Design and Evaluation of a Low-Cost Passive Radar Receiver Based on IoT Hardware

Daniel Moser*†, Giorgio Tresoldi†, Christof Schüpbach* and Vincent Lenders*
*armasuisse Science + Technology, Thun, Switzerland
Email: daniel.moser@armasuisse.ch, christof.schuepbach@armasuisse.ch, vincent.lenders@armasuisse.ch
†ETH Zürich, Switzerland
Email: tgiorgio@student.ethz.ch

***Abstract*—Recent years saw an increase in computation power on Internet of things devices such as the Raspberry Pi. It is now common for such platforms to boast multiple CPU-cores with clock rates of 1 gigahertz and higher. We have taken this evolution as a motivator to see how far we can push the limit in performing complex operations on a large amount of data by implementing a passive radar system on the Raspberry Pi. To keep the costs of our system further down, we evaluated the use of low-cost RTL-SDR receivers.**

Our work shows that today's IoT devices allow real-time processing for passive radar applications for both, FM and DAB signals. With our low-cost receiver, we were able to receive echos of aircraft several kilometers away.

I. Introduction

Internet of things (IoT) devices, such as low-cost software-defined radio dongles and Raspberry Pis [1] have recently made air traffic monitoring accessible to the public. As these devices are very affordable, many researchers, aviation enthusiasts, and even corporations have started recording ADS-B and secondary radar (Mode S) signals using them to track aircraft. Web-based platforms such as FlightAware [2], Flightradar24 [3] or the OpenSky Network [4] have emerged which rely on thousands of such IoT devices to operate popular global live flight tracking services on the Internet.

The success of these IoT devices for receiving aircraft signals has motivated us to investigate their applicability for building passive radar systems. In contrast to ADS-B or secondary radar, a passive radar system tracks aircraft by leveraging signal reflections from illuminators of opportunity such as FM, DAB, or DVB-T transmissions. Despite the popularity of ADS-B and secondary radar for air traffic surveillance, passive radar is an interesting alternative as it allows to track aircraft or drones which are not equipped with or have their transponder switched off.

Passive radar is, however, inherently more computationally expensive and requires much higher signal quality than ADS-B or Mode S decoding. These signals are not modulated for the purpose of aircraft tracking and the signal strength of the reflections is naturally much weaker compared to the high-power signals transmitted actively by the aircraft. A key question we evaluate in this work is therefore whether the limited capabilities of IoT devices are sufficient to detect aircraft using passive radar algorithms.

To our advantage, we have experienced a vast increase of computing power, not only in high-end devices, but also in IoT computing platforms such as the more recent versions of the Raspberry Pi. While a decade ago, a Linux cluster of six computers was capable of producing one range doppler map every 5 seconds [5], we will show that today, with computing equipment for less than *100USD*, we can implement a passive radar receiver that can detect aircraft reasonably well. We rely on a Raspberry Pi 3 Model B equipped with an RTL-SDR radio stick to receive the radio signals, digitise them, and perform our signal processing. To perform complex signal processing operations, we design a software architecture which relies on the embedded GPU of the Raspberry Pi, leveraging the parallel architecture available on such processing units.

The contributions of our work is to show how today's low-power and affordable IoT devices can be used to conduct computation-heavy signal processing to an extent where we can actually see targets from signals purely processed on a recent Raspberry Pi computer. We have shown that – despite the low resolution of the RTL-SDR – its performance is still suitable for passive radar applications.

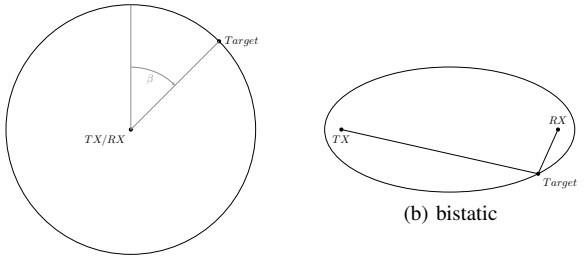
II. Background

The concept of passive radar systems has already been known for several decades and first implementations date back to World War II [6]. Main advantages for such a system include much lower energy consumption due to omission of own transmission stations and – apart from visual detection of its antennas – higher chances of the radar system staying hidden from malicious third parties.

