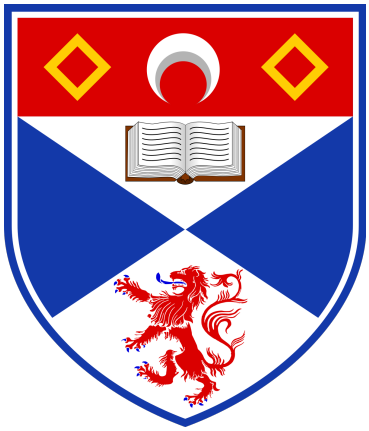

University of St Andrews

School of Physics and Astronomy

Millimetre-Wave Cloud Profiling Radar

Project Report

180014855

Module: PH4111

Word count: 2300

April 7, 2022

Abstract

Insert abstract here.

Contents

Acronyms

ADC	Analogue to digital converter
CPI	Coherent processing interval
CPR	Cloud profiling radar
CPU	Central processing unit
DAQ	Data acquisition
DDS	Direct digital synthesiser
DFT	Discrete Fourier transform
FFTW	Fastest Fourier Transform in the West
FMCW	Frequency-modulated continuous wave
IF	Intermediate frequency
Intel MKL	Intel Math Kernel Library
NetCDF	Network common data form
OpenMP	Open multi-processing
SIMD	Single instruction, multiple data
SNR	Signal-to-noise ratio

1 Introduction

*

2 Signal processing theory

2.1 FMCW-Doppler radar

FMCW radar emit consecutive chirps - signals whose frequency is linearly modulated by sawtooth waves. The reflected chirp arrives at a time Δt after emission, leading to a phase difference as illustrated in Fig. 1. The transmitted and received signals are mixed (Fig. 2) and their intermediate frequency (IF, essentially the difference frequency) is extracted. The target range relates to the time delay and hence the intermediate frequency f_{IF} by

$$r = \frac{c\Delta t}{2} = \frac{c f_{IF} T_c}{2B}, \tag{1}$$

where r is the distance from the radar and T_c is the chirp duration.[3] In practice, there are many received signals, thus one performs a discrete Fourier transform of the IF signal to obtain a range profile, known as a range-FFT or *fast time* FFT because of the short timescale (and high sampling rate) over which samples are taken.

To obtain velocity information one extracts the phase change in each range bin across multiple consecutive chirps by performing a Doppler-FFT or *slow time* FFT. The phase difference δ relates to the target velocity v by[3]

$$\delta = \frac{4\pi v T_c}{\lambda}. \tag{2}$$

The sequence of chirps is known as a coherent processing interval (CPI), the duration of which

*The code is available on GitHub.[1]

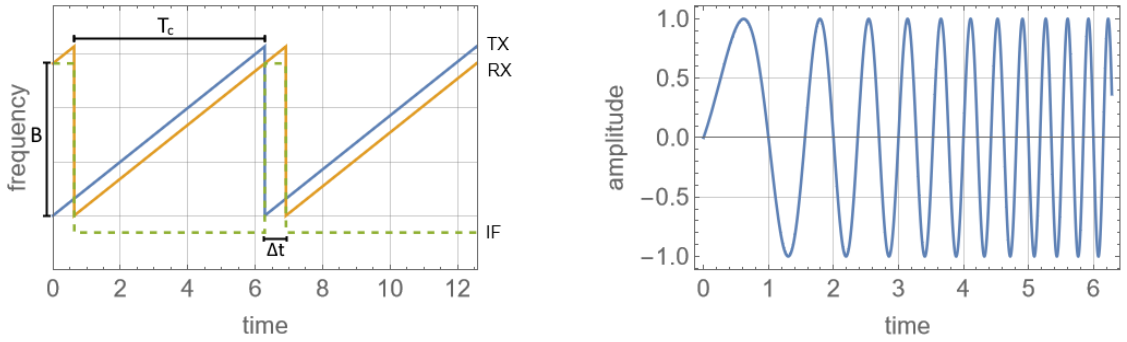

Figure 1: Left: transmitted (TX) and received (RX) sawtooth chirps and their difference frequency (IF). Right: illustration of TX chirp waveform. (Own work)

