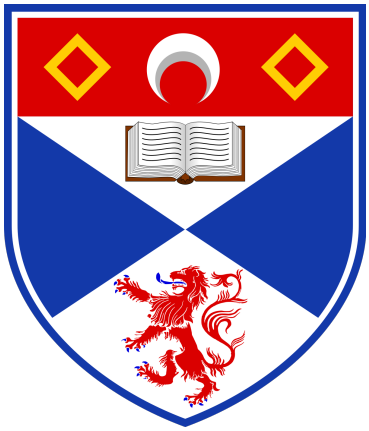

University of St Andrews

School of Physics and Astronomy

Millimetre-Wave Cloud Profiling Radar

Project Report

180014855

Module: PH4111

April 8, 2022

Science is often highly collaborative. Please clarify here which parts of the work reported have been done by you and which parts include the work of others.

For example, this might include provision of simulation or analysis codes which you utilised or modified, provision of experimental data to analyse, collaborations wit others in performing new measurements, if you built up things from scratch in your project or started from pre-existing setups, and others. Providing this information helps clarify the scope and focus of your project work.

- Mark I code was supplied by Dr Duncan Robertson.
- Mark II code was developed by Dr Samiur Rahman.
- New code is a complete rewrite from scratch.

Abstract

This report details recent upgrades to the cloud profiling radar software, including continuous real-time data acquisition, signal averaging, and extraction of statistical properties from the velocity spectrum. Performance is compared to two previous implementations.

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

The current cloud profiling radar code is of limited usefulness to meteorologists due to low sensitivity of clouds, lack of continuous real-time display, This report details the implementation of these new features and the construction of new, modular design to the cloud profiling radar software.*

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.[2] 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.[4] The phase difference δ relates to the target velocity v by[2]

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

*The code is available on GitHub.[1]

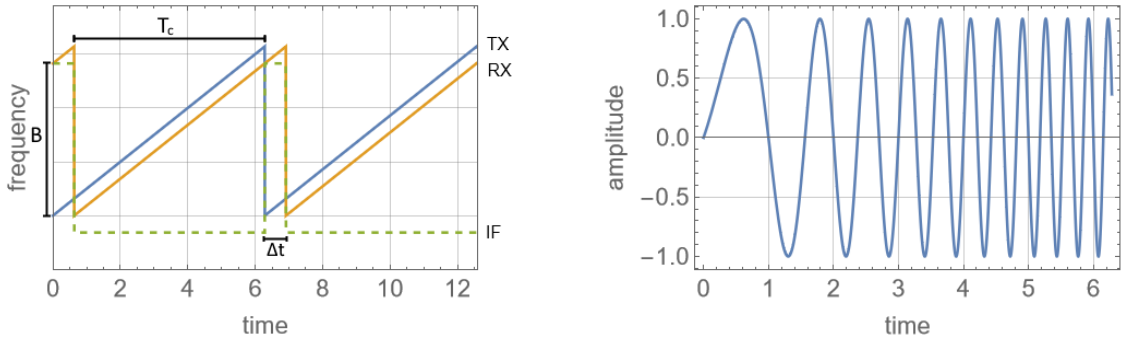

Figure 1: Left: transmitted (TX) and received (RX) sawtooth chirps and the intermediate frequency (IF). Right: illustration of TX chirp waveform. (Own work)