Passive radar utilises signals from a so-called *illuminator of opportunity* whose signals can be used by one or more receivers. Such a receiver is usually equipped with at least two phase-coherent channels. With one antenna pointed towards the actual illuminator – called *reference channel*, while the *surveillance channel* antenna(s) is/are pointed towards the sector of interest. In this work, we focus on passive radar systems which are equipped with only one surveillance channel. The distributed design of the transmitter and receiver mean that passive radar systems in general are *bistatic* as opposed to classical *monostatic* radar systems. The large turning parabolic antenna in classical radar systems will receive both range and bearing of the echo. Passive bistatic radar systems on the other

978-1-7281-1679-2/19/$31.00 ©2019 IEEE

hand measure the time delay of the echos that do not constitute the absolute range of the target and the receiver but rather the full additional time the signal travels from transmitter to the target and from there to the receiver. Targets in passive bistatic radar hence can only be located to an ellipse in single-receiver, single-transmitter passive radar setups (see Figure 1). Additionally, such systems measure the Doppler shift from the relative speed of the target to the transmitter and the receiver.



(a) monostatic (b) bistatic

Fig. 1. Comparison of the monostatic (a) and bistatic (b) radar topologies. Usually, monostatic radars can extract both range and bearing from the echos while passive bistatic radars with a single surveillance channel can only extract the bistatic range and Doppler shift.

Previous work needed a cluster of six computers to calculate a range-Doppler map (RDM) in around 5 seconds [5]. The advent of ARM based low-cost computing platforms has pushed their computational power past what was considered cutting edge one decade ago. Recent years, therefore, saw efforts to bring passive radar processing to embedded devices, e.g. the Nvidia Jetson platform offering 192 CUDA cores and a quad-core ARM CPU[7]. Our work intends to push the boundary on what is actually possible for a processing setup of less than 100USD. This opens new passive radar applications for crowdsourcing or various research use cases.

III. System Architecture

This section describes the system architecture of our passive radar system and details on the signal processing optimisations we took to speed-up the calculations.

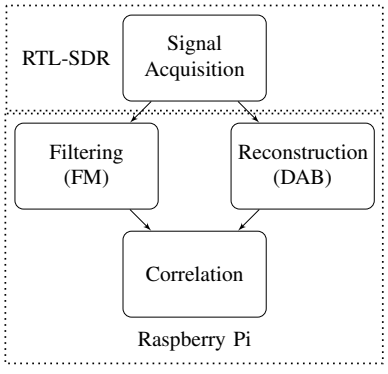


Fig. 2. High level flow of signal data within our processing pipeline.

Figure 2 gives a high-level overview of our architecture. Signal acquisition is performed through the RTL-SDR software-defined radio platform. It hands over the raw IQ samples to either the filtering or the reconstruction stage, depending on whether we use frequency modulation (FM) broadcast or digital audio broadcast (DAB) signals as the basis of our computations. This distinction arises from the fact that DAB is transmitted as a digital signal as opposed to the analogue FM transmissions. For FM-based operation, we need to record the reference channel in order to filter the surveillance signal and to provide the reference signal for the correlation stage. For DAB, on the other hand, we only need the surveillance signal, from which we can reconstruct the reference signal.

The processed samples are finally handed to the correlation stage, where an RDM is computed, which then shows the received echos. We omit the detection stage in this work as actual localisation and tracking of targets using passive radar systems requires multiple sensors deployed at different locations. The localisation therefore needs to be conducted at a centralised hub, which, in our architecture, would then perform the actual target detection, localisation and tracking.

	FM	DAB
Frequency	88 - 108 MHz	174 - 230 MHz
Modulation	analogue (FM)	digital (OFDM/DQPSK)
Bandwidth	0.05 MHz	1.536 MHz
Availability	Global	Local
Network	multi-frequency	single-frequency
Content dependency	yes	no

TABLE I
COMPARISON OF FM AND DAB SIGNAL'S PROPERTIES.

