Degree of Bachelor of Science in Engineering by coursework and research

Using a GPU as a co-processor for a PC-based PCL System

André Jan Schaafsma

A project report submitted to the Department of Electrical Engineering, University of Cape Town, in partial fulfilment of the requirements for the degree of Bachelor of Science in Engineering.

Cape Town, October 2007

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Bachelor of Science in Engineering in the University of Cape Town. It has not been submitted before for any degree or examination in any other university.

Signature of Author .....

Cape Town

22 October 2007

Abstract

Passive Coherent Location is a method of radar which exploits so-called 'illuminators of opportunity' in order to detect targets. The implementation of the signal processing aspects on a PC of a PCL system is explored, as well as the possible performance enhancements from using a consumer-grade graphics card's GPU as a co-processor for the processing tasks.

A number of aspects of the GPU's performance are explored, specifically regarding memory transfer capabilities.

While the GPU was found to provide a significant speed advantage in terms of processing time, its performance is still limited by current memory technology.

Contents

List of Figures

List of Symbols

Δt	—	Time delay from target echo
c	—	Speed of light
f_d	—	Doppler Frequency
R	—	Target Range

Chapter 1

Introduction

1.1 Requirements

Analyse the applicability of modern consumer graphics hardware as a coprocessor in a PC based PCL system. Currently, the complexity of the signal processing which could be applied to the return signal is severely limited by the processing power available in a PC. To what extent would the addition of a consumer grade graphics processor improve the capabilities of a PC as a radar signal processing system?

Consider all aspects of performance, including data transfers, memory loading and the CPU power required to manage the GPU.

Deliverables:

- Benchmarks of data transfer between system RAM and graphics card RAM
- Benchmarks of the performance of basic radar signal processing operations on the graphics processor.
- A system performance comparison between a PC and a PC with a modern graphics processor for performing basic radar signal processing
- C or C++ code for benchmarks

1.2 Requirements Review

A passive coherent location system is a type or radar system that makes use of “illuminators of opportunity”in the atmosphere, such as radio and television signals, to detect targets. The signal processing aspect of this system can be rather processing intensive for the computer running the system.

Consumer grade graphics cards have the potential to lessen the computational load on the computer through the use of their graphics processors and pipelines.

The extent of this performance enhancement will provide important insight into current research into PCL, particularly with regards to examining the costs of implementing such a system.

In order to investigate the performance of such a system, several steps need to be taken. These are:

1. Investigate the different methods of implementing a PCL system which have been developed.
2. Decide which method would be the most suited to this application.
3. Develop a high-level algorithm for this method in Matlab.
4. Investigate the accuracy of this implementation.
5. Implement this algorithm in C/C++ to use a graphics card's GPU as a coprocessor, using the Rapidmind toolkit.
6. Test the accuracy of this system, as well as compare the performance to the same system running just on a CPU.
7. Use the Rapidmind toolkit to perform other required benchmarks, such as memory transfer times.

These tasks have been divided into chapters as follows:

1.2.1 Chapter Two: PCL and High-level Implementation

This chapter will be examining the techniques which have been suggested for implementing PCL systems and selecting an algorithm to be implemented.

In order to examine the suitability of the received signals for target location, the Radar Ambiguity Function will be used to analyse the effects that a target's range and velocity would have on the reflected signals received from the target.

Detection of a target is done through comparison to a reference signal. This reference signal is taken as the television/radio signal that's received directly from the station transmitter.

A small segment of this reference signal is windowed off and used as a matched filter. This windowed segment is also the one used to calculate the radar ambiguity function.

The windowed reference segment is used as a matched filter and cross-correlated with the received reflected signals in order to detect the location and velocity of the targets.

The code developed is tested by using a number of simple signals with known range and Doppler characterstics to determine accuracy. In addition, simulated data is used to test the suitability of such a high-level implementation to target location.

The code is able to determine the range and Doppler shifts of simple signals. However, a simulated environment results in relatively large data sets, which simply take too long to process in such a high-level implementation.

While such a high-level implementation is useful to test the accuracy of the algorithm, it will never be suited to an actual PCL implementation, due to the performance issues.

The Matlab code developed for this implementation can be found in Appendix A.

1.2.2 Chapter Three: PCL Implementation on a GPU

This chapter will deal with the implementation of the algorithm which was developed in the previous chapter. This algorithm will be implemented in C/C++ to run on a graphics card's GPU, using the Rapidmind toolkit.

The Matlab code developed in the previous chapter will need to be translated into C/C++ code. In addition, the mathematical operations need to be examined to see which would run optimally on the CPU and which would run optimally on the graphics card's GPU.

Once the implementation is in place, the system will need to be tested in order to examine its performance characteristics. The accuracy will be tested through the use of sample data obtained through a simulator developed by Marc Brooker.

The speed performance of the system will need to be compared to the speed of a similar system running purely on a PC's CPU. This will be done by running the same sets of test data on both implementations in order to test the speed performance.

The purpose of such an investigation is to determine the extent to which a GPU could speed up the performance of a PCL system and what kind of performance enhancement can be expected.

The code developed did not reach the level of maturity where testing was possible.

It is expected, however, that such an implementation would show a great improvement, due to the nature of the calculations involved in the algorithm.

1.2.3 Chapter Four: GPU Performance Benchmarks

This chapter will be examining the performance characteristics of the graphics card being used.

The characteristics which will be examined include the graphics card's power performance, memory transfer time between the computer's memory and the graphics card's memory, and the processing power of the GPU itself.

The memory transfer benchmarks will be performed through the use of the Rapidmind toolkit. The other performance characteristics will be examined through research.

Further GPU performance considerations will be examined, along with tests performed to examine these considerations.

It was found that the relatively low power consumption of a GPU offers a significant advantage over large-scale receivers normally used for radar detection. However, the power consumption of the entire system still needs to be taken into account, as this power only applies to the signal processing component of the system.

While the memory loading times yielded an insight into the expected time for memory loading, these benchmarks still need to be compared to some standard before yielding more insightful results. An example would be comparing this performance to that of another model graphics card.

A range of different benchmarking tests would also yield more insight, such as transfers of different data types.

While the parallel processing capabilities of a GPU have been mentioned as one of the biggest advantages when compared to a CPU, the memory aspect still restricts the performance.

1.2.4 Chapter Five: Results and Conclusions

This chapter will be presenting a summary of the results obtained from all the investigations. The results from the performance and memory transfer benchmarks will be presented.

Possibilities for further research will be suggested, such as expanding the PCL algorithm to include target classification, as well as tracking.

Chapter 2

PCL and High-level Implementation

2.1 Chapter Overview

This chapter will be examining the techniques which have been suggested for implementing PCL systems and selecting an algorithm to be implemented.

In order to examine the suitability of the received signals for target location, the Radar Ambiguity Function will be used to analyse the effects that a target's range and velocity would have on the reflected signals received from the target.

Detection of a target is done through comparison to a reference signal. This reference signal is taken as the television/radio signal that's received directly from the station transmitter.

A small segment of this reference signal is windowed off and used as a matched filter. This windowed segment is also the one used to calculate the radar ambiguity function.

The windowed reference segment is used as a matched filter and cross-correlated with the received reflected signals in order to detect the location and velocity of the targets.

2.2 Overview of Traditional Radar Systems[1]

The goal of any radar system is to detect targets from a distance. The method of detection differs from one implementation to another, but each revolved around the idea of electromagnetic waves being reflected off the targets of interest and then being received and analysed by the radar system.

2.2.1 Basic Principle

Traditional radar systems are also referred to as active radar systems.

These systems actively emit predetermined signals and use the returns generated from those signals reflecting off targets to investigate the properties of these targets.

