site system here cannot unambiguously detect a target. A multi-site or multi-channel system is required for such detection. This work, therefore, focuses on a low-cost single site system which can be incorporated into a larger network of sensors.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

The research questions posed are:

1. Is real-time processing achievable using mobile non-specialized OTS hardware?

   With most passive radar systems, the objective has been performance and, as such, high-end processing hardware has been used, even to the extent of computer clusters [12]. However, in a power and cost sensitive environment, is it possible to achieve real-time processing on low-power mobile processors, and what steps/restrictions/simplifications are necessary to achieve it? This has been previously considered [13], however, it was performed for a specific cases and the underlying trends were not explored.

2. Which factors constrain the performance of a low-cost passive radar?

   As with any system, it is important to understand the performance limiting factors within the system. In systems with highly constrained resources it is even more critical. A low-cost passive radar is an example of such a system, and in order to successfully implement it, the factors which limit its performance must be understood so that the resources available can be optimally distributed, thereby maximising performance. A similar analysis has been performed focusing on the receiver [7]. The analysis in this study extends this to the rest of the system.

3. Is it possible to detect aircraft with non-specialized receivers, such as a low cost OTS SDR?

   The receiver is a key component which imposes many limitations on the system. As such, it necessary to determine what these are and, if a decrease in the receiver performance can be tolerated, still allow for adequate performance. This has been previously explored in literature [10], but with limited detection ranges displayed and exploration into the reasoning behind the limited performance.

4. What are the considerations necessary when selecting a site for a low-cost passive radar and do the priorities differ from conventional passive radar site selection?

   The reduction in performance of key system components, necessary to decrease the overall system cost, may impose additional restrictions on the system (such as vulnerability to out-of-band interference). These need to be understood and quantified.

5. Which floating-point capable processors are best suited to a low-cost passive radar system?

   Different processor architectures must be explored, not only for performance, but in order to find a solution that provides sufficient performance at a minimum cost, i.e. providing a suitable cost-performance trade-off. Some exploration into this has been performed in literature [11], but the analysis has only been done for specific configurations.

1.3 SCOPE

Due to the vastness of passive radar, the subject in its entirety is not considered in this study. The following constraints are imposed on this study to limit this scope.

Transmitter

In order to condense the topic to a manageable scope, only commercial FM radio based passive radar is considered. This implies a limit on the bandwidth of the signal of 150 kHz, and that the centre frequency lies between 88 and 108 MHz.

FM passive radar was chosen due to its suitability in the area where the study was based. This decision also allows the results of the study to be used in the study of the deployment of passive radar as a low cost radar in Africa, where analogue FM remains a vital and abundant communication medium.

Mobility

It is further assumed that the radar's transmitter and receiver are static, and thus clutter lies close to and is typically centred on a zero Doppler shift. This leaves the radar defined as a passive bistatic radar.

Direction finding

In this study the direction of targets is not considered. This constraint was applied as direction finding in passive bistatic radar still requires research and development. One solution which is currently under development uses multiple sites [14] [15] (either transmitters or receivers), i.e. a multistatic system [16]. In such a system the processing at each site is identical to that considered in this study, and the position of a target is determined by integrating and comparing the results of each site [15]. Hence a number of systems could be combined to enable unambiguous detection.

Targets

To evaluate the radar system, only aircraft with ADS-B transceivers were considered, as this provided a means of control data, which was used to verify the position of the aircraft. An obvious limitation is that the detection accuracy measurable is limited by the accuracy of the ADS-B detections. Furthermore, civilian aircraft typically have a large RCS and have significant velocity so as to allow distinction from clutter. This has been an approach in a number of studies [5,7,11-13,17]

Processors

Of the five major processor families (ASIC, FPGA, DSP, GPP, and GPU) only GPUs and GPPs, or more specifically CPUs, were considered in this study. ASIC solutions were excluded as they typically have the highest development and production cost (in the low quantities expected for this system) of the families.

The passive radar environment lends itself well to floating point processing because of the high dynamic range of components of the signals received and processed by the system. Furthermore, many of the computationally intensive operations can be heavily parallelised. The processors considered were thus selected to support floating point processing and parallel architectures.

The development cost of both DSPs and FPGAs, both in terms of time and development tool cost, typically is significantly higher than the CPU variant for two reasons.

Firstly, the drivers and operation software for the receivers considered in this study were only available for CPU based systems with full OS. The use of a platform without such an OS would require the development of driver and communication software/hardware to allow the integration of an OTS receiver. This development is significant and was beyond the scope of this study.

Secondly, while DSPs and FPGAs may provide a lower cost system when the packages themselves are considered, the requirement for their integration into a development board (or custom board) results in increased cost for a one-off unit. This was found to negate many of the benefits gained by using a DSP or FPGA processor for such low quantities. As such both FPGAs and DSPs are beyond the scope of this study.

GPUs are considered as they are paired with a CPU which allows the benefits of integrating a low-cost CPU into a system with a full OS. Furthermore, the algorithms used in performing the radar processing can exploit parallel processing architectures [5], and require high dynamic range (which can be achieved by using floating point data structures). Additionally, GPUs can allow both a high performance to power ratio [18] and a high performance to cost ratio, conforming well to the requirements of the problem.

1.4 RESEARCH CONTRIBUTION

A DPI and clutter cancellation algorithm was developed which was not based on the conventional least squares principle, but rather correlation. This was done in an attempt to reduce the processing requirements of the passive radar system, of which the adaptive cancellation filter contributes the majority. A paper was submitted for publication based on this filter architecture and the analysis conducted in Section 5.2.6.1.

1.5 OVERVIEW OF STUDY

The structure of the study and the content of each chapter is summarised below.

1.5.1 Chapter 2 Passive Radar

Chapter 2 provides an introduction to passive radar, and thus outlining how passive radar functions. In order to gain full understanding, a brief introduction to some important general radar concepts is also provided.

1.5.2 Chapter 3 Signal Acquisition

Chapter 3 discusses the non-digital design aspects. These include receiver selection, site selection and antenna selection. The factors which influence the performance of a passive radar are discussed here and strategies on how to approach these challenges are presented.

1.5.3 Chapter 4 Processing Platforms

Chapter 4 introduces the processing platforms which were investigated for the implementation of the passive radar. The differences between and structures of CPUs and

GPUs are discussed. Their general operation is also discussed as an understanding thereof is required to effectively optimise their use.

1.5.4 Chapter 5 Processing Chain

Chapter 5 discusses the digital signal processing chain used for passive radar. Algorithms and applicable optimizations are presented and evaluated. Each processing chain is discussed in terms of its effectiveness and performance on each of the processing platforms.

1.5.5 Chapter 6 Results

Chapter 6 demonstrates the functionality of a radar system designed and implemented using the principles presented in the previous chapters. The system was deployed and its output was recorded. The detection capability, accuracy, and financial implications of the implemented system are evaluated in this chapter using the recorded data.

1.5.6 Chapter 7 Conclusion and Future Work

Chapter 7 presents a summary of the study and its achievements as well as improvements which may be implemented in the future.

CHAPTER 2 PASSIVE RADAR

2.1 INTRODUCTION

Radar is defined as "an electromagnetic system for the detection and location of objects that operates by transmitting electromagnetic signals, receiving echoes from objects (targets) within its volume of coverage, and extracting location and other information from the echo signal" [19]. While this indeed describes a traditional active radar, radars have expanded beyond this meaning. At their essence, however, radars still remain systems which use electromagnetic signals reflected off objects to detect the object's presence.

A passive radar, rather than transmitting signals which then reflect off objects and are received, receives the reflections from transmissions of opportunity [9] (much like the human eye uses light originating from the sun). While various designs for passive radar exist, the scope for this research is on a separated reference architecture, as this is the simplest form of passive radar, requiring the least hardware and processing, and which aligns well with the goal of a low-cost passive radar.

2.2 BASIC PASSIVE RADAR ARCHITECTURE

In its simplest form, the separated reference architecture uses two antennas and receiver channels [5]. A passive radar does not have direct access to the transmitted signal, which is required for coherent processing. Instead, an approximation of the transmitted signal is received via one of the receiver channels, i.e. the reference channel. Ideally, only the transmitted signal is received, but this is not achievable in practical systems [20].

When the transmitter of opportunity is transmitting certain digital signals, such as DAB+, the reference channel can be extracted from the surveillance channel via the use of a channel model. In this case, a separate reference antenna is not required. However, this study does not include this case, as the signals considered useable by the system are FM analogue signals (see Section 1.3).

The purpose of the other receiver channel, the surveillance channel, is to receive the signals reflected off of objects of interest (and ideally nothing else). In practice, only a small portion of the signals reflected from targets ever reach the surveillance antenna [12]. Without careful design, these signals would fall below the SFDR of the receiver and not be detectable [8]. The reason that SFDR is generally a larger problem than receiver sensitivity, is due to the relatively large amount of energy which is received directly from the transmitter of opportunity [1]. Even with careful design extensive signal processing is required to suppress DPI and detect targets [9].

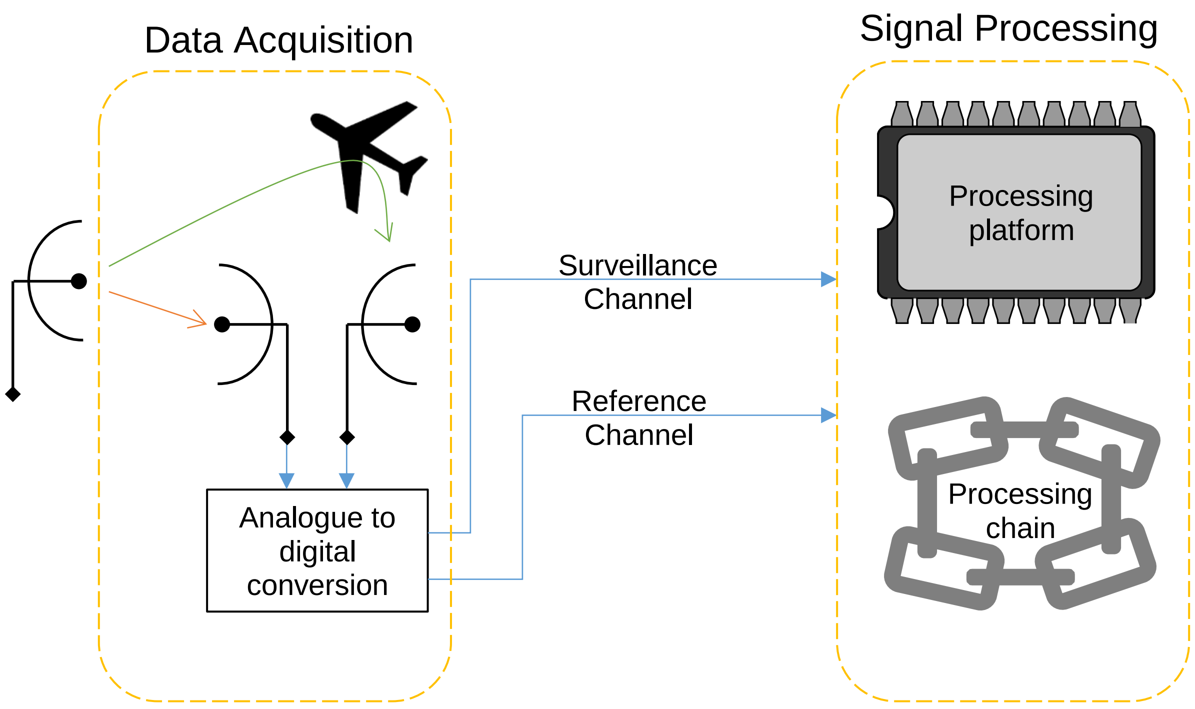


Figure 2.1 System diagram of an analogue passive radar.

A conceptual diagram of a generalised passive radar is shown in Figure 2.1. Data Acquisition concerns the process of receiving useful signals and manipulating them to the point where they can be processed. Signal processing is the process of manipulating the received data channels such that targets may be detected.

A typical processing chain is shown in Figure 2.2. In active radar, the cross-correlation and Doppler processing stage comprises the majority of processing [21], although, due to the control which active radar has over the transmitted signal, this processing stage is implemented differently to how it is performed in passive radar [12]. This description of the matched filter is expanded on in Section 2.6.1.

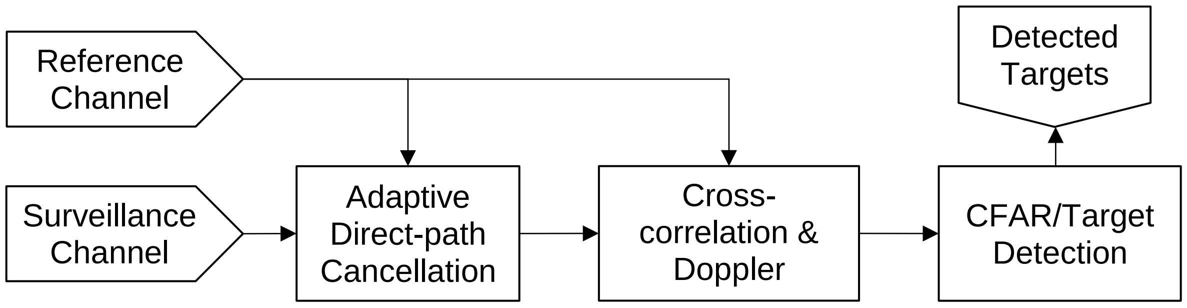


Figure 2.2 Typical passive radar processing chain.

An additional step, which active radar does not typically include, is the removal of DPI. This step is required to remove large signals the sidelobes of which would otherwise mask any targets [5]. The functionality of this processing block is covered in Section 2.6.2.

Lastly, a target detection scheme is required to automatically detect targets. This normally comprises, but is not limited to, a CFAR detector. However, due to some of the unique aspects of passive radar, the implementation of the detector differs somewhat from conventional active radar [5]. This is discussed in Section 2.6.3. Finally, the raw detections need to be associated with their respective targets. This is performed by a tracking filter.

2.3 PASSIVE RADAR IN CONTEXT

In this section passive radar is contextualised and classified within the field of radar.

2.3.1 Transmitter/receiver orientation

The vast majority of radars are monostatic [21], i.e. the distance between the transmitter and receiver is negligible and can be considered to be in the same location. Many pulse radars go further than this and use the same antenna for transmitting and receiving.

Alternatively, the transmitter and receiver can be separated creating a bistatic system. Most passive radars are bistatic systems, with the receiver in one location and the transmitter in another [22]. There are also passive radars which use multiple receiver or transmitter sites. This is termed a multistatic system [22].

2.3.2 Continuous and pulse radar

Pulsed radar operates by transmitting a short pulse, and remaining silent while listening for echoes [21], as shown in Figure 2.3. This architecture has been widely adopted as it reduces the dynamic range of the system by only receiving when the transmitter has stopped transmitting. Unfortunately, it also means that the echoes received from targets cannot be integrated over the entire CPI. Instead, each pulse can only be integrated for as long as its duration [21]. Thus, pulsed radars typically have a higher peak-to-average transmission power than continuous radar.

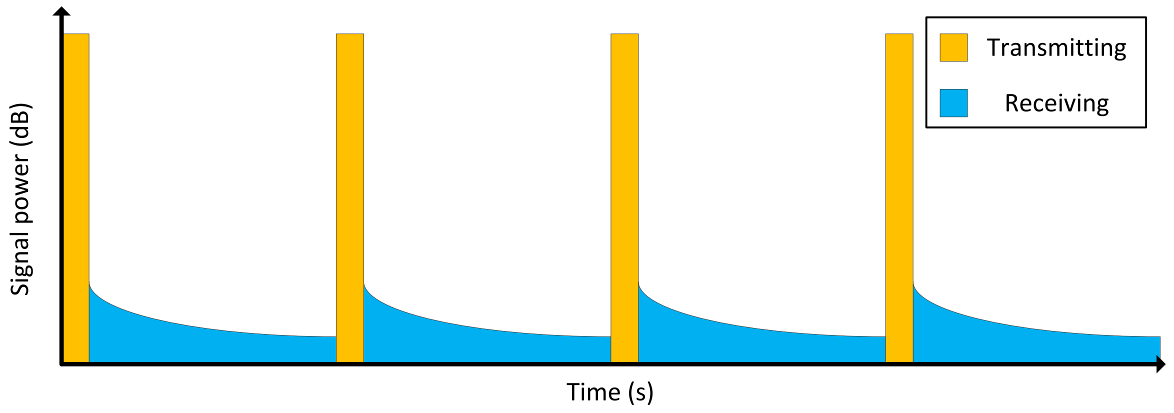


Figure 2.3 Diagram of pulsed radar signal strength.

Continuous radar transmits and receives signals simultaneously. This allows the echoes to be integrated over the entire CPI, thus allowing for reduced peak transmit power. One of the major challenges in CW radar is reducing the leak through from the transmitter to the receiver channel, which degrades the radar's performance [23].

Passive radar is a special case of bi/multistatic CW radar. DPI is a major design challenge in active CW, where the designer has control over the receiver and transmitter. In passive radar, where one cannot control the transmitter, DPI becomes a critical performance limiting factor.

2.4 HISTORY OF FM-BASED PASSIVE RADAR

The first passive radar was developed as a by-product of the development of radar. This was during the Daventry experiment in 1935 [4]. This radar detected a Heyford bomber at a distance of 8 km using a shortwave radio transmitter. Other passive radar systems were also deployed by the German armed forces, which exploited the transmissions of the British Chain Home Radars [4].

After WWII little interest remained in passive radar until the advent of high performance digital signal processing hardware. In 1986 an experiment was conducted using television broadcasts [24], which were chosen due to their resemblance to a pulsed wave form. Due to the limitations posed by the available ADC hardware, the system was unable to successfully detect targets. However, the experiment uncovered the major challenges in passive radar (such as dynamic range), and thus opened the door to future work.

A further system was developed, in 1997 that exploited television broadcasts [25]. This system was able to detect aircraft at a range of 260km (with a direct path distance of 150km). While the system was able to detect aircraft at a vastly improved range, it was only able to determine their Doppler shift and bearing.

Further investigation into passive radar [26] was conducted in the 1990’s [4]. The investigation found potential in broadcast transmitters as transmitters of opportunity, as well as an additional motivation for passive radar in that it holds potential for stealth mitigation.

FM-based passive radar was first introduced in 2005 [1]. A detailed design of the system was presented, which used SDRs to digitise the signal, and used a computing cluster to achieve real-time processing. The system was able to detect targets at up to a range of 150km, with a direct path distance of approximately 50 km. In the same year the theoretical performance of FM-based passive radar was published [27]. In this work, the radar range equation was derived in the context of passive radar.

In 2007 FM signals were analysed for their role in passive radar [28]. Here it was shown that the range resolution of the system is dependent on the standard deviation and kurtosis of the transmitted signal.

This led to an investigation into improving this limit, which in turn led to a method being developed that combined multiple FM-channels [29]. In the same work, a comparison between super heterodyne and direct receiver architectures was conducted. It found that, while comparable, the direct sampling architecture yielded performance advantages over the super-heterodyne architecture.

In 2010 an airborne passive radar demonstrator [30] was experimented. The system was able to detect high velocity airborne targets at ranges (as shown in the publication) up to 35 km. However, the signal had to be recorded and processed at a later stage.

A passive radar based on an SDR and using generic processing software was demonstrated in 2011 [10]. However, the system was only able to detect targets with a bistatic range of 63 km, with a direct path distance of approximately 47 km. The system was also unable to operate in real-time, and processing was performed on recorded signals.

A long-range FM-based passive radar was demonstrated in 2012 [31]. This was able to detect targets up to a range of 300 km, while 60 km from the transmitter. It was shown that a high dynamic range is essential for long range detection, and that a greater direct path distance lessens the required dynamic range requirement of the system.

In 2014, a generalised processing architecture for passive radar was presented [13]. Here the term “separated reference architecture” was coined, and an exploration into the use of GPU’s was investigated.

A system [11] using an SDR with a custom fixed frequency RF front-end [7], and an NVIDIA Jetson TK1 Development Board to process the data was demonstrated. The system could detect targets up to a range of approximately 100 km, with a direct path distance of approximately 100 km. The system employed a CGLS filter to perform DPI cancellation, and was just able to run in real-time.

2.5 COMPONENTS OF THE RECEIVED SIGNALS AND ANTENNA CONFIGURATION

Theoretically, signals received by a passive radar's reference channel are composed of three major components [5], shown in Figure 2.4. The largest contribution is DPI, which is the signal received directly from the transmitter [2] (the red signal in Figure 2.4). The second largest contribution (the yellow signal) typically found in received passive signal consist of reflections from large stationary objects such as buildings, mountains and hills [2]. The final component (the green signal) is comprised of reflections from moving objects which are the reflections of interest [2].

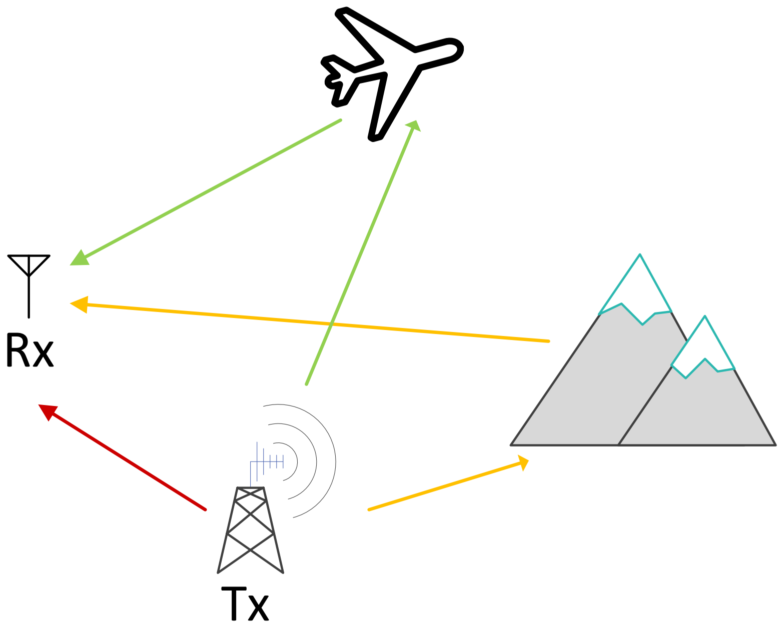


Figure 2.4 Diagram of the signal components received by a passive radar.

A certain level of DPI is required by the passive radar system. However, it is only desirable in the reference channel [2]. The DPI (in a static system) has approximately no delay and no Doppler shift and thus approximately resides at a range of 0 km with a Doppler shift of 0 Hz. Clutter is a comparatively large signal, although smaller than the DPI, and is normally characterised by a low Doppler shift, although its range (or delay) may vary. Finally, target reflections are characterised by a low signal amplitude with a significant Doppler shift generated by their higher velocities.

2.6 PASSIVE RADAR PROCESSING

In order to detect target reflections among the much larger unwanted reflections, interference and noise, the signal needs to be filtered and processed. The components of the processing chain have been widely discussed in literature [11], and these are introduced below.

2.6.1 Matched filtering

In order to increase the SINR, a matched filter is used to determine the probability of a reflection's presence. A matched filter is defined as a filter used to maximise the SNR of a target derived from the transmitted signal [21]. The mathematical description of the filter starts with the definition of the reference and surveillance channels as vectors r and s respectively. Given that a CPI consists of N samples s and r are both N element vectors.

The filter coefficients for the matched filter are given as

$$h(t) = \hat{r}^*(-t) \tag{2.1}$$

where

$\hat{r}$ is a delayed and/or modulated version of r, and

t is time.

In pulse-Doppler radar this process is carried out in two steps [21]. In the first step each pulse is compressed by applying a matched filter via correlation. Following this, the second step uses a DFT on inter pulse samples to extract frequency, i.e. Doppler information. Doppler processing is one of the most valuable tools available to the radar designer, as it allows targets of interest (which are generally moving) to be distinguished according to their radial or bistatic velocity.

In continuous wave and passive radar (which is not taking advantage of a pulsed source) there are no pulses and, as such, pulse compression and Doppler processing are combined and performed simultaneously [12]. In passive radar this can be achieved by cross-correlating the surveillance and modified reference channel in either the time [12] or frequency domain [5]. These techniques are compared in Section 5.3, but a brief explanation of the differences is given below.

The time domain method is the closest to the pulse compression technique seen in pulse radar. In this method, each Doppler index is generated by cross-correlating the surveillance channel with an appropriately modulated reference channel in the time domain.

$$y(t) = \int_{-\infty}^{\infty} s(t)\hat{r}^*(-t)\,\partial t \tag{2.2}$$

Frequency correlation is the opposite, where each range index is generated by cross-correlating the surveillance channel with an appropriately delayed version of the reference channel in the frequency domain.

Before pulse compression and, therefore, the matched filter, the range resolution of the system is related to the pulse length. In CW radar this would make the range resolution equal to the CPI. However, pulse compression decouples these two parameters and instead couples the range resolution to the bandwidth of the transmitted signal, specifically [21]

$$\Delta R = \frac{c_0}{2B} \tag{2.3}$$

where

c_0 is the speed of light in a vacuum, and

B is the instantaneous bandwidth of the transmitted signal.

This has been shown to hold true for passive radar where B is the instantaneous bandwidth and ΔR is the range resolution [32].

Target separation is also achievable in the Doppler dimension. Here the Doppler resolution is given as

$$\Delta R' = \frac{1}{CPI}. \tag{2.4}$$

This is clearly more consistent than the range resolution, as it is not dependent on the transmitted signal, which in passive radar is not controllable or predictable.

2.6.2 DPI cancellation

As a result of the matched filter, the signal space is transformed to an amplitude-range-Doppler (ARD) map, with the amplitude representing the received signal strength at a range-Doppler index. However, the sidelobes from strong signal components mask the smaller desirable reflections [1]. In order to remove the sidelobes, the contribution of the large unwanted components is estimated and subtracted from the surveillance channel, [9] [2]

$$\hat{s} = s - A\hat{x} \tag{2.5}$$

where

$\hat{s}$ is the surveillance channel after cancellation,
A is a matrix of signal components which are removed, and
$\hat{x}$ is a vector of weights estimating the contribution of each of the signal components in A.

The estimation can be performed using a linear least-squares algorithm [2], a clean technique [9], adaptive filters [12] [33], or correlative cancellation.

Studies have previously been conducted into effectiveness of various filters for the application of DPI cancellation. It has been found that Wiener-Hoph filters achieve near optimal performance [9], but can be computationally intensive. Some adaptive filters were found to have satisfactory performance [9], but incurs significant computational cost. CGLS has been proposed as an alternative to solve the least squares problem [13], with significantly fewer computations.

The various implementations are compared in Section 5.2, and an introduction to each is given in Addendum A.

2.6.3 CFAR detection

CFAR is a common and well-studied automatic thresholding scheme used to detect targets in radar systems [21]. The technique is based on a likelihood ratio hypothesis test [21].

If the interference power is assumed to be Gaussian distributed, the variance of the noise is known, and a desired probability of false alarm (P_{FA}) is given, a threshold which maximises the probability of detection (P_D) can be defined using the Neyman-Pearson bound [21]

$$T = \sqrt{-E\sigma_n^2 \ln(P_{FA})} \tag{2.6}$$

where

σ_n^2 is the variance of the interference, and

E is the energy in the matched filter coefficients.

In practice this is difficult to implement, as the interference level is often unknown and fluctuating. To compensate for this CFAR is used to estimate the interference variance. CFAR begins by defining a number of bins that are used to estimate the interference level.

Cell averaging is a simple CFAR technique and uses an average to estimate the interference. The threshold is then defined as

$$T_{CA} = \alpha_{CA}\hat{\sigma}_N^2 \tag{2.7}$$

with α_{CA} the CFAR constant defined as [21]

$$\alpha_{CA} = N\left(P_{FA}^{-1/N} - 1\right). \tag{2.8}$$

In active radar, the interference bins can be allocated in all dimensions. This is because the range, angular, and Doppler resolution remain constant. In passive radar, however, the fluctuating bandwidth [32] in signals, transmitted by the transmitter of opportunity, causes the range resolution of the radar to fluctuate. This makes the choice of the number of guard bins in the range dimension difficult to determine and inconsistent. As such, in passive radar systems, interference estimation is often constrained to the Doppler and angular dimensions. Alternatively, the range resolution needs to be estimated before the detection stage.

2.7 SUMMARY

FM-based passive radar is an extension of active continuous wave radars. It shares many of the same principles and performance characteristics. There are however, some additional challenges to face; key among these is DPI.

Passive radar has been around since the advent of radar, but FM-based passive radar only became useable in the early 2000s. Since then these systems have become cheaper and have been slowly maturing, but a successful system built on commercial OTS components has not been seen.

CHAPTER 3 SIGNAL ACQUISITION

The passive radar system designed is based on a commercial FM transmitter (88 to 108 MHz). The choice of using commercial FM transmitters was made as these transmitters typically transmit with high power, and coverage is readily present throughout Africa [34] (where this study was located).

The signal acquisition sub-system of a passive radar, as is depicted in Figure 2.1, can be broken into multiple sub-sections. The components of the signal acquisition subsystem are shown in Figure 3.1. Here there are two major divisions.

The first, site and transmitter selection (covered in Section 3.2), concerns receiving the most desirable analogue signals. The second, the receiver, is responsible for bringing the signals to baseband and digitising them.

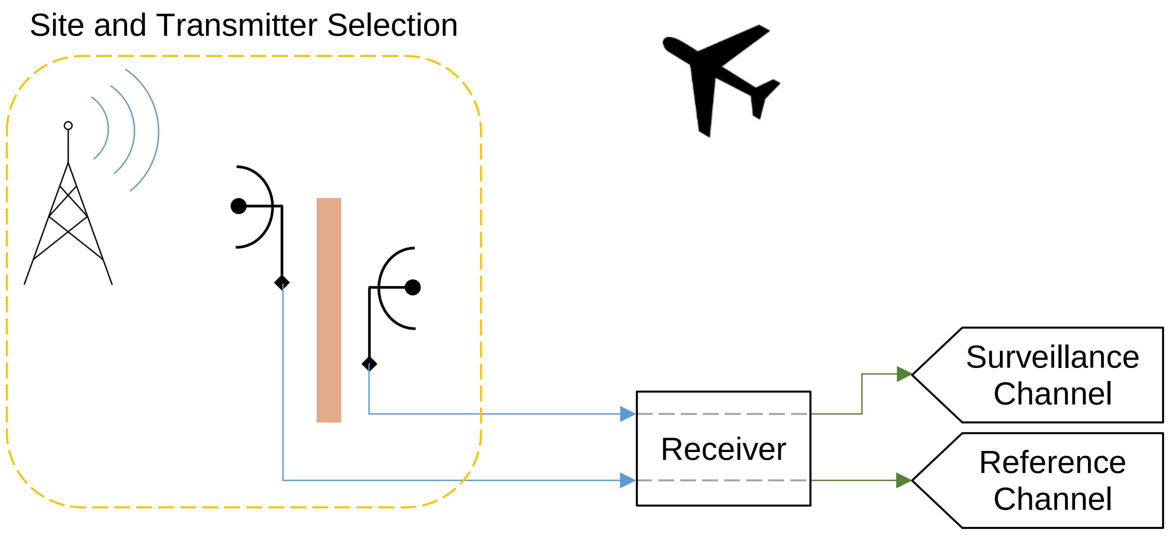


Figure 3.1 System diagram of the signal acquisition sub-system.

3.1 PERFORMANCE FACTORS

Before delving into the sub-systems of signal acquisition, it worth understanding the major parameters and factors which govern the performance of the signal acquisition sub-system.

Similar to active radar, the objective of signal acquisition in passive radar is to maximise the signals received from potential targets and to minimise the signals received from interference and noise. To this extent, the passive radar may face similar restrictions to a conventional radar.

Noise is defined as, “an unwanted disturbances superposed upon a useful signal that tend to obscure its information content” [35]. Within the system there are many sources of noise, from thermal noise within components to jamming. Most of this noise is combated by using integration gain. However, there are also coherent sources of noise which must be intentionally removed. The major performance constraints and the factors which influence these are discussed below.

3.1.1 Active interference

Active interferers are interferers which emit energy into the system’s antennas within the band of operation. These interferers can take many forms, from pirate transmitters to intentional electronic attacks. Such interference reduces the performance of the system and can even result in false targets being produced.

Active jamming can affect the system in a number of ways, depending on the knowledge the attacking system has about the radar and the type of jamming implemented. However, in the context of the experimental system, a deliberate electronic attack is a fairly remote possibility. A more likely scenario is one where another transmitter is projecting energy in the band, unintentionally interfering with the radar.

There is little literature on the subject [36]. As such, a rudimentary analysis of the effect of a pirate transmitter on a passive radar is presented here. The reference signal is modelled as

$$r(t) = pk(t) + ql(t) + n(t) \tag{3.1}$$

where

$k(t)$ is the normalised signal transmitted by the transmitter of opportunity,

$l(t)$ is the normalised signal transmitted by the interfering source (assumed to be uncorrelated),

$n(t)$ is AWGN, and

p and q are complex weights which account for the gain of the signals as received by the reference antenna.

The surveillance signal is modelled as

$$s(t) = rk(t) + sl(t) + m(t) + n(t) \tag{3.2}$$

where

$m(t)$ comprises of target echoes received by the surveillance antenna, and

r and s are complex weights which account for the gain of the signals as received by the surveillance antenna.

Therefore, in order to allow for successful DPI cancellation, by implementing (5.1),

$$\frac{p}{q} \approx \frac{r}{s} \tag{3.3}$$

must hold.

As there are likely to be spatial differences between sources, it is unlikely that this equality will hold, and, therefore, the cancellation will be impaired and targets are likely to be masked by the DPI's sidelobes. This analysis does not take into account the effect that an additional in-band transmitter will have on $l(t)$, as well as the actual representation of targets. However, given the effect an additional interferer has on the reference signal and cancellation it can already be established that the presence of an additional, in-band interferer is highly detrimental to system performance. The high vulnerability is also found in simulations [36] where a noise jammer with a power output of $1 - 10$ W (-77.10 dB to -57.10 dB compared to the transmitter power) has a dramatically detrimental effect on the radar. With a jammer transmit power of 10 W, the simulated radar was incapable of detecting the simulated target.

3.1.2 Sensitivity

Sensitivity of an electronic system can be described as the smallest signal level which can create a desired output. In a radar, sensitivity can also be termed as a minimum detectable signal [37]. In terms of the radar system, this level can be governed by a number of factors, although the level is predominantly governed by the receiver. i.e. the smallest signal which can be digitized. This level is determined by the receiver's capability to digitize a signal, as well as the efficiency with which energy is harvested and delivered to the receiver.

3.1.3 Dynamic range

The dynamic range of the system refers to the smallest signal which can be utilized compared to the largest. In terms of this specific system, dynamic range refers to the smallest signal which can be utilized compared to magnitude of the direct path signal (as the direct path signal can be assumed to be the largest signal present on the system [12].

The dynamic range of any digital system is related to the quantisation of the signal level. In a fixed point system, this level sets a minimum signal which can be represented relative to full scale. This level is referred to as the quantisation noise and is given as [38]

$$n_q = -6.02b - 1.76 \text{ dBFS} \tag{3.4}$$

where

b is the number of bits used for quantisation.

Equation (3.4) is, however, only a true representation when the signal is sampled at the Nyquist rate. Oversampling can effectively result in a higher dynamic range, or bit depth. The additional dynamic range in bits achieved by oversampling is [39]

$$b_{os} = \log_2\left(\frac{f_s}{f_N}\right) \tag{3.5}$$

where

f_s is the sampling frequency, and

f_N is the sampling frequency where $f_N < f_s$.

It should be noted that practically a few of the LSBs may not be correctly quantised by the ADC, and as such the theoretical dynamic range of the quantisation should be considered as a best-case scenario, rather than the expected value.

Aside from the dynamic range limitation imposed by digitisation, the analogue components of the receiver impose a limitation known as the SFDR. This limitation defines “the available signal range as the difference in magnitude between the amplitude of the fundamental and the amplitude of the largest spurious component in the frequency band of interest” [40]. The presence of the distortion effectively masks signals less powerful than it and thus limits the dynamic range. The dynamic range achieved is governed by the smallest of these ranges, presented above.

3.2 SITE AND TRANSMITTER SELECTION

The primary objective when selecting an appropriate site is to maximise the ratio between the received power from target and DPI (the reasoning for which is shown later in this section). Furthermore, due to relaxed filters found in lower end receivers, reducing just-out-of-band interference is also a consideration [7].

In order to find the optimal orientation/site, an estimation for the power received from a target and the DPI is required. This kind of estimation has been considered with in-depth models which take into account the terrain [41] and its effect on EM propagation [5]. However, this information was not available and, as such, a simpler model is considered where propagation and environmental effects are considered to be negligible.

The layout of the environment is shown in Figure 3.2. In Figure 3.2 the transmitter is located at Tx and the target at T. The distance between T and Tx is given as R_0. The receiver is located at Rx with the distance between Rx and T denoted by R_1 and the distance between Tx and Rx is denoted by R_2, i.e. the baseline distance. The angle between TxT and TxRx is given as γ.

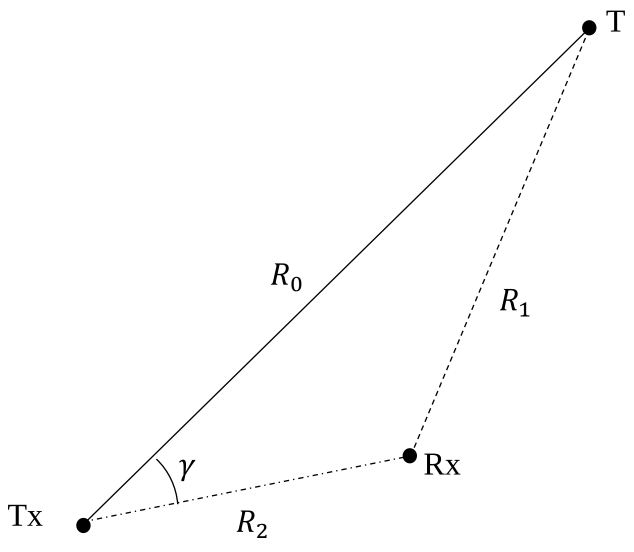


Figure 3.2 Signals received at a passive radar site.

The power received from DPI can be approximated using the Friss transmission equation [42],

$$P_r = P_t G_r G_t \left(\frac{\lambda}{4\pi R}\right)^2 \tag{3.6}$$

where

P_t is the power transmitted,

G_t is the gain of the transmission antenna in the direction of transmission,

G_r is the gain of the receive antenna in the direction of reception,

λ is the wavelength of the carrier frequency, and

R is the distance between the reception and transmission antennas.

In the context of the passive radar environment this becomes

$$P_{DPI} = P_t G_{r2} G_t \left(\frac{\lambda}{4\pi R_2}\right)^2, \tag{3.7}$$

while power received from the target can be written by the bistatic radar equation [8] [43],

$$P_r = \frac{P_t G_r G_t \lambda^2 \sigma}{(4\pi)^3 R_{Tx}^2 R_{Rx}^2} \tag{3.8}$$

where

R_{Tx} is the distance from the transmitter to the target, and

R_{Rx} is the distance from the target to the receiver.

