2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing

GPU-based Barrel Distortion Correction for Acceleration

Luo Shuhua, Zhang Jun
School of Information Science and Engineering, Central South University
Hunan Engineering Laboratory for Advanced Control and Intelligent Automation
Changsha, China
junzhang@mail.csu.edu.cn, luoshuhua10@163.com

***Abstract*—Geometric correction is a practical and effective barrel distortion correction method. It mainly consists of two stages: the first stage is to take coordinates mapping from distortion image to correction image; the second stage is bilinear interpolation. It involves a certain amount of calculation, and the larger the image is, the more the quantity of calculation is. What's more, the processing speed of geometric correction implemented on central processing unit (CPU) can't meet the need of high-speed in real-time application fields. Compared with serial processing pipeline of CPU, graphics processing unit (GPU) has special parallel processing pipeline which is suitable and fast for mass data calculation in parallel. Therefore, it can provide an implementation of geometric correction on GPU with considerable acceleration effect. The paper proposes an implementation of geometric correction on GPU by using open graphics library (OpenGL) and graphics library shading language (GLSL) for portability. The experiment results show that the full execution performance of the implementation on GPU is over 190 times speedup of that completely on CPU at most, which obtains a high-speed processing effect.**

***Keywords*—geometric correction; barrel distortion; GPU; radial distortion**

I. INTRODUCTION

Wide-angle lens are widely used for obtaining wider field of vision in a variety of fields, for example, security monitor, navigation, medical domain and surveillance. Meanwhile it leads to a corresponding disadvantage of serious barrel distortion which appears in the captured images. Barrel distortion shows nonlinear changes in the outer area of the image where objects appear smaller than their actual size since the image area is compressed in the outer region of the image. It has been an obstacle for image identification, analysis and judgment in application fields where obtaining estimation of quantitative parameters is critical especially in clinical application. Therefore, it is necessary and important to utilize barrel distortion correction methods to get undistorted images in case significant errors introduced by distortion images disturb image analysis [1] [2].

Several researchers have proposed various mathematical models of image distortion. Generally, there are radial distortion model, hexagonal lattice pattern and grid-like correction template [3]. But they are complicated to implement and need high precision templates. Geometric distortion correction is a practical and effective barrel distortion correction method. It takes coordinates mapping and grayscale interpolation [4]. As for space mapping of coordinates, it can be divided into forward mapping from distortion image to correction image and back mapping from correction image to distortion image. As for grayscale interpolation, it can be classified as bilinear interpolation, nearest neighbor interpolation and double tri-linear interpolation [5]. Yaqiang Liu etc. proposed a kind of geometric distortion correction method [3]. It obtained radial distortion coefficients on the basis of geometric distortion characteristics of images captured by wide-angle lens, took forward mapping according to the distance between distortion point and distortion center, and utilized bilinear interpolation to get undistorted pixel values for restoring image. All the above are software solutions implemented on CPU which has the characteristic of serial processing. However, they can't meet the requirement of high-speed which is often required for real-time applications in medical, navigational and surveillance fields.

Hunan Science and Technology Project (2013GK3006).

As for barrel distortion correction and the algorithms that include a large scale of calculations, there is a common point that computational manipulation on all pixels have the same functionality, which is slowly processed on CPU. The serial processing mode of CPU means processing pixels in sequence, so it appears the disadvantage of low-speed when dealing with large image. While parallel processing mode of GPU means processing a certain amount of pixels at the same time by a certain amount of processors. Thus, the time to simultaneously deal with these pixels is greatly reduced than the time to serially deal with these pixels. Fortunately, GPU has this kind of special parallel processing pipeline which can be utilized to take normal graphics rendering and accelerate mass parallel data calculations. There are dozens of (or even hundreds of) vertex shader processors and fragment shader processors to simultaneously perform pixel operations according to certain calculation procedure. What's more, single instruction and multiple data (SIMD) architecture of shader processor array is suitable and fast for mass data calculations in parallel. Consequently, it provides us with a GPU-based implementation of barrel distortion correction by using open graphics library (OpenGL) and graphics library shading language (GLSL) for portability, which can achieve a significant speed-up effect and meet the requirement of real-time. The rest of this paper is organized as follows. Barrel

