

Development and Calibration of a Low-Cost Radar Testbed Based on the Universal Software Radio Peripheral

Jonas Myhre Christiansen, Norwegian Defence Research Establishment, Kjeller, Norway

Graeme E. Smith, The Ohio State University, Columbus, OH, USA

INTRODUCTION

This article presents the development, calibration, and preliminary testing of a software-defined radio (SDR) suitable for use as a cognitive radar (CR) testbed. Experimental testbeds are critical for CR research. The dynamic response of the radar to the environment limits the suitability of simulations and prerecorded data for testing. If a simulation is used, it would require a very high degree of fidelity. Realistic models of clutter and target are needed at a level suitable to simulate received waveforms on a pulse-by-pulse basis. Such low-level modeling is possible, but results in high-processing overheads and long run times. When prerecorded data are used, it must be oversampled such that the CR processing can demonstrate adaption by subsampling. Such methods can work, as shown by Bell [1] and Mitchell [2], but place limitations on what can be achieved. Furthermore, there are few publicly available datasets with suitable oversampling. A previous article by Smith [3] introduced the cognitive radar engineering workspace (CREW) system developed by The Ohio State University. While the CREW system has been invaluable for conducting CR research [4], [5], it is cost prohibitive for widespread deployment. Here, we report on a smaller form-factor testbed that uses a USRP [6] for its digital backend, and can be fabricated for a few thousand dollars.

Authors' current addresses: Jonas Myhre Christiansen, Norwegian Defence Research Establishment, 2007, Kjeller, Norway, E-mail: (jonas-myhre.christiansen@ffi.no). Graeme E. Smith, The Ohio State University, Columbus, OH 43210, USA.

Manuscript received March 31, 2019, revised August 21, 2019, and ready for publication November 13, 2019.

Review handled by Albert Huizing.

0885-8985/19/\$26.00 © 2019 IEEE

Research on USRP's and other software-defined radio (SDR) platforms has been reported for several years with a notable uptick in such publications since 2011. An article from 2011 [7] demonstrated a USRP-based frequency-modulated continuous wave radar for weather surveillance applications. It demonstrated the ability to detect multiple targets using delay lines in a loop-back configuration. Moreno et al. [8] discussed the potential for USRP-based radar, and demonstrated an experimental characterization of a system based on a USRP NI2920. Other work [9] demonstrated a simple experimental radar using the USRP B210. It showed range profiles with target responses attributed to a large building. These publications show the potential of using USRPs in experimental testbeds for radar. Kirk et al. [10] have used the USRP in an experimental radar system for a CR application focusing on radio frequency interference (RFI) mitigation. The method introduces notches into the transmit waveform bandwidth based on the sensed spectral environment. This requires notching the spectral content of the waveform on a pulse-by-pulse basis, and, hence, the timing requirement is extreme. Previous work by the authors [11] demonstrated an experimental radar system with detection performance for air targets of several kilometers. Therefore, adaptive pulse repetition frequency (PRF) selection on a dwell-by-dwell basis was demonstrated and the flexibility of selecting waveform parameters was explained.

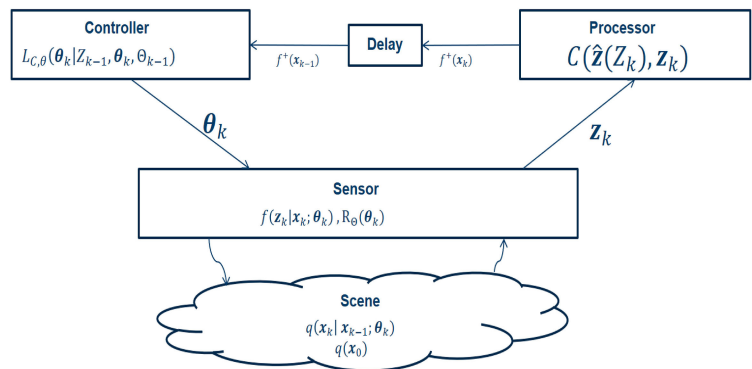
SDRs are attractive because they increase the flexibility of radar systems. When the radar's architecture is defined in software, it is easy to change. Developing radars that have different operating modes become much easier than hard-coded systems, where waveforms and other parameters requires more effort to change. Alternatively, the same physical system can be deployed for a completely different activity by downloading new software to it. Invariably, there is some radio frequency (RF) hardware that remains. The more flexible this is made, the



more diverse the operating modes and uses of the radar can be. The flexible RF front-end can be viewed as providing a large parameter space for the radar to operate in. The precise parameter selections are then a function of the software. It is the presence of this large, selectable parameter space that gives rise to a desire to make radars cognitive. In an ideal situation, the radar parameter selections would be matched to the environment to maximize performance—for example, avoid interference, maximize SINR, minimize track variance, etc. The designer cannot know ahead of time what the environment will be to a sufficient level of detail to match the parameter selections to it. Instead, if performance is to be maximized, the radar must decide autonomously how to utilize the parameter space in response to the environment measurements it is making. It is thus needed to autonomously decide on operating parameters that leads to the field of CR.

The best example available for system performance optimization in response to stimuli is human cognition [12]. In human cognition, sensory inputs are processed through the central nervous system and result in actions being undertaken. When the actions are executed, the relation of the person to the environment changes, and the sensory inputs change accordingly. New actions are then decided on and executed. This process continues in an endless cycle. Parallels can be drawn to radar sensing using SDR with a large parameter space and an algorithm that decides radar parameters based on perceptions. The rise of the SDR and the interest in CR therefore go hand in hand.

This article shows the development and characterization of a radar testbed based on the USRP, which allows testing of some adaptive or cognitive algorithms. The testbed is flexible in parameter selection, which allows for selection of radar parameters on a dwell-by-dwell basis. As shown previously in this section, such a testbed is important in the field of cognitive and adaptive radar since the selection algorithms usually are dependent on the environment in some sense. The testbed is relatively low cost.



