Calculation of liquid crystal wave-front generation atmospheric turbulence simulator based on GPU

Kong Yue[1], Xu Xiping[1], Ni Xiaolong[1,2]

(1. School of Opto-Electronic Engineering,Changchun University of Science and Technology, Changchun 130022, China;
2. Fundammontal Science on Space-Ground Laser Communication Technology Laboratory, Changchun University of Science and Technology, Changchun 130022, China)

Abstract: In order to achieve the purpose of real-time atmosphere turbulence simulation by used liquid crystal spatial light modulator, A new computing method of liquid crystal atmospheric turbulence simulator real-time wave-front generation based on GPU has been proposed. According to the characteristics of liquid crystal turbulence simulator, which is high resolution, high precision, The Compute Unified Device Architecture was discussed in this paper. Furthermore, a wave-front generation model based on GPU was established with parallel optimization and share memory optimization. Finally, the experimental results by used CPU and GPU wave generation was given. The result shows that: the consumed time by proposed method is less than 2 ms for a Zernike polynomial with 231 wave-front values in resolution of 256×256, which is two orders of magnitude less than that of CPU, fully meets the real-time wave-front generation requirements.

Key words: graph processing unit; compute unified device architecture; liquid crystal; wave-front calculation

CLC number: TP332.3 **Document code:** A **Article ID:** 1007-2276(2014)09-3061-05

基于 GPU 的液晶大气湍流模拟器的波面生成计算

孔 悦[1],徐熙平[1],倪小龙[1,2]

(1. 长春理工大学 光电工程学院,吉林 长春 130022;
2. 长春理工大学 空地激光通信技术国防重点学科实验室,吉林 长春 130022)

摘 要: 提出了一种基于 GPU 的液晶大气湍流模拟器实时波面生成的计算方法,为了让液晶空间光调制器进行大气湍流类比。依据液晶湍流模拟器高分辨率、高精度的特性讨论 CUDA 的算法。此外,建立一种基于 GPU 波面生成的模型并进一步对其优化。最后给出使用 CPU 和 GPU 后的结果并进行类比。结果表明:采用 231 项 Zernike 系数生成分辨率为 256×256 的波前所需时间少于 2 ms,与传统的采用 CPU 生成的方法相比速度提升两个量级,满足实时波面生成的要求。

关键词: 图形处理器; CUDA; 液晶; 湍流计算

收稿日期:2014-01-05; **修订日期**:2014-02-10

作者简介:孔悦(1988-),女,硕士生,主要从事精密测控技术与仪器方面的研究。Email:yuekong705@163.com

导师简介:徐熙平(1969-),男,教授,博士生导师,主要从事光电检测技术与质量控制方面的研究。Email:xxp@cust.edu.cn

0 Introduction

Atmospheric turbulence plays an important role in laser′s transmission. Light beam will occur random fluctuation, cause the beam drift, beam expension, the intensity fluctuation phenomenon, and poor beam quality. Moreover, with the development of astronomical imaging, laser communication, tracking and other high technology, and the influence arouses people more and more attention. However, only by means of field measurements, repeated experiments and so on, not only labourpower, material resources, financial resources are wasted by this way, but also it is very difficult to reflect turbulent changes in all kinds of weather conditions[1-3] completely and accurately. Therefore, We need a turbulence simulation device which can simulate atmosphere turbulence. At present, there are many methods of simulating atmospheric turbulence, such as atmospheric turbulence simulator based on liquid crystal, random phase plate and so on[4].

Liquid crystal device has the characteristics of high pixel density, high precision, phase modulation phase programming real-time control, especially it is suitable for high precision controllable turbulence simulation. For the liquid crystal light modulator of common 256×256 pixels, it requires 65 536 control signals, and it is difficult to achieve the real-time wave-front generation by means of traditional CPU calculation. This method makes wave-front repeatability of the system higher, and larger gap between the actual turbulence [5]. However, with the rapid development of GPU technology, current GPU possesses strong parallel operation capability and the floating-point operation capability. Besides, wave-front generation calculation is a typical large-quantity repeated calculation and it is suitable for the structure of GPU to make parallel optimization[6]. In this paper, a new calculation of the liquid crystal atmospheric turbulence simulator wave-front generation based on GPU is presented. First, select the top 231 Zernike model to calculate turbulence and simulation turbulence experiment and then better experimental results is obtained.