978-0-7695-5088-6/13 $31.00 © 2013 IEEE
DOI 10.1109/HPCC.and.EUC.2013.121

distortion and geometric correction are introduced in Section II. Section III describes the implementation on GPU with OpenGL and GLSL in detail. The following is experiments and evaluation in Section IV. The conclusion is provided in Section V.

II. Geometric Distortion Correction

A. Barrel Distortion

The configuration of the camera lens produce distorted images. There are three most common lens which are shown in Fig. 1. The left is standard lens, incidence angle equals exit angle when scene light pass through it, so there is no distortion in captured images. The right is phone lens, when scene light pass through it, incidence angle is greater than exit angle and the captured image shows pincushion distortion. The middle is wide-angle lens, incidence angle is less than exit angle when scene light pass through it, it will produce barrel distortion which is showed in image. The deformation is relatively small in the center of the image and greater when father away from the center of the image.

Barrel distortion is nonlinear, which consists of radial distortion and tangential distortion. But radial distortion is generally much larger than tangential distortion. When using nonlinear optimization algorithm to correct nonlinear distortion, introducing too much nonlinear parameters not only reduce accuracy but also lead to system instability [6]. Thus, this paper only considers the impact of radial distortion and directly ignores tangential distortion.

B. Geometric Correction

Geometric correction for barrel distortion mainly takes forward mapping and bilinear interpolation to correct distortion image in which the distortion is assumed as purely radial.

The first step is forward mapping of all pixels in the distorted image space (DIS) to the corrected image space (CIS) as

$$\begin{aligned} x_c &= x_d * \left(1+k_1 * r_d^2 + k_2 * r_d^4 + \cdots\right) \\ y_c &= y_d * \left(1+k_1 * r_d^2 + k_2 * r_d^4 + \cdots\right) \end{aligned} \tag{1}$$

Here the tangential distortion is ignored and high-order terms are omitted. (x_d, y_d) represent the distortion point

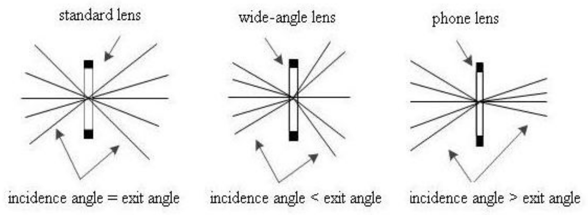
standard lens
wide-angle lens
phone lens
incidence angle = exit angle
incidence angle < exit angle
incidence angle > exit angle

Fig. 1. Schematic diagram of lens distortion.

coordinate in DIS, and (x_c, y_c) represent the corrected point coordinate in CIS. k_1, k_2 are distortion coefficients associated to lens which are adjusted according to actual correction effect. r_d^2 is the distance between distortion point and distortion center as

$$r_d^2 = (x_d - x_0)^2 + (y_d - y_0)^2 \tag{2}$$

(x_0, y_0) is the center of distortion image.

The second step is bilinear interpolation which makes corrected image appear smooth and improve its quality. The new values of pixel location (x_d, y_d) can be computed by linearly interpolating with the intensity values of the four neighboring pixels of (x_d, y_d). Let the integer parts of the coordinate values be represented by i and j as

$$\begin{aligned} i &= \lfloor x_d \rfloor \\ j &= \lfloor y_d \rfloor \end{aligned} \tag{3}$$

and the fractional parts of the coordinate values represented by u and v as

$$\begin{aligned} u &= x_d - i \\ v &= y_d - j \end{aligned} \tag{4}$$

Then the four neighboring pixels of (x_d, y_d) are (i, j), $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$ as illustrated in Fig. 2. Finally the intensity $dst(x, y)$ for pixel at (x_c, y_c) in CIS can be computed as

$$\begin{aligned} dst(x, y) &= a * src(i, j) + b * src(i, j+1) \\ &\quad + c * src(i+1, j) + d * src(i+1, j+1) \end{aligned} \tag{5}$$