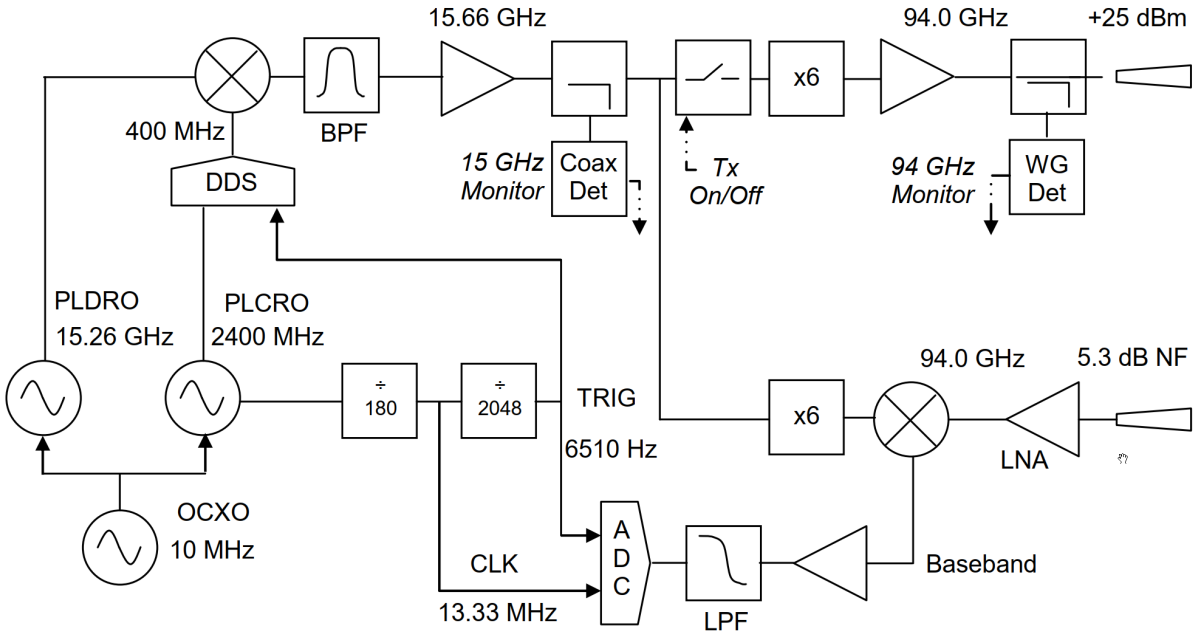

Figure 2: Simplified radar system diagram.[3]

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

In this context, a *frame* is a sequence of CPIs to be averaged together. Non-coherent averaging will increase the signal-to-noise ratio (SNR) and hence improve sensitivity to clouds.[5]

2.3 Doppler moments

Statistical properties of the velocity spectrum are of great interest to meteorologists.

3 Requirements

3.1 Current state of radar software

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

Mark I does not support real-time data acquisition and its processing is limited to an unaveraged power spectrum and range-Doppler plot.

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

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

3.3 Project goals

The main project goals are:

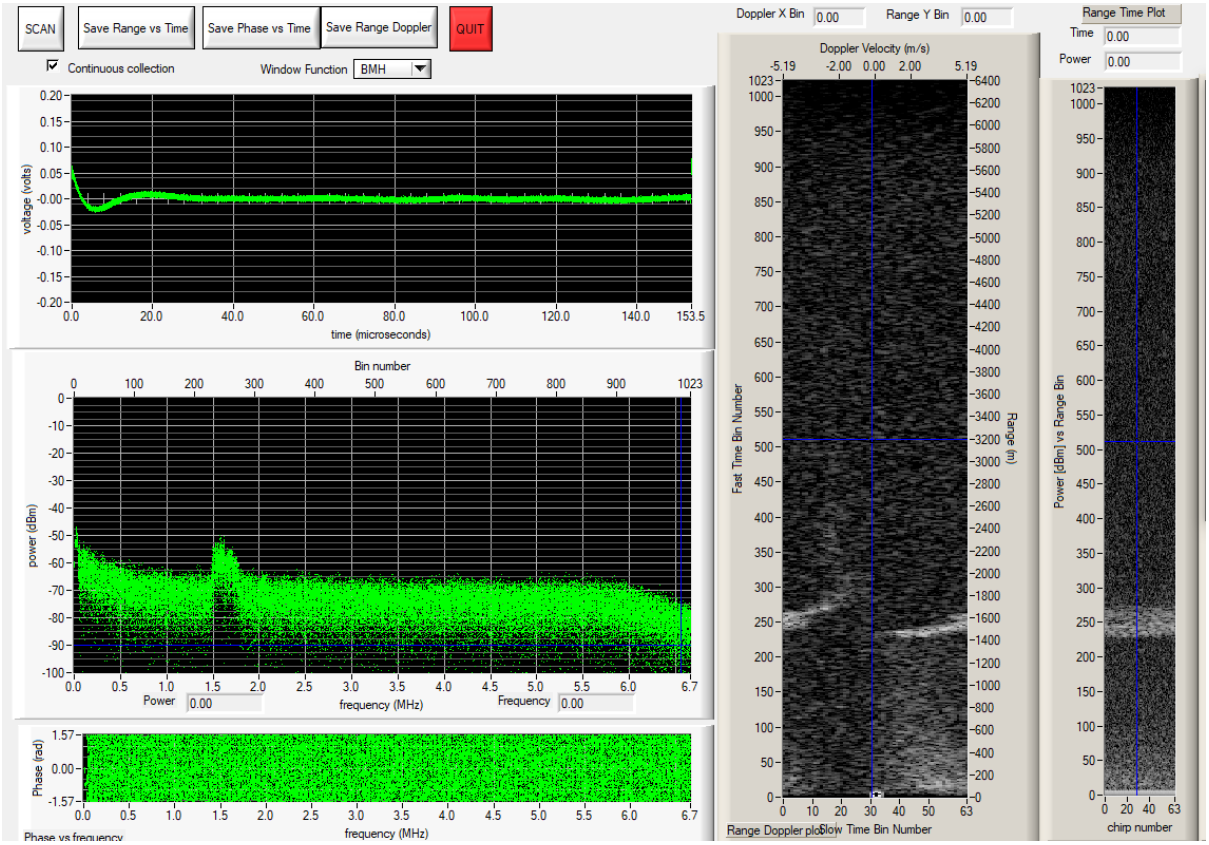

Figure 3: Screenshot of Mark I user interface, taken during precipitation.

Continuous real-time processing. Ensure no frames are dropped and the processing is real-time.

User interface Combine hardware control and sensors into new code. Implement two different modes of display: operational (range and velocity resolution, etc.) and engineering (bandwidth, etc.). Add plot showing reflectivity as a function of range and time.

Signal averaging. Implement averaging of power spectra and range-Doppler data to significantly improve sensitivity to clouds.

Doppler moments. Display velocity spectrum for a particular range bin and calculate its moments.

Calibration. Convert output from dBm to dBZ.

Effect of ambient temperature. Sensors inside the radar enclosure can record the temperature. Record temperature over a long period of time and correlate to radar performance.

Cross-check results with a disdrometer. Use adjacent disdrometer to record the drop-size distribution while raining. The reflectivity can then be calculated and compared to the radar output.