In terms of the environment in Figure 3.2, the power received from the target becomes

$$P_T = \frac{P_t G_r 1 G_t \lambda^2 \sigma}{(4\pi)^3 R_0^2 R_1^2}. \tag{3.9}$$

Assuming that the transmission antenna is isotropic, the target to DPI ratio (TDR) becomes

$$TDR = \frac{G_{r1} R_2^2 \sigma}{4\pi G_{r2} R_0^2 R_1^2}. \tag{3.10}$$

For example, it is assumed that a typical direct path distance of 40 km is present. The TDR, for this scenario, is shown in Figure 3.3. Here the TDR is that received at the site and, as such, both antenna gains are set to unity. The target is also set to exist on the vector originating at the receiver and crossing through the receiver which is shown in Section 3.2.1 to be the optimal configuration. The RCS of the target is set to be 100 m^2 which is representative of a large passenger aircraft [44].

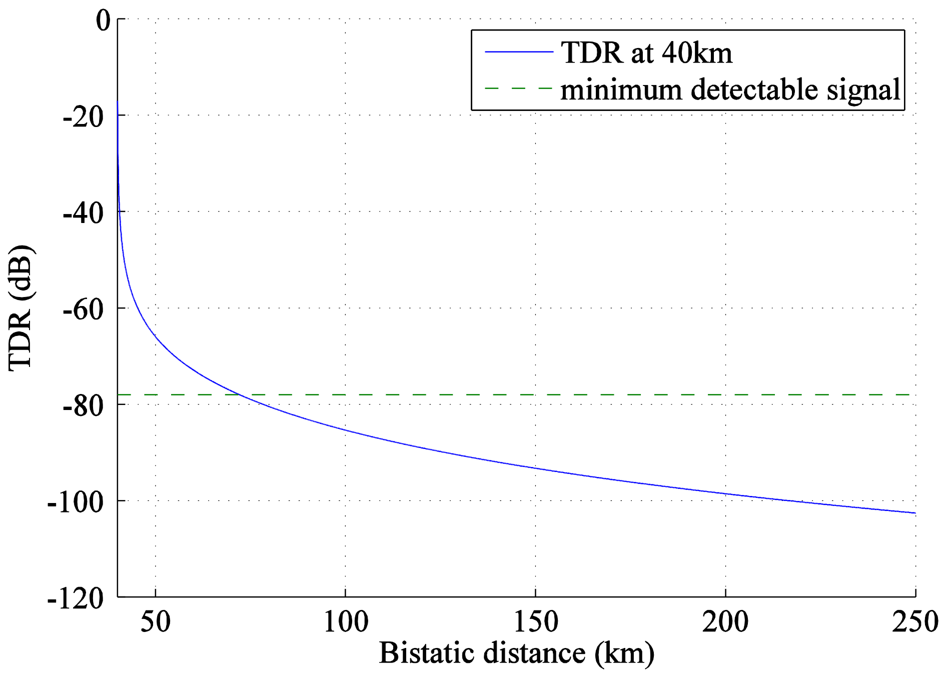


Figure 3.3 Surveillance to direct path signal ratio.

Even in this near optimal configuration, the ADC has been fully saturated, and if no steps are taken to remove the DPI prior to digitisation, the maximum detectable bistatic range for a receiver with a 78 dBc dynamic range is 72.0 km, which results in a 16.0 km distance from the receiver. It is, therefore, clear that, even in such convenient conditions, the radar would have a poor detection range for even the largest targets and is limited by the dynamic range of the receiver.

3.2.1 Optimal position

Setting the objective to maximise the TDR for a given target, $R_1$is then defined in terms of R_0, R_2, and γ.

$$R_1^2(\gamma) = R_0^2 - 2R_0R_2\cos(\gamma) + R_2^2 \tag{3.11}$$

Therefore, it can be seen that $R_1(\gamma)$ reaches maximum and minimum values

$$R_1(0) = R_0 - R_2 \tag{3.12}$$

$$R_1(\pi) = R_0 + R_2 \tag{3.13}$$

with

$$TDR(0) = \frac{G_{r1}\sigma}{4\pi G_{r2}}\left(\frac{R_2^2}{R_0^2(R_0 - R_2)^2}\right) \tag{3.14}$$

$$TDR(\pi) = \frac{G_{r1}\sigma}{4\pi G_{r2}}\left(\frac{R_2^2}{R_0^2(R_0 + R_2)^2}\right). \tag{3.15}$$

Hence it can be seen that the TDR is maximised for a given range from the receiver when the site intersects the line between the transmitter and the target, i.e. $\gamma = 0$. Furthermore, it can be seen that, when $\gamma = 0$, as R_2 increases so does the TDR, though this is intuitive as this moves the receiver closer to the target and further from the transmitter.

Additionally, the effect that increasing R_2 has on the TDR with a constant R_1, needs to be determined, i.e. does increasing the direct path distance increase the surveillance range? Rewriting (3.14) in terms of R_1 results in

$$TDR(0) = \frac{G_{r1}\sigma}{4\pi G_{r2}}\left(\frac{R_2}{R_1(R_1 + R_2)}\right)^2 \tag{3.16}$$

which we can analyse using

$$\frac{R_2}{R_1 + R_2}. \tag{3.17}$$

The expression in (3.17) begins at 0 when $R_2 = 0$ and converges to 1 as $R_2 \rightarrow \infty$. Hence it can be seen that the TDR ratio is maximised when $\gamma = 0$ and the baseline distance is maximised. The same result has been found in literature [31].

3.2.2 Antenna selection

The requirements for a passive radar system differ slightly from conventional radar and communication systems. This is especially true in the directionless system considered in this report. The simplicity of the directionless system, does not allow nulls to be arbitrarily steered towards interference sources. Null steering has been shown to be an effective method of reducing interference and DPI [45].

3.2.2.1 Surveillance antenna

The surveillance antenna has the primary function of receiving reflection from targets. As such, it should limit energy received from all other sources of radiation, primarily in-band sources. However, it should allow a wide beamwidth (in most cases) as to enable detection of targets from a wider range of angles [43].

There are two performance limiting parameters which a well-designed antenna can improve. The first is with the minimum signal power required for detection. This parameter is potentially limited by the sensitivity of the receiver. A higher gain in the direction of a target results in more energy being delivered to the receiver. The second is DPI rejection, which can be achieved by introducing nulls and low sidelobe levels in the direction of interference sources. However, it should be noted that beam width and gain are inversely related, as such both a wide beamwidth and a high gain cannot be achieved.

In passive radar this is not a common limitation as the minimum signal power required for detection is typically limited by the dynamic range of the receiver. The instantaneous dynamic range of the signal can be reduced by not receiving energy from the direct path. This appears in (3.10). The ratio between these gains,

$$G_p = \frac{G_{r1}}{G_{r2}}, \tag{3.18}$$

represents the relative antenna gain between the direction of the direct path and the direction of the target. G_p can often have a significant influence over the TDR. As such, placing the antenna such that DPI lies in a null may be more beneficial than selecting an optimum position.

Therefore, selecting a surveillance antenna optimised for a high front-to-back ratio (or deep nulls) is more beneficial that an antenna optimised for absolute gain. This conclusion is only valid where the dynamic range of the receiver is the limiting factor.

3.2.2.2 Reference antenna

The reference antenna is required to receive a copy of the transmitted signal. To this end, the signal received should match that transmitted as much as possible. However, in practice, it is found that non-idealities cause this not to be the case.

It was found that the main causes of poor reference signal recreation are multi-path [20] and antenna modulation. Multipath is caused by the transmitted signal reflecting off objects. These add additional signal components (similar to those which make up clutter in the surveillance channel) to the reference channel.

The additional components affect the cancellation filter by ensuring that each column of A not only affects its own range-Doppler index, but also those corresponding to the additional components (with the appropriate offsets). As these additional components are not ideally weighted, the flexibility of the filter is hindered, and it can no longer reach an optimal solution.

The matched filter is affected by correlating not only with the range-Doppler bin desired but also with the additional components. This increases cross-bin dependence and sidelobe levels.

Antenna modulation is the process where the movement of an antenna creates a modulation on the received signal. This can be seen on large flexible antennas. While the modulation is low, due to extremely long CPIs used in passive radar, it can have a large effect on the accuracy of the transmitted signal's copy received from the reference antenna.

A highly directional antenna with a narrow beam is desirable in order to limit the multipath received by the reference antenna. Antenna modulation is reduced by securing the antenna, or using a stable and inflexible design.

3.2.3 Additional considerations

Beyond the position of the site, DPI rejection can be aided by placing the receiver behind a large object such as a hill or building, as depicted in Figure 3.1. This is termed DPI shielding. DPI shielding was found to be more effective than optimising the antenna pattern. It should be noted though, that large near-field structures alter the effective far-field antenna pattern of the antennas (although to an extent this is the objective with the DPI shielding). Consequently, the effectiveness of using a directional antenna can be inhibited. The effective antenna patterns must, therefore, be considered within their respective operational environments and not as independent element.

3.2.4 Transmitter selection

Transmitter selection is based on a number of factors but ultimately aims to increase SDR while ensuring that the transmitted signals remain as powerful as possible. In a system with poorer analogue filtering, a more powerful transmitter (in comparison to other sources in the spectral vicinity) is desirable as this improves the ratio between the desired signals and those of other transmitters, which in turn, results in the ADC being more effectively utilized, and thus enhancing its performance.

Finally, in the case where TDR is no longer the defining performance factor, a stronger transmitter increases the distance at which a target's reflection would fall within the minimum power receivable by a receiver [43].

3.3 RECEIVER SELECTION

The RF digital receiver, used to sample the reflections, is one of the most vital pieces of hardware in the system, due to its role sampling the EM spectrum into digital space. The receiver imposes fundamental limits on the system by defining the dynamic range, being the source of quantisation and other noise sources [7], and establishing and limiting the coherency between channels. Essentially, if the receiver fails to capture a signal, or distorts it, no amount of processing can detect or recreate it.

Table 3.1 Potential receivers.

Receiver	ADC bits/Rx channels	Operating Frequency [MHz]	Analogue bandwidth range [MHz]	SFDR [dBc]	Price per unit [$]
Pervices Crimson	$16/4$	$0-6000$	322	40	6750-00
Ettus USRP N210 (WBX)	$14/1$	$40-2200$	40	88	1717-00 (480-00)
Ettus USRP B210	$12/2$	$70-6000$	$0.2-56$	78	1100-00
Epiq solutions sidekiq	$12/2$	$70-6000$	$0.4-50$	68.7	8000-00
Quadrus DRU-244A-2-2-PCI	$16/2$	$DC-437$	$0.0008-40$	80	1990-00
Airspy	$12/1$	$24-1800$	$6-9$	80	199-00
HackRF One	$8/1$	$1-6000$	$1.75-20$	$-$	299-95
RTL-SDR Dongle	$8/1$	$24-1760$	$?-2.8$	~45	19-95

The receiver constitutes a large portion of the system's cost, as such, keeping the receiver's cost to a minimum had a large, positive, benefit on the system's cost. However, due to the importance of the receiver within the system, this cost saving had to be balanced against the receiver's performance, minimising cost without hindering the system's operation.

To achieve these goals, a range of SDRs were considered as they allowed for a configurable system with a software definable bandwidth and frequency, while still residing at a comparatively low price point. Any receiver considered must be able to operate in a configuration where at least two channels are synchronously and coherently sampled. The absence of thereof would render the reference and surveillance channels incoherent, resulting in any of the subsequent signal processing being ineffective. The receivers which were found to be potentially suitable for use as a low-cost passive radar receiver are shown in Table 3.1.

From Table 3.1 it can be seen that while some receivers can support narrow bandwidths, the majority are designed for wider bandwidth systems. Furthermore, it should be noted that even when the analogue bandwidth is narrow, due to the design of the RF receiver chain, full use of the ADC is not possible in congested spectra. An example of a common RF chain is shown in Figure 3.4. In this receiver chain the AGC is placed before the LPF. This is done in architectures such as the Ettus B210, Ettus WBX daughter board and Epiq solutions sidekiq.

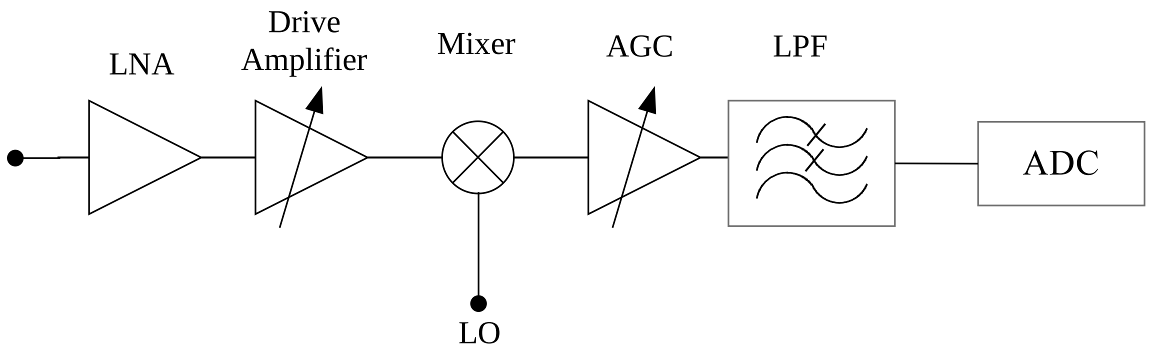


Figure 3.4 Super heterodyne RF receiver chain.

This architecture results in the AGC setting the gain such that the sum of all the mixed down signals would saturate either the AGC amplifier or the ADC. The issue arises when the signals of interest are of sufficiently small bandwidth and amplitude that the power of the signal entering the LPF and leaving the LPF are significantly different. This architecture is used for its simplicity and to lower noise in wide bandwidth environments.

In the case of passive radar, the surveillance signal is usually smaller than other signals in the FM band. This results in the above architecture limiting the saturation level of the ADC, thus decreasing the effective bit depth and SFDR.

Two receivers were then chosen for further investigation. The USRP B210 was chosen as it allows for both channels to be supported on a single board, with a low price point and configurable analogue bandwidth. The USRP N210 was chosen as, although it resides at a higher price point, it has a larger SFDR and bit depth.

The Airspy and RTL-SDR were not chosen as they do not have digital synchronisation, and as such, significant development and/or processing would be required to ensure synchronous reception of data from the devices. The Quadrus was also not investigated as it uses direct sampling and, as such, it requires additional filtering to prevent aliasing, which increases the development time and cost. The Pervices Crimson and Epiq solutions Sidekiq where not evaluated as neither offered sufficient performance to justify the cost.

3.3.1 Channel coherency

The coherency between two receiver channels was evaluated. Intra-channel coherency is required as the processing blocks require that the reference and surveillance channels are coherent. This requires the channels to keep a constant phase difference and relative amplitude. It has been previously show that degradation in channel coherency significantly degrades SNR during processing [46].

The phase stability was evaluated at 94.2 MHz as this corresponds with the frequency of a local radio station used to implement the radar (see Section 3.2.4). The experimental configuration is shown in Figure 3.5. A −10 dBm sinusoidal signal was generated using a signal source and fed through a power splitter directly to the inputs of each Rx channel on the receiver. This configuration ensured that the signals fed to the Rx channels were as coherent as possible.

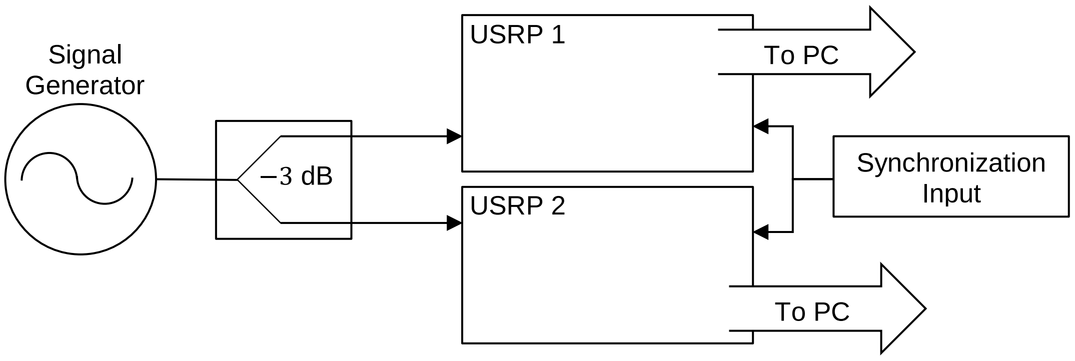


Figure 3.5 Phase stability evaluation configuration for N210.

The sample streams from the receivers were then saved to file and processed offline to evaluate the coherency of each channel. This methodology ensures that any significant incoherency was solely resultant from the receiver/s.

Four receiver configurations were evaluated, the means by which synchronisation is achieved is discussed later in this chapter. The configurations evaluated are:

- 2x N210s with WBX daughter boards, synchronised via a MIMO cable,
- 2x N210s with WBX daughter boards, synchronised via a PPS system,
- a standalone B210, and
- a B210 using an external reference and PPS source.

Two synchronisation methods are provided by the USRP manufacturer to synchronise boards. The first is a MIMO cable which is a plug-and-play solution allowing two boards to share digital and analogue reference signals. The second is the PPS system which requires that a 10 MHz reference sinusoid and a PPS signal (i.e. a block wave with a period of 1 Hz) be fed to each board.

The experimental configuration was set to gather samples at a rate of 200 kS/s for 15 minutes.

3.3.1.1 Metrics

The metrics used to evaluate the data collected are defined below.

Phase Noise

Relative phase is the difference in phase between two samples of the same signal. The phase error between two samples is calculated as

$$\phi_i = \arg(x_i y_i^*). \tag{3.19}$$

The phase noise for the channel is defined as the RMS phase noise between two vectors,

$$\varphi = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\phi_i^2} - \frac{1}{N}\sum_{i=1}^{N}\phi_i. \tag{3.20}$$

Phase drift

Phase drift is defined as the long term deviation in the relative phase between two channels. The major causes of phase drift in the receivers were found to be PLL slipping/ loss of lock and oscillator frequency offset. The phase drift was evaluated by fitting a line to the unwrapped relative phase between the channels. The gradient of that line is then the mean phase drift. The gradient for the line of best fit for a set of data is given as

$$m = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{3.21}$$

where

x_i and y_i are elements of the data coordinates x and y, and

$\bar{a}$ represents the mean of a vector a.

In the context of the phase drift (3.21) becomes

$$\Delta\phi = \frac{\sum_{i=1}^{N}(t_i - \bar{t})(\phi_i - \bar{\phi})}{\sum_{i=1}^{N}(t_i - \bar{t})^2} \tag{3.22}$$

where

t is a vector containing the times at which the samples were taken, and

ϕ is a vector of containing the relative phase between two channels.

3.3.1.2 Synchronisation

Achieving synchronisation between boards is achievable using several methods which provided by the SDR OEM. Only two of these were relevant to this implementation and are described below.

MIMO cable

Synchronisation via MIMO cable uses a plug-and-play cable which allows one USRP to act as a slave to another, piggybacking on the host's synchronisation and data link to the PC. A MIMO cable allows a maximum of two SDRs to be synchronised.

PPS

PPS synchronisation uses two input signal to synchronise SDR boards. The first is a 10 MHz reference which is used to synchronise the PLLs on a board and thus the mixing and sampling. The second signal is CMOS signal which sends a block wave at 1 Hz. The second signal is used to synchronise the clocks on the SDRs. This configuration can synchronise more than two boards.

3.3.1.3 Results

The performance of the receivers was evaluated and the results are shown in Table 3.2.

Table 3.2 Intra-channel coherency of receiver configurations.

Test configuration	Phase noise [mrad]	Maximum relative phase deviation [rad]	Phase drift [µrad/s]
N210 (MIMO)	15.006	0.368	7.434
N210 (PPS)	12.961	$70.18e-3$	1.135
B210	15.89	$55.61e-3$	-40.34
B210 (PPS)	4.490	π	76.83×10^{-3}

As shown in Table 3.2, the USRP B210 with an external reference clock has the lowest relative phase noise, and phase drift. However, it also experienced the largest maximum phase deviation. This deviation could be caused by the PLL slipping a cycle (and thus causing a phase shift) or a loss of samples. Without an external reference the B210 performed comparably to the N210 and experienced more stability than when fed with an external reference.

3.3.2 Conclusion

Of the two receivers which were further investigated, the N210 provides larger bit depth and higher dynamic range, but has a fixed analogue bandwidth, which, in a congested spectrum, reduces the achievable effective bit depth and dynamic range for the narrowband channel [7]. This occurs when signals which are not of interest are digitized by the ADC, thus reducing the signal of interest's dBFS.

The B210 configuration can therefore remain competitive in effective bit depth and dynamic range while providing the possibility for more coherent channels with an external reference. When the lower price is also considered the B210 becomes the obvious choice for a low cost receiver.

3.4 SUMMARY

The issues concerning signal acquisition for a passive bistatic radar have been discussed. The major performance limiting factors have been identified. Of these, it has been shown that target reflection to direct path ratio is the most critical consideration in most cases. Methods to address this have been explored, including site selection and direct path shielding.

Finally, a variety of suitable, and currently available, OTS receivers were evaluated for their suitability with regards to passive radar. It was found that the Ettus USRP B210 is currently the most suitable for use in a low-cost FM-based passive radar.

CHAPTER 4 PROCESSING PLATFORMS

Two general categories of processing platforms were considered, namely CPUs and GPUs. The basic differences and optimisation focus for each are discussed below. The processing platform, as it exists in the system, is depicted in Figure 2.1. Once the general concepts of the processors are presented below, the platforms on which the system was implemented are presented, and their specifications are compared.

4.1 CENTRAL PROCESSING UNIT

A CPU is the heart of most modern computers and comes in a variety of architectures and configurations with various performance characteristics. The most common architectures families are x86, which is a family of 32-bit processor architectures with backward-compatible CISC instruction sets [47]. A 64-bit variant is also available called x86-64. The second family of CPUs is the ARM family of processor architectures, which is an architecture optimised for power efficiency, and as such implements a RISC instruction set.

The x86 architecture family is commonly found in desktop and laptop PCs, as well as in many servers. The ARM architecture is found in many mobile and embedded devices such as smart-phones.

Optimisation on the CPU was performed by analysing the nature of the architectures. It should be noted that the majority of optimisation can be performed by using efficient algorithms and good programming practice. These are not discussed here, as the focus of this section is on architectural specific considerations.

4.1.1 Multithreading

Many CPUs have multiple cores. While the conventional programming model uses a single thread [48], to fully take advantage of a CPU's processing capabilities, all of its cores should be utilized [49]. The threading model implemented by CPU's is a heavy-thread model when compared to the thread model implemented in GPUs [50].

A heavy-thread model means that each thread is best utilised when computing a heavy work load. As such, each thread is able to process faster. There are, however, typically fewer processing cores, and the overhead for the creation, destruction and management of each thread is significant [49]. This model follows MIMD parallelism, or task parallelism, where threads can function independently of one another.

In general, a CPU is best utilised when the number of working threads is equal to the number of physical processing cores, or, if the CPU is hyper-threaded (i.e. has more virtual cores than physical cores) and the task is suitable, the best performance may be achieved when the number of active threads is equal to the number of virtual cores. Furthermore, a thread with a light-thread load may require comparable or more processing to initialise and manage, than the thread itself would require to execute [49].

For example, consider an N iteration loop that could execute in parallel on an M core processor, such that $N \gg M$. It would be more efficient to produce M threads of which each computes N/M iterations of the loop, than to initiate N threads each to compute an iteration of the thread.

Multithreaded implementation of most functionality is implemented in the linear algebra functions found in the libraries [51], as well as, the DFT functions found in the DFT libraries [51]. A common method of implementing this kind of parallelism is to use OpenMP which is a set of routines and compiler directives which allow for easy implementation of parallel processing on many common CPU architectures.

4.1.2 SIMD instructions and processor intrinsics

The x86 and ARM CPU architectures come in variants which have intrinsic instructions, allowing for higher computation rates. These functions are accessed via enhanced instruction sets [52]. These enhanced instructions allow a program to use dedicated hardware to perform multiple operations per instruction.

The first variant of these enables SIMD computing. This enables a single instruction to be applied to multiple pieces of data. Typically, these involve larger registers, such that multiple pieces of data fit in a register, and dedicated hardware which uses the registers [53]. For floating point arithmetic, the IEEE 754 floating point model is typically used [47] which defines a single-precision floating point number with 32 bits, which is henceforth referred to as a float.

The x86 architecture can come in a variety of extensions including AVX and SSE. SSE2 is now standardised on x86_64 architectures. The SSE2 standard allows for data to be stored in 128 bit, registers (i.e. 4 floats). Where each instruction is executed on all 4 floats resulting in the potential to increase the data throughput of each core of the processor 4 fold.

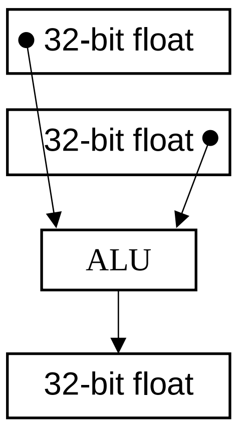


Figure 4.1 SISD floating point operation.

Figure 4.1 depicts a typical SISD operation where each register holds a single float. An instruction feeds the data from the two input registers and writes the result to the output register. Figure 4.2 depicts a typical SIMD operation. Two registers are still used, however, each is now 128 bits (assuming SSE2 is used). The operation is then completed by additional ALUs, or a specialised ALU which exist explicitly in hardware.

A common difficulty in implementing SIMD, and prevalent in SSE2, is that the data must be correctly aligned in memory. If the memory is not aligned a penalty is paid which can mitigate the benefit of using SIMD. SIMD is also implemented in ARM architectures. The most common version, NEON, is similar to SSE2 in that it implements 128 bit registers which can perform SIMD operations on 4 floats simultaneously. However, NEON, unlike, SSE2 is not IEEE-754 compliant.

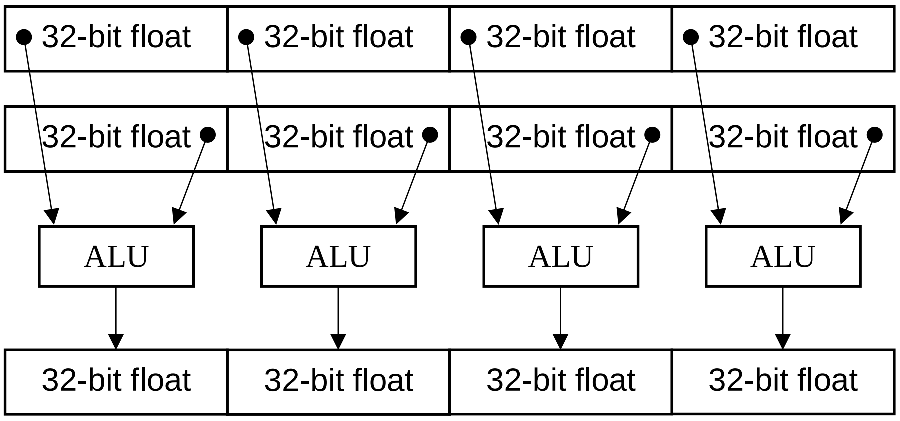


Figure 4.2 SIMD floating point operation.

Further extended instructions are implemented which can combine operations and reduce processing time. One of the most common is the inclusion of a fused add-multiply instruction (or multiply-accumulate (MAC) in DSPs). This allows the

$$c = c + a \times b \tag{4.1}$$

operation to be performed in a single instruction. This is commonly used to accelerate operations such as dot-products and matrix multiplications. The instructions are available on certain x86 and ARM architectures.

4.2 GRAPHICS PROCESSING UNIT

The GPU is a processor which is designed to rapidly create and manipulate images. Some GPUs can be separated from this role and be used for general computing tasks. These are known as GPGPUs [54]. The highly parallel structure of GPUs allows them to achieve a higher number of FLOPS than CPUs (up to 6.6 TFLOPS) as well as attaining high energy efficiency. There are a multitude of GPGPU architectures, but there are common characteristics among most which must be understood to fully utilize the processing power of a GPGPU.

4.2.1 Threading model

Unlike the CPU, GPUs use a light-threading model, and are highly parallelisable. Essentially this means that GPUs can efficiently split tasks into many simple threads without the incursion of a large overhead [54].

Achieving this level of parallelism does introduce a few drawbacks, specifically with regard to GPU architecture. It should be noted that the specific architectural design discussed here applies to NVidia GPUs. Although other GPU architectures may differ in specifics, the general concepts remain relevant.

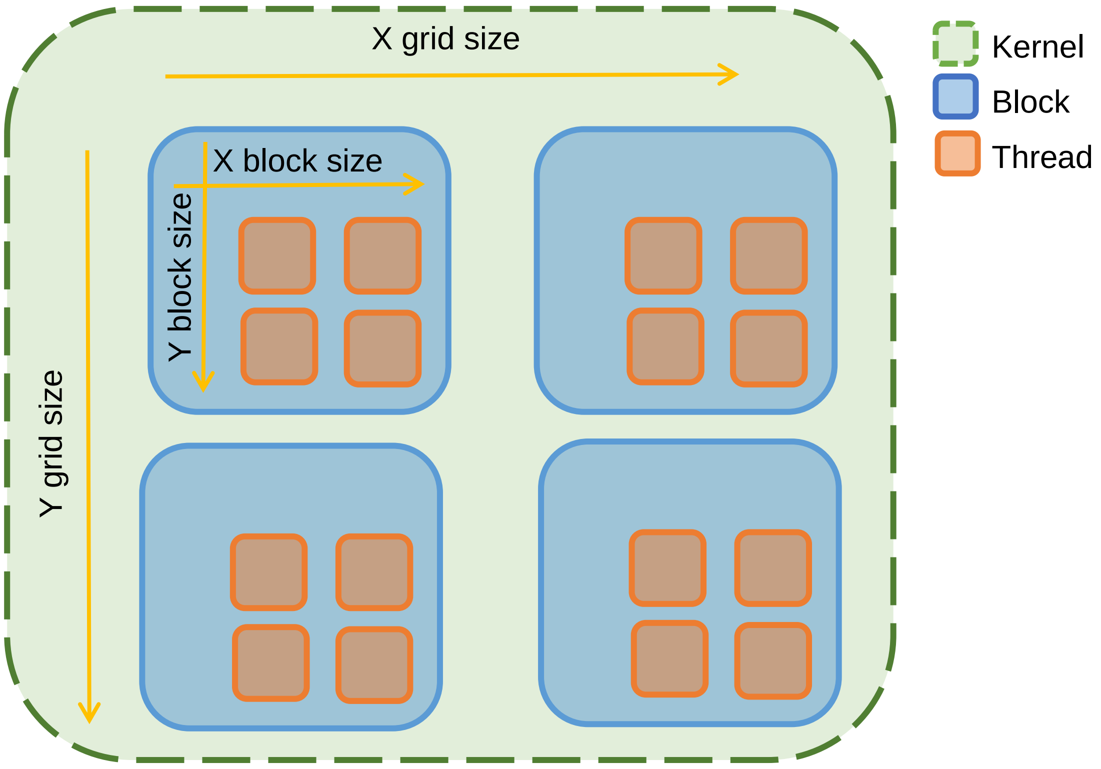


Figure 4.3 Diagram of a GPU's thread structure.

4.2.1.1 Thread structure

In order to introduce mass parallelism, GPU architecture splits the thread space into a structure which is simpler and easier to manage, as shown in Figure 4.3. Each call to the GPU is called a kernel. The thread structure of a kernel defines a two level grid. The base level defines a block of threads [50]. This block can be 3 dimensional, however the number of treads in each block is limited by the architecture, and NVidia GPUs limit this to 1024 threads per block [54]. The next level in thread space defines a grid of thread blocks. The grid can support blocks in 3 dimensions, NVidia GPUs (as of compute capability 3.0) limit the x dimension of the grid to $[2^{31} - 1]$ blocks and the other two dimensions to 65535 [54].

Each thread can read its own grid and block index as well as the size of the block and grid. It cannot, however, execute a separate program as is achievable with CPU threads [50], although, through the use of the threads index, each thread can follow different behaviour within that program.

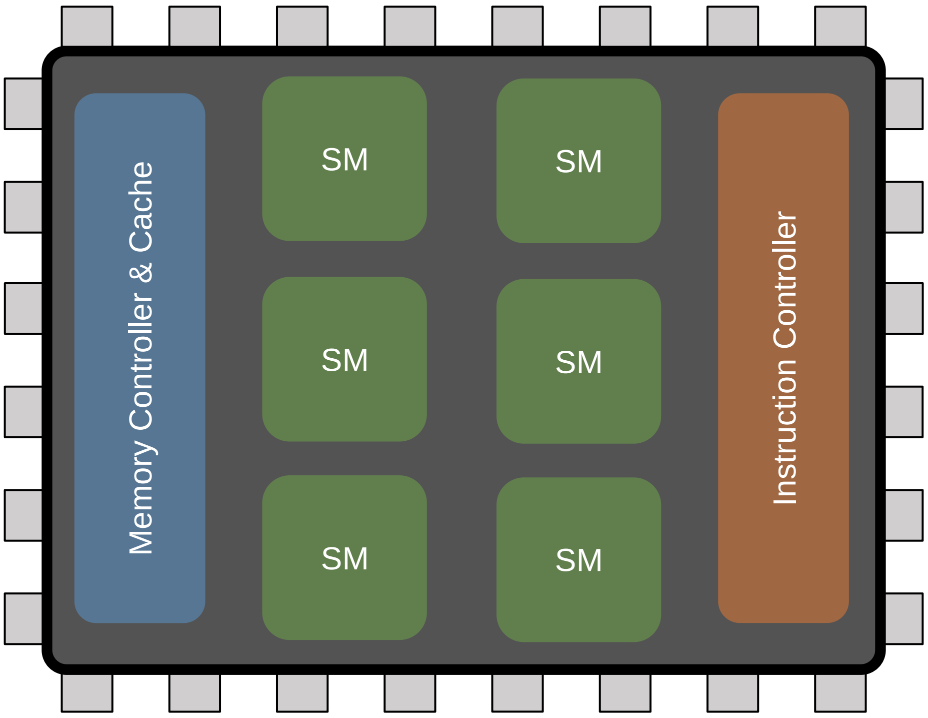


Figure 4.4 Diagram of a GPU's functional components.

It is important to note that whilst mass parallelism is achievable, communication between threads is limited. Many of the common multithreading tools available to a designer in the CPU environment are not available or, if they are available, their scope is limited. Some of the most useful tools are limited within the scope of a block [54]. These include synchronisation and the use of explicit writeable caches (shared memory). The lack of full synchronisation between blocks can affect the operation and performance of typical function such as reductions.

4.2.1.2 Execution of threads

In order to maximise the utilisation of the hardware, it is important to understand how the threads and blocks are mapped onto the GPU. The GPGPU operates by dividing the task load into a scalable problem size [54]. At the highest level, a GPGPU can be seen to contain a collection of SMs, each of which manage and execute many threads [54]. A functional diagram of a GPU is shown in Figure 4.4.

Upon a kernel call a number of blocks are defined which must be executed. The instruction controller (shown in Figure 4.4) on the GPU then allocates blocks to each SM as that SM becomes vacant. Each SM can handle more than one block concurrently (assuming that there are enough resources to launch multiple blocks of that type). There is, however, a limit to the number of blocks which each SM can concurrently execute [54].

A number of factors determine the number of blocks which an SM can concurrently execute. Firstly, assuming there are sufficient resources available, there is an architectural limit to the number of blocks that will concurrently execute on an SM. For GPUs with a compute capability of $3.x$, the limit is 16 blocks per SM, while for GPUs with a compute capability of $5.x$, the limit is 32 blocks per SM [54].

Secondly, there are a number of resources which a processor must be able to allocate to a block. As such, a limit in each resource reflects as a limit to the number of blocks which each SM can handle. Many of these resources are memory related and are discussed in Section 4.2.2, including registers and shared memory. The SM is functionally depicted in Figure 4.5 below. The memory related resources are shown in blue.
Aside from memory, there are further restrictions to occupancy of an SM. Each SM has a maximum number of threads and warps which it can handle. This means that a limited block size can result in underutilisation of each SM and thus the GPU.

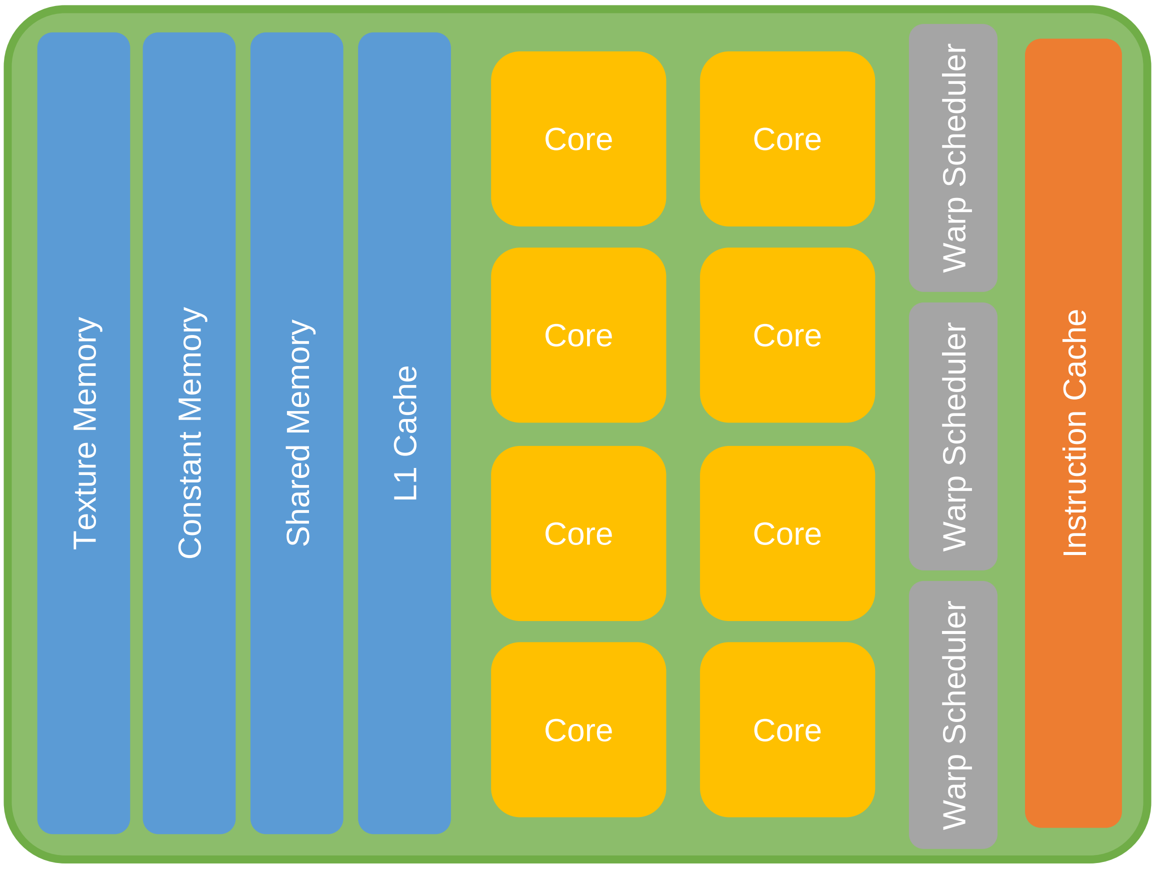


Figure 4.5 Diagram of an SM's functional units.

Each SM can be broken down into a number of warps. A warp is an instruction which takes control of a maximum of 32 cores which are executed with SIMT. Within a warp, each group of threads executes the same instruction which is broadcast by a warp scheduler. This architecture limits the overhead in dispatching, scheduling, and context scheduling each thread to the point where it is handled by dedicated on-chip hardware.