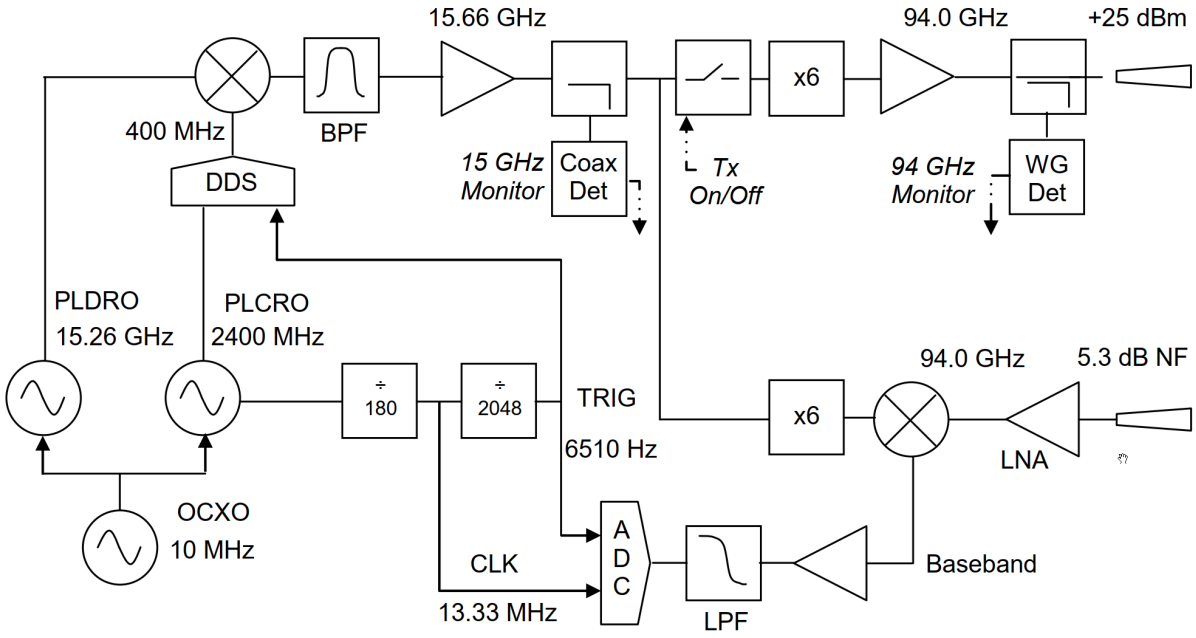

Figure 2: Simplified radar system diagram.[2]

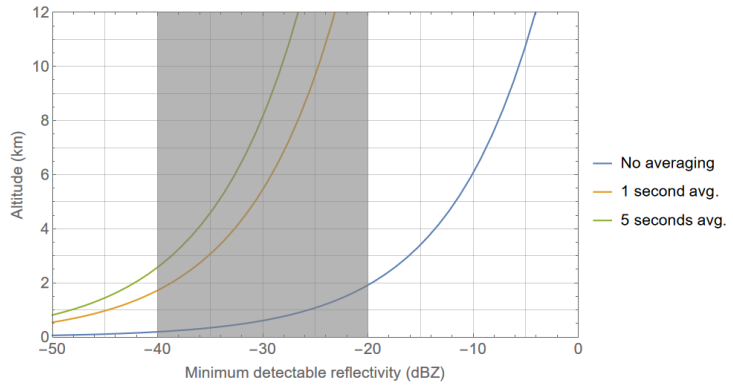

Figure 3: Radar sensitivity (Own work).

determines the velocity resolution v_{res} by

$$v_{res} = \frac{\lambda}{2T_f}, \tag{3}$$

where T_f is the CPI duration.

Each CPI can be thought of as a matrix where each row represents the fast-time samples. The fast-time FFTs are performed row-by-row to get several range profiles and, afterwards, the slow-time FFTs are performed column-by-column to obtain the velocity spectra.

2.2 Signal averaging

Explain with signal averaging we can get $\sqrt{N}$ improvement in SNR, why that's necessary to detect clouds

2.3 Doppler moments

Statistical properties of the velocity spectrum are of great interest to meteorologists. In particular, the mean, variance, skew, and kurtosis ...

3 Requirements

There are two software implementations to date, named Mark I and Mark II. Mark I

Mark II supports real-time data acquisition (see Sec. 4.1.2) and saving of raw data (ADC integers) to disk, however does not perform any processing beyond a range-Doppler plot for each CPI.

3.1 User interface and hardware control

It should also be noted that these implementations exclusively deal with the data acquisition and processing aspect of radar software. The hardware control runs in a separate process. One of the extra project goals is to add a hardware control panel containing controls for bandwidth, chirp or continuous wave mode, transmitter on/off as well as output from various sensors. The panel could be switched between engineering and operational mode, where operational mode hides low-level controls and replaces them with more intuitive controls, such as range resolution instead of bandwidth. The