1 Turbulence wave-front generated by Zernike modes

As mentioned in reference 7, Any wave-front function $\varphi(x,y)$ within the circular aperture can be expanded in the form of Zernike polynomials[7].

$$\varphi(x,y)=\sum_{i=1}^{i=\infty} a_i \cdot z_i(x,y) \tag{1}$$

Where a_i is the i−th term Zernike mode coefficients, z_i is the first i Zernike mode. The foundation of atmospheric turbulence simulation is to generate the Zernike polynomial coefficients a_i, so that we can generate the wave-front obey the Kolmogorov turbulence theory. Zernike polynomial coefficients can be assumed to be zero mean Gaussian random variable, the number of Zernike polynomials patterns p is selected, assuming that A is the coefficient vector.

$$A=[a_1,a_2,\cdots,a_p]^{\mathrm{T}} \tag{2}$$

By Eq.(1) we can get

$$a_i=(\varphi,z_i) \tag{3}$$

By Eq.(3) and Kolmogorov turbulence theory, the covariance matrix C of coefficient A can be obtained.

$$\langle a_i a_i\rangle=c_{ij} \tag{4}$$

$$c=[c_{ij}]\quad i=1,2,\cdots,p,\ \mathrm{j}=1,2,\cdots,p \tag{5}$$

Where D is the diameter of the telescope, r_0 is the atmospheric correlation length, r_0 value reflects the intensity of the turbulence in the case of D determines, $(D/r_0)^{5/3}$ can be substituted into a specific value to reflect the turbulent correlation strength. Matrix C is the covariance matrix of coefficients A by the covariance matrix C we can get that the each Zernike modes isstatistically independent. To generate random turbulence wave-front, it must be expressed as a defined amount of a combination of random variance, thus by using Karhumen-Loeve transform and the singular value decomposition of the covariance matrix C we can get.

$$C=VSV^{\mathrm{T}} \tag{6}$$

Where S is a diagonal matrix. V is a unitary matrix, also the expansion coefficients of the Karhumen-

Loeve polynomials expansed by Zernike polynomials. Then we can get vector B with zero mean and covariance matrix is S Gaussian random.

$$A=VB \tag{7}$$

Where B is the coefficient of wave-front $\varphi(x,y)$ expensed by Karhumen-Loeve polynomial, then zero mean Gaussian random vectors A can be got. By Eq.(1), atmospheric turbulence according to Kolmogorov turbulence theory can be got[8–9]. The correspondence relationship between Zernike coefficients and the control voltage v of the liquid crystal spatial light modulator is:

$$V=Ra \tag{8}$$

Where V is the control voltage matrix with 65 536 rows, a is the Zernike coefficient matrix with 231 rows, R is the phase modulation value of each pixel matrix corresponding to the control voltage with total of 65 536 rows and 231 rows.

2 GPU compute unified device architecture

At present, Which adopts GPU as the main framework is NVIDIA′s CUDA (Compute Unified Device Architecture). it is parallel computing architecture, it makes GPU solve complex computing problems. Generally,a typical CUDA system includes CUDA instruction set architecture (ISA) and parallel GPU computing engine[10].

CUDA makes a new algorithm to use hardware resources provided by GPU, so it takes massive data computing application a more powerful calculation ability than CPU. Under the framework of CUDA, a program is divided into two parts: Host and Device.

The Host program is running on the CPU, and the Device program is running on GPU. Device program called "kernel". Usually Host application data will copy to the graphics card memory, when it is ready, then the display chip executes the Device client program. when compute finishes Host client program will result from the graphics memory retrieval[11]. The structure is shown in Fig.1.

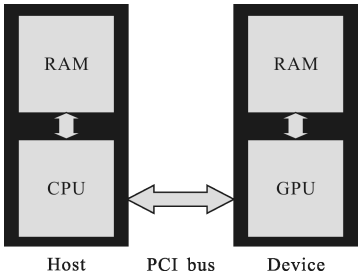

Fig.1 GPU compute unified device architecture

Compared with CPU , GPU is more suitable for intensive, parallel computing. In particular, GPU is very suitable to deal with those that can be expressed as parallel computing (parallel execution of the same program on multiple data) problem, generating of turbulence wave-front control signal is this calculation. so using GPU for liquid crystal turbulence simulator computing can greatly accelerate the speed of calculation.

2.1 GPU wave-front generating computing

According to Eq. (8) we learn that the key operation of the wave-front generation is matrix operation. Using GPU for the parallel operation optimization is the key to improve the turbulence generation rate. Under the framework of CUDA, GPU minimum operation unit is Thread, numbers of Thread can be composed of a Block. A Block Thread can access the same shared memory, and can be used for fast synchronized action. The number of Thread belongs to each Block is limited. However, a number of Block can form Grid. CUDA operation structure is shown in Fig.2.