These systems are normally classified into one of two categories. Pulsed radar (PR) systems emit pulses at predetermined moments in time. Continuous wave (CW) systems emit a constant stream of electromagnetic radiation.

2.2.2 Method of Target Detection

A traditional radar system consists of a transmitter and receiver located at the same place. In PR systems, the transmitter and receiver is often the same antenna, which is switched between transmit mode and receive mode during different phases. In CW systems, a separate receiver and transmitter is required.

Range Detection

The basic principle behind range detection involves transmitting a pulse and investigating the time taken for the pulse to return, in order to determine the distance from the transmitter to the target. The time delay between transmitting the pulse and receiving the echo represents the time it would take for an electromagnetic wave to travel to the target and back.

Using the time delay, the distance of a target can be calculated as follows:

$R = \frac{c\Delta t}{2}$

where R is the range in metres, c is the speed of light and Δt is the time delay.

In PR systems, a series of pulses is emitted at predetermined intervals. This could lead to ambiguity in the range detection, as an echo from a very distant target could arrive back at the receiver after the next pulse in the series is transmitted. The concept of range ambiguity and ways to resolve this will be explored later.

Velocity Detection

Detecting the velocity of a moving target is done through examination of the Doppler shifts introduced by the movement of the target. The Doppler shift is the change in the frequency of the transmitted pulse due to either compression by a target moving towards the receiver (closing target), or expansion caused by a target moving away from the receiver (opening target).

The Doppler frequency is related to target velocity and angle for a closing target as follows:

$f_d = \frac{2v}{\lambda}\cos\theta$

And for an opening target is:

$f_d = \frac{-2v}{\lambda}\cos\theta$

Velocity detection is performed by comparing the frequency of the received echoes to the frequency of the transmitted signals to determine the Doppler shift and, hence, the target's velocity.

2.3 Overview of PCL

2.3.1 Basic Principle

Passive Coherent Location is an implementation of radar that makes use of "illuminators of opportunity" to detect targets, as opposed to traditional radar systems which actively emit signals in order to detect targets. These illuminators of opportunity are signals which are always present in the atmosphere and can thus be exploited, such as television, radio and cellphone signals.[2]

2.3.2 Method of Target Detection

A number of different methods for target detection in passive systems have been suggested. The basic principle behind each method remains the same. A particular signal is chosen as a reference signal, this being a suitable illuminator of opportunity.

This signal is then used as a reference and compared to other signals at the same frequency and the similarities are used to determine the presence of any targets.

The details of the comparision process differs from one implementation to another. However, most implementations use an adaptation of the matched filter detection method, using the reference signal as the matched filter.

The biggest differences in implementations lie in the signal processing methods used. A wide variety of filtering methods are suggested, with Kalman filters being one of the most universally mentioned.

2.4 Comparision Between Traditional Radar and PCL

While both PCL and traditional radar system have their own sets of advantages and disadvantages, making each method suited to different applications. A few of these differences will be examined.

2.4.1 Active vs. Passive Detection

The active nature of traditional radar systems allows for the detection of a radar system, as the signals being emitted can be traced back to their origin. This can be problematic in military applications where stealth and secrecy is of prime importance.

A passive system, in contrast, exploits signals which are already present in the atmosphere. As the system is only receiving, not transmitting, this makes it next to impossible to pinpoint the location of a passive receiver.

In this regard, a passive system provides more security than an active system would. However, a passive system can only be employed in an area where suitable signals are present at all times.

2.4.2 Shape of Waveforms

In traditional radar systems, the shape of the waveforms being transmitted is predetermined. As a result, these waveforms can be optimised to produce the maximum possible accuracy and efficiency.

PCL is dependent on signals already present. As a result, designers of such systems are unable to control the quality of the signals used for detection. The system has to be able to adapt to a wide range of possible signals.

With regards to the ability to control the shape of the waveforms, and thus the quality of the waveforms being used for detection, traditional radar systems have a great advantage over PCL.

2.4.3 Cost of Implementation

As a traditional radar system needs to transmit signals of its own, the equipment required includes both transmitting and receiving hardware. The cost of a suitable transmitter can often be prohibitively expensive.

A passive radar system only requires equipment for receiving and signal processing. As a result, the cost is dependent only on the cost of the receiver and the complexity of the processing system.

The biggest advantage of passive systems over traditional systems is that of cost. And with the relatively low cost of consumer-grade graphics cards, examining this approach to the signal processing tasks can greatly reduce the overall cost of implementation for a PCL system.

2.5 Important Radar Concepts

There are a few concepts which are key to the functioning of the PCL algorithm to be implemented. These concepts will be briefly explored.

2.5.1 Matched Filter Detection

A matched filter is a type of filter which has the same impulse response as that of the signal which it is designed to detect. The filter is matched to the signal in order to take on the same characteristics.

The greatest advantage of a matched filter is that it produces the maximum possible SNR at its output when used to detect the signal it is matched to. This peak SNR is achieved, even when the signal is buried in white noise. This makes a matched filter the ideal type of filter to make use of when the characteristics of the desired signal are known.

In practice, it's often difficult to achieve a perfect matched filter. Some form of trade-off, in terms of accuracy, is necessary in order to compensate for any such imperfections. [1]

2.5.2 Radar Ambiguity Function

The radar ambiguity function provides insight into the characteristics of any signal when used as a matched filter. The function provides a look at the interference caused by a target's range and Doppler shift, when compared to a reference target.

The ambiguity function can be plotted as a two-dimensional graph showing the output of the matched filter over a range of different ranges and Doppler shifts.

The general form of the ambiguity function for a signal s(t) is as follows:

$| \chi(\tau; f_d) |^2 = | \int s(t) s*(t-\tau) e^{j2\pi f_d t} dt |^2$ [1]

This represents the auto-correlation of the signal over a range of different ranges and Doppler frequencies. This can be seen as the theoretical output of a matched filter over this selection of ranges and Doppler frequencies.

One of the properties of the ambiguity function is that its maximum value occurs at the origin. This maximum value represents the output of a reference target at zero range and Doppler shift, in other words the response which a matched filter would produced when compared with itself.

An example of a radar ambiguity plot can be seen in Figure 2.1. This clearly indicates the peak occurring at the origin, with the function falling off pretty rapidly. The contour plot, shown in Figure 2.2 provides a different perspective, with the peak clearly being shown at the origin.