A. Illuminators of Opportunity

We have implemented our system for both analogue FM as well as the digital DAB radio transmissions. Both offer unique advantages – but on the other hand have distinct disadvantages. Table I gives an overview of the two illuminator signals.

The performance of an FM-based passive radar system is highly dependent on the transmitted content. Voice transmissions are the least useful transmissions due to the lower entropy in the signal and additional pauses in between words and sentences. Music transmissions from Pop and Rock with their high dynamic compression lead to higher entropy in the spectrum and do not tend to have high correlation with other parts of the transmission. DAB transmissions, on the other hand, are less dependent on the content of an individual transmission, because, there are multiple program streams interleaved in both frequency and time. Also, the bit stream is interleaved and scrambled to add additional entropy to the stream, making demodulation and decoding more robust.

B. Radio Frontend

Our proposed low-cost system makes use of an RTL-SDR platform. Specifically, we use the RTL-SDR V3 dongle [8] equipped with a TCXO which provides a frequency stability of better than $0.2\,PPM$ [9]. This software-defined radio uses a Realtek RTL2832U chip, featuring a 7-bit ADC and a Rafael Micro R820T2 chip as radio tuner. The acquisition of FM signals for passive radar purposes requires two phase-coherent

978-1-7281-1679-2/19/$31.00 ©2019 IEEE

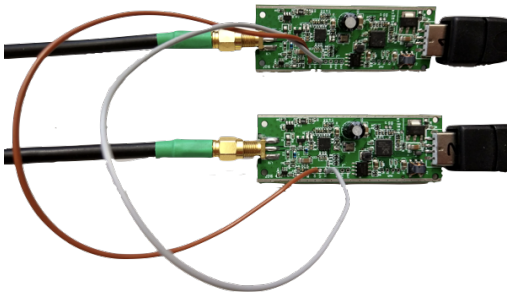


Fig. 3. Our phase coherent setup, consisting of two RTL-SDR dongles with the clock signal being distributed from one PCB to the other through the added cables.

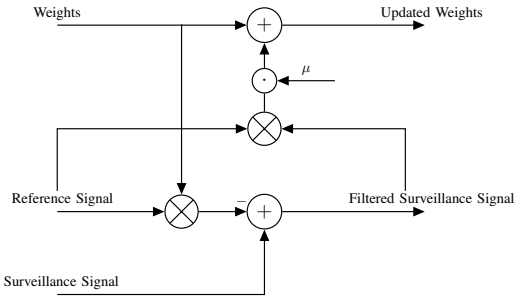


Fig. 4. Block diagram of the FBLMS filter.

radio frontends. The RTL-SDR V3 offers solder pads on its PCB to extract the clock signal from one receiver and feed it into additional devices (see Figure 3). For the reception of FM signals, we use two phase-coherent RTL-SDRs with sampling rates of $240\,kS/s$ each and for DAB we use a single RTL-SDR with a sample rate $2.048\,MS/s$.

C. Computation Platform

Our passive radar system is implemented for the Raspberry Pi platform. In particular, we use the version 3 model B, which is equipped with a 1.2GHz quad-core ARM Cortex-A53 processor with a dual core VideoCore IV GPU [1]. In general, we aimed to write the software pipeline in a hardware agnostic way and use compiler flags to either use the generic FFTW library [10] or GPU_FFT [11] for instances running on the Raspberry Pi. A disadvantage that comes with using the Raspberry Pi's GPU is that we are not able to calculate arbitrary FFT lengths but only those of powers of two. This limitation will dictate our design decisions for the final length of the correlations in both the time and the frequency domain.

Our pipeline uses a modular approach, where each processing step is its own program and – at the moment – each of them read the source signal from a file and write their output to a new file.

D. Signal Processing

We implemented the processing steps as described below and as shown in Figure 2.

