GPU Accelerate RD Algorithm Based on MATLAB

[I]Feng He, [II]Yi Ren, [III]Yi Chen, [IV]Qingliang Cao
[I]College of Optoelectronic Engineering, [II,III,IV]College of Communication and Information Engineering
[I,II,III,IV]Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Abstract
With the development of GPU, MATLAB can provide support for NVIDIA GPU acceleration by using Parallel Computing Toolbox, which is very simple and does not need to execute the underlying programming. In order to improve the efficiency of SAR imaging, this paper introduces two methods of the GPU acceleration in MATLAB and puts forward how to use the methods to speed up the RD algorithm. After analyzing the speedup ratio of RD algorithm under different conditions, we conclude that use GPU to accelerate RD algorithm in MATLAB can increase the computing speed several times.

Keywords
GPU; Parallel Computing; SAR; MATLAB; Speedup; RD

I. Introduction

In recent years, with the improvement of hardware technology, the spatial resolution of SAR images has been improved greatly and the number of the data gradually increase. So its operation efficiency was confined by the underlying software architecture which formed the bottleneck of low efficiency[1]. Usually people adopt DSP technology, however, this technology need application programmer to have high level understanding and cognition of the architecture of the hardware system, which is disadvantage for the updated code portability of hardware platform[1]. Nowadays, the software and hardware technology of graphics processor become more and more mature, its powerful floating-point operations and parallel processing ability have been widely used in SAR image processing.

GPU□Graphic Process Unit) is the core component of computer display device, which is used for calculation in graphics display. Usually a GPU contains several stream processors, due to the need of image processing, the stream processors are designed to work in parallel. GPU which dedicate to provide more computing units and high data bandwidth in a limited area is good at large-scale concurrent computing, through a large number of computing units for parallel processing and repetitive operations to realize the strong calculation ability and reduce the cost of the system. The fig.1 illustrates the difference between the computational resources of CPU and GPU. Relative to CPU, GPU has a longer pipeline, and doesn't have a cache [2]. The CPU is designed for serial processing, whereas GPU is designed for parallel processing with more number of ALUs that helps to run parallel code to solve computational intensive problems [3]. CUDA is a software and hardware system that can make the GPU work as a device for data parallel computing. It was published by the NVIDIA Company in 2007. Through this system, researcher can develop parallel computing program on Personal Computer only need an ordinary GPU that support the CUDA system. Due to the increase of parallel computing demand, the parallel computing method has been aroused many attentions, which is more economical compared with the cluster and parallel computer.

MATLAB is a kind of efficient technical computing language, which is very popular in the application of scientific research and teaching. It contains rich toolbox functions, and can get a good solution of problems in the field of system simulation and calculation which encountered in the study. But MATLAB computing efficiency is low, compared with other high-level languages, MATLAB program execution is slow [4]. Scientific researchers always use two measures to improve the speed of MATLAB. One is to buy some expensive equipments such as servers or workstations to increase hardware performance. Although this method can solve the problem of speed, it gives researchers a financial burden [4]. The other way is transplanting MATLAB algorithms and reprogramming by high level languages such as C++, which can improve efficiency of computing, but it demands programming skills for researchers. It would be very time-consuming and laborious, and increase research effort [4]. In order to solve this problem, MathWorks announced that by using Parallel Computing Toolbox, MATLAB can provide support for NVIDIA GPU. With this support, engineers and scientists can use MATLAB to carry on the GPU parallel computing. It is very simple and does not need additional programming technology.

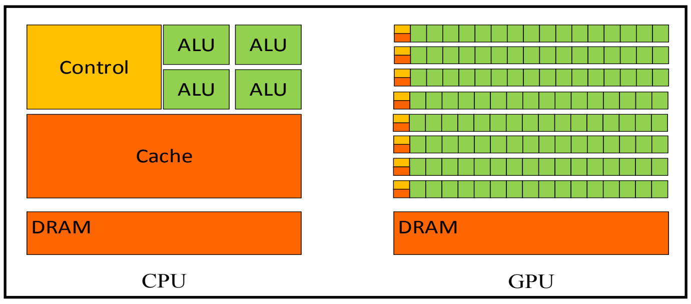

Fig.1: Computational Resources of CPU and GPU

II. Configuration

The basic configuration includes Windows7 operating system, MATLAB R2010b or later, the GPU support CUDA 1.3 version or later [5], and corresponding driver. GPU can be connected to the host through PCI Express slot, usually a host can be installed several graphics cards. In this paper, taking the cost into consideration, we only use one GPU. After installing the driver and software, the gpuDevice command can be used in the MATLAB command window to check the CUDA version and other GPU related information such as compute capability, max number of the Grid and Block, whether to support double-precision floating-point arithmetic etc.