To put SDR used in radar in context with the field of adaptive and cognitive radar (CR). Section Introduction introduced SDR, adaptive, and CR research. Section Hardware shows the hardware of the testbed, and explains how it is put together. Section Software introduces the software architecture, the language and libraries used in the radar program. It also contains a link to the entire radar software available from Github. Section Radar Calibration shows a radar cross section (RCS) calibration of the system, together with a system characterization. Section Experimental Results shows some results from a set of experiments performed using the radar testbed. Both detection ranges for different targets, and the use of adaptivity on track update interval is introduced. Section Conclusion discusses the conclusions of the development of the testbed.

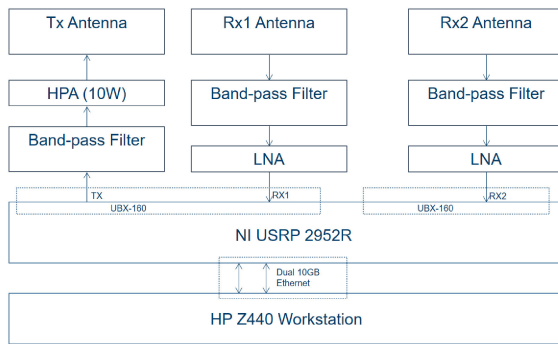
HARDWARE

The SDR system reported on in this article uses the USRP X310 from Ettus as its digital back end. The X310 was selected due to its high sampling rate, two input channels, and the quality of RF front-end cards. The principles of the development shown in this article apply to all USRPs using the USRP Hardware Driver (UHD) library, but with some limitations on sampling rate, number of channels, or RF front-end cards.

Figure 1 shows a block diagram of the system and images of the different components are given in Figure 2. A list of components is given in Table 1.

SOFTWARE-DEFINED RADIO

The USRP X310 is a device with an field-programmable gate array (FPGA) and two analog-to-digital converters (ADCs) and two digital-to-analog converters (DACs) devices. It supports sampling rates up to 200 MS/s with IQ modulation, and, hence, it can sample a bandwidth of up to 200 MHz at 14 bit in receive and 16 bit at transmit.

**Figure 1.**

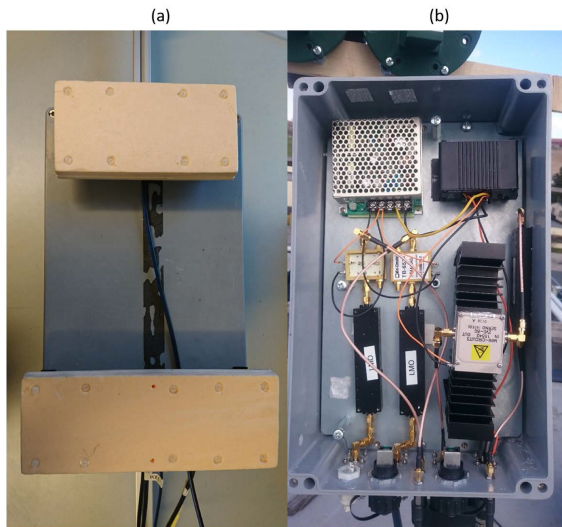
Block diagram of the CR SDR system.

RF front-end cards can be inserted, which enables the device to sample at RF frequencies supported by the different cards. Software-controlled amplification and local oscillator (LO) is available to fine tune the signal on both transmit and receive. The device used in this testbed has the UBX-160 cards inserted, which allows it to cover a 160-MHz bandwidth tuneable between 400 MHz and 6 GHz. The carrier frequency is software controlled, as most of the tunable parameters are. This allows for a highly flexible device for transmitting and receiving signals, and, hence, it should be well suited for radar applications.

The device is controlled by a host computer through a dual 10 Gb/s Ethernet connection, which allows for a high data transfer rate between host and the device. The connection itself supports transmit and receive at full bandwidth on two channels. The limiting factor is the host

Table 1.

SDR Parts List		
Type	Manufacturer	Model
Workstation	HP	Z440
2× 10 Gb PCI-express card	Intel	783345-01
2× 10 Gb SFP Ethernet cables	Generic	Na
USRP	Ettus	X310
2× USRP RF Frontend cards	Ettus	UBX-160 card
High power amplifier	MiniCircuits	ZVE-8G+
2× LNA	MiniCircuits	PSA4-5043+ on TB-653+
3× S-band bandpass filters	K&L	8FV40-3200/T200-O/O
2× S-band E-patch RX antennas	Generic	Na
1× S-band E-patch TX antenna	Generic	Na
SMA cables	Generic	Na
12-V dc Supply	Generic	Na
12 V–5 V dc–dc converter	Generic	Na
Weather proof box	Generic	Na

**Figure 2.**

Pictures of the different components of the CR system; where (a) shows the antenna setup with the transmit antenna on top and receive antennas at the bottom, and (b) shows the RF front-end electronics with filters and amplifiers.

computer, which has to process all the data. The host computer controls the device through the UHD library, which supports functions communicating with the FPGA on-board the device. All functions called on the FPGA are predefined from Ettus and are contained in the FPGA binary image supplied by Ettus. There is no need for FPGA programming, and, hence, the development can be performed in languages supported by the UHD library which is easier to develop and debug than developing FPGA functions.

RF EQUIPMENT

To achieve long detection ranges in a pulsed mode, a power amplifier is used. The MiniCircuits ZVE-8G+ [13] amplifier with an third-order intermodulation product (IP-3) [14] of approximately 10 W is used producing a 20-dB amplification over the maximum output power of the SDR, given in Table 2. The amplifier can be run in a

Table 2.

Ettus X310 with UBX-I60 Cards	
USRP parameters	
Number of channels	2
Frequency range	400 MHz–6 GHz
Maximum output power	100 mW
digital-to-analog converter (DAC) resolution	16 bit
analog-to-digital converter (ADC) resolution	14 bit
Noise figure	4–5 dB at 3.1 GHz
Maximum instantaneous bandwidth	160 MHz TX and RX
Maximum I/Q sample rate	200 MS/s
Memory size	1024 MB

saturated mode when operating at a maximum output power, and to avoid nonlinearities a cosine-based waveform must be used. Single tone pulses, linear chirps, and nonlinear chirps are acceptable waveforms. If the amplifier is operated below the compression point, nonconstant modulus waveforms such as orthogonal frequency division multiplexing [15] and noise pulses can be used.

