JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 12 : 2285~2290 Dec. 2017

GPU를 이용한 위상 측정법의 가속화

김호중[1*] · 조태훈[2]

Acceleration of Phase Measuring Profilometry using GPU

Ho-Joong Kim[1*] · Tai-Hoon Cho[2]

[1*]Department of Computer Engineering, KOREATECH, Cheonan, 31253, Korea
[2]School of Computer Science and Engineering, KOREATECH, Cheonan, 31253, Korea

요 약

최근 산업의 여러 분야에서 자동화 시스템이 발전함에 따라 3D 측정에 의한 물체의 높이 검사의 필요성이 점차 대두되고 있다. 여러 3D 측정 방법 중에서 본 논문에서 다루는 방법은 위상 측정법으로, 위상 측정법이란 프린지 패턴의 위상값을 이용하여 물체의 높이를 구하는 방법이다. 위상 측정법은 연산량이 많이 필요한 알고리즘이기 때문에 이를 효율적으로 해결할 방법이 필요하다. 본 논문에서는 이를 위해 NVIDIA에서 나온 CUDA를 사용할 것을 제안했다. 또 CUDA에서 제공하는 Pinned memory와 Stream을 사용할 것을 제안하였다. 이를 통해 정확도를 유지하면서 측정 속도는 크게 향상시킬 수 있었고 실험을 통해 성능을 입증하였다.

ABSTRACT

Automation systems are evolving in many areas of industry in recent years. At the same time, the necessity of the height inspection of the object by the 3D measurement is gradually increasing. Among the various 3D measurement methods, this paper discusses phase measuring profilometry(PMP). The PMP is a method of obtaining the height of an object using the phase value of the fringe pattern. Since the PMP is an algorithm requiring a large amount of computation, a method for efficiently solving the problem is needed. In this paper, we propose to use CUDA from NVIDIA to solve this problem. We also propose using pinned memory and streams provided by CUDA. This can greatly improve the measurement speed while maintaining accuracy. Finally, we demonstrate the performance of the proposed method through experiments.

키워드 : 고속화, 위상 측정법, CUDA, 3D 측정

Key word : Acceleration, Phase Measuring Profilometry, CUDA, 3D measurement

Received 27 July 2017, **Revised** 01 August 2017, **Accepted** 22 September 2017

* **Corresponding Author** Ho-Joong Kim(E-mail:hjhjhjof@gmail.com, Tel:+82-41-560-1114)
Department of Computer Engineering, KOREATECH, Cheonan, 31253, Korea

Open Access https://doi.org/10.6109/jkiice.2017.21.12.2285 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

최근 산업의 여러 분야에서 자동화 시스템이 발전함에 따라 3D 측정에 의한 물체의 높이 검사의 필요성이 점차 대두되고 있다. 이 중 위상 측정법이 많이 사용되면서 더 높은 정확성과 빠른 속도를 위한 연구들이 활발하게 이루어지고 있다. 많은 연구들 중 몇 가지만 소개해보면 다음과 같다. 정확한 위상값을 구하기 위한 방법[1], 주위의 빛을 고려한 위상값의 분석 및 보정 방법[2], 두 개의 주기를 이용한 위상 측정법[3] 등이 연구되고 있다. 또 위상 측정법을 이용한 빠른 픽셀 매칭 방법[4], 직선 라인위에서 움직이는 물체를 위한 위상 측정법[5] 등이 있다.

위상 측정법은 PCB 부품의 불량을 찾는 곳에 많이 사용되고 있는 매우 효과적인 방법이지만 많은 연산량에 의해 속도 측면에서 느리다는 문제점을 가지고 있다. 본 논문에서는 이를 해결하기 위한 연구를 진행하였다. NVIDIA에서 제공하는 CUDA를 이용하였고 특히, Pinned memory와 Stream을 사용해 성능을 더욱 끌어올렸다[6].