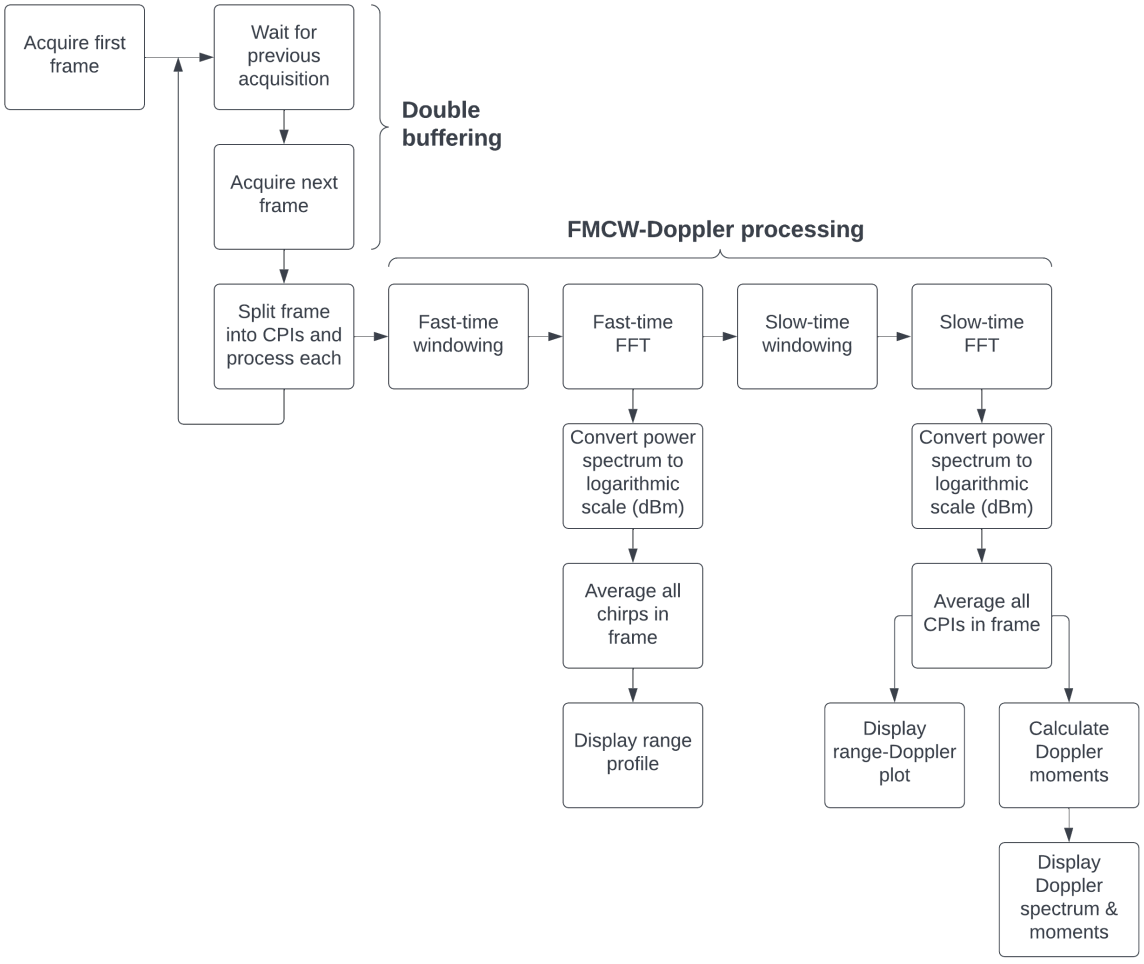

Figure 4: Simplified block diagram illustrating key signal processing steps. (Own work)

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
```

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
    acquire frame into buffer
    wait for previous acquisition to finish
    copy into thread safe queue
end while
```

```
while keep acquiring data do
    copy next frame in thread-safe queue
    process frame
    display processed frame
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 ([6]), 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.

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

Algorithm 3 Key steps in FMCW-Doppler process.