The antenna subsystem consists of an E-patch antenna array with two elements spaced by approximately half a wavelength for angle-of-arrival (AOA) measurements. The manufactured transmitter antenna is an E-patch antenna mounted with a short distance from the receiver array to provide isolation between transmitter and receiver channels.

To calculate the angle from bore-sight for detected targets, we can use an AOA calculation exploiting the measured phase difference for the receiver antennas. If we define the element spacing to be L , the angle from bore sight to be θ , the phase difference measured between the antennas to be ϕ and the wavelength to be λ , the angle can be calculated as follows:

$$\phi = 2\pi \frac{L}{\lambda} \sin(\theta) \Rightarrow \theta = \sin^{-1}\left(\frac{\phi}{2\pi} \frac{\lambda}{L}\right). \quad (1)$$

From this, the unambiguous angle can be calculated setting $\phi = \pm\pi$:

$$\theta_{\max} = \sin^{-1}\left(\frac{\pi}{2\pi} \frac{\lambda}{L}\right) = \sin^{-1}\left(\frac{\lambda}{2L}\right). \quad (2)$$

The receiver antennas are mounted with a spacing of 5.5 cm, which is 0.58 times the wavelength at 3.15 GHz.

The calculated unambiguous angle is $\pm 59.7^\circ$. The simulated antenna diagram showed a 10 dB loss at $\approx \pm 60^\circ$, and, hence, large targets may fold in through antenna side lobes.

The RF equipment is mounted in a weather proof box, shown in Figure 2(b) and connected to the SDR with coaxial cables. The antennas and box are mounted on a simple antenna rig depicted in Figure 2(a).

PROCESSING COMPUTER HARDWARE

The radar's data processing system consists of three modules: a FPGA, a workstation, and a graphical processing unit (GPU). The FPGA was part of the USRP and provides low-level signal processing and control of the USRP hardware. Its most significant signal processing role is to perform digital down conversion (DDC) of the received waveform so that true baseband can be passed to the workstation for further processing. Future implementation of the matched filter on the FPGA could be beneficial to free up resources on the GPU for its other tasks.

A GTX 1080 Ti GPU is used for processing that can be efficiently parallelized. Currently, this is the implementation of the matched filter, the discrete Fourier transform (DFT) required for Doppler, and the constant false alarm rate (CFAR) processing. Stages, such as detection clustering could be implemented in a parallel fashion on GPU, for the time being. However, they are implemented on the Central Processing Unit (CPU), where the CPU is a multi-core Intel Xeon processor of model E5-1650 with 6 cores running at 3.6 GHz.

SOFTWARE

The USRP has several possibilities for software development. The most common approaches are as follows:

- (1) GNU Radio.
- (2) Matlab USRP support through Communication Toolbox.
- (3) LabView NI USRP module.
- (4) C++ UHD library.
- (5) Python wrapper for C++ UHD library.

GNU radio is a graphical radio programming interface that allows development with Python and C++. The initial investigation into using GNU Radio for radar applications showed that it was hard to get access to important features in the USRP. For example, the functionality required for phase synchronization between channels and synchronizing the signal between transmit and receive was not available.

Table 3.

List of Dependent Software Libraries	
Library	Description
UHD [17]	UHD library
Qt [18]	Qt is used for GUI programming
Qwt [19]	Plotting library for Qt
Boost [20]	Boost C++ libraries
CUDA [21]	GPU programming library

MATLAB has USRP support through the Communication Toolbox, although it contains a subset of the features available in the UHD library. These limitations lead to the conclusion that it was not the optimal development environment for a USRP-based radar.

National Instruments (NI) LabView supports control of the USRP through the USRP module. In addition, LabView can be used to write software to run on the USRP's FPGA. The possibility of writing both controller and FPGA code in the same environment is a positive feature. Unfortunately, the engineering flexibility required for CR applications seems difficult to achieve in the LabView environment.

The C++ UHD library allows the necessary control of the USRP to create a radar system with pulse-to-pulse coherency and coherency between multiple channels. Pulse-to-pulse coherency is essential for Doppler processing, and coherency between channels are essential for AOA processing. Arbitrary signal generation in C++ is a straightforward task, and signal processing in C++/CUDA is also possible. The UHD library does not allow for programming the FPGA directly. However, implementation of blocks on the FPGA through the Xilinx Vivado is possible as is the use of the RF-network-on-a-chip (RFNoC) to place functionality on the FPGA. There are methods in the UHD library to allow access to these newly developed FPGA blocks. The outcome of the above evaluation was to select the C++ UHD environment to develop the USRP software-defined radio (SDR). All programming comments in the remainder of this article should be taken as applying to this software development approach.

A Python wrapper of the C++ UHD library has been written, and it is shown to be slightly slower than a C++ implementation [16]. It does not currently support Ettus RFNoC, but it could be an interesting programming environment to consider for future development, since it allows the same control of the USRP as the C++ UHD library with the simplicity of programming in Python.

SOFTWARE DEVELOPMENT ENVIRONMENT DEPENDENCIES

As shown in the previous section, the C++ UHD library was selected as the programming environment for developing the USRP radar. The UHD library allows for control of the USRP functions, such as setting up the radio, transmitting a time series, receiving data coherently over several channels, and timing of the functions. The extra functionality needed in a radar system is the waveform generation which is done in C++ using the standard C++ library complex functions. Signal processing of the received data is done on the GPU using the CUDA programming language for parallel signal processing. The matrix operations needed in the Kalman filter tracker are supplied in the Boost library. Boost has many other features used in the program such as thread handling, timing, and file system control. Qt and Qwt were used for the graphical user interface (GUI) and data display such as plan position indicator (PPI) and range-Doppler displays. Table 3 shows the software dependencies.

WAVEFORM GENERATION

Waveform generation is done in a C++ class object, based on the complex number manipulation from the standard library. The only dependencies are the standard library functions for vector and complex numbers. Three waveforms are currently implemented: rectangular, linear frequency modulation (LFM), and nonlinear frequency modulation (NLFM) pulses. The waveform generation has run-time adjustable parameters such as pulse length, bandwidth, PRF, and number of pulses generated. The waveform is generated on a dwell-by-dwell basis.