III. Approach Analysis

GPU Processing for Built-in MATLAB functions [5]: The simplest method is to use the Built-in MATLAB functions in the device. In the CUDA programming model, CPU is regarded as the host and the GPU is regarded as device. Nowadays, the MATLAB supports hundreds of GPU enabled functions such as FFT, matrix multiplication and left matrix division to run on GPU. The built-in functions such as gpuArray and gather functions are used to transmit data from CPU to GPU and to get result back from GPU to CPU as shown in fig.2 [3]. Different MATLAB support the number of the built-in functions are different, we can check the MATLAB built-in functions that support gpuArray through the methods('gpuArray') command in the MATLAB command window. By using gpuArray and gather functions, we can decide how to move data between MATLAB workplace and GPU memory, or create data stored in GPU memory directly, so as to reduce the data transmission between the host and device. In order to implement the optimal flexibility and usability, we also use the data on the GPU and the data on the MATLAB workspace in the same function. Through these Built-in MATLAB functions we can execute the standard functions on the GPU to accelerate the code without spending a lot of time on them.

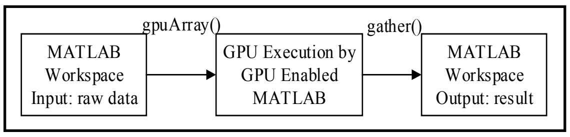
MATLAB Workspace Input: raw data
gpuArray()
GPU Execution by GPU Enabled MATLAB
gather()
MATLAB Workspace Output: result

Fig. 2: Data and Result Exchange between CPU and GPU

GPU Processing for Non-Built-in MATLAB functions: There is another way to use the GPU for our own MATLAB functions. We can define the MATLAB functions to perform the scalar arithmetic operations for data that was loaded on the GPU memory. Through this approach, it good for us to expand and define the function that performed on the GPU to build complex applications and achieve performance acceleration. Although this approach is simple, the kernel calls and data transmission are less than the above method. To run our own MATLAB code on a GPU, we can use GPU-enabled function arrayfun which applies our own function to each data element by an elementwise operation [5]. It can make full use of data parallelism and minimize the cost of data transmission and kernel calls.

IV. RD Algorithm

Range doppler algorithm is the most general method of the synthetic aperture radar image processing. It uses the theory that radar target echo signal can be approximated as independent linear FM signal in azimuth and range and then use the matched filter in the range and azimuth respectively, thereby simplifying the imaging process. The basic idea is to decompose 2-D processing into two 1-D form and the algorithm consists of three main steps which are range compression, range migration correction and azimuth compression. Firstly, to make the calculation convenience, matched filtering can be used in the frequency domain through the FFT, which is easy to use IFFT and compensate the range migration at the same time. So it can make the echo envelope alignment. Besides, we can multiply the linear frequency phase factor as the same time as using matched filtering in frequency domain. In the azimuth direction, usually the azimuth imaging process needs to use for the signal pulse compression, because of the impact response function is a 2-D function that related to the distance and direction. The calibration algorithm for linear range migration imaging in time domain as fig. 4 shown [6].

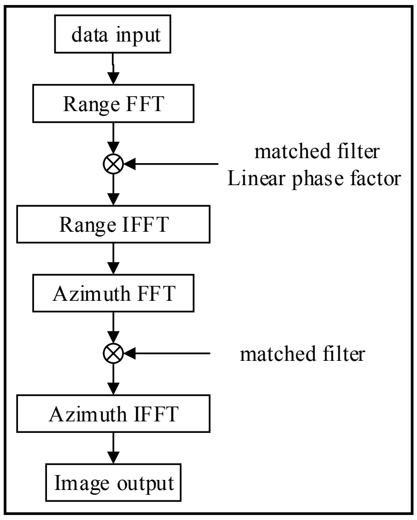

Fig.3: block diagram of RD

The basic process of point target imaging on GPU as the same as it on the CPU. The first step is initialization phase. On the CPU, set some parameters such as the position and of target, sampling points of range and azimuth, bandwidth etc. Then transport the data from CPU to GPU through the gpuArray function or store the data on the GPU directly. By doing that we can start to calculate the echo data. The point target echo can be generated by using the arrayfun to call the GPU kernel. In this article we implement five point targets imaging, so it needs to call the kernel five times, each call completes the calculation of one point. After all the point target echo are calculated, we can calculate the sum of five data on the GPU to get the whole data. And then use the MATLAB build-in functions that support GPU processing to realize the FFT and IFFT operations, in order to improve the efficiency.