Where $src(i, j)$, $src(i, j+1)$, $src(i+1, j)$ and $src(i+1, j+1)$ are the intensities for the four neighboring pixels at locations (i, j), $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$ respectively. The corresponding weighted coefficients of four pixels are as (6).

$$\begin{aligned} a &= (1-u) * (1-v) \\ b &= (1-u) * v \\ c &= u * (1-v) \\ d &= u * v \end{aligned} \tag{6}$$

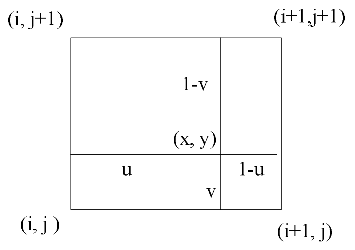

Fig. 2. Schematic drawing of bilinear interpolation.

III. Geometric Correction Implementaton on GPU

The implementation of geometric correction for barrel distortion on GPU relies on collaborative work of CPU and GPU. CPU is responsible for overall scheduling, synchronization issues and less calculation operation, and send GPU-call instructions to GPU. While GPU is responsible for main computation part of algorithm especially mass data calculations, floating point computing and image rendering, and finally sends processed image data to frame buffer for display. Therefore, the processing efficiency of geometric correction is dramatically improved in such a system that CPU and GPU play their respective advantages to rationally bear the workload of algorithm.

The most important technology in geometric correction implementation on GPU is called GPGPU, which utilizes the heterogeneous computing resources of GPU to complete a large scale of computations together with CPU. Classical GPGPU programmable pipeline is shown in Fig. 3. The shader processors in GPU consist of vertex shader processors and fragment shader processors: vertex shader processors deal with vertex data, while fragment shader processors deal with rasterized pixel data. The parallelized data to be calculated are written by CPU into the texture buffer via generating texture using glTexImage2D, read in GPU by texture mapping in texture cache and processed via dozens of (or even hundreds of) vertex shaders or fragment shaders with certain calculation procedures. The data outputted from fragment shaders are rendered to the texture buffer which can be mapped by the next process stage again, or to the off- screen frame buffers which can be read by CPU.

As for implementation of GPGPU in this paper, it is OpenGL and GLSL not CUDA that are used to realize GPU-based implementation of geometric correction for portability, because CUDA has some limits to graphics card and GPU chips. OpenGL is a large graphics API which can realize complex graphics effects. GLSL could be utilized to realize general-purpose computing on GPU. By using GLSL customized program can be written to replace the default graphics operations and realize customized graphics algorithms. Both OpenGL and GLSL have a platform-independent generality so that they are used to implement geometric correction for portability.

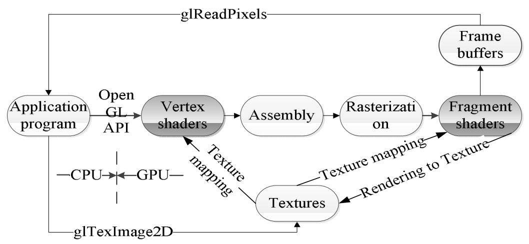

Fig. 3. Classical GPGPU programmable pipeline.

GPU-based implementation of geometric barrel distortion correction and the load distribution between CPU and GPU are arranged as Fig. 4. CPU takes charge of sending GPU-call instructions to GPU for parallel processing. GPU undertakes the whole calculations of geometric correction for barrel distortion with black shadow filling. The implementation starts from a series of initializations which consists of GLUT library (an important library of OpenGL), GLEW library (the extension library of OpenGL), and rendering environments. The second part sets texture format, reads distortion image and transfer the data to GPU by generating texture, and uses glTexture2D to bind image data to texture0.