RANGE-DOPPLER PROCESSING

The range-Doppler processing is based on a matched filter implemented with zero filling, fast Fourier transform (FFT), and inverse fast Fourier transform (IFFT) for fast convolution in the range domain, and a FFT in the slow time domain for Doppler processing. The processing is standard for a range-Doppler radar, but the parallel nature of the processing allowed it to be efficiently implemented on a GPU. The GPU implementation introduced a speedup factor of 50 compared to the implementation on the CPU, even when using all six cores of the CPU. The use of the GPU allows an inexpensive workstation machine to perform real-time processing for radar applications.

DETECTION PROCESSING

The detection processing implemented is the cell averaging constant false alarm rate (CA-CFAR), [22] which

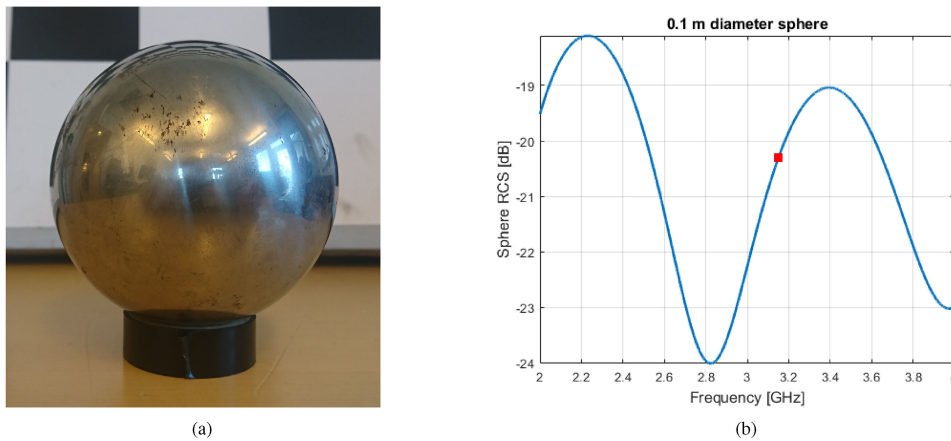


Figure 3.

(a) 100-mm-diameter solid stainless steel ball bearing sphere. (b) RCS calculation of a 100-mm-diameter sphere; blue line is the calculated RCS in dBsm as a function of frequency, and the red square is the 3.15 GHz point with value of -20.31 dBsm.

compares the cell under test (CUT) to an average of the signal power in a window of neighboring cells. If the CUT is above a given threshold, it is counted as a detection. The threshold is calculated to give a specified false alarm rate. This method allows for a constant false alarm rate detection against a varying noise floor [23]. The cell averaging can be implemented as a filter. The filter is applied through convolution that makes use of the FFT functions that can be optimized to run on the GPU. As such, the CA-CFAR can be implemented on the GPU to reduce runtime.

It should be noted that the CA-CFAR is likely to create more than one detection per target. This is common for the method. A clustering routine is used to identify individual targets from the multiple detection the CA-CFAR produces. At the current time, the clustering algorithm runs on the CPU.

ANGLE ESTIMATION PROCESSING

The angle estimation processing is based on the multiple signal classification (MUSIC) [24] algorithm. With two elements in the antenna array and an assumption of only one target in the range-Doppler resolution cell. This assumption is reasonable since the resolution cells are 0.75 m by 0.39 m/s when the bandwidth is 160 MHz and the dwell time is 12.3 ms at 3.15 -GHz carrier frequency, and it seems unlikely that two targets could exist in such a small cell. The method can be simplified to solve the trigonometric equation given in (1). The method is dependent on the measured phase difference of a target in the two antennas, the distance between the elements and the wavelength.

TRACKER

The tracker is a range-Doppler tracker, where the measurement and state space are the range and range-rate

(which is proportional to Doppler velocity). A linear motion model assumption is made. While such a simple motion model does not necessarily give the best track performance, it is a simple implementation and has proven successful in the research conducted so far. Track initialization is currently implemented by the radar operator via a manual mouse click on the detection in the GUI. There is a plan to implement a multiple hypothesis tracker with converted range, bearing, and range rate measurement [25], but this is outside the scope of this article.

RUN-TIME CONSIDERATIONS FOR SIGNAL PROCESSING

The new generation of gaming GPU's contains several thousands of cores combined with large memory, and can be regarded as extremely powerful for calculations. The price range for gaming GPUs is much lower than NVIDIA's Tesla and Quadro cards. As demonstrated here, a gaming GPU provide single point precision calculations at rates fast enough to conduct real-time radar signal processing.

Most, if not all stages of the signal processing can be implemented in parallel. Parallel processing efficiently implemented on GPU is much faster than if implemented at CPU, and can even compete with the processing on FPGA. The fast compilation and easy debugging allows it to be a great platform for both development of new algorithms, but also achieve real-time requirements on the signal processing.

SOFTWARE AVAILABILITY

The software reported on in this article for controlling the USRP, performing the radar signal processing, and creating the GUI can be downloaded from Github (<https://github.com>)

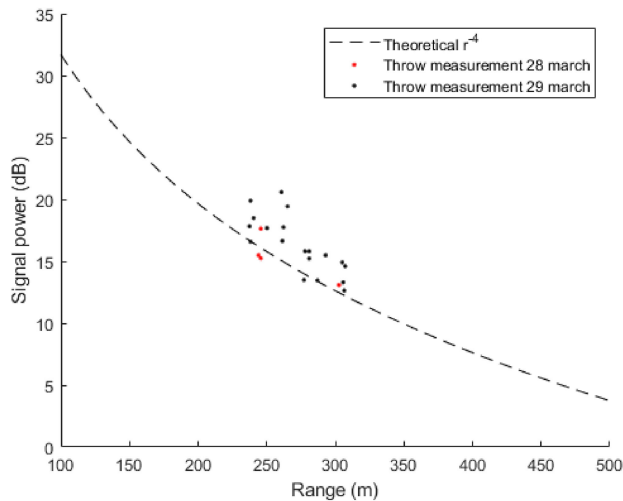


Figure 4.