V. Experiment and Results

To test the performance of accelerating RD algorithm, the simulation of the algorithm was implemented on the Windows operating system and the Table.1 lists some information of CPU and GPU.

Table.1: the model of CPU and GPU

Name	model	configuration
CPU	AMD Athlon(tm) II X3 445	3.1GHZ, 4GB RAM
GPU	NVIDIA GeForce GT 730	902MHZ,384Cores, 1GB VRAM

The table.2 and table.3 list the speedup under different sampling points. As the table.2 and table.3 shown, with the increase of sampling points, the speedup gradually increase. When the sampling points up to 2048*2048, the speedup can up to 2.7 times by using Multiple CPU cores. When we use one CPU core, the speedup can reach to 4.3. From these we can see the floating-point computation and parallel ability of GPU. It can also improve the efficiency of RD algorithm while the point target imaging are similar both on the GPU and CPU, as the fig. 4 shown.

Table. 2 :the speedup using several CPUs

sampling points	512*512	1024*1024	2048*2048
Multiple CPU	0.621427s	1.548426s	5.497399s
GPU	0.489895s	0.784984s	2.020836s
Speedup	1.268490	1.972557	2.720359

Table 3 :the speedup using one CPU

sampling points	512*512	1024*1024	2048*2048
One CPU	0.901665s	2.490860s	8.920586s
GPU	0.494413s	0.819009s	2.057353s
Speedup	1.823708	3.041310	4.335953

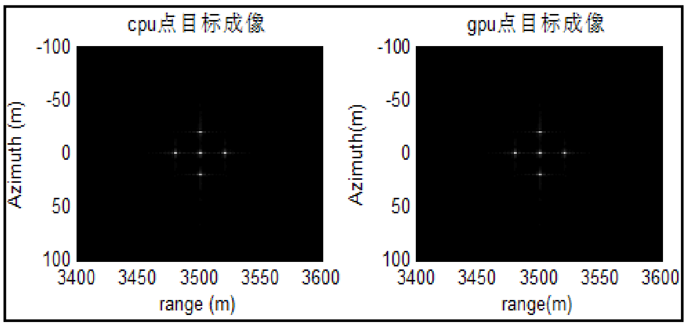

Fig.4: the results of RD algorithm

VI. Conclusion

TThis paper introduces some basic methods to implement the GPU parallel computing in the MATLAB, and at the end we accelerate the RD algorithm by using the gpuArray and arrafun functions. The experimental results prove that the GPU can improve the computing efficiency of the RD algorithm. In addition, it can save time and improve the efficiency of the code for the researchers who are not good at programming.

References

[1] *Gao Yueqing, Zhang Yan, Liu Weiguang. Study on CS Algorithm for SAR imaging Based on CUDA [J].Computer&NetWork, 2012, 38(07):55-57*

[2] *Liu Shaobo, Liu Minggui, Zhang Guohua. Model of accelerating MATLAB computation based on CUDA [J]. Application Research of Computers, 2010, 27 (6): 2140-2143.*

[3] *Vilas H.Naik, Chidanand S. Kusur. Analysis of performance enhancement on graphic processor based heterogeneous architecture: A CUDA and MATLAB experiment [C]. Conference on Parallel Computing Technologies, 2015:1-5*

[4] *Zhong Liaobo. Comparative Analysis of GPU an d CPU[J]. Technology and Market,2009,16(9):13-14*

[5] *Jung W. Suh, Youngmin Kim. Accelerating MATLAB with GPU Computing: A Primer with Examples [M]. 225 Wyman Street, Waltham, MA 02451, USA :Morgan Kaufmann Publishers,2014:100-108*

[6] *Zheng Bao, Mengdao Xing, Tong Wang. Radar imaging technology [M]. BeiJing: Publishing House of Electronics Industry, 2005.4:132-146*