CUDA를 이용한 3D 측정 속도 향상

김호중* · 조태훈*

*한국기술교육대학교

Improving 3D Measurement Speed using CUDA

Ho-Joong Kim* and Tai-Hoon Cho*

*KoreaTech

E-mail : hjhjhjof@naver.com

요 약

최근 3D 측정을 위해서 Fringe pattern을 이용하는 방법이 많이 사용되고 있다. 이는 측정할 물체에 pattern을 뿌려 얻은 위상 값을 이용하여 측정하는 방법이다. 이를 위해선 위상 값을 계산하고 높이를 계산하는 등의 많은 연산을 요구 한다. 많은 연산량에 따라 시간이 많이 걸리고 있으며, 본 논문에서는 이 시간을 감소시키기 위해서 엔비디아의 CUDA를 이용한 방법을 제시한다. 또 위상 값과 높이를 계산하는 방법을 소개하고 CPU 버전과 CUDA 버전 사이의 비교를 통해 정확한 시간 차이를 보인다. 이 방법을 이용하면 같은 연산을 더 짧은 시간 내에 처리할 수 있어 같은 시간에 많은 부품을 검사할 수 있기 때문에 매우 효과적이라 할 수 있다.

ABSTRACT

Recently, a method using a fringe pattern is widely used for 3D measurements. This is a method of measuring by using a phase value obtained by projecting a pattern to an object to be measured. This method requires many operations such as calculating the phase value and calculating the height. It takes a lot of time depending on the amount of computation. In this paper, we present a method using NVIDIA' s CUDA to reduce this time. And we introduce the method of calculating phase value and height. It also shows the exact time difference between the CPU version and the CUDA version. This method is very effective because it can process the same operation in a shorter time.

키워드

Cuda, Fringe projection, Fast Speed, 3D measurement

Ⅰ. 서 론

Fringe Projection Profilometry (FPP)는 정확성이 높고 여러 물체가 한 영상에 섞여 있어도 측정이 가능하다는 이점이 있기 때문에 3D 측정 분야에서 많이 사용되고 있다. [1] 이는 특정한 함수에 의해 만들어진 Fringe pattern을 물체에 뿌린 영상에서 위상 정보를 얻어 물체의 높이를 측정하는 방법이다. 그렇지만 이 방법은 연산량이 많기 때문에 처리 속도가 지연될 수 있다. 따라서 이는 고속 처리와 맞지 않기 때문에 본 논문에서는 NVIDIA의 CUDA(Compute Unified Device Architecture)를 사용하는 방법을 제안한다. 이를 사용하면 GPU를 이용하여 병렬 처리를 하기 때문에 연산 시간을 줄일 수가 있다.

본 논문의 구성은 본론, 실험 결과, 결론으로 되어 있으며 다음과 같다. 본론에서 위상 값 및 높이를 계산하는 방법을 설명하고, CUDA에 대하여 설명한다. 실험 결과에서는 같은 데이터에 대하여 CPU 버전과 GPU 버전의 측정 결과를 비교하여 CUDA 사용의 이점을 보인다.

Ⅱ. 본 론

앞서 말한바와 같이 본론에서는 위상 값 및 높이를 계산하는 방법에 대하여 설명하고 마지막으

로 CUDA에 대하여 설명한다. 기본 사항으로 본 논문에서는 Fringe Pattern으로 사인파를 사용하였다. 그리고 측정하고자 하는 물체에 어느 한 주기의 사인파를 뿌렸을 때 패턴의 굴절되는 정도가 2π를 넘어간다면 논문[2, 3] 에 따라서 주기가 더 큰 영상을 이용하여 작은 주기의 영상을 Unwrap 해 주었다. 또 카메라 캘리브레이션은 Guo의 방법[4]을 이용하여 이미 되어 있는 상태에서 3D 측정을 진행하였다.

2.1 위상 값 계산

본 논문에서는 사인파를 이용해 위상 값을 계산 하는데 위상천이 양은 $\pi/2$이고, 4-단계 위상천이 방법을 사용하였다. 순차적인 위상 천이에 의한 4장의 사인파 그레이팅 밝기 영상은 식 (1)과 같이 표시된다.[5]

$$I_n(x,y) = I_b(x,y) + I_m(x,y)\cos[\phi(x,y) - (n-1)\frac{\pi}{2}],$$
$$n = 1,2,3,4 \qquad (1)$$

$$\phi(x,y) = \tan^{-1}\left[\frac{I_4(x,y) - I_2(x,y)}{I_1(x,y) - I_3(x,y)}\right] \qquad (2)$$

n은 4장에 대한 순차 번호이고, $I_b(x,y)$는 배경 밝기, $I_m(x,y)$는 fringe modulation, $\phi(x,y)$는 (x,y)에서의 위상 값을 뜻한다. 식 (1) 을 풀어 쓰면, (x,y)에서의 위상 값은 식 (2) 와 같이 된다. 이를 보면 두 번의 뺄셈과 한 번의 나눗셈, 한번의 $\tan^{-1}$ 연산이 들어가는 것을 확인할 수 있다. 이는 각 픽셀마다 필요한 연산이며, 고해상도의 이미지를 처리하는 데 있어서 많은 시간을 요구한다.