1) Direct Signal Interference (DSI) Suppression: In an FM setup, the surveillance reception chain is still highly influenced by the very strong direct signals. In order to make echos more visible in the surveillance signal, we first need to remove the direct signal interferences.

To this end, we make use of a fast-block least mean square (FBLMS) filter. It offers a good compromise between computation time and DSI removal performance [12]. The FBLMS filter is an adaptive filter working on signal blocks of both the reference and the surveillance signal. The filter produces weights to estimate the amount of reference signal in the surveillance channel which are then removed from the surveillance signal and output as the assumed "clean" surveillance signal. The filtered signal and the reference signal are then used to update the weights of the filter for the next block. Figure 4 gives a schematic overview of the filter design.

2) Reference Signal Reconstruction: As the DAB system omits the reference signal chain, we need to reconstruct the original transmitted signal from the surveillance chain. We therefore demodulate the OFDM signal and map the point-clouds of the $\pi/4$ differential QPSK constellation to an optimal, "noise-less" signal and treat this as the assumed direct signal (see Figure 5). This approach is analogous to various previous work [13], [14]. For further processing steps we remove the null symbol since it does not add much energy and is thus not suitable for the correlation process.

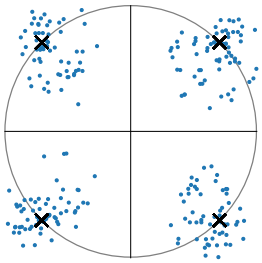


Fig. 5. The demodulated OFDM symbols (dots) are mapped to one of the four expected carrier values (x) on the unit circle according to their positioning in the four quadrants of the complex plane.

3) Range-Doppler Map Generation: In the final stage of our pipeline, we take the surveillance and the reference signals and correlate them over a certain integration time, in order to lift the weak echos above the noise floor. In a classical approach [5], one would create frequency shifted copies of the reference signal and correlate each of the copies to the surveillance signal and stack the result into a two dimensional matrix.

To reduce computational load, we instead use the batches algorithm [15] as the basis of our pipeline. This algorithm approximates the phase progression by assuming a constant phase during each processed batch. In order to create the RDM, we cut both input signals into equal-sized, time-aligned batches. The corresponding batches from each channel are cross-correlated using the fast Fourier transform (FFT) and the results stacked into a two dimensional array. Finally, we compute the FFT of the two dimensional array column-wise which yields the RDM. Figure 6 gives a visual overview

978-1-7281-1679-2/19/$31.00 ©2019 IEEE

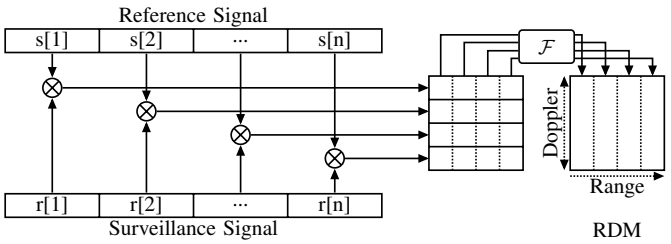


Fig. 6. Generic schematic of the Range-Doppler map generation using the batches algorithm. The $\otimes$ symbol denotes the cyclic cross correlation.

over the RDM generation procedure as implemented in our processing pipeline.

This step is quite heavy on the processing resources as it requires a total of three Fourier transforms per batch. One Fourier transform plus an additional inverse transform are needed for the correlation of the two corresponding batches and one Fourier transform operation is performed on each column of the intermediate two-dimensional array. For DAB, we use an FFT size of 2048 samples for the range domain and 512 samples in the Doppler domain which gives us a coherent processing interval (CPI) of $0.6\,s$ and an integration time of $0.5\,s$ as we need to remove the cyclic prefix before each DAB frame. These values translate to a range resolution of $\sim 150\,m$ and a Doppler resolution of $1.56\,Hz$. For FM processing, we use an FFT size of 512 samples for both range and Doppler domains, which yields an integration time of $1.1\,s$.