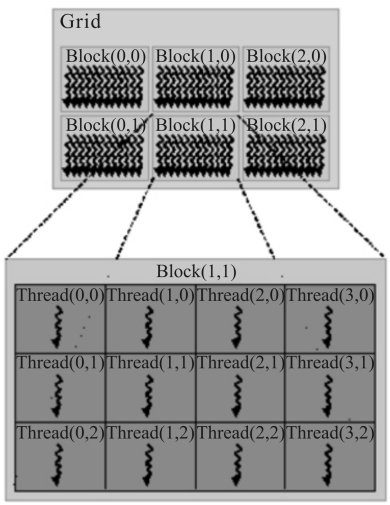

Fig.2 CUDA operation structure

Each Block corr esponds to a physical operation core, making full use of Block resources is the key for parallel optimization. Turbulence generating operation a total of 256×256 independent operation, in order to achieve maximum parallel optimization, we need to set the number of Thread as 256×256. The GPU Geforce GTX 580 has 1 536 stream processors, in order to take full advantage of GPU computing resources, we should make the number Block system more than 1 536. The number of thread in each block should be a multiple of the size of warp. Warp is a minimum single instruction multiple threads (SIMT) unit. It makes 32 parallel threads as a group to create, manage, schedule and execute threads. Then set the Thread number on 32, and we can calculate the required Block number 2 048.

Because the CPU and GPU communication bandwidth is low, in order to reduce the communication times between CPU and GPU, we need to stores Eq.(8) in the shared memory for optimization. To sum up, the driving voltage computing progam of each pixel is shown below.

```
__global__ void Matrix_Mul (int*md, int*nd, int*pd, int width)
{int bx, by, tx, ty;
bx = blockIdx.x;
by = blockIdx.y;
tx = threadIdx.x;
ty = threadIdx.y;
int mulResult=0;
for ( int i = 0; i<gridDim.x; ++i ){
__shared__ int d_m[TILE_WIDTH][TILE_WIDTH];
__shared__ int d_n[TILE_WIDTH][TILE_WIDTH];
d_m [ty][tx] =* (md+(by*blockDim.y+ty)*width+i*blockDim.x+tx);
d_n[ty][tx] = *(nd + (i * blockDim.y + ty) * width + bx * blockDim.x + tx);
__syncthreads();
for ( int j=0; j<blockDim.x; ++j ){
mulResult += d_m[ty][j]*d_n[j][tx];}
__syncthreads();}
pd [(by*blockDim.y+ty)*w idth+bx* blockDim.x+tx]= mulResult;}
```

2.2 Desgin of turbulence simulation software

The turbulence generation also needs to add the CUDA computing program. cu file to the turbulence simulation MFC program. The flowchat is shown in Fig. 3.

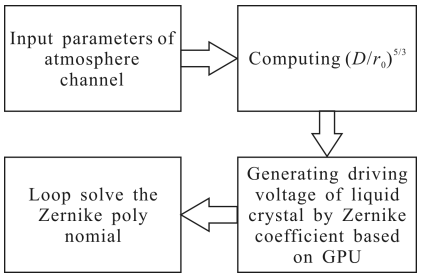

Fig.3 Turbulence simulation flowchart

When the program ru n s, first it needs to enter the horizontal wind speed, vertical wind speed, receive port diameter, wavelength, wave height, ground elevation, transmission distance, transmission link zenith angle and other parameters, then calculate the value of $(D/r_0)^{5/3}$. After that, bring them into a Zernike polynomial to get the Zernike coefficient. Finally, the control voltage of liquid turbulence simulator is generated by GPU.

3 Experiments and results

Liquid crystal spatial light modulator uses the HSP256-635 of BNS company. It has 256 pixels × 256 pixels. GPU using the Geforce GTX680 chip of NVIDIA company, which has 1 536 stream processors, the core frequency is 1 006 Hz, memory frequency is 6 008 Hz. CPU uses the i7-3930k processor of Intel company, the core frequency 3.2 GHz, 16 G system memory. Liquid crystal turbulence simulator test experimental light path is shown in Fig.4.

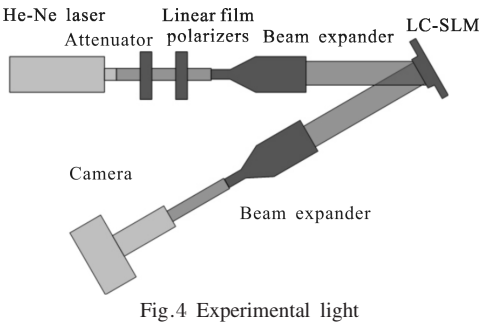

Fig.4 Experimental light

He－Ne laser generates 633 nm laser beam, first through the filter to make the beam intensity adjustment as the sensor′s range. And then through the linear film polarizer, the beam adjustment for single direction of linearly polarized light, so that the liquid crystal spatial light modulator can be used. Then the beam expander expanse the beam to match the L－C SLM. The LC－SLM generates turbulence wave－front. Finally, through the beam expander, the beam matches the diameter of the sensor to real-time display, we can see the turbulence wave-front generated by the simulator. During the experiment, using the CPU and the GPU to generate wave-front distortion by the same Zernike coefficients, the simulation results is shown in Fig.5, the r_0 here is 5 cm, the C_n^2 is 35×10^{-17} and the Greenwood frequency f_g is 6.48.