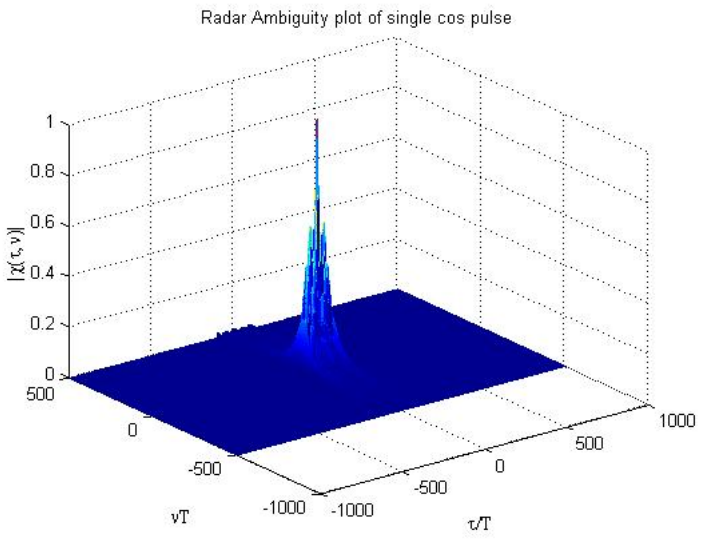

Figure 2.1: Radar Ambiguity plot of single cycle cos pulse

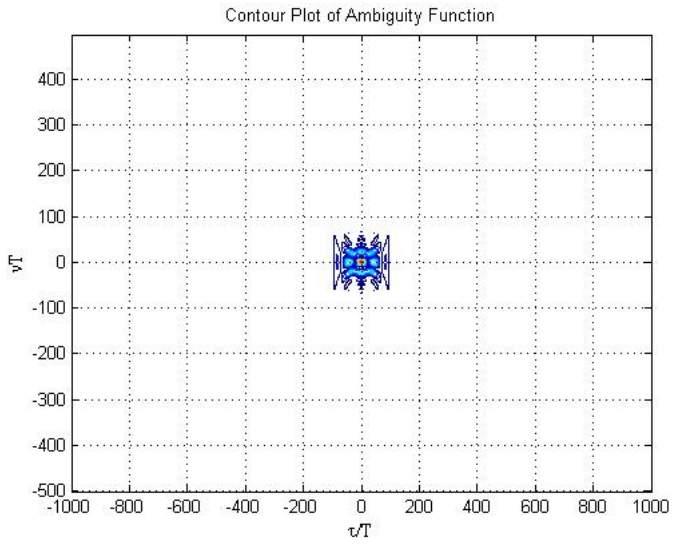

Figure 2.2: Contour plot of the radar ambiguity function of a single cycle cos pulse

2.6 PCL Algorithm to be Implemented

2.6.1 Basic Overview

The basic principle of a PCL system involves reception of two signals. One, a direct signal from the chosen illuminator, acts as a reference signal. The other signal is a combination of all the reflected signals received from the targets.

A brief overview of the steps involved in processing these signals is as follows:

1. Select a small portion of the reference signal to use as a matched filter.
2. Compare this matched filter to the reflected signals.
3. Use the radar ambiguity function to determine range and Doppler shift.
4. Use the range and Doppler shift to determine the location and velocity of the target relative to the receiver.

2.6.2 Detailed Overview

The steps outlined in the basic principle will now be explored in further detail. These steps are the ones which will be used to implement the algorithm.

1. Use a windowing function to isolate a small part of the reference signal.
2. Cross-correlate this fragment of the reference signal with the reflected signals.
3. Find the radar ambiguity function of the cross-correlation of these two signals.
4. Scan through this radar ambiguity function at different Doppler shifts to locate the peaks.

5. The largest peak will indicate the range and Doppler shift of the target.

6. Once the range and Doppler shift has been determined, the target's range and velocity can be calculated.

2.7 Description of the Matlab Code

The steps followed by the code are as follows:

1. The two signals (reference and return) are cross correlated in the time domain

2. An FFT is then performed on this cross correlated signal to translate it to the frequency domain

3. The radar ambiguity function is calculated and plotted both as a 3D plot and a contour plot

4. The locations of the peaks of the ambiguity function are located

The Matlab code can be found in Appendix A.

2.8 Testing of the Algorithm

2.8.1 Basic Waveforms

A few basic waveforms with known range and Doppler characteristics were used to test the basic functionality of the algorithm.

The first waveform used for testing purposes is a single cycle cos pulse, which is shown in Figure 2.3.

The return signal was first shifted in time, to simulate a target at a certain range. The return pulse from this can be seen in Figure 2.4

The same pulse was shifted in frequency to simulate the effects which a Doppler shift would have on the return signal. The return from this pulse can be seen in Figure 2.5

Finally, the return signal was shifted both in time and frequency to represent a moving target at a distance, inducing both a range shift and Doppler shift. This return pulse can be seen in Figure 2.6.

Each of these returns were cross correlated with the original pulse and the radar ambiguity function calculated. The results of these tests can be seen below.

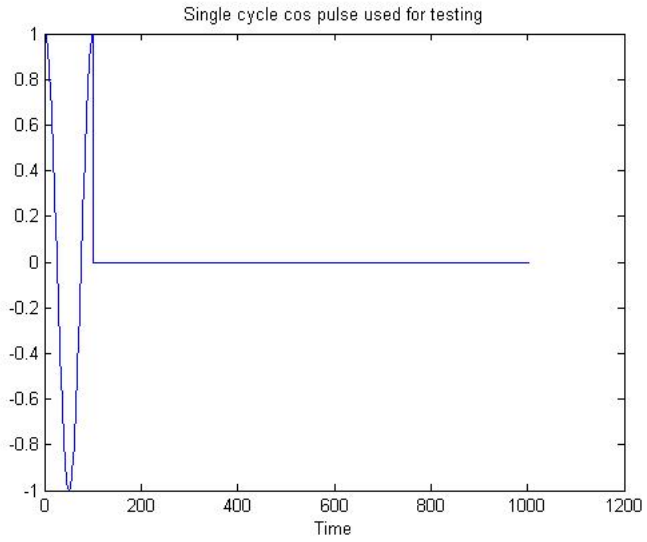

Figure 2.3: Single Cycle cos pulse used for testing

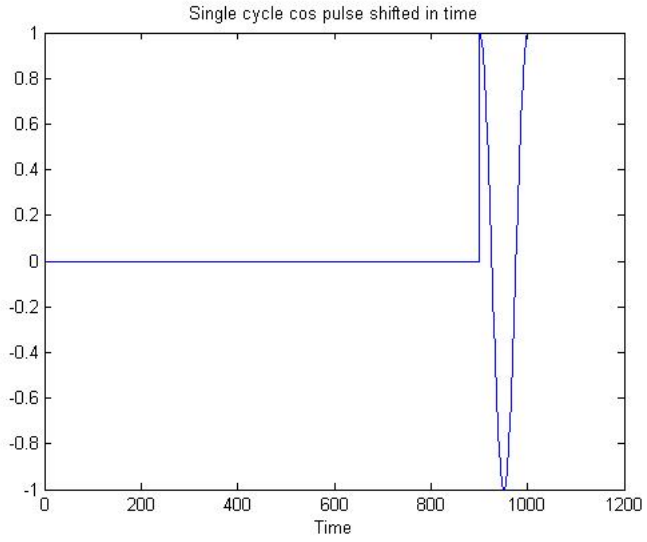

Figure 2.4: Single cycle cos pulse shifted in time

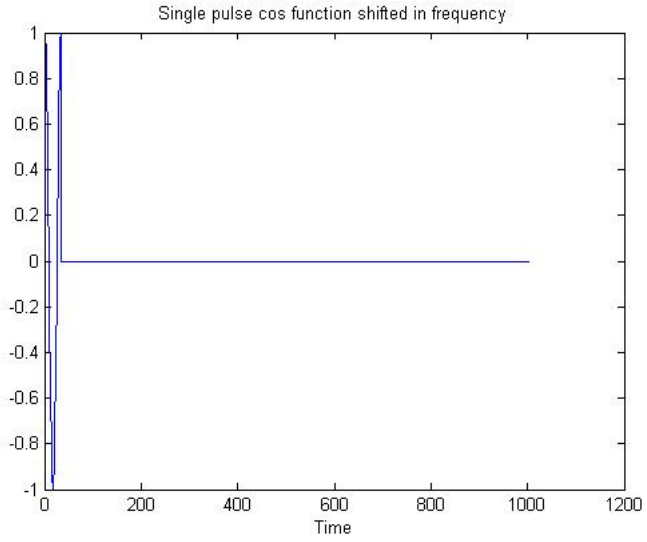

Figure 2.5: Single cos pulse shifted in frequency

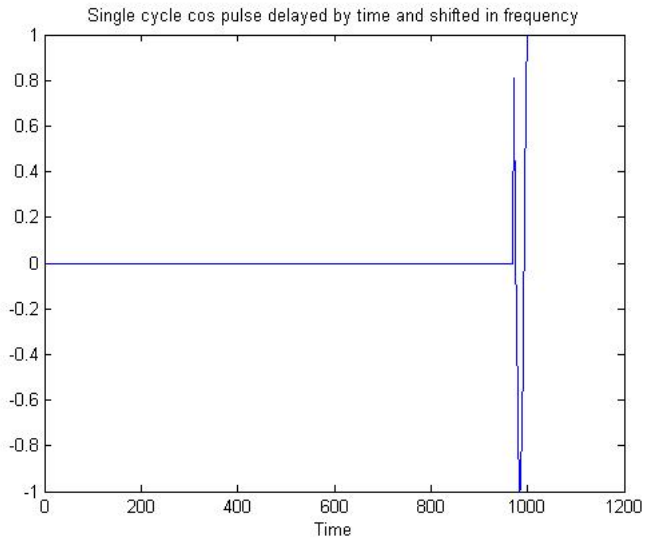

Figure 2.6: Single cycle cos pulse shifted both in time and frequency

2.8.2 Simulated Environment

A number of simulated data sets were provided through a simulator developed by Marc Brooker. This data represents the reflections which would have been received from both a stationary and a moving target.