As our reference signal reconstruction (DAB) and DSI suppression (FM) stage already transforms our recorded time signal to the frequency domain and also generates the reference signal in the frequency domain, we can already omit the initial FFT operation for the cyclic cross correlation. To further improve the performance of our program, we do not immediately perform an inverse Fourier transform in the cross correlation but rather fill the two dimensional matrix first and then perform a two dimensional FFT over the whole matrix.

Because the unstable oscillator in our SDR causes frequency drifts at the radio frontend, we perform a recentering of each RDM through automatic detection of the zero-Doppler line as a final step before the output.

IV. Evaluation

In this section, we evaluate our prototype system based on the *processing overhead* and *passive radar* performance. We mainly show the results for our DAB based system and include some data on the processing performance of the FM system.

A. Processing Performance

Even though the Raspberry Pi 3 platform offers a quad-core ARM processor, its computational power alone is too low to offer real-time processing of the whole passive radar pipeline. We therefore evaluate how much speed-up can be achieved when using the GPU for FFT processing.

Figure 7 shows the processing times for reference signal and range-Doppler map generation. We performed the computations in the following configurations:

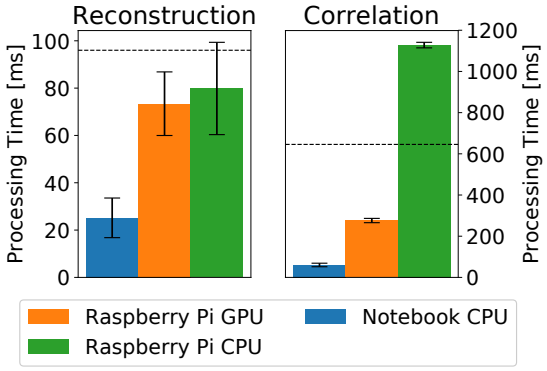


Fig. 7. Performance measurements for reference signal reconstruction of one DAB frame (left) and correlating the reference and surveillance data and building a range doppler map (right). Each bar shows the mean processing time and the standard deviation. The dashed lines mark the real-time limit for both computations.

Notebook: For performance reference, we used a current notebook equipped with an Intel Core i7-6500 processor with 4 logical 2.5 GHz cores running Ubuntu 18.04 LTS 64bit.

Raspberry Pi CPU: We compiled our software without changes to the source code on the Raspberry Pi 3 Model B to get the baseline performance of the platform's CPU only. It ran Raspbian 9 Stretch, a fork of Debian optimised for the Raspberry Pi, for both configurations.

Raspberry Pi GPU: We adapted the source code of our software to perform the FFT operations on the Raspberry Pi's GPU.

Figure 7 depicts the performance of these configurations for both the signal reconstruction and the range-Doppler map generation. We calculated the average and standard deviation for the processing time of the reference signal generation over approximately 5.500 DAB frames. The reference time for processing a single DAB frame ($96\,ms$) is denoted by the dashed line. All our configurations perform on average faster than real-time for the reference signal reconstruction. As expected, the notebook performs best and reconstructs a reference frame in $25.2 \pm 8.4\,ms$. The Raspberry Pi's performance for both configurations are very close. With the GPU performing only slightly faster than the CPU, with $73.4 \pm 13.4\,ms$ and $79.9 \pm 19.5\,ms$. These numbers reveal that using the GPU only improved the average processing time by less 10.0%. While this fact might be surprising at first, these performances are owed to the current software architecture where each DAB frame is reconstructed sequentially. However, copying the data to the relevant buffer, initiating the GPU call and reading the result for a single FFT adds a lot of overhead. In a future iteration of our implementation, we intend to parallelise the OFDM demodulation so we can perform multiple FFTs in one batch, again leveraging the faster FFT performance of the GPU.