Measurements of thrown sphere at different ranges, shown as dots, compared to a $\frac{1}{r^4}$ curve from the radar equation. The $\frac{1}{r^4}$ curve is fitted to the measurements in an attempt to perform an absolute calibration of the radar system.

github.com/jonasmc83/USRP_Software_defined_radar) at https://github.com/jonasmc83/USRP_Software_defined_radar.

RADAR CALIBRATION

Radar calibration is critical for evaluating the performance of a new system. This section reports on the two calibration tests made of the USRP-based radar to evaluate the system performance. The first was RCS calibration. The second, receive channel phase characterization, which is required for moving target indication and angular measurement performance prediction.

RCS CALIBRATION

The RCS calibration was performed against a small metal sphere in an open field environment. The sphere was a

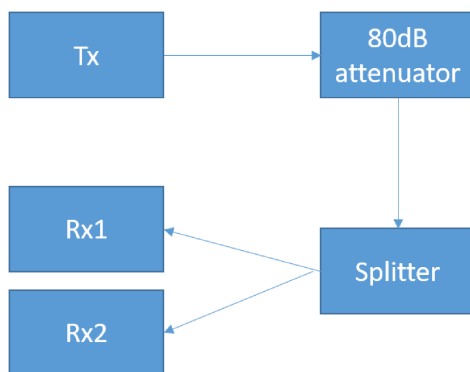


Figure 5.

Diagram displaying the characterization setup, with a 80 dB attenuator and a power splitter to emulate a point target.

100-mm diameter solid stainless steel ball bearing, shown in Figure 3(a). A RCS calculation of a 100-mm diameter sphere is shown in Figure 3(b), where the RCS for the carrier frequency 3.15 GHz used in this article is -20 dBsm. The small size of the sphere compared to the wavelength results in the scattering is in the resonance region. Due to the uncontrolled nature of the test environment, it was not possible to perform background subtraction that is common in RCS measurements to separate the calibration sphere from the clutter. Instead, the robustness of the ball bearing was exploited. It was thrown by a volunteer giving it a Doppler shift that allowed its response to be separated from the ground clutter.

A set of eight range positions were selected, and the volunteer threw the sphere 1–3 times at each range position. This gives a set of signal power measurements for the sphere as a function of range. These measurements are depicted in Figure 4, where each dot is a measurement. The scene used made it difficult to perform measurements closer than 230 m, and the low RCS of the sphere made it hard to detect the sphere past 300 m. The signal power variation is approximately 2–3 dB. We can say that this is an absolute calibration using the knowledge of the sphere RCS calculated as -20 dBsm, but due to variation in the measurements it is accurate within 2–3 dB. The line shows a $\frac{1}{r^4}$ curve fitted for the dataset of thrown spheres, and using the knowledge of the RCS to be -20 dBsm, the curve can be used for estimating the RCS of other targets.

CHANNEL-PHASE CHARACTERIZATION

The phase noise for each channel is a limiting factor in the detection of slow moving targets close to the clutter ridge. The noise of the phase difference between channels is a limiting factor for angular estimation. A loop-back configuration to characterize the phase noise of each channel, and the phase difference between each channel, was set up. The loop-back configuration was achieved by disconnecting the antennas from the RF front-end, attaching two attenuators of 40 dB of total 80 dB attenuation and a power splitter to have two attenuated signal paths and then connecting the two signal paths to the receivers. This setup allows the transmit signal to be attenuated and fed into the two receiver stages, as shown in Figure 5. The cable delay is less than one fast time sample, so this configuration produces an ideal target response at zero range. A measurement of the absolute phase noise of each channel were performed together with the phase difference between channels based on this response.

A dwell consists of a number of pulses, 4096 in this case. Each sample in Figure 6(a) shows two times the standard deviation calculated over these 4096 pulses, and, hence, the dashed curves shows an estimate of 2σ . The

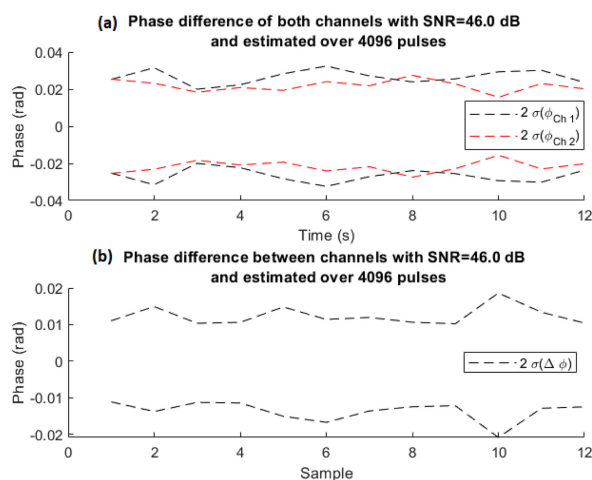


Figure 6.

(a) Shows the phase noise for each channel. (b) Shows the noise of the phase difference between channels.

phase noise is similar for both channels, and this is expected since the channels are equal in the SDR and in the RF chain. Figure 6(b) shows the same parameters as for (a), when looking at the phase difference of the two channels. The noise shown here directly affects the AOA measurement and will introduce uncertainty in the angular measurements. An error of 0.02 radians as given in Figure 6 corresponds to approximately 0.3° error in the angular estimate.

The conclusion we can draw from this characterization is that the phase from pulse-to-pulse shows only a small variation due to phase noise. This implies coherency from pulse-to-pulse, and, hence, we can perform Doppler processing. The phase difference between channels is also constant, with a small variation due to phase noise. The implication from this is that we can do AOA for angular measurements, since the angular measurement is dependent on coherency between channels. The SDR radar system can therefore do Doppler and AOA processing.

EXPERIMENTAL RESULTS

DETECTION RANGE

The system contains a 10-W power amplifier and earlier predictions of the detection performance of this system [11] showed a substantial detection range, greater than 5 km for airliner size targets. The same prediction shows detection ranges of 3 km and lower for smaller targets such as a light aircraft. Figure 7(a) shows a detection of a small aircraft at 2.8 km distance, and Figure 7(b) shows a detection of a large airliner aircraft detected at 5.5 km. This fits well with the previous predictions of the detection capabilities of this radar system.

