Universal Journal of Electrical and Electronic Engineering 7(1): 19-27, 2020 http://www.hrpub.org
DOI: 10.13189/ujeee.2020.070102

MATLAB-Based GPU Acceleration for Multiple-Input Multiple-Output Radar Beamforming Algorithm

Mostafa Hefnawi*, Gillian Rideout

Department of Electrical and Computer Engineering, Royal Military College of Canada, Canada

Received November 19, 2019; Revised December 30, 2019; Accepted January 13, 2020

Abstract Multi-input and multi-output (MIMO) radar systems possess numerous advantages, such as scanning an entire region much faster than a phased array, enhancing the spatial resolution, mitigating interference and multipath fading, and improving the probability of detection of targets. MIMO radar systems use adaptive beamforming techniques such as the minimum variance distortionless response (MVDR) to obtain the best possible estimation of the direction of arrival (DOA) of the targets. The MVDR algorithm has been thoroughly investigated for traditional phased array radars. For MIMO radar systems, however, it requires significant signal processing, which can introduce substantial latency, especially in the case of large MIMO systems where few hundred antennas are used; this can be solved by using a Graphics Processor Unit (GPU), which contains thousands of cores and can execute many operations in parallel. This paper presents a MATLAB-based approach for GPU parallelization of the minimum variance distortionless (MVDR) beamforming algorithm in a MIMO radar system. Two MIMO radar systems are considered. The first one is a simulated MIMO radar which is used for automotive adaptive cruise control (ACC), and the second one is an experimental monostatic MIMO radar that is based on a vector network analyzer (VNA). It is shown that the GPU achieved a speedup of up to 7 times while successfully detecting all targets.

Keywords MIMO Radar, GPU, MVDR

1. Introduction

The application of multi-input and multi-output (MIMO) techniques to radar systems has received considerable attention recently [1-13]. Unlike traditional phased-array radars in which a steered beam is used at the transmitter to scan a sector, MIMO radar transmits different waveforms from each omnidirectional antenna element simultaneously, allowing for a sector scan rate that is several times faster than steered beam radars. A band of matched filters can extract the waveforms bearing with the target information at the receiver end. Generally, MIMO radar is implemented using either widely separated antennas [7-8] or colocated antennas [9-13]. In a MIMO radar with widely separated antennas, each transmit-receive pair sees a different aspect of the target, which improves the detection probabilities of targets. This configuration relies on space diversity to improve the detection performance but does not provide the flexibility of transmitting the desired beampattern. On the other hand, in a MIMO radar with collocated antennas, the antenna elements are closely spaced so that each transmit-receive pair sees the same aspect of the target. While this configuration does not provide spatial diversity, it relies on waveform diversity to achieve a better spatial resolution by combining all the transmitting paths in a virtual array with an extended aperture. This configuration also has many other advantages compared to the spatial diversity configuration, such as excellent clutter interference rejection capability, improved parameter identifiability, and enhanced flexibility for transmitting beampattern design. In this paper, therefore, only the colocated antenna configuration will be examined. Two MIMO radar systems are considered. The first one is a simulated MIMO radar, which is used for automotive adaptive cruise control (ACC) [14] and the second one is an experimental monostatic MIMO radar that is based on a vector network analyzer (VNA) [15]. A generic MIMO radar design is illustrated in Figure 1, where there are M independent orthogonal transmitters and N receivers. Each receiver would have M matched filters that correspond to the M distinct transmitted waveforms. Having M independent transmitters and N receivers produces $M \times N$ paths for the return from the k^{th} target, which in turn creates a virtual array with a narrower beam and a larger effective aperture in comparison to a phased array.

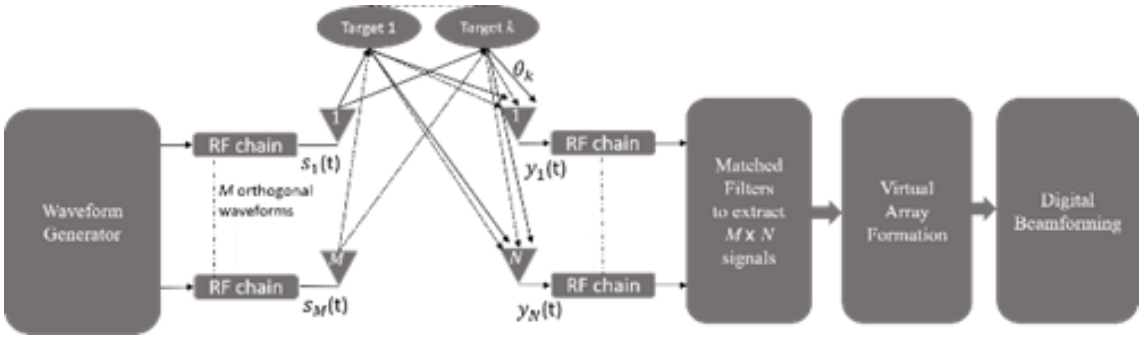


Figure 1. Generic MIMO radar [15].