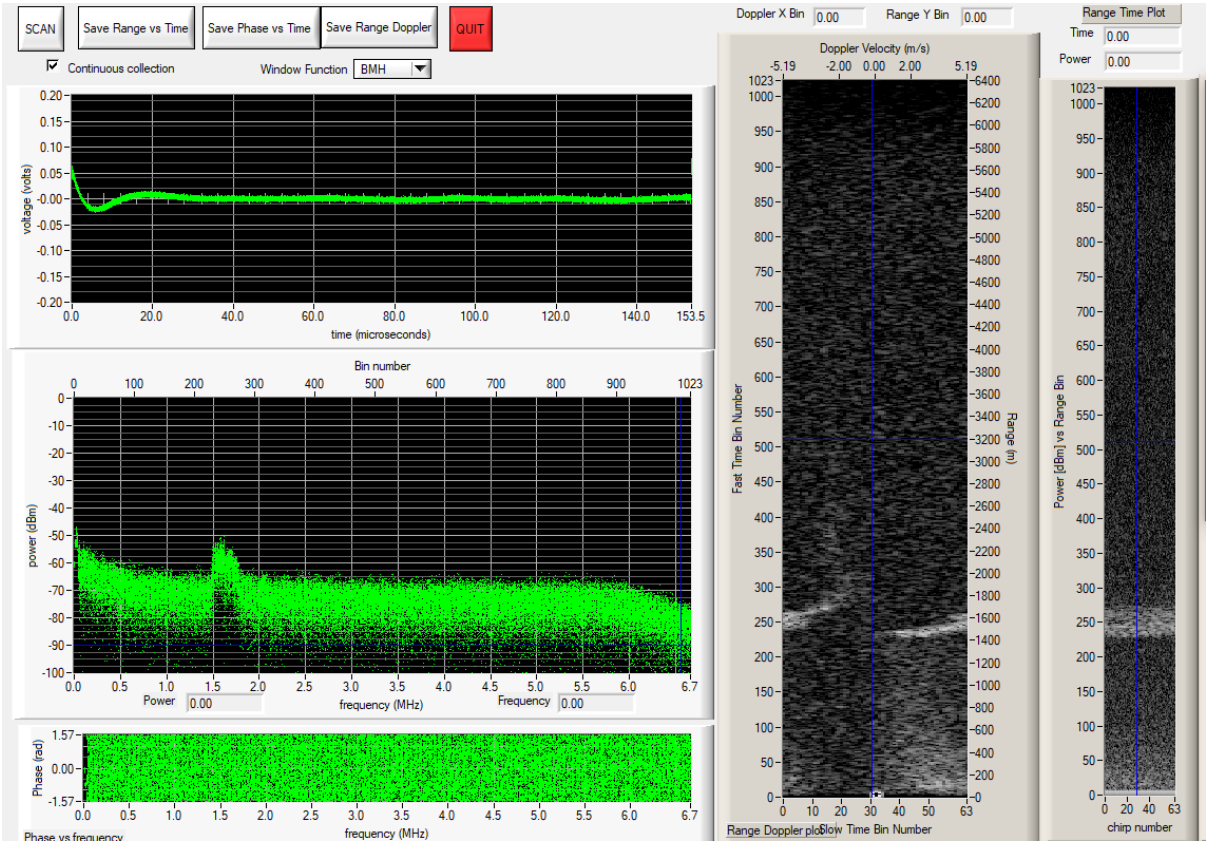

Figure 4: Screenshot of Mark I user interface, taken during precipitation.

panel would also contain controls for adjusting averaging time and velocity resolution.

4 Design and implementation

4.1 Data acquisition

4.1.1 Double buffering

To ensure real-time output, a double buffering approach is used. One buffer is used to store the incoming frame and another stores the most recently acquired frame. While the next frame is being acquired, the most recent frame is processed and displayed. The algorithm is shown in Alg. 1. Double buffering guarantees continuous real-time acquisition only if the processing time is less than the acquisition time.

Algorithm 1 Double buffering approach.

```
acquire frame asynchronously into buffer A
while keep acquiring data do
    wait for previous acquisition to finish
    acquire frame asynchronously into buffer B
    process frame in buffer A
    display processed frame
    swap A and B pointers
end while
```