2.8.3 Test Results

Basic waveforms

The contour plot of the ambiguity function for the time shifted pulse can be seen in Figure 2.7. This ambiguity function takes on the same shape as that of the cos pulse by itself, with the only difference being a shift along the t/T axis. This indicates that the pulse has been shifted in time only.

The contour plot for the frequency shifted cos pulse can be seen in Figure 2.8. In this case, the function has experienced a fair amount of distortion. However, the peaks can still clearly be seen, having shifted along the vT axis. This indicates a shift in the Doppler frequency.

The contour plot for the pulse shifted both in time and frequency can be seen in Figure 2.9. In this case, the ambiguity function displays a combination of the effects seen in the previous two examples. The peaks can still clearly be seen, having been shifted along both axes.

Simulated Environment

The complexity of the simulated data proved to be difficult for the high-level implementation to handle. The amount of data points present in these simulations take up too much

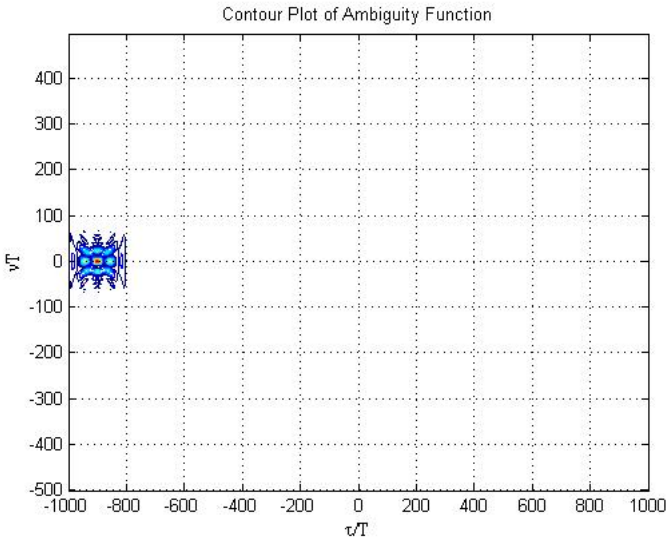

Figure 2.7: Contour Plot of ambiguity function for time shifted cos pulse

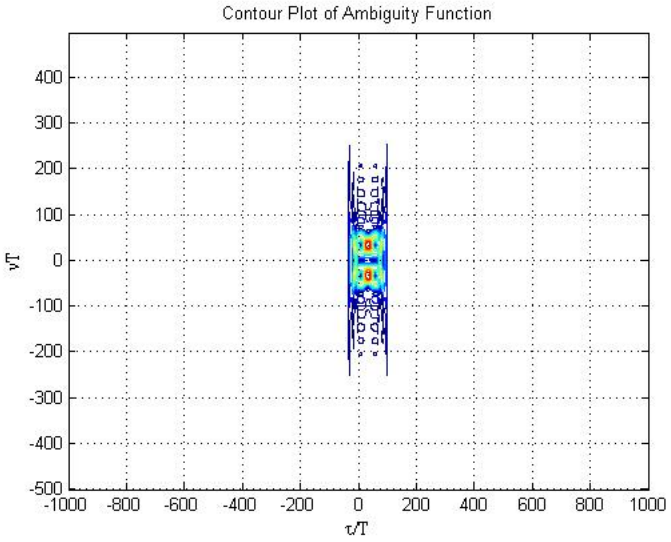

Figure 2.8: Contour plot of ambiguity function for frequency shifted cos pulse

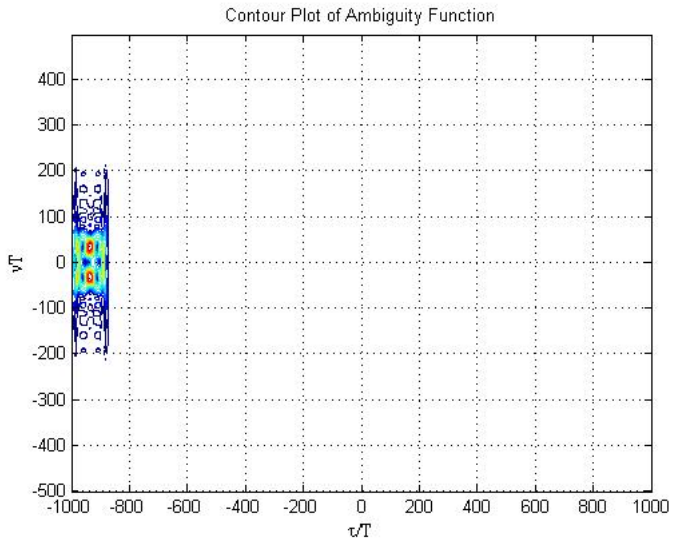

Figure 2.9: Contour plot of ambiguity function for cos pulse shifted in both time and frequency

processing power to be processed in Matlab within an acceptable period of time. A one second sample could take anywhere from half an hour onwards to process.

Clearly, the high-level implementation can be seen as a tool for verifying the accuracy of the algorithm. However, this implementation cannot be used as an actual processing tool for any real-world applications.

The algorithm needs to be transferred to a much lower level in order to be useful in any practical manner.

Chapter 3

PCL Implementation on a GPU

3.1 Introduction

This chapter will deal with the implementation of the algorithm which was developed in the previous chapter. This algorithm will be implemented in C/C++ to run on a graphics card's GPU, using the Rapidmind toolkit.

The Matlab code developed in the previous chapter will need to be translated into C/C++ code. In addition, the mathematical operations need to be examined to see which would run optimally on the CPU and which would run optimally on the graphics card's GPU.

Once the implementation is in place, the system will need to be tested in order to examine its performance characteristics. The accuracy will be tested through the use of sample data obtained through a simulator developed by Marc Brooker.

The speed performance of the system will need to be compared to the speed of a similar system running purely on a PC's CPU. This will be done by running the same sets of test data on both implementations in order to test the speed performance.

The purpose of such an investigation is to determine the extent to which a GPU could speed up the performance of a PCL system and what kind of performance enhancement can be expected.

3.2 GPU Characteristics

The graphics card being used for this investigation is a nVidia GeForce 7600. This graphics card is equipped with 256MB of memory with a 128-bit memory interface and memory bandwidth of 22.4 GB/sec.[3]

This model graphics card is the only one which is being examined for this investigation. As such, the characteristics of other GPUs will not be considered for this investigation, even though such comparisions could provide valuable insight into performance comparisions.