The generic received signal for the system illustrated in Figure 1 is given by [2]:

$$y_n[n] = \sum_{k=1}^{K} \alpha(\theta_k) \sum_{m=1}^{M} e^{-j\omega_c \tau_{mn}(\theta_k)} s_m[n] + w_n[n], \quad (1)$$

where α is the complex amplitude of the k^{th} return signal from a target located at an angle θ_k, s_m is the baseband samples of the m^{th} transmitted signal, $\tau_{mn}(\theta_k)$ is the total phase delay between the *m*th transmitting element, the k^{th} target, and the n^{th} receiver, ω_c is the carrier frequency, [n] is the time index, and w_n is an additive white Gaussian noise with zero-mean and a covariance matrix $R_w = \sigma^2 I_M$.

It is essential to note in (1) that a MIMO radar takes the sum of returns from all targets, whereas a phased array radar will only have a return from its directed beam. Extracting the independent transmit and receive paths is the most critical step in designing a MIMO radar as these signals are used to form the virtual array, will define the two-way antenna pattern, and contains the necessary information to conduct beamforming on reception. Time-division multiplexing (TDM) MIMO was selected as the orthogonal waveform to be implemented in this paper for its simplicity to form a virtual array and to reduce the number of matched filters required on reception. The received signals from individual elements can be mapped directly to a virtual element as follows [2]:

$$Virtual\ Array(x) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(x - (x_m + x_n)), \quad (2)$$

where M and N are the numbers of transmitting and receiving antennas, respectively, x_m and x_n are the physical locations of the m^{th} transmit element and the n^{th} receive element.

Using a virtual array allows for the system to increase the angular resolution and accuracy beyond the physical number of elements that are present in the transmit and receive arrays [2]. It also allows for a sparsely filled array to maintain the same resolution of a full array, without the added cost of more hardware. On the other hand, MIMO radar systems use adaptive beamforming techniques such as the minimum variance distortionless response (MVDR) to obtain the best possible estimation of the direction of arrival (DOA) of the targets [14-14]. The MVDR technique attempts to minimize the noise and interference by creating nulls toward their directions and maintaining a fixed gain in the look direction. The MVDR spectrum given in [22] can be extended to MIMO radar as follows:

$$S = \frac{1}{\boldsymbol{s}_v^H(\theta_t) \boldsymbol{R}_{xx}{}^{-1} \boldsymbol{s}_v(\theta_t)} \quad (3)$$

where $\boldsymbol{s_v}$ is the steering vector of the virtual array and $\boldsymbol{R}_{xx}$ is the covariance matrix at the output of the virtual array.

The MVDR algorithm has been thoroughly investigated for traditional phased array radars. For MIMO radar systems, however, it requires matrix inversions for each pulse that is transmitted by the MIMO radar. In a small system, the time to complete these calculations on a Central Processing Unit (CPU) may be acceptable. However, when a system grows to be quite large such as in the case of large MIMO systems where few hundred antennas are used, the required calculation time to carry out the MVDR beamforming algorithm would be enormous and unacceptable. An alternative to the CPU is the Graphics Processor Units (GPU) that contain thousands of cores and can execute software programs in parallel [23]. It is lucky then that many algorithms used within MIMO radar systems are highly parallel and computationally intensive, which makes them ideal candidates for a GPU implementation. A survey paper completed in 2017 showed that there had been much success in using GPUs in traditional single input single output (SISO) radar systems [21]; significant computational speedups were found to have occurred. As well, several papers were found which had success integrating GPU and MIMO in communication scenarios that provided optimization or greater accuracy [22-26]. However, there was little research to be found concerning incorporating MIMO radar and GPUs. This paper explores the use of GPU technology in a traditional MIMO radar. More specifically, the paper investigates the speedup achieved when using a MATLAB-based GPU acceleration of the MVDR algorithm in a MIMO radar.