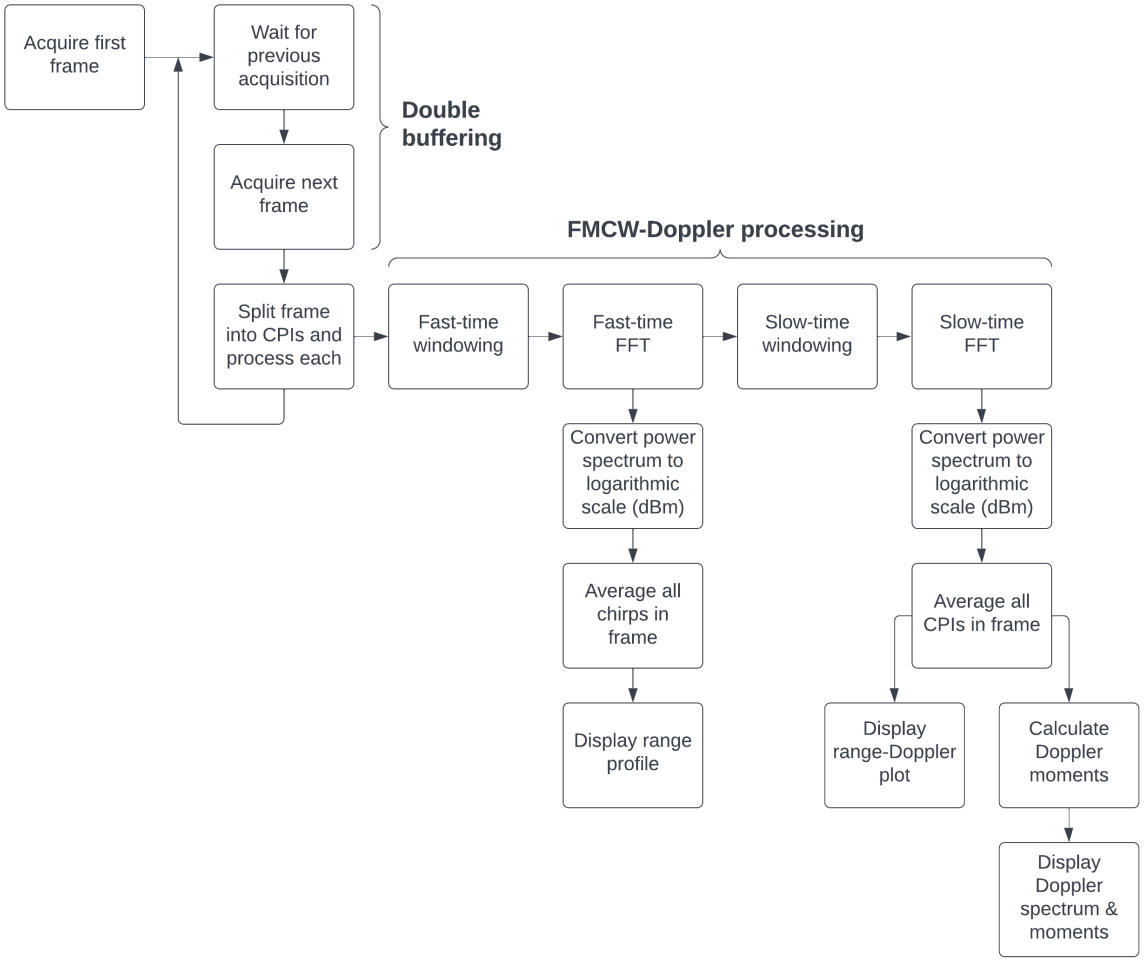

Figure 5: Simplified block diagram illustrating key signal processing steps. (Own work)

4.1.2 Alternatives to double buffering

The approach in Mark II used one acquisition thread and one processing thread. The acquisition thread repeatedly acquires and appends frames to a thread-safe queue, which the processing thread pulls from.

The main issue with this approach is the acquisition thread wastes CPU time waiting for the frame to be acquired and transferred to memory.

Algorithm 2 Acquisition (left) and processing (right) threads.

while keep acquiring data **do**	**while** keep acquiring data **do**
acquire frame into buffer	copy next frame in thread-safe queue
wait for previous acquisition to finish	process frame
copy into thread safe queue	display processed frame
end while	**end while**

The double buffering approach uses only two buffers, which avoids unnecessary copying of data between threads and the additional overhead of using a thread-safe queue.

4.2 FMCW-Doppler processing

The software uses a third-party library, FFTW v3[4], for the two discrete Fourier transform steps in the FMCW-Doppler process.

Newer versions of LabWindows CVI use FFTW behind the scenes, however the interface is not as flexible. FFTW supports multiple transforms of non-unit stride,[†] allowing one to easily perform multiple transforms along different dimensions. This is particularly useful for the column-wise slow-time transform, because it means the data does not need to be transposed before transforming. In CVI, the FFT routines are limited to a single contiguous 1D array at a time.

In FFTW, *plans* are created

The window functions available in the new software are flat-top, Hann, Blackman, and Blackman-Harris.

4.2.1 Averaging

In each frame, two averaging processes take place. Averaging of power spectra is done over each chirp. For a frame containing 256 CPIs each containing 64 chirps, this amounts to 16384 chirps averaged together.

