CUDA-OpenGL GPU-based real time beam tracking by MEMS phase SLM

Tang, Chin-I, Deng, Xianyue, Takashima, Yuzuru

Chin-I Tang, Xianyue Deng, Yuzuru Takashima, "CUDA-OpenGL GPU-based real time beam tracking by MEMS phase SLM," Proc. SPIE 12014, Emerging Digital Micromirror Device Based Systems and Applications XIV, 1201408 (1 March 2022); doi: 10.1117/12.2608031

SPIE.

Event: SPIE OPTO, 2022, San Francisco, California, United States

CUDA-OpenGL GPU-based real time beam tracking by MEMS phase SLM

Chin-I Tang, Xianyue Deng, and Yuzuru Takashima*
James C. Wyant College of Optical Sciences, University of Arizona, 1630 E. University Blvd., Tucson, AZ 85721, USA

ABSTRACT

CUDA-OpenGL interoperability enables to drastically reduce the computational time for CGH calculation and display on Spatial Light Modulators via HDMI display channel. The fast calculation method enables on-the-fly diffractive beam steering by Micro Electro Mechanical System based phase light modulator with YOLOv4-tiny model based object recognition to do AI-based dynamic beam tracking in order to trace the object of interest.

Keywords: GPU, CUDA-OpenGL, Computer Generated Hologram, LiDAR, Bean steering, Phase Light Modulator, MEMS

1. INTRODUCTION

The computation speed of Computer Generated Hologram (CGH) for beam steering is critical for application of spatial light modulators (SLMs) for LiDAR. Especially, capability of steering beam in a random-access manner based on an object recognition opens up a path way to a Foveated lidar by which laser beam scans only the region of interest, not scanning the entire FOV by traditional raster scanning. Recently, it is becoming more popular to utilize Graphics Processing Unit (GPU) to speed up the calculation procedure of CGH. Especially, GPU calculation is suitable for non-iterative CGH calculation because of the independency of each pixel and the low complexity of the computation.

We present a pixel-wise parallel processing by using CUDA-OpenGL, instead of CPU, to reduce the time required for CGH calculation and to display. CUDA is a parallel computing platform created by Nvidia which makes developers easier to access GPU resources by organizing threads, blocks, and grids for CUDA kernel [1]. OpenGL is a commonly used Application Protocol Interface (API) for accelerating 2D or 3D graphics rendering by GPU which is widely applied in various areas, such as Game development, medical imaging, and modeling [2]. CUDA-OpenGL interoperability combines the advantages of GPU-based calculation and GPU-accelerated display via mapping buffers from OpenGL to CUDA memory, so that OpenGL could access and display the computation results directly from CUDA without transferring data between CPU and GPU [1]. Additionally, we combine OpenCV and deploy YOLOv4-tiny model for object detection and recognition through the frame captured by a USB camera [3]. The object will be framed in a rectangle region of interest with its label along once it is detected by the model, then the coordinate of the rectangle will be assigned to CUDA CGH calculation, corresponding CGH will then be displayed by OpenGL, the rectangular area will be scanned in real time at the maximum speed of current generation of Texas Instrument Phase Light Modulator (PLM) [4].

In this paper, we implement and demonstrated adaptive laser beam steering in conjunction with real-time AI-based object recognition. First object position and extent are recognized by deep learning model (YOLOv4-tiny). Beam steering pattern to cover the extent of the object is calculated and CGHs are generated on-the-fly by CUDA-Open GL to steer beam towards the direction of the object. The speed of object tracking, and beam steering is matched to the state-of-the-art of MEMS phase SLM, 180 points/s. In section 2, we address single point CGH calculation algorithm, implementation of it to GPU accelerated computational architecture, along with benchmarking of computation speed. In section 3, the integration of object tracking, and beam steering is addressed. In section 4, real time multipoint and variable beam ratio beam steering is addressed. In section 5, limitation and scalability of the approach for lidar with adaptive laser beam steering are addressed.

*ytakashima@optics.arizona.edu

Emerging Digital Micromirror Device Based Systems and Applications XIV,
edited by John Ehmke, Benjamin L. Lee, Proc. of SPIE Vol. 12014,
1201408 · © 2022 SPIE · 0277-786X · doi: 10.1117/12.2608031

2. LASER BEAM STEERING WITH CGH

To steer beam, phase holograms are encoded to the Phase Light Modulator (PLM) such as Liquid Crystal on Silicon (LCoS), and Micro Electro Mechanical (MEMS) spatial light modulator (SLM) that modulates the phase of the incident beam. Figure 1 schematically shows beam steering by SLMs.