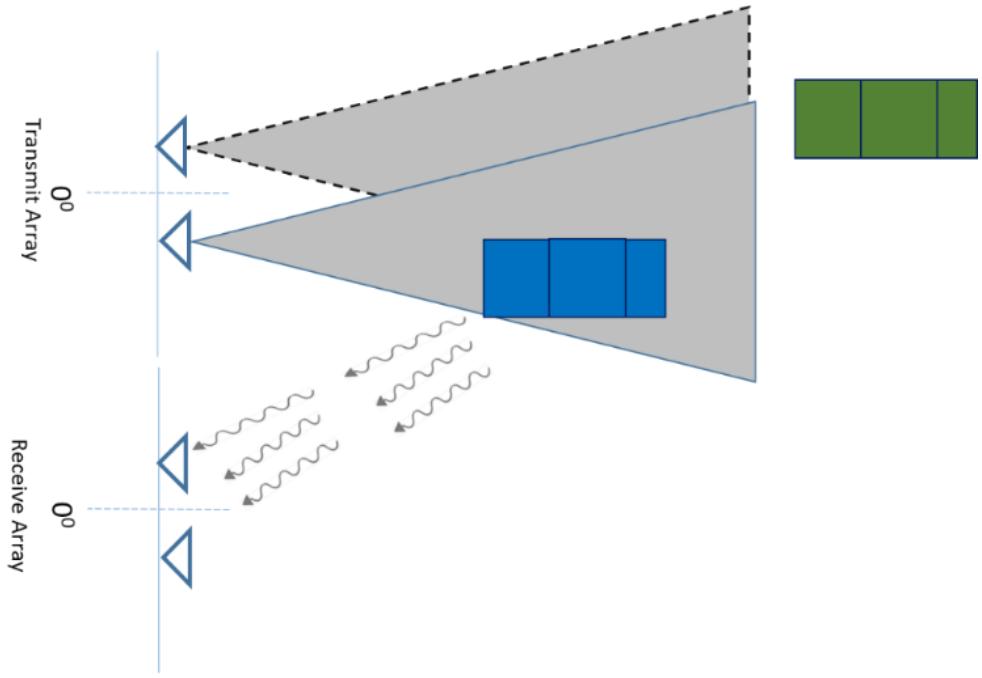


Figure 2. Automotive MIMO Radar for ACC

2. GPU Acceleration of an Automotive MIMO Radar

Figure 2 shows the first MIMO radar system considered in this paper. It consists of a simulated automotive MIMO radar that is used for adaptive cruise control (ACC) onboard an autonomous vehicle that tracks two moving vehicle targets [27]. The simulation of this system is performed using the Phased Array Toolbox (PAT) of MATLAB [28], and the GPU acceleration of the MVDR algorithm is performed using the Parallel Computing Toolbox (PCT) [29].

2.1. Waveform Generation

In this simulation, a frequency modulated continuous (FMCW) Waveform is created using the phased.FMCWWaveform object and then transmitted with specified peak power using the phased.Transmitter object. Figure 3 shows how the FMCW waveform can be simulated, and Table 1 summarizes its parameters that are based on vehicle ACC [14]. The sweep bandwidth is determined according to the range resolution of 1 m. The sweep time for the FMCW wave was chosen to be 5.5 times the round-trip time to account for the time needed for the signal to travel the maximum unambiguous range. The sample rate of the FMCW signal is the same as the sweep bandwidth.

```
waveform = phased.FMCWWaveform('SweepTime',tm, ...
'SweepBandwidth',bw,'SampleRate',fs);
  sig = waveform();
  transmitter= phased.Transmitter('PeakPower',...
  0.001,'Gain',36);
  txsig = transmitter(sig);
```

Figure 3. Waveform generation

Table 1. Transmitted FMCW properties

Properties	Values
Maximum Unambiguous Range	200 m
Sample Rate	150 MHz
Sweep Time	7.3 µs
Sweep Bandwidth	150 MHz

2.2. Transmit and Receive Arrays Simulation

Figure 4 shows how the radar transmitter and receiver are simulated. The transmitter and receiver use uniform linear arrays (ULAs) with identical sensor elements that are created by the phased.ULA object.

```
dt = lambda/2;% half-wavelength ant. spacing at TX
dr = lambda/2; ;% half-wavelength ant. spacing at RX
txarray = phased.ULA(Nt,dt);%Nt:nbr of elements at TX
rxarray = phased.ULA(Nr,dr);%Nr:nbr of elements at RX
txradiator=phased.Radiator('Sensor',txarray,'OperatingFrequency'...
,fc,'PropagationSpeed',c, 'WeightsInputPort',true);
rxcollector=phased.Collector('Sensor',rxarray, ...
'OperatingFrequency',fc,'PropagationSpeed',c);
receiver=phased.ReceiverPreamp('Gain',40, ...
'NoiseFigure',4.5,...'SampleRate',fs);
```

Figure 4. Transmit and receive arrays Simulation

After the construction of the ULAs, a radiator and a collector are created using the phased.Radiator and phased.Collector System objects. The radiator converts the waveform into radiated wavefields transmitted from antenna arrays and the collector converts incident wave fields into signals that are preamplified by the phased.ReceiverPreamp System object. The parameters of this simulation are shown in Table 2.