3.3 The Rapidmind Toolkit

The Rapidmind toolkit is a development platform which allows for code written in C/C++ to run on either a graphics card's GPU or a Cell BE processor. The toolkit acts as an interpretive layer between the C/C++ code developed and the GPU or Cell BE processor. [4]

3.3.1 Advantages of using Rapidmind

Some of the advantages of using the Rapidmind toolkit to program the GPU are as follows: [4]

- Standard C/C++ can be used, simplifying development for experience programmers
- Rapidmind is continually updated through pluggable modules to support any new technologies
- Issues such as synchronisation is handled automatically by the toolkit, so programmers need not be concerned with these issues
- Allows for data to be run in parallel, increasing efficiency of programs by adapting to multi-core processors
- Options to expose hardware specific features, increasing program efficiency on specialised hardware
- Can easily be integrated into existing C/C++ code by enabling only certain sections of code to make use of Rapidmind's features

These advantages are the main reasons why Rapidmind was chosen as the toolkit for development of the PCL algorithm.

3.4 Code Development

The algorithm follows the same outline as the one developed in the previous chapter. The biggest task in converting this algorithm lies in converting the steps into C++ code and integrating this with the Rapidmind toolkit.

Several sets of example code are available from the Rapidmind website, including a set of FFT methods. These would serve as a starting point for the development of the algorithm code.

The algorithm needs to be broken down into smaller steps than they were in the Matlab implementation. The steps which were outlined are as follows:

1. Window off the segments of the reference signal and the reflected signal as required
2. Perform a cross correlation of the two signals in the time domain
3. Perform an FFT on this cross correlated signal to transfer it to the frequency domain
4. Calculate the radar ambiguity function from this frequency domain signal
5. Scan through the radar ambiguity function at different Doppler frequencies to find the peak and hence the target's range and Doppler shift

While the code for implementing the FFTs and cross correlation is available, a working implementation of the full algorithm using this approach has not yet been finalised.

3.5 Testing of the Code

The code developed did not reach the level of maturity where testing was possible.

However, had the code development reached this stage, the same basic test waveforms used to test the high-level implementation would also have been used to verify the accuracy of the algorithm.

In addition, the simulated data which had proven to be too complex for the high-level implementation would have given the best indication of the accuracy of the GPU implementation, as well as an indication of its performance.

It is expected, however, that such an implementation would show a great improvement, due to the nature of the calculations involved in the algorithm.

The algorithm requires a relatively large amount of FFTs to be calculated. While traditional programming methods on a CPU would require that they be calculated serially, the GPU's architechture allows for a number of these calculations to be run in parallel.

However, the GPU's performance is still limited by a number of factors which will be examined in the next chapter.

Chapter 4

GPU Performance Benchmarks

4.1 Introduction

This chapter will be examining the performance characteristics of the graphics card being used.

The characteristics which will be examined include the graphics card's power performance, memory transfer time between the computer's memory and the graphics card's memory, and the processing power of the GPU itself.

The memory transfer benchmarks will be performed through the use of the Rapidmind toolkit. The other performance characteristics will be examined through research.

Further GPU performance considerations will be examined, along with tests performed to examine these considerations.

4.2 Power Characteristics

A number of test results were examined, which provide a look at the power consumption of the graphics card, as well as comparing these results to a number of other graphics cards. These tests measured the total system power of the computers being used to test the graphics cards, and not just the cards themselves. [5]

Figure 4.1shows the power consumption of a number of graphics cards which are idle. The GeForce 7600 GT, which is the card being used as the basis for this investigation, draws, on average, 127W of power when running idle.

The power consumption of the same graphics cards while active can be seen in figure 4.2. The GeForce 7600's power consumption increases to 213W when in use. This level of power consumption is very favourable when compared to the average power consumption of a radar transmitter. However, this cannot be viewed in isolation, as it only takes into account the power required for the signal processing aspect of the system and not the receiver equipment.

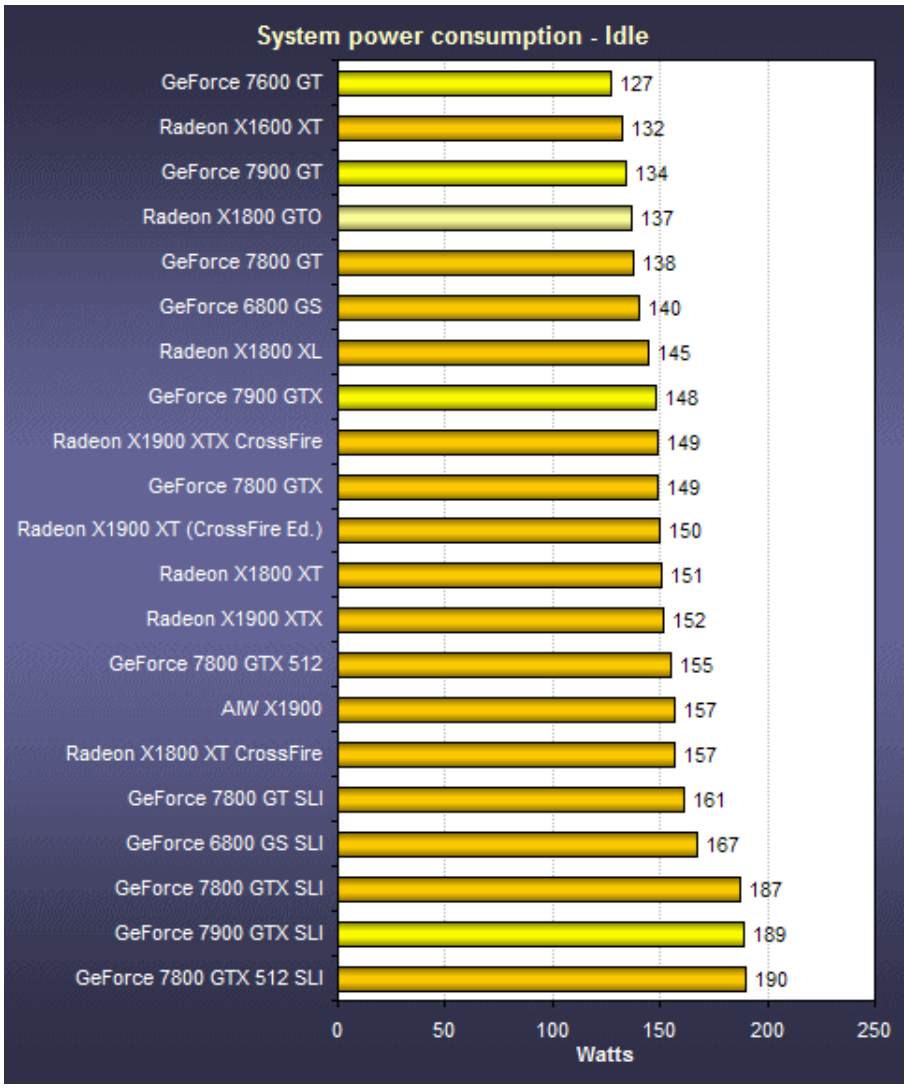

Figure 4.1: Power Consumption of Graphics Cards in Idle Mode[5]