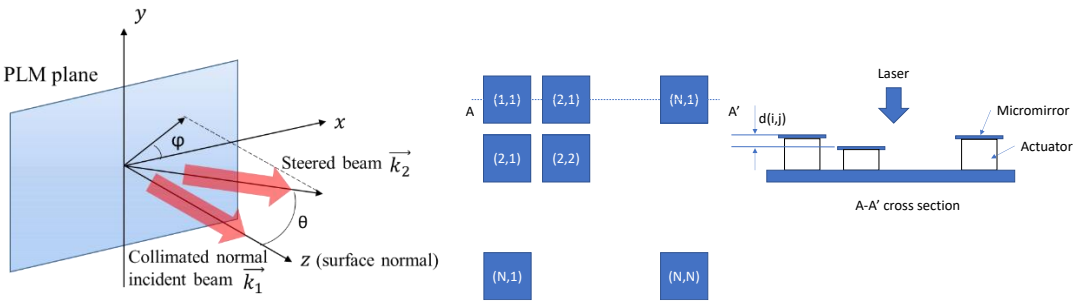


Figure 1. (a) Schematic of laser beam steering by PLM. Vector k_2 represents the direction of the steered beam. Vector k_1 is the incident beam direction, collimated normal incident in our case. (b) The pixel array arrangement of PLM. Each pixel has its coordinate (x_h, y_h).

When the plane wave is incident on the phase SLM, the direction of the oncoming wave is deviated if the phase change linearly, which is a phase tilt. The phase tilt results in the change of propagation of the plane wave, or lateral shift of the focal spot at the lens' focal plane when focusing optics is incorporated. The two-dimensional prism phase is defined in Equation (1).

$$\emptyset_{prism}(x_h, y_h) = (\alpha(\Delta x x_h + \Delta y y_h))\ mod\ 2\pi \quad (1)$$

α is a coefficient that depends on the wavelength and imaging characteristics. The distance of lateral shift on the image plane is $\sqrt{\Delta x^2 + \Delta y^2}$. $\emptyset_{prism}(x_h, y_h)$ represents the phase of the pixel located (x_h, y_h).

3. CUDA-OPENGL INTEROPERABILITY IMPLEMENTATION OF REAL-TIME CGH BEAM STEERING

Due to the large amount of independent calculation as well as the low complexity of the computation, CGH processing task for beam steering application is very suitable to compute using GPU. Besides, rendering is also handled by GPU so that we can minimize the data transfer between CPU and GPU and reduce the time cost effectively.

CUDA is a parallel programming platform created by Nvidia which makes developers easier to access GPU resources by organizing threads, blocks, and grids for CUDA kernel (Fig. 2). A grid is composed of a set of blocks, a block is composed of a set of threads. One thread is a unit of parallel processing in GPU, handles the calculation of one pixel. The functions to run on the GPU are CUDA kernels, which are called by thread. Programmers can write their own kernels depends on their need. The core concept of CUDA is processing threads in parallel to approach faster computational speed. OpenGL is a commonly used API for accelerating 2D or 3D graphics rendering by GPU which is widely applied in various areas, such as Game development, medical imaging, and modeling.

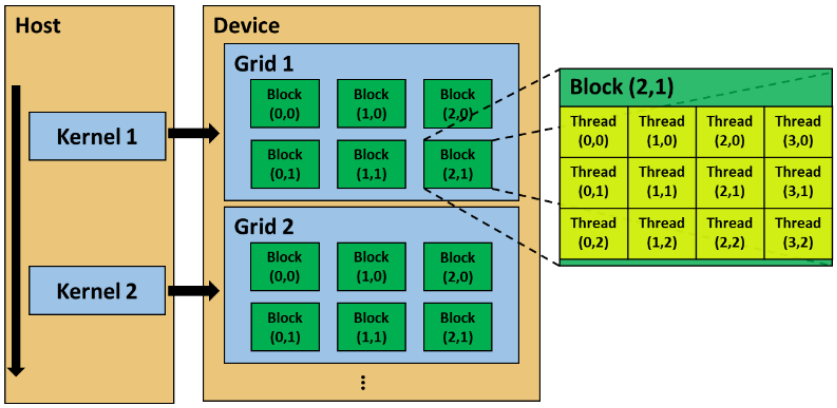


Figure 2. The structure of GPU. A grid is composed of a set of blocks, a block is composed of a set of threads. One thread is a unit of parallel processing in GPU, handles the calculation of one pixel. The functions to run on the GPU are CUDA kernels, which are called by thread. Programmers can write their own kernels depends on their need.