Table 2. Antenna array parameters

Sensor parameters	Value
Operating Frequency (GHz)	77
Transmit Array Spacing dt	0.5λ
Receive Array spacing dr	0.5λ
Transmitter Peak Power (W)	0.001
Transmitter Gain (dB)	36
Receiver Gain (dB)	40
Receiver Sample Rate (MHz)	75
Receiver Noise Figure (dB)	4.5

2.3. Target Simulation

The radar targets consist of two moving cars that are created using the system object phased.RadarTarget which models how the radar signal will be reflected from the targets. The measure of a target's ability to reflect radar signals in the direction of the radar receiver is called the radar target cross-section (RCS). The motion of a platform in space is simulated using phased.Platform object. This platform can be a target or a radar transmitter/receiver. The model created assumes that the platforms undergo their motion at a constant velocity during the simulation where positions and velocities are defined within the global coordinate system. The position and motion of the autonomous vehicle and the two cars are presented in Table 3, whereas the code used to define the position and motion of the target is illustrated in Figure 5.

Table 3. Position and motion of the autonomous vehicle and the two cars

Parameters	Car 1	Car 2	Ego vehicle
Velocity in x direction (km/h)	-80	96	100
Azimuth	-30 degrees	10 degrees	0
RCS (m^2)	100	200	100
Starting Position (m)	x direction: 34.64	x direction: 49.24	x direction: 0
	y direction: -20	y direction: 8.68m	0
	z direction: 0.5	z direction: 0.5	0.5
Distance between car and autonomous vehicle sensor	40 m	50 m	0

```
radarspeed = 100*1000/3600;% vehicle speed 100 km/h
radarmotion=phased.Platform('InitialPosition', ...
[0;0;0.5],'velocity', [radar-speed;0;0]);
cardist = [40 50]; % Dist. radar to cars (m)
carspeed = [-80 96] * 1000/3600; % km/h -> m/s
caraz = [-30 10];
carrcs = [100 200];
carpos = [cardist.*cosd(caraz); ...
cardist.*sind(caraz);0.5 0.5];
cars = phased.RadarTarget('MeanRCS',carrcs, ...
'PropagationSpeed',c,'OperatingFrequency',fc);
carmotion = phased.Platform('InitialPosition', ...
carpos,'velocity',[carspeed;0 0;0 0]);
```

Figure 5. Target motion code

2.4. Virtual Array Processing

The raw data that is received by the physical receive array must be processed to associate the returns with the correct transmit element in order to form the virtual array data. Since this simulation uses TDM-MIMO FMCW waveform with two transmit elements, the virtual array data can be processed by doing two sweeps of the data. Given that there are 64 radar sweeps completed, 32 of these sweeps will correspond to transmit element zero, and 32 belong to transmit element one. Because the transmitters are toggled midway through the simulation, it is deduced that every other data cube will belong to the same transmitter, with the odd elements corresponding to element zero and the even corresponding to element One.

These two arrays are then catenated together to create the virtual array data. The code used to create the virtual array is shown in Figure 6.

```
% the data from the two transmit antenna
% elements extracted from two consecutive
Nvsweep = Nsweep/2;
xr1 = xr(:,:,1:2:end);
xr2 = xr(:,:,2:2:end);
% These two arrays are then catenated together to
% create the virtual array data as follows:
xrv = cat(2,xr1,xr2);
```

Figure 6. Virtual array creation

2.5. GPU Acceleration of the MVDR Spectrum

CUDA is a programming language that exploits the parallel processing capabilities of GPUs and is used in heterogeneous computing [30]. NVIDIA has developed CUDA extensions that can be used with other languages, such as FORTRAN and Python [31], which allows the coder to use a language of their choice and decreases the learning curve associated with parallel programming. MATLAB users can also compile their complete algorithms to run on any modern NVIDIA GPUs using the Parallel Computing Toolbox (PCT) without having to use CUDA programming [31]. In this paper, the PCT of MATLAB is used to achieve a GPU acceleration of the MVDR algorithm in a MIMO radar system. The MVDR spectrum was run on the GPU using the gpuArray function. When a gpuArray argument is supplied to any GPU-enabled function, the function runs automatically on the GPU. Figure 7 shows the code used for the GPU-based MVDR spectrum. The GPU used to carry out the parallel algorithm is a NVIDIA GeForce GTX TITAN which is based on the Kepler Architecture [32]. The specifications for this GPU can be found in Table 4 [33]. On the other hand, the CPU simulation is performed using a high-performance Linux computer with Intel Xeon CPU E5-2660 operating at 2.60 GHz and 128 GB of RAM and running a 64-bit operating system.

```
% copy the virtual array to the GPU
xrv_parallel = gpuArray(xrv);
% copy the virtual steering vector to the GPU
sv_parallel = gpuArray(sv);
% Compute the covariance matrix on the GPU
Rxx_parallel = transpose(xrv_parallel)* xrv_parallel;
% Compute the MVDR spectrum on the GPU.
S_parallel=transpose(sv_parallel)*(Rxx_parallel\sv_parallel);
```