```
for each CPI in the frame do
    copy ADC integers (uint16) into buffer (float64)
    for each chirp do
        apply fast-time windowing to chirp samples
    end for
    execute fast-time FFT
    for each chirp do
        for each range bin in chirp do
            calculate FFT magnitude
            convert to logarithmic scale
            apply fast-time corrections
            apply fast-time windowing correction and FFT normalisation
        end for
    end for
    for each column do
        apply slow-time windowing to each range bin
    end for
    execute slow-time FFT
    for each chirp do
        for each range bin in chirp do
            calculate FFT magnitude
            convert to logarithmic scale
            apply fast-time corrections
            apply slow-time windowing correction and FFT normalisation
            transpose array index and shift so spectrum is centred about DC
        end for
    end for
    display processed frame
    swap A and B pointers
end for
```

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. 5) Fig. 6 shows the effect of applying the Blackman-Harris window is as expected.

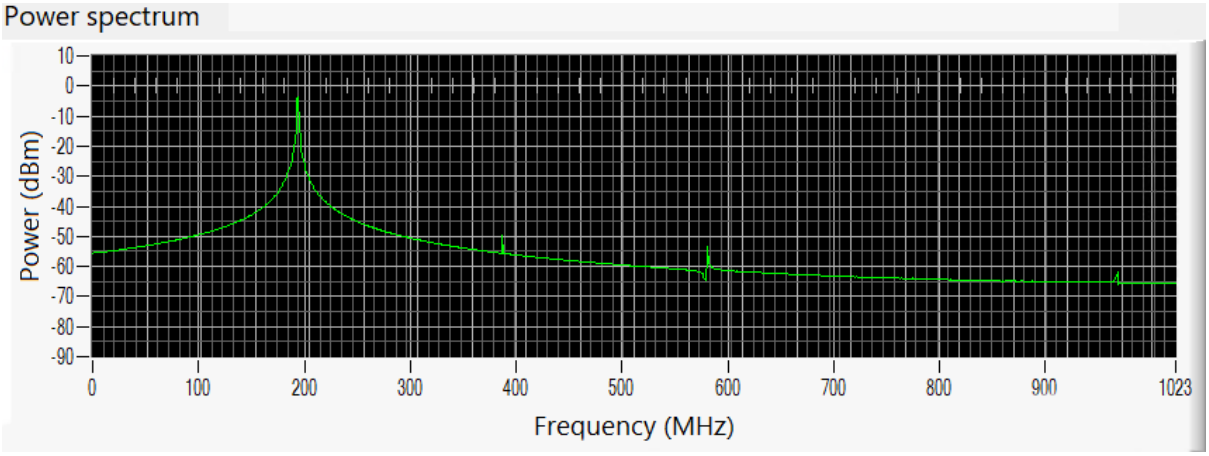