One of the issues that arises with this architecture, is the handling of branched code (i.e. if-else statements). If two threads within a warp reach a condition statement and branch off to execute different code, the warp would have to issue different instructions to each thread. To avoid this each condition would run as shown in Figure 4.6.

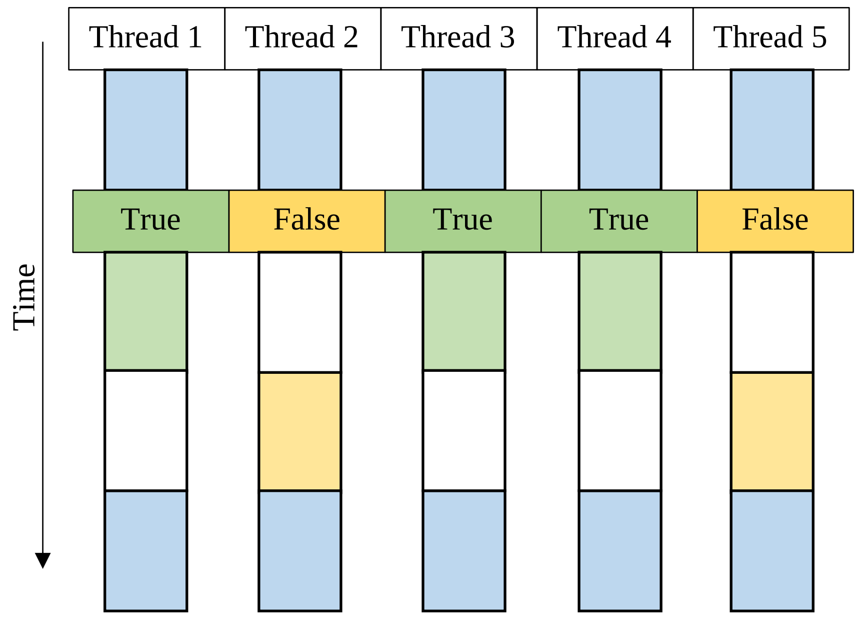


Figure 4.6 SIMT branch execution.

In Figure 4.6, each thread executes the same instructions in the blue region, then a condition instruction is met. The threads for which the condition block is true then execute the code in the branch (green region) while the blocks for which the condition is false do nothing (white region). When the true branches have returned the false branches execute (yellow region) while the threads for which the condition was true do nothing. The false branches return and the threads continue executing the same instructions together again. Note that true branches do not necessarily execute first, this is merely a demonstrative explanation.

In this manner, branched code can be handled but incurring a significant penalty when threads on the same warp execute divergent code. It is thus the designer's task to ensure inter-warp divergence is limited and that synchronisation between blocks is minimised. This is because inter-block synchronisation is only achievable through multiple kernel launches. Furthermore, when optimizing for a specific architecture, the block and grid sizes should be selected to allow for maximum utilisation of the GPU's processing performance.

4.2.2 Memory structure

Due to the mass parallelism implemented on GPUs it is possible to achieve extremely high computational rates. Often though, a major limiting factor is being able to feed the processing cores such that they can continuously process data. In order to feed the cores at a high enough rate, GPUs implement a complex memory hierarchy which allows designers to cache and feed data to the processors.

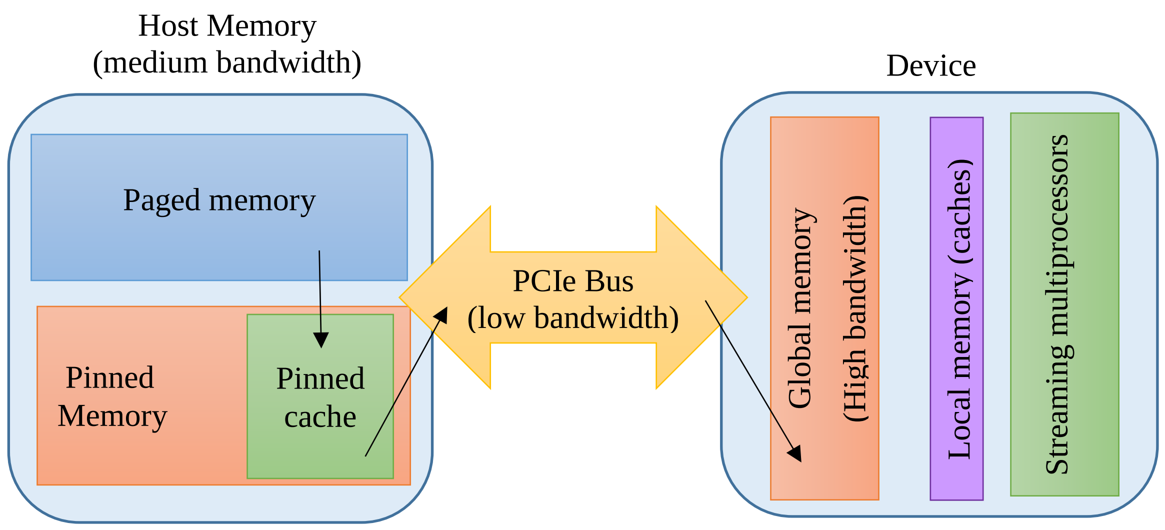


Figure 4.7 Memory transfer from paged memory to global memory on a discrete GPU.

The memory of discrete GPUs differs from that of embedded GPUs, largely because discrete GPUs have a frame buffer on separate memory hardware to the host CPU, while embedded GPUs share the same physical memory. A typical memory copy from paged memory to global memory on a discrete device is shown in Figure 4.7.

As can be seen from Figure 4.7, the transferring from paged memory to the device or global memory requires copying the paged memory to a cache or buffer of pinned memory. The GPU driver requires that memory copies to and from the host performed using pinned memory. The pinned cache is then transferred across a PCIe bus and is copied to global memory which sits on the device. This can be compared to an embedded GPU where the CPU and GPU share the same RAM, as shown in Figure 4.8. While the GPU must still access pinned memory, there is no need to transfer the memory across a slower bus allowing the CPU and GPU to use the same memory. The disadvantage with this configuration is that the GPU must access slower CPU memory which can limit the performance of the GPU.

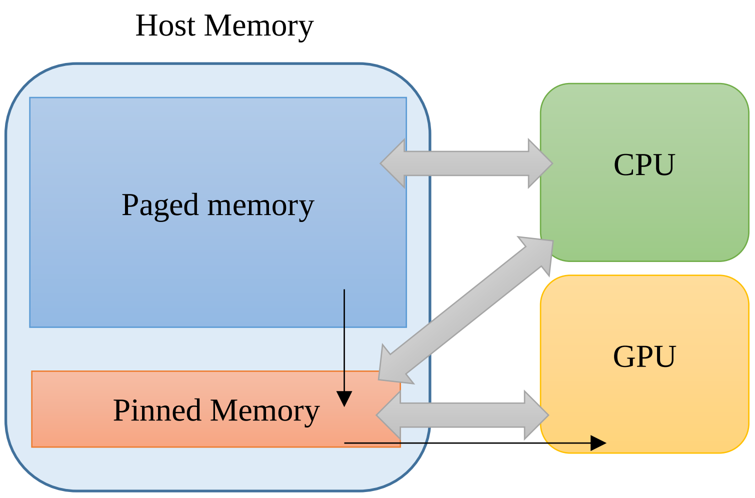


Figure 4.8 Memory transfer from paged memory to global memory on an embedded GPU.

Due to the limited automatic caching and memory management in GPUs [54], memory management must be handled/optimised manually. In order to successfully optimise memory access, the memory model must be understood. The memory types are detailed in Addendum B.

4.3 COMPARING GPU AND CPU PROCESSORS

From the previous section it is clear that many similarities exist between CPUs and GPUs, and that many problems can be solved using either one. This section serves to highlight the differences between the two processing platforms with regard, to the processors’ suitability in solving different problems.

It should be noted that a GPU cannot operate independently of a host CPU, and as such, on a system with a GPU non suitable problems can be handed off to the other processor, although in the case of a discrete GPU there is a transfer cost which must be taken into account.

4.3.1 Task and data parallelism

The first and probably most crucial concept to grasp when deciding between processing platforms is the difference between task and data parallelism. Task parallelism refers to a task’s capability to be separated into different parts which are processed concurrently [49].

Each part, or thread, can process the same data or different data, and can execute the same task or different tasks. Data parallelism refers to the same task being performed on different data concurrently [49]. As such, data parallelism is more restrictive than task parallelism; however, many computing operations still exhibit behaviour which can be data parallelised.

The CPU is an example of a processor which can utilise task parallelism using threads. Each thread can operate independently and threads can run concurrently (assuming a multi-core CPU). CPUs can take further advantage of data parallelism within each thread. This is predominantly performed using SIMD instructions. Furthermore, in the case of a serial task, the CPU is more suited as its heavy-thread structure and typically higher clock speed result in higher FLOPS/core. Thus, when processing a serial task that can only use one core, a CPU would normally outperform a GPU.

GPUs, on the other hand, cannot easily take full advantage of task parallelism (as discussed in Execution of threads), as task parallelism is executed serially instead of in parallel within a warp, and outside of that, task parallelism will often impair memory coalescing [49], thus slowing the execution of kernels. Rather, GPUs are generally more suited to data parallelism, and even more so, mass data parallelism, where all the cores can be utilised.

4.3.2 Latency and throughput

Assuming the task is efficiently implemented on both processors, task suitability can be decided by other factors such as latency. Even if a GPU is well utilised, the memory transfers required and driver calls which kernels must pass through, generally result in a higher latency (time between issuing the task and processing to start, including the time after completion of processing to fetch the results) than a CPU [55].

On the other hand, when computational requirements are limiting execution time and sufficient data parallelism exists, GPUs can achieve higher throughput. In operations requiring sufficient computation, the increase in throughput can compensate for the latency experienced in issuing the task to a GPU.

In general GPUs are more suited to large computationally intensive tasks which can tolerate high latency. Conversely CPUs are suited to tasks with lower latency but also lower throughput. Although not considered in this study, it should be noted that FPGAs and DSPs generally exhibit even lower latency than CPUs.

4.4 IMPLEMENTATION HARDWARE

Two processing platforms were used to evaluate the computational performance of the design. Both platforms were chosen to be mobile and to have CUDA capable GPUs to allow for the easy transportation and implementation of the system. Mobile systems were chosen as mobility compliments the concept of a low-cost passive radar, by allowing a disposable [56] or low-cost system to deploy in remote regions where power or access restrictions could cripple conventional radar.

4.4.1 Mobile system

The mobile system in this context refers to an OTS laptop which was used as the host for the receiver and to perform processing. The laptop used Windows 8.1 as an operating system and CUDA 7.5 was used to implement the GPU aspects of the system. The host code (C++) was compiled using MSCV12.0. The host specifications for the mobile section are shown in Table 4.1.

Table 4.1 Mobile system host specifications.

CPU	# Cores	Memory / BW	SIMD Extensions	Processor TDP
Intel® Core™ i7-4710MQ	4 (8 threads) @ 2.5 GHz (3.5 GHz boost)	16GB DDR3 1600 MHz / 38.4 GB/s	SSE4.1/4.2, AVX2	47 W

The specifications for the GPU on the mobile system are shown in Table 4.2. The peak FLOPS of the mobile GPU is 3104 GFLOPS while the mobile CPU has 27.76 GFLOPS. It can therefore be seen that the raw processing power of the GPU far exceeds that of the CPU. However, utilising that capability is difficult.

Table 4.2 Mobile system device specifications.

GPU	# CUDA Cores	Memory / BW	Compute capability	Processor TDP
GeForce GTX 880m	1536 @ 954 MHz	8GB GDDR5 @ 2500 MHz / 160 GB/s	3.0	122 W

An example of this can be seen when the memory bandwidths are compared. The CPU has a memory bandwidth of 38.4 GB/s which given a 4-byte float translates to 9.6 Gfloats/s and a theoretical compute performance of 27.76 GFLOPS, which translates to a 2.89 times higher compute to memory ratio.

The GPU with a compute performance of 3104 GFLOPS and a memory bandwidth of 40 Gfloats/s has a compute to memory ratio of 77.6. Therefore, minimising global memory access is critical when designing high performance kernels.

4.4.2 Embedded system

The embedded system was an NVidia Jetson TK1 development board [57] designed for low power mobile computing. The development board was used to host the receiver as well as perform all the signal processing. The embedded system ran Jetson TK1 DevKit L4T OS as the operating system, essentially a modified version of Ubuntu for ARM. CUDA 6.5 was used for all GPU aspects, as later versions of CUDA are not supported for the development board. Host code was compiled using g++ 4.9. The host specifications for the embedded system are shown in Table 4.3.

Table 4.3 Embedded system host specifications.

CPU	**# Cores**	**Memory / BW**	**SIMD Extensions**	**Processor TDP**
"4-plus-1" core ARM cortex-A15	4 @ 2.32 GHz	2GB DDR3L @ 933MHz / 14.93 GB/s	NEON	<11 W (entire board)

The specifications for the device are shown in Table 4.4. The Jetson TK1's GPU has a theoretical maximum floating point calculation rate of 326 GFLOPS. This is much lower than the mobile processor but requires only 11 W for the entire development board, while the mobile systems GPU and CPU alone require 169 W. The embedded system is thus a low power, low cost, low performance system when compared to the mobile system.

Table 4.4 Embedded system device specifications.

GPU	**# CUDA Cores**	**Memory / BW**	**Compute capability**	**Processor TDP**
Tegra TK1	192 @ 852 MHz	2GB DDR3L @ 933MHz / 14.93 GB/	3.2	<11 W (entire board)

4.5 SUMMARY

Two processing platforms are considered. It was shown that GPUs are suited to arithmetically intense tasks that exhibit massive data parallelism. On the other hand, CPUs are best suited to more complex memory accessing patterns and serial algorithms.

CHAPTER 5 PROCESSING CHAIN

The processing chain takes the raw samples of the electromagnetic energy received by the antennas and processes them to extract usable information. The processing chain is broken down into blocks. The operation of the processing chain, and hence the association between the blocks which are discussed later in this chapter, is shown in Figure 5.1.

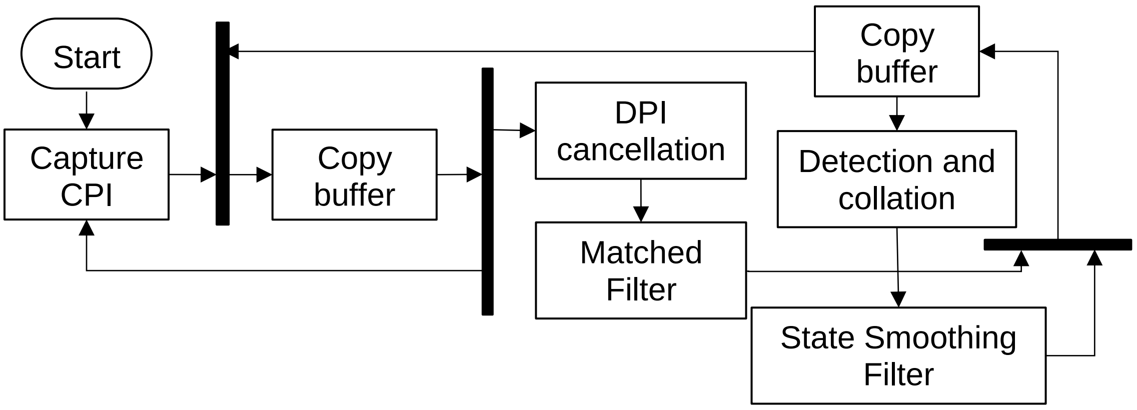


Figure 5.1 Detailed Processing chain.

The metrics used in the evaluation of the processing chain are detailed in Addendum C.

5.1 INITIAL EVALUATION

In order to gain basic understanding into the passive radar processing chain, and to initially evaluate where improvements could be made, a processing chain was developed, based on literature [12] and fed with a second of captured data. The processing chain was implemented in MATLAB on the mobile platform. In the literature [12], an adaptive lattice filter was used to remove the direct path and clutter. A 50 stage filter was used in the literature. It was found, however, that in order to cancel all the significant clutter, a 200 stage filter was required. In other literature it has been shown that a Wiener-Hoph filter is optimal [9]. As such, a Wiener-Hoph filter was also implemented.

Two methods for implementing a matched filter are presented. The first uses FDC, while the second uses a decimating filter followed by FDC. The decimating method requires fewer computations than straight FDC, however it results in sub-optimal correlation due to aliasing. All other parameters were kept equivalent to those specified in [12]. The average runtimes of the simulations are shown in Table 5.1.

Table 5.1 Average execution time of processing blocks for the preliminary investigation.

Function	Average Execution Time (s)
Wiener-Hoph Filter	13.09
Adaptive Lattice Filter	23.65
FDC	2.754
Decimated FDC	3.045
CFAR	7×10^{-3}

As can be seen from Table 5.1, the adaptive filters take far longer to execute than the matched filters. Additionally, it was found that the execution time of these processing blocks is not directly related to the number of computations, but is also dependant on the processing block's ability to effectively use the processing platform's hardware.

In order to realise a real-time system on either of the platforms, it is clear that a more efficient processing chain is required. As the adaptive filters are the most demanding, the process began and mainly focused on reducing the execution time of the DPI and clutter cancellation filter. The matched filters also require a reduction in execution time. However due to the highly optimised FFT algorithms already available in many libraries there is limited improvement which can be made aside from implementing the most optimised variant and streamlining memory access.

The Matlab implementation was no longer used, and an operational system was developed based on C++ and CUDA.

5.2 CANCELLATION FILTER

The cancellation filter interacts with the remainder of the processing chain as depicted in Figure 5.1. The cancellation filter operates by removing specified components from the surveillance signal, namely the direct path and clutter. These components create sidelobes on the ARD region where targets exist, thus preventing the targets from being detected. The resultant surveillance channel is defined by

$$\hat{s} = s - \widehat{w} \tag{5.1}$$

where

s is the surveillance channel, and

$\widehat{w}$ is the filters estimate of the direct path and clutter components in s.

Alternatively, in range-Doppler space, (5.1) can be written as

$$\hat{s} = s - A\hat{x} \tag{5.2}$$

where

$\hat{x}$ is a vector containing the estimated filter weights, and

A is defined in Section 5.2.1.

The cancellation filter, as an adaptive filter is described functionally in Figure 5.2.

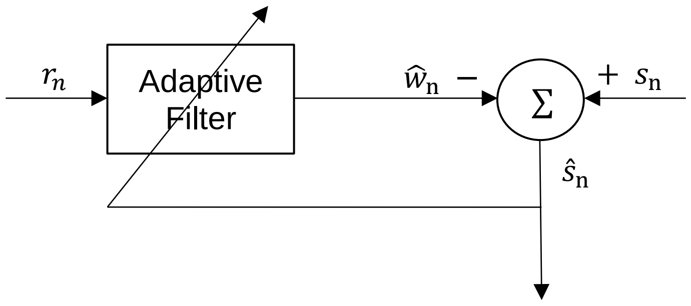


Figure 5.2 Functional Diagram of cancellation filter.

From Figure 5.2, it can be seen that the DPI and clutter cancellation filter is essentially an adaptive interference cancellation filter. The difference between a conventional interference filter and a DPI cancellation filter is that the cancellation region is less strictly defined.

5.2.1 Choice of cancellation region

The region where cancellation should be applied is where the DPI and clutter lie. This area is defined with regard to the cancellation filter by defining a matrix A which is an $M \times N$ matrix. N defines the length of the surveillance and reference channels over which cancellation applies, and M defines the number of ARD bins which are cancelled as well as the degrees of freedom allocated to the filter. It should be noted that, due to the long CPI resultant from a desired SNR gain, it is assumed that $N \gg M$.

The cancellation region is, therefore, selected by the choice of the columns of A. Each column of A is created by delaying r by the number of samples corresponding to range index desired, and modulating that result with respect to the Doppler index required.

Ideally the clutter and DPI would exist solely without a Doppler shift; in practice this is rarely the case. Clutter spread can be caused by non-stationary antennas, moving clutter such as trees and imperfect coherency between channels. In order to consistently remove the DPI and clutter, the cancellation needs to be widened to include additional Doppler indices. Two methods can be used to achieve this.

The first method is simply to include the additional ARD bins into A. The result is that M becomes larger, thus increasing the computational requirements. The second method [2] involves breaking the reference and surveillance channels into batches of equal size and applying the filter to each batch. The effective cancellation width in Doppler is then

$$B_d = \frac{F_s P}{N} \tag{5.3}$$

where

F_s is the sampling frequency, and

P is the number of batches.

The batch length is then given by

$$N' = \frac{N}{P}. \tag{5.4}$$

This is obviously only an option when P is a factor of N.

Several methods can be used to estimate the filter weights. The most applicable methods are grouped into three categories and discussed below.

5.2.2 Wiener-Hoph Filter

The Wiener-Hoph filter is a linearly optimum filter [58] which can be applied to stochastically stationary processes. The filer can find the optimal least squares solution. However, it is normally regarded as computationally cumbersome [59] and ill-suited to real-time applications [9]. The filter is defined by solving the Wiener-Hoph equations [58],

$$\sum_{i=0}^{\infty} \hat{x}_i \, \mathrm{E}[r_{n-k} r_{n-i}^*] = \mathrm{E}[r_{n-k} s_n^*] \tag{5.5}$$

where

$\mathrm{E}[x]$ is the statistical expectation of x,

n is the current sample number,

$\hat{x}_i$ is the ith filter weight,

x^* is the complex conjugate of x, and

$k = 0,1,2, \dots, \infty$.

The filter described by (5.5) is infinitely long and requires infinite statistical information, which is clearly impractical. For the case when the number of filter weights is finite, the filter can be rewritten as

$$\sum_{i=0}^{M-1} \hat{x}_i \, \mathrm{E}[r_{n-k} r_{n-i}^*] = \mathrm{E}[r_{n-k} s_n^*] \tag{5.6}$$

where

$k = 0,1,2, \dots, M-1$.

The filter can now be expressed in matrix form by defining the $M \times M$ auto-correlation matrix,

$$C = \mathrm{E}[u(n) u^H(n)] \tag{5.7}$$

where

$u(n)$ is a vector of the filter's inputs (defined by the selection of the cancellation region) at sample n.

Next a cross-correlation vector is defined as

$$d = \mathrm{E}[u(n)s_n^*]. \tag{5.8}$$

The filter is then written as

$$C\hat{x} = d. \tag{5.9}$$

The auto-correlation matrix, C, is now an $M \times M$ matrix and d is an M element vector. The square system can now be solved via a number of numerical linear equation solvers which are suited to square matrices. However, the statistical properties of the system are unknown and thus the statistical expectation cannot be calculated. Instead, it is estimated.

C and d may be estimated from the inputs to the filter. The estimations are known as sample covariance matrices. The estimate of C is found using

$$\hat{C} = A^H A \tag{5.10}$$

and the estimate of d is found using

$$\hat{d} = A^H s. \tag{5.11}$$

It should be noted that the estimates in (5.10) and (5.11) need to be scaled by $1/N$. However, in (5.9) the two constants cancel and, as such, there is no need to calculate and scale these values.
The approximate computational complexities for a variety of common solvers are shown in Table 5.2, these are derived in Addendum A. Where the NM^2 term is related to the formation of C, and the N^3 term is related to the factorisation.

Table 5.2. Computational cost of Wiener filters

Solver	Time cost	Flop count
LU	$\frac{2}{3}M^3 + 2NM^2$	$\frac{8}{3}M^3 + 16NM^2$
LDL	$\frac{1}{3}M^3 + 2NM^2$	$\frac{4}{3}M^3 + 8NM^2$
Cholesky	$\frac{1}{3}M^3 + 2NM^2$	$\frac{4}{3}M^3 + 8NM^2$

5.2.3 Linear least squares

Linear least squares solves the system of linear equation

$$\min_{x} \|b - Ax\| \tag{5.12}$$

There are a number of methods which can be used to directly solve the least squares problem. Assuming that A is invertible, the solution is given by

$$x = (A^H A)^{-1} A^H b. \tag{5.13}$$

However, due to numerical limitations and stability the inverse is rarely ever calculated. A few methods for solving (5.12) are introduced in Addendum A. The computational complexities for these are also derived in Addendum A, and the approximate computational complexities are shown in Table 5.3.

Table 5.3. Computational cost of least squares filters

Filter	Time cost	Flop count
QR	$2NM^2$	$8NM^2$
CGLS	$4NM(K+1)$	$16NM(K+1)$

5.2.4 Miscellaneous Filters

Several other filters have been explored in literature. These include the CGLS filter, and the Gradient adaptive lattice filter. These are introduced in Addendum A, where their computational complexities are defined. Another method is introduced below, called correlative cancellation. The approximate computational complexities are shown in Table 5.4.

Table 5.4. Computational cost of miscellaneous filters

Filter	Time cost	Flop count
CLEAN	$2NM^2$	$8NM^2$
Correlative	$4NM(K+1)$	$16NM(K+1)$
Gradient Adaptive Lattice	$26NM$	$69NM$

Additional filters are also explored in literature, however, many of them require careful tuning to achieve suitable cancellation results [33]. These filters were, therefore, not further explored.

5.2.4.1 Correlative cancellation

Correlative least squares is an algorithm intended to allow sufficient, but not optimal, DPI cancellation with minimal time cost and FLOP count. The algorithm operates by iteratively removing the DPI and clutter components relating to each row of A.

This algorithm can be seen to operate similarly to a CLEAN algorithm except, instead of the algorithm determining which column of A is the largest contributor, the algorithm uses approximations to predetermine which column of A to remove. As such, compared to the CLEAN algorithm, the computational cost of removing each component is comparatively low.

The algorithm has two loops. The inner loop runs for each column of A, i.e. $i \in 1,2, \ldots M$. First, the component of the ith column of A in b is estimated by

$$x_{ij} = \frac{\hat{b}_{ij} \cdot A_i^*}{\|A_i^*\|^2} \tag{5.14}$$

where

$\hat{b}_{ij}$ is the resultant surveillance channel after iteration ij.

While $||A_i^*||^2$ is required it can be approximated by $|A_i| = |A_0|$ and as such only needs to be calculated once. The update is calculated using

$$\hat{b}_{(i+1)j} = \hat{b}_{ij} - x_{ij}A_i. \tag{5.15}$$

The outer loop runs over K iterations such that $j \in 1,2, \ldots, K$.

5.2.5 Optimisation

What optimisation is available for the general operations used in the cancellation filters, is already implemented in mature libraries. These libraries were used for the realisation of the filters. In order to reduce the algorithmic requirements, the generality of the system must be reduced.

Firstly, the assumption is placed that the DPI clutter exists where the reference signal is unmodulated, i.e. the DPI and clutter are centred on the zero Doppler index. Secondly, it is assumed that a widening in the cancellation region in the Doppler dimension (as described in Section 5.2.1) results in A consisting of delayed versions of the reference channel. The problem is then further restricted by defining each row of A as

$$A_i = r_{2-i:N-i} \tag{5.16}$$

where

r is the reference channel vector, and

$i \in 1,2 \ldots, M$.

The elements of r where $i < 1$ are samples received before the start of the surveillance channel. Hence, A is now defined as a sampled tapped delay line.

5.2.5.1 Algorithmic optimisation

When analysing the time cost and FLOP count required, it can be seen that for the group of reduced linear solvers (refer to Table 5.2), the majority of the processing is required to reduce the system, primarily (5.10), with a time cost of $2NM^2$ operations.

The first, and obvious step, would be to take advantage of the Hermitian structure of C. Doing so allows only the upper or lower triangular sections to be explicitly calculated, while the remaining entries can be filled in as the conjugate of the corresponding element such that

$$C^*_{j,i} = C_{i,j} \tag{5.17}$$

where

$i \neq j$.

This reduces the time cost to $NM(1 + M)$ operations.

Secondly, the standard matrix multiplication algorithm calculates each element of C as,

$$C_{i,j} = \sum_{k=1}^{N} A_{i,k} A^{H}_{k,j} = \sum_{k=1}^{N} A_{i,k} A^{*}_{j,k}. \tag{5.18}$$

Given that A is described by (5.16) and

$$r_i = 0 \;\; \forall\, i \in Z < 1 \tag{5.19}$$

each element of C where $i \geq j$ is given by

$$C_{i,j} = \; r_{i:N+i} \cdot r^{*}_{i-j+1:N+i-j+1} \tag{5.20}$$

for all elements of C there exists only $NM + M^2$ unique products. The matrix can then be filled by calculating the elements $C_{M,1:M}$ (essentially a matrix-vector product) and filling the diagonals as shown in Figure 5.3.

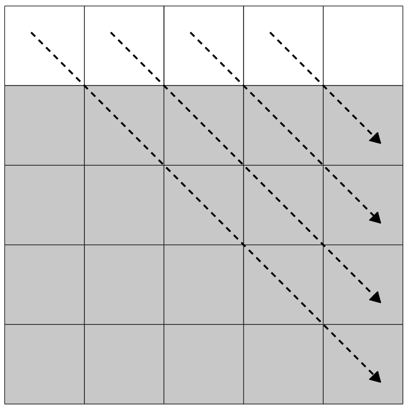


Figure 5.3 Optimised matrix fill algorithm.

Once the unshaded elements ($C_{M,1:M}$) are calculated, the matrix is populated by sequentially applying (5.21) in the direction of the arrows (shown in Figure 5.3).

$$C_{i+1,j+1} = C_{i,j} + A_{i,i+N+1} A^{*}_{j,i+N+1} - A_{i,i} A^{*}_{j,i} \tag{5.21}$$

The time cost is therefore reduced to $2NM + 2/3M^2$ and the FLOP count to $8NM + 8/3M^2$.

5.2.5.2 GPU optimisation

The GPU allows for a high number of FLOPS to be processed. Utilizing this potential, however, requires careful design and specific strategies. In many of the functions above, the computational complexity is in the order of the size data. For example, $s = Ax$ has a time cost of $2NM$, but to execute this the processor must access NM data pieces. This generally results in a bottle neck where the processors spend the majority of the application runtime waiting for the information which they require.

However, if the restrictions to the problem are kept in mind, many of the data dimensions can be reduced to N instead of NM. This allows for an increase in the processing to memory access ratio. The strategies and principles followed in order to implement this are discussed in Section 4.2.
It should be noted that by replacing the library implementations of the functions with those discussed in this section, the cancellation filter's dependence on A is removed and it no longer needs to be explicitly calculated. This allows for a reduction in both memory usage and processing.

Matrix-vector multiplication

In this function the matrix A $(N \times M)$ is multiplied by an $M \times 1$ column vector x resulting in an $N \times 1$ column vector y. The required operation for each element of y is

$$y_i = x \cdot r_{i-M+1:i}. \tag{5.22}$$

Firstly, parallelism can be established by allowing each thread to calculate each element of y. Access redundancy in x and in sections of r are evident. In order to reduce the number of accesses to global memory, shared memory is used. This is achieved by loading a block of r and x elements in to shared memory. The operations which use those elements execute, and once completed, the next blocks are loaded in. This technique provides acceleration because it reduces the latency when elements are accessed multiple times within a thread block. However, in situations where elements are not accessed numerous times, this would not provide any performance gain and may even result in degradation.

The library (gemv2N) [60] and optimised (shiftvec_AB) kernels were then profiled using the final implementation configuration, i.e. $N' = 4000$, and $M = 256$, on the mobile system. The results are shown in Table 5.5.

Table 5.5 y=Ax kernel profiles.

Kernel name	Execution time (μs)	Occupancy (%)	Achieved GFLOPS
gemv2N	741.17	56.25	117.32
shiftvec_AB	166.00	100	492.44

As seen in Table 5.5, there is a marked improvement in both execution time and achieved FLOPS. The optimised algorithm ran 4.46 times faster and achieved 4.20 the FLOPS.

Matrix-transpose vector multiplication

In this function the matrix A^H $(M \times N)$ is multiplied by an $N \times 1$ column vector x, resulting in an $M \times 1$ column vector y. The required operation for each element of y is

$$y_i = x \cdot r_{i-M+1:i}. \tag{5.23}$$

Parallelism could be achieved by allowing each thread to calculate an element of y, but as M is typically a relatively small dimension, this would not fully utilize the GPU. Instead, A was divided into shorter pieces such that each thread could calculate a section of an element of y, which were later consolidated to form the elements of y.

The library (gemv2T) and optimised (shiftvec_2AtB) kernels were then profiled using the final implementation configuration, i.e. $N' = 4000$, and $M = 256$, on the mobile system. The results are shown in Table 5.6.

Table 5.6 y=A^{H}x kernel profiles.

Kernel name	Execution time (μs)	Occupancy (%)	Achieved GFLOPS
gemv2T	917.01	75	99.121
shiftvec_2AtB	176.59	50	462.56

As seen in Table 5.6, there is a marked improvement in both execution time and achieved FLOPS. The optimised algorithm ran 5.19 times faster and achieved 4.67 the FLOPS.

When comparing the profiles of shiftvec_2AtB and shiftvec_AB, it can be seen that there is a large deviation in occupancy. However, the achieved FLOPS remains fairly constant. This would indicate that memory access remains the bottle-neck and further improvement is required to effectively utilize the GPU.

Matrix multiplication with its transpose

The optimization of $C = A^H A$, was implemented using the reduced algorithm discussed in Section 5.2.5.1. Parallelism was achieved by allowing each thread to calculate each diagonal. Global memory access was reduced through the use of shared memory.

The library(cgemm) and optimised (CeAhA and CeAhA_upper) kernels were then profiled using the final implementation configuration, i.e. $N' = 4000$, and $M = 256$, on the mobile system. The results are shown in Table 5.7.

Table 5.7 C=A^{H}A.

Kernel name	Execution time (ms)	Occupancy (%)	Achieved GFLOPS
cgemm	19.287	50	1087.36
CeAhA	2.044	75	40.357
CeAhA_upper	1.791	75	45.586

Table 5.7 shows three kernels, cgemm which is a library matrix multiplication function, CeAhA which is an optimised function which writes out the whole of C, and CeAhA_upper which is an optimised function but only writes out the upper triangular entries, which can be used for the LDLT and Cholesky factorisation.

From Table 5.7 it can be seen that an improvement in execution time can be found between the library functions and the optimised functions requiring 9.436 and 10.769 times less time to execute for CeAhA and CeAhA_upper kernels, respectively.

The decrease in FLOPS is largely as a result of the decrease in parallelism between the library algorithm and the optimised algorithm. In the current configuration, only 256 of the 1536 processing cores available (which is 16.67% of the total GPU capacity).

Furthermore, it can be seen that the writing out the entire matrix requires a further 14.13% of the required execution time. This reduction in execution time may be utilized by a Cholesky or LDLT factorisation over a factorisation which requires the entire matrix to be filled, such as LU factorisation.

5.2.6 Experiments and results

In order to compare the performance of the filters, a case study was held on a typical set of data. From this set, the performance of each algorithm was evaluated in terms of computational performance and the cancellations' effectiveness.

5.2.6.1 Cancellation effectiveness

The effectiveness of the cancellation was evaluated by analysing a typical set of recorded data (described below). The data was processed using each of the filters discussed in Section 5.2.

In some experiments, an estimate of the noise or noise floor is required. Due to the nature of real data, the noise floor is not analytically determinable. Instead, an empirical noise floor was determined. This was calculated by selecting a section of the ARD where no targets, or clutter, are present (determined by analysing the ARD produced by QR least squares) and averaging that region such that

$$A_N = \sqrt{\frac{1}{N}\sum_{i=1}^{N} z_i z_i^*} \tag{5.24}$$

where

A_N is the amplitude of the noise floor,

N is the number of bins in the cancellation region, and

z_i is the i^{th} bin used to estimate the noise floor.

The effectiveness of the two iterative techniques can be largely based on the estimate of the filter weights. Two cases were, therefore, examined. The first case was using a poor estimate which was found by injecting a zero vector as the initial estimate. The second case examined the effectiveness of the filters when a good estimate was available. This estimate was derived by applying a QR least squares to the previous CPI of data. The least squares filter is solved using QR decomposition, while the Wiener-Hoph filter is solved for via LU decomposition.

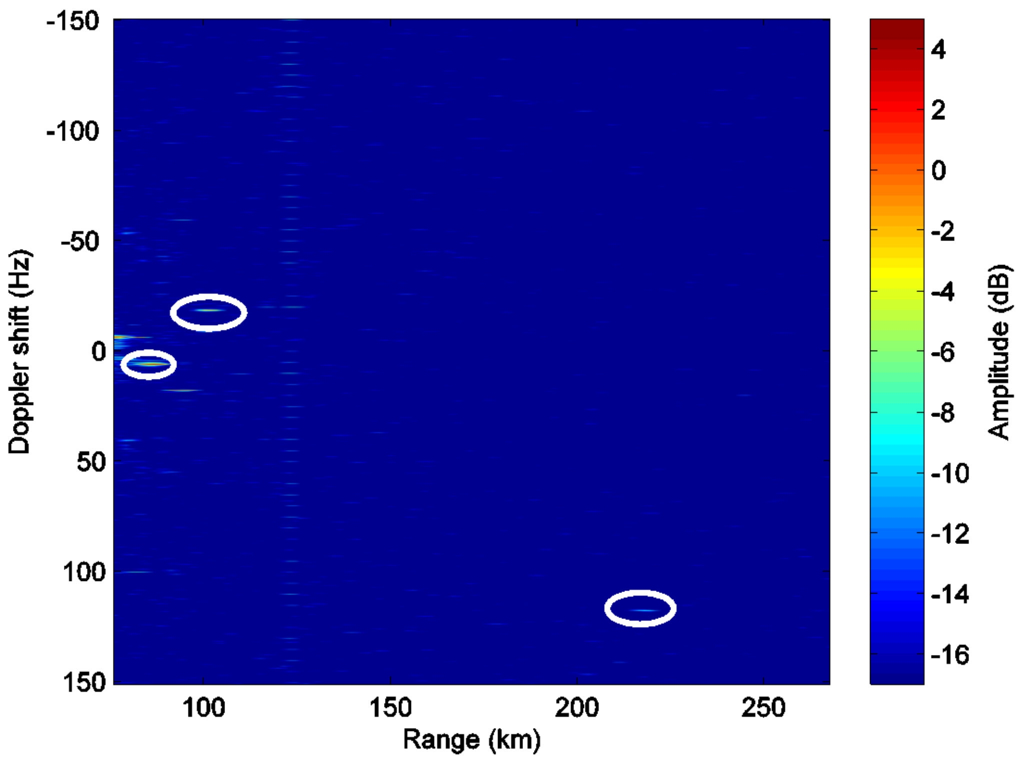


Figure 5.4 Signal space of data used for analysis.

Dataset

The dataset used for the analysis was sampled from a commercial FM radio transmitter transmitting at 93.46 MHz. The baseline distance between the transmitter and receiver was 74.46 km, and the receiver was set to receive at a sample rate of 200 kS/s.

The configuration of the filters was equivalent to that of the final implementation and is detailed in Table 5.8, unless otherwise stated. The dataset used for the analysis of the effectiveness of the cancellation filters contains three targets highlighted in Figure 5.4.

Targets 1 and 2 have low bi-static ranges and are located at a range of 87.17 km and at a Doppler shift of 5.5 Hz (Target 1), and at a range of 102.2 km and a Doppler shift of 18.5 Hz (Target 2). Target 3 is further from both the DPI and clutter and is located at a range of 216.1 km and a Doppler shift of 117.5 Hz.

Table 5.8 Cancellation filter configuration.