TRACKING SMALL TARGETS

The USRP radar is able to track small targets at short range. Using a short pulse allows it to monitor a surveillance volume close to the radar. An experiment with a small unmanned aerial vehicle (UAV)—a DJI Mavic—was performed. The pulse length was set to $0.8 \mu\text{s}$, which gives minimum range of 120 m. The PRF was set to 100 kHz, which gives an unambiguous range of 1.5 km. The experiment is shown in Figure 8(a), which shows a picture of the UAV, a satellite photo of the test range, and the position of the radar and track of the UAV. The UAV path in the overlay is the actual track of the UAV and it illustrates the angular accuracy of the AOA estimation. Figure 8(b) shows plots of the range, Doppler velocity, and signal-to-noise ratio (SNR) of the UAV as it flies first outward, and then back toward the radar. The black dots are the radar detections and green squares are the tracker output. The tracker keeps track until 350 m. It loses track when the target stops as it prepares to change the direction. The lost track happens because the track merges into the stationary clutter and is not detected for several dwells. The operator starts a new track as soon as the target becomes separate from the clutter and the UAV is successfully tracked as it approaches the radar. Using the rule of thumb from the radar equation that a doubling in range corresponds to a 12 dB drop in SNR, a detection range of 600 m can be extrapolated.

ADAPTIVE UPDATE INTERVAL METHOD USING THE FULLY ADAPTIVE RADAR (FAR) FRAMEWORK

The article by Christiansen [26] shows a method of calculating a track update based on a cost function to optimize some measure of resource usage, weighted against the global mean square error of the tracker [1]. The article shows the method applied on a simulated case. The method has been implemented on the USRP radar and the required interval until the next track update dynamically adapted based on the predicted track error. The method was used in the case of the UAV track shown in Figures 8 (a) and (b). Figure 9(a) shows the tracker parameters of range uncertainty, velocity uncertainty, and selected update interval, respectively. The tracker uncertainty is relatively constant since the optimization method decreases the update interval as the target moves further from the radar. The measurement uncertainty increases when the target is farther from the radar due to reduced SNR. The optimizer selects shorter update intervals since the tracker has lower state uncertainty with shorter update intervals and, hence, the global track error remains quite constant despite the SNR drop. Figure 9(b) shows the

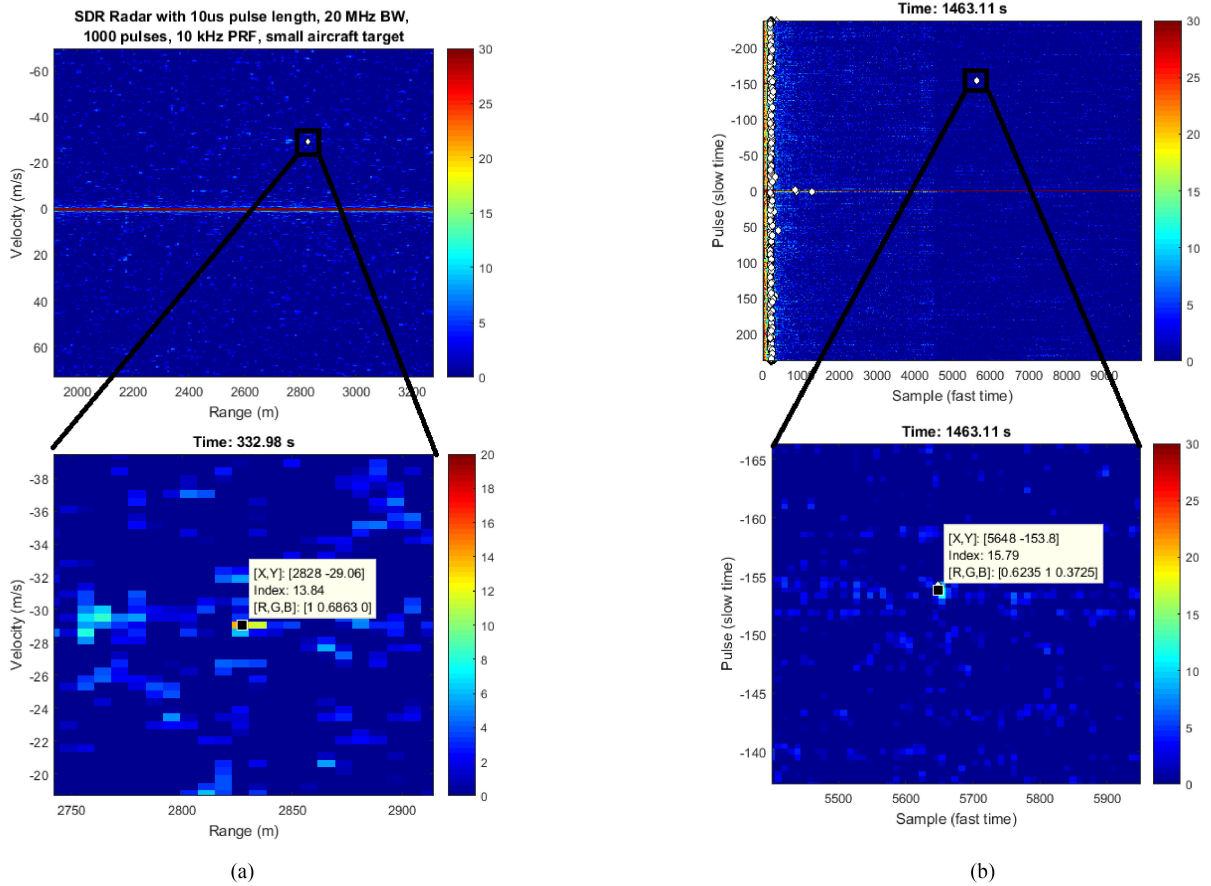


Figure 7.