4.3 Testing

To facilitate testing the software, a mock implementation of the DAQ was created. Raw data (ADC integers) saved by the Mark II can be loaded by the mock implementation and used as if the data was just acquired. This feature is extremely useful for reproducible debugging and testing. A few snapshots of rain data were saved and used to test the new code when the weather was too clear.

A signal generator was attached to the second input channel of the DAQ card. Checking to see if the input power matches the peak value in the fast-time power spectrum provides confirmation that any corrections or scaling performed during the fast-time processing is correct.

Setting the input frequency to be slightly off bin centre introduces spectral leakage (Fig. 6) Fig. 7 shows the effect of applying the Blackman-Harris window is as expected.

[†]Stride is the number of memory locations between each successive element in an array.

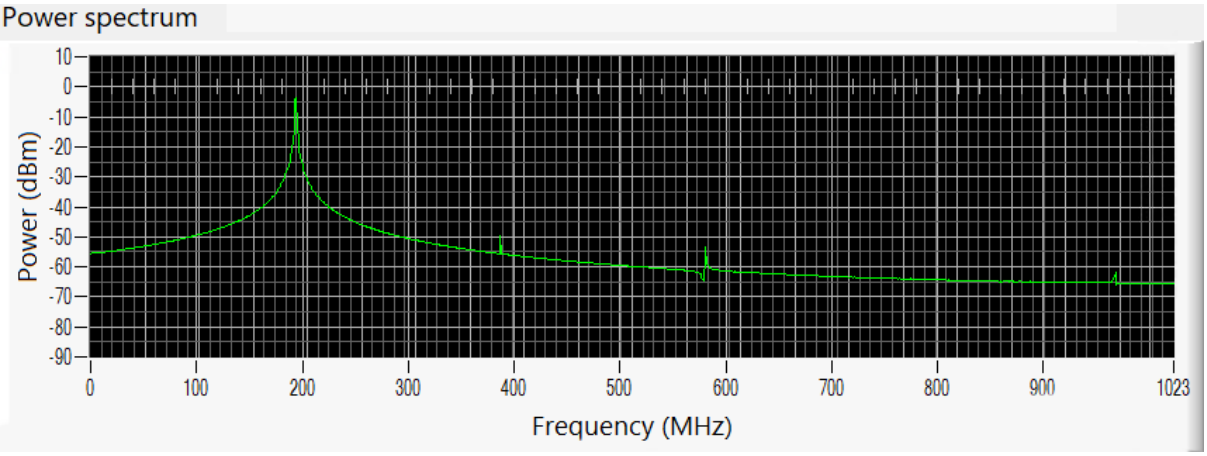

Figure 6: Fast-time power spectrum without windowing.

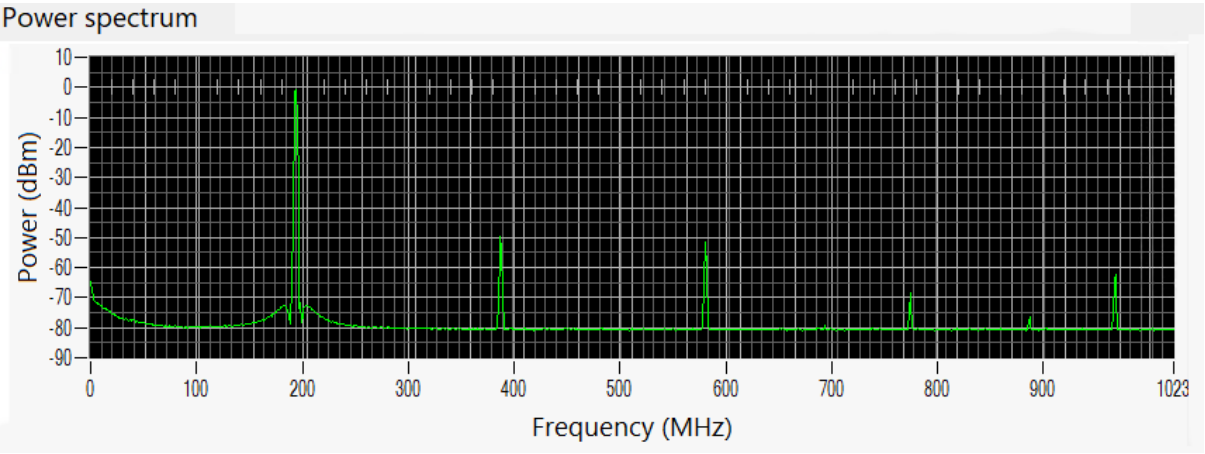

Figure 7: Fast-time power spectrum with Blackman-Harris windowing.

