실시간 깊이 지도 획득을 위한 효율적인 병렬 처리

*조칠석 *전지인 **추현곤 *박종일1)

*한양대학교 **한국전자통신연구원

cscho@mr.hanyang.ac.kr, starlove7@mr.hanyang.ac.kr, hyongonchoo@etri.re.kr, jipark@hanyang.ac.kr

Efficient Parallel Processing for Depth-Map Estimation in Real-Time

Chil-Suk Cho*, Ji-In Jun*, Hyon-Gon Choo** and Jong-Il Park

*Hanyang University **Electronics and Telecommunication Research Institute

요약

Depth map를 구하는 방법 중 많이 사용되어지는 방법으로 stripe 패턴을 이용하는 방법이 존재한다. 이 방법은 Pro-Cam 시스템을 이용하며 프로젝터로 조사한 패턴을 카메라로 촬영하여 원래의 패턴과 촬영된 패턴간의 기하학적인 관계를 구하여 depth map를 구하는 방법이다. 본 논문에서는 이와 같이 구조광을 이용하는 depth map 획득 시스템을 효과적으로 multi-thread를 사용하여 실시간 처리하는 것을 제안한다. 일반적으로 자주 사용되는 multi-threading 기법에는 CPU의 thread를 이용하는 OpenMP와 GPU의 thread를 이용하는 CUDA가 있다. 이 두 가지 기법은 수행하는데 차이점이 존재하기 때문에 상황에 따라 OpenMP가 더 좋은 효율을 보이는 부분이 있고 CUDA가 더 좋은 효율을 보이는 부분이 있다. 때문에 우리는 이 두 가지에 대해서 각 부분의 특성에 맞게 더 좋은 효율을 보이는 multi-thread를 이용하였다. 결과적으로 우리는 1280×800의 영상에 대해 25fps 이상의 depth map를 획득하였다.

1. 서론

3D 영화를 통해 생긴 많은 관심은 3D방송으로 이어지고 있다. 3D 기술은 depth map를 필요로 한다. 원활한 3D방송을 위해서는 높은 해상도에서 실시간으로 depth map를 획득하고 이를 적용하는 기술이 필수적이다. 이전부터 연구되어왔던 3D depth map를 얻는 연구들은 낮은 해상도를 가지거나 실시간으로 다루지 못하였다.[1-8] 이것을 위하여 우리는 높은 해상도의 영상에서 depth map를 획득하는 과정에서 CUDA와 OpenMP를 효과적으로 적용하여 실시간으로 처리하였다.

본 논문에서 사용한 구조광을 이용한 깊이지도 획득 기술은 이전부터 많이 연구되어 왔었던 방법이다. 크게 네 가지로 분류할 수 있으며 여기에는 black&white 패턴 [1], gray scale stripe 패턴 [2, 3], color 패턴 [4, 5], color stripe 패턴 [6-8]을 이용한 방법이 존재한다. 이중 우리는 de Bruijn sequence를 기반으로 한 color stirpe 패턴을 이용한 [8]의 방법을 사용 하였다. 높은 해상도의 영상을 사용함에 따라 계산에 소모되는 시간이 증가 할 수 밖에 없기 때문에 알고리즘의 개선만으로는 한계가 있다. 때문에 우리는 각 연산별로 반복되어 수행되는 작업들을 각 부분에 맞게 최적화된 multi-threading 을 통하여 고속화 하였다.

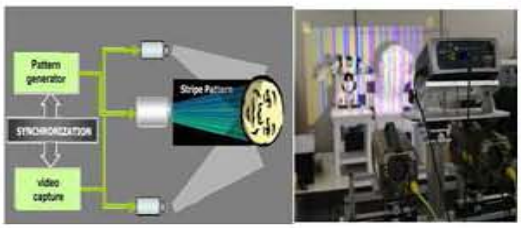


그림 1. 구조광을 이용한 depth map 획득 시스템

본 논문의 구성은 다음과 같다. 2장에서는 우리가 사용한 depth map 획득 시스템에 대하여 설명하고, 3장에서는 각각의 연산들이 수행하는 작업과 해당하는 최적화된 multi-threading 방식을 제안한다. 4장에서는 이를 구현한 실험 결과를 보이며 제안한 방법의 타당성을 증명한다. 마지막으로 5장에서 결론과 앞으로의 연구방향을 정리한다.

2. Depth map 획득 시스템

그림 1은 본 논문에서 사용한 구조광을 이용한 depth map 획득 시스템의 구성도이다. 프로젝터를 통해 패턴 영상을 조사하고, 이를 두 대의 카메라로 촬영한 영상을 통해 depth map를 추출한다. 카메라로