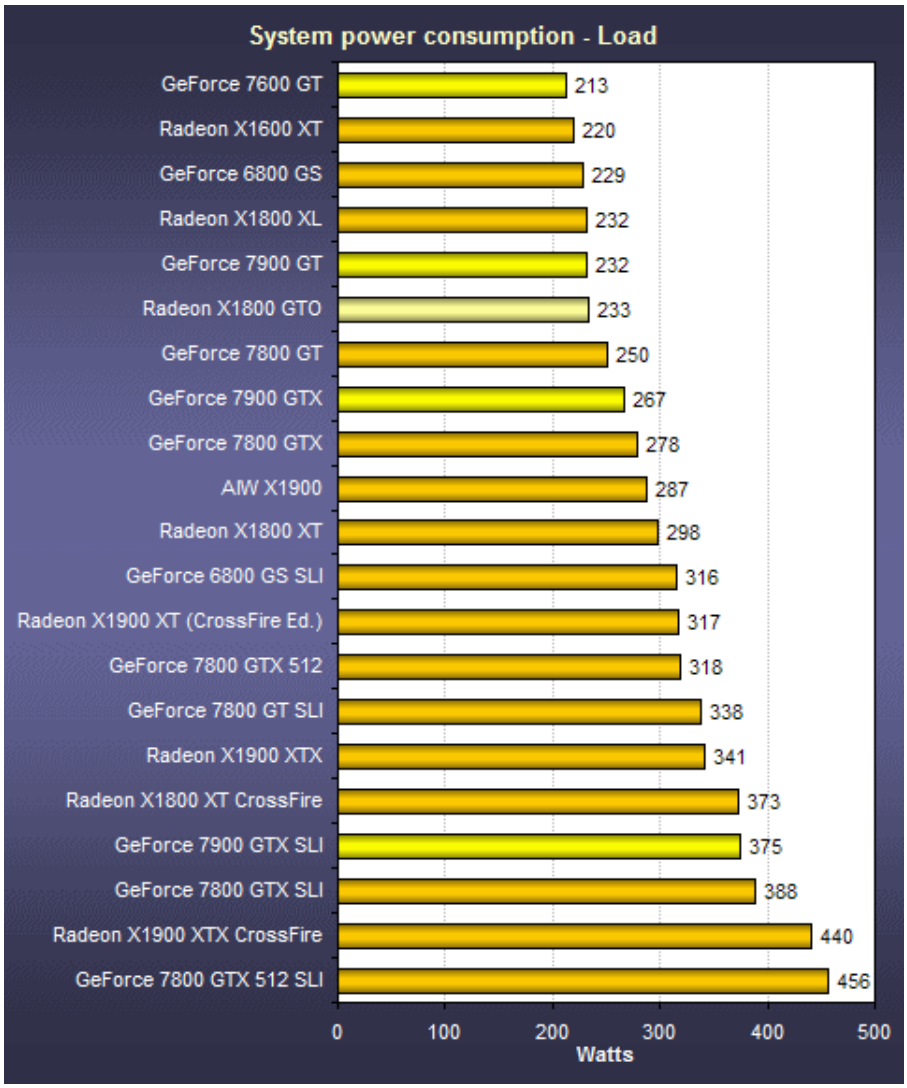

Figure 4.2: Power Consumption of Graphics Card under load [5]

With the relatively low power consumption of a GPU, it offers a significant advantage over large-scale receivers normally used for radar detection.

If it were possible to implement a PCL algorithm to run purely on a graphics card, without the requirement of a PC, this power consumption could be reduced even further.

4.3 Memory Transfer Test Descriptions

The memory transfer tests were performed by using the Rapidmind toolkit to transfer a series of arrays of floating point numbers to the GPU and to perform a basic addition on these.

The tests were performed with a number of different elements for the arrays, starting with 100 elements each. The number of elements was increased up to 4 000 000 elements per array.

Rapidmind's Timer was used to measure the amount of time taken to load these arrays, as well as process the basic addition.

These tests were also expanded to accommodate a larger number of arrays.

4.4 Memory Transfer Test Results

The transfer times for loading and adding two and three arrays into memory can be seen in Figure 4.3. The memory loading times tend to remain rather close to each other when dealing with arrays of less than 1 000 000 elements. However, as the number of elements increase beyond this, the loading times before greater at an almost exponential rate.

Increasing the number of arrays to four highlighted the graphics card's memory limitations. An attempt to process four arrays of 3 000 000 elements each lead to a memory overflow.

One aspect of the Rapidmind toolkit which was highlighted during the benchmarking process is the fact that Rapidmind makes use of a dynamic run-time compiler. Due to this aspect, the first run of any program will normally run slightly slower, as it takes a short amount of time to compile first. Once this compilation has been done once, it doesn't need to recompiled. So subsequent runs of the program will be faster than the initial one. [6]

4.5 GPU Performance Evaluations

A number of tests performed by Michael McCool, Kevin Wadleigh, Brent Henderson and Hsin-Ying Lin have yielded a number of results which provide insight into the performance enhancements which can be gained from using a GPU as a co-processor in mathematically intense computations. [7]

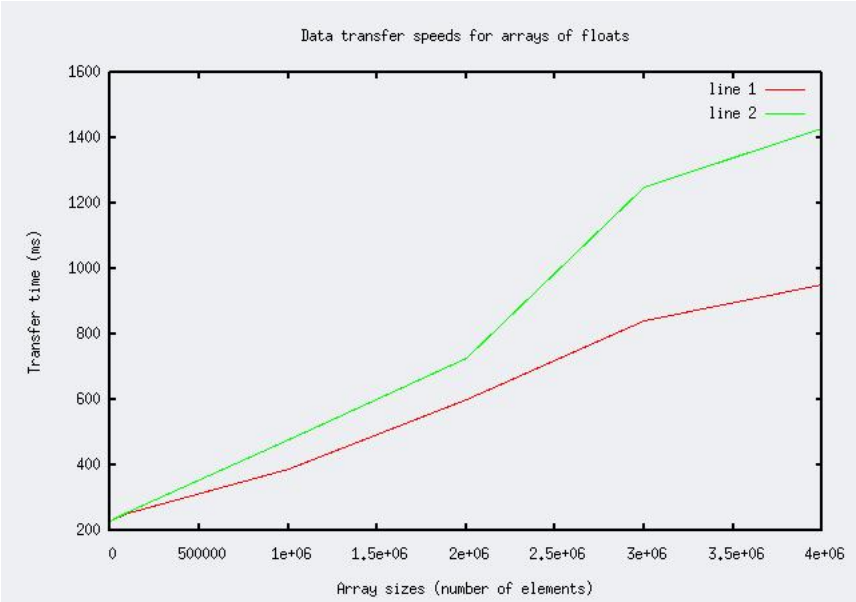

Figure 4.3: Data Transfer speeds for arrays of floats

A general overview of the performance comparison between a GPU and a CPU, running a series of different algorithms, can be seen in Figure 4.4. While this shows that the GPU outperforms the CPU in all tested tasks, the Black-Scholes European option pricing benchmark shows the most remarkable improvement. This vast improvement is due to the nature of this algorithm, for which a large number of calculations can be run in parallel. This parallel aspect highlights one of the most important advantages of a GPU over a CPU for performing certain mathematical operations.