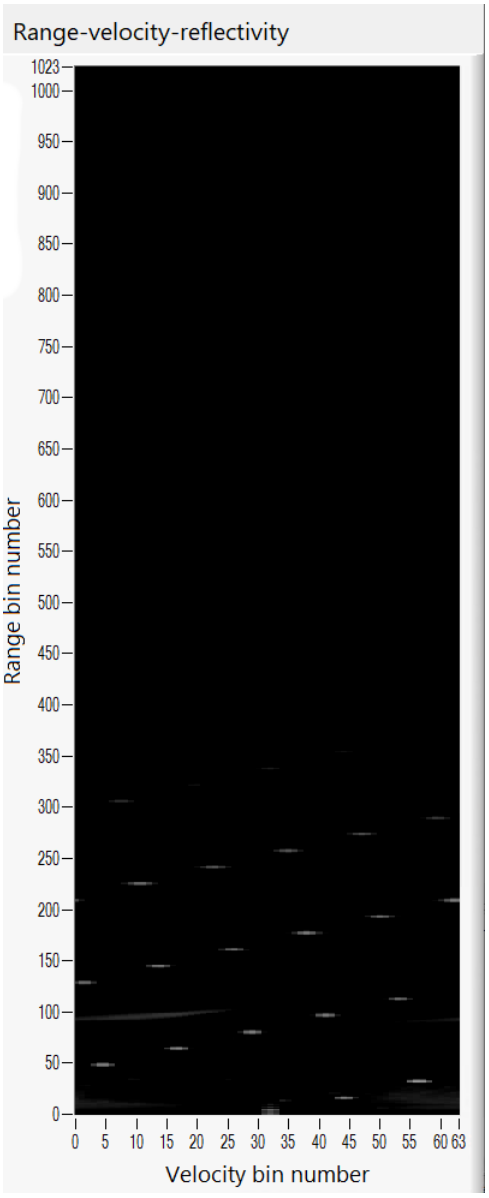

Figure 8: Screenshot of the new user interface.

4.4 Averaging

How is averaging implemented? Not much to say here really

5 Results

5.1 Performance

The results in Tbl. 2 show that the new code has continuous, real time acquisition. One CPI containing 64 chirps each with 2048 samples lasts for 19.7 ms, which is more than the processing time of (13.9$\pm$0.1) ms.

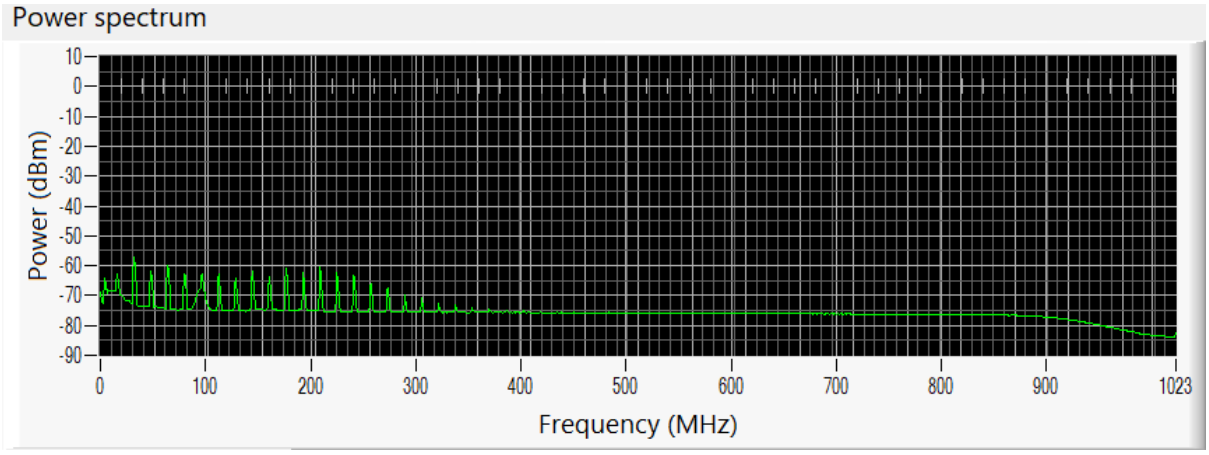

Figure 9: Screenshot of the new user interface.

