Physica A 416 (2014) 400–410

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

Optimized Laplacian image sharpening algorithm based on graphic processing unit

Tinghuai Ma [a,*], Lu Li [b], Sai Ji [b], Xin Wang [c], Yuan Tian [d], Abdullah Al-Dhelaan [d], Mznah Al-Rodhaan [d]

[a] Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing 210-044, China

[b] School of Computer & Software, Nanjing University of Information Science & Technology, Jiangsu, Nanjing 210-044, China

[c] Huafeng Meteorological Media Group, Beijing 100080, China

[d] Computer Science Department, College of Computer and Information Sciences, King Saud University, Riyadh 11362, Saudi Arabia

HIGHLIGHTS

- Propose a parallel Laplacian sharpening method based on CUDA with global memory.
- Design an improved parallel algorithm with shared memory of GPU, increase the efficiency.
- Compare the efficiency between shared memory with global memory in GPU, and prove the outperform is better than traditional OpenCV algorithm.

ARTICLE INFO

Article history:
Received 15 June 2014
Received in revised form 22 August 2014
Available online 16 September 2014

Keywords:
Image sharpening
Laplacian algorithm
GPU
CUDA

ABSTRACT

In classical Laplacian image sharpening, all pixels are processed one by one, which leads to large amount of computation. Traditional Laplacian sharpening processed on CPU is considerably time-consuming especially for those large pictures. In this paper, we propose a parallel implementation of Laplacian sharpening based on Compute Unified Device Architecture (CUDA), which is a computing platform of Graphic Processing Units (GPU), and analyze the impact of picture size on performance and the relationship between the processing time of between data transfer time and parallel computing time. Further, according to different features of different memory, an improved scheme of our method is developed, which exploits shared memory in GPU instead of global memory and further increases the efficiency. Experimental results prove that two novel algorithms outperform traditional consequentially method based on OpenCV in the aspect of computing speed.

1. Introduction

Many researchers have devoted themselves to the research of image processing which aims at improving quality of images to enhance the visual effect of people. Image sharpening as a technology is widely used in practical application to make the edges, contour lines, and details of the image clearer. For example, during photographing, pictures will become blurry because of shaking and under this case, the normal solution is scaling down the resolution to make the pictures clearer.

* Corresponding author. Tel.: +86 25 5873 1267.
E-mail address: thma@nuist.edu.cn (T. Ma).

http://dx.doi.org/10.1016/j.physa.2014.09.026
0378-4371/

However, using image sharpening can well solve this problem without affecting the image quality. Along with the scientific and technological progress, image processing plays a more and more important role in complicated scientific fields like materials science [1,2], biological engineering [3,4] and physics [5–7]. Laplace operator [8] is a second derivative operator which is utilized widely [9–11]. It can enhance the discontinuity of gray values in the image so that it is used in image sharpening, edge detector and abating the gradual change of gray areas [12].

Several aspects of image processing make itself computationally challenging. A single image can be viewed as a data set with considerable size and obviously this set is consisted of many pixels. In many algorithms, usually there are several operations to be performed on each pixel in the set, which brings a large amount of computations. For the time-saving reason, parallel schemes have been proposed to accelerate image processing in early years [13,14] and these early attempts reveal the importance of speeding up image processing. In digital image processing, the computation speed is always a negative factor that restricts the application of graphics technology. With the rapid evolution of modern computer graphic and image technology, the amount and complexity of computation are both increasing and many applications require real-time response [15]. However, the speed of current image processing based on CPU is not fast enough. How to realize high speed computing under the existing hardware condition is a research frontier.