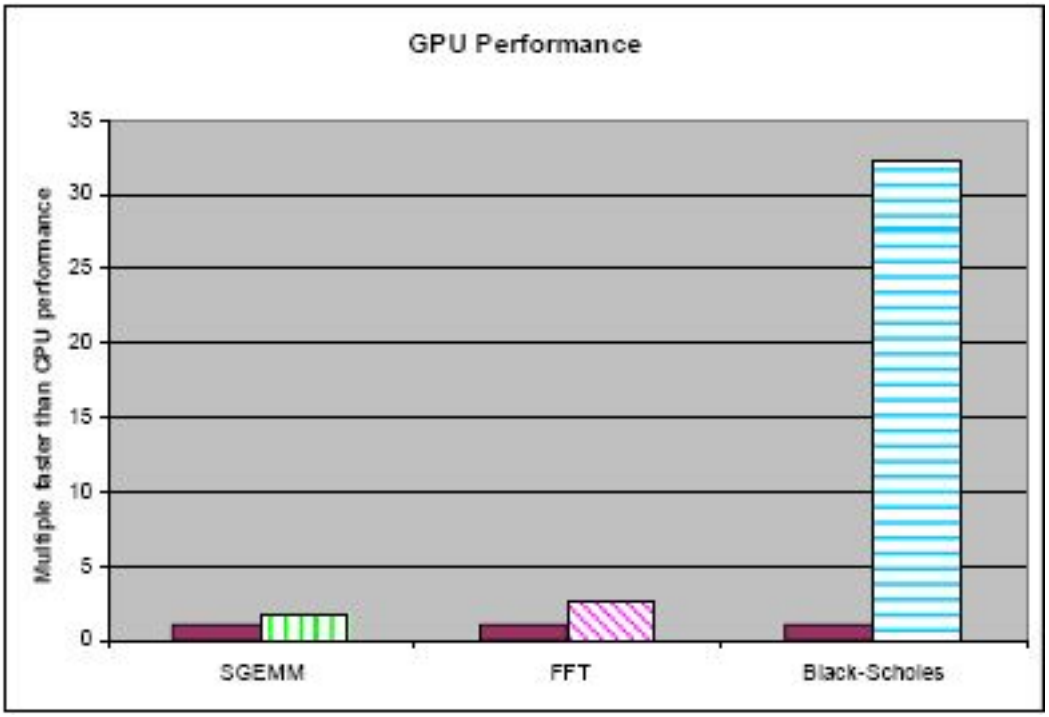

Figure 4.4: GPU Performance Characteristics for Different Processing Tasks[7]

The results of the tests performed on the FFT algorithm will now be examined.

These tests were performed on two different GPUs (an ATI x1900 XT and an nVidia 7900 GTX), using the Rapidmind toolkit to to implement the C++ code on the GPUs. These were performed to similar algorithms running on high-end workstations, as opposed to typical desktop machines.

Figure 4.5 shows the results of a series of 2D FFT tests. The tests were conducted with the algorithm being run on the ATI and nVidia GPUs and compared to similar FFT algorithms being run on two different CPUs, namely an Intel MKL and AMD ACML.

While the CPU implementations show better performance for smaller data sets, the GPUs show a large improvement in performance as the data sets increase in size. The nVidia 7900, in particular, shows a large improvement over the ATI x1900 XT.

The reason behind the performance advantage of CPU implementations for smaller data sets is cited as being a result of the initial overhead required to set up the processing on the graphics card. In addition, restrictions in transfer speeds between the GPU and memory can cause a bottleneck.

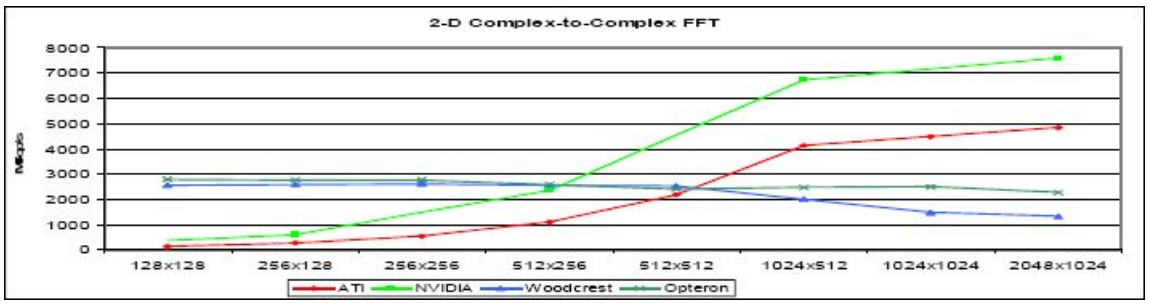

Figure 4.5: 2D FFT Performance Comparison Between ATI and NVidia GPUs and FFT Algorithms on a standard CPU [7]

4.6 Conclusions

The relatively low power consumption of a GPU offers a significant advantage over large-scale receivers normally used for radar detection. However, the power consumption of the entire system still needs to be taken into account, as this power only applies to the signal processing component of the system.

While the memory loading times yielded an insight into the expected time for memory loading, these benchmarks still need to be compared to some standard before yielding more insightful results. An example would be comparing this performance to that of another model graphics card.

A range of different benchmarking tests would also yield more insight, such as transfers of different data types.

While the parallel processing capabilities of a GPU have been mentioned as one of the biggest advantages when compared to a CPU, the memory aspect still restricts the performance.

Chapter 5

Results and Conclusions

This chapter will be presenting a summary of the results obtained from all the investigations. The results from the performance and memory transfer benchmarks will be presented.

Possibilities for further research will be suggested, as well as examining possibilities for improvement regarding the research already conducted.

5.1 High-level Implementation

A high-level implementation of a PCL algorithm in a language such as Matlab provides valuable insight into the accuracy of such an algorithm under controlled test circumstances. However, such an implementation is far too computationally intensive to be used as a real-world implementation of the system.

The Matlab code was found to provide an accurate presentation of the radar ambiguity function and, through that, an accurate measure of the range and Doppler shifts of the test target.

Further research could be conducted to extend the algorithm to the tracking of multiple targets, as well as the problem of joint target tracking and classification.

5.2 GPU Implementation

Implementing the high-level algorithm to make use of a GPU as a co-processor would provide a significant speedup over a pure CPU-based implementation. However, there are still a number of tasks which need to be implemented on the CPU.

While the steps of the algorithm have been outlined, attempts to fully implement the algorithm still remain inconclusive. Further work needs to be done in order to finalise this implementation.

Further possibilities for research include testing implementations to see if it would be

possible to find one which can be implemented to run purely on a GPU, without the need to use a PC's CPU for certain tasks.

5.3 GPU Performance Benchmarks

Examining the power consumption of the graphics card has revealed that this aspect compares favourably to that of large-scale radar receivers. In addition, if it were possible to implement the PCL algorithm entirely on a graphics card, without the requirement of a PC, this power consumption could be reduced even further.

The memory transfer benchmarks need to be compared to some standard before the results can reveal anything conclusive. However, the tests did reveal the limitations of the graphics card's memory capacity. A series of more complex data types, as well as more complex operations on the data would yield even further insight into the memory capabilities.

Research into the performance benchmarks of a GPU has revealed that the parallel processing capabilities of a GPU makes it suited to certain mathematically intense tasks, such as FFTs. However, the biggest limiting factor to the performance of a GPU as a general-purpose processor lies in its limitations concerning memory bandwidth and memory transfer speeds.

Possibilities for further research include examining the performance of similar processors, such as the Cell BE processors, as well as keeping track of future development with regard to memory technologies.

5.4 Possibilities for Further Research

There are several ways in which this research can be taken further, in order to explore further implementations of the PCL signal processing aspects. Some of these include:

- Expanding the algorithm to detect multiple targets
- Expanding the algorithm to implement joint target tracking and classification [8]
- Investigating the possibilities of running the signal processing purely on a GPU
- Investigate future trends of GPU development, particularly regarding memory transfer and access advancements.
- Compare the performance of a GPU to that of a Cell BE processor.

