Optik 126 (2015) 1067–1071

Contents lists available at ScienceDirect

Optik

journal homepage: www.elsevier.de/ijleo

Modification of common Fourier computer generated hologram's representation methods from sequential to parallel computing

Ghaith Makey, Moustafa Sayem El-Daher*, Kanj Al-Shufi

Higher Institute for Laser Research and Applications, Damascus University, Damascus, Syria

ARTICLE INFO

Article history:
Received 2 February 2014
Accepted 12 February 2015

Keywords:
Computer generated holography (CGH)
Parallel processing
Fourier hologram

ABSTRACT

The continuous improvement of GPUs for general purpose parallel computing has made it tempting to modify many physical computational techniques from sequential to parallel computation. By adding the capability of using CUDA-enabled GPU to MATLAB, a friendlier programming environment became available to physicists and researchers for implementing GPU parallel computing in their scientific calculations. This paper is dedicated to modify four methods of Fourier computer generated holography to utilize parallel computation of GPU. discussion of benefits, limitations and a speed-up comparison is presented.

1. Introduction

GPUs have pushed the parallel computing implementation to new levels; many algorithms and calculations were modified from serial to parallel, where the main purpose for this modification is speed-up. Actually whether we need this speed-up or not, parallel computations will keep on spreading and we eventually have to understand its algorithms to deal efficiently with this type of computing.

When referring to speed-up, the computer generated holograms (CGH) field looks like a good candidate which can benefit tremendously because of the huge amount of calculations needed for its computation [1]. In addition to that, parallel computing of optical problems like holography makes sense because of the parallel physical nature behind the optical problems that deal with light wavefronts.

The main problem of implementing GPU parallel computing was the difficulty of its programming for many scientists even when using simplified programming languages like CUDA, but with the continuous improvement of MATLAB's GPU integration this is becoming less of a problem.

In this work we will explore the modification of four common methods of Fourier CGH to parallel computation within MATLAB environment. The methods are: Kinoform (with and without diffuser stage), Detour phase (with and without diffuser stage), Lee and Burckhardt [1,2].

2. Related work

Applying GPUs general purpose parallel computing in many research areas was the subject of many publications in a variety of scientific fields; we will concentrate our literature review only to those related to CGH's GPU parallel computations and those that used MATLAB for GPU parallel computing.

In 2006, Masuda et al. [3] used Shader language and graphic API to general in-line CGH by summing interferences patterns [4].

In 2009, Pan et al. [5] developed split look-up tables (S-LUT) and implemented it on GPU to compute full parallax CGH; in this work CUDA was used.

In 2010, Zschau et al. [6] introduced SeeReal's Sub-Hologram-technology where almost all calculations were done on GPU based on Microsoft Direct3D 9. In the same year Shimobaba et al. [7] compared AMD and NVIDIA GPUs in CGH generation, and the generation method was the same as in reference [3].

In 2012, Reid et al. [8] provided a study on the challenges in using GPUs for the reconstruction of digital hologram images where the hologram itself was recorded by digital camera, their work was done by CUDA and they got impressive speed-up. In the same year Song et al. [9] used multi GPUs to speed up generation of CGH; they used CUDA and OpenMP, and CGH was generated by same method as [3] while implementing look-up table (LUT [10]).

In 2013, we [11] provided a method of using both CUDA and MATLAB in a hyper approach to speed up the calculations of Detour

* Corresponding author. Tel.: +963 11 33924790; fax: +963 11 211 9896.
E-mail address: eldaherm@scs-net.org (M.S. El-Daher).

http://dx.doi.org/10.1016/j.ijleo.2015.02.076
0030-4026/

phase; such an approach requires knowledge in CUDA programming.

There are papers focused on speeding up other applications, like image processing. Those methods were focused on the use of CUDA kernels within MATLAB environment [12–14] (note that in our approach there is no need to write CUDA kernels). Regarding the same applications there are papers that used MATLAB as platform for sequential calculations to be compared with CUDA parallel computations [15,16].

Besides MATLAB's Parallel Computing Toolbox (PCT), there are papers (not related to CGH topics) that used third party toolboxes for MATLAB GPU computing like Jacket and GPUmat [17,18]. In 2011 Zhang et al. [19] provided a comparison between Jacket, GPUmat and PCT in GPU computing.