CUDA-OpenGL interoperability combines the advantages of GPU-based calculation and GPU-accelerated display via sharing OpenGL resources with CUDA, and mapping buffer object from OpenGL to CUDA memory. In this way, CUDA is responsible for CGH computing, OpenGL could render the results from buffer object directly. The algorithm process is described below, and the flowchart is shown in Fig. 3.

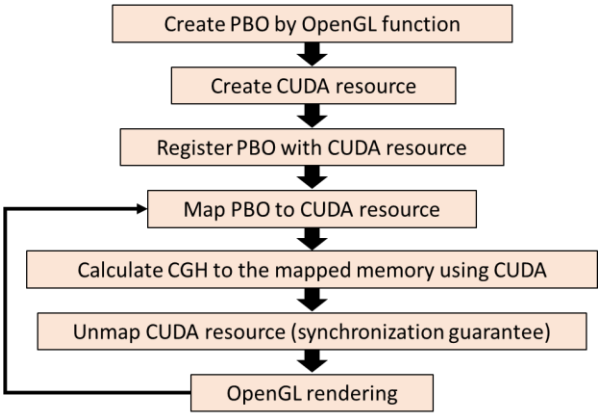
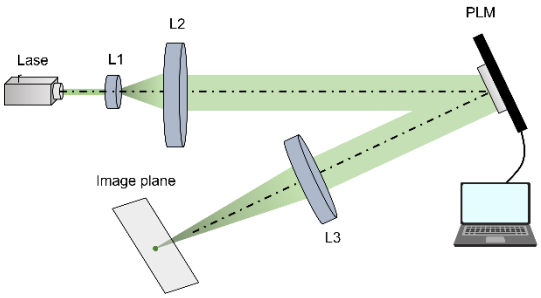


Figure 3. The flow chart of CUDA-OpenGL interoperability. First, Create pixel buffer object (PBO) and graphic resource. Second, register the PBO for access by CUDA then map the graphics resource with CUDA. Use CUDA to write CGH image to the mapped memory, and unmap graphic resource. Last, render the results by OpenGL.

4. EXPERIMENT SETUP AND TI-PLM BEAM STEERING RESULTS

In this work, we utilize CUDA-OpenGL interoperability to realize GPU processing. The maximum number of threads per block is 1024, thus we allocate the size in two dimensions, (32, 32), and implement Eqn. (1) to CUDA kernel. As for PLM, Texas Instruments Phase Light Modulator (TI-PLM) was used as SLM [4]. The TI-PLM has a maximum phase modulation depth of 2pi at wavelength of λ = 532nm. Over 60Hz of HDMI channel, 180Hz of CGH display is feasible via RGB encoded mode, i.e., three different CGHs are stacked together to form a colored CGH and send to PLM via 60Hz HDMI. Figure 4 shows optical set up of the experiment. A 10mW, 532nm CW DPSS laser (Thorlabs DJ532-10) is expanded by a beam expander (L1 and L2) and illuminates the 0.45 inch TI-PLM, connected to host PC. The hardware specification is listed in Fig. 4. The diffraction pattern is observed at a screen placed at the back focal place of focusing lens (L3) f=300mm.



Hardware information	
CPU	Intel(R) Core(TM) i7-10750H
OS	Windows 10 Home 64 bit
Clock frequency	2.60 GHz
Memory capacity	16 GB
GPU	NVIDIA GeForce GTX 1650 Ti
CUDA version	11.4

Figure 4. Our experimental setup and laptop information. 532nm CW laser passing through the beam expander (L1 and L2) is reflected by phase light modulator connected with a Laptop, and passing through the focusing lens (L3) with f=300mm, focusing on the image plane.

