GPU Accelerated MIMO Array InISAR 3-D Imaging

Xie Xiao-Chun [1] and Yu Ling-Juan [2]

[1] Gannan Normal University, Ganzhou 341000, China, xiexiaochun@sina.com

[2] Jiangxi University of Science and Technology, Ganzhou 341000, China, yljsmile@163.com

Abstract

The efficient separation of scatterers which have same Range-Doppler value is a key problem in InISAR 3-D imaging. In this paper, a GPU accelerated MIMO array InISAR 3-D imaging algorithm is proposed. A MIMO radar array, which is located along the z-axis , is used measure the positions of scatterers of a synthesis scatterer. To speed up the algorithm, a data-parallel programming model based on Jacket is applied. The simulation results show that the proposed method is effective and precisionist for 3-D image reconstruction.

1. Introduction

Inverse synthetic aperture radar (ISAR) is a technique to generate a two-dimensional high resolution image of a maneuver target [1]. The high resolution of a two-dimensional (2-D) image in the range direction is obtained by transmitting a wide band signal and the cross range resolution is achieved by the relative motion between the radar and targets. Three-dimensional (3-D) images are capable of providing a more reliable description of target features, which is advantageous to target recognition [2]. Therefore, high-resolution InISAR (Interferometric ISAR) 3-D imaging has come into focus of the radar signal processing community[3-5].

With the development of Compute Unified Device Architecture (CUDA), many people try to improve radar imaging algorithms by GPU [6, 7]. But programming in CUDA is not convenient for the developers who don't understand the details of GPU. By packing the M language into a GPU compatible form, Jacket for Matlab enables users to write and run code on the GPU in the native M language used in Matlab. Based on Jacket, we use a data-parallel programming model to speed up the InISAR 3-D imaging algorithm.

The paper is organized as follows. In section 2, the MIMO array configuration and imaging algorithm of InISAR 3-D imaging are introduced briefly. Then the implement detail of the algorithm on Jacket is described Section 3. The simulation and analysis is performed in section 4, followed by a short conclusion.

2. InISAR 3-D Imaging Based on MIMO Array

Based on the MIMO radar theory[8], a MIMO antenna array configuration is designed for 3-D InISAR imaging, as shown in Figure 1(a). In this configuration, N_R receivers and N_T transmitters are aligned along Z-axis with different space. The distance between receivers is D_R and the distance between transmitters is D_T, where $D_T \geq N_R D_R$. Transmitter T_i and receiver R_j form a bistatic ISAR imaging system. the equivalent virtual array of this configuration can be shown as Figure 1(b) and virtual monostatic antenna TR_{ij} denotes the bistatic ISAR T_i/R_j, and $R_{Aij}=(R_{ATi}+ R_{ARj})/2$.

The multi-baseline echo data obtained by InISAR is in range-frequency domain, azimuth-space domain and elevation-space domain. The diagram of the InISAR 3D imaging algorithm used to process InISAR multi-baseline echo data, which was introduced detailedly in reference [3], is shown in Figure 2. Note that, most approach of this imaging algorithm can be processed in parallel in GPU.

978-1-4673-5225-3/14/$31.00 ©2014 IEEE

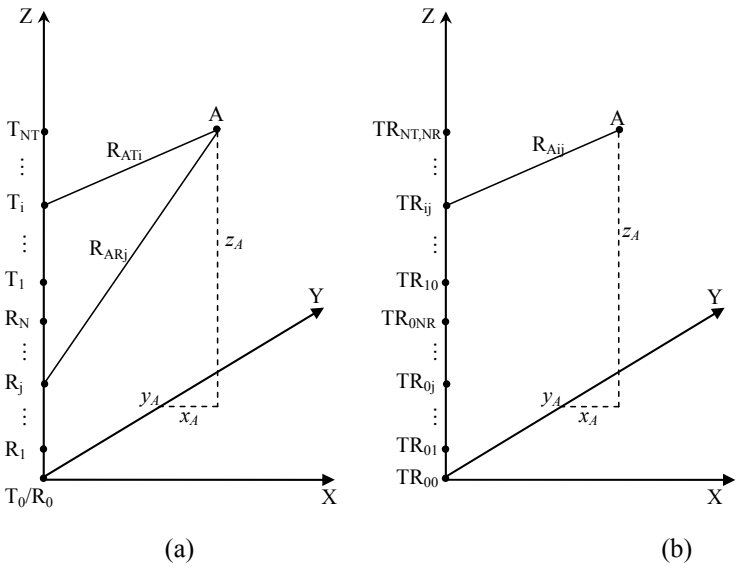

Figure 1. The 3-D InISAR MIMO array configuration.
(a) The geometry of the MIMO antenna array.
(b) The virtual array of the MIMO antenna array.