The third part actives texture0, reads it from the external memory to GPU, and enables shader for texture rendering. The next is to pass some variable values like distortion coefficients (K1 and K2) into shader, and draw a quadrerilateral of image-size for vertex mapping. As for vertex shader, it only takes the mapping of vertex coordinates and texture coordinates. In order to process every pixel in fragment shader, rasterization is to rasterize all pixels which belong to the drawing area. As for fragment shader, it takes forward mapping and bilinear interpolation for three color channels (RGB) of every pixel value by using uniform value of distortion coefficients (K1 and K2). The outputted data are rendered back to texture1 for displaying the correction image in frame buffer. If the correction image don’t achieve good result, the variable values can be changed by calling glutKeyboardFunc and the distortion image can be corrected again with calling glutPostRedisplay until the corrected image meets the requirements.

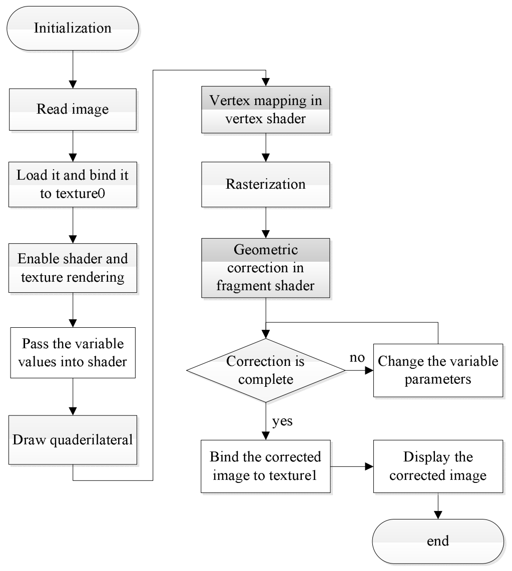
Initialization
Read image
Load it and bind it to texture0
Enable shader and texture rendering
Pass the variable values into shader
Draw quaderilateral
Vertex mapping in vertex shader
Rasterization
Geometric correction in fragment shader
Correction is complete
no
Change the variable parameters
yes
Bind the corrected image to texture1
Display the corrected image
end

Fig. 4. GPU-based implementation flow of geometric barrel distortion correction.

IV. Experiments and Analysis

For the performance evaluation of GPU-based implementation of geometric barrel distortion correction, the platform is using OpenGL 2.0 in visual studio 2008 on NIVIDIA GTX 650 that contains 384 shader processors with 80 GB/s memory bandwidth. While the platform of CPU-based implementation is using C language in visual studio 2008 on Pentium Dual-Core 3.19 GHz CPU.

Three groups of experimental images are done with the pixels of 500*335, 756*429 and 1400*850 which are shown in Fig. 5. The images in the right column are correction images, while the images in the left column are distortion images. We can see that the buildings in the first group, the ceiling in second group and the doors in the third group apparently show barrel-type distortion in the left column, while the final result in the right column are respectively corrected. The final distortion coefficients of three images are shown in table I. K1 and K2 are respectively the second-order and the fourth-order distortion coefficients in (1). The second-order distortion coefficient is negative value, while the fourth-order distortion coefficient is positive value. They can be adjusted by keyboard keys to meet the requirements.

In order to show acceleration effect of GPU, we measure the average execution time for three distortion images. The time consumption comparison between the implementation on GPU and the implementation completely on CPU is shown in table II. In the three experiments, the full execution time is all reduced greatly by using GPU to accelerate the data calculations. The full execution performance can achieve a speedup more than 190 times at most and 50 times at least. What's more, the speedup of the third experiment is much more than the first experiment. It is because the larger the image is, the more the quantity of calculation is, and the utilization ratio of the hardware computing resources can be more highly utilized.