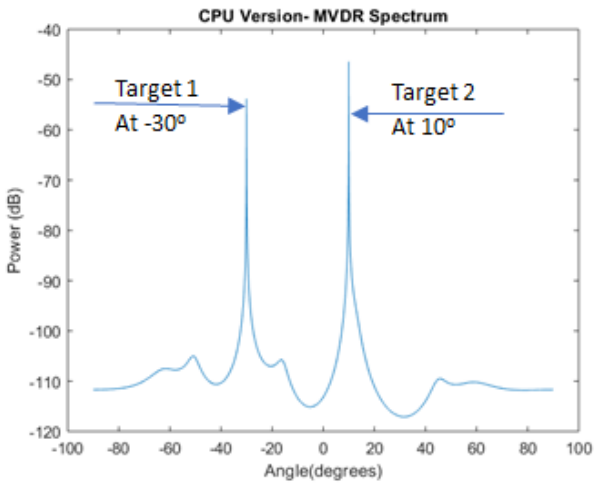
Figure 7. GPU-based MVDR spectrum

Table 4. NVIDIA GeForce GTX TITAN operating specifications [33]

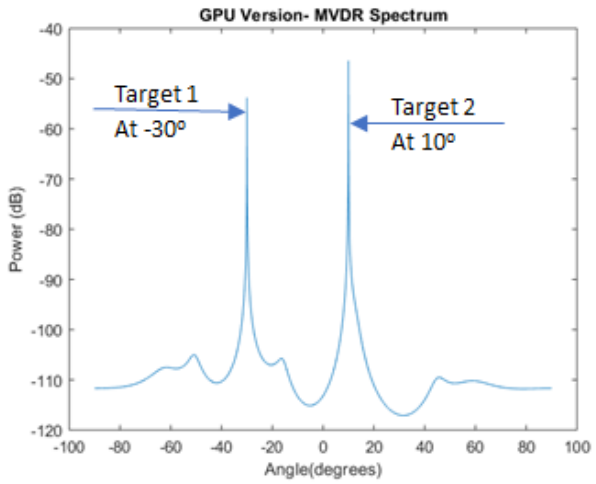
CUDA Cores	2688
Base Clock	837 MHz
Boost Clock	876 MHz
Memory Clock	6.0 Gbps
Memory Bandwidth	288.4 GB/sec
Computing Capability	3.5
Max Threads/Warp	32
Max Threads/Multiprocessor	2048
Max Blocks/Multiprocessor	16
Max Shared Memory/thread block	48 kilobytes
Max Registers/block	65536
Max X Grid Dimension	2^{32}-1

2.6. Simulation Results and Analysis

For simplicity, the MIMO radar system with only two transmit elements were simulated. The number of receivers was varied from four to 100 to determine how an increased number of receivers will affect the speedups and execution times. The CPU-based and GPU-based MVDR spectrums with two transmitters and four receivers are shown in Figure 8. Both spectrums are similar and clearly show two independent targets represented by the azimuth of both cars. Table 5 shows the CPU and GPU execution time for a varying number of receiving elements, as well as the speed up that was obtained by utilizing the GPU. It is shown that as the number of receivers is increased the overall speedup also is increased. This is an expected result since large data sets will take longer to execute serially than in a parallel operation. The strength of the GPU's parallel computing capability is highlighted once the number of receivers is increased to 100, as the overall speedup of the MVDR algorithm is increased to 7.26x.



(a)



(b)

Figure 8. MVDR Spectrum for two transmitting elements and four receiving elements: **(a)** CPU-based MVDR, **(b)** GPU-based MVDR

Table 5. CPU and GPU execution time and speedup for two transmit antennas and different number of receive antennas

Number of Receivers	CPU Execution Time (ms)	GPU Execution Time (ms)	Speedup
4	269.68	53.57	5.04
8	270.68	54.46	4.97
16	343.704	60.44	5.69
24	387.34	61.92	6.26
48	529.22	88.30	6.00
75	703.60	118.13	5.96
100	890.50	122.78	7.26

Overall, this simulation was a success and has proven that a simulated MIMO radar system can be successfully integrated with a GPU. However, with a bigger, more complex system (thousands of TX/RX elements instead of the hundred in this system), utilizing shared memory would allow for an even more significant advantage. The faster execution time associated with shared memory is a well-documented property [20,32]. Unfortunately, this possibility was not explored due to MATLAB's limitations.