본 논문의 2장에서는 위상 측정법에 대한 설명 뒤, 속도에 관한 문제점을 제시한다. 3장에서는 CUDA에 대한 설명 및 제안하는 방법에 대해 설명한다. 4장에서는 실험을 통해 제안하는 방법의 성능을 입증한다. 마지막으로 5장에서 결론을 통해 본 논문을 마무리한다.

Ⅱ. 위상 측정법

2.1. 위상 측정법

위상 측정법이란 Phase Measuring Profilometry(PMP)로써 특정한 수식에 의해 만들어진 프린지 패턴(Fringe pattern)을 물체에 투영하여 얻은 위상 정보를 통해 높이를 측정하는 방법이다. 이 방법은 빠른 처리 속도, 높은 정확성, 손쉬운 시스템 구성, 여러 물체가 섞여 있어도 한 번에 측정이 가능한 점 등의 이점이 있어 실제로 3D 측정에 많이 쓰이고 있는 방법이다[7]. 그림 1은 위상 측정법의 광학적 기하도를 나타낸다. 기준면 위에 측정할 물체를 올려놓고 그 위에 카메라를 설치한 뒤 프린지 패턴을 투영하여 영상을 얻는 방법이다.

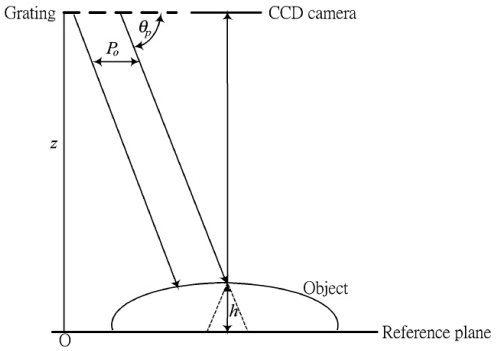

Fig. 1 Optical geometry of PMP

P_0는 기준면에서의 사인파의 주기, θ_p는 기준면과 프로젝션 축과의 각도, z는 기준면과 카메라 사이의 높이, h는 물체의 높이를 의미한다[8].

위상 측정법의 단계로는 그림 2와 같다. 그림에서 3D 캘리브레이션이란 실제 높이를 구하기 위한 사전 작업이며 본 논문에서는 Guo의 방법[9]을 이용하였다.

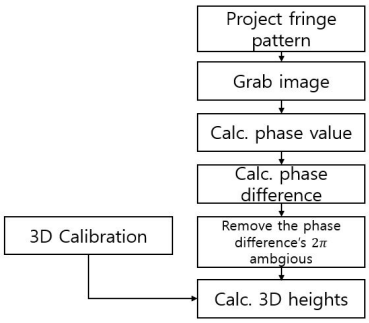

Fig. 2 Processing steps of PMP

2.2. 위상값 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 밝기 영상은 식 (1)과 같이 표시된다.

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

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

n은 4장에 대한 순차 번호이고, $I_b(x,y)$는 배경 밝기, $I_m(x,y)$는 프린지 변조, $\phi(x,y)$는 (x,y)에서의 위상 값을 뜻한다. 식 (1) 을 이용한 위상 값은 식 (2) 와 같다.

2.3. 위상차 2π 모호성 제거

위상차란 기준면의 위상값과 측정할 위상값의 차이를 의미한다. 이런 위상차는 2π 주기로 반복되는 성질이 있다. 높은 물체에 의해 위상차가 2π를 넘어간다면 이 값은 제대로 된 위상차라 할 수가 없다. 이런 위상차의 2π 모호성을 방지하기 위해서 사인파의 크기가 다른 두 주기의 영상을 이용하는 방법을 사용했다[10,11].

$$\triangle\phi_r(x,y) = \triangle\phi_s(x,y) + 2\pi n \tag{3}$$

$$n = INT\left(\frac{T2/T1 \times \triangle\phi_l(x,y) - \triangle\phi_s(x,y)}{2\pi}\right) \tag{4}$$