Figure 5: Averaged fast-time power spectrum of sine wave without windowing. Spectral leakage is noticeable at the peak near the 200th range bin. The leakage masks aliases from the signal generator. Note that the x-axis label is incorrect: it should be the range bin number.

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.

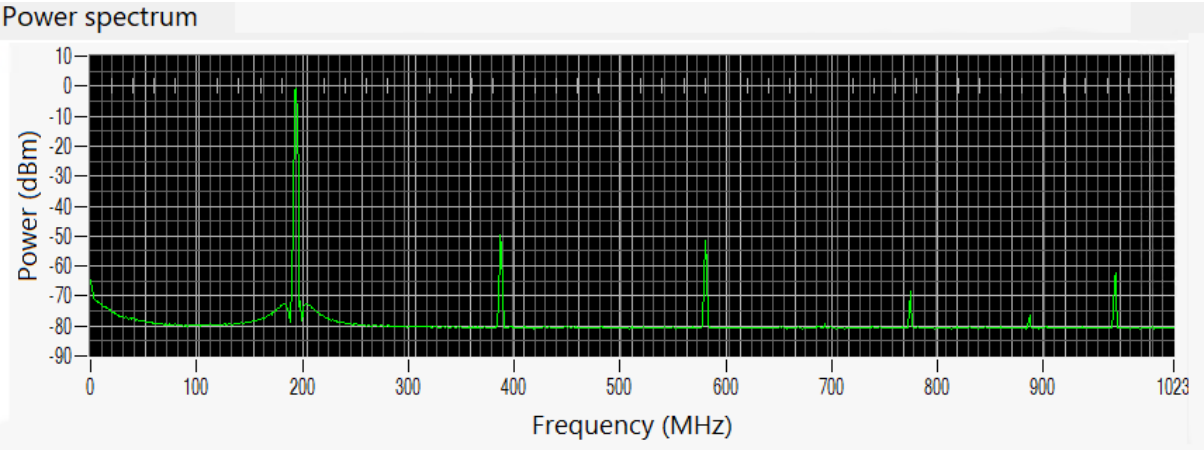

Figure 6: Averaged fast-time power spectrum of sine wave with Blackman-Harris windowing applied. Spectral leakage is greatly reduced and aliases from the signal generator are no longer obscured. Note that the x-axis label is incorrect: it should be the range bin number.

	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 Conclusions

6.1 Future work

6.1.1 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.

6.1.2 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.

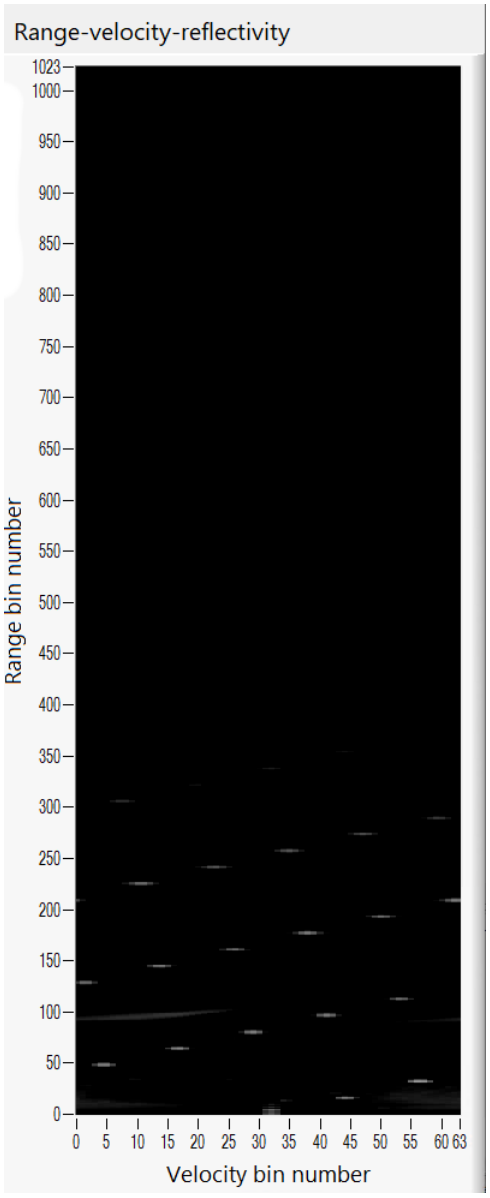

Figure 7: Range-Doppler plot showing a cloud and some precipitation. Averaging over 5 seconds and Blackman-Harris windowing was applied. A hardware issue is visible as a mysterious staircase pattern.

6.1.3 Performance improvements