Demonstration of detection ranges of the USRP-based SDR radar system. (a) Small aircraft detected at 2.8 km distance with USRP-based SDR radar system. (b) Airliner type target (Boeing 737 class) detected at 5.5 km distance with USRP-based SDR radar system.

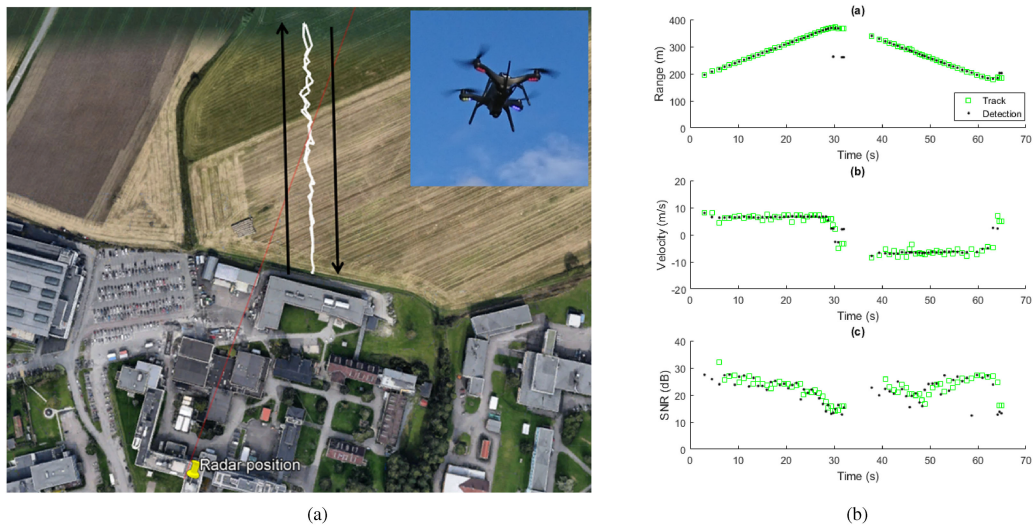


Figure 8.

Experiment showing tracking of small UAV target out to 350 m and back. (a) Experimental setup of UAV flying outward, turning, and flying inward. (b) Range, range-rate (Doppler velocity) and SNR plot, respectively, from top to bottom.

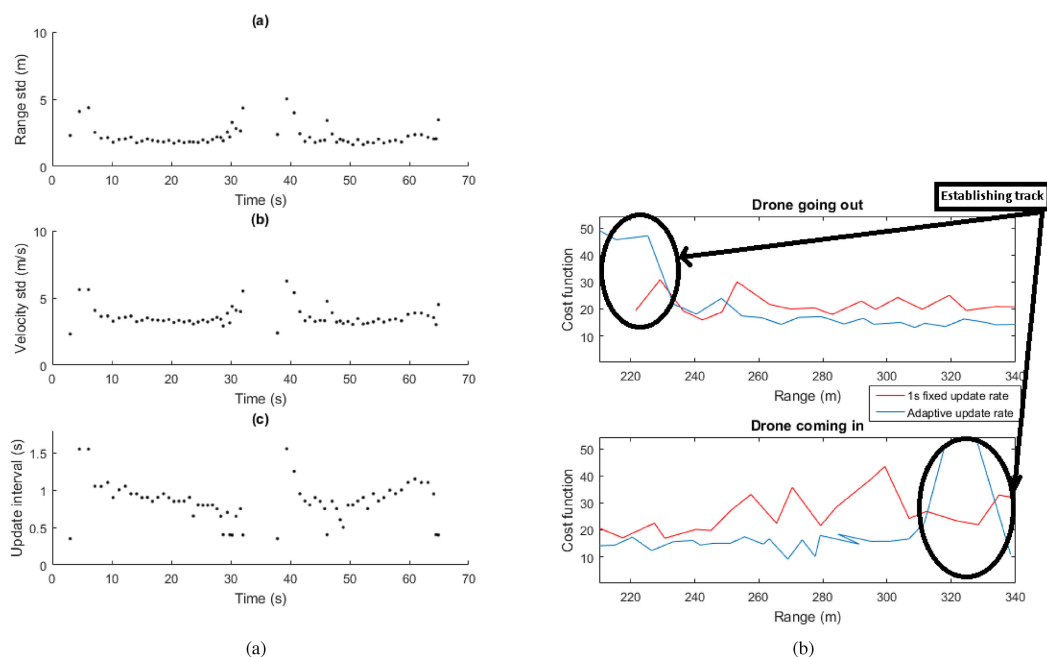


Figure 9.

Experiment showing tracking of small UAV target, with showing track parameters and cost function values. (a) Tracker parameters, with plots of range uncertainty, velocity uncertainty, and the selected next update interval, respectively. (b) Cost function values based on true posterior information for two experiments, where one experiment with fixed update interval of 1 s is shown in red, and adaptive update interval is shown in blue.

result of a cost function evaluation using the actual update interval selected for a 1 s fixed update interval experiment, and an experiment with adaptive update interval. The cost function values based on the true posterior information is smaller when using the adaptive update interval, a method which attempts to minimize this cost function based on predicted posterior information.

The experiment shown here is a work in progress. Research is already underway to expand the parameters adapted to include PRF and number of pulses in a dwell. The available parameters to optimize over in this SDR testbed are currently carrier frequency, bandwidth, pulse length, PRF, number of pulses in dwell.

CONCLUSION

The testbed shown in this article is based on the Universal Software Radio Peripheral (USRP) X-310, which is a member of a line of software-defined radio (SDR)s available from Ettus. The USRPs are notable for their relatively low price and high levels of system configuration. The RF hardware used for the front-end is all connectorized, commercial off the shelf. As such, it too is available at low cost and with minimal lead times. The testbed can therefore be kept at low cost and, hence, affordable to academia and smaller research projects. The software is based on the UHD driver library in C++ and CUDA code for signal

processing. The program can therefore run on a workstation with a NVIDIA GPU installed. The source code for the entire radar project can be found at https://github.com/jonasmc83/USRP_Software_defined_radar.