식 (3)에서 $\triangle\phi_r(x,y)$는 언랩(Unwrap)을 한 실제 위상차를 의미하고 $\triangle\phi_s(x,y)$는 작은 주기 위상차를 의미한다. 또 INT는 가장 가까운 정수를 의미하고 n은 식 (4)에 의해 계산된 정수이다 . 식 (4)의 $T2$는 큰 주기의 값이고 $T1$는 작은 주기의 값을 의미하며 $\triangle\phi_l(x,y)$은 큰 주기로 얻은 영상의 위상차, $\triangle\phi_s(x,y)$는 작은 주기로 얻은 영상의 위상차를 의미한다. 식 (4)를 통해 정수 n을 계산하고, 이를 식 (3)에 적용시켜 계산하면 언랩한 위상차를 구할 수 있다.

2.4. 높이 계산

Guo의 방법으로 각 픽셀마다 높이 측정을 위한 계수를 구해놓은 상태이기 때문에, 이를 이용하여 입력 위상값에 대해 높이를 구할 수 있다. 높이값을 계산하는 식은 식 (5)와 같다. $h(x,y)$는 픽셀위치 (x,y)에의 높이를 의미하고 $\triangle\phi_{input}(x,y)$는 픽셀위치 (x,y)에서 기준면과 측정할 물체 사이의 위상차를 의미한다.

$$h(x,y) = \frac{\triangle\phi_{input}(x,y)}{c_0(x,y) \times \triangle\phi_{input}(x,y) + c_1(x,y)} \tag{5}$$

또 $c_0(x,y)$, $c_1(x,y)$는 앞서 구해놓은 3D 캘리브레이션 계수를 의미한다.

지금까지 위상 측정법을 여러 수식과 함께 설명하였다. 이 수식들은 모두 픽셀별로 이루어져야 하는 작업이고, 고해상도 영상을 처리하려면 연산량이 많아 속도가 느린 문제점이 있다. 이를 해결하기 위한 방법으로 CUDA에 대해 설명하고 앞서 설명한 수식에 적용하는 것을 제안한다.

Ⅲ. CUDA

3.1. CUDA의 정의 및 처리 과정

CUDA는 그래픽 처리 장치(GPU)에서 수행하는 병렬 처리 알고리즘을 C 프로그래밍 언어를 비롯한 산업 표준 언어를 사용하여 작성할 수 있도록 하는 GPGPU (General-Purpose computing on Graphics Processing Units, GPU 상의 범용 계산) 기술이다. 이처럼 CUDA는 병렬 처리 알고리즘에 특화되어 있기 때문에 영상처리 및 컴퓨터 비전 분야에서 많이 사용되고 있는 추세이다.

CUDA의 처리 과정은 다음과 같다. 우선 CPU에 있는 메모리를 GPU로 전송해주어야 하고 커널 함수를 호출한다. 이 커널 함수에서 실제 병렬처리를 수행하고, 결과를 다시 CPU의 메모리로 전송을 해주면 된다. 주의해야 할 점은 메모리가 큰(고해상도의 영상) 데이터를 전송하는 시간이 많이 걸리기 때문에 이를 고려하여 전송하는 횟수를 최대한 줄이도록 해야 한다.

3.2. 제안하는 방법

본 논문에서는 앞서 설명한 위상값 계산, 2π 모호성, 높이 계산에 대하여 CUDA로 변환하는 것을 제안한다. 또 성능을 극대화하기 위해 Pinned memory와 스트림(Stream)을 사용하는 것을 제안한다.

```
// Kernel function
__global__ void kernel_function(parameters)
{
    int index = blockIdx.x * blockDim.x + threadIdx.x;

    // Do something..
}

// Function call
kernel_function <<< blocks, threads >>> (parameters);
```

Fig. 3 Function call and index approach in CUDA

그림 3을 보면 기본적인 CUDA의 문법을 알 수 있다. 커널 함수는 GPU에서 실행되는 함수를 의미하며, 함수 안의 알고리즘은 병렬로 실행이 된다. index는 병렬 처리를 위한 인덱스이고, 함수 호출의 blocks는 스레드 블록 수, threads는 각 블록 당 스레드 수를 의미한다.

