Accelerating the image registration of MRI volumes by modern GPGPU parallel computation

S-Y. Ju[1], Y-W. Tang[1], and T-Y. Huang[1]
[1]Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

Introduction

Image registration [1] has been an important topic in the MRI applications, such as longitudinal follow-up studies, brain-normalization for group statistics and motion correction for fMRI studies. There are many different algorithms of image registrations. In general, the calculations require a lot of iterations of coordinate transformations to find the displacements and rotations (3D volume: six degrees of freedom) between two image volumes. Thus, image registration is generally a time-consuming problem even working with the state-of-the-art workstations. In our study, we proposed to use the parallel computing on recently advanced general-purpose computation on graphic processing units (GPGPU) to accelerate the registration calculations, especially for the popular SPM system [2].

Theory

The image registration generally requires three steps. First, the coordinates of the target image volume is transformed with a formula, as given by EQ.1, where *x, y, z* are the original position of pixel and *x', y', z'* are the new position of pixel after transformation. In the transformation matrix, the parameters with r label are the rotation factor determined by three Euclidean angle, θ, ψ, φ and the parameters with t label are the translation factor. Second, a cost function (e.g. correlation, mutual information) is calculated by measuring the similarity of the reference volume and the target volume. Third, an optimization algorithm checks the cost function and estimates new affine matrix for next iteration loop. Note that the transformation process has to be applied on each pixel in the 3D volume. The calculation of each pixel is independent to each other and thus the transformation is suitable for parallel computation.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \text{(EQ.1)}$$

Materials and Methods

Our goal was to implement an accelerated registration that is compatible to the popular SPM system [2]. In the SPM program, the cost function of the image registration is mutual information (MI) [1]. To calculate MI, a joint histogram of the reference volume and the transformed target volume is generated by a dynamic-link library "spm_hist2.dll". In our study, we re-implemented the library with compute unified device architecture (CUDA) programming model[3], a program tool for executing the computation on GPU hardware. First, the reference volume, the target volume, and the transformation matrix were transferred from MATLAB-based SPM program to C environment through the "mexfunction" interface of MATLAB. Second, the data was duplicated in the DRAM of the graphic card. Then, the coordinate transformations on the pixels of the target volume were calculated with massive parallel threads of the GPU. Finally, the joint histogram of the two volumes (i.e. reference and target) was transferred back to the MATLAB workspace for further iterations.

The GPU-accelerated image registration was applied on the simulated datasets with different matrix size and the volunteer datasets (N=3) acquired 3.0T MR system (Siemens Trio, Germany) using an eight-channel head phase-array (T1-weighted 3D MPRAGE, matrix: 256 × 256 × 176, resolution: 1mm^3). The volunteer had two sessions of T1-MPRAGE scans. Between each scan, the volunteer was asked to leave the scanner and then enter the scanner again, mimicking the longitudinal follow-up study. The obtained data sets were transferred to a computer workstation equipped with GeForce GTX295 (NVIDIA, USA, processor cores per GPU: 240 RAM per GPU:896MB) and Core i7 920 (Intel, USA, 4 cores) to test the efficiency of the proposed method. The iteration setting of SPM was set to two mm for coarse pre-registration and one mm for the precise iteration loop.

Results

Figure 1 shows the speed comparison on matrix size of the image volumes. The simulation measured the time of a single execution on "spm_hist2.dll" (both CPU version and GPU version) with different matrix sizes. Note that the acceleration factor reached to about 25 while the size of image volume was around 256^3. For the volunteer datasets, the average acceleration factor was 23.28±2.20 (see table 1).

Discussions and Conclusions

In our study, the recently advanced GPU parallel computing technique was proposed to speed up the calculation of the image registration. Since SPM was widely used in our society, we decided to implement our program fully compatible with it. The original SPM program "spm_hist2.dll" written in C language was re-programmed into CUDA parallel kernel. Thus, the SPM users can simply replace the registration library to the program we developed to gain the benefit of GPU parallel computing. Using the commercially available GPU hardware (cost ~500 USD), we got ~23-fold speed up or 300~400 seconds of the time reduction from the volunteer study. The huge amount of time reduction can be beneficial to the clinical practice which requires faster response as possible or the scientific computing which generally sacrifices the precision for shorter computing time. More acceleration may be obtained while using the hardware equipped more GPU cards. We applied this method to the application of automatic slice positioning (ASP) (data not shown and will be presented in another abstract) and the time reduction by GPU-computation largely increased the practicality of it (ASP). In conclusion, GPU-accelerated computation is a promising method which dramatically speeds up the computation of the image registration problem.

References

[1] Studholme (1999) Pattern Recognition, 32:71-86. [2]SPM, http://www.fil.ion.ucl.ac.uk/spm/ [3] CUDA, http://www.nvidia.com

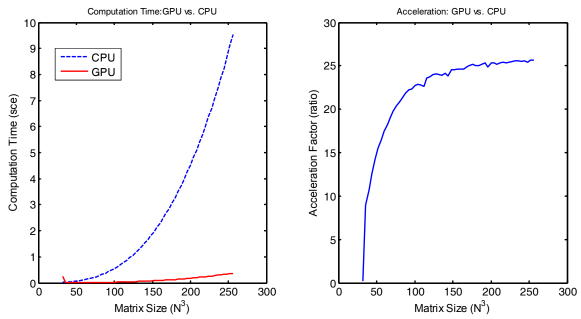

Figure 1: Computing speed comparison using simulated data (CPU versus GPU). Note that the computation time of CPU increased a lot while the matrix size of the input data is larger (see left figure, blue dashed line) whereas the computation time of GPU only increase a little. (blue solid line). The acceleration factor reached to about 25 while matrix size is 256^3 (see right figure).

Table 1. Calculation time of image registration (CPU versus GPU)

Subject #	1	2	3	Avg±Std
CPU (sec)	414.82	378.55	372.62	388.66±22.85
GPU (sec)	17.56	14.97	17.81	16.78±1.57
Acc. factor	23.62	25.29	20.92	23.28±2.20

Table 1: Comparison on the calculation time of CPU and GPU. Note that GPU method dramatically reduced the image registration process of the volunteer datasets with matrix size 256 × 256 × 176.