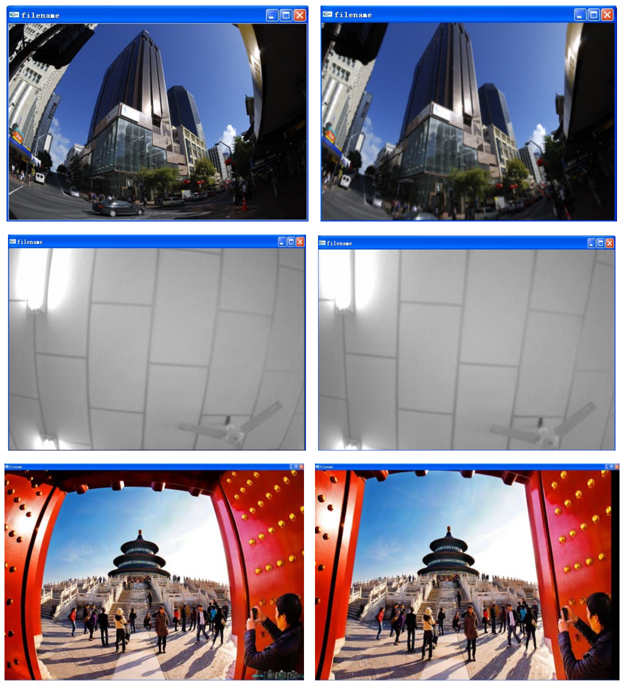

Fig. 5. Example images used in experiments (the top are used in experiment 1, the middle are used in experiment 2 and the below are used in experiment 3)

TABLE I. Distortion Coefficients of Three Experiments

	K 1	K 2
Experiment 1	-0.955	0.255
Experiment 2	-0.77	0.375
Experiment 3	-0.685	0.435

TABLE II. Time Consumption Comparison between CPU and GPU

	CPU time (ms)	GPU time (ms)	CPU time /GPU time
Experiment 1	27.243	0.536	50.83
Experiment 2	51.335	0.604	84.95
Experiment 3	197.307	1.038	190.08

V. Conclusion

Geometric barrel distortion correction is practical and effective. However, it can't meet the demand of high-speed processing in real-time application fields when it is completely implemented on CPU. This paper provides an GPU-based implementation of geometric correction, which fully utilizes special parallel computing resources in GPU hardware to accelerate the algorithm. It is realized with OpenGL and GLSL to meet the need of portability because both of OpenGL andGLSL have a platform-independent generality. Three group images of different sizes are experimented to prove the acceleration effects of geometric correction implemented on GPU. The experiment results show that the execution performance of the implementation on GPU is over 190 times speedup of that completely on CPU at most and 50 times at least, and can meet the requirements of real time.

Acknowledgment

This work was supported partly by Hunan Science and Technology Project (2013GK3006).

References

[1] A. Sunneberg, M. Giger, L. Kern, C. Noll, K. Stuby, K. B. Weber, and A. L. Blum. "How reliable is determination of ulcer size by endoscopy", *Brit. Med. J,* vol. 24, pp. 1322–1324, 1979.

[2] C. Margulies, B. Krevsky, and M. F. Catalano. "How accurate are endoscopic estimates of size," *Gastroint. Endoscopy*, vol. 40, pp. 174–177, 1944.

[3] Liu Yaqiang, Chen Wenyi. "A correction method of barrel distortion image", Journal of Xi'an university of posts and telecommunications, vol. 17, 2012(3): 27-36.

[4] Feng Dengguo, Zhang Yang, and Zhang Yuqing. "Survey of information security risk assesmrnt", Journal of China institute of communications, vol. 25, 2004(7): 10-18.

[5] Qiu Wei. "Telecommunication network security risk assesment studies", Telecommunication network technology, 2009, 3(3): 1-5.

[6] Ai Lili, Yuan Feng and Ding Zhenliang. "Further study on radial distortion model for photographic objective", vol. 28, 2008(10): 1930-1933.