Parameter	**Symbol**	**Value**
CPI length	N	800×10^3
Batch Length	N'	40 000
Number of Batches	P	20
Number of bins cancelled	M	256
Number of iterations	K	30
Doppler resolution	Δ_{Dop}	0.25 Hz
Minimum range resolution	Δ_R	1.5 km

Magnitude of cancellation region bins

The magnitude of the ARD bins in cancellation region was evaluated, as this is the region where the filters have the greatest influence. The magnitude is determined by

$$RMS_c = \sqrt{\frac{1}{P}\sum_{i=1}^{P} z_i z_i^*}, \tag{5.25}$$

where z_i is the i^{th} bin in the cancellation region, and P is the number of ARD bins which form the cancellation region. The results found for the non-iterative filters are shown in Table 5.9.

Table 5.9 Magnitude of cancellation region bins for non-iterative filters.

Algorithm	Magnitude of cancellation region bins (dB)
No Cancellation	116.1
Least squares filter	−52.30
Wiener-Hoph filter	−52.30
GAL filter	56.32

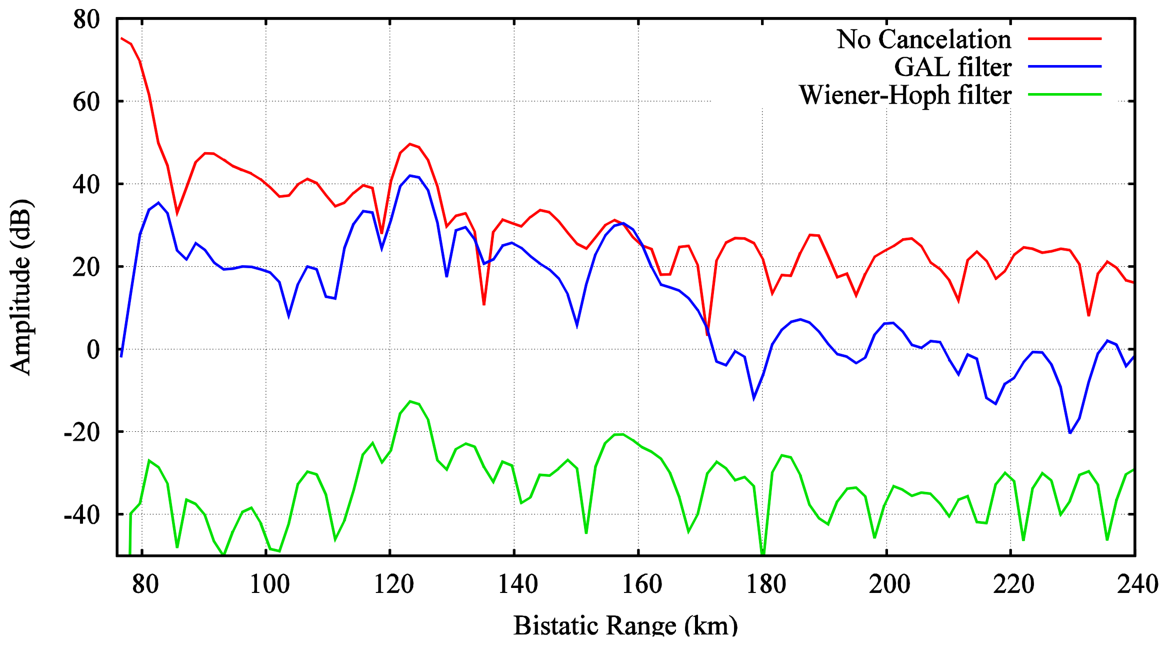


Figure 5.5 DPI and Clutter return for non-iterative cancellation filters.

As can be seen from Table 5.9, the least squares filter, which is theoretically the most numerically accurate of the methods investigated [61], performs well and reduces the clutter region by 168.38 dB. The Wiener-Hoph filter provides the same reduction of the cancellation region as the least squares filter (168.38 dB). The GAL filter did provide some reduction, however, it is significantly less than the other two filters in Table 5.9 (59.68 dB).

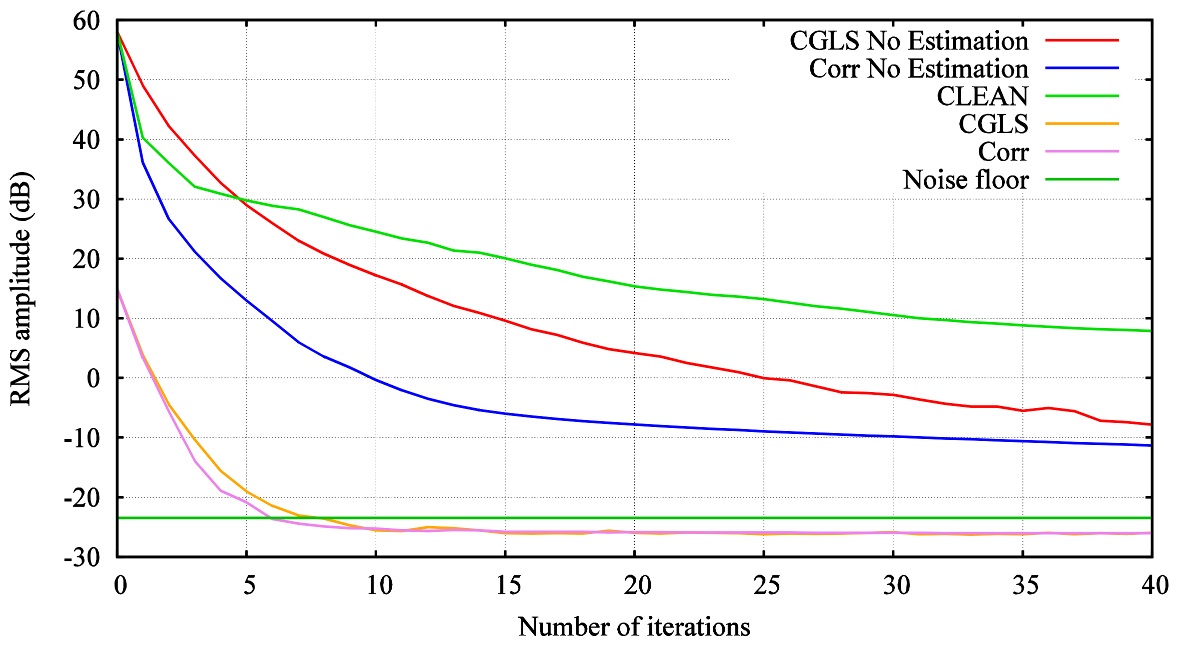


Figure 5.6 Magnitude of cancellation region bins for iterative filters.

The GAL filter's clutter region (shown in Figure 5.5), although lower than the unfiltered signal, has residual elements which are 73 dB above the noise floor. These residual elements can still result in the masking of targets. The direct path, present at 74.46 km, is shown to be supressed. However, the GAL algorithm was unable to supress some of the strong clutter returns.

The results for the iterative methods when fed a poor estimate are shown in Figure 5.6 along with the empirical noise floor. As can be seen in Figure 5.6, the correlative filter converges faster than the CGLS solution. It can be seen that the iterative filters perform far better with a good estimate, achieving a cancellation below the optimal noise floor (the noise floor after an optimal filter is used). Without an estimate after 40 iterations, the cancellation region does not reach the optimal noise floor for either iterative filter. When a good estimate is provided, the correlative filter takes 8 iterations to achieve an RMS cancellation region of 1 dB short of the optimal, while CGLS takes 10 iterations.

In the first iteration, the CLEAN algorithm provided good cancellation, but the cancellation subsequent iterations diminished. This results from the low number of components that CLEAN algorithm cancels each iteration. As such, when there is a single strong component (such as DPI) it effectively cancels it. However, when there are multiple components with similar amplitudes (such as clutter), cancelling a single component has a diminished effect.

Euclidian norm of surveillance channel

In order to evaluate filter performance based on the LS filter's cost function (5.12), the norm of the filtered surveillance channel is evaluated. The norm $\|x\|$, is calculated as

$$\|x\|^2 = \sum_{i=1}^{N} x_i x_i^* \tag{5.26}$$

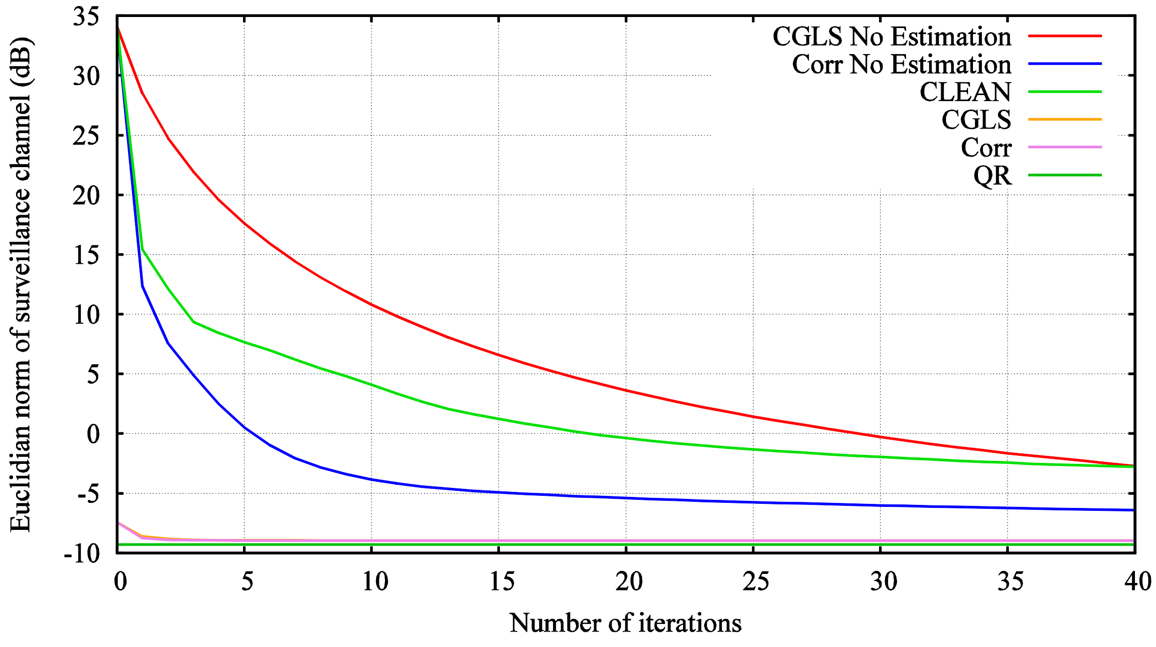


Figure 5.7 Norm of surveillance channel after iterative filters.

The resultant norms after the non-iterative filters were applied are shown in Table 5.10.

Table 5.10 Norm of surveillance channel for non-iterative cancellation filters.

Algorithm	**Norm of surveillance channel (dB)**
No Cancellation	34.05
Least squares filter	−9.30
Wiener-Hoph filter	−9.30
Gal filter	8.787

For the non-iterative methods, a similar trend is found for the surveillance channel norm as with the cancellation region. The least squares and Wiener-Hoph filter both perform identically, and far better than the GAL filter. Their similarity is not unexpected due to the similarities between the filter equations. The equation for Wiener-Hoph filter, (5.9) , is expanded to,

$$A^H A\hat{x} = A^H s. \tag{5.27}$$

This is then simplified to

$$A\hat{x} - s = 0. \tag{5.28}$$

The resemblance to the least squares filter is therefore clear, by comparing (5.12) and (5.28). As such, it can be expected that the filters will have approximately identical results, with differences being caused primarily by numerical inaccuracies.

The results for the iterative filters are shown in Figure 5.7. The resultant Euclidian norm of the surveillance channel after the iterative filters are applied, follows the trends from Figure 5.6. Here the Correlative filter continues to perform better than the CGLS solution. The improvement, however, is only significant when there is poor estimation. The clean algorithm displayed performance in between the CGLS and correlative algorithms without estimation. Once again initially converging quickly, but with quickly diminishing returns.

Noise floor

An empirical estimation of the noise floor clutter was taken, as given by (5.25). The region used for estimation was set away from targets and clutter. The noise floor was analysed as it gives a good indication of the effect of sidelobes on the signal space as well as the direct effect DPI and clutter cancellation has. The estimated noise floors achieved for each of the non-iterative solvers are shown in Table 5.11.

The empirical noise floor for the iterative cancellation filters is shown in Figure 5.8. The results for the empirical noise floor displays the same trends as seen in the previous two experiments. The optimal filters result in a 48.11 dB reduction in the noise floor, while the GAL filter results in a 26.93 dB reduction.

Table 5.11 Estimated noise floor after non-iterative cancellation filters.

Algorithm	Empirical noise floor (dB)
No cancellation filter	24.63
Least squares filter	−23.48
Wiener-Hoph filter	−23.48
GAL filter	−2.308

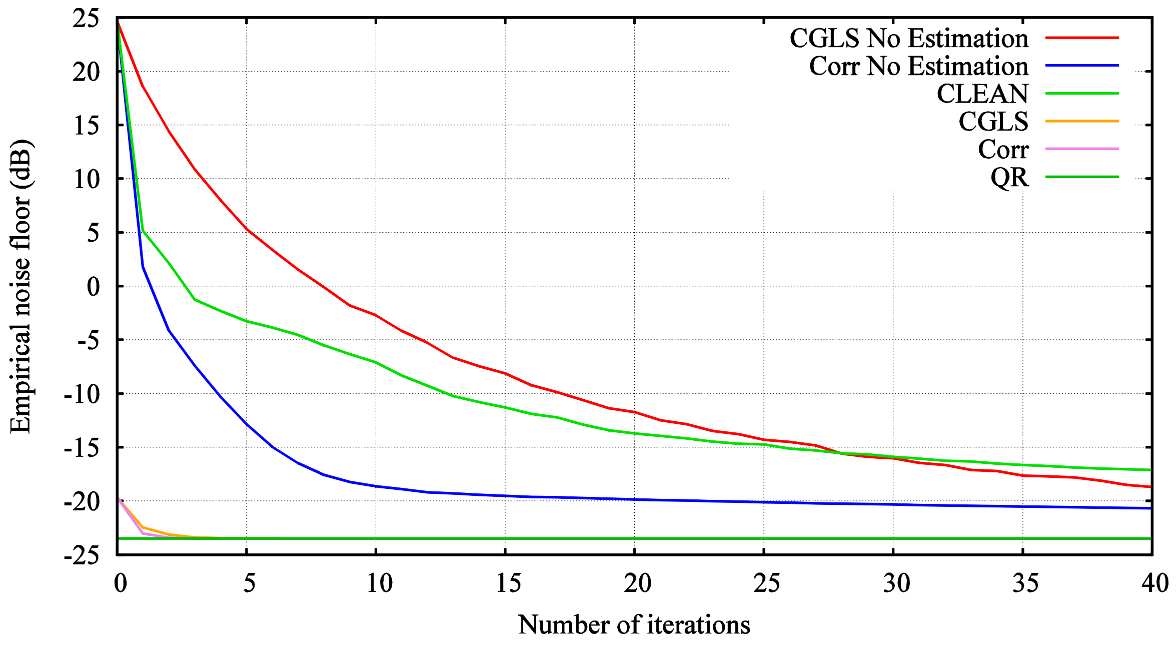


Figure 5.8 Empirical noise floor of iterative cancellation filters.

The empirical noise floor found after applying the iterative filters follows the trends found in the above cases. This demonstrates how the DPI and clutter's sidelobes contribute significantly to the noise floor.

Once again, the correlative filter outperforms the CGLS filter, but the difference is not significant when a good estimate is provided. The difference between the optimal, LS solution and the iterative solutions is smaller than with the cancellation region and noise. The Correlative algorithm requires 5 iterations to come within 0.05 dB of the optimal noise floor while the CGLS algorithm requires 7. The CLEAN algorithm shows similar trends as the first two metrics, performing in between the CGLS and correlative filters without estimation.

SNR of target

The SNR was calculated by taking the magnitude of Target 3's bin and dividing it by the empirical noise floor. The empirical SNR for the non-iterative methods is shown in Table 5.12.

Table 5.12 Empirical SNR of non-iterative cancellation filters.

Algorithm	Empirical SNR (dB)
No cancellation filter	-8.616
Least squares filter	18.76
Wiener-Hoph filter	18.76
GAL filter	6.166

In Table 5.12, it can be seen that the empirical SNR of all the cancellation filters results in a positive SNR. A clear target, however, was only exposed when the Weiner-Hoph and the Least-Squares filters were used. The components which make up the empirical SNR estimate are

$$\mathrm{SNR}_{estimate} = \frac{S_{emperical}}{N_{empirical}} = \frac{S_{actual} + n(t)}{N_{empirical}} \tag{5.29}$$

where

$S_{empirical}$ is the power of the sampled signal,

$N_{empirical}$ is the power of the sampled noise,

S_{actual} is the power of the actual signal, and

$n(t)$ is a noise signal.

As can be seen in (5.30), the empirical signal power is the sum of the signal and noise. As such, the empirical SNR is only a good approximation of the actual SNR when the actual SNR is large. In situations where the SNR is low, such as the case without filtering or the GAL filter, the empirical SNR is mainly a function of the fluctuations in $n(t)$ and, as such, a poor indication of the actual SNR. Furthermore, the vast differences in the empirical noise floor support the hypothesis that the actual SNR without cancellation, and using the GAL filter, is far lower than the empirical SNR would estimate.

The practical implications for the reduced performance achieved with the GAL filter are that all but the strongest targets will be obscured. Due to this lack in performance, the GAL filter was no longer considered a viable alternative, and was not pursued any further.

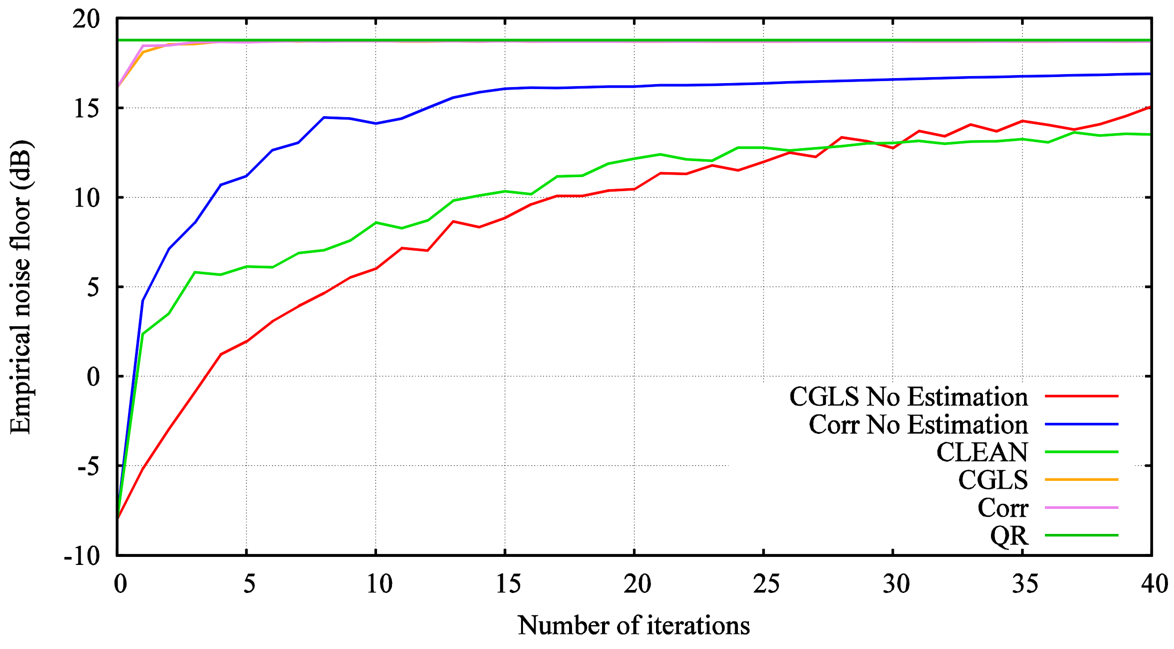


Figure 5.9 Empirical SNR for iterative cancellation.

The empirical SNR achieved by the iterative filters is shown in Figure 5.9. The Correlative filter is shown to achieve a higher SNR than the equivalent CGLS filter. While the filter achieves the SNR of the QR solution when a good estimate is provided there is still a 3.32 dB deficit after 40 iterations between the Correlative filter and the QR filter when no estimate is given. The CLEAN initially performs better than the CGLS algorithm without estimation, but after 32 iterations, the CGLS filter outperforms the CLEAN filter.

Discussion of results

It can be seen from the results above that, in practice, the difference in cancellation is insignificant between the least squares and Wiener-Hoph filters. It can also be seen that when an iterative approach is adopted, and a poor estimate is used, sub-optimal cancellation is achieved and detection is impaired. Hence, it is evident that iterative techniques are only a good design choice when a good estimate is available. This is especially apparent where processing is restricted and the number of iterations would be limited.

The correlative and CGLS filters, with a good estimate reached within 3 dB of the optimal solution within 3 iterations co-insides with results found in literature [13]. This provides a solution that would require far less computation than a full solver.

The CLEAN filter performed well when there was significant DPI but poorly once some cancellation was applied. As such, it did not provide sufficient cancellation and could not effectively use prior cancellation. It was therefore, found that the CLEAN filter was not a suitable design choice and it was no longer pursued as a design choice. The GAL filter displayed inadequate cancellation of clutter and was excluded as a design option. This could be a result of the tuning sensitivity of adaptive filters [33], however, more successful parameters were not found.

Other literature [9] [33] has also found that the least square or Wiener filters are most successful, along with some well-tuned adaptive filters. They also found that the majority of adaptive filters are less than optimal, which coincides with the finding here.

5.2.6.2 Computational performance

The computational performance of each filter was measured by running the filter over an 11 minute recording, which results in 165 CPIs where a CPI is 800×10^3 samples. Each CPI was processed in 20 batches, resulting in a filter length of 40×10^3 samples. The execution of the test procedure is detailed in Figure 5.10.

The mobile platform used the Intel MKL library [51] for CPU BLAS, solver, and DFT functions. The MKL library is a highly optimised and parallelized library for Intel processors. The matrix-matrix and matrix-vector library functions were sourced from cuBLAS [60]. cuBLAS is an optimised library which implements BLAS functions on CUDA GPU kernels. The solver on the GPU was implemented via cuSolver [62]; a library for implementing linear algebra solvers via CUDA GPU kernels. GPU based DFTs were implemented using the cuFFT library [63]; a library for implementing optimized DFTs as CUDA GPU kernels.

The embedded system used the same libraries for the GPU implementations. As the CPU was not an Intel device or based on the x86 instruction set, the MKL libraries could not be used. Instead, the DFTs were implemented via FFTW [64]; a fast implementation of DFTs. The BLAS and solver functions wee sourced from ATLAS [65]; a library of linear algebra functions which is tuned/optimised for each device.

The timings were recorded via a system clock and the average time was taken by dividing the total time by the number of CPIs. The filters were set up as described in Table 5.8, unless otherwise stated. Hence, the average times are for each CPI or 20 filter runs.

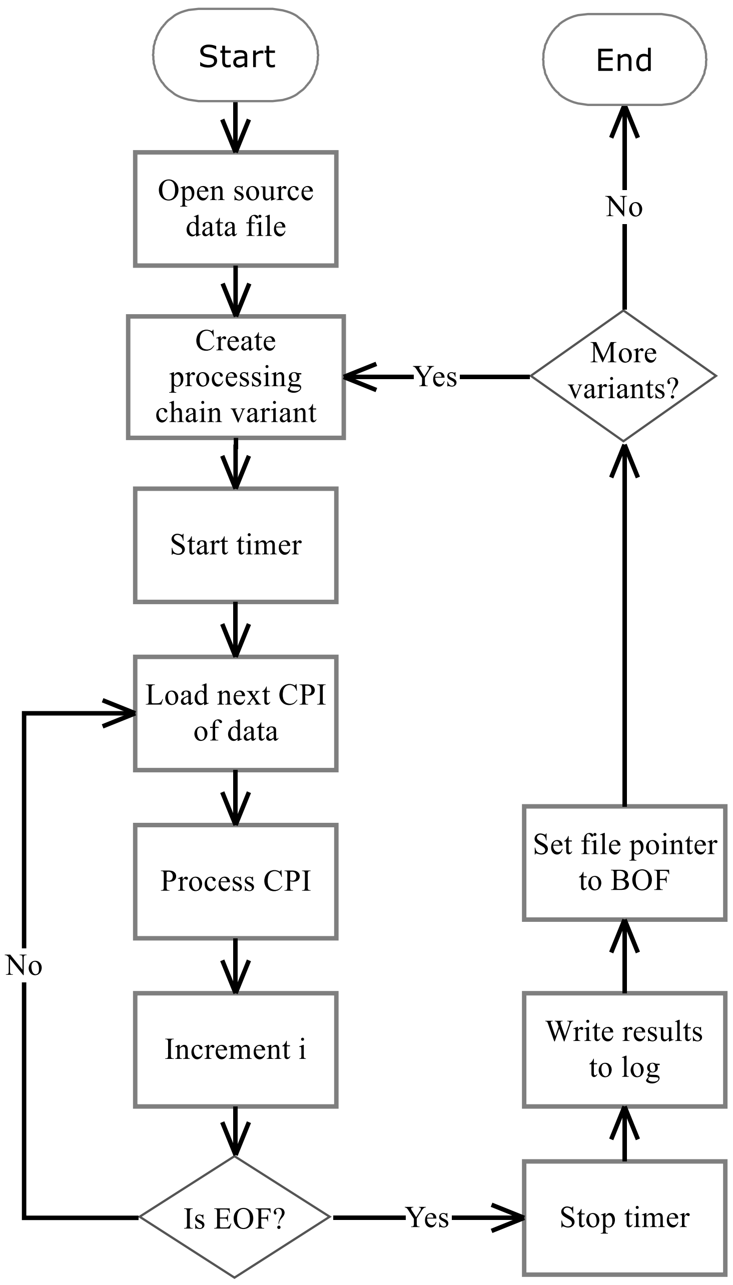


Figure 5.10 Testing procedure for compute performance.

Average execution time for CGLS

The achieved average execution time for the CGLS cancellation filter implemented on the mobile platform when K was 15 is given in Figure 5.11.

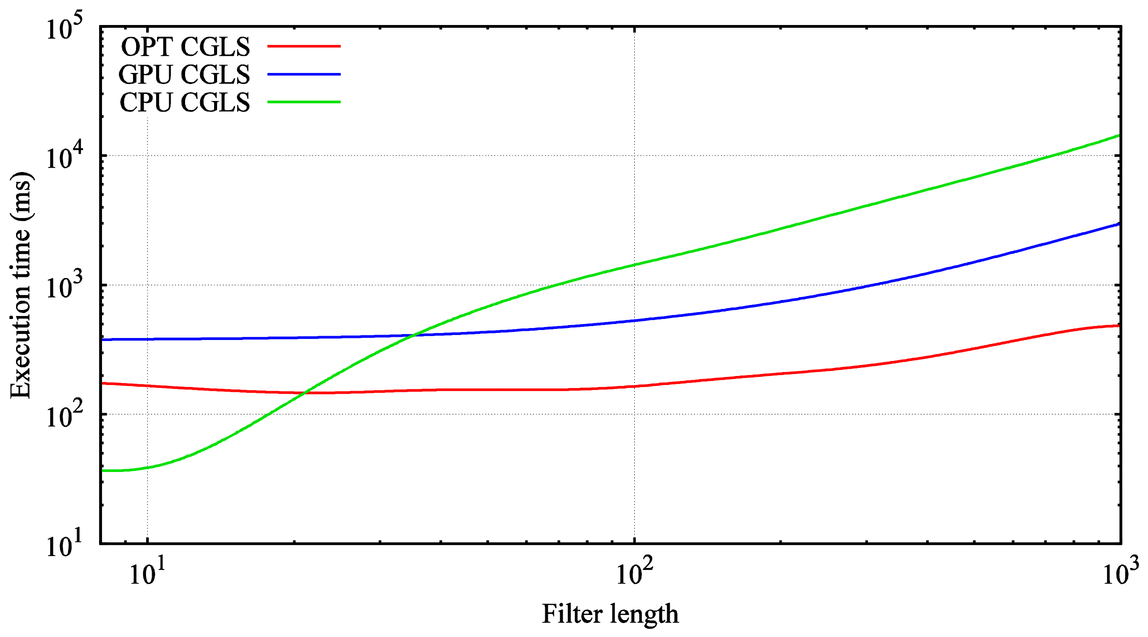


Figure 5.11 Average execution time for CGLS filter of varying filter size.

As can be seen from Figure 5.11, the optimised GPU filter outperforms the CPU filter for $M > 21$, and continues to outperform the library based GPU filter for the entire range of the test. At the design size of $M = 256$, the optimised filter had an average execution time of 223.2 ms which is 3.91 time faster than the library GPU based filter (which took an average of 871.9 ms to execute) and 15.69 times faster than the CPU filter (which took an average of 3 502 ms to execute).

The optimised filter here used the architecturally optimised matrix-vector operations as discussed in Section 5.2.5.2. The GPU implementation used the equivalent functions found in CUDA BLAS, and the CPU implementation used equivalent functions found in MKL BLAS.

Average execution time for LU

The average execution times found for each of the LU cancellation filters when implemented on the mobile platform are shown in Figure 5.12. In this figure the optimised LU filter and both hybrid filters use the matrix-matrix and matrix-vector algorithms as discussed in Section 5.2.5.2. The hybrid algorithms transfer the data back to the host CPU and solve the reduced problem using the MKL libraries. The CPU algorithm uses the algorithmic optimisation discussed in Section 5.2.5.1, and solves the reduced problem using the MKL libraries. The GPU LU cancellation filter and optimised cancellation filters both solve the reduced system on the GPU using the cuSolver library. The GPU LU cancellation filter implements all other matrix-vector and matrix-matrix functions from the CUDA BLAS library.

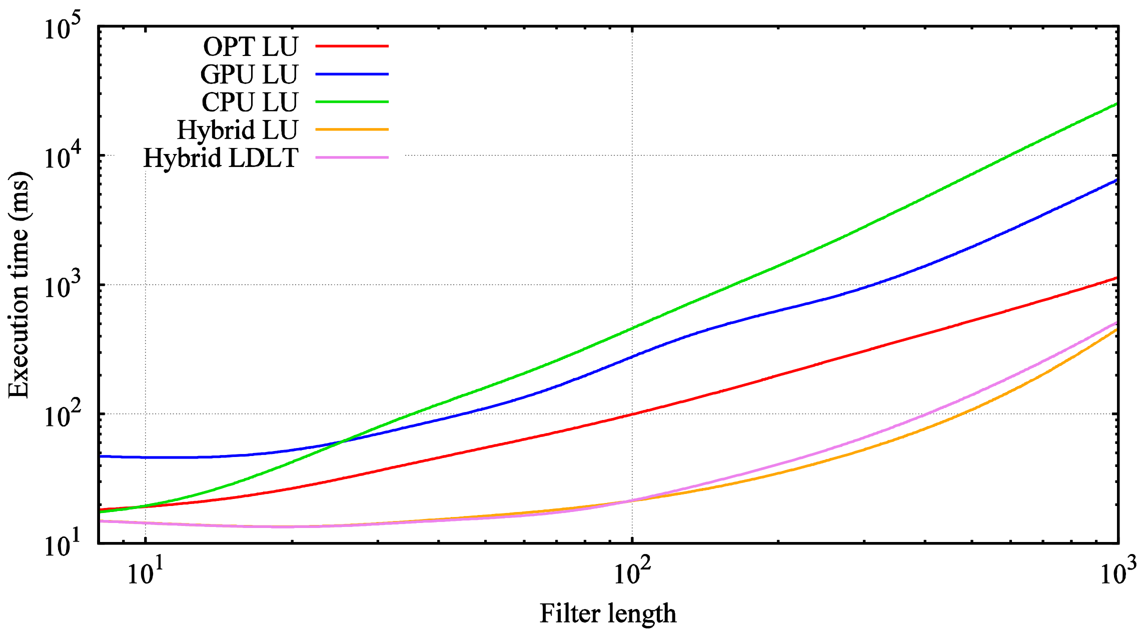


Figure 5.12 Average execution time for LU cancellation filters.

As can be seen from Figure 5.12, the LDLT generally filter takes longer to solve, even though the algorithm takes advantage of the Hermitian form of C. For this reason, it was not investigated further, although, for much larger M it may be an attractive choice.

It can be seen that as filter sizes become larger, the deficit between the hybrid and optimised LU cancellation filters diminishes. This demonstrates a typical trend in GPU computing; performing comparatively better for large problem sets [55].

At the design size of $M = 256$, the hybrid LU cancellation filter performs best requiring an average of 44.60 ms to execute while the optimised implementation took 260.0 ms (5.83

times slower than the hybrid LU cancellation filter). The GPU LU cancellation filter took 797.9 ms (17.89 times slower than the hybrid LU cancellation filter), and the CPU filter took an average of 2 124 ms to execute (47.63 times slower than the hybrid LU solver).

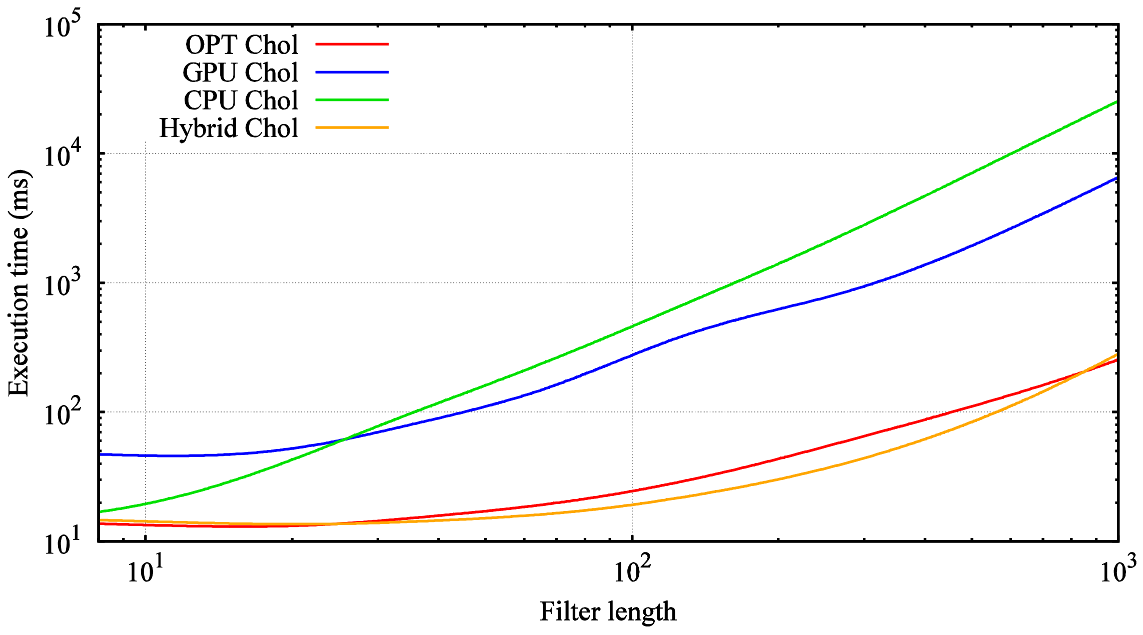


Figure 5.13 Average execution time for Cholesky cancellation filters.

Average execution time for Cholesky

The average execution times found for the implementations of the Cholesky cancellation filter are shown in Figure 5.13.The CPU implementation implements the $A^H A$ algorithm discussed in Section 5.2.5.1, and matrix-vector and solver functions found in the MKL library. The GPU Cholesky cancellation filter uses matrix-matrix and matrix-vector functions from the CUDA BLAS library, and solver functions from the cuSolver library. The optimised Cholesky cancellation filter solves the reduced system using the cuSolver library, but uses the optimised functions discussed in Section 5.2.5.2.

The hybrid Cholesky filter proved to be the fastest over the majority of the test range. An interesting observation that is apparent in Figure 5.13, is that the difference between the hybrid and optimal implementations is minimal. Also where $M > 804$ the optimal implementation executes more quickly. This is as a result of the Cholesky algorithm not requiring pivoting to obtain a factorisation and, as such, is more parallelisable and subsequently better suited to the GPU.

At the design filter size of $M = 256$, the hybrid implementation takes an average of 37.64 ms to execute, while the optimised implementation takes 55.69 ms. The GPU implementation takes 791.1 ms (which is 21.01 times slower than the hybrid Cholesky filter), while the CPU filter took an average of 2163 ms to execute (which is 56.74 times slower than the hybrid Cholesky filter).

Average execution time for non-optimised cancellation filters

The last two filter categories analysed are ones which could not be optimised via the methods described in Section 0. The first is the correlative filter, which was implemented on the GPU using the CUDA BLAS libraries, and on the CPU using MKL libraries.

The second method implemented was the QR cancellation filter which was implemented on the GPU via the cuSolver library, and on the CPU via the MKL library.

The average execution time achieved when running the QR and correlative cancellation algorithms on the mobile platform is shown in Figure 5.14. In Figure 5.14 it can be seen that the CPU correlative filter executed quickest for the entire range.

An interesting observation here is that the GPU correlative filter is nearly consistently slower than the CPU implementation. This is a result of the reduction operation which is serially executed. As such, it cannot take full advantage of the GPU's parallelism. The GPU QR implementation is also slower than the CPU implementation for the majority of the test range. This demonstrates how the QR cancellation filter, in its current implementation, is not well suited for implementation on the GPU when used to solve highly rectangular problems.

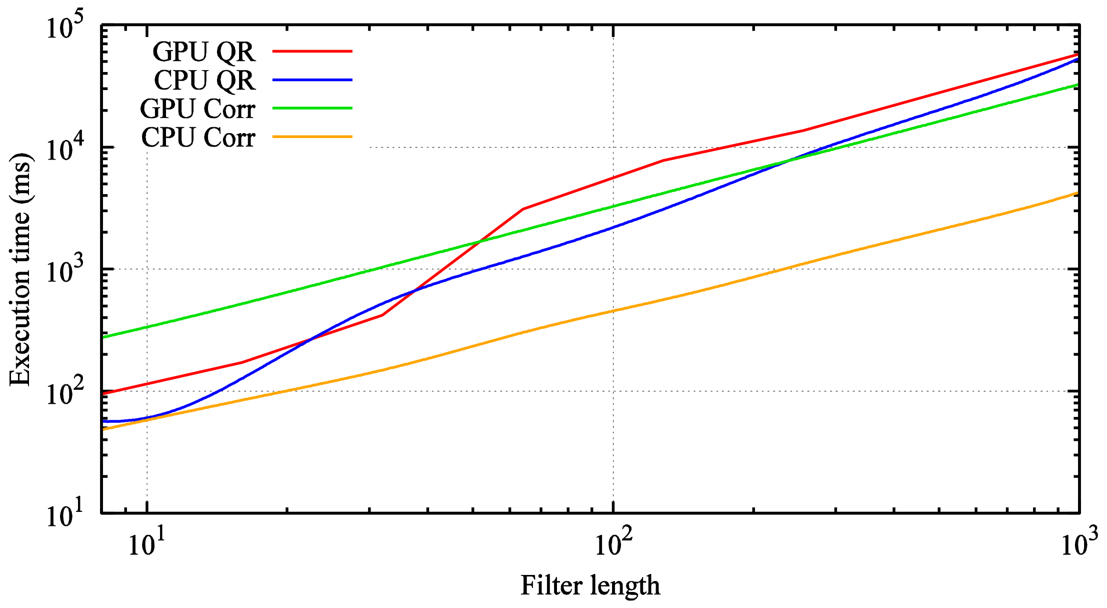


Figure 5.14 Average execution time for miscellaneous cancellation filters.