In order to meet the need of computer game industry, the GPU emerges. When GPUs first came out, they were sophisticated hardware only used in the fields like graphic processing, video codec and large 3D games. But now they are designed for general purpose computing, as GPUs possess the strong power of floating-point computing, parallel computation and high memory bandwidth. Moreover, GPU can perform Single Instruction Multiple Data (SIMD) operations that are suitable for tasks with requirement of high speed and short delay [16]. Because of its strong computing power, GPU has been attracting attentions of more and more researchers from various fields such as physics [17–20], medical science [21] and data mining [22]. Paper [17] does some simulations on a single GPU to produce bubbles and solid structures in a tiny, annular container, partially filled with a granular material, shaken vertically and paper [18] implements MC program on a many-core GPU after being fully parallelized. In Ref. [19], high resolution numerical simulations including noise in the diffusion coefficient for the bacteria is implemented using GPUs. The Ref. [20] presents the strategies and performance of the GPU implementation of a complex networks package and the Jacket toolbox of MATLAB is used. Whatever the problem is, when using GPU to accelerate an algorithm, the parallelism properties of the algorithm should be analyzed first, and then the parallel method can be designed including the use of threads and memory, which is the general process of parallelization with GPU. What we believe is that, along with the development of GPU, it will make great contributions to frontier subjects and drive technology progress.

There are various kinds of processing techniques in the field of image processing. Some techniques are suitable for GPU implementation such as pyramidal blending [23], spatial diffusion [24], image compression [25] and so on. In Ref. [26], four methods implementing neighboring filtering are proposed and compared. The four methods are named Line Access, Block Access, 4X-Line Access and 4X-Block Access. It is worth mentioning that the shared memory is utilized in the latter three methods. The experimental results show that 4X-Block at the block size of 16 has the fastest computing speed among them, which loads and processes data in shared memory in GPU. It is also shown that this method is about $1.45\times$ faster than the basic method implemented on GPU. In Ref. [27], A vessel enhancement algorithm based on GPU is proposed. Vessel enhancing diffusion (VED) is basically a smoothing filter with strength. The direction of diffusion is determined by a "vesselness" measure. When processing small-sized image data, the acceleration ratio can reach 74, however, when data capacity getting larger, the ratio fell to 28. Authors claim that reason for this reduction is the limits of memory size. Paper [28] proposes a GPU implementation of fixed-sized kernel median filters. This method is implemented through memory fine tuning and use of GPU registers and its median drastically outperforms existing implementations. In order to reduce data transmission time, the texture memory is used to store the input image data and the pinned memory is used to copy output image back to CPU. To use the register efficiently, the forgetful algorithm is adopted during the searching of median value. Besides, it is optimized to hide latencies generated by memory access and data dependent instruction calls, taking advantages of window overlapping and Instruction Level Parallelism (ILP). Edge detection technique is mapped to an implementation on CUDA using shared memory and coalescence [29]. In the parallel version, take the limited shared memory size into consideration, not all the rows are transferred to shared memory at the same time, instead, authors start by sending the initial n rows. When the first row is no longer necessary, it is replaced with the $(n+1)$th row. Authors [30] use CUDA 3.0 to develop parallel quick shift because it operates on each pixel of an image, and the computation which takes place at each pixel is independent of its distant surroundings. As global memory on GPU is relatively slow, and there will be an exponential increase in demand for shared memory when the parameter in quick shift changes, authors map the image and the estimate of the density to 3D and 2D textures, respectively. Results show that method with texture outperforms those without texture and implemented on CPU.

To keep up with demanding computations, combining the traditional Laplace-based image sharpening with modern GPU parallel computing technology is necessary. In this paper, GPU-based Laplacian image sharpening (GLIS) and GPU-based Laplacian image sharpening with shared memory (SM_GLIS), two novel parallel implementations of Laplacian sharpening suitable for GPU are proposed, which are pixel-level fine-grained parallelism optimizations. GLIS is implemented with global memory and gets good returns. SM_GLIS is an optimizing version of GLIS after further discussion on the differences between global memory and shared memory in GPU. In the experiment, the comparisons between the execution time of OpenCV-based Laplacian sharpening and the proposed parallel methods are presented, which prove the superiority of proposed algorithms.