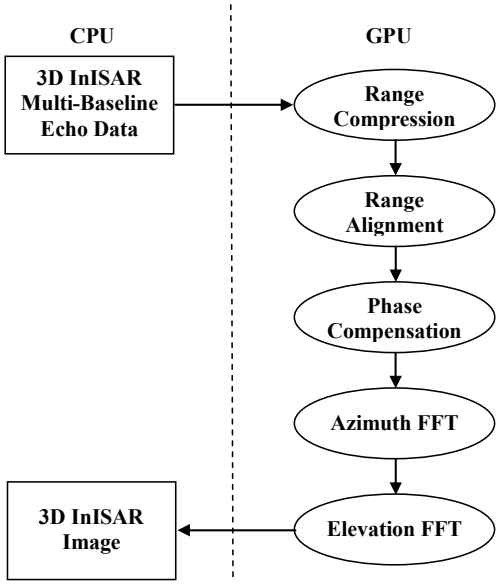

Figure 2. Block Diagram of InISAR 3D Imaging Algorithm

3. Processing Details with GPU in Jacket

GPUs are especially well-suited to address problems that can be expressed as data-parallel computations with high arithmetic intensity. Under the help of Jacket, we use a data-parallel programming model to speed up the InISAR 3D imaging algorithm which discussed above.

In the procedure of InISAR 3-D imaging algorithm, it includes multiple operations of FFT/IFFT. Thus, the efficiency of FFT/IFFT affects processing time of the whole algorithm obviously. In our design, we adopt Jacket to perform 1-D FFT/IFFT along the range, the azimuth and the elevation direction separately, which also supplies the functions to perform batch processing for 1-D FFT/IFFT.

In the traditional process of range alignment, similar operation, range shift estimation and correction, are applied on each range data. The range alignment algorithm described above was relied heavily on nested FOR-loops. In Jacket, the GFOR/GEND loop construction may be used to simultaneously launch all of the iterations of a FOR-loop on the GPU, as long as the iterations are independent. So we design a parallelism process as shown in Figure3. In this process, the whole range data is divided into several parts along azimuth direction, and each part data is assigned to a GPU thread.

There are two phase compensation should applied on every range bin, and also relied heavily on nested FOR-loops. Based on Jacket, we arrange the data in 3D matrix, which can take advantage of the parallel mechanism in the vector multiplication.

4. Simulation and Discussion

In this section, we test the proposed method using echo data of a target which contains isolated scatterers. The target consists of seven scatterers located at the local coordinates as show in Figure 4. The target is located 10 km away from the origin on the Y-axis, the wavelength of transmitted signal is $\lambda = 0.03$ m, the signal bandwidth is 300 MHz (range resolution is therefore 0.5 m). The target rotates around Z-axis with speed 0.015 rad/s. The imaging time is 2 s, and during this time interval, the rotated angle is 0.03 rad, so the resulting cross-range resolution at Y-axis is 0.5 m. There are three receivers and three transmitters are aligned along Z-axis, as shown in Figure 1. Antenna zero is located at the origin.

The distance between receivers is 75 m and the distance between transmitters is 225 m. Thus the resolution on Z-axis is 0.5 m.

The test is done on the desktop computer which equips with 4-core Intel i5 760@2.8GHz CPU, Nvidia GeForce GT 430@700MHz and 12GB CPU memory. The software of the computer includes MATLAB 2010a, version 2.2 Jacket and version 4.2 CUDA.

The reconstructed 3-D image of all scatters reconstructed by the proposed method is showed in Figure 4, where we plot the real positions and the reconstructed positions of all scatters together as for comparison. The small reconstruction error of scatterer 5's z-coordinate occurred due to the approximation in the imaging algorithm[3].

In Table 1, we valid the precision of GPU accelerated method of each imaging step. In the valid, we use SNR and RMS as criterion:

$$SNR = 20\lg\left(\left|I_{CPU}\right| / \left|I_{CPU} - I_{GPU}\right|\right) \tag{1}$$

$$RMS = -20\lg\left(\left|I_{CPU} - I_{GPU}\right| / \sqrt{N}\right) \tag{2}$$

where I_{CPU} is the processing result of CPU based method, I_{GPU} is the processing result of GPU based method, and N is total elements number of the processing result.

From Table 1, we can conclude that the GPU accelerated method has no notable effect on the processing precision. For convenience to see the acceleration effect, we define the speedup factor as:

$$Speedup = \left(CPU \text{ elapsed time}\right) / \left(GPU \text{ elapsed time}\right) \tag{3}$$

From Table 2, we can find that the GPU has accelerated the processing speed in each step of the algorithm.