At the design size of $M = 256$, the CPU correlative filter took 1.108 s, the GPU correlative filter took 8.318 s, the CPU QR filter took 8.591 s, and the GPU QR filter took 13.605 s. As the CPI is 4 s long, none of these filters could run in real-time on the mobile platform, except for the CPU based correlative filter.

5.2.7 Conclusion

Of the methods the best performers are shown in Figure 5.15. Of these, the optimised Cholesky and hybrid LU filters perform the best. For $M \gg 1000$ it is possible that the CGLS algorithm will be the best performer, for the fixed value of K, however that is out of the scope of this study.

While the Cholesky filters are the fastest, they do not provide a numerically stable solution, and so are not suitable for all applications. It was, however, found that the algorithm used in the hybrid Cholesky filter was more stable than the algorithm used in the GPU Cholesky solver. If the GPU Cholesky solver was used on a semi-definite/indefinite matrix, the algorithm would return NaN, The CPU implementation, however, returns poorly estimated filter weights.

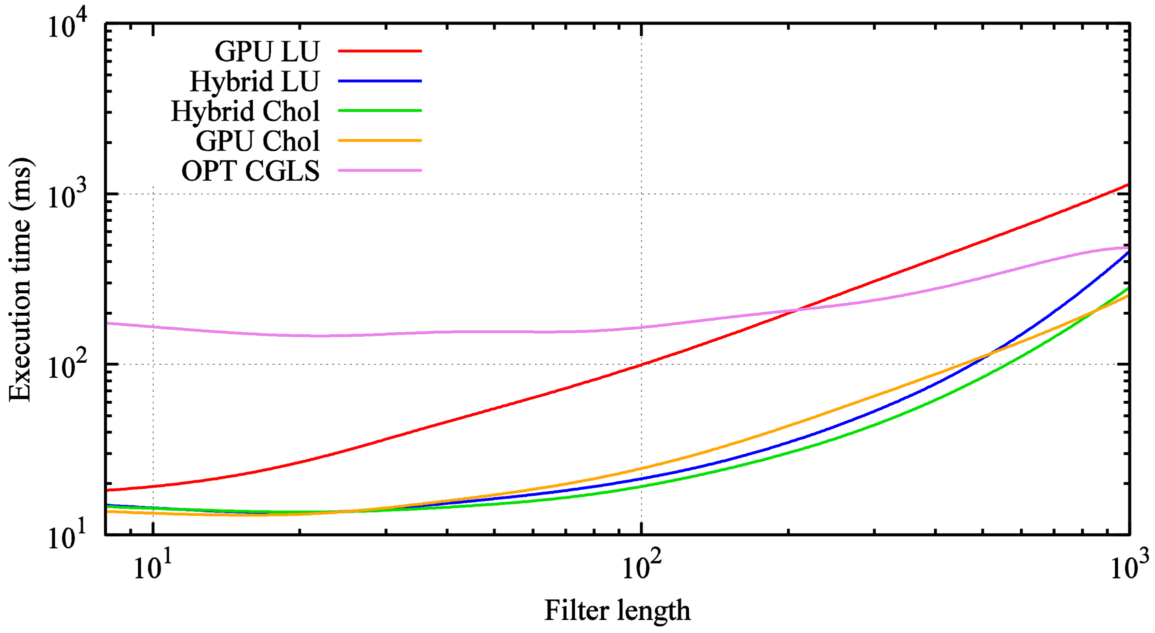


Figure 5.15 Average execution time of best performing cancellation filters.

Nonetheless, it was found that the hybrid LU filter was able to provide the best performance while allowing for equally efficient cancellation as for the other non-iterative algorithms.

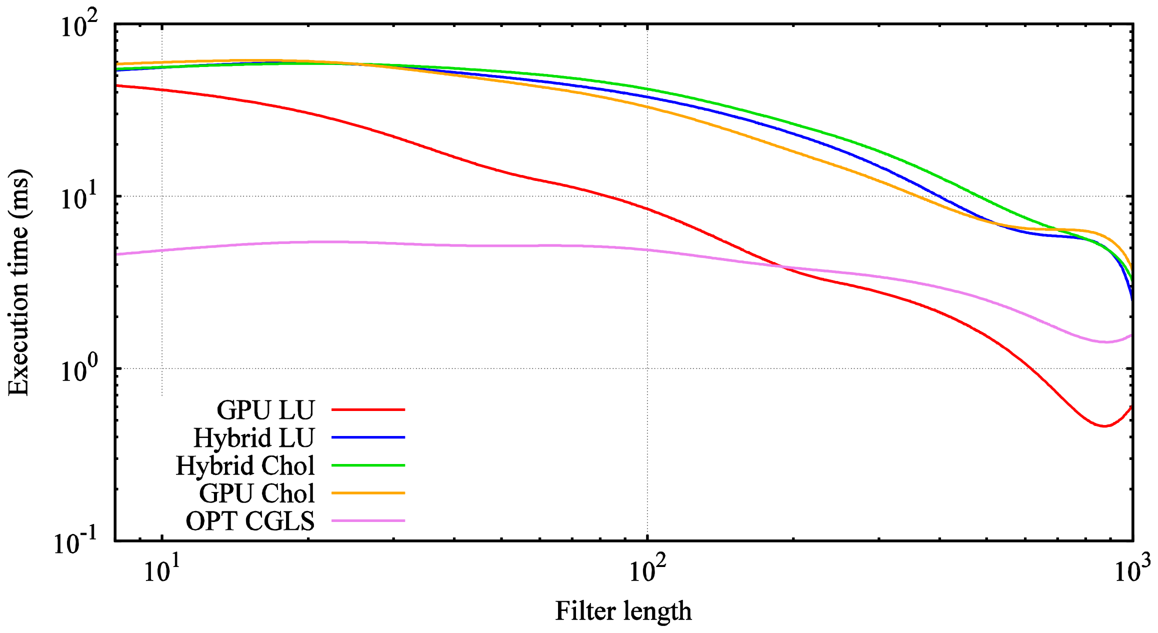


Figure 5.16 Average throughput of best performing cancellation filters.

The average throughput found for the best performing filters is shown in Figure 5.16. These are the same results as previously presented, but they are presented in this format for easier comparison. Figure 5.16 depicts similar findings to those in Figure 5.15. The hybrid LU filter is still the best performing stable filter, with an average throughput of 17.94 MS/s at the design size of $M = 256$. The hybrid Cholesky filter was the best overall with average throughput of 21.25 MS/s

It was found that, when using library functions, the sub-optimal CGLS algorithm out performs the other filters, these results were also in literature [9] [13]. However, once the matrix operations were optimised, the Wiener filters ended up being the fastest solution, as well as being more accurate than the CGLS algorithm.

5.3 MATCHED FILTER

The matched filter combines pulse compression and Doppler filtering into a single processing block. The matched filter is depicted within the processing chain in Figure 5.1. The matched filter's implementation becomes computationally intensive as the reference signal must be correlated with the surveillance signal in both time and frequency.

The matched filter converts the reference and surveillance vectors of length N into a Range-Doppler map with N_{Dop} Doppler indices and N_R range indices.

5.3.1 Time correlation

Time correlation [38] correlates the signal in time. However, in order to reproduce the necessary Doppler spectrum, the surveillance signal must be correlated with a correctly Doppler modulated signal for each Doppler index [12]. The Doppler modulated reference vector is given by

$$r_i'(f) = r_i e^{j\frac{if}{F_s}} \tag{5.30}$$

where

F_s is the sampling frequency.

The correlation can then be performed by a series of dot products between a shifted $r'(f)$ and the surveillance channel, i.e. each Range-Doppler bin if formed by

$$RD_{ij} = s \cdot r'_{2-i:2-i+N}(f_j). \tag{5.31}$$

Hence, assuming $r'(f_j)$ is pre-calculated, the creation of each bin has a time cost of $2N$, and a FLOP count of $8N$.

Alternatively, correlation can be performed in the frequency domain, occasionally referred to as fast convolution [66]. This method performs correlation in the time domain by exploiting the time-frequency duality between multiplication and convolution (i.e. time convolution is performed by multiplying in the frequency domain).

Each signal is taken to the frequency domain using a DFT. The surveillance channel needs only to be converted once while each modified reference vector (corresponding to each Doppler index) must be converted to the frequency domain such that,

$$S = \mathcal{F}\{s\}$$
$$\widehat{R_\iota}(\omega) = \mathcal{F}\{\widehat{r_\iota}\}.$$

It is important to note that, in the form above, circular convolution is performed. However, as conventional convolution is required, the reference and surveillance vectors should be zero-padded before converting to the frequency domain, such that the new vectors are $N + N_R - 1$ elements long.

Each modified surveillance channel, now in the frequency domain conjugated and multiplied with the reference channel (also in the frequency domain).

$$Y_i = S\widehat{R_\iota}^*(-\omega) \tag{5.32}$$

Once the multiplication is performed, all the resultant vectors are brought back to the time domain and the relevant elements are then kept as the elements of the ARD map.

The algorithm requires $2N_{Dop} + 1$ DFTs (including conversion back to the time domain) on $N + N_R - 1$ element vectors, which will be denoted as N_{RN}. An additional $N_{Dop}N_{RN}$

complex operations are required to perform (5.32). If it is assumed that an FFT is used to perform the DFT, and that the time cost of an FFT for an N element vector is $2N\log_2(N)$, the time cost for the matched filter is $N_{RN}N_{Dop} + N_{RN}\left(4N_{Dop} + 2\right)\log_2(N_{RN})$.

5.3.2 Frequency correlation

The next option is to perform correlation in frequency [5, 12]. In this approach the time-frequency duality between multiplication and convolution is once again exploited to perform correlation. In this case the correlation takes place in the frequency domain, i.e. over the Doppler dimension.

Each range index is formed by multiplying the modified surveillance channel (in this case it is just delayed) with the reference channel such that

$$y_i = r^*\hat{s}_i \tag{5.33}$$

The Doppler index of the ARD is then created by using a DFT to transform y_i to the frequency domain. Equation (5.33) requires N multiplications, and each DFT requires $2N\log_2(N)$ operations. Therefore, the application of the matched filter has a time cost of $N_R[N + 2N\log_2(N)]$.

Table 5.13 Algorithmic complexity of matched filters.

Matched Filter	Time cost	FLOP count
Time-domain time correlation (TDTC)	$2N_RN_{DOP}N$	$8N_RN_{DOP}N$
Frequency-domain time correlation (FDTC)	$N_{RN}N_{Dop} + 2N_{RN}(2N + 1)\log_2(N_{RN})$	$6N_{RN}N_{Dop} + 8N_{RN}\left(2N_{Dop} + 1\right)\log_2(N_{RN})$
Frequency-domain correlation (FDC)	$N_RN[1 + 2log_2(N)]$	$N_RN[6 + 8log_2(N)]$

The algorithmic cost for each matched filter is shown in Table 5.13. It can be seen from Table 5.13 that the optimal selection for the matched filter algorithm depends on the problem size. It has been proposed [13] that FDTC is a more suitable option when $N_R > N_{Dop}$. This is supported by the results in Table 5.13. The minimum N_R: N_{Dop} ratio required for FDTC to have less computational complexity than FDC is shown in Figure 5.17.

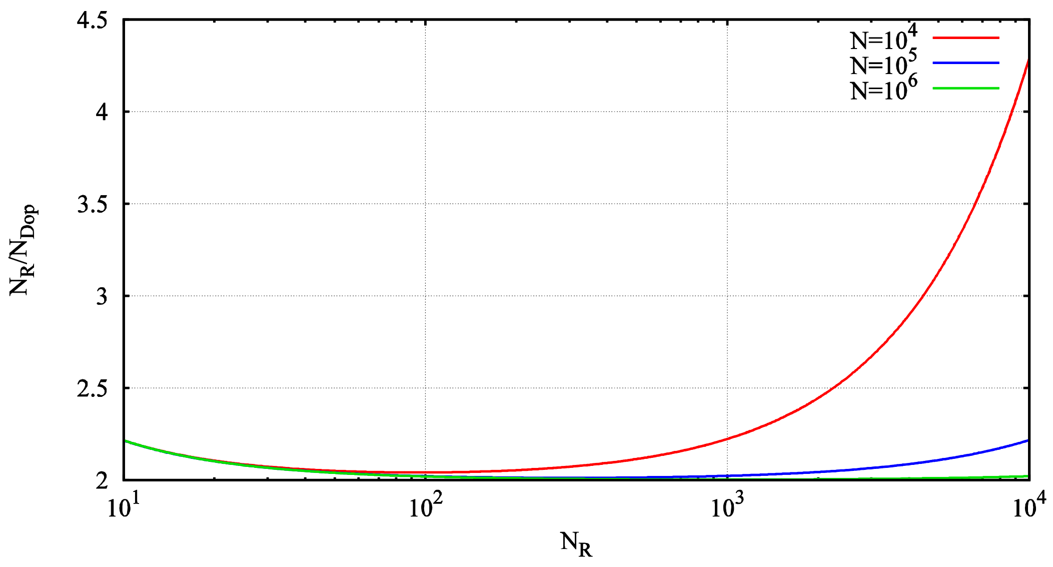


Figure 5.17 N_R to N_{Dop} ratio required for FDTC to have less computational complexity than FDC.

From Figure 5.17 it can be seen that for most practical ARD dimensions, FDTC is a suitable choice when $N_R > 2.5N_{Dop}$. For TDFC to have a lower time cost than FDC $N_R(1 + 2\log_2 N) > N_{Dop}(10 + 8\log_2 N)$.

5.3.3 Reduced computation filters

A number of matched filter approaches which exhibit reduced computational load have been explored in literature, such as the Batches algorithm [67]. However, it has been found [11] that, in order for these methods to remain accurate, they must be carefully tuned. These methods are therefore undesirable, due to this unreliability, and are not further considered.

5.3.4 Matched filter performance

The performance of the matched filter was evaluated using FDC. This was chosen, as the low bandwidth FM signals used as the transmissions for this study result in a much finer

Doppler resolution than range resolution, and as such $N_{Dop} \gg N_R$. This lends itself well to the FDC.

The implemented FDC matched filter was implemented on the CPU and GPU and configured with $N_{Dop} = 1207$ and $N_R = 150$. The filter was run over an 11 minute data set using the procedure detailed in Figure 5.10.

The matched filters were run starting with a CPI length of 8 in powers of 2 until 2^{24}. The embedded system only ran to 2^{23} as it was limited by available memory. The FFTs were implemented on the mobile host using the MKL library and FFTW was used on the embedded CPU. The FFTs on both GPUs were taken from the cuFFT library.

The average execution time is shown in Figure 5.18. Here it can be seen that the mobile CPU implementation has a lower execution time for $N < 7060$, while the mobile GPU implementation executes, on average, faster for data lengths greater than 7060. The embedded GPU requires a CPI length greater than 8656 to outperform the embedded CPU.

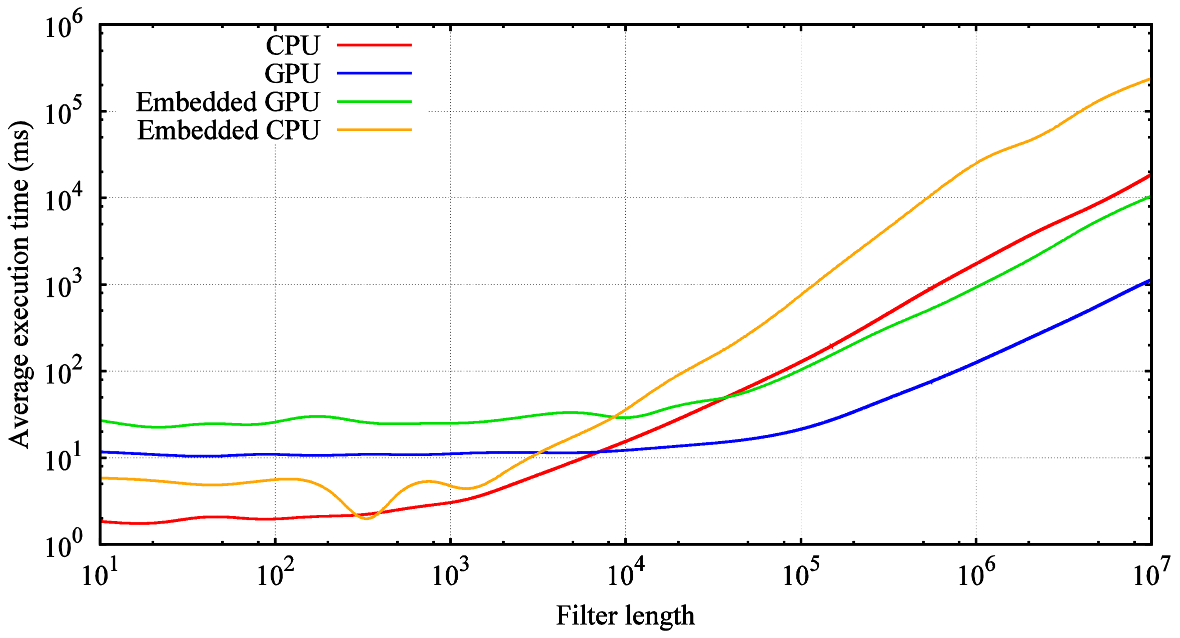


Figure 5.18 Execution time of matched filters.

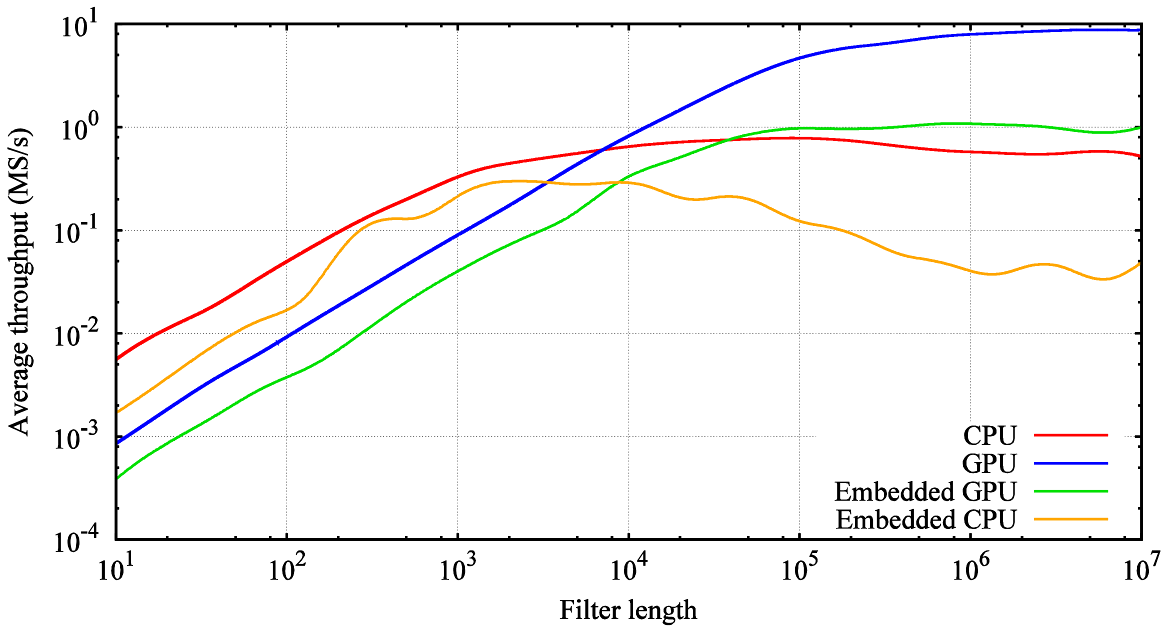


Figure 5.19 Average throughput of matched filters.

The average throughput of the matched filters is shown in Figure 5.19. As can be seen, the filters require a minimum length to achieve a high throughput, and, once a peak is reached, average throughput gradually decreases as the data length increases. However, this trend is not found in the embedded CPU implementation, as after the peak is reached, performance decreases rapidly.

For the mobile CPU implementation, the peak was reached when $N = 91.70 \times 10^3$ and with a peak throughput of 782.9 kS/s. The mobile GPU implementation did not reach its maximum throughput in the scope of the experiment. The maximum length tested was $2^{24} \approx 16.78 \times 10^6$ which relates to a CPI of approximately 84 s. Using such long CPI lengths (assuming the sampling rate is near the Nyquist rate) cause range and Doppler gate mitigation which have the effect of mitigating processing gain by spreading the received reflection power over multiple bins. As such, it is not practically valuable to investigate the performance of the filters any further.

Nevertheless, the mobile GPU implementation achieved its maximum throughput at $N = 2^{24}$ which was 11.57 MS/s. At a more suitable CPI of 4 s (800×10^3 samples) the mobile implementations were re-run. The mobile GPU implementation could manage an average throughput of 7.764 MS/s, while the CPU implementation achieved 589 kS/s. At typical FM passive radar matched filter sizes, the mobile GPU implementation achieved 13.18 times the throughput of the CPU implementation.

The embedded GPU implementation peaked at $N = 800 \times 10^3$ with a throughput of 1.084 MS/s. The embedded CPU reached a maximum throughput at $N = 2270$ with rate of 299.1 kS/s. The embedded GPUs maximum throughput was only 3.62 times higher than the maximum of the embedded CPU implementation. At a conventional CPI for a real system of $N = 800 \times 10^3$, the throughput of the embedded is 24.08 times faster than the embedded CPU, which had an average throughput of 45.02 kS/s.

5.4 TARGET DETECTION AND EXTRACTION

The ARD produced by the matched filter then needs to be processed to determine whether a bin contains interference or interference as well as a reflection from a target.

5.4.1 Detection

Automated detection was performed using a CFAR algorithm as described in section 2.6.3. The censored mean CFAR [21] was used as it provides some tolerance for intra-target interference. The censored mean CFAR creates the interference statistic by sorting the interference cell by their magnitude, and excluding the N_C largest interference samples. The assumption is that the largest interference samples are the most likely to contain a reflection from a target, and should, therefore, be excluded from the estimate.

The drawback with this method is that when the number of interference samples containing reflections from targets is less than N_c the CFAR loss (i.e. the loss in detection probability as a result of the threshold set by the CFAR algorithm being higher than the Neyman-Person bound for a specific PFA) is higher than the equivalent ca-CFAR [21]. This issue results from the interference statistic being estimated with fewer interference samples.

It is possible to determine the PFA and P_D for a given threshold, assuming that the noise in the system is AWGN. The PFA for a given CFAR constant (α) under these assumptions is given for censored CFAR as

$$P_{FA} = \prod_{i=1}^{N-N_C} \gamma_i(\alpha) \tag{5.34}$$

where

$$\gamma_i(\alpha) = \frac{u_i}{u_i + \alpha} \tag{5.35}$$

where

$$u_i = \frac{N - i + 1}{N - N_c - i + 1}. \tag{5.36}$$

5.4.2 Target extraction

Target extraction is the process of grouping bins which pass the detector into targets and estimating their position within the signal space such that a set of targets may be passed to the state smoothing filter. This was accomplished by using the binary signal space produced by the detector. This binary signal space represents ARD bins which contain a reflection from a target with a 1 and bins with only interference as a 0.

While this signal space can be passed directly to the state smoothing filter, targets may be represented by more than one bin, and as such the system will interpret this as the existence of multiple targets in close proximity. Due to the type of target (i.e. large passenger and commercial aircraft, which typically would avoid close proximity with other aircraft), it was assumed that this would be far less likely than a single target occupying multiple adjacent bins. This spreading can be caused by a number of factors including range and Doppler gate migration, micro-Doppler effects, or a reduction in the source signal’s effective bandwidth over the CPI. The latter is a common occurrence in passive radar.

Extraction of these target bins was performed by using collation. This grouped all connected targets. This allowed targets which occupy multiple bins to be interpreted as one target. It also, however, means that targets in close proximity will be interpreted as a single target. A connected components labelling algorithm was used to group any positive bins with 8 degrees of connectivity together. The degrees of connectivity define which adjacent bins are considered candidates for connectivity. This is shown in Figure 5.20, which demonstrates that for a two-dimensional signal space eight degrees of connectivity defines all adjacent bins as candidates, while four degrees excludes the diagonals.

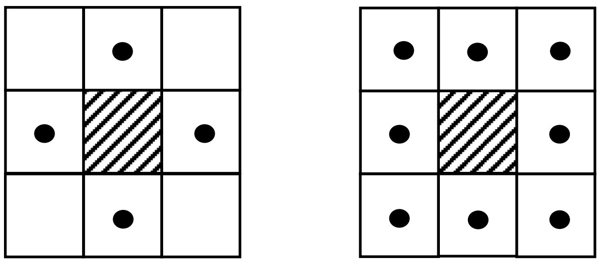


Figure 5.20 Four (left) and eight (right) degrees of connectivity.

The grouping of targets also compensates for some of the target smearing that may occur. Target smearing can easily occur over the Doppler dimension due to the fine Doppler resolution and long CPI. Target smearing is also seen in range when the instantaneous bandwidth of the transmitted signal is reduced. This can occur during speech and periods of silence. When this does occur, the range resolution increases [32] and a target can occupy multiple bins in the range dimension. This reinforces the concept that connected bins are more likely to be a single large or smeared target than it is to be multiple, closely packed, targets.

Once the bins where grouped, their bistatic range and bistatic range rate were estimated. This was performed using a simple weighted mean scheme.

The bistatic range was estimated as

$$R_i = \frac{\sum_{j=1}^{N_i} \|z_{ij}\| r_{ij}}{\sum_{j=1}^{N_i} \|z_{ij}\|} \tag{5.37}$$

where

R_i is the bistatic range of the ith group of connected detections,
N_i is the number of connect bins in the ith group of connected detections,

z_{ij} is the value of the jth bin in the ith group of connected detections, and
r_{ij} is the range of the jth bin in the ith group of connected detections.

The bistatic range rate was similarly estimated as

$$\frac{\partial R_i}{\partial t} = \frac{\sum_{j=1}^{N_i} \|z_{ij}\| d_{ij}}{\sum_{j=1}^{N_i} \|z_{ij}\|} \lambda \tag{5.38}$$

where

d_{ij} is the Doppler frequency of the jth bin in the ith group of connected detections in Hz, and

λ is the wavelength of the carrier frequency.

5.5 TARGET STATE SMOOTHING

The state smoothing filter has the task of determining which detected targets (received from the target extractor) belong to the same physical target, if any, as well as predicting and refining the position of the raw detections. Unlike the previous processing steps, the state smoothing filter is the first which concerns historical information, and operates beyond a single CPI of time [68].

Tracking filters which only use Doppler information have been suggested [68]. This is done due to the poor range resolution (and coupled accuracy) achieved with FM signals. For this implementation it was decided that while inaccurate, useful information still exists in the bistatic range measurements. Hence, the state smoothing filter incorporates this data, but with far less emphasis compared to the Doppler measurements. Other methods explored in literature include: Kalman filters [17], as well as the extended Kalman filter coupled with a particle filter [69].

5.5.1 Target model

The first step in predicting and refining a target's position, is the need to create and define a model which can predict the target's trajectory. The nature of this problem has led to many solutions, ranging from FIR filters to deep learning algorithms.

The problem with the design of complex models is that they require detailed models of the environment off of which the filter's behaviour is defined. For example, a Kalman filter requires a model of the measurement noise or error [70]. These filters have been shown to outperform simpler architectures, but only when the models are valid. The generation of accurate noise models was outside of the scope of this study, and as such filters which require noise or environment models are not considered. Machine learning algorithms are also out of the scope of this study, as the amount and diversity of data required to train a filter which could be consistently and generally used was not available.

As such, the filter model was loosely based around an alpha-beta tracking filter [71] as this could be hand-tuned and would still provide consistent and valuable results. Due to the comparatively static flight paths of commercial airlines (which were the subjects of this study) a constant velocity model was originally investigated, as it had previously been used in passive radar target tracking [68]. It was found, however, that while the trajectory of the targets may be relatively constant, the translation to bistatic range resulted in that assumption being poor.

Instead a constant acceleration model was adopted. This allowed the target movement to be a quadratic estimation This defines a target in terms of its bistatic range ($R(t)$), bistatic velocity ($R'(t)$) and bistatic acceleration ($R''(t)$). As $R''(t)$ is constant, the predicted state of the target Δt seconds into the future from a time t_k is easily found via integration such that

$$R''(t_k, \Delta t) = R''(t_k) \tag{5.39}$$

$$R'(t_k, \Delta t) = R'(t_k) + \Delta t \, R''(t_k) \tag{5.40}$$

$$R(t_k, \Delta t) = R(t_k) + \Delta t \, R'(t_k) + \Delta t^2 \, R''(t_k). \tag{5.41}$$

5.5.2 Model estimation

In order to predict the position of a target, information is required to estimate the model parameters. This data is derived from the detections sourced from the target extraction. The model parameters could be set by the latest detection such that $R(t_k) = R_i$, $R'(t_k) = \frac{\partial R_i}{\partial t}$, and $R''(t_k) = 0$. However, this strategy would discard the historical information which is already known about the target. In order to incorporate this information, the form of an alpha-beta filter was adapted. At time step k the target's state is updated by defining

$$\Delta t = t_k - t_{k-1} \tag{5.42}$$

where

t_{k-1} is the time at which the target's state was last updated.

The targets current state is then estimated using

$$R_{est} = R(t_{k-1}, \Delta t) \tag{5.43}$$

$$R'_{est} = R'(t_{k-1}, \Delta t). \tag{5.44}$$

Error terms are then defined for each state such that

$$R_E = R_i - R_{est} \tag{5.45}$$

$$R'_E = \frac{\partial R_i}{\partial t} - R'_{est}. \tag{5.46}$$

The state of the track is then updated such that

$$R(t_k) = R(t_{k-1}) + \alpha R_E + \beta R'_E \Delta t \tag{5.47}$$

$$R'(t_k) = R'(t_{k-1}) + \epsilon \frac{R_E}{\Delta t} + \gamma R'_E \tag{5.48}$$

$$R''(t_k) = R''(t_{k-1}) + \zeta \frac{R'_E}{\Delta t^2} + \eta \frac{R'_E}{\Delta t} \tag{5.49}$$

where

$\alpha, \beta, \epsilon, \gamma, \zeta,$ and η are filter coefficients and are ≤ 1.

The smaller a coefficient is, the less effect that error will have on the updated state parameter. Controlling these parameters allows for different filter profiles. Each filter configuration can filter out more measurement error, but it must sacrifice the ability to efficiently track target manoeuvres. Alternatively, the filter can allow a track to closely follow a manoeuvring target but include more measurement error in the positional estimation.

5.5.3 Detection association

Consequently, determining which detections received from target extraction belongs to a track, which are new targets, and which are false detections, becomes important. This process was realised by defining a range error model function

$$E = \left(\frac{R_{est} - R_i}{R_t}\right)^2 + \left(\frac{R'_{est} - R'_i}{R'_t}\right)^2 \tag{5.50}$$

where

R_t and R'_t are thresholds which determine the maximum error allowable in each dimension.

In (5.50), $E \geq 1$ represents an error that exceeds the limits, which ultimately define an ellipsoid in the signal space wherein potential targets must lie.

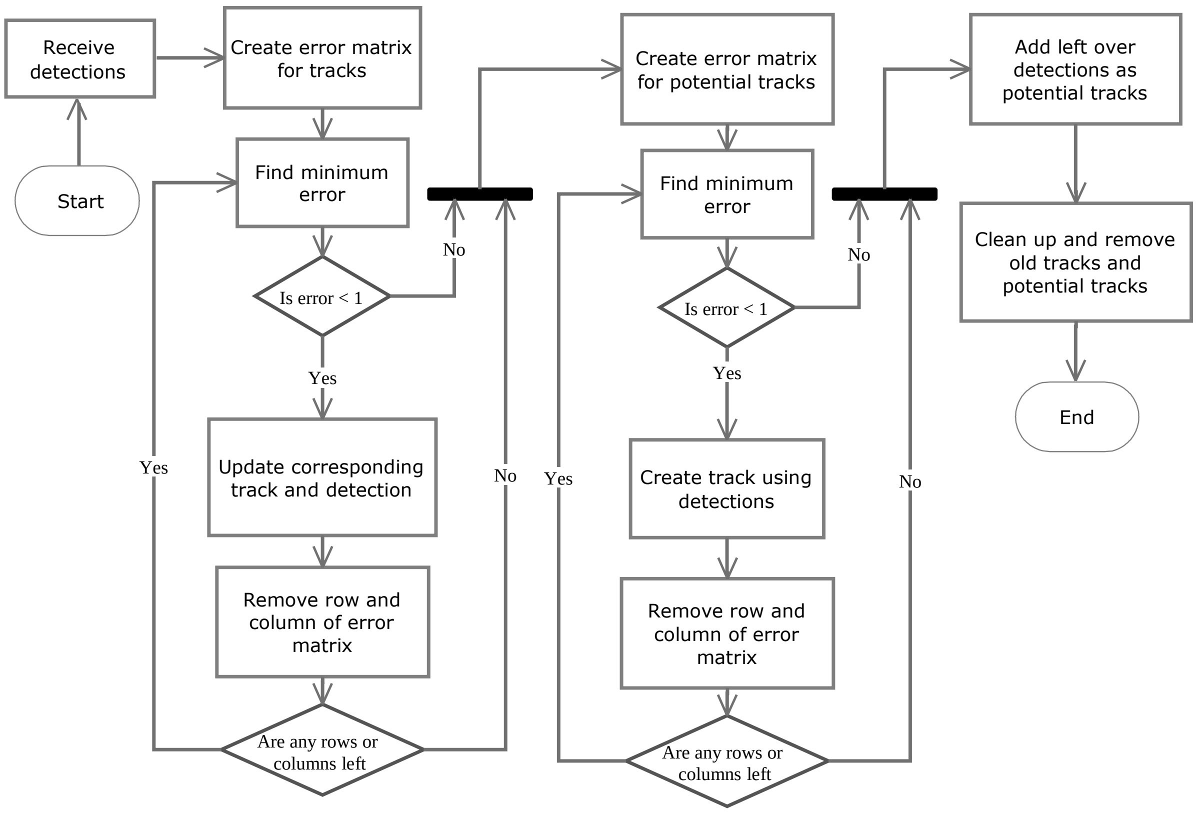

Receive detections
Start
Create error matrix for tracks
Find minimum error
Is error < 1
No
Yes
Update corresponding track and detection
Remove row and column of error matrix
Are any rows or columns left
Yes
No
Create error matrix for potential tracks
Find minimum error
Is error < 1
No
Yes
Create track using detections
Remove row and column of error matrix
Are any rows or columns left
Yes
No
Add left over detections as potential tracks
Clean up and remove old tracks and potential tracks
End

Figure 5.21 Processing of detections in state smoothing filter.

The process of associating raw targets to tracks was executed sequentially by assigning a maximum of one detection to each track per CPI. Once a detection was found to belong to a track, it would no longer be considered a candidate for any other tracks. The association process is shown in Figure 5.21. This approach is adapted from that given in literature [17].

The association process, shown in Figure 5.21, begins by creating an error matrix E which calculates the error between each track and detection. E_{ij} is the error between the ith detection and the jth track, determined using (5.50).

Once the error matrix is calculated the minimum element is found (E_{min}) and checked to ensure that it is below the threshold. If E_{min} is smaller than the threshold, the track is updated with the corresponding detection using the process described in Section 5.5.2. The row and column of the Error matrix is removed to remove the track and detection as candidates for further iterations. This process continues until the Error matrix is empty, or E_{min} exceeds the threshold.

Once this occurs, a new Error matrix is formed. However instead of using tracks, the potential tracks are used. The same process is followed as above in order to update the potential tracks. Once the error fails the threshold or the error matrix is empty, the remaining detections are appended to the vector of potential tracks, such that

$$R(t_k) = R_i$$
$$R'(t_k) = R_i'$$
$$R''(t_k) = 0$$

The final operation required is to determine when a potential track should be upgraded to a track, and when a track is no longer active. This was achieved by defining a health system, where each track and potential track has a health value which is increased when the track is updated, and decreased if it is not updated during an association procedure. In the implementation a 3-2 weighting is used where an update increases the health of a track by 3 and not updating decreases the track's health by 2. A potential track was upgraded to a track when the health ≥ 7 and destroyed when the health ≤ 0.

5.6 SYSTEM OUTPUT

The output of the system is therefore a set of targets, and their trails. Each target has an associated bistatic range and bistatic velocity. However, this data alone is not enough to position a target. Rather each bistatic range, assuming a 2D model, describes an oval of Cassini [43]. Therefore, in order to locate a target, either an angle of arrival in azimuth is required or another sensor.

If a 3D position is required, an azimuth and elevation angle is required. Alternatively, the measurements from three sensors are required [43]. The data from the matched filter can be easily combined with other coherent systems to allow for phased coherent processing. Alternatively, a multi-site system can be established, with each node being a radar as described in this report.

5.7 SUMMARY

The elements of the passive radar processing chain were investigated, and the performance critical elements were discussed. These included the cancellation filter, matched filter, target detection and state smoothing.

Different methods for implementing the cancellation filter were investigated, and it was found that by exploiting redundancy in the calculation of a sample covariance matrix, a Wiener filter with an LU solver is the most suited to a PCL using FM signals. In cases where an occasional failure to converge is tolerable, a Cholesky solver can be used to increase the filter throughput even further.

It was shown that three main options exist to realise the matched filter. Of these, it was shown that, generally, where the number of range bins is 2.5 times greater than the number of Doppler bins, the FDTC algorithm has a lower computational complexity. However, when the number of Doppler bins is greater than the number of range bins, FDC has the lowest computational complexity.

An initial processing architecture was built based on those found in literature. Using this system, it was found that the cancellation filter required more computation time than the matched filter. After moving to a real-time and optimised system, it was found that the matched filter now took 16.6 times longer to execute than the hybrid LU filter.

A target extraction scheme was introduced which was optimised for sparse targets. A state smoothing filter scheme was developed to refine the bistatic range and velocity measurements, as well as associating detections to a target. The state smoothing filter operated on a constant acceleration model, and used an alpha-beta like scheme to update the target states.

CHAPTER 6 RESULTS

The results of tests performed using the implemented system are presented and evaluated in this section. The configuration of the test system is described below. Then the detection, state smoothing, and processing performance is evaluated, and compared to existing and comparable systems.

The deployment was held with the aim of monitoring the airspace over OR Tambo International Airport. The results of the system are compared to ADSB data. From this, the detection and state smoothing performance is derived. The cost of the system is then presented along with the final performance of the system. Finally, the system is compare with two other systems presented in literature.

6.1 CONFIGURATION AND IMPLEMENTATION

In this section the deployment and the design choices are discussed in detail.

6.1.1 Deployment site

The deployment site is depicted in Figure 6.1. The goal for this passive radar was to monitor the air space around OR Tambo International Airport. In order to achieve this goal a suitable transmitter was selected. This transmitter was the Gelukstroom transmitter (Tx in Figure 6.2, at $25°41'21.00"S$ and $27°59'2.00"E$) and was chosen because of its high transmit power (110 kW at 94.2 MHz) and favourable orientation and distance from the area of interest. The general area of interest is highlighted by the yellow ellipse. The passive radar used a separated reference architecture which means it used two antennas. The first is a reference antenna, which was pointed towards the receiver (depicted by the red arrow in Figure 6.1. The second antenna is the surveillance antenna, which is aimed towards the area of interest, and is depicted in Figure 6.1 by the blue arrow.