Multithreading. Many of the calculations in the FMCW-Doppler process are performed on large arrays. Splitting the workload across multiple threads would greatly improve throughput. An easy way of parallelising loops with is to use OpenMP.

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.

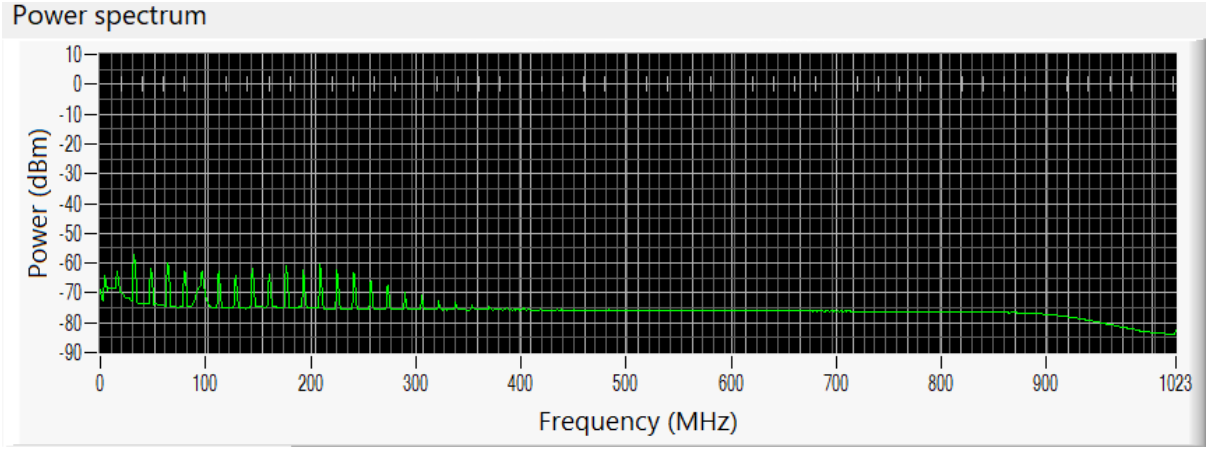

Figure 8: Averaged power spectrum of data in 7. Two small bumps are visible near the 10th and 95th range bins. A hardware issue is visible as a series of spikes.

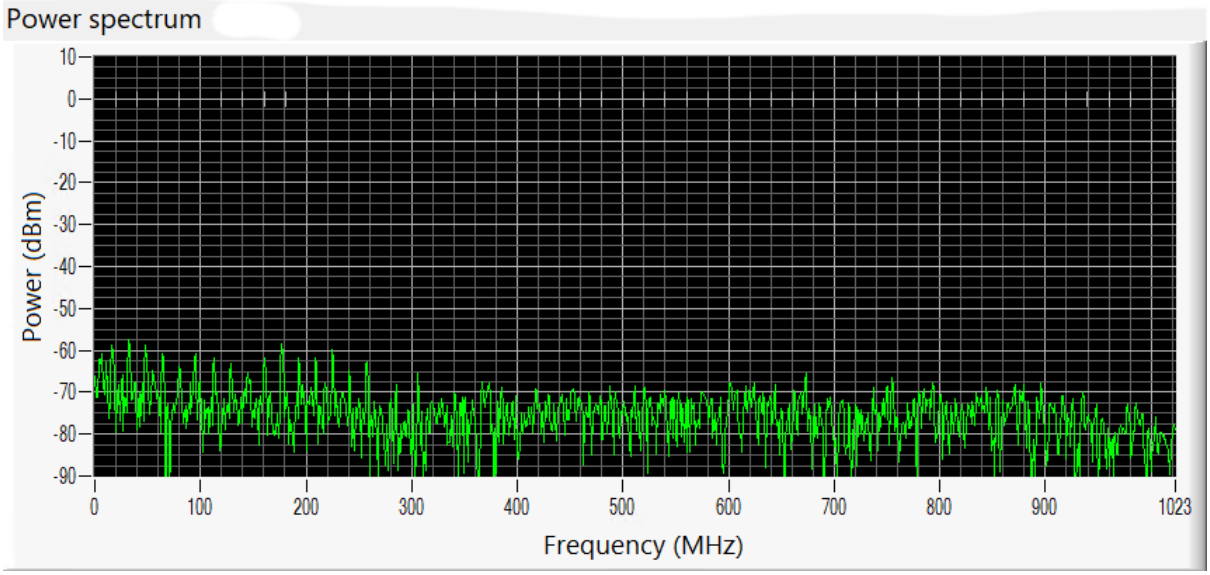

Figure 9: Unaveraged power spectrum of data in 7.

6.2 Adapt code for other radars

Both the Mark I and Mark II code exist as a single source file and are heavily tied to CVI and DAQ card library. The new code was built with portability and modularity in mind, so each component (data acquisition, FMCW processing, user interface) is split into its own source file. Only the user interface module depends on CVI.

The two biggest features were successfully implemented: continuous real-time data acquisition and signal averaging. Progress was made towards extraction of Doppler moments, however some debugging is required. The modular design will allow for easier addition of features and for it to be adapted to other radars. The performance improvements mean the code could be used for radars that require fast response times such as drone detection.