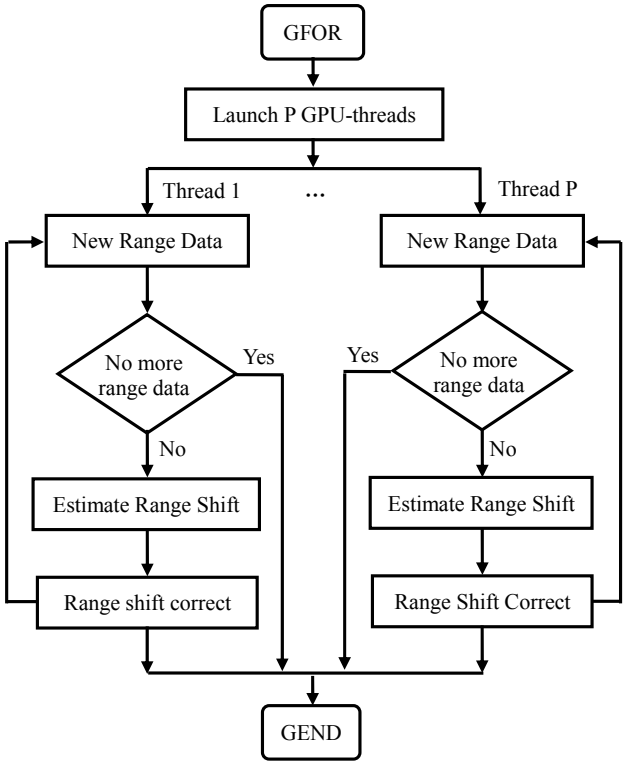

Figure 3. Parallelism of Range Alignment

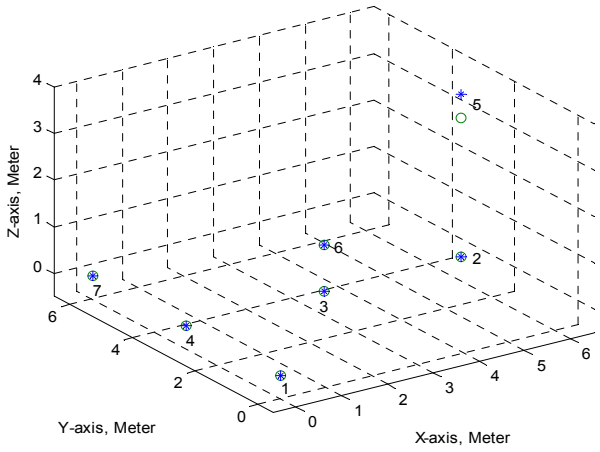

Figure 4. 3-D model and imaging result of the target.

Table 1 Processing precision of GPU accelerated method

	SNR(dB)	RMS(dB)
Range Compression	130.0041	138.9708
Motion Compensation	123.3243	132.2908
Phase Compensation	123.1645	132.1310
Azimuth FFT	123.0612	102.0274
Elevation FFT	124.8383	94.3002

Table 2 Operation Elapsed Time on CPU and GPU

	CPU Elapsed time (/s)	GPU Elapsed time (/s)	Speedup
Data Transmission	/	1.712	/
Range Compression	3.3068	2.0133	1.6425
Motion Compensation	59.8278	22.4203	2.6685
Phase Compensation	0.5013	0.0260	19.2808
Azimuth FFT	1.5251	0.0562	27.1370
Elevation FFT	10.7283	2.4180	4.4368
Total	75.8893	28.6458	2.6492

5. Conclusion

We discussed InISAR 3-D imaging algorithm based on MIMO radar array and the detailed implementation GPU acceleration in Jacket for Matlab. By exploiting the performance of GPU, we show the potential of using GPU for real-time InISAR processor design.

Acknowledgements

This work was supported by Jiangxi Province Educational Committee Science Foundation for Youths (No. GJJ13387) and the Jiangxi Natural Science Foundation (No. 20114BAB201020).

Reference

1. D. R. Wehner, “High-resolution Radar,” Artech House, 1995.

2. Qun Zhang and Tat Soon Yeo, “Three-dimensional SAR imaging of a ground moving target using the InISAR technique,” IEEE Transactions on Geoscience and Remote Sensing, Vol. 42(9), pp: 1818 - 1828, 2004.

3. Xie Xiaochun and Zhang Yunhua, “3D ISAR imaging based on MIMO radar array,” 2nd Asian-Pacific Conference on Synthetic Aperture Radar, Xian, Shanxi, 2009, pp: 1018 – 1021.

4. Falcone, P., Colone, F., Macera, A., Pastina, D. and Lombardo, P., “Advances in ISAR processing for high resolution cross-range profiling with passive radar,” 13th International Radar Symposium (IRS), Warsaw, 2012, pp: 421 - 425.

5. Yabo Liu, Mingcong Song, Kun Wu, Wang, R. and Yunkai Deng, "High-Quality 3-D InISAR Imaging of Maneuvering Target Based on a Combined Processing Approach," IEEE Geoscience and Remote Sensing Letters, , Vol. 10(5), pp: 1036 - 1040, 2013.

6. Rubin G, Sager E V and Berger D H, “GPU acceleration of SAR/ISAR imaging algorithms,” System Planning Corporation, Whitepaper, 2010.

7. Malanowski, M., Krawczyk, G. , Samczynski, P. , Kulpa, K. , Borowiec, K. and Gromek, D., “Real-time high-resolution SAR processor using CUDA technology,” 14th International Radar Symposium (IRS), Dresden, 2013, pp: 673 – 678.

8. Stoica P and Li J, “MIMO radar signal processing,” Hoboken, New Jersey, Wiley & Sons, 2008.