The choice of receiver sites was largely limited by accessibility. Of these, the sites closest to the direct path between the area of interest and the transmitter were investigated. The final site was chosen by measuring the DPI at each site and ensuring unobstructed reception from the area of interest. The site chosen is shown as Rx in Figure 6.2, at $25°52'27.39"S$ and $28°14'12.18"E$. As shown in Figure 6.1, the distance between the transmitter and the receiver is 32.68 km while the distance from the receiver to OR Tambo International Airport is 28.90 km. The bistatic distance from transmitter to OR Tambo International Airport and back to the receiver is 84.13 km.

It was found that the site ultimately produced good results because of two complementary factors. Firstly, the site was on a hill facing towards OR Tambo International Airport. This enabled the unobstructed reception of signals when they originated from a source with an elevation above approximately 4°. Secondly, the site was located such that the direct path was blocked by the hill itself, with a steep embankment providing shielding from the direct path. The site is shown in Figure 6.2. It should be noted that the surveillance antenna was installed in an array as the deployment was used for multiple experiments.

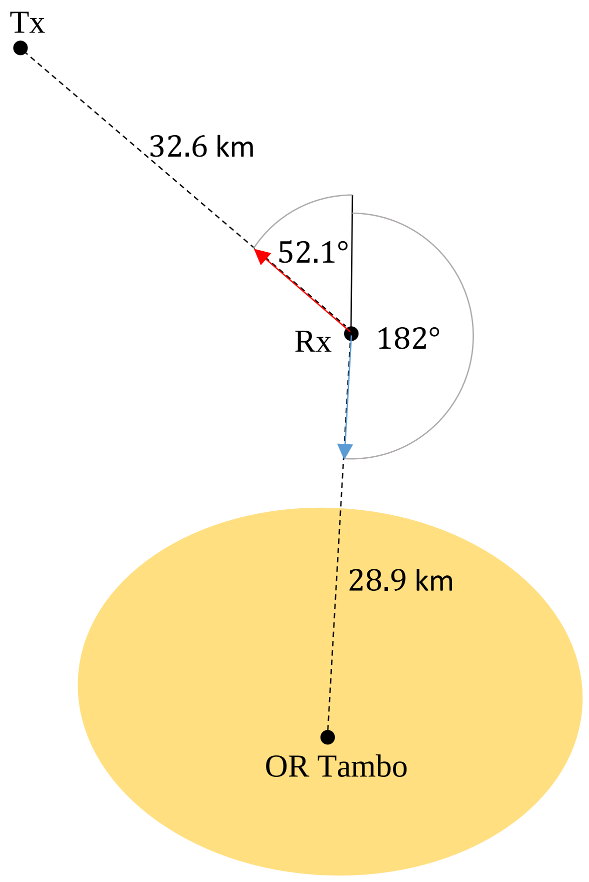


Figure 6.1 Over head diagram of passive radar deployment.

The reference antenna used for the deployment is shown in Figure 6.2. It is an 8 element Yagi-Uda antenna with a boom length of 2.4 m and a forward gain of 10 dBi [72]. This antenna was chosen for its narrow 3 dB beamwidth which is approximately 15° in elevation and 13° in azimuth.

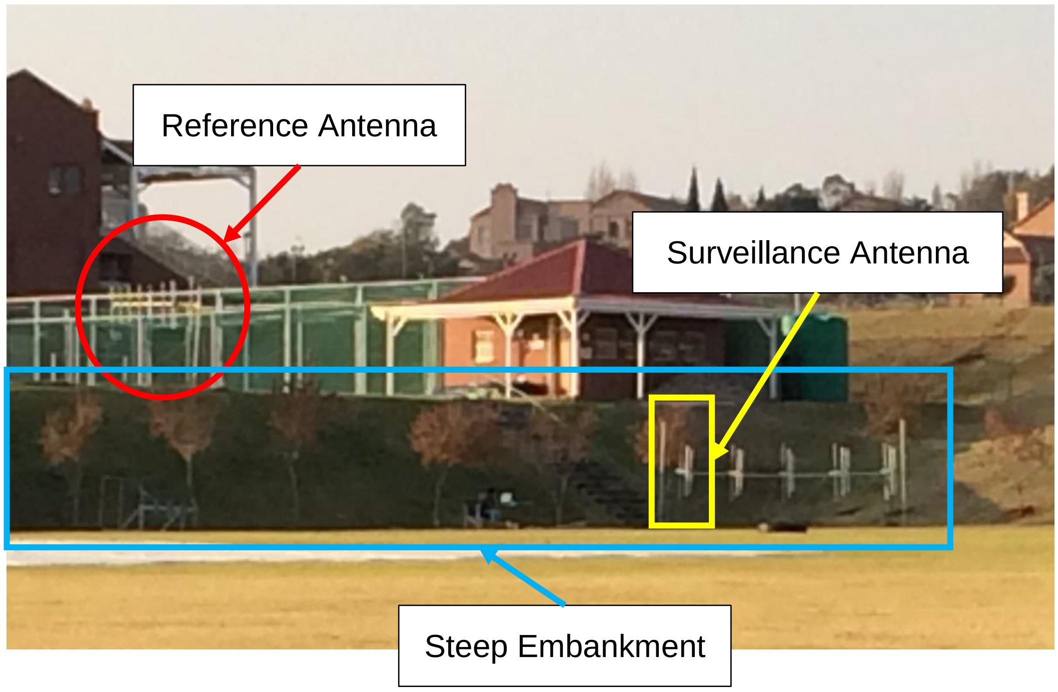


Figure 6.2 Image of deployment site.

The surveillance antenna is a 3-element Yagi-Uda antenna, with a boom length of 1.3 m and a forward gain of 6 dBi [72]. It was chosen to supply a wide beamwidth, while still providing attenuation for sources originating outside of the area of interest. This was chosen to maximise the probability of intercept of a target while decreasing DPI and clutter. The 3 dB beamwidth of this antenna is approximately 60° in elevation and 35° in azimuth. The surveillance antenna reaches a 10 dB beamwidth in azimuth at approximately 60° and nulls at $\pm 90°$.

Given the beam pattern, the probability of detection is impaired when a target is out of the main beam and is further impaired as the azimuth angles converge to the nulls. Figure 6.1 can be updated and is shown in Figure 6.3. Here the green region represents the angles where there is a high P_D, yellow represents the angles of impaired P_D, and red where there is a low P_D.

6.1.2 Configuration

The configuration parameters for the radar itself are shown in Table 6.1.

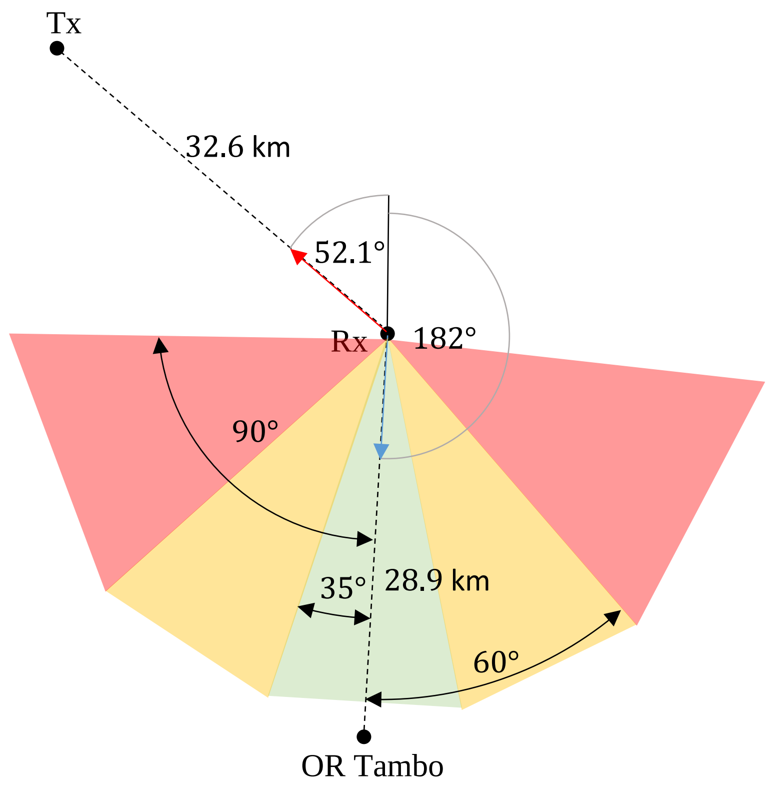


Figure 6.3 Angle versus probability of detection for radar deployment.

The CPI was chosen to be 4 s [5, 13]. This CPI was chosen to maximise the processing gain (and thus the SNR) while limiting target smearing, in both range and Doppler, to a manageable level. This resulted in a Doppler bin size of 0.25 Hz. This long CPI length was only possible due to the targets of interest; as commercial aircraft tend to fly on slow, relatively constant trajectories. If the targets of interest were more agile, then a shorter CPI would be required, although this would result in a lower P_D.

The sample rate was chosen at 200 kS/s as this was somewhat above the bandwidth of the signal, as well as a factor of the digital clock on the receiver, which is a requirement for this specific receiver. The baseline distance, maximum bandwidth, and operating frequency are all resultant from the choice of transmitter and location, as has been discussed in previous sections.

Table 6.1 Radar configuration.

Symbol	**Description**	**Value**
$\mathbf{CPI}$	CPI	4 s
$\mathbf{F_s}$	Sample rate	200 kS/s
$\mathbf{BW}$	Maximum bandwidth of source signal	150 kHz
$\mathbf{R_0}$	Direct path distance	32.6 km
$\mathbf{N}$	CPI length	800 000
$\mathbf{F_0}$	Operating frequency	94.2 MHz
$\mathbf{P}$	Number of blocks	10
$\mathbf{N'}$	Block length	80 000
$\mathbf{M}$	Cancellation filer length	256
$\mathbf{K}$	Number of CGLS iterations	30
$\mathbf{N_R}$	Number of range bins	150
$\mathbf{N_D}$	Number of Doppler bins	1001

The number of blocks was chosen to select the cancellation width in the Doppler dimension. It was observed that the majority of clutter existed within 5 m/s which translates to a Doppler offset of 1.57 Hz. This results in a lock length of 0.637 s or 6.28 blocks. This was increased to 10 to round-off the number of the blocks, block length, and other variables in subsequent operations.

The cancellation filter length was set to 256, which cancelled 383.7 km of clutter. This value was chosen as it cancelled all significant clutter without requiring excessive processing.

The number of range bins (N_R) was chosen to extend just past the distance receivable by the maximum range at which targets were detectable (which was empirically found). The maximum bistatic range at which a target was detectable was just below 200 km. The 150 range bins, in this configuration, results in a maximum detectable bistatic range of approximately 225 km (see Table 6.2) which leaves room for detections past where they were detected during the evaluation of the radar.

The number of Doppler bins was chosen to be able to detect, at least, the fastest commercial aircraft, which (for a Boeing 747-8) has a maximum velocity of around 988 km/h (274.5 m/s). As there is negligible computational cost in computing further Doppler bins using FDC, this was extended in the eventuality that other, faster, aircraft could also be detected.

From these design choices, some of the basic parameters of the radar can be defined and are shown in Table 6.2.

Table 6.2 Performance limits of radar configuration.

Description	Formula	Value
Doppler resolution	$\frac{1}{CPI}$	0.25 Hz / 0.796 m/s
Bistatic range resolution	$\frac{c_0}{BW}$	1998.6 m
Maximum Bistatic range	$R_0 + N_R \frac{c_0}{F_s}$	224.88 km
Maximum Bistatic range rate	$\left(\frac{(N_D - 1)}{2}\right)\left(\frac{\lambda}{CPI}\right)$	397.8 m/s
Range bin size	$\frac{c_o}{F_s}$	1498.96 m
Doppler bin size	$\frac{1}{CPI}$	0.25 Hz / 0.796 m/s

The configuration for target detection and extraction is shown in Table 6.3.

Table 6.3 Detection and state smoothing configuration.

Symbol	Description	Value
$\boldsymbol{N_S}$	Number of CFAR interference samples	30
$\boldsymbol{N_g}$	Number of CFAR guard bins	8
$\boldsymbol{N_C}$	Number bins used for CFAR censoring	2
$\boldsymbol{P_{FA}}$	Probability of false alarm	10^{-8}
$\boldsymbol{\alpha}$	Alpha parameter of state smoothing filter	0.4
$\boldsymbol{\beta}$	Beta parameter of state smoothing filter	0.1
$\boldsymbol{\epsilon}$	Epsilon parameter of state smoothing filter	0.4
$\boldsymbol{\gamma}$	Gamma parameter of state smoothing filter	0.8
$\boldsymbol{\zeta}$	Zeta parameter of state smoothing filter	0
$\boldsymbol{\eta}$	Eta parameter of state smoothing filter	0.02
$\boldsymbol{L_+}$	State smoothing filter health increment	3
$\boldsymbol{L_-}$	State smoothing filter health decrement	2
$\boldsymbol{L_{max}}$	Maximum health value for a track	50

Many of these were empirically tuned so as to provide better performance in this environment. The P_{FA}, however, was chosen specifically for the purpose of demonstrating the results in a report. To that end, the P_{FA} was chosen to minimise false detections such that the generated results and figures more easily illustrate the underlying concepts. This decision has an adverse effect on the P_D, as selecting a low P_{FA} increases the detection thresholds, and thus, lowers the P_D. The state smoothing filter was able to successfully operate and filter out the false detections in the case of a higher P_{FA}. In operation, a P_{FA} of 10^{-6} would be more appropriate, as this strikes a more favourable balance between P_{FA} and P_D. The FAR achieved when $P_{FA} = 10^{-8}$ is 1.50×10^{-3} FA/CPI.

Another observation made was that the theoretical P_{FA} selected (with a threshold calculated as described in Section 5.4.1) did not result in the correct P_{FA}. This indicates that the assumption that the noise is AWGN is not entirely valid. However, this is not entirely unexpected, as previously established, the main source of noise in the system is DPI which does not conform to AWGN.

The state smoothing filter parameters were tuned to support slow manoeuvres, but also to bridge gaps in detection. In this configuration, the accuracy of the bistatic range rate is significantly better than the accuracy of the bistatic range. The bistatic range rate's influence over the track's state was, therefore, emphasised resulting in more accurate state smoothing.

The system's digital processing was implemented on an embedded platform using an NVidia Tegra K1 processor. The radar used an Ettus USRP B210 SDR as a digital receiver. The hardware configuration is shown in Figure 6.4.

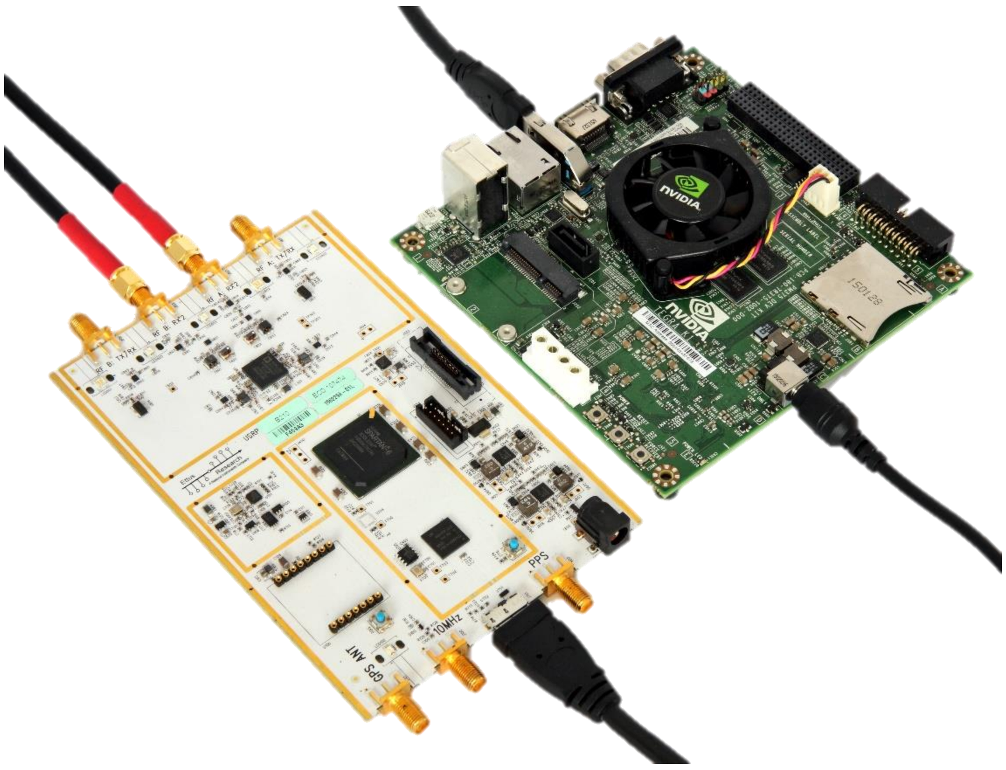


Figure 6.4 Hardware configuration.

6.2 DETECTION CAPABILITY

In order to assess the detection capability of the system, the system was deployed and set to record for 27 minutes. The recording was then processed to characterise the system.

6.2.1 Control data

In order to determine the metrics above, as well as the function of the radar, the real positions of the targets were required. This was obtained via ADS-B transmissions which were recorded by a server. ADS-B [73] is a system which many commercial aircraft have installed which continuously broadcasts information about the aircraft, including its position, altitude, velocity and heading.

The specific information which was used is transmitted over an extended squitter which has a data block described in Figure 6.5.

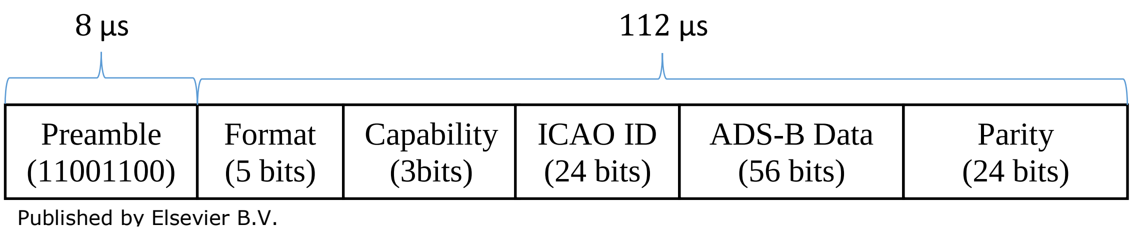


Figure 6.5 ADS-B extended squitter. Adapted from **[73]**, with permission.

The format for a Mode-S position is 17 or 10001_2. The ADS-B data block is also broken into a number of message types with a 5-bit sub-format code being the first 5 bits of the ADS-B data. The two position squitters transmitted are ground and airborne positions. Both record the position as geodetic coordinates and are encoded into CPR format.

It should be noted that it was found that a delay of approximately 8 seconds existed on the data. This was removed when the data was compared. The ADS-B detections recorded over the experiment are shown in Figure 6.6.

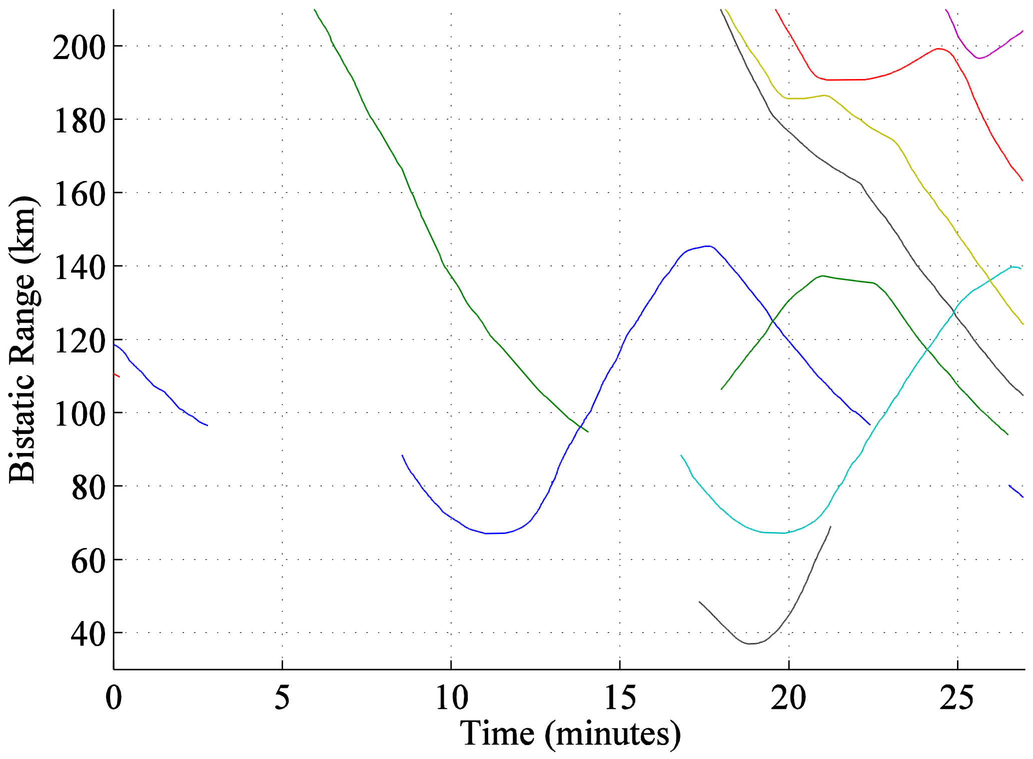


Figure 6.6 Bistatic range of ADS-B detections versus time.

Figure 6.6 demonstrates that, a total of 11 targets were detected with ADS-B transceivers. A number of these either begin or end at a bistatic range of approximately 90 km which corresponds to a local airport (OR Tambo International Airport) where many flights either began or ended.

The angle of the ADS-B targets is shown in Figure 6.7. From Figure 6.7 it can be seen that many of the tracks either end or originate at 0°. This is expected as the surveillance antenna was set to point towards OR Tambo.

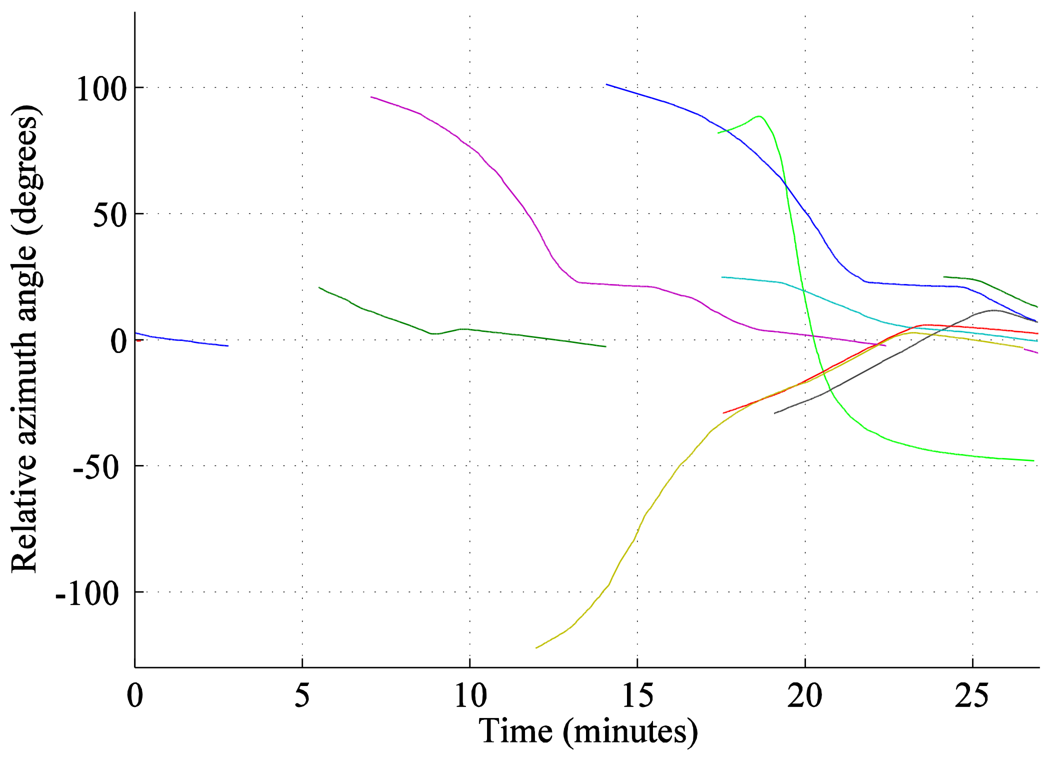


Figure 6.7 Relative angle of ADS-B targets.

6.2.2 Detection probability

Detection probability refers to the probability that a target will be detected by the radar once it enters a specific range. This value was empirically determined by comparing the time where a target was present and the amount of time it was detected by the system.

It is assumed that a target was detectable half a CPI before and after a detection, such that it was present for the entire CPI. The empirical probability of detection is therefore given as

$$\tilde{P}_D = \sum_{i=1}^{N} \frac{t(R_i \leq R_0)}{t(\hat{R}_i \leq R_0)} \tag{6.1}$$

where

R_i is the range of the i^{th} target,

$\hat{R}_i$ is the detected range of the i^{th} target,

N is the number of targets,

R_0 is the range threshold, and

$t(x \leq x_0)$ is the amount of time that $x \leq x_0$.

The detections received from target extraction are shown in Figure 6.8.

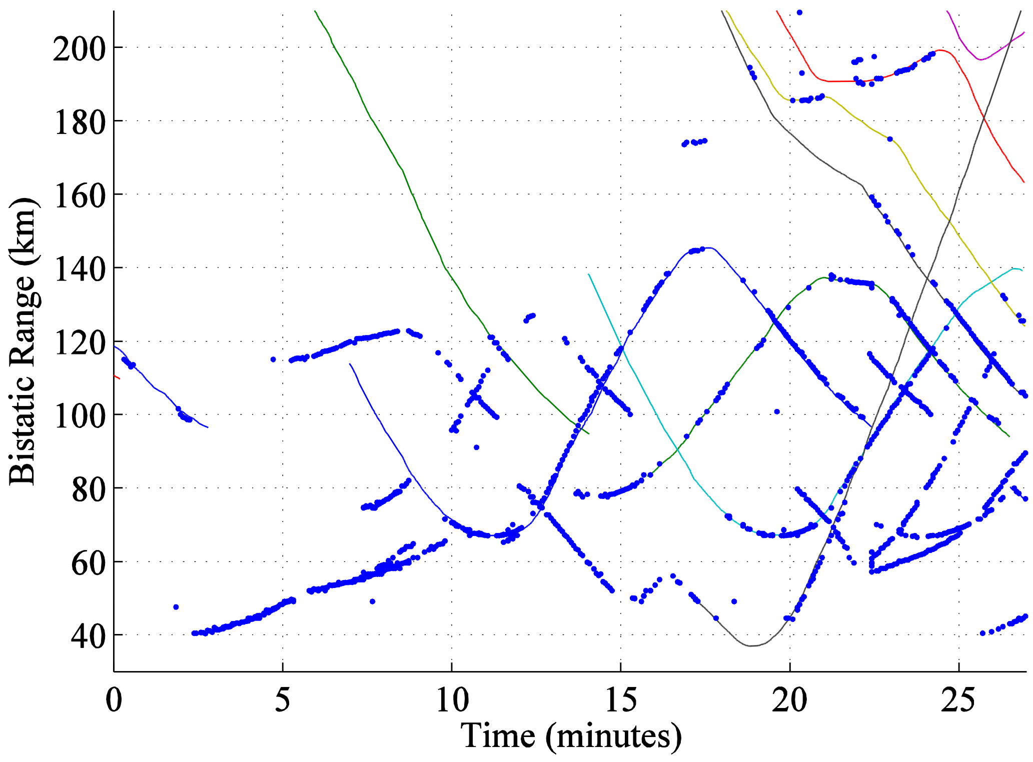


Figure 6.8 Passive radar detections compared to ADSB tracks.

In Figure 6.8 it can be seen that 10 of the 11 ADSB targets were detected during the test interval. The only ADS-B track that was not detected, never had a bistatic range lower than 196.6 km and a monostatic distance of 91.28 km. It can also be seen that there were no detections with a range below 40 km. While there were detections below that threshold, there was significant DPI interference and, as a result, many false detections. Note that the minimum bistatic range a target can exist at is the baseline distance, as this would represent a target on the line between the transmitter and receiver. In this case that is 32.6 km. In order to compensate for the DPI, a clutter notch was established for the first 7.5 km of bistatic range (which is potentially only 3.7 km from the receiver).

Two of the ADS-B targets (the blue and turquoise tracks) were not originally detected, however, both of these targets originated from the right with azimuth angles greater than $90°$. However, as each entered the main beam of the antenna it was detected. The blue ADS-B target (purple in Figure 6.7) was detected once it reached $77.86°$, which is far beyond the beamwidth of the antenna. The turquoise track (blue in Figure 6.7) was detected once it reached $78.08°$, which is also beyond the beamwidth of the surveillance antenna.

The furthest confirmed detection (i.e. confirmed with ADS_B data with the furthest bistatic range) had a bistatic range of 198.2 km, and a monostatic range of 88.2 km. The confirmed detection with the furthest monostatic range, had a range of 90.35 km and a bistatic range of 194.5 km.

In terms of angle, the furthest detection left occurred at $-103.8°$, which is completely out of the main beam. The furthest detection right was at $83.61°$ which is at approximately -20 dB on the surveillance antenna.

From these an area of operation is defined such that targets are only considered detectable when they have an azimuth between -103.8 and 83.61. The P_D per CPI was empirically calculated for ADS-B targets in the area of operation. The empirical P_D per CPI for a target within a certain bistatic range is shown in Figure 6.9.

As can be seen from Figure 6.9, the P_D per CPI begins at 0.48 for a target in the range of 41 to 43 km, and ends with a P_D per CPI of 0.421 for a target in the range of 41 to 210 km. The maximum P_D per CPI was 0.84 for a target between 41 to 69.6 km.

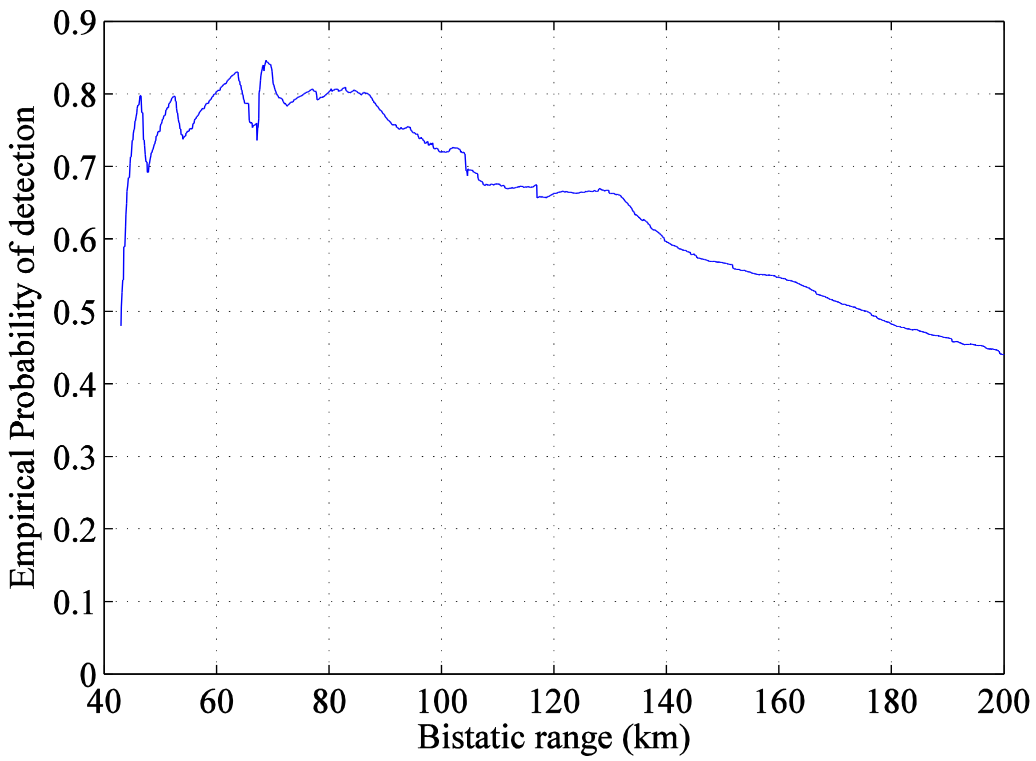


Figure 6.9 Empirical probability of detection per CPI within a given bistatic range.

Figure 6.9 can be broken into three main sections. The first region is before 46.4 km. In this region it can be seen that the probability of detection is low. However, this section of the statistic was not built on a large sample, as only one target entered this region over the test period. Furthermore, that target had an angle outside of the main beam of the surveillance antenna and is, therefore, expected to have a reduced probability of detection.

The second region lies between 46.4 km and 84.8 km. This region is categorised by fluctuations and a steadily, although slowly, increasing P_D. This increase is unexpected when taking into account the radar range equation (3.8). A possible explanation would be that targets in this region were obscured by uncancelled components of DPI. This end value, however, does coincide with the distance of the airport at which the majority of ADS-B targets converge. Moreover, the P_D around the airport is smoother than the fluctuating area. This suggests that the fluctuation is due to the small sample size and the large number of variables which are not accounted for, such as RCS and direction.

The third region exists from 84.8 km to the end range of 210 km. Here a fairly constant decline in P_D is visible which conforms to a linear profile. This decrease in P_D is expected, as the power received from a target is expected to decrease as range increases. With that decrease in power, comes a decrease in SNR, and finally a decrease in P_D.

P_D per CPI remains above 0.5 in the range 52 to 175.7 km. At a bistatic range of 52 km the closest detection reached a monostatic range of 10.0 km. At a bistatic range of 159 km the furthest detection was at a range of 68.6 km.

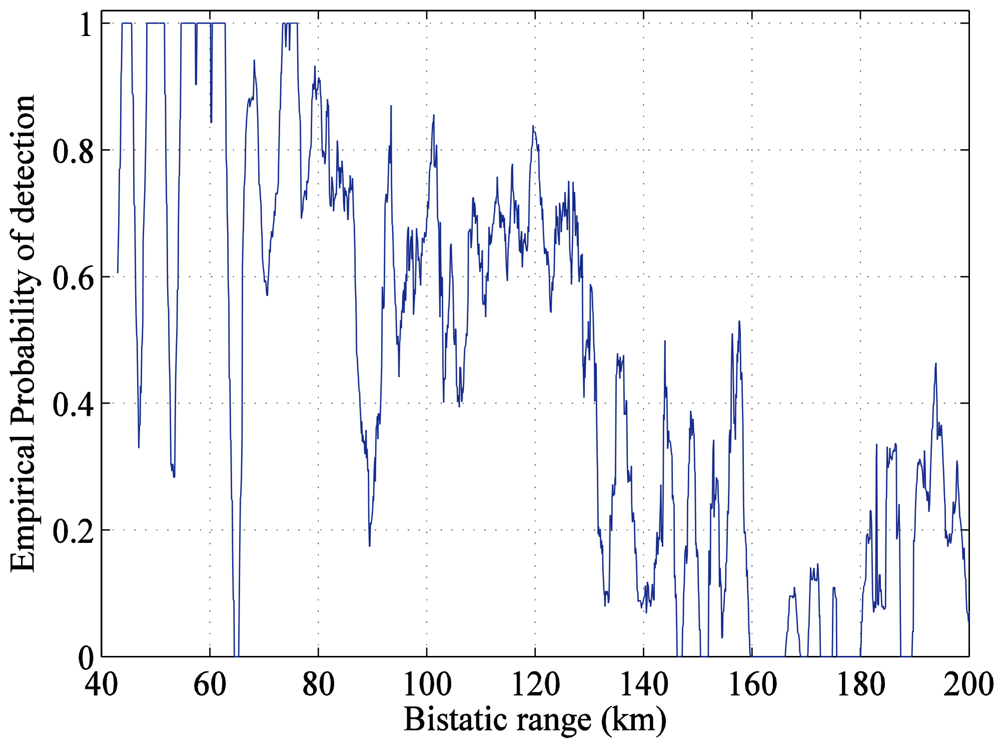


Figure 6.10 Empirical probability of detection per CPI of a target at a given range.

Figure 6.10 shows the empirical probability of detection per CPI (for a target within the area of operation), where each value is the probability calculated using all tracks within 1 km of the centre point. It should be noted that the there is insufficient data to form reliable statistics, but rather broad trends. Figure 6.10 demonstrates that certain range regions have a relatively high probability of detection, while others are apparent blind spots. The most prominent of these is from $159 - 180$ km where almost no targets were detected. Conversely $48.5 - 52$ km and $54 - 63.5$ km detected nearly all ADS-B targets while in that range.

6.2.3 Location error

The location error is the RMS error between the measured position of a target compared to its actual position. This was determined by

$$R_E = \sqrt{\frac{1}{\sum_{i=1}^{N} P_i} \sum_{i=1}^{N} \sum_{k=1}^{P_i} \left(\hat{R}_i(t_k) - R_i(t_k)\right)^2} \tag{6.2}$$

where

$\hat{R}_i(t)$ is the measured bistatic range of the i^{th} target at time t,

$R_i(t)$ is the actual bistatic range of the i^{th} target at time t,

P_i is the number of detections for the i^{th} target, and

$t_k$is the time at the k^{th} detection for a given target.

In order to determine the location error two separate metrics are used. The first was to analyse the location error of the raw detections and the second was to use the state smoothing filter’s estimation.

6.2.3.1 Raw detections

The raw detections were passed through the state smoothing filter in order to group the detections. All grouped detections are shown in Figure 6.11.mOf these the groups, the target groups corresponding to ADS-B targets were retained and used to evaluate the accuracy. The retained groups are shown in Figure 6.12. The absolute estimation error of the groups is shown in Figure 6.13. Here it is shown that the maximum error was 3.24 km. The RMSE for the detections is 0.950 km or 63% of a range bin’s width.

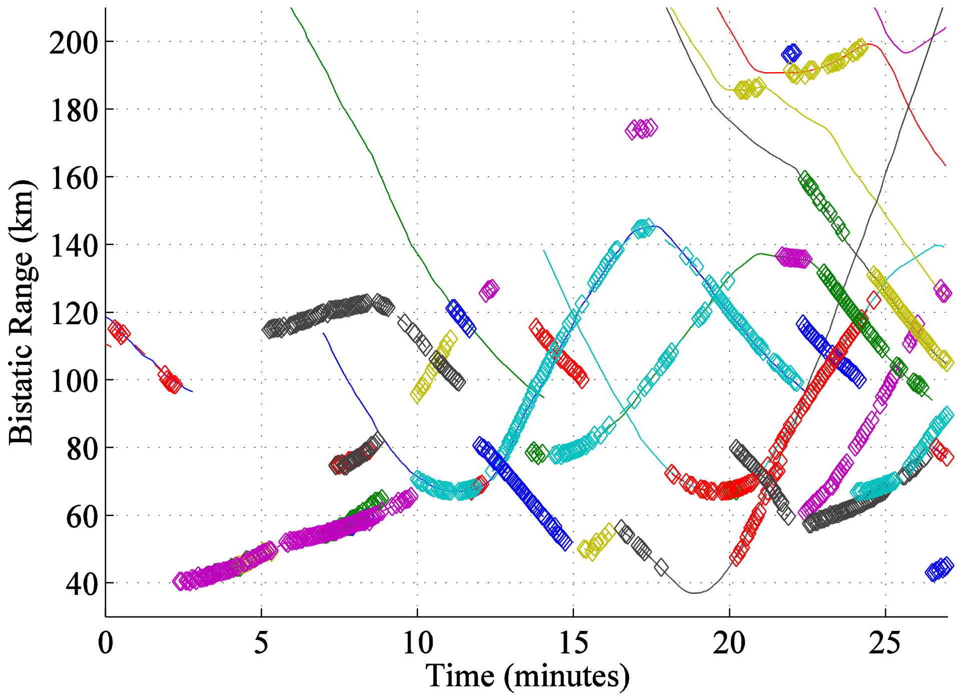


Figure 6.11 All grouped detections.