3. GPU Acceleration of a Vector Network Analyzer (VNA)-Based MIMO Radar

GPU acceleration of the MIMO radar was also tested in a laboratory setting using experimental data of a VNA-based 4x4 monostatic MIMO radar system [15]. Figure 9 shows the diagram of the experimental MIMO radar using the Keysight N5244A 4-port PNA-X VNA [16]. A four-element microstrip patch antenna array, with a half-wavelength antenna spacing and operating at 8.925 GHz, is connected to a 4-ports VNA, which is connected to a computer running MATLAB/SIMULINK. The VNA measures the phase-coherent scattering (S) parameters between each of its ports. The S-parameter data are then saved in an I and Q format, which can be used as radar returns and can be processed by Simulink to form a virtual array and perform and beamforming so that range and angle information from targets could be extracted. The high-level Simulink model used to process the radar returns is shown in Figure 10. In a laboratory setting, clutter may be removed by performing two sets of measurements: one with targets and another one with only the background.

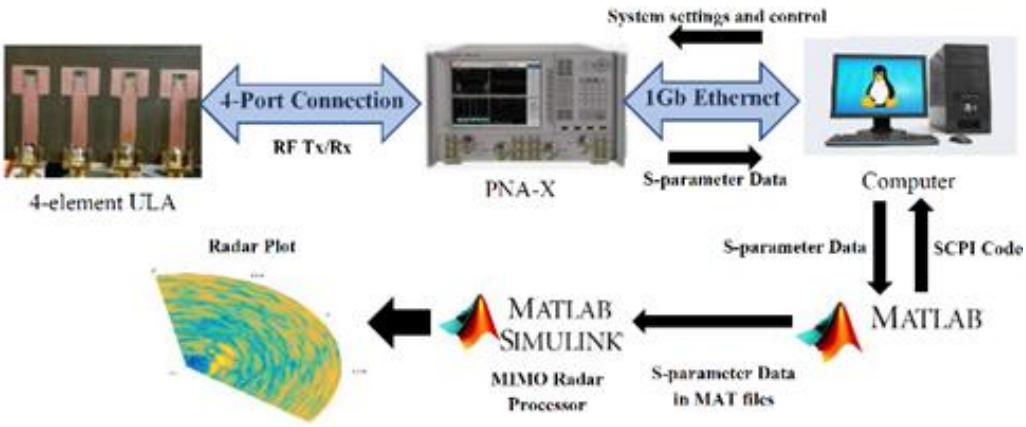


Figure 9. System Overview of VNA MIMO radar [15].

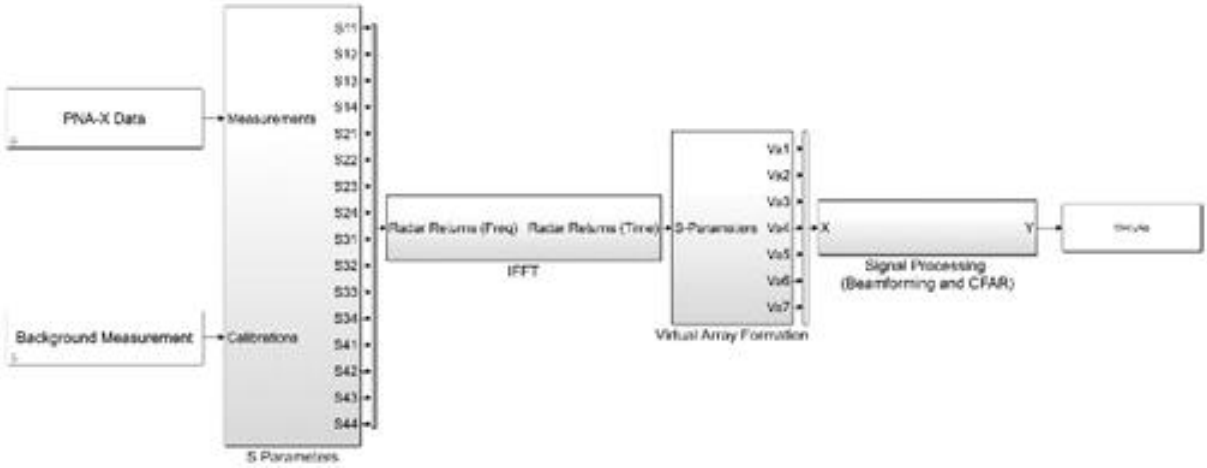


Figure 10. Simulink overview of the PNA-X MIMO radar [15].