For the generation of the range-Doppler map, we computed the mean and standard deviation over a total of 127 range-Doppler maps. Again the notebook outperforms the Raspberry

978-1-7281-1679-2/19/$31.00 ©2019 IEEE

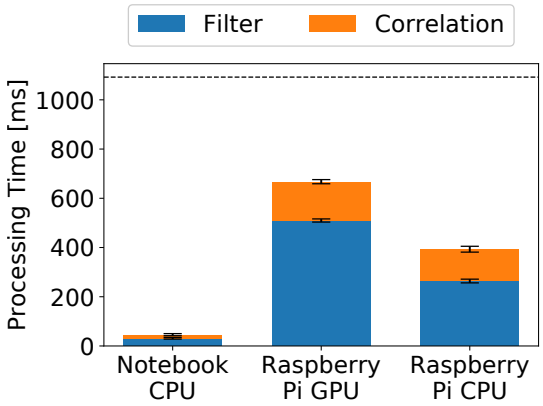


Fig. 8. FM performance measurements for direct signal interference filtering (blue) and correlating the reference and surveillance data and building a range doppler map (orange). Each bar shows the mean processing time and the standard deviation. The dashed line mark the real-time limit for the computations.

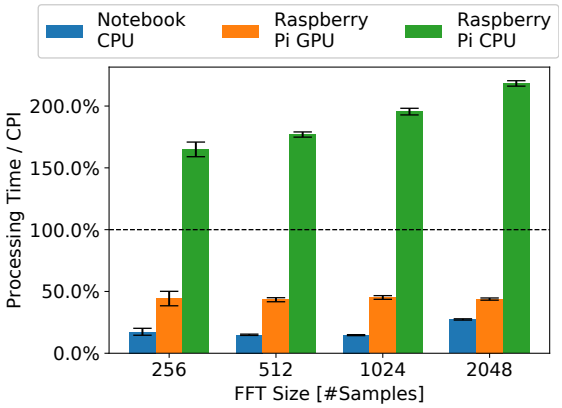


Fig. 9. Performance evaluation for range-Doppler map processing with varying FFT sizes in the Doppler domain relative to the CPI (dashed line). The results for an FFT size of 512 samples corresponds to the correlation part of Figure 7.

Pi. It was able to compute a RDM in $60.3 \pm 8.6\,ms$. The integration time of $645.7\,ms$ is marked by the dashed line in the figure. All configurations performing below said line provide real-time generation for range-Doppler maps. For this step, we clearly see the advantage of using the Raspberry Pi's GPU to offload the FFT computations. On the CPU the Raspberry Pi generated one RDM every $1127.9 \pm 13.0\,ms$ while the GPU pushed this number down to $276.1 \pm 10.4\,ms$, offering a performance boost of a factor of approximately 4. Because we actually perform a 2D-FFT over the resulting matrix, we can first perform 512 FFTs with a size of 2048 and then 2048 FFTs with a size of 512 samples. Handing this amount of data directly to the GPU where the FFTs are performed in parallel, the performance is a lot faster than the CPU computing all these FFTs in sequence. Making use of the GPU allows us to reach real-time performance on a Raspberry Pi.

For our FM processing pipeline, we processed approximately 200 range-Doppler maps with a size of 512 times 512 samples. The results are shown in Figure 8. We measured an average of $509.8 \pm 6.4\,ms$ for the filtering stage and $157.7 \pm 8.5\,ms$ for generating the range-Doppler map on the Raspberry Pi's GPU. Due to the decreased FFT sizes, the FM pipeline runs faster than its DAB counterpart. The CPI for the range-Doppler map was $1\,s$. Running the same pipeline on the Raspberry Pi's CPU yields lower processing times due to the sequential FFT processing in the DSI filtering stage, where the call to the GPU for an individual FFT operation is slower than directly calculating it on the CPU.

To evaluate how the size of the range-Doppler map influences the processing time, we varied the FFT size (i.e. the integration time while keeping the batch length constant) in the Doppler domain in our DAB pipeline (see Figure 9). As the integration time increases with the higher FFT size, we present the required processing time relative to the CPI duration. It is apparent that even with a size of 256 FFTs in the Doppler domain, the Raspberry Pi's CPU is too slow to process the data in real time. The Raspberry Pi's GPU manages to process the range-Doppler map within less than $50\,\%$ of the CPI duration. While the processing time for the Raspberry Pi's GPU remains constant relative to the CPI, the longer FFT sizes start to narrow down the performance gap between the notebook's CPU and the Raspberry Pi's GPU. Such long integration times of two seconds and more might however not be useful in an avionics context due to the high velocities at which aircraft travel.