Pinned memory란 GPU의 DMA(Direct Memory Access)가 CPU의 개입 없이 Host(CPU)와 Device (GPU)사이의 전송을 가능하게 해주는 메모리이다. 즉, 이 메모리는 RAM에 저장되므로 Device가 Host의 도움 없이 전송할 수 있다. 그림 4는 Pinned memory에 대한 코드를 보여준다. 두 가지 방법이 있으며, 둘 중 하나의 방법으로 Pinned memory를 만들면 된다. 특히 CPU와 GPU 사이의 메모리 전송이 많은 데이터에 대해 만들면 더 효과적이다.

```
// First method
float* pinned_memory;
cudaMallocHost((void**)&pinned_memory, size_of_data);
cudaFreeHost(pinned_memory);

// Second method
float* pinned_memory2 = new float[size_of_data];
cudaHostRegister(pinned_memory2, size_of_data, 0);
cudaHostUnregister(pinned_memory2);
delete pinned_memory2;
```

Fig. 4 Code about pinned-memory

CUDA에서 제공하는 스트림을 이용하여 속도를 더 향상시켰다. 2.3절에서 설명했듯이 본 논문에서는 두 주기의 프린지 패턴을 사용하였고, 두 주기 영상에 대한 연산을 위해 각각 스트림을 할당하였다. 이 연산들은 서로 영향을 미치지 않는 연산이기 때문에 같은 시간에 처리가 가능하다. 이러한 이유로 각각의 스트림을 이용하여 연산을 처리하였다.

```
cudaStream_t stream_1, stream_2;
cudaStreamCreate(&stream_1);
cudaStreamCreate(&stream_2);
...
cudaMemcpyAsync(gpu_data1, cpu_data1, size_of_data,
                cudaMemcpyHostToDevice, stream_1);
cudaMemcpyAsync(gpu_data2, cpu_data2, size_of_data,
                cudaMemcpyHostToDevice, stream_2);
...
kernel_function <<< blocks, threads, 0, stream_1 >> > (parameters);
kernel_function <<< blocks, threads, 0, stream_2 >> > (parameters);
...
cudaStreamDestroy(stream_1);
cudaStreamDestroy(stream_2);
```

Fig. 5 Code about stream

그림 5는 두 개의 스트림을 생성하는 코드이다. 코드에 나와 있듯이, 스트림을 만들고 활용하기 위해서 cudaMemcpyAsync나 커널 함수를 호출할 때에 스트림을 지정해주어야 한다.

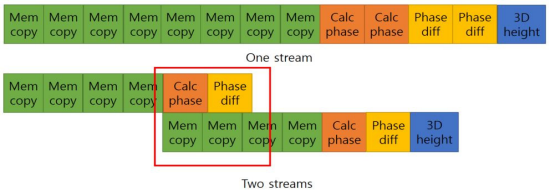

Fig. 6 Processing flow of CUDA

그림 6은 입력 데이터에 대한 처리 과정을 스트림을 사용했을 경우와 그렇지 않았을 경우에 대해 비교한 것이다. 이 그림에서 위의 것은 하나의 스트림을 이용해 두 주기의 데이터를 처리한 결과이고 아래 것은 두 스트림을 이용하여 두 주기의 데이터를 처리한 결과이다. 눈 여겨 보아야 할 것은 두 개의 스트림을 사용 했을 때 연산을 하는 부분과 데이터를 전송하는 부분이 겹치는 시간이 발생한다는 것이다.

이렇게 CUDA를 활용해서 위상값 계산, 2π 모호성, 높이 계산에 대하여 적용하면 성능을 올릴 수 있다.

Ⅳ. 실험 결과

본 논문에서 제안한 방법의 성능 평가를 위한 하드웨어 시스템 구성은 표 1과 같다. 소프트웨어 시스템 구성은 표 2와 같다. 공통사항으로 3904×3904크기의 영상을 캡처하는데 Active Silicon사의 CoaXPress 프레임그래버와 ISVI사의 카메라를 사용하였다. 카메라의 프레임레이트는 110장/초이며 캡처시간은 약 $9ms$이다. 이에 대한 측정시간은 결과에서 제외하였다.