This paper differs from the previous work in both: its focus in Fourier holograms, and its approach to use MATLAB-only programming to access the parallel power of GPU (NVIDIA TESLA C2050).

3. Fourier holograms

To solve the representation problem of the complex field generated in Fourier hologram plane, there are a number of methods which are commonly used [1,2]:

1. Kinoforms which depend only on the phases of Fourier coefficients as they are considered to have the majority of information about source image (object). By introducing a diffuser stage (adding random phases) to source image, Kinoforms can generate quiet accurate image for any object but paying the price of having speckles.
2. Detour phase, as regarded by Goodman [2]: "The oldest and perhaps the best known method for creating holograms from computed complex fields", Detour phase transfers every Fourier coefficient of the complex field in hologram plane to a cell with an aperture, the area of this aperture is proportional to the amplitude of the coefficient and the aperture's position within the cell is proportional to the phase of this coefficient. By practice and when applying this method on Spatial Light Modulators the cell size will decrease the ability to represent the amplitude of Fourier coefficients; to overcome that, adding diffusing stage like in Kinoforms might make sense [20,21].
3. Lee: In this method each Fourier coefficient represented by a cell contains 4 gray-level subcells; the first represents the real and positive part of the coefficient, the second the imaginary and positive part, the third the real and negative part and the last the imaginary and negative part.
4. Burckhardt: In this method each cell contains 3 subcells with 3 gray-level phases one at 0°, next at 120° and the last at 240°.

4. Hardware specifications

The platform used is Windows 7. The CPU used is AMD Phenom^{TM} 9850 Processor 2.51 GHz. All baseline methods used on the CPU are sequential. The GPU card used is NVIDIA TESLA C2050 with computing capability of 2, 448 thread processors and clock value of 1.15 GHz.

5. GPU parallel computing with MATLAB

In MATLAB R2012a, Parallel Computing Toolbox V6.0 was presented. This toolbox has overcome many limitations which were outlined in [19] and it is expected to be further improved in the future. This toolbox requires the GPU to be included in CUDA 1.3 devices or more. When working with this toolbox, three major options are available:

1. Using GPU arrays with MATLAB built-in functions.
2. Executing custom MATLAB functions on elements of GPU arrays.
3. Creating kernels from existing CUDA code and PTX files.

As our work is focused on the implementation of MATLAB only, we have used both first and second options.

In addition to the options above, other third party toolboxes may be used, mainly Jacket and GPUmat, but we preferred to work with Mathworks toolbox as it is friendlier to MATLAB users and expected to improve in next versions as it was greatly improved in this version.

MATLAB parallel functions may be slower than those written by CUDA; however working with MATLAB has many benefits in prototyping and testing thanks to its variety of functions, toolboxes and visualization methods.

6. Work procedure

6.1. Sequential computations

We wrote sequential function for the following Fourier CGH methods:

1. Kinoform (the simplest of all methods).
2. Kinoform with diffusing stage.
3. Detour phase with cell size of 16×16.
4. Detour phase with cell size of 16×16 after adding diffusing stage (both Detour phase methods were the heaviest on memory).
5. Lee.
6. Burckhardt.

The input is a grayscale bmp image with variable dimensions, all the functions start with inverse Fourier transform and Fourier shift functions, then each function differs according to its related representation method. To verify the results we wrote a function that calculates and shifts Fourier transform for the generated hologram to plot the result image. Fig. 1 shows the source image while Figs. 2–7 show simulation's output images for each method.

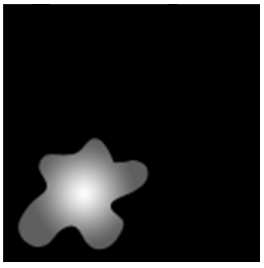

Fig. 1. Source image for all sequential and parallel functions; only the resolution is variable.

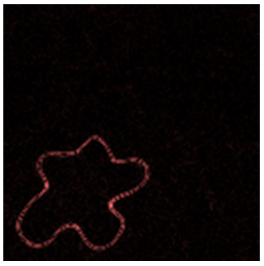

Fig. 2. Simulation result of illuminating the generated Kinoform. The results of sequential and parallel computations were identical (color map is set to red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)