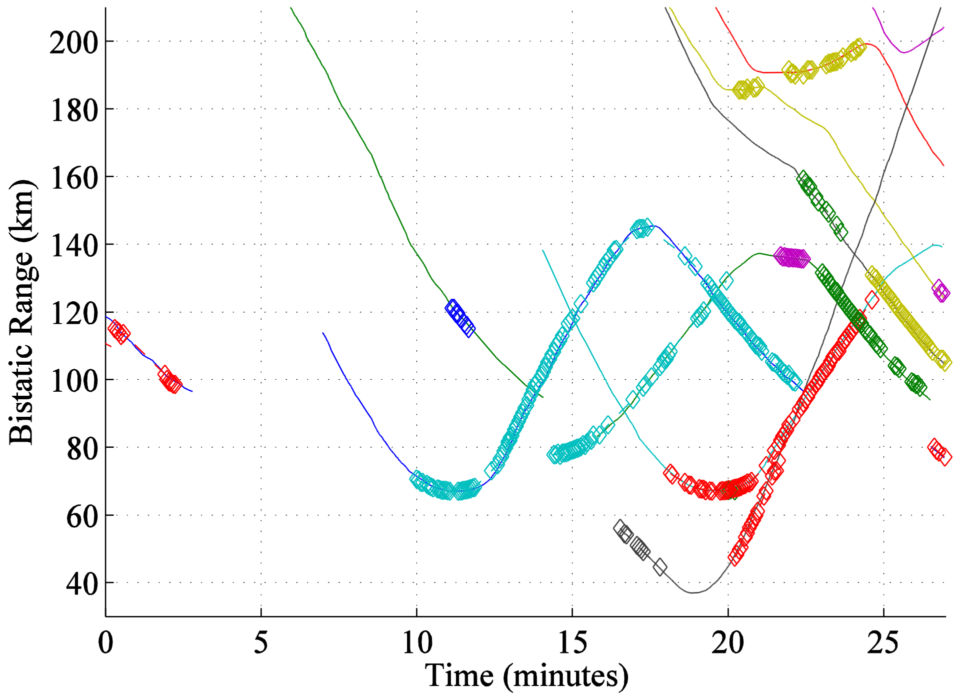


Figure 6.12 Grouped detections used for error estimation.

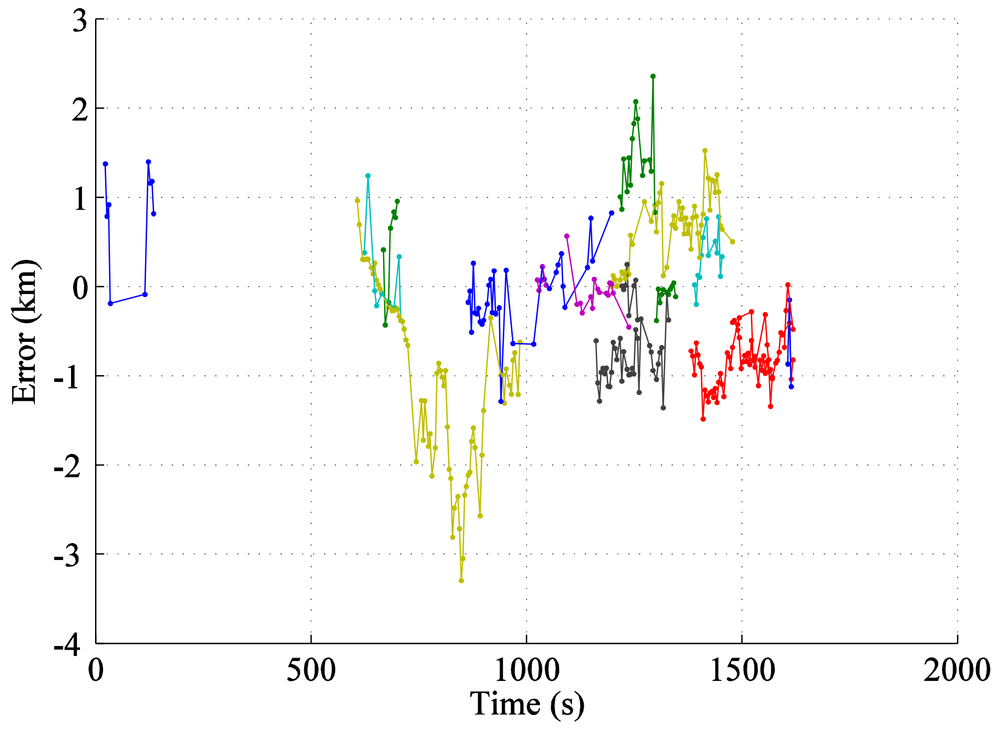


Figure 6.13 Absolute error of target groups.

6.2.3.2 State smoothing filter estimation

In order to evaluate the estimation error of the state smoothing filter, conditions were applied to only evaluate the tracking estimation while a target was being successfully tracked. A successful track was defined as a track which, once established, has had sufficient time for the filter to settle (which was set to 3 more consecutive detections) and while tracking there is no more than one miss. If consecutive misses occur the settling criteria is reset.

The error of the successful tracks is shown in Figure 6.14. The RMSE of the predicted position does not decrease by much to only 0.927 km. This is largely a result of the error's form.

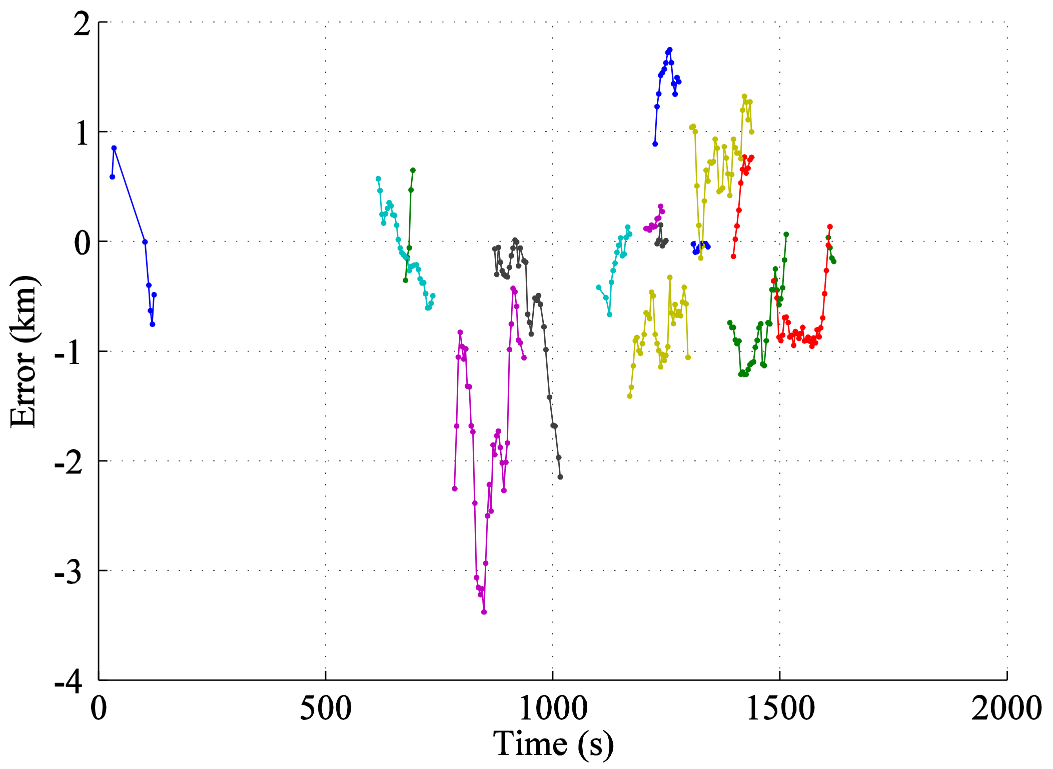


Figure 6.14 The location error of successful tracks.

The largest contributors to the error are depicted in Figure 6.14. These sections have a lower frequency component compared to what the state smoothing filter was set to reject and, as such, little error rejection occurs. Furthermore, the state smoothing filter here was configured in such a way that target grouping and gap filling were prioritized over estimation accuracy (i.e. the filter was configured such that tracking of targets were prioritized, but sacrificing noise rejection).

The tracks' error plots do not oscillate around zero as would be expected from a measurement noise, but rather have an offset bias. This implies that the measurement noise does not have a zero mean. As each track has a unique offset, the measurement error could either result from an error in the estimation of the bistatic range from the ADS-B data, or other sources of inaccurate data. Examples are the positions of the transmit and receive antennas.

Nonetheless, even a strict filter (as it essentially acts as a high pass filter) would be ineffective at rejecting this type of measurement error.

An example of gap filling as performed by the state smoothing filter is shown in Figure 6.15. Here it can be seen that the detections are intermittent. However, the state smoothing filter successfully predicts the state of the target and allows for the track to be continued over the turn. The success of the track estimation is resultant from the accurate Doppler measurement rather than the position estimate. Furthermore, state smoothing was also found to be more accurate in Doppler than it is in range.

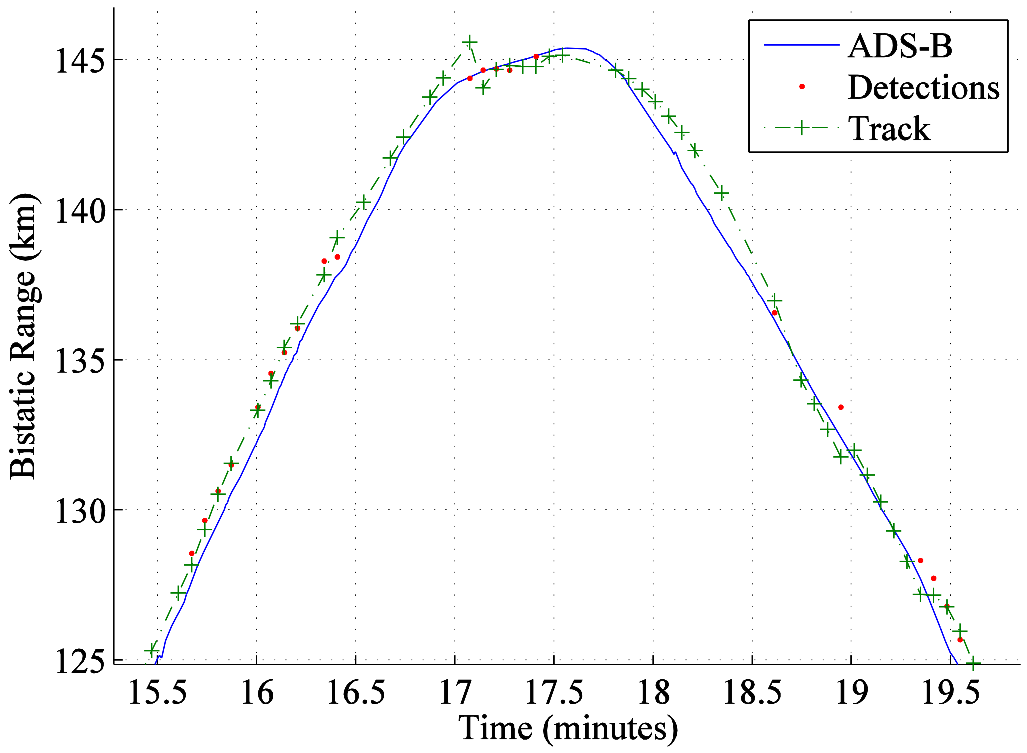


Figure 6.15 State smoothing filter gap filling and prediction.

6.3 PROCESSING PERFORMANCE

The processing performance was evaluated using the procedure described in Figure 5.10. The metrics used for measurement are the same as posited in Addendum C.

6.3.1 Mobile platform

The mobile system was tested over a number of cancellation filter sizes (M) and data lengths (N). For this set of experiments only the hybrid LU processing chain was considered as, from the results in Figure 5.16, it executed the quickest. Its CPU based counterpart was included for comparison. The optimised CGLS solver was also included as it is an iterative algorithm based purely on the GPU, which may result in better performance on some systems or with some configurations. The average throughput when $M = 256$ is shown in Figure 6.16.

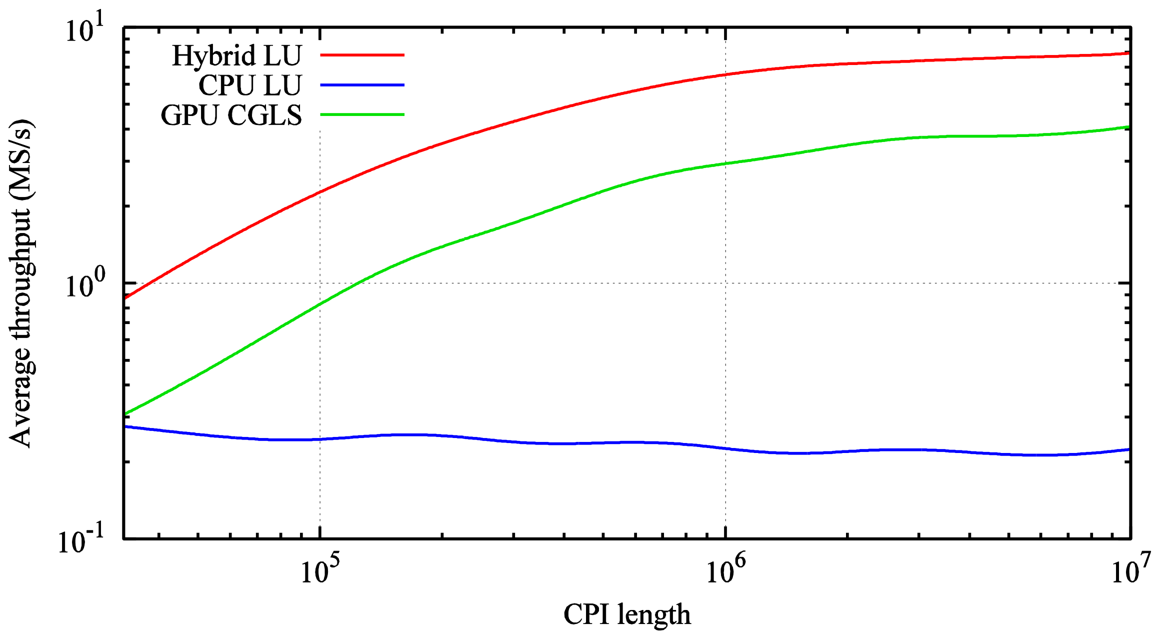


Figure 6.16 Average throughput of mobile processing chains.

In Figure 6.16 it is shown that the hybrid LU processing chain outperforms the other methods, as expected. The Hybrid LU processing chain reaches a maximum throughput when $N = 2^{24}$, with a throughput of 9.747 MS/s. At the chosen CPI length of $N = 800\,000$, the hybrid LU processing chain achieved a throughput of 6.192 MS/s.

This was far greater than the CPU implementation which achieved a maximum at $N = 3300$ of 274.9 kS/s. The CPU LU processing chain was 26.45 times slower than the hybrid processing chain when $N = 800\,000$.

The CGLS processing chain achieved a maximum throughput when $N = 2^{24}$, with a throughput of 4.597 MS/s. At the chosen CPI length of $N = 800\,000$, the CGLS processing chain achieved a throughput of 2.781 MS/s.

6.3.2 Embedded platform

The processing performance of the embedded platform was evaluated adopting the same technique as for the mobile system. However, due to memory restrictions, the CPU LU implementation could not be effectively implemented.

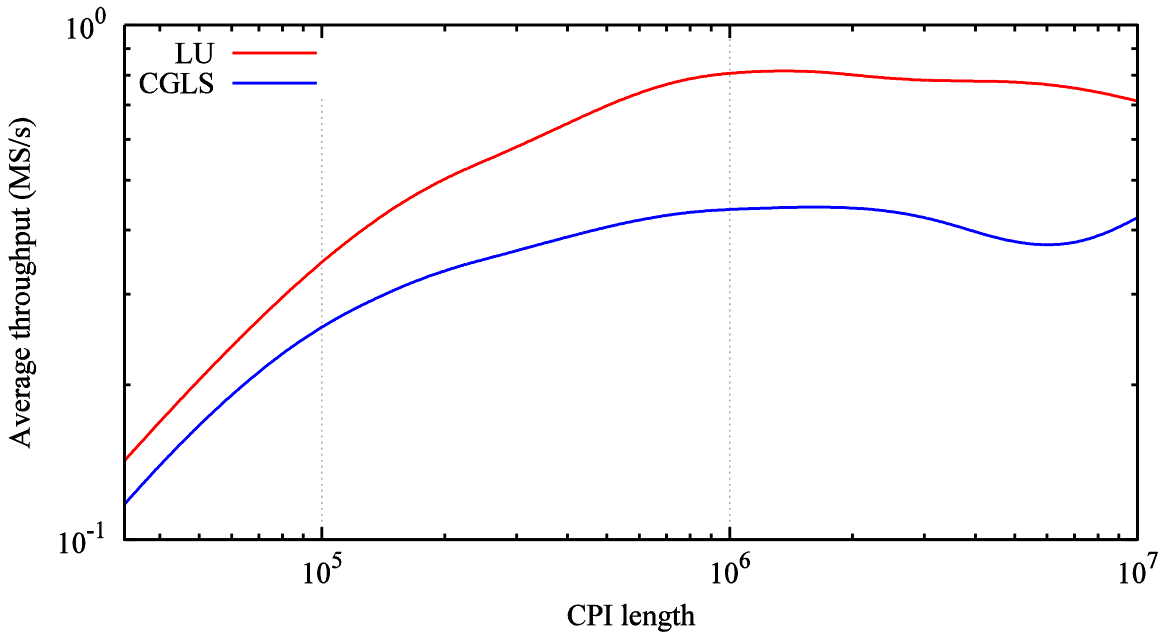


Figure 6.17 Average throughput for embedded processing chains.

The achieved throughput of the embedded system is shown in Figure 6.17. When compared to the results of the mobile system, it is evident that the trends seen are different. Unlike the mobile system, the LU processing chain reached maximum throughput at $N = 1.37 \times 10^6$

When configured for the implementation i.e. $N = 800\,000$, the throughput of the LU processing chain was found to be 786.9 kS/s. The CGLS processing chain achieved an average throughput of 432.9 kS/s at $N = 800\,000$. For the implementation the LU processing chain was 45.0% faster than the CGLS processing chain.

6.4 COST

The hardware cost to implement each system is shown in Table 6.4.

Table 6.4 Hardware cost of systems.

Item	Mobile	Embedded
Antennas and masts	$335.05*	
Cables and Misc. RF connectors	$40.23*	
Receiver	$1100.00	
Processing Platform	$1849.99	$192.00
Total	$3325.27	$1667.28

* Converted from Rand incl. VAT at an exchange rate of 13.49 R/$

6.5 DISCUSSION OF RESULTS

6.5.1 Detection capability

The radar demonstrated the capability to detect most of the ADS-B targets, only failing to detect 1 out of 11 targets. However, it is also clear, as shown in Figure 6.8 and Figure 6.11, that a number of other aircraft-like targets appear that have no ADS-B tracks. Although these could not be confirmed, there are 9 additional long tracks which exhibit the behaviour of aircraft.

Additionally, one of these targets, the purple track from 2.37 minutes to 9.8 minutes in Figure 6.11, displays propeller modulation [74]. This could be resultant from either a propeller driven aircraft or a helicopter (a rotating wing). The modulation is shown in Figure 6.18, the red ellipses indicate the target as well as the propeller modulation. What can also be seen (highlighted in the green rectangle) is the residual DPI which can cause false targets. The difference in frequency between the propeller targets and the aircraft itself is 81.61 Hz which results in a propeller speed of 4896.6 RPM.

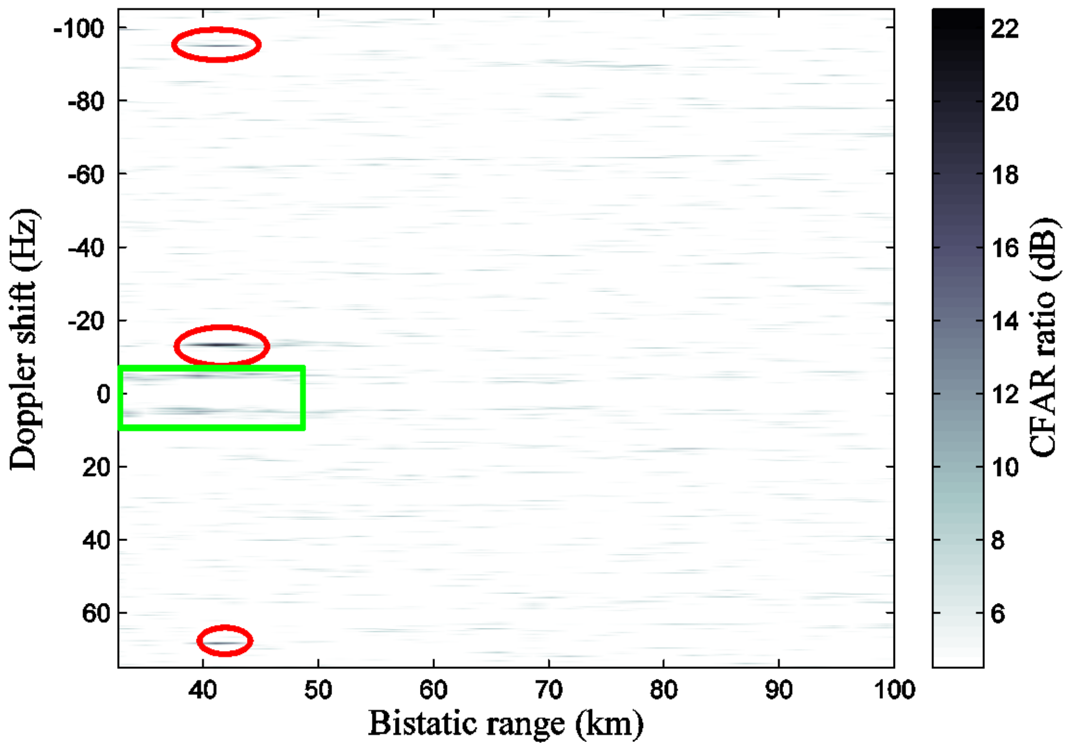


Figure 6.18 Snap shot of propeller modulation.

The probability of detection was estimated using recorded data, while the amount of data used is not sufficient to derive accurate statistics about the radar installation, it can be used to identify strong trends. This is compounded by the lack of information regarding the RCS of targets as well as their orientation, although the orientation can be roughly estimated by analysing the targets flight path. While this information would be vital to characterise the system, the primary purpose of this study is to demonstrate the function, feasibility and design procedures encountered in a low-cost passive radar system, and not that of developing a statistical understanding of its operation.

By referring to Figure 6.6, it can be seen that only one target was available from which to gather information before a bistatic range of 67 km. This target had a particularly poor rate of detection. This had a direct impact on the probability of detection below 67 km in bistatic range (Figure 6.9). However, what is visible in Figure 6.11 is that there are an additional 4 unconfirmed tracks in that region. Thus, indicating that there were additional aircraft in operation whom were not transmitting ADS-B data.

Looking back at Figure 6.9, it can be seen that, as other aircraft are considered, the probability of detection improves significantly. It is then shown in Figure 6.10 that a blind spot between 159 and 185 km of bistatic range was observed between which no targets were detected. Approximating the orientation from the target's flight path offers an explanation for this blind spot. By analysing Figure 6.8, it can be seen that in this dead spot the bistatic range rate of targets was always fairly large and negative. This would suggest that the aircraft were orientated with their front/rear perpendicular to the range ellipsoid, and therefore would have a smaller RCS than when the aircraft's broadside/flank is perpendicular.

From 69 km to 210 km, the general observation is that probability of detection fell as bistatic range increased. This trend is expected given equation (3.8).

The accuracy of the radar was found to be within one range bin with an RMSE of 0.950 km. The state smoothing filter resulted in little improvement, producing an RMSE of 0.927 km, which is a mere 2.48% improvement. However, the state smoothing filter did successfully group targets together and was capable of bridging gaps in certain scenarios. Furthermore, the errors found were not consistent with zero mean additive noise and, therefore, the filter was not successful in reducing the location error.

6.5.2 Processing performance

From the results in Section 6.3, it is clear that the system was capable of operating in real-time on both the mobile and embedded platforms. The hybrid LU solution was most effective for the mobile system, reducing processing latency to 185.2 ms while the CGLS algorithm had a processing latency of 460.5 ms. On the embedded system this was not the case where the CGLS algorithm was most effective with a processing latency of 2.741 s, the hybrid LU system had a processing latency of 3.064 s.

This discrepancy in results originates from the relative processing power of the CPU and GPU on each platform. On the mobile platform, the relative CPU power is far closer to the GPU than on the embedded system, and so the advantages seen through hybrid processing on the mobile platform is not experienced to such a degree as on the embedded platform.

The significant difference in performance between the mobile and embedded system was expected when considering the difference in cost and power consumption between systems. The embedded system cost 50.14% less but achieved only 6.76% of the performance of the mobile system. However, the embedded system requires less than 12 W to operate, which may be a more crucial factor in certain scenarios. The use of the mobile platform was shown to result in a more versatile system, being able to handle up to approximately 2 MHz signal, or be able to process 21 channels in a multi-channel system.

6.5.3 Comparison to existing systems

The system developed in this study is comparable to two other approaches in literature. The first [11], implements a passive radar using an Ettus SDR coupled with a custom built RF front-end. The processing is handled by an NVIDIA Jetson TK1 Development Board. The second system [10], feeds the captured signals directly into an Ettus SDR, and then processes the signals offline on a PC. A comparison between these systems and the one in this study is shown in Table 6.5. Here the system was configured to match that of Tong's [11] as closely as possible to allow for the best comparison.

From Table 6.5 it can be seen that the system proposed here is able to process a CPI approximately four times faster than the Tegra system [11] using a similar hardware configuration. Furthermore, the proposed system(i.e. the embedded system presented in this study) used an optimal least-squares solver, while the Tegra system (Tong's [11]) used a sub-optimal CGLS algorithm.

The bistatic detection range achieved by the proposed system is far greater than the SDR system [10].

Table 6.5 Comparison of existing systems.

Matched Filter	**Tegra System [11]**	**SDR System [10]**	**Proposed System**
CPI Length	800×10^3	Unknown	800×10^3
Cancellation filter algorithm	CGLS	Unknown	Hybrid LU Wiener
Number of Bins Cancelled	150	30*	150
Number of cancellation batches	Unknown	Unknown	20
Number of Range Bins	150	30*	150
Number of Doppler Bins	1601	Unknown	1601
Direct path distance	90 km**	48.3 km	28.6 km
Maximum bistatic detection range	250 km	65 km	198.2 km
Maximum detection range from receiver	100 km	Unknown	90.35 km
Cost excluding antennas	\$5260.00	\$700 (receiver only	\$1332.23
Average processing time	$\lesssim 4$ s	Not real-time	959.84 ms

* Estimated from Fig. 9. in [10]

** Estimated from Fig. 7. in [11]

The Tegra [11] system achieved a much higher bistatic detection range, but a very comparable range from the receiver. This is due to much longer direct-path used by the Tegra system [11]. As discussed in Section 3.2.1, an increase in direct path distance should increase the detection range from the receiver. However, due to the differences in sites, targets, and other environmental factors, these results cannot be directly compared. In order to do so, the systems would need to be deployed and run alongside one another.

It should however be noted, that a similar system [7] has also been presented by some of the developers of the Tegra system [11]. This system was not limited by real-time aspects, and instead focused on the performance of the receiver. In this case the maximum bistatic detection range was 383 km, which (with an estimated direct path distance of 90 km) requires 196 range bins to detect such targets. Hence, it stands to reason that the performance limiting factor of the Tegra system [11] is the processing power. With 200 range bins, the proposed system would require 1.593 s to process the results for each 4 s CPI. This means that employing the processing chain proposed in this study would allow the full range to be processed in real-time and even allow for overlapping CPIs. The system depicted in Fig 12. of [7] shows the use of two Tegras to enable the processing. By using the processing chain presented in this study, both channels could be processed on one board.

The hardware of the proposed system cost 3.95 times less than the hardware of the Tegra system [11], while managing to achieve comparable detection performance and improved processing performance.

6.6 SUMMARY

It was shown that the system was able to detected all aircraft with ADSB transceivers, except one target which did not come closer than 91 km of the receiver site. The system achieved a detection accuracy of 0.95 km. The system extracted those detections and successfully associated them with the correct targets. Once a state smoothing filter was applied, the system achieved a tracking accuracy of 0.927 km.

The system operated in real time. On the mobile platform the system required 129.2 ms to process a 4 s CPI, while the embedded system required 1.017 s. The total hardware cost for the mobile system was $3325.67, while the embedded system cost $1667.28.

When compared to other systems in literature it was found that the system cost approximately four times less, required approximately a quarter of the processing time, but achieved a detection range 9.65% shorter than that of the most comparable system.

CHAPTER 7 CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Passive radar is an emerging technology but its theoretical usefulness in low cost scenarios is subdued by requirements for processing performance and specialized hardware. In order to evaluate the practical potential for passive radar as a low cost radar, the factors which limit passive radar performance were investigated, identified and evaluated. The predominant factors were found to be related to dynamic range and direct path interference. General design practices were also investigated and developed from findings in literature.

A low cost passive radar was designed and implemented on a mobile PC as well as on an embedded processor. The detection capabilities of the radar were demonstrated, key stress points in the processing chain were evaluated and several strategies were developed which were required to enable real-time processing in the processing restrictive environments in which the radar was implemented. The radar was thus successfully implemented and tested.

The main findings of the study are detailed below.

7.1.1 Passive radar performance

It was found that a general limiting factor in passive radar performance is dynamic range and not sensitivity [8], as is generally the case in active radar.

7.1.2 Optimum position

It was found that without knowledge of the terrain and its influence on EM propagation, the optimum position for a passive radar is between the area which it is intended to survey and the receiver. Furthermore, it was found that the longer the base line between the transmitter and the receiver, the further the range of the radar will be, assuming that signals' powers remain above the receiver's sensitivity.

7.1.3 Antenna and transmitter selection

It was found that the surveillance antenna for a passive radar should prioritise a high front-to-back gain ratio or should place nulls in the direction of the transmitter. This should be prioritized over forward gain.

The transmitter should be chosen to provide sufficient transmission power; such that dynamic range is the limiting performance factor rather than sensitivity.

7.1.4 Cancellation filter schemes

Several cancellation schemes were investigated. It was found that significant acceleration could be achieved by migrating the processing to a GPU, which executed up to 4.01 times faster than the equivalent CPU implementation.

By applying some restrictions to the definition of the cancellation filter, it was found that the memory requirements could be dramatically reduced by removing the need to explicitly store the cancellation matrix. Furthermore, this allowed the optimization of some functions and generally reduced the memory transfer requirements of functions. This allowed for execution time to be decreased by up to a further 21.01 times over the standard GPU implementation.

7.1.5 Matched filter processing

Various schemes were investigated for matched filter processing. It was shown under which conditions each is preferred. It was found that the matched filter could be migrated to the GPU and achieved a performance increase of up to 13.18 times faster than the CPU implementation.

7.1.6 Radar performance

A passive radar system was implemented using OTS hardware on an embedded processor as well as a laptop PC. Both systems were able to run in real time, i.e. they could process data as fast as/faster than it arrived. The laptop PC produced a significantly lower processing latency, but also resulted in a costlier system with significantly higher power requirements.

The radar was able to detect a target up to 90.35 km (bistatic range of 194.5 km) from the receiver. Of the 11 aircraft which had ADS-B transceivers (typically large passenger aircraft) and entered the maximum range of the radar, 10 were detected by the radar. The estimation error and probability of detection were estimated from data gathered. It was found that the radar had a bistatic range RMSE of 0.950 km which was less than the range resolution. It was found that the probability of detection for a target entering the maximum detectable range was 0.367 per CPI. The probability of detecting a target, within a specified bistatic range, dropped below 0.5 when the bistatic range was greater than 137.2 km.

The proposed system to other systems in literature and it was found that the proposed system cost approximately a quarter, had an 8.72% shorter detection range, and required approximately a quarter of the processing time of the most comparable system

It was, therefore, shown that with careful design and deployment, a low cost passive radar can be built using OTS components and can achieve reasonable performance. While the performance was not that of an active radar, or high performance passive radar, a low cost passive radar has been shown to be an alternative where power and cost are limiting factors.

7.2 FUTURE WORK

While a low cost passive radar system has been shown to be functional, its probability of detection was less than optimal. Improvements and future considerations are shown below.

7.2.1 Analogue DPI cancellation

The digital DPI and clutter cancellation implemented is near optimal in its ability to remove DPI and clutter. The system, however, is now limited by the quality of the sampled data. To this end, further cancellation needs to take place before conversion to the digital domain. In order to accomplish this objective, adaptive analogue DPI cancellation is required. If integrated into the system, analogue cancellation would result in improved performance, robustness and versatility by decreasing the dynamic range of signal components fed to the receiver.

7.2.2 Improved filtering

As discussed in Section 3.3, OTS receivers are not generally optimised for or support narrow analogue bandwidths. As such, improved pre-AGC analogue filters would allow the receiver to better utilize the ADC and improve the effective dynamic range achievable. Such an implementation has been addressed in literature [7]

7.2.3 Unambiguous target position

Currently the system is only capable of detecting the bistatic range, and bistatic range rate of a target. On its own, this information in not particularly valuable in real world scenarios. The bistatic range defines the position of a target on an ellipsoid [42]. In order to determine the position of a target, its direction from the receiver is also required. This can be achieved by incorporation direction finding or by employing a multistatic system [16].

7.2.4 Independent channel gain

On the current receiver, the gain of each channel has to be set to the same level. The issue arises when two channels have vastly differing amplitudes. Such a case exists between the reference and surveillance channels. As such, the surveillance channel's ADC could not be fully utilized without clipping the reference channel. Independent gain control would alleviate this issue and result in better digitisation of the surveillance channel.

REFERENCES

[1] P. Howland, “Editorial: Passive radar systems,” *IEE Proceedings - Radar, Sonar and Navigation,* pp. 105-106, 2005.

[2] F. Colone, D. O'Hagan, P. Lombardo and C. Baker, “A Multistage Processing Algorithm for Disturbance Removal and Target Detection in Passive Bistatic Radar,” *IEEE Transactions on Aerospace and Electronic Systems,* vol. 45, no. 2, pp. 698-722, 2009.

[3] J. Palmer, D. Cristallini and H. Kuschel, “Opportunities and current drivers for passive radar research,” in *2015 IEEE Radar Conference,* 2015.

[4] H. Kuschel and D. O'Hagan, “Passive radar from history to future,” in *11-th INTERNATIONAL RADAR SYMPOSIUM,* 2010.

[5] M. Inggs, C. Tong, R. Nadjiasngar, G. Lange, A. Mishra and F. Maasdorp, “Planning and design phases of a commensal radar system in the FM broadcast band,” *IEEE Aerospace and Electronic Systems Magazine,* vol. 29, no. 7, pp. 50-63, 2014.

[6] D. O’Hagan, H. Kuschel, J. Heckenbach and M. Ummenhofer, “Signal reconstruction as an effective means of detecting targets in a DAB-based PBR,” in *11th International Radar Symposium (IRS),* 2010.

[7] M. Inggs, R. Geschke, J. Coetser and D. O'Hagan, “High sensitivity fixed tuned direct conversion receiver for FM band commensal radar,” in *2015 IEEE Radar Conference,* 2015.

[8] H. Harms, J. E. Palmer, S. J. Searle and L. M. Davis, “Impact of quantization on passive radar target detection,” in *IET International Conference on Radar Systems (Radar 2012),* 2012.

[9] J. Palmer and S. Searle, “Evaluation of adaptive filter algorithms,” in *2012 IEEE Radar Conference (RADAR),* 2012.

[10] S. Heunis, Y. Paichard and M. Inggs, “Passive radar using a software-defined radio platform and opensource software tools,” in *2011 IEEE RadarCon (RADAR),* 2011.

[11] C. Tong and J. Coetser, "A minimal architecture for real-time, medium range aircraft detection using FM-band illuminators of opportunity," in *2015 IEEE Radar Conference (RadarCon)*, 2015.

[12] P. Howland, D. Maksimiuk and G. Reitsma, "FM radio based bistatic radar," *IEE Proceedings - Radar, Sonar and Navigation,* vol. 152, no. 3, pp. 107-115, 2005.

[13] C. A. Tong, "A Scalable Real-time Processing Chain for Radar Exploiting Illuminators of Opportunity," University of Cape Town, Cape Town, 2014.

[14] K. Kulpa, "Multi-static entirely passive detection of moving targets and its limitations," *IEE Proceedings - Radar, Sonar and Navigation,* vol. 152, no. 3, pp. 169-173, 2005.

[15] M. Duan and W. Koch, "Multistatic target tracking for non-cooperative illumination by DAB/DVB-T," in *RADAR '08. IEEE Radar Conference, 2008.*, 2008.

[16] M. Inggs and Y. Paichard, "Multistatic Passive Coherent Location radar systems," in *EuRAD 2009. European Radar Conference, 2009.*, 2009.

[17] P. E. Howland, "Target tracking using television-based bistatic radar," *IEE Proceedings - Radar, Sonar and Navigation,* vol. 146, no. 3, pp. 166-174, 1999.

[18] M. Rofouei, T. Stathopoulos, S. Ryffel, W. Kaiser and M. Sarrafzadeh, "Energy-aware high performance computing with graphic processing units," 2008.

[19] "IEEE Standard Radar Definitions," *IEEE Std 686-1997,* pp. i-35, 1998.

[20] F. Colone, R. Cardinali, P. Lombardo, O. Crognale, A. Cosmi, A. Lauri and T. Bucciarelli, "Space-time constant modulus algorithm for multipath removal on the reference signal exploited by passive bistatic radar," *IET Radar, Sonar & Navigation,* vol. 3, no. 3, pp. 253-264, 2009.

[21] M. Richards, J. Scheer, J. Scheer and W. Holm, *Principles of modern rada*r, SciTech Publishing, Incorporated, 2010.

[22] P. Falcone, F. Colone, A. Macera and P. Lombardo, "Two-dimensional location of moving targets within local areas using WiFi-based multistatic passive radar," *IET Radar, Sonar Navigation,* vol. 8, no. 2, pp. 123-131, 2014.

[23] P. Beasley, "The Influence of Transmitter Phase Noise on FMCW Radar Performance," in *2006. EuRAD 2006. 3rd European Radar Conference*, 2006.

[24] H. D. Griffiths and N. R. W. Long, "Television-based bistatic radar," *IEE Proceedings For Communications, Radar and Signal Processing*, vol. 133, no. 7, pp. 649-657, December 1986.

[25] P. E. Howland, "Television based bistatic radar," University of Birmingham, Birmingham, 1997.