Appendix A

Matlab Source Code

cmplxAmbiguity.m[9]

Matlab code representing the starting point for calculating and plotting the radar ambiguity function for two signals

```
1  function Z = cmplxAmbiguity(normalize, fftPoint, sampleRate
       , Ut1, Ut2)
2  % THIS FUNCTION IS THE ENTRY POINT
3  % This function plots the monostatic ambiguity function of
       one or two signals (crossambiguiity function)
4  %
5  %
6  % Imput parameters:
7  % normalize = it normalises the function if it is 'true'
8  %
9  % fftPoint = it is the n-point DFT (for more see the
       function "fft" in the MATLAB functions reference )
10 %
11 %sampleRate =  this is the chip length in a single pulse (
       the sampling rate).
12 %  if the signal is [0 1 0] four samples per pulse
13 %              ____
14 %             |    |
15 %         ____     ____
16 %       then  sampleRate = 4.
17 %
18 %  if the signal is [0 1 0] - one sample per pulse
19 %              _
20 %             | |
21 %           _   _
22 %       then  sampleRate = 1.
```

```
23 %
24 % Ut1 = the first input signal
25 % Ut2 = the second input signal - if the signals are the
       same... only the first is acceptable
26 %
27 %
   ------------------------------------------------------------------------

28 % ********  SAMPLE m-SYNTAX -  a barker code length 13
       ******************
29 %
   ------------------------------------------------------------------------

30 %
31 %  barker=[-1 -1 -1 -1 -1 1 1 -1 -1 1 -1 1 -1];
32 %  Z = cmplxambiguity('true', 256, 1, barker, barker);
33 %     . 'true' -  normalize it
34 %     .  256    - 256 point of FFT
35 %     .  1      - the sample rate is 1 i.e. one bit one
       symbol
36 %     .  barker - the vector of the input signlal (
37 %                          if I want to find the
       ambiguiity function of one signal only the second imput
       signal could be skipped)
38 %                          if I want to find the cross
       ambiguiity function of two signals only the second imput
        signal should is obligated and have ot have same length
        as the previous)
39 %
40 %
41 %
42 %
   ------------------------------------------------------------------------

43 % ********  VERIFICATION  ******************
44 %
   ------------------------------------------------------------------------

45 % CHECKED with:
46 % barker Pulselength = 13, sampleRate =1;
47 % rectangular pulse Pulselength = 1, sampleRate =14;
```

```
48 % rectangular pulse Pulselength = 1, sampleRate =8;
49 %
50 %
   ------------------------------------------------------------------------

51 % ********  REFERENCE FOR CHECKING  *****************
52 % COMPARED with "Radar Signals" Levanon N., Eli Mozeson,
       Wiley 2004
53 %
   ------------------------------------------------------------------------

54 %
55 %
56 % author: Vladimir A. Kyovtorov, e-mail: vladimir_ak@yahoo.
       com, copyright 2006
57
58 if nargin < 4
59     error('wrong number of arguments !!!');
60 end
61
62 if ~(strcmpi(normalize,'true')|strcmpi(normalize,'false'))
63     normalize = 'false';
64 end
65
66 try
67     Ut2;
68 catch
69     Ut2 = Ut1;
70 end
71
72 Z = ambiguityILL(Ut1, Ut2, normalize, fftPoint);
73
74 % ********  PLOT the ambiguity function  *****************
75
76 % ------------------------ axes estimating
       ------------------------------------
77 ambigSize = size(Z);
78 pulseLength = length(Ut1);
79 [x,y] = coordAxes(fftPoint, ambigSize(2), sampleRate,
       pulseLength);
80
```

```
81 % ------------------------ ploting
       ------------------------------------
82 figure;
83 mesh(x, y, Z);
84 title('Complex ambiguity function');
85 axes_handle = xlabel('t/T');
86 set(axes_handle,'FontName','Symbol');
87 axes_handle = ylabel('nT');
88 set(axes_handle,'FontName','Symbol');
89 axes_handle = zlabel('|c(t,n)|');
90 set(axes_handle,'FontName','Symbol');
91
92 figure;
93 contour(x, y, Z);
94 title('Contour Plot of Ambiguity Function');
95 axes_handle = xlabel('t/T');
96 set(axes_handle,'FontName','Symbol');
97 axes_handle = ylabel('nT');
98 set(axes_handle,'FontName','Symbol');
```

ambiguityILL.m[9]

Function for calculating the ambiguity function

```
1  function Z = ambiguityILL(Ut1, Ut2, normalize, fftPoint)
2  % IMPORTANT: the logical 0 must be presented with -1
3
4  A=xcorril(Ut1, Ut2);
5  B=fft(A,fftPoint);
6  Z=abs(B);
7  clear A B;
8  p = size(Z);
9  lengthX = p(1);
10 halfLength=floor(lengthX(1)/2);
11 A1=Z(1:halfLength,:);
12 A2=Z(halfLength+1:lengthX,:);
13 X=cat(1,A2,A1);
14 Z=X;
15 clear A1 A2 X;
16
17 k = strcmpi(normalize,'true');
18 if k == 1
19     m = max (Z);
```

```
20      mm = max (m);
21      Z=Z./mm;
22  end
```

coordAxes.m[9]

Matlab function for scaling the axes of the graph

```
1  function [x,y] = coordAxes(freq, time, sampleRate,
       pulseLength)
2  % time = imput sequence Length (x axis)
3  % freq = the frequency axis length (y axis)
4  % sampleRate =  this is the chip length in a single pulse.
       if the pulse is [1 1 1 1]
5  %              ____
6  %             |    |
7  %         ____     ____
8  %       then  sampleRate = 4.
9  %
10 %
11 % tb = pulse jength
12 % x axis  label = [tao/tb]
13 %
   ____________________________________________________________________

14 % ********  VERIFICATION  *****************
15 % CHECKED with:
16 % barker Pulselength = 13, sampleRate =1;
17 % rectangular pulse Pulselength = 1, sampleRate =14;
18 % rectangular pulse Pulselength = 1, sampleRate =8;
19 %
   ____________________________________________________________________

20 % ********  REFERENCE FOR CHECKING  *****************
21 % COMPARED with "Radar Signals" Levanon N., Eli Mozeson,
       Wiley 2004
22 %
   ____________________________________________________________________

23
24  half = floor((time)/2);
25  x = -half:half;
26  x = x/sampleRate;
```

```
27
28 half = (freq)/2;
29 y1=1:half-1;
30 y2 = (-half):0;
31 y = cat(2,y2,y1);
32 y = (y*sampleRate*pulseLength)/(freq); %normalize according
       to the pulse length
```

xcorril.m[9]

Matlab code for performing the cross correlation of two functions

```
1  function c = xcorril(a,b)
2
3  %if nargin == 1
4  %       b = a;
5  %end
6
7  [ma,na] = size(a);
8  [mb,nb] = size(b);
9
10 b = conj(b(mb:-1:1,:));
11 apad = [a ; zeros(mb-1,na)];
12
13 c = zeros(na+nb-1,na+nb-1);
14
15 for k=1:(na+nb-1)
16         count = k                  *(k<min(na,nb)) ...
17                 +min(na,nb)        *(k>=min(na,nb))*(k<=max(na
                      ,nb)) ...
18                 +(na+nb-k)         *(k>max(na,nb));
19
20         starta = 1                 *(k<=nb) ...
21                 +(k-nb+1)          *(k>nb);
22
23         startb = (nb-k+1)          *(k<=nb) ...
24                 +1                 *(k>nb);
25
26         for i=0:(count-1)
27                 c(i+1,k) = filter( b(:,startb+i), 2, apad
                      (:,starta+i) );
28         end
29 end
```

Bibliography

[1] B. R. Mahafza, *Radar Systems Analysis and Design Using MATLAB, Second Edition*. Chapman & Hall/CRC, 2005.

[2] A. D. Lanterman, "Tracking and recognition of airborne targets via commercial television and FM radio signals."

[3] "http://www.nvidia.com/page/geforce_7600.html."

[4] "http://www.rapidmind.com."

[5] "http://techreport.com/articles.x/9529/13."

[6] "https://developer.rapidmind.net/documentation/faq/programming-faq/."

[7] M. McCool, K. Wadleigh, B. Henderson, and H.-Y. Lin, "Performance Evaluation of GPUs Using the Rapidmind Platform," tech. rep., Rapidmind Inc, Hewlett-Packard Company, October 2006.

[8] S. M. Herman, *A Particle Filtering Approach to Joint Passive Radar Tracking and Target Classification*. PhD thesis, University of Illinois, 2002.

[9] "http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=10961&objectType=file