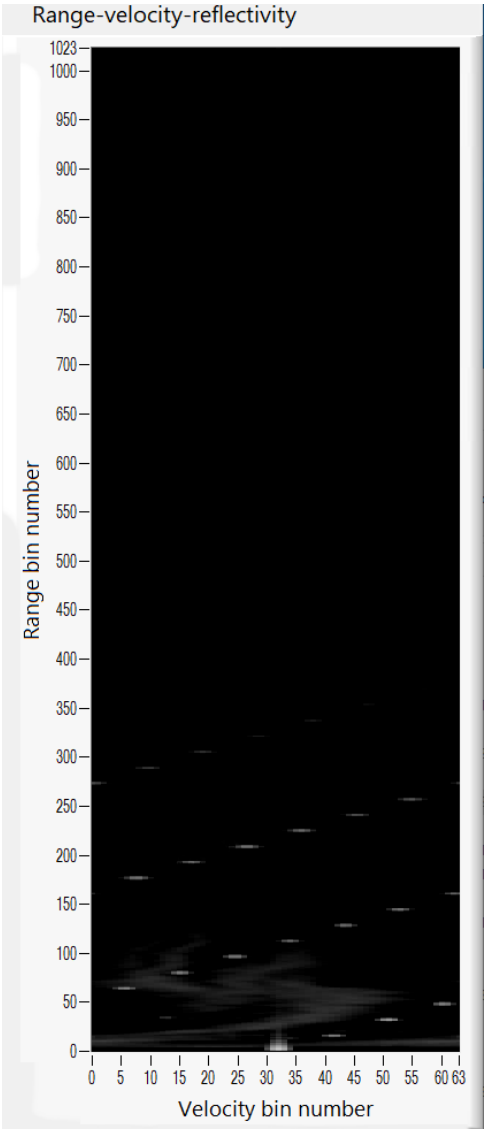

Figure 10: Screenshot of the new user interface.

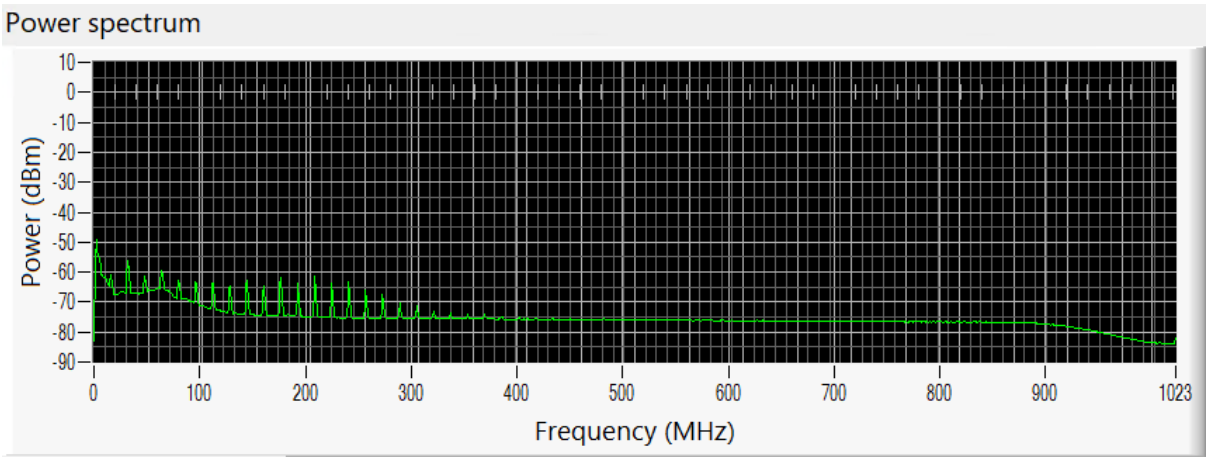

Figure 11: Screenshot of the new user interface.

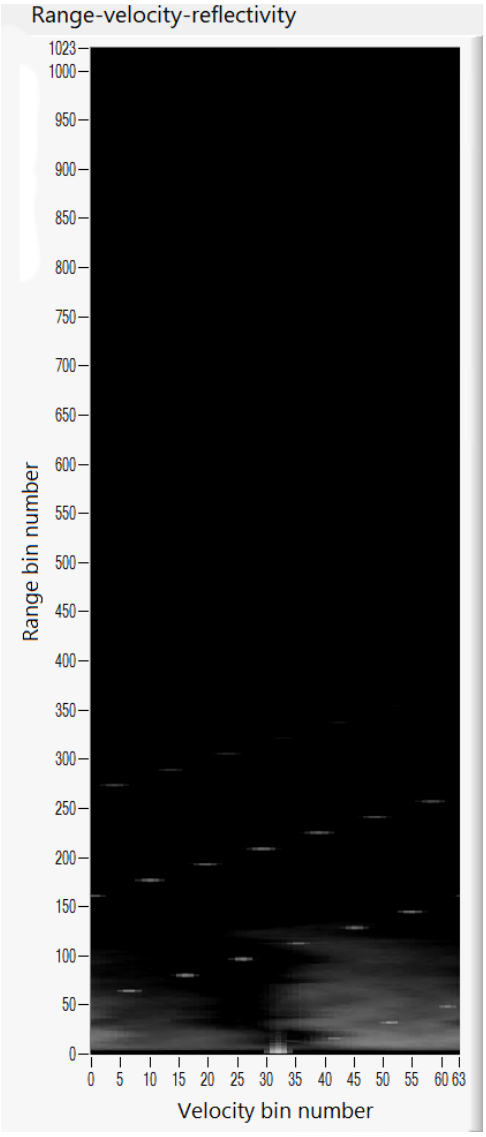

Figure 12: Screenshot of the new user interface.