Our results have shown both the FM and DAB systems being able to perform passive radar computations in real time. The over-all performance of the Raspberry Pi platform even suggests that additional processing steps – such as additional filtering or CFAR processing – could actually be implemented while still performing faster than real-time.

B. Passive Radar Performance

To evaluate the performance of our system as a passive radar receiver, we went to Zurich airport and set up our equipment at the car park at the end of the runway. The signals were recorded using a Yagi antenna pointed parallel the direction of the runway facing away from the airport.

The results of our DAB measurements are shown in Figures 10 and 11. Figure 10 shows a max-hold over a series of 753 range-Doppler maps. Several distinct tracks of aircraft during their approach to the airport can be seen in the lower left. The aircraft started to appear at a bistatic range of approximately $30\,km$. There is also a weaker track visible extending from the zero Doppler line at approximately $45\,km$ towards $55\,km$ bistatic range and $150\,Hz$ Doppler. As our experimental site was located in a vale, longer range detection of echos was not feasible. The figure also shows strong interferences close to zero range and zero Doppler which seems to be related to the receiver's frequency instabilities [16]. These interferences appeared periodically after a number of generated RDMs. Figure 11 shows a filtered version of the same data, revealing an additional track starting from the left around $50\,Hz$ Doppler shift.

978-1-7281-1679-2/19/$31.00 ©2019 IEEE

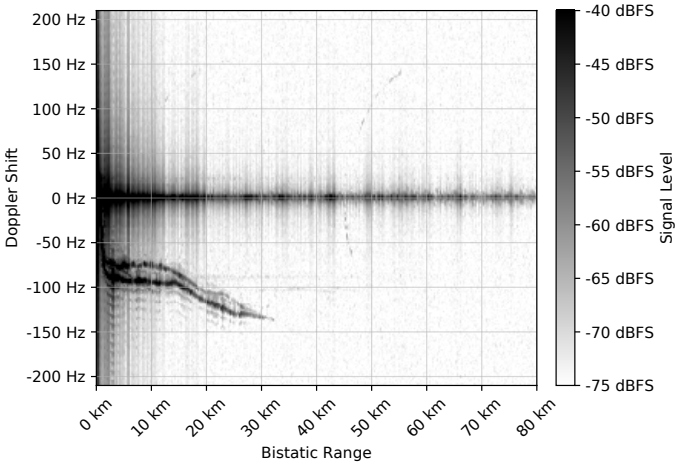


Fig. 10. Max-hold of a series of 753 range-Doppler maps over a duration of 486 seconds based on our DAB setup.

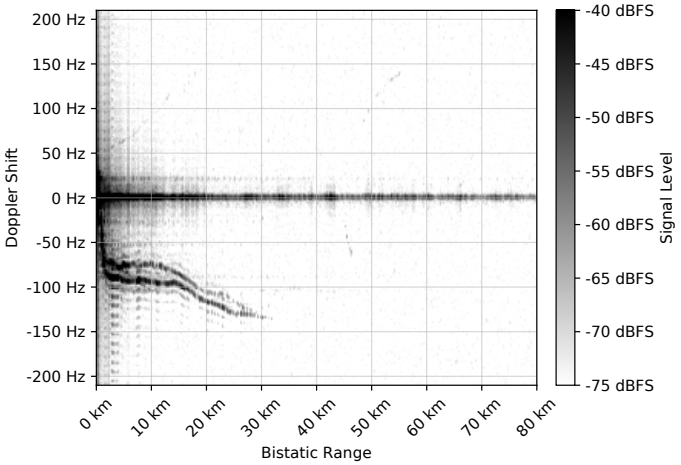


Fig. 11. A visualisation of the same data from Figure 10 with frames removed where the noise exceeds 10% of the average background noise.