[26] NATO Defense Research Group, in *Symposium on 'passive and noise radar'.*

[27] H. D. Griftiths and C. J. Baker, "Passive coherent location radar systems. Part 1: performance prediction," *IEE Proceedings - Radar, Sonar and Navigation*, vol. 152, no. 3, pp. 153-159, 2005.

[28] A. Lauri, F. Colone, R. Cardinali, C. Bongioanni and P. Lombardo, "Analysis and Emulation of FM Radio Signals for Passive Radar," in *2007 IEEE Aerospace Conference*, 2007.

[29] C. Bongioanni, F. Colone and P. Lombardo, "Performance analysis of a multi-frequency FM based Passive Bistatic Radar," in *2008 IEEE Radar Conference*, 2008.

[30] J. Brown, K. Woodbridge, A. Stove and S. Watts, "Air target detection using airborne passive bistatic radar}," *Electronics Letters*, vol. 46, no. 20, pp. 1396-1397, 2010.

[31] M. Malanowski, K. S. Kupla, P. Samczynski, J. Misiurewicz and J. Kupla, "Long range FM-based passive radar," in *IET International Conference on Radar Systems (Radar 2012)*, 2012.

[32] H. Griffiths and C. Baker, "Measurement and analysis of ambiguity functions of passive radar transmissions," in *2005 IEEE International Radar Conference*, 2005.

[33] L. J. Garry, G. E. Smith and C. J. Baker, "Direct signal suppression schemes for passive radar," in *Signal Processing Symposium (SPSympo), 2015*, 2015.

[34] M. Meyers, "Radio and Development in Africa.," International Development Research Centre (IDRC) of Canada, 2009.

[35] Institute of Electrical and Electronics Engineers, Radatz, J. and IEEE Computer Society. Standards Coordinating Committee, The IEEE Standard Dictionary of Electrical and Electronics Terms, IEEE, 1997.

[36] M. Inggs, C. Tong, D. O'Hagan, U. Böinger, U. Siegenthaler, C. Schüpbach and H. Pratisto, "Noise jamming of a FM band commensal radar," in *2015 IEEE Radar Conference*, Johannesburg, 2015.

[37] A. Haeff, "Minimum Detectable Radar Signal and Its Dependence upon Parameters of Radar Systems," *Proceedings of the IRE,* vol. 34, no. 11, pp. 857-861, 1946.

[38] E. C. Ifeachor and B. W. Jervis, *Digital Signal Processing: A Practical Approach*, 2nd ed., Prentice Hall, 2002.

[39] B. Lathi and Z. Ding, *Modern Digital and Analog Communication Systems*, 4th ed., Oxford, 2009.

[40] IEEE, "IEEE Standard for Terminology and Test Methods of Digital-to-Analog Converter Devices," *IEEE Std 1658-2011,* pp. 1-126, 2012.

[41] M. Inggs, G. Lange and Y. Paichard, "A quantitative method for mono- and multistatic radar coverage area prediction," in *2010 IEEE Radar Conference,* 2010.

[42] F. Ulaby, E. Michielssen and U. Ravaioli, *Fundamentals of Applied Electromagnetics*, 6th ed., New Jersey: Pearson Education, 2010.

[43] N. J. Willis and H. D. Griffiths, *Advances in bistatic radar,* vol. 2, SciTech Publishing, 2007.

[44] A. David, C. Brousseau and A. Bourdillon, "Simulations and measurements of a radar cross section of a Boeing 747-200 in the 20–60 MHz frequency band," *Radio Science,* vol. 38, no. 4, pp. n/a-n/a, 2003.

[45] M. Malanowski and K. Kulpa, "Digital beamforming for Passive Coherent Location radar," in *RADAR '08. IEEE Radar Conference, 2008.*, 2008.

[46] C. Tong, M. Inggs and F. Maasdorp, "Performance improvements using the separated reference configuration for a multi-static FM broadcast band radar system," in *2013 International Conference on Radar*, 2013.

[47] Intel, “Intel® 64 and IA-32 Architectures Software Developer’s Manual Volume 3A: System Programming Guide, Part 1,” 2015.

[48] L. B. Das, *The X86 Microprocessors: Architecture and Programming (8086 To Pentium)*, Pearson Education, 2010.

[49] K. Jotwani and N. Hwang, *Advanced Computer Architecture Parallelism, Scalability, Programmability*, New Delhi: Mc Graw Hill Education, 2011.

[50] N. Wilt, *The CUDA Handbook: A Comprehensive Guide to GPU Programming*, Pearson Education, 2013.

[51] Intel Corporation, “User's Guide for Intel® Math Kernel Library 11.3 Update 1 for Windows* OS,” 2015. [Online]. Available: https://software.intel.com/en-us/mkl-for-windows-userguide. [Accessed 10 February 2016].

[52] Intel, “Intel® 64 and IA-32 Architectures Software Developer’s Manual Volume 2 (2A, 2B & 2C): Instruction Set Reference, A-Z,” Intel, 2015.

[53] R. Y. Kain, *Advanced computer architecture A systems design approach*, New Jersey: Prentice-Hall inc., 1996.

[54] Nvidia, “CUDA C Programming guide,” 1 September 2015. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide. [Accessed 12 October 2015].

[55] C. Gregg and K. Hazelwood, “Where is the data? Why you cannot debate CPU vs. GPU performance without the answer,” in *2011 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2011.

[56] W. du Plessis, “Disposable EW – Keeping Pace with Rapid Advances,” in *Proceedings of the Aardvark Roost Biennial International EW Conference 2015*, Pretoria, 2015.

[57] Nvidia Corporation, “Jetson TK1 Development Kit Spesification,” 2014.

[58] S. Haykin, *Adaptive Filter Theory*, 4th ed., Pearson Education, 2008.

[59] S. Searle, L. Davis and J. Palmer, “Signal processing considerations for passive radar with a single receiver,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[60] Nvidia, "cuBLAS Library," September 2015. [Online]. Available: http://docs.nvidia.com/cuda/pdf/CUBLAS_Library.pdf. [Accessed 13 February 2016].

[61] L. Trefethen and D. Bau, *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 1997.

[62] Nvidia, "cuSOLVER Library," 1 September 2015. [Online]. Available: http://docs.nvidia.com/cuda/pdf/CUSOLVER_Library.pdf. [Accessed 17 February 2016].

[63] Nvidia, "cuFFT Library User's Guide," 1 September 2015. [Online]. Available: http://docs.nvidia.com/cuda/pdf/CUFFT_Library.pdf. [Accessed 17 February 2016].

[64] M. Frigo and S. G. Johnson, "FFTW," 25 November 2012. [Online]. Available: http://www.fftw.org/fftw3.pdf. [Accessed 17 February 2016].

[65] R. C. Whaley and J. J. Dongarra, "Automatically Tuned Linear Algebra Software (ATLAS)," in *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, 1998.

[66] O. Rioul and P. Duhamel, "Fast algorithms for discrete and continuous wavelet transforms," *IEEE Transactions on Information Theory,* vol. 38, no. 2, pp. 569-586, 1992.

[67] M. Cherniakov, "FMCW like approach," in *Bistatic Radars: Emerging Technology*, Wiley, 2008, pp. 301-303.

[68] F. D. V. Maasdorp, M. R. Inggs and R. Nadjiasngar, "Target tracking using Doppler-only measurements in FM broadcast band commensal radar," *Electronics Letters,* vol. 51, no. 19, pp. 1528-1530, 2015.

[69] A. Benavoli and A. Di Lallo, "Why should we use particle filtering in FM band passive radars?," in *European Radar Conference, 2008. EuRAD 2008*, 2008.

[70] H. Li, X. Ji and G. Zhao, "TOA-based target tracking using improved particle filter in passive bistatic radar with glint noise," in *6th International Congress on Image and Signal Processing (CISP), 2013*, 2013.

[71] D. Tenne and T. Singh, “Optimal design of α-β- (γ) filters,” in *Proceedings of the 2000 IEEE American Control Conference, 2000.*, 2000.

[72] Radio Frequency Industries, “Broadcast Band Yagi Base Direcional Antennas (88-110 MHz.),” 24 Febuary 2010. [Online]. Available: http://www.rfind.co.za/broadcast%20band%20yagi%20%20base%20antennas.pdf. [Accessed 20 October 2015].

[73] D. McCallie, J. Butts and R. Mills, “Security analysis of the ADS-B implementation in the next generation air transportation system,” *International Journal of Critical Infrastructure Protection,* vol. 4, no. 2, pp. 78--87, 2011.

[74] F. Maasdorp, J. Cilliers, M. Inggs and C. Tong, “Simulation and measurement of propeller modulation using FM broadcast band commensal radar,” *Electronics Letters,* vol. 49, no. 23, pp. 1481-1482, 2013.

[75] G. H. Golub and C. F. Van Loan, “Symmetric Indefinite Systems,” in *Matrix Computations*, Johns Hopkins University Press, 1996, pp. 161-174.

[76] Z. Zlatev and H. B. Nielsen, “Solving large and sparse linear least-squares problems by conjugate gradient algorithms,” *Computers & Mathematics with Applications,* vol. 15, no. 3, pp. 185 - 202, 1988.

[77] R. Fry and D. Gray, “CLEAN deconvolution for sidelobe suppression in random noise radar,” in *2008 International Conference on Radar*, 2008.

[78] B. Friedlander, “Lattice filters for adaptive processing,” *Proceedings of the IEEE,* vol. 70, no. 8, pp. 829-867, 1982.

ADDENDUM A DERRIVATION OF COMPUTATIONAL COMPLEXITIES

Relevant linear algebra routines are introduced and derived below.

A.1. LU FACTORIZATION

LU factorization can be used to solve a system of equations by factorising a matrix into an upper and lower triangular form, such that

$$C = LU \tag{A.1}$$

where

L is a lower triangular matrix, and

U is an upper triangular matrix which as diagonal elements equal to one.

The LU factorisation can be performed using Gaussian elimination [61], where the elimination matrices are stored and used to determine U. However, this is potentially numerically unstable, as a divide by zero (or even by a small number) causes numerical errors.

An example of this is described here. Assume the factorisation of

$$C = \begin{bmatrix} 25 & 5 & 1 \\ 64 & 12.8 & 1 \\ 144 & 12 & 1 \end{bmatrix} \tag{A.2}$$

into LU form by replacing C with L. The process is initialised by calculating the elimination matrix for the first iteration ($i = 1$),

$$E_i = I - \frac{1}{C_{ii}} C_{i+1:m,i} \tag{A.3}$$

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -2.56 & 1 & 0 \\ -5.76 & 0 & 1 \end{bmatrix}. \tag{A.4}$$

By multiplying the elimination with C with E_i C is updated such that

$$C = \begin{bmatrix} 25 & 5 & 1 \\ 0 & 0 & -1.56 \\ 0 & -16.8 & -4.76 \end{bmatrix}. \tag{A.5}$$

The second elimination matrix becomes undefined as

$$E_2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -C_{32}/C_{22} & 1 \end{bmatrix}, \tag{A.6}$$

which results in a divide by zero error. U is calculated by

$$U = \left(\prod_{i=1}^{M-1} E_i \right)^{-1}. \tag{A.7}$$

The non-diagonal elements of U can alternatively be calculated by

$$u_{ij} = -e_{ij} \tag{A.8}$$

where

u_{xy} is an element of U, and

$e_{xy}$is an element of E_y.

In order to remedy the numerical instability in LU factorisation, the factorisation is modified to [61]

$$PC = LU \tag{A.9}$$

where,

P is a perturbation matrix.

This can be solved via Gaussian elimination with partial pivoting. In this method a row interchange is performed so that the magnitude of the elimination multiplier is minimised. This is performed by performing row interchanges such that C_{ii} in each step is maximised. This modified LU factorisation is both more stable and more numerically accurate.

The factorization of C has an approximate time cost of $\frac{2}{3}M^3$. Using Gaussian elimination this requires $\frac{1}{3}M^3 - M^2 + \frac{2}{3}M$ multiplications and the same number of additions. The number of divisions required is $\frac{1}{2}M^2 - \frac{1}{2}M$. The flop count for LU Factorization is therefore $\frac{8}{3}M^3 - M^2 - \frac{5}{3}M$.

Forward substitution is then used to solve for a vector z in

$$Lz = d \tag{A.10}$$

x is then found by backward substituting

$$Ux = z. \tag{A.11}$$

Finally, the filter output is found by

$$\hat{b} = b - Ax \tag{A.12}$$

The algorithmic cost for LU factorization least squares is given in Table A.A.1.

Table A.A.1 Algorithmic cost for LU factorisation least-squares.

Equation	Time cost	Flop count
$\boldsymbol{d = A^H b}$	$2NM$	$8NM$
$\boldsymbol{C = A^H A}$	$2NM^2$	$8NM^2$
$\boldsymbol{LU = C}$	$\frac{2}{3}M^3$	$\frac{8}{3}M^3 - M^2 - \frac{5}{3}M$
$\boldsymbol{Lz = d}$	M^2	$8NM$
$\boldsymbol{Ux = z}$	M^2	$8NM^2$
$\boldsymbol{\hat{b} = b - Ax}$	$2NM + N$	$8NM + 2N$
Total	$\frac{2}{3}M^3 + 2NM^2 + 4NM + 2M^2 + 2N$	$\frac{8}{3}M^3 + 16NM^2 + 24NM + 2N - M^2 - \frac{5}{3}M$

A.2. CHOLESKY FACTORIZATION

The Cholesky factorisation factorises a positive-definite Hermitian matrix [61] such that

$$A = LL^H \tag{A.13}$$

where

L is a lower triangular matrix.

The factorisation therefore has a time cost of $\frac{1}{3}M^3 + \frac{1}{2}M^2 + \frac{1}{6}M$, which is approximated as $\frac{1}{3}M^3$. The number of multiplications required are $\frac{1}{6}M^3 + \frac{1}{2}M^2 + \frac{1}{3}M$, and the number of additions are $\frac{1}{6}M^3 - \frac{1}{6}M$. The total FLOPs required for Cholesky factorisation is therefore $\frac{4}{3}M^3 + 3M^2 + \frac{5}{3}M$.

The filter weight vector, x, is then found by following a similar methodology to LU factorisation least squares, where U is replaced by L^H. The algorithmic cost for Cholesky factorisation is shown in Table A.2.

Table A.2 Algorithmic cost of Cholesky factorisation least squares.

Equation	Time cost	Flop count
$\boldsymbol{d = A^H b}$	$2NM$	$8NM$
$\boldsymbol{C = A^H A}$	$2NM^2$	$8NM^2$
$\boldsymbol{LL^H = C}$	$\frac{1}{3}M^3$	$\frac{4}{3}M^3 - M^2$
$\boldsymbol{Lz = d}$	M^2	$4M^2 + 10M$
$\boldsymbol{Ux = z}$	M^2	$4M^2 + 10M$
$\boldsymbol{\hat{b} = b - Ax}$	$2NM + N$	$8NM + 2N$
Total	$\frac{1}{3}M^3 + 2NM^2 + 4NM + \frac{3}{2}M^2 + N$	$8NM^2 + \frac{4}{3}M^3 + 16NM + 7M^2 + 2N$

It should be noted that with real data, C is not guaranteed to be positive-semidefinite (within numeric precision). Hence, another factorisation method is required to ensure a solution can be found. Alternatives include LU or LDLT factorisation.

A.3. LDL FATORIZATION

LDL factorisation [75] factorises a Hermatian matrix into

$$A = LDL^H \tag{A.14}$$

where

L is a unit lower triangular matrix, and

D is a diagonal matrix.

The factorisation can be completed with approximately $1/3N^3$ complex operations. However, in order to ensure numerical stability a pivoting technique is required [75]. This can be achieved using Baunch-Kaufman diagonal pivoting. This increases the computational requirements by $O(N^2)$ complex operations, and decreases the arithmetic intensity of the factorization.

A.4. QR FACTORIZATION

The least squares problem can be solved using the QR factorization [61],

$$A = QR \tag{A.15}$$

where

Q is an orthogonal matrix such that $Q^{-1} = Q^H$, and

R is an upper triangular matrix.

QR least squares is possibly the most common numerical linear least squares solver used by libraries when solving for rectangular linear least squares. It is implemented by, LAPACK and MATLAB as the default solver for rectangular systems. This is, however, primarily due to its numerical stability and accuracy.

The QR decomposition can be calculated using the Householder algorithm [61]. R is found by sequentially updating A with $k = 1,2, \dots, M$ by defining

$$g = A_{k:N,k} \tag{A.16}$$

then v_k is defined as

$$v_k = \operatorname{sign}(g_1)\|g\|e_1 + g \tag{A.17}$$

where

e_1 is a $m - k$ element vector with the first element equal to one, and the rest equal to zero.

v_k is then normalised by

$$v_k = \frac{v_k}{\|v_k\|} \tag{A.18}$$

and A is updated by

$$A_{k:N,k:M} = A_{k:N,k:M} - 2v_k\left(v_k^H A_{k:N,k:M}\right). \tag{A.19}$$

Once the algorithm is completed R is written in place of A, and the reflection vectors $v_1, v_2, \dots, v_M$ are stored.

In order to calculate the least-squares solution, a vector d is defined as

$$d = Q^H b. \tag{A.20}$$

Finally, x is found via backward substituting

$$Rx = d. \tag{A.21}$$

Given that the only place Q is needed for the algorithm is in (A.20), Q does not need to be explicitly calculated. Instead d can be calculated by sequentially applying

$$b_{k:N} = b_{k:N} - 2v_k(v_k^H b_{k:N}) \tag{A.22}$$

for $k = 1,2,\dots,n$, after which b has been overwritten by d.

A.4.1 Computational cost

The computational cost of QR least squares commences with the computational cost of (A.17). In the interest of clarity, $l = N - k + 1$ and $l_1 = M - k + 1$ is defined. Determining $||g||$ requires $2l$ operations, with half of those being composed of magnitude operations and half additions. As e_1 has only one non-zero element, only 1 multiplication and addition is required to form v_k. However, the entire g vector does need to be copied requiring l operations. Therefore, (A.17) requires $3l + 2$ operations to execute. Summing this over the n iterations results for $k = 1,2,\dots,M$ results in $3\left(MN - \frac{1}{2}M^2\right) + \frac{7}{2}M$ operations.

The normalisation of v_k then requires the calculation of the norm of v_k which requires $2l$ operations, and a division which can be represented as one division and $2l$ multiplications. When this is completed for all M iterations, the total time cost is $4NM - 2M^2 + 3M$ operations.

The bulk of the work in QR factorisation is performed by (A.19). First $v_k^H A_{k:N,k:M}$ requires the multiplication between a $1 \times l$ vector and an $l \times l_1$ results in a $1 \times l_1$ vector. This operation has an approximate time cost of $2l_1 l$, of which half are multiplications and half are additions. The multiplication between that vector and v_k requires a further $l \times l_1$ operations, resulting in a $l \times l_1$ matrix. The scalar multiplication and addition requires a further l, and $l \times l_1$ operations respectively. In total (A.19) requires $4(l \times l_1) + (l)$ operations, of which $2(l \times l_1)$ are complex multiplications, $2(l \times l_1)$ are complex additions, and l are scalar multiplications.

Summing this over the M iterations results in a time cost of $2NM^2 - \frac{2}{3}M^3 + 3MN - \frac{1}{2}M^2 + \frac{7}{6}M$. The total QR factorisation using the Householder algorithm therefore, has a time cost of $2NM^2 + 10MN - \frac{2}{3}M^3 - 4M^2 + \frac{23}{3}M$.

The calculation of d using the implicit Q, is performed by applying (A.22). This operation requires $2l$ operations to perform $v_k^H b_{k:N}$, a further l to multiply the result with v_k. A further l operations are required to perform the update. The scalar multiplication can be performed by applying the multiplication to the result of $v_k^H b_{k:N}$ which requires a single operation. Each iteration therefore has a time cost of $4l + 1$. Summing these overall n iterations results in a time cost of $4nm - 2n^2 + 3n$.

Equation (Rx=d) has a time cost of M^2. This requires M divisions, $M(M-1)/2$ multiplications and $M(M-1)/2$ subtractions. Thus, the backward substitution requires $4M^2 + 10M$ FLOPs.

The filter output requires calculating (5.1) which has a time cost of $2NM + N$. This requires NM multiplications, NM additions and N subtractions for a FLOP count of $8NM + 2N$.

The time cost and flop count for QR factorisation least squares is shown in Table A.3.
.

Table A.3 Arithmetic cost of QR factorisation least squares.

Equation	Time cost	FLOP count
$\boldsymbol{QR} = \boldsymbol{A}$	$2NM^2 + 10MN - \frac{2}{3}M^3 - 4M^2 + \frac{23}{3}M$	$8NM^2 + 37NM - \frac{8}{3}M^3 - \frac{29}{2}M^2 + \frac{115}{6}M$
$\boldsymbol{d} = \boldsymbol{bQ}^H$	$4NM - 2M^2 + 3M$	$16NM - 8M^2 + 14M$
$\boldsymbol{Rx} = \boldsymbol{d}$	M^2	$4M^2 + 10M$
$\widehat{\boldsymbol{b}} = \boldsymbol{b} - \boldsymbol{Ax}$	$2NM + N$	$8NM + 2N$
Total	$2NM^2 + 16NM - \frac{2}{3}N^3 - 5N^2 + \frac{32}{3}N + N$	$8NM^2 - \frac{8}{3}M^3 + 61NM - \frac{37}{2}M^2 + 2N + \frac{259}{6}M$

A.5. CONJUGATE GRADIENT LEAST SQUARES

Conjugate gradient least squares is an iterative least-squares algorithm which can be efficient for well-defined or large matrices [76]. This algorithm can be used to form an approximation of the filter coefficients with fewer computations than other algorithms [76] (such as factorisation). It has previously been implemented to reduce the computational load as opposed to a least squares algorithm [13].

The CGLS requires initialization which follows the procedure

$$r_0 = b - Ax_0 \tag{A.23}$$

$$s_0 = r_0 A^H \tag{A.24}$$

$$\gamma_0 = \text{norm}^2(s_0) \tag{A.25}$$

$$p_0 = s_0 \tag{A.26}$$

where

x_0 is the initial estimate for x.

The time cost and flop count for each of equations (A.23) to (A.26) is shown in Table A.4.

Table A.4 Algorithmic cost of CGLS initialization.

Equation	**Time cost**	**FLOP count**
$\boldsymbol{r_0 = b - Ax_0}$	$2NM + N$	$8NM + 2N$
$\boldsymbol{s_0 = r_0 A^H}$	$2NM$	$8NM$
$\boldsymbol{\gamma_0 = \mathrm{norm}^2(s_0)}$	$2M$	$4M$
$\boldsymbol{p_0 = s_0}$	M	$2M$
Total	$4NM + 2N + 2M$	$16NM + 4N + 4M$

Each iteration then refines the estimate of x and the remainder, i.e. the filter output on the ith iteration is stored in r_i. The iteration is carried out by [76]

$$q_i = Ap_{i-1} \tag{A.27}$$

$$\delta_i = \mathrm{norm}^2(q_i) \tag{A.28}$$

$$\alpha_i = \frac{\gamma_{i-1}}{\delta_i} \tag{A.29}$$

$$x_i = x_{i-1} + \alpha_i p_{i-1} \tag{A.30}$$

$$r_i = r_{i-1} + \epsilon_i q_i \tag{A.31}$$

$$s_i = r_i A^H \tag{A.32}$$

$$\gamma_i = \mathrm{norm}^2(s_i) \tag{A.33}$$

$$\beta_i = \frac{\gamma_i}{\gamma_{i-1}} \tag{A.34}$$

$$p_i = s_i + \beta p_{i-1}. \tag{A.35}$$

The time cost and flop count for equations (A.27)-(A.35) is given in Table A.5. If the algorithm is run for K iterations such that $i \in 1,2 \ldots K$. The total time cost can be seen to be $4NM + 2N + 2M + K(4NM + 4N + 6M + 2)$. The total FLOP count can be seen to be $16NM + 4N + 4M + K(16NM + 12N + 20M + 28)$.

Table A.5 Algorithmic cost of a CGLS iteration.

Equation	Time cost	FLOP count
$\boldsymbol{q_i = Ap_{i-1}}$	$2NM$	$8NM$
$\boldsymbol{\delta_i = norm^2(q_i)}$	$2N$	$4N$
$\boldsymbol{\alpha_i = \frac{\gamma_{i-1}}{\delta_i}}$	1	14
$\boldsymbol{x_i = x_{i-1} + \alpha_i p_{i-1}}$	$2M$	$8M$
$\boldsymbol{r_i = r_{i-1} + \epsilon_i q_i}$	$2N$	$8N$
$\boldsymbol{s_i = r_i A^H}$	$2NM$	$8NM$
$\boldsymbol{\gamma_i = norm^2(s_i)}$	$2M$	$4M$
$\boldsymbol{\beta_i = \frac{\gamma_i}{\gamma_{i-1}}}$	1	14
$\boldsymbol{p_i = s_i + \beta p_{i-1}}$	$2M$	$8M$
Total	$4NM + 4N + 6M + 2$	$16NM + 12N + 20M + 28$

A.6. CLEAN CANCELLATION

The CLEAN algorithm [77] operates by iteratively deconvolving the surveillance channel with the strongest component in the ARD. While simple to implement, it has been found to be ineffective as a cancellation scheme in literature [33]. Within the context of the cancellation filter, deconvolution is performed with the strongest column of A. The steps of the algorithm are

1. Set $\hat{s} = s$.
2. Create a return profile by using
$$w = A^H \hat{s}. \tag{A.36}$$
3. Select the largest return using
$$w_{\max} = \max(|w|). \tag{A.37}$$
4. Subtract the normalised component from $\hat{s}$ using
$$\hat{s} = \hat{s} - \frac{w_{\max}}{\|A_{\max}\|^2} A_{\max} \tag{A.38}$$

where

A_i is the i^{th} column of A.

5. Test for stop condition (possibly number of iterations). If passed exit, else return to step 2.

The CLEAN algorithm therefore operates in a loop defined by steps 2 to 5. The algorithmic cost of the loop is shown in Table A.6.

Table A.6 Algorithmic cost of CLEAN cancellation loop.

Equation	Time cost	FLOP count
$\boldsymbol{w} = \boldsymbol{A}^{\boldsymbol{H}}\hat{\boldsymbol{s}}$	$2NM$	$8NM$
$\boldsymbol{w}_{\mathbf{max}} = \mathbf{max}(\lvert\boldsymbol{w}\rvert)$	$2M$	$4M$
$\hat{\boldsymbol{s}} = \hat{\boldsymbol{s}} - \frac{\boldsymbol{w}_{\mathbf{max}}}{\lVert\boldsymbol{A}_{\mathbf{max}}\rVert^2}\boldsymbol{A}_{\mathbf{max}}$	$2N + 1$	$8N + 14$
Total	$2NM + 2N + 2M + 1$	$8NM + 8N + 4M + 14$

The calculation of $\lVert A_{\max}\rVert^2$ can be reduced to the calculation of $\lVert A_{\mathrm{i}}\rVert^2$ for $i = 1,2,\dots,M$, which can be calculated before the loop. The calculation of these normalisation weights has a time cost of $2NM$ and has a FLOP count of $4NM$.

Given K iterations of the loop, the time cost of the CLEAN algorithm is $2KNM + 2NM + 2KN + 2KM + K$. The FLOP count is $8KNM + 4NM + 8KN + 4KM + 14K$.

A.7. CORRELATIVE CANCELLATION

The computational complexity for Correlative cancellation is derived below. The time cost and FLOP count for each iteration are shown in Table A.7.

Table A.7 Algorithmic cost of correlative cancellation inner loop.

Equation	Time cost	FLOP count
$\boldsymbol{x}_{\boldsymbol{ij}} = \lvert\boldsymbol{b}_{\mathbf{0}}\rvert(\widehat{\boldsymbol{b}}_{\boldsymbol{ij}} \cdot \boldsymbol{A}_{\boldsymbol{i}}^{*})$	$2N + 1$	$8N + 4$
$\widehat{\boldsymbol{b}}_{(\boldsymbol{i}+\mathbf{1})\boldsymbol{j}} = \widehat{\boldsymbol{b}}_{\boldsymbol{ij}} - \boldsymbol{x}_{\boldsymbol{ij}}\boldsymbol{A}_{\boldsymbol{i}}$	$2N$	$8N$
Total	$4N + 1$	$16N + 4$

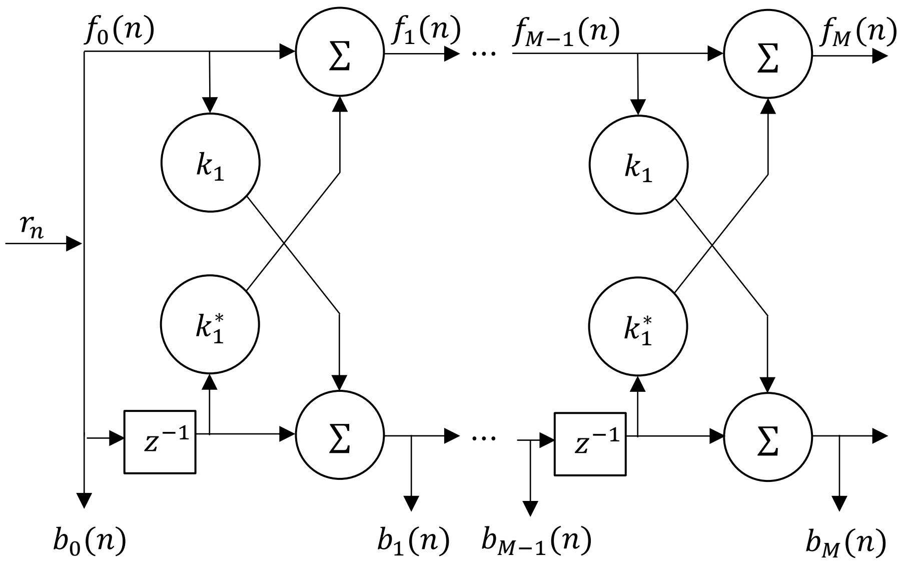
The total time cost required to apply the correlative cancellation filter is therefore $4KNM + KM + 2N$, and a FLOP count of $16KNM + 4KM + 6N$.

A.8. GRADIENT ADAPTIVE LATTICE FILTER

The GAL filter is a joint estimator made up of two parts [12] [78],

- an M stage lattice predictor, and
- an adaptive NLMS tapped delay line.

The filter follows the form shown in Figure 5.2. The lattice predictor section has a structure depicted in Figure A.1 [78].



Copyright © 1982, IEEE

Figure A.1 Diagram of a lattice predictor. Adapted with permission of IEEE, from **[78]**; permission conveyed through Copyright Clearance Center, Inc.

As can be seen from Figure A.1, the forward prediction error can be expressed as [58],

$$f_m(n) = f_{m-1}(n) + k_m^* b_{m-1}(n-1), \tag{A.39}$$

where

$f_m(n)$ is the mth forward prediction error for the nth sample,

$b_m(n)$ is the mth backward prediction error for the nth sample, and

$m = 1,2, \ldots, M$.

The backward prediction errors can be expressed as

$$b_m(n) = b_{m-1}(n-1) + k_m f_{m-1}(n). \tag{A.40}$$

The backward prediction errors are then fed to the tapped delay line. In order to update the filter weights an energy estimate is required and is defined as [58],

$$\epsilon_{m-1}(n) = \beta\epsilon_{m-1}(n-1) + (1-\beta)(|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2) \tag{A.41}$$

where

β is a scalar constant such that, $0 < \beta < 1$ which allows the filter to handle the non-stationary stochastic processes.

The reflection coefficients (k_m) are updated using [58]

$$k_m(n) = k_m(n-1) - \frac{\mu}{\epsilon_{m-1}(n)}(f^*_{m-1}(n)b_m(n) + b_{m-1}(n-1)f^*_m(n)). \tag{A.42}$$

Once the backward errors are calculated, they are fed to the tapped delay line. The output of the tapped delay line taps are

$$y_m(n) = y_{m-1}(n) + h^*_m(n)b_m(n) \tag{A.43}$$

where

$h_m(n)$ is the m^{th} tap weight, and

$y_0(n) = 0$.

The output of the filter is defined as

$$\widehat{w}_n = y_M(n). \tag{A.44}$$

In order to adjust the tap weights error terms are then defined as

$$e_m(n) = s_n - y_m(n). \tag{A.45}$$

Next a normalizing factor is defined

$$||\boldsymbol{b}_m(n)||^2 = ||\boldsymbol{b}_{m-1}(n)||^2 + |b_m(n)|^2. \tag{A.46}$$

The filter weights are then adjusted via [58],

$$h_m(m+1) = h_m(m) + \frac{\mu}{||\boldsymbol{b}_m(n)||^2}b_m(n)e^*_m(n). \tag{A.47}$$

The computational complexity of the algorithm can be analysed by separating the filter into its two parts. The computational complexity for each stage of the lattice filter (including reflection coefficient update) is shown in Table A.8.

Each stage is calculated M times for each sample and each of N samples must pass through the lattice predictor. As such, the calculations in Table A.8 need to be repeated NM times. The lattice predictor has a time cost of $17NM$ and a FLOP count of $45NM$.

Table A.8 Algorithmic cost of an adaptive lattice stage.

Equation	**Time cost**	**FLOP count**
(A.39)	2	8
(A.40)	2	8
(A.41)	6	10
(A.42)	7	19
Total	17	45

The algorithmic cost for each tap for each sample is shown in Table A.9.

Table A.9 Algorithmic cost of an adaptive NLMS tapped line.

Equation	**Time cost**	**FLOP count**
(A.43)	2	8
(A.45)	1	2
(A.46)	2	4
(A.47)	4	10
Total	9	24

As the calculation of each tap must be repeated M times and for each of N samples, the totals in Table A.9 should be multiplied by NM. The total time cost of the filter is, therefore, $26NM$ and the flop count is $69NM$.

It was found that simpler adaptive filters, such as a NLMS filter [58], were completely ineffective at removing DPI and clutter. This result coincides with other similar studies in literature [12].

ADDENDUM B GPU MEMORY TYPES

B.1. PAGED MEMORY

Paged memory exists on the host. This is the standard memory type allocated by host programs. Paged memory refers to the memory's property of being able to be paged out. This allows the operating system to move the memory to a new address (normally to make space for large arrays that would otherwise not fit), or if memory runs out, the memory can be moved to another storage device such as the hard drive.

B.2. PINNED MEMORY

Pinned memory is an alternative form of host memory, but unlike paged memory, it cannot be moved or swapped out. This allows the GPU drivers to use this memory directly using dedicated hardware called direct memory access (DMA). The disadvantage of pinned memory is that the allocation of large amounts of pinned memory can cause system instability.

The use of pinned memory allows for the acceleration of copying memory from the host to a discrete device. Using pinned memory also allows for a wider range of access schemes such as zero-copy memory. This allows the GPU to access the host memory directly, without an explicit copy. There are very few motivations for doing this with a discrete GPU. In an embedded system, however, as depicted in Figure 4.8, the use of pinned memory can increase performance and reduce memory usage by avoiding memory copying. Pinned memory on an embedded system can reduce memory requirements by avoiding explicitly storing a copy of the data in global memory.

B.3. GLOBAL MEMORY

Global memory sits on the device memory (on the GPU card in a discrete system) and is typically high bandwidth memory, when compared to host memory types, which uses long bus lengths to increase memory bandwidth. This is the default memory allocated for the device.

Global memory is the default read and write memory space for kernels, and as such, efficient accessing of global memory can yield significant performance gains in memory-bound kernels. Global memory access attempts to group memory requests such that bus usage is maximised. This allows for the number of requests to the DRAM to be minimised, thus maximising the effective bandwidth achieved. The process of grouping memory requests in this way is known as coalescing, and is an important consideration when writing GPU kernels.

B.4. SHARED MEMORY

Shared memory exists on the GPU chip as an explicitly managed cache. It has the scope of a single thread block, and can be used to efficiently share information across threads, and to minimise global memory access.

The shared memory exists physically as a block per SM. The shared memory on each SM is divided into banks, 32 on all current compute capabilities [54]. Each bank can only handle one memory request (although broadcasting is supported). As such, bank conflicts can reduce access efficiency and degrade performance.

B.5. REGISTERS

A large register block is available on each SM, from which registers are allocated to each thread. This is the fastest form of memory available to a thread running on a kernel, but is a limited resource and its scope is only for a single thread.

Registers often limit the number of concurrent blocks which can reside on an SM for a given kernel, and so it is advised to minimise register usage and prefer alternative memory sources, such a shared memory [54].

B.6. CONSTANT MEMORY

Constant memory exists in the device memory, and is cached in a dedicated cache on each SM. Constant memory is limited in size (64 KB for all current compute capabilities [54]), but can be read by all threads in a kernel. As such, constant memory is optimised for broadcasting data to threads, but cannot be written to inside a kernel.

B.7. TEXTURE MEMORY

Texture memory resides in device memory, and like constant memory is read only, cached on chip in a dedicated cache, and can be read by all threads. The main advantage of using texture memory is the way in which it is optimised.

Unlike global and constant memory, texture memory is not optimised for linear memory access (coalesced access), but rather for spatially near accessing with regard to a 2D array. This memory type is primarily used for the efficient rendering of textures (which are generally stored in a 2D array), however certain applications may find benefit in the non-linear access optimization.

ADDENDUM C EVALUATION METRICS

Before evaluating the algorithms to be used, the algorithm requirements and performance limiting factors are defined on three levels.

C.1. ARITHMETIC METRICS

These metrics define the requirements of the algorithm from an arithmetic perspective, i.e. the number of arithmetic operations required to perform a function.

Time cost

Here we define the complex operation, which is an elementary operation between two complex numbers or between a complex number and a real number. These include, but are not limited to: addition, subtraction, multiplication, division, exponents and magnitude operations. It is these basic operations which form the base of complex algorithms. Time cost has little bearing on execution time due to the differences in the implementation of these operations.

FLOPs

FLOPs refers to the number of floating point operations. Here the complex operation itself is not recognised, but rather the scalar operations required to perform a complex operation. In most floating point architectures these operations require one instruction, and as such is an indication of the number of instructions required to perform an operation.

C.2. ARCHITECTURAL METRICS

Architectural metrics include the typical performance limiting characteristics on processor architectures. These include memory requirements, register, and processor specific limitations. While this applies to the CPU implementations, the purpose of these metrics serves to provide insight into what facets of the algorithm limit its effectiveness on a GPU.

Registers per thread/block

This refers to the number of hardware registers the driver needs to allocate per thread/block. This can limit the size of the possible block, or the occupancy of the GPU.

Shared memory per block

Shared memory is a cache local to each block of threads. It is a limited resource and as such can limit the occupancy or prevent the execution of a kernel.

Implementative metrics

Implementative metrics refer to the performance achieved in an implementation of each algorithm. As such the results are unique to each hardware configuration.

Execution time

This is the average time taken to apply the algorithm. This includes (if applicable) the time required to transfer memory to the processing platform.

Maximum throughput

This is the theoretical maximum sample rate at which the algorithm could process data in real-time.

FLOPS

FLOPS is the floating point operations per second. This refers to the average achieved FLOPS for an algorithm or kernel by a particular implementation. Note that this differs from FLOPs as FLOPs is the plural of FLOP while FLOPS represents the number FLOPs per second.

Occupancy

Occupancy refers to the average percentage of a GPU which is utilized. Thus occupancy is a form of measuring the efficiency with which the GPU is utilized, an important consideration when optimizing an implementation for a specific platform.