The system calibration section introduced a method for absolute RCS calibration and channel characterization. The single channel phase characterization showed that the system is coherent from pulse-to-pulse and, hence, can perform Doppler processing for pulse integration and clutter suppression. The phase difference between channels characterization showed that the system is coherent between channels, and, hence, can perform AOA processing for angular estimation.

Earlier work has shown that the system's detection ranges for large airliners as 5.5 km and a small aircraft as 2.8 km. Experiments have shown a detection range of a small UAV as more than 350 m. An experiment was conducted tracking a small UAV with a method of adaptive track update interval implemented. This illustrated that the system can autonomously adapt its processing in real time. Parameters that can be adapted at runtime are carrier frequency, bandwidth, pulse length, PRF, and number of pulses in dwell.

The field of cognitive and adaptive radar research is dependent on radar testbeds to perform experiments in real time. In these systems, radar parameters selection is dependent on the scene the radar observes. In order to validate algorithms it is therefore necessary to perform real-

world experiments. Such a testbed should be able to adapt its parameters on-the-fly, such as PRF, pulse length, bandwidth, waveform type, number of pulses in burst and carrier frequency, in order to adapt its performance based on the radar perception. This USRP-based testbed presented here is a demonstration of how such a testbed can be produced on a low budget.

REFERENCES

- [1] K. L. Bell, C. J. Baker, G. E. Smith, J. T. Johnson, and M. Rangaswamy, "Cognitive radar framework for target detection and tracking," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 8, pp. 1427–1439, Dec. 2015.
- [2] A. E. Mitchell, G. E. Smith, K. L. Bell, A. J. Duly, and M. Rangaswamy, "Fully adaptive radar for variable resolution imaging," in *Proc. Int. Conf. Radar Syst.*, Oct. 2017, vol. 1, no. 1, pp. 1–6.
- [3] G. E. Smith *et al.*, "Experiments with cognitive radar," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 31, no. 12, pp. 34–46, Dec. 2016.
- [4] A. E. Mitchell, G. Smith, K. L. Bell, A. Duly, and M. Rangaswamy, "Hierarchical fully adaptive radar," *IET Radar, Sonar/Navigation*, vol. 12, no. 12, pp. 1371–1379, Dec. 2018. [Online]. Available: <http://digital-library.theiet.org/content/journals/10.1049/iet-rsn.2018.5339>
- [5] A. E. Mitchell, G. E. Smith, K. L. Bell, A. J. Duly, and M. Rangaswamy, "Cost function design for the fully adaptive radar framework," *IET Radar, Sonar/Navigation*, vol. 12, no. 12, pp. 1380–1389, 2018.
- [6] "Ettus USRP site," [Online]. Available: <https://www.ettus.com/>, Accessed: Mar. 27, 2019.
- [7] A. Prabaswara, A. Munir, and A. B. Suksmono, "GNU Radio based software-defined FMCW radar for weather surveillance application," in *Proc. 6th Int. Conf. Telecommun. Syst., Services, Appl.*, 2011, pp. 227–230.
- [8] H. O. Moreno *et al.*, "Potentialities of USRP-based software defined radar systems," *Prog. Electromagn. Res. B*, vol. 53, pp. 417–435, 2014.
- [9] B. Bleszynski, "A simple radar based on USRP software defined radio," in *Proc. Signal Process. Symp.*, 2017, no. 1, pp. 1–4.
- [10] B. H. Kirk, J. W. Owen, R. M. Narayanan, S. D. Blunt, A. F. Martone, and K. D. Sherbondy, "Cognitive software defined radar: waveform design for clutter and interference suppression," in *Proc. SPIE*, vol. 10188, 2017, Art. no. 1018818.
- [11] J. M. Christiansen, G. E. Smith, and K. E. Olsen, "USRP based cognitive radar testbed," in *Proc. IEEE Radar Conf., RadarConf.*, 2017, pp. 1115–1118.
- [12] J. M. Fuster, *Cortex And Mind Unifying Cognition*. London, U.K.: Oxford Univ. Press, 2003.
- [13] "MiniCircuits ZVE-8G+ HPA Datasheet," 2016. [Online]. Available: <https://www.minicircuits.com/pdfs/ZVE-8G+.pdf>
- [14] "MiniCircuits Modem Amplifier Terms Defined," 2016. [Online]. Available: <https://www.minicircuits.com/pages/pdfs/amp3-4.pdf>
- [15] B. Dixon, R. Pollard, and S. Iexekiel, "A discussion of the effects of amplifier back-off on OFDM," Institute of Electrical and Electronics Engineers (IEEE), Jan. 2003, pp. 14–19.
- [16] "Python UHD wrapper," 2019. [Online]. Available: https://kb.ettus.com/UHD_Python_API
- [17] "USRP Hardware driver website," 2016. [Online]. Available: <https://www.ettus.com/sdr-software/detail/usrp-hardware-driver>
- [18] "Qt website," 2016. [Online]. Available: <http://www.qt.io>
- [19] "Qwt website," 2016. [Online]. Available: <http://qwt.sourceforge.net>
- [20] "Boost C++ libraries website," 2016. [Online]. Available: <http://www.boost.org>
- [21] "NVIDIA CUDA Website," [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html, Accessed: Jul. 26, 2017.
- [22] H. Rohling, "Ordered statistic CFAR technique—An overview," in *Proc. 12th Int. Radar Symp.*, 2011, pp. 631–638.
- [23] M. A. Richards, J. A. Scheer, and W. A. Holm, *Principles of Modern Radar, Volume I - Basic Principles*. Rijeka, Croatia: SciTech, 2010.
- [24] R. Schmidt and X. W. Af, "Multiple Emitter Location and Signal Parameter," *IEEE Trans. Antennas Propag.*, vol. AP-34, no. 3, pp. 276–280, Mar. 1986. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1143830>
- [25] S. Bordonaro, P. Willett, and Y. Bar-Shalom, "Consistent linear tracker with converted range, bearing, and range rate measurements," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 6, pp. 3135–3149, Dec. 2017.
- [26] J. M. Christiansen, K. E. Olsen, and G. E. Smith, "Fully adaptive radar for track update-interval control," in *Proc. IEEE Radar Conf.*, Apr. 2018, pp. 0400–0404.