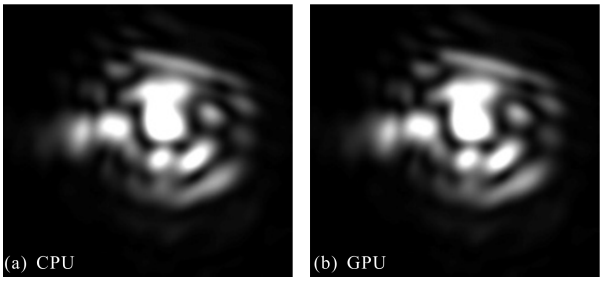

Fig.5 Contrast diagram of wave-front generated by CPU and GPU

According to Fig.5, we know that using CPU and GPU can generate the same wave-front. While using GPU to generates distortion wave-front needs 2.3 ms, which is 213 ms using CPU. Wave-front generate CPU the same time required for 215 ms. So using GPU to generate distortion wave-front can decrease time by two orders of magnitude less than CPU. It fully meets the 10 ms requirements for real-time simulation of atmospheric turbulence.

4 Conclusion

In this paper, a new calculation of the liquid crystal atmospheric turbulence simulator wave-front generation based on GPU is presented. According to experiments and analysis, by using GPU compared with the traditional CPU in the same precision, the GPU requires for 2.3 ms, two orders of magnitude less than CPU. This result greatly increases the wave-front generation velocity of the system. This method meets the time requirements of real-time wave-front generation computational.

References:

[1] Chen Chunyi, Yang Huamin, Jiang Huilin, et al. Research progress of mitigation technologies of turbulence effects in atmospheric optical communication[J]. ***Acta Armamentar* II**, 2009, 30(6): 779－788. (in Chinese)

[2] Shen Yong, Liu Jianguo, Zeng Zongyong, et al. Performance testing of atmospheric turbulence simulator [J]. ***Journal of Atmospheric and Environmental Optics***, 2011, 6(3): 231－234. (in Chinese)

[3] Li Dayu, Hu Lifa, Cao Zhaoliang, et al. Liquid crystal atmosphere turbulence simulator [J]. ***Acta Photonica Sinica***, 2006, 35(12): 1960－1963. (in Chinese)

[4] Gan Xinji, Guo Jin, Fu Youyu, et al. Simulating turbulence method of the atmosphere scene simulator[J]. ***Semiconductor Optoelectrctronics***, 2006, 27(6): 764－766. (in Chinese)

[5] Cai Dongmei, Yao Jun, Jiang Wenhan, et al. Performance of liquid-crystal spatial light mudulator using for wave-front correction[J]. ***Acta Photonica Sinica***, 2009, 29(2): 285－290. (in Chinese)

[6] Li Dayu, Hu Lifa, Mu Quanquan, et al. Wave-front calculation of liquid crystal adaptive optics based on CUDA [J]. ***Optics and Precision Engineering***, 2010, 18(4): 848－853. (in Chinese)

[7] Li Jingzhen. Handbook of Optics [M]. Shanxi: Shaanxi Science & Technology Press, 2010: 7. (in Chinese)

[8] Duan Jin, Wang Xize, Jing Wenbo, et al. The atmosphere turbulence simulation based on Zernike polynomial [J]. ***Journal of Changchun University of Science and Technology* (*Natural Science Edition*)**, 2010, 33(3): 60－62. (in Chinese)

[9] Han Liqiang, Wang Qi, Shida Katsunori. Outage probability of free space optical communication over atmospheric turbulence[J]. ***Infrared and Laser Engineering***, 2010, 39(4): 660－663. (in Chinese)

[10] Wang Jiang′an, Zhao Yingjun, Chen Dong, et al. Effects of turbulence sizes on the error rate of atmospheric laser communication system [J]. ***Infrared and Laser Engineering***, 2009, 38(4): 655－659. (in Chinese)

[11] Lv Jie, Zhang Tianxu, Zhang Biyin. Applications of MPI parallel-computing on image processing [J]. ***Infrared and Laser Engineering***, 2004, 33(5): 496－499. (in Chinese)