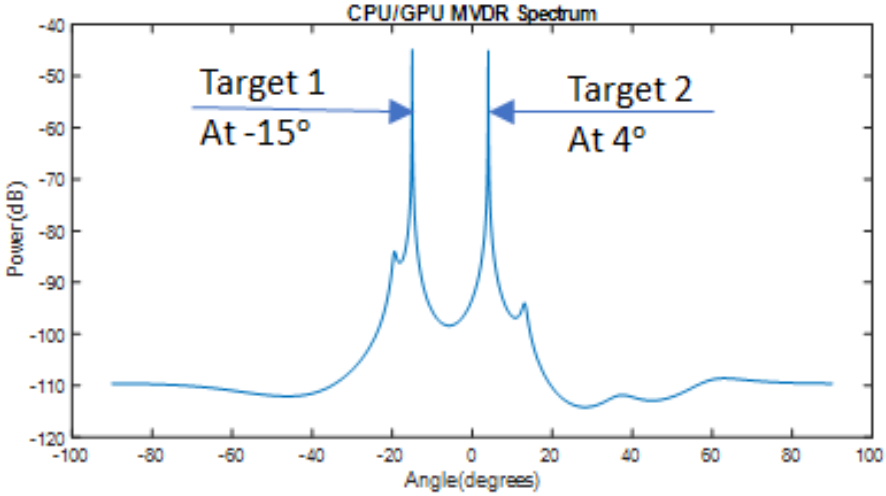


Figure 11. MVDR spectrum of two targets at 4 m separated by 19°

The matrices of the PNA-X Data and Background Measurement from Figure 9 are measured with the same settings to ensure that they have the same matrix dimensions. These two matrices are the input to the Simulink model, which processes the data one sweep at a time. The data from both matrices are sent to the S-parameters block, which subtracts the background measurement from the data with targets and converts the received data from amplitude and phase to I/Q data. The I/Q data are then sent to an IFFT block to convert the data from the frequency domain to the time domain. An IFFT is conducted on each S-parameter and then buffered over a specified coherent integration interval. After the S-parameter radar returns are converted to the time domain, they are sent to the virtual array block for coherent integration and to create a virtual array. It should be noted that since a 4-element ULA is used in a monostatic scheme, the received signals will be mapped to a 7-element virtual array, which will be used to compute the CPU-based and the GPU-based MVDR spectrums. Two cylindrical metallic targets were placed at a range of 4 m then separated in azimuth by 19°. Figure 11 shows the MVDR spectrum that can distinguish between the two closely spaced targets with a GPU execution time that is 5.00 times faster than the CPU. This result concurs with the speedup of 5.04 achieved with the 2 x 4 MIMO radar simulation data that corresponds to an 8-element virtual array.

4. Conclusions

The GPU-based MVDR algorithm was tested using an ACC MIMO-TDM radar system on board of an autonomous car and using experimental data of a VNA-based MIMO radar. The ACC MIMO-TDM radar system was simulated using two transmit antennas and a different number of receive antennas varying from four to 100. All these scenarios were successful at resolving the two moving targets and reached a speedup of 7.26x when a 100-element array was used at the receiver. On the other hand, the virtual array experimental data were extracted from a VNA-based 4x4 monostatic MIMO radar system

and used with the MVDR spectrum to distinguish between two closely spaced targets with a GPU execution time that is 5.00 times faster than the CPU. These experiments proved that it is possible to integrate a GPU within a MIMO radar system to accelerate the MVDR algorithm.

Acknowledgements

We are very grateful to the Canadian Microelectronics Corporation (CMC) for providing the Heterogeneous Parallel Platform to perform the MIMO radar CPU-GPU Simulations.

REFERENCES

[1] J. Li, P. Stoica. MIMO Radar Signal Processing, Wiley-IEEE Press, Hoboken, NJ, 2008.

[2] P. Sévigny. MIMO Radar: Literature survey of papers between 2003 and September 2008, DRDC, Ottawa, ON, TM 2008-333, Mar. 2009.

[3] E. Brookner. MIMO radars demystified; and their conventional equivalents, IEEE International Symposium on Phased Array Systems and Technology (PAST),1–10, 2016.

[4] P. W. Moo, Z. Ding. Tracking Performance of MIMO Radar for Accelerating Targets, IEEE Trans. Signal Process, Vol. 61, No. 21, 5205–5216, 2013.

[5] I. Bekkerman, J. Tabrikian. Target Detection and Localization Using MIMO Radars and Sonars, IEEE Trans. Signal Process, Vol. 54, No. 10, 3873–3883, 2006.

[6] H. Sun, F. Brigui, M. Lesturgie. Analysis and comparison of MIMO radar waveforms, International Radar Conference, 1–6, 2014.

[7] E. Fishler, A. Haimovich, R. S. Blum, L. J. Cimini, D. Chizhik, and R. A. Valenzuela, "Spactial Diversity in Radars-Models and Detection Performance," IEEE Trans. on Signal Processing, Vol. 54, Iss. 3, 823–837, 2006.

[8] A. M. Haimovich, R. S. Blum, and L. J. Cimini. MIMO radar with widely separated antennas. IEEE Signal Processing Magazine, Vol. 25, Iss. 1, 116 - 129, 2008.