본 논문의 실험은 모두 GPU의 메모리를 할당하는, cudaMalloc 부분을 제외한 결과이다. 이것은 시간이 오래 걸리지만 프로그램이 시작될 때 한 번만 할당해 놓으면 되는 작업이기 때문에 실제 측정 시간에는 영향을 미치지 않는다. 그러므로 이 시간은 측정시간에서 제외하였다. 제안된 방법의 성능평가 결과는 표 3에서부터 표 6에 제시하였다. 첫째, Pinned memory의 사용 유무에 따른 실험 결과는 표 3과 같다. 표 3은 3904×3904

크기의 float형 데이터를 CPU에서 GPU로 복사하는 시간을 측정한 것이다. 이를 보면 Pinned memory를 사용한 것이 그렇지 않은 것에 비해 약 3배 빠른 것을 확인할 수 있다.

Table. 1 Hardware system components

	Model	Specifications	
CPU	Intel Core i7-6700K 4.00Ghz	Number of core	4
		Number of thread	8
GPU	NVIDIA GeForce GTX 1060 6GB	CUDA cores	1280
		Maximum number of threads per block	1024

Table. 2 Software system components

Software	Version
Windows 10	64bit
Microsoft Visual Studio	2013
CUDA	8.0

Table. 3 Experimental results of Pinned memory

Memory type	Memory copy time(ms)
Not pinned memory (Pageable memory)	14.9
Pinned memory	4.7

둘째, 입력 데이터 처리에 두 개의 스트림을 사용 하였고 하나의 스트림은 작은 주기의 데이터를 계산하기 위해 사용하고 다른 스트림은 큰 주기의 데이터를 계산하기 위해 사용했다. 이로써 앞서 언급한 겹치는 시간을 발생시켰고, 시간을 단축시킬 수 있었다.

그리고 셋째, 위상값 계산, 위상차 계산 그리고 높이 계산까지의 과정에 대해 CPU 버전과, Pinned memory와 스트림을 적용한 CUDA 버전을 비교하였다. 실험에서 사용한 영상의 해상도는 3904×3904이며 3D 캘리브레이션을 위해 기준면부터 300, 600, ... , 2700, 3000 μm의 캘리브레이션 플레인을 사용하였다. 결과 값을 확인하기 위해 $3000 \mu m$의 플레인의 높이를 재 측정하여 에러를 계산하였다. CPU 버전의 시간 측정 결과는 표 4와 같고 CUDA 버전의 시간 측정 결과는 표 5와 같다. 두 결과를 비교해보면 CUDA 버전이 CPU 버전에 비해 약 80배 빠른 것을 확인할 수 있다. CPU 버전과 CUDA 버전의 정확도를 비교한 결과는 표 6과 같다.

Table. 4 Time measurement results of CPU version

Work	Time(ms)
Phase calculation	1728.8
Phase difference calculation	46.4
Height calculation	84.5
Total	1859.7

Table. 5 Time measurement results of CUDA version

Work	Time(ms)
Copy (Host to Device)	10.0
Phase calculation	2.6
Phase difference calculation	2.3
Height calculation	2.6
Copy (Device to Host)	5.4
Total	22.9

Table. 6 Error measurement results for each version

Version	True(μm)	Avg(μm)	Sigma(μm)	RMSE(μm)
CPU	3000	3000.3	5.1	5.2
CUDA	3000	3000.3	5.1	5.2

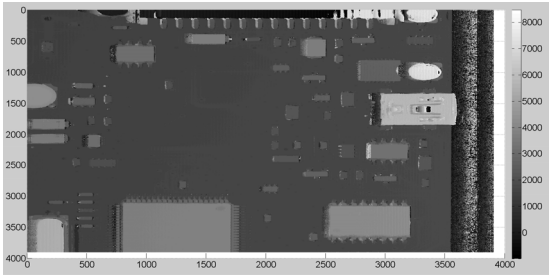

Fig. 7 Example of 3D height data of PCB

표 6에서 True는 실제 높이 값을 의미하고 Avg는 측정한 평균값을 의미한다. Sigma는 표준편차, RMSE는 Root Mean Squared Error를 나타낸다. 두 버전의 에러 측정 결과가 동일한 것을 볼 수 있다. 마지막으로 그림 7은 측정한 3D 높이 데이터의 예이다.