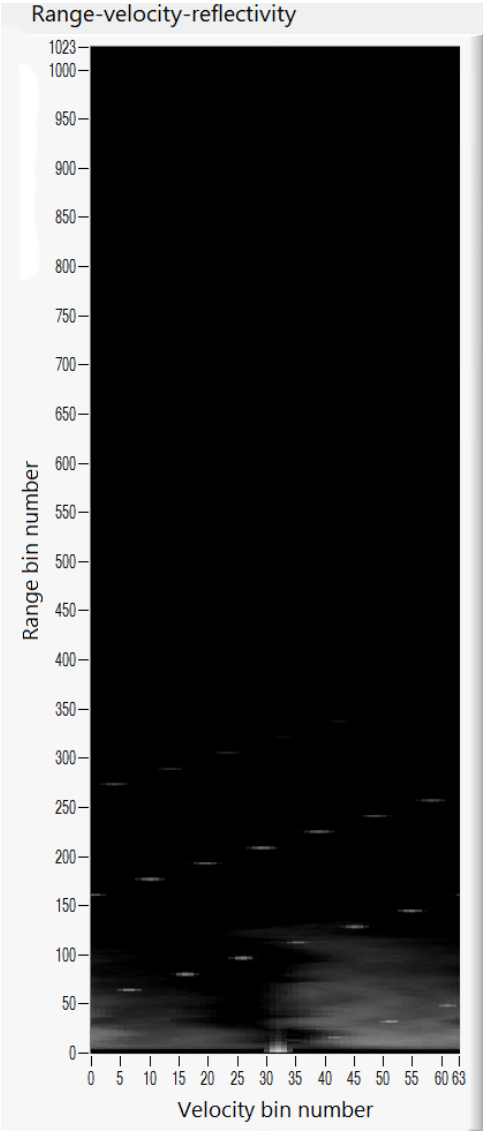

Figure 10: Range-Doppler plot from the new user interface showing heavy rain. Averaging time is 5 seconds and Blackman-Harris windowing has been applied. Some DC noise is visible at zero range and zero velocity (bin number 31).

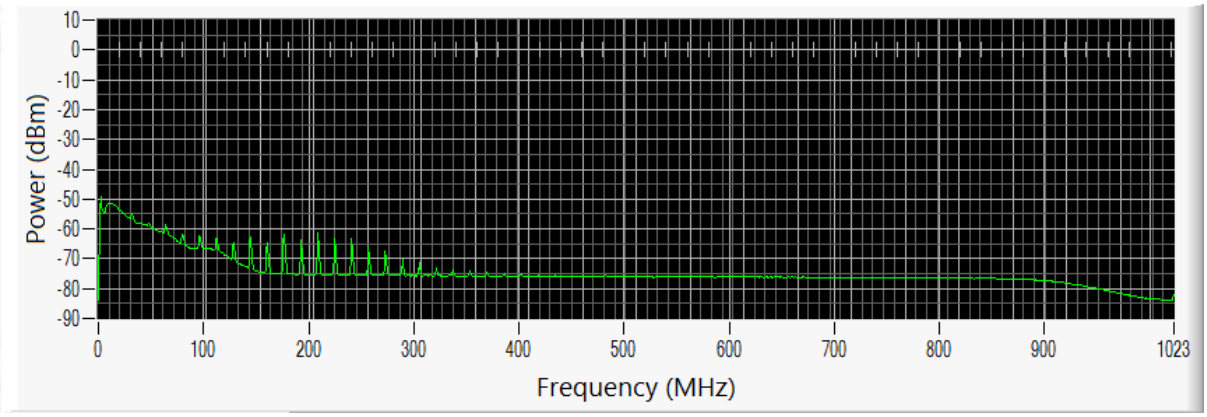

Figure 11: Averaged range profile for the same data in Fig. 10. The rain is visible as a large bulge between range bins 1 and 150.

References

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

[2] 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).

[3] 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.

[4] M. A. Richards, "Doppler processing," in *Principles of Modern Radar*, M. A. Richards, J. A. Scheer, and W. A. Holm, Eds., 1st ed., vol. 1, Scitech Publishing, 2010, ch. 17, pp. 625–673, ISBN: 978-1-891121-52-4.

[5] ——, "Digital signal processing fundamentals for radar," in *Principles of Modern Radar*, M. A. Richards, J. A. Scheer, and W. A. Holm, Eds., 1st ed., vol. 1, Scitech Publishing, 2010, ch. 14, pp. 536–538, ISBN: 978-1-891121-52-4.

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