2.2 3D 높이 계산

앞서 말한 바와 같이 캘리브레이션은 Guo의 방법으로 수행하였고, 그에 따라 각 픽셀별로 높이를 측정하기 위한 계수가 두 개씩 존재한다. 이 계수들을 이용하여 높이를 측정하는데 필요한 식은 다음과 같다.

$$h_i = \frac{(\phi_i^{obj} - \phi_i^{ref})}{(coeff_i^0 * (\phi_i^{obj} - \phi_i^{ref}) + coeff_i^0)} \qquad (3)$$

h_i는 i 픽셀에서의 높이를 나타내고, ϕ_i^{obj} 는 측정할 물체 영상 i 픽셀의 위상 값이며, ϕ_i^{ref} 는 캘리브레이션 기준면 영상 i 픽셀의 위상 값을 나타낸다. $coeff_i^0$ 과 $coeff_i^1$ 은 캘리브레이션을 통해 구해 놓은 i 픽셀의 계수를 의미한다. 이 식을 살펴보면 픽셀 별로 두 번의 뺄셈과 덧셈, 곱셈, 나눗셈이 한번 씩 필요한 것을 확인할 수 있다. 이 또한 모든 픽셀마다 필요한 연산이기 때문에 많은 처리량을 요구한다.

2.3 CUDA

CUDA는 그래픽 처리 장치(GPU)에서 수행하는 병렬 처리 알고리즘을 C 프로그래밍 언어를 비롯한 산업 표준 언어를 사용하여 작성할 수 있도록 하는 GPGPU 기술이다.[6] CUDA의 처리 과정을 간단히 살펴보면 <그림 1>과 같다.

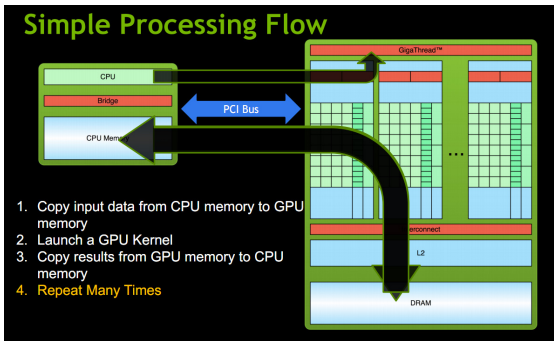

그림 1. CUDA Processing Flow

CUDA를 이용하여 병렬처리를 하기 위해선 그림에 나와 있듯이 CPU 메모리에 저장 되어 있는 데이터를 GPU로 Copy 해주어야 한다. 그렇지만 고해상도의 데이터를 Copy하는 데에는 시간이 많이 걸리기 때문에, 이를 고려하여 Copy하는 횟수를 최대한 줄이도록 해야 한다. 이 후에 GPU Kernel에서 병렬처리 알고리즘을 수행하고, 그 결과를 CPU로 다시 Copy하여 원하는 결과를 얻을 수 있다.

CUDA의 구조를 살펴보면 <그림 2>와 같다.[7]

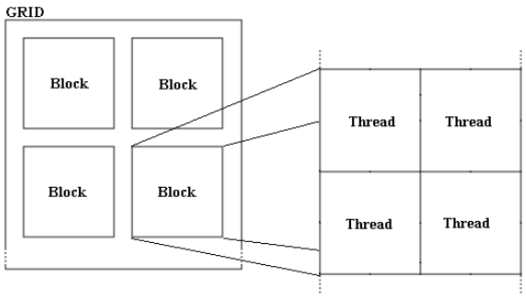

그림 2. CUDA Architecture

위의 그림을 보면 CUDA의 구조는 Grid, Block, Thread로 이루어져 있는 것을 볼 수 있다. Grid는 주어진 Block들이 모여 이룬 것을 의미하고, Block은 Thread의 집합체이다. 이런 Block 안에 각각의 Thread 들이 한 픽셀의 연산을 담당하고, 한 Block안의 Thread들은 동시에 수행이 된다. 이로 인해 병렬처리가 가능한 것이다. CUDA는 Kernel 안에서 공통적으로 사용되는 인덱스에 관한 변수가 있는데 <표 1>과 같다.

표 1. Kernel 함수 내의 인덱스 변수

blockIdx	Grid안의 Block 인덱스
threadIdx	Block안의 Thread 인덱스
blockDim	Block안의 Thread 개수

이러한 인덱스 변수를 통하여 1차원 데이터의 각 픽셀에 접근할 수 있는데, 식을 보면 다음과 같다.

$$int\ idx = threadIdx.x + blockIdx.x * blockDim.x; \quad (4)$$

따라서 Kernel 함수 내에서 이를 통해 각각의 픽셀에 접근하여 원하는 연산을 수행하면 병렬처리가 가능해진다.

지금까지 3D 높이 측정을 위해 필요한 연산과 CUDA에 대해 간략하게 알아보았고 다음으로 실험 결과에 대해서 설명한다.