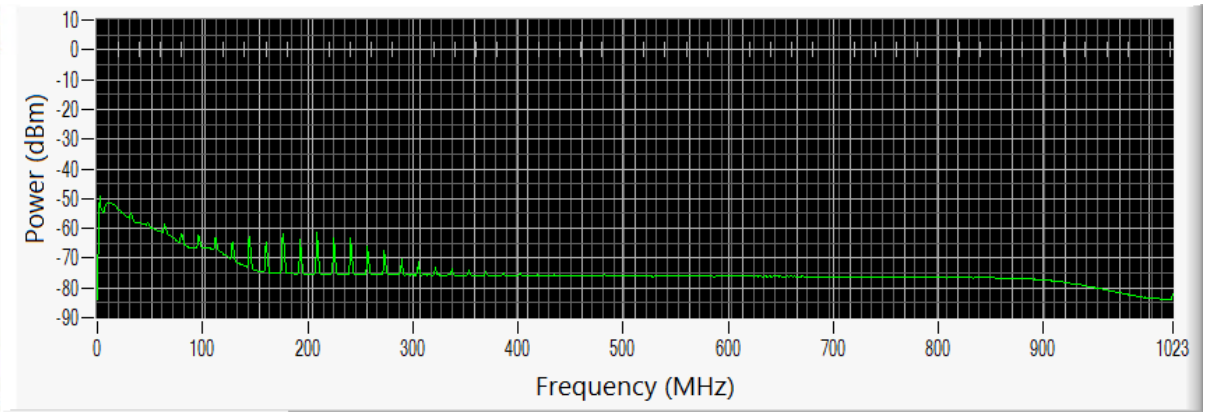

Figure 13: Screenshot of the new user interface.

	Processing time
Mark II	(67.6±0.6) ms
New	(13.9±0.1) ms

Table 2: Processing time of Mark II vs new code. Measured as the time taken to process one CPI (64 chirps, 2048 samples). For new code this also includes power spectrum averaging and Doppler moment extraction. Number of samples: 400. Both built using CVI v19.0.0. Release build.

There are several reasons for the speed improvements in the new code. Both the Mark I and Mark II frequently copy data into temporary arrays while processing, which is avoided in the new code. In addition, these temporary arrays are variable-length (VLAs) which are generally slower than arrays with a size known at compile-time or dynamically allocated arrays. The new code leverages FFTW's advanced interface to FFT the entire 2D CPI array in one go. Since FFTW knows the dimensions of the input it might be using optimisations suited for multidimensional transforms.

6 Future work

6.1 Calibration

Currently the output power is in dBm rather than dBZ. Since the conversion to dBZ is dependent on bandwidth and the hardware control was not fully implemented, it was decided to not convert to dBZ yet.

6.2 Doppler moments

Extracting the Doppler spectrum from the averaged range-Doppler data has already been implemented along with UI controls for displaying Doppler moments, however the results are wildly incorrect.

the formulas used for determining the moments are incorrect.

6.3 Effect of ambient temperature

6.4 NetCDF

6.5 Performance improvements

6.5.1 Multithreading

Many of the calculations in the FMCW-Doppler process are highly parallel and are performed on large
An easy way of parallelising loops with is to use OpenMP

6.5.2 SIMD

Single instruction, multiple data (SIMD) is another type of parallel processing where a single instruction operates simultaneously on multiple pieces of data. FFTW uses this extensively. Many of the steps in the FMCW-Doppler process are adding or multiplying rows or columns by the same constants which makes SIMD highly effective.

7 Conclusions

References

[1] [Online]. Available: https://github.com/sb362/fmcw.

[2] D. A. Robertson and R. I. Hunter, "A solid state 94 GHz FMCW Doppler radar demonstrator for cloud profiling," in *Radar Sensor Technology XXI*, vol. 10188, Proc. SPIE, 2017, pp. 345–351. DOI: 10.1117/12.2261871.

[3] C. Iovescu and S. Rao. "The fundamentals of millimeter wave radar sensors," TI Instruments, [Online]. Available: https://www.ti.com/lit/wp/spyy005a/spyy005a.pdf (visited on Jan. 20, 2022).

[4] "FFTW home page," [Online]. Available: https://www.fftw.org/ (visited on Apr. 3, 2022).