[9] P. Stoica, J. Li. MIMO radar with colocated antennas, IEEE Signal Processing Magazine, Vol. 24, No. 5, 106-114, 2007.

[10] J. Li, P. Stoica, L. Xu, W. Roberts. On parameter identifiability of MIMO radar, IEEE Signal Process. Letter, Vol. 14, No. 12, 968-971, 2007.

[11] P. Stoica, J. Li., Y. Xie. On probing signal design for MIMO radar, IEEE Trans. Signal Processing, 55, 8, 4151-4161, 2007

[12] T. Aittomaki, V. Koivunen. Signal covariance matrix optimization for transmit beamforming in MIMO radars. In Proc. 38th Asilomar Conf. Signals, Syst. Comput., Pacific Grove, CA, 182-186, 2007.

[13] Fuhrmann, D.R. ; San Antonio, G. Transmit beamforming for MIMO radar systems using signal cross-correlation, IEEE Trans. Aerospace and Electronic Systems, Vol. 44, No. 1, 171-186, 2008.

[14] Automotive Adaptive Cruise Control Using FMCW Technology, Online available from https://www.mathworks.com/help/phased/examples/automotive-adaptive-cruise-control-using-fmcw-technology.html.

[15] M. Hefnawi, J. R. Bray, J. N. Bathurst, and Y. M.M. Antar. MIMO Radar Using a Vector Network Analyzer, Electronics 2019, Vol. 8, Iss. 12, 1447. https://doi.org/10.3390/electronics8121447

[16] N5244A PNA-X Microwave Network Analyzer, 43.5 GHz|Keysight. Online available from http://www.keysight.com/en/pdx-x201767-pn-N5244A/pna-x-microwave-network-analyzer-435-ghz?cc=CA&lc=eng.

[17] J. Capon. High- Resolution Frequency-Wavenumber Spectrum Analysis, Proceedings of the IEEE, Vol. 57, No. 57, No. 8, 1408–1418, Aug. 1969.

[18] J. Benesty, J. Chen, and Y. Huang. A Generalized MVDR Spectrum, IEEE Signal Processing Letters, Vol. 12, No. 12, Dec. 2005.

[19] S. A. Vorobyov. Principles of minimum variance robust adaptive beamforming design, Signal Processing, Vol. 93, No. 12, 3264–3277, Dec. 2013.

[20] D. B. Kirk and W. W. Hwu, Programming Massively Parallel Processors, Third. Cambridge, MA, United State: Elsevier, 2017.

[21] R. S. Perdana, B. Sitohang, and A. B. Suksmono. A survey of graphics processing unit (GPU) utilization for radar signal and data processing system, 6th International Conference on Electrical Engineering and Informatics (ICEEI), Langkawi, 1–6, 2017.

[22] M. Wu, Y. Sun, and J. R. Cavallaro. Reconfigurable Real-time MIMO Detector on GPU, 43rd Asilomar Conf. Signals, Systems, and Computers, Pacific Grove, CA, USA, 2009, 690–694.

[23] J. Sanson, A. Gameiro, D. Castanheira, and P. P. Monteiro. Comparison of DoA Algorithms for MIMO OFDM Radar, 15th European Radar Conference (EuRAD), 2018, 226–229.

[24] T. Nylanden, J. Janhunen, O. Silven, and M. Juntti. A GPU implementation for two MIMO-OFDM detectors, International Conference on Embedded Comp. Systems: Architectures, Modeling and Simulation, Samos, Greece, 2010, 293–300.

[25] T. Chen and H. Leib. GPU acceleration for fixed complexity sphere decoder in large MIMO uplink systems, in 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), Halifax, NS, Canada, 2015, 771–777.

[26] S. Roger, C. Ramiro, A. Gonzalez, V. Almenar, and A. M. Vidal. Fully Parallel GPU Implementation of a Fixed-Complexity Soft-Output MIMO Detector, IEEE Transactions on Vehicular Technology, Vol. 61, No. 8, 3796–3800, 2012.

[27] Increasing Angular Resolution with MIMO Radars, Online

available from https://www.mathworks.com/help/phased/examples/increasing-angular-resolution-with-mimo-radars.html.

[28] Phased Array System Toolbox, Online available from https://www.mathworks.com/products/phased-array.html.

[29] Parallel Computing Toolbox, Online available from https://www.mathworks.com/products/parallel-computing.html.

[30] CUDA GPUs, NVIDIA Developer, Online available from https://developer.nvidia.com/cuda-gpus.

[31] Numerical Analysis Tools, Online available from https://developer.nvidia.com/numerical-analysis-tools.

[32] NVIDIA, NVIDIA CUDA C Programming Guide. Santa Clara, CA, 2012.

[33] GeForce GTX TITAN | Specifications | GeForce. Online available from https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan/specifications