Ⅲ. 실험 결과

실험은 Windows 10(64bit), Microsoft Visual Studio 2013의 환경에서 진행되었고, CPU는 Intel Core i7-6700K 4.00Ghz이며, CUDA에 사용된 GPU는 NVIDIA GeForce GTX 1060 6GB이다. <그림 3>과 같이 캘리브레이션을 위해 쓰인 영상의 해상도는 3904x3904이며 사인파를 사용하였다.

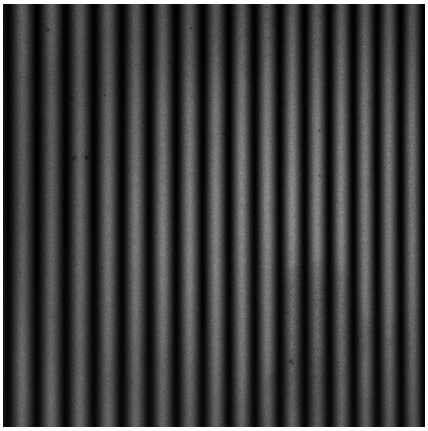

그림 3. Calibration Plane

앞서 말한바와 같이 Guo의 방법을 이용하여 캘리브레이션을 하기위해 0, 300, 600, ~ , 3000 μm의 캘리브레이션 플레인을 사용하였다. 실험은 캘리브레이션이 되어 있는 상태에서, 앞서 설명했던 알고리즘을 CPU 버전과 CUDA 버전으로 구현하여 시간 비교를 하였다. 실험을 위해 측정할 물체는 3000μm의 플레인을 재 측정하여 에러를 측정하였다. CPU 버전으로 실험한 결과는 <표 2>와 같고, CUDA 버전으로 실험한 결과는 <표 3>과 같다.

표 2. CPU 버전 측정 시간

위상 계산	1728.85
위상 차이 계산	46.49
높이 계산	84.53

[단위:ms]

표 3. CUDA 버전 측정 시간

Copy (Host to Device)	10.08
위상 계산	2.63
위상 차이 계산	2.39
높이 계산	2.68
Copy (Device to Host)	5.48

[단위:ms]

표 4. 에러 측정 결과

	true	avg	sigma	rmse
CPU	3000	3000.3	5.1	5.2
CUDA	3000	3000.3	5.1	5.2

[단위:μm]

우선 <표 2>를 보면 CPU 버전의 측정 시간을 확인할 수 있고, <표 3>을 보면 CUDA 버전의 측정 시간을 확인할 수 있다. <표 3>에서 Copy는 본론에서 설명했듯이 데이터를 전송하는 시간을 의미한다. 또, Host는 CPU를 나타내고 Device는 GPU를 나타낸다. 두 개의 표를 비교해보면 CPU보다 CUDA로 처리한 것이 훨씬 빠른 것을 확인할 수 있다. <표 4>에서 true는 실제 높이 값을 의미하고 avg는 평균을, sigma는 표준편차, rmse는 Root Mean Square Error을 의미한다. 이를 보면 두 버전의 에러 측정 결과가 완전히 동일한 것을 확인할 수 있다.

따라서 CUDA를 통해 같은 알고리즘을 더 빠른 시간에 처리가 가능하다는 것을 확인할 수 있다.

Ⅳ. 결 론

본 논문에서는 3D 측정에서 많이 사용되고 있는 알고리즘에 대해 CUDA를 이용하여 더 빠르면서 CPU와 같은 결과를 내었다. 이것은 같은 시간 내에 많은 부품을 검사할 수 있다는 의미로, 매우 효과적인 방법이라 할 수 있다. 추후에는 간단한 높이 계산뿐만 아니라, 정확한 높이 측정을 위한 작업에도 CUDA를 적용시켜 시간을 더욱 단축시키는 연구가 필요하다.

참고문헌

[1] Zhaoyang Wang, Dung A. Nguyen and John C. Barnes, “Some practical considerations in fringe projection profilometry”, Optics and Lasers in Engineering, vol. 48, no. 2, pp.

218-225, 2010.

[2] H. Zhao, W. Chen, Y. Tan, "Phase-unwrapping algorithm for the measurement of three-dimensional object shapes," Applied Optics, vol. 33, no. 20, pp. 4497-4500, July. 1994.

[3] J. Li, L. G. Hassebrook, and C. Guan, "Optimized two-frequency phase-measuring-profilometry light-sensor temporal-noise sensitivity," Journal of the Optical Society of America A, vol. 20, no. 1, pp.106-115, Jan. 2003.

[4] Hongwei Guo, Haitao He, Yingjie Yu, Mingyi Chen, "Least-squares calibration method for fringe projection profilometry", Optical Engineering, vol. 44, Issue. 3, 2005.

[5] V. Srinivasan, H. C. Liu, M. Halioua, "Automated phase measuring profilometry of 3-D diffuse object," Appl. Opt. 23, no. 18, pp. 3105-3108, 1984.

[6] https://ko.wikipedia.org/wiki/CUDA

[7] R. Giot, "Image Processing Algorithm Optimization with CUDA for Pure Data", Pure Data Convention, 2011.