Ⅴ. 결 론

본 논문에서는 3D 측정에서 각광을 받고 있는 위상 측정법에 대해 설명하였고 속도 측면에서의 문제점을

제시했다. 이를 해결하기 위해 CUDA 사용을 제안하였다. 특히 CUDA에서 Pinned memory와 스트림을 함께 사용할 것을 제안하였고 실험을 통해 성능을 입증하였다. 따라서 위상 측정법 분야에서 CUDA를 이용하여 성능을 올릴 수 있음을 입증하였다. 추후에는 앞서 소개한 위상 측정법의 기본적인 알고리즘 이외의 후처리 작업에 대해서도 CUDA를 적용하는 연구가 필요하다.

ACKNOWLEDGMENTS

This paper was supported by the Education and Research Promotion Program of KOREATECH.

REFERENCES

[1] M. Liu, S. Yin, S. Yang and Z. Zhang, “An accurate projector gamma correction method for phase-measuring profilometry based on direct optical power detection.” in *Proceedings of the SPIE 9677*, pp. 96771D1-1, Oct. 2015.

[2] P. Zhou, X. Liu, Y. He and T. Zhu, “Phase error analysis and compensation considering ambient light for phase measuring profilometry.” *Optics and Lasers in Engineering*, vol. 55, pp. 99-104, April 2014.

[3] B. Wang, Y. Liang, and H. Deng, “Dual-frequency grating method based research on phase measurement profilometry (PMP) technology.” in *Proceedings of the SPIE 9298*, pp. 929808, Nov. 2014.

[4] X. Xu, Y. Cao, Y. Wang, C. Chen, G. Fu and S. Sun, “A fast pixel matching method based on phase feature extraction in online phase-measuring profilometry.” *Journal of Modern Optics*, vol. 64, pp. 1-8, May 2017.

[5] C. Chen, Y. Cao, L. Zhong and K. Peng, “An on-line phase measuring profilometry for objects moving with straight-line motion,” *Optics Communications*, vol. 336, pp. 301-305, Feb. 2015.

[6] J. Luitjens, “CUDA Streams : Best Practices and Common Pitfalls” in *Proceedings of GPU Technology Conference*, San Jose, pp. 2-35, 2014.

[7] S. Zhang, “High-resolution, Real-time 3D Shape Measurement,” Ph. D. dissertation, Stony Brook University, Stony Brook, New York, 2005.

[8] H. Yen, D. Tsai and J. Yang, “Full-Field 3-D Measurement of Solder Pastes Using LCD-Based Phase Shifting Techniques,“ *IEEE Transactions on Electronics Packaging Manufacturing*, vol. 29, no. 1, pp. 50-57, Jan. 2006.

[9] H. Guo, H. He, Y. Yu, and M. Chen, “Least-squares calibration method for fringe projection profilometry,” *Optical Engineering*, vol. 44, no. 3, pp. 033603, March 2005.

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

[11] 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.

김호중(Ho-Joong Kim)

2015. 2 : 한국기술교육대학교 컴퓨터공학부 졸업
2015. 9 ~ 현재 : 한국기술교육대학교 컴퓨터공학과 재학
※관심분야 : 영상처리, 컴퓨터비젼

조태훈(Tai-Hoon Cho)

1991. 5 : Virginia Tech 전기공학과 공학박사
1992. 3 ~ 1998.2 : LG산전연구소 수석연구원
1998. 3 ~ 현재 : 한국기술교육대학교 컴퓨터공학부 교수
※관심분야 : 영상처리, 컴퓨터비젼