Performance improvement: Raster scanning

To benchmark the performance of CGH calculation, we compared number of scanning points per seconds for a) CPU and b) GPU-based approach by scanning through 31x31 (961) points on the focal plane (Fig. 5). In raster scanning task, each CGH is calculated and sent to PLM using HDMI in real-time. We test the CGH processing time cost on both CPU and GPU by scanning through 31x31 (961) points on the focal plane. As Table 1 shows, CPU-based approach only performs 0.579 point per second because it iterates from pixel to pixel, which is far from real-time application. As for GPU part, we apply CUDA-OpenGL structure mentioned earlier to process each pixel in parallel. 961 points are scanned though within 5.4 seconds, resulting in 180 points per second, which achieve the upper speed limitation of current generation of TI-PLM.

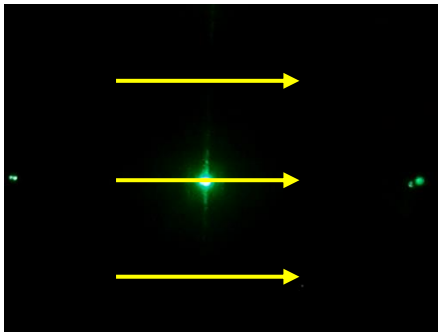


Figure 5. Raster scanning test implementing on both CPU and GPU. Scan through a rectangle with 31x31 points.

Table 1. Experiment results comparison between CPU and GPU. CPU only performs 0.579 points per second, GPU performs 180 points per second, 310.6 times faster than CPU.

Performance	
CPU	0.579 pts/second
GPU	180 pts/second
Speedup	310.9X

Dynamic beam steering

After testing the raster scanning speed of our proposed approach, we utilize OpenCV to deploy deep learning model, YOLOv4-tiny, for object detection and recognition through the frame captured by USB camera. The object will be framed in a rectangle region of interest with its label along once it is detected by the model, then the coordinate of the rectangle will be assigned to CGH calculation which is mentioned in section 3, the corresponding CGH will then be rendered by OpenGL, the rectangular area will be scanned through in real time at the maximum speed of current generation of Texas Instrument PLM.

Figure 6. (a) A pedestrian walking across the road is captured by video camera. (b) CGH is calculated on-the-fly by to scan the angular extent and position of the pedestrian (Movie is available on archived presentation).

5. DISCUSSION

In our experiment results, the current frame rate is limited by PLM but the CGH calculation speed for our laptop is 144 frames per sec. The newer generation 0.67 inch PLM with gray scale RGB encoding can benefit the fast speed we proposed. Moreover, further enhancement of frame rate can be feasible by calculating and displaying CGH to steer beam into multiple points/frame.

6. CONCLUSION

We present a deterministic CGH beam steering method using CUDA-OpenGL interoperability enable to reduce the required time for CGH generation and display. Our proposed approach for CGH calculation and display on GTX 1650Ti, i7-10750H@2.6GHz performs faster than the display upper limit of current PLM, resulting the beam steering speed of 180 points per second, and could be further accelerated by either improving the performance of PLM or applying our program on a better GPU. In addition, we successfully combine YOLOv4-tiny model with our GPU-based CGH beam steering, allowing our system to do AI-based dynamic beam tracking in order to trace the object of interest. Last, we also demonstrated multi-beam steering which results in 574 points per second, 3.19 times more efficient than single beam steering. Moreover, this multi-beam generation approach also capable of beam energy modulation, allowing to do real time multipoint and variable beam ratio beam steering.

7. ACKNOWLEDGMENTS

We acknowledge generous support by Semiconductor Research Corporation, Texas Instruments, Mitsubishi Electric, and Tech Launch Arizona on this study of DMD based lidar.

REFERENCES

[1] NVIDIA, “CUDA TOOLKIT DOCUMENTATION”, 23 November 2021, < https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html >
[2] OPENGL, “The Industry's Foundation for High Performance Graphics”, 04 June 2021, < https://www.opengl.org//>
[3] OPENCV, “OpenCV: Open Source Computer Vision Library”, 06 Jan. 2021, < https://github.com/opencv/opencv>
[4] Bartlett, Terry A., William C. McDonald, and James N. Hall. "Adapting Texas Instruments DLP technology to demonstrate a phase spatial light modulator." Emerging Digital Micromirror Device Based Systems and Applications XI. Vol. 10932. International Society for Optics and Photonics, 2019.