Even though we had detections in RDMs based on FM signals, due to the analogue modulation the results were too noisy to visualise in a meaningful way without additional CFAR processing.

V. Conclusion

In this work, we have shown that the advent of IoT devices is allowing computations on low-cost, small footprint platforms where a decade ago a whole cluster of computers was necessary. We presume that the performance figures of our system could still be drastically improved using more optimised code, data-structure optimisations, and processor specific instruction sets. As future work, we will investigate how the current Raspberry Pi's ARM Neon architecture could be used for optimisation of sequential FFT and non-FFT processing stages.

Our findings proved that low-cost radio platforms such as the RTL-SDR – despite being limited and noisy compared to professional SDR platforms – are still more than capable to acquire and digitise signals in a quality suited for passive radar operations. Our system which costs less than 100USD is capable of detecting aircraft with considerable SNR up to 30 km bistatic range.

Acknowledgment

We would like to thank Prof. Dr. Laurent Vanbever from ETH Zurich's Networked Systems Group for his support and feedback. This work was partially supported by the Zurich Information Security and Privacy Center (ZISC). It represents the views of the authors.

References

[1] E. Upton. (2016, February) Raspberry Pi 3 on sale now at $35. [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/

[2] FlightAware. Accessed: 2019-02-18. [Online]. Available: https://www.flightaware.com

[3] Flightradar24. Accessed: 2019-02-18. [Online]. Available: https://www.flightradar24.com

[4] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing Up OpenSky: A Large-scale ADS-B Sensor Network for Research," in *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, ser. IPSN '14, Berlin, Germany, April 2014, pp. 83–94.

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

[6] H. Griffiths and N. Willis, "Klein heidelberg—the first modern bistatic radar system," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 4, pp. 1571–1588, Oct 2010.

[7] J. L. Sendall *et al.*, "Implementation of a low-cost bistatic radar," Master's thesis, University of Pretoria.

[8] RTL-SDR V3. Accessed: 2019-02-18. [Online]. Available: https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/

[9] R. Calvo-Palomino, F. Ricciato, D. Giustiniano, and V. Lenders, "LTESS-track: A Precise and Fast Frequency Offset Estimation for Low-cost SDR Platforms," in *Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, ser. WiNTECH '17. New York, NY, USA: ACM, 2017, pp. 51–58. [Online]. Available: http://doi.acm.org/10.1145/3131473.3131481

[10] M. Frigo, "A fast fourier transform compiler," in *Proceedings of the ACM SIGPLAN 1999 Conference on Programming Language Design and Implementation*, ser. PLDI '99. New York, NY, USA: ACM, 1999, pp. 169–180. [Online]. Available: http://doi.acm.org/10.1145/301618.301661

[11] A. Holme. (2015, March) GPU_FFT. [Online]. Available: http://www.aholme.co.uk/GPU_FFT/Main.htm

[12] J. L. Garry, C. J. Baker, and G. E. Smith, "Evaluation of Direct Signal Suppression for Passive Radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3786–3799, July 2017.

[13] C. Schüpbach, C. Patry, F. Maasdorp, U. Böniger, and P. Wellig, "Micro-UAV detection using DAB-based passive radar," in *2017 IEEE Radar Conference (RadarConf)*, May 2017, pp. 1037–1040.

[14] O. Mahfoudia, F. Horlin, and X. Neyt, "On the feasibility of DVB-T based passive radar with a single receiver channel," in *International Conference on Radar Systems (Radar 2017)*, Oct 2017, pp. 1–6.

[15] C. Moscardini, D. Petri, A. Capria, M. Conti, M. Martorella, and F. Berizzi, "Batches algorithm for passive radar: a theoretical analysis," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 2, pp. 1475–1487, April 2015.

[16] C. Schüpbach and S. Welschen, "Direct signal interference mitigation by slow-time frequency correction for ofdm-based passive radar," in *2018 19th International Radar Symposium (IRS)*, June 2018, pp. 1–10.