1) 교신 저자

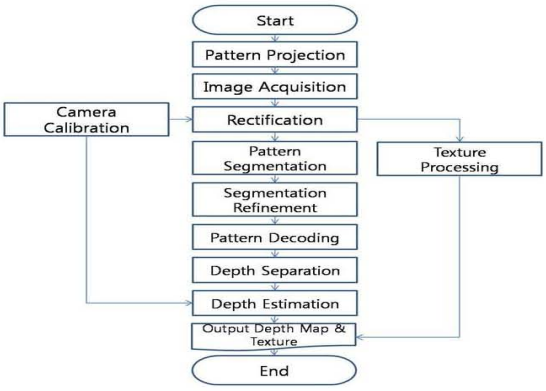


그림 2. Depth map 획득을 위한 수행 과정

획득한 영상은 각 프레임당 1280×800의 크기를 갖는 stereo 영상이다. 전체적인 연산과정은 그림 2와 같다. 큰 영상에서의 데이터 처리를 실시간으로 하기 위해서는 병렬화 기법을 적용하여 고속화 하는 것이 필수적이다. 전체 연산 과정 중 우리가 중점적으로 고속화를 다루는 부분은 image acquisition, pattern segmentation, pattern decoding, depth estimation 부분이며 3장에서 각 연산별로 수행하는 동작과 어떠한 multi-thread가 최적화된 방법인지를 밝히겠다.

3. 가속화 방향

일반적으로 우리가 많이 다루고 잘 알려진 병렬화 기법에는 CUDA와 OpenMP가 있다. CUDA는 GPU를 이용하며 OpenMP는 multi CPU를 이용한다. 기본적으로 병렬화 방법을 택하였을 때는 사용되어진 thread의 수에 비례해서 속도개선이 이루어진다. 때문에 많은 고속화를 다룬 기존의 연구들에서는 thread의 수가 많은 GPU를 이용한 병렬화를 다뤄왔다. 이 두 가지 방법은 비슷한 방법으로 병렬화를 수행하지만 큰 차이점이 존재한다. CUDA는 SIMT(single- instruction, multi-thread) 구조이다. 때문에 분기문이나 반복되는 알고리즘을 다룰 경우 원하는 결과에 한참 못 미치는 결과를 얻게 된다. 그렇기 때문에 우리는 각 연산별로 최적화된 multi-threading을 제안한다.

3.1 Image acquisition

우리가 사용한 de bruijn color sequence를 사용한 depth map 획득 과정에는 좌, 우 시점 각각 3장씩의 영상이 필요하다. 영상을 읽어 들이는 시간도 전체 수행과정의 일부분이기 때문에 좌우를 각각 OpenMP를 통하여 처리 하였다. 처음 시스템이 구동될 때는 좌우 각각 3장씩의 영상을 얻어오며 이후에는 영상이 가지고 있던 memory 공간상에 처음의 영상을 빼고 현재의 영상을 추가로 얻어와 항상 3장씩의 영상이 존재하게 된다.

실험에 앞서 우리는 camera calibration을 수행 하였으며, 이를 통해서 얻어진 map data를 통하여 rectification을 수행하였다. 이는 픽셀대 픽셀로 이뤄지는 단순 계산이기 때문에 CUDA로 수행하게 된다.

이렇게 얻어온 좌, 우 각각 3장의 영상(패턴이 다른 영상)은 그림 4(b)와 같이 각각 1장의 영상으로 합쳐서 textured image를 만들게 된다. Textured image는 ROI 영역을 구하기 위한 과정으로, 이과정도 단순한 계산으로 이뤄져 CUDA에서 효율적으로 고속화 할 수 있다.

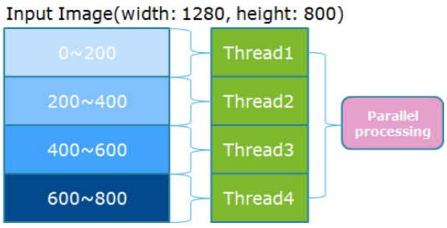


그림 3. OpenMP를 통한 병렬 처리

3.2 Pattern segmentation

Image acquisition을 통해서 얻은 패턴이 존재하는 영상으로부터 패턴을 구분하는 단계이다. 해당 영역에서 각 화소별 정확한 stripe 생상을 검출하기 위해 CIELab 색공간으로 변환하여 화소별 색 성분을 구한다. 검출된 색에 따라 0-red, 1-green, 2-blue, 4-unknown or black 영역으로 인덱스를 부여한다. Segmentation 단계를 수행한 이미지는 그림 4(c)와 같다. Pattern segmentation은 각 화소별로 패턴비트를 검출하는 단순 비교를 통한 과정으로 CUDA를 사용하여 높은 효율을 보인다.

3.3 Pattern decoding

Pattern decoding은 기본적으로 [8]의 방법을 사용한다. 이 과정은 앞에서 구한 segment 된 이미지를 이용한다. 가로로 1차원의 얻어진 sequence와 원래의 sequence를 비교하여 cost function을 계산한다. 두개의 sequence가 같은 경우 score를 증가시키고 이후 최고 score를 가진 값을 결정하게 된다. 이 과정은 상당히 많은 조건문과 이전정보를 참조해야 하는 연산으로 CUDA를 통한 고속화가 어려운 부분이었다. 때문에 우리는 이 연산이 가로로 수행된다는 것을 이용하여 OpenMP를 통해 고속화를 수행하였다. 우리가 사용한 영상이 1280×800 크기를 갖는 영상이기 때문에 이를 세로로 200씩 나누어 4등분하여 4개의 thread를 이용하여 병렬화를 수행하였다(그림 3). 그 결과 CUDA에 비해 더 높은 고속화된 결과를 얻을 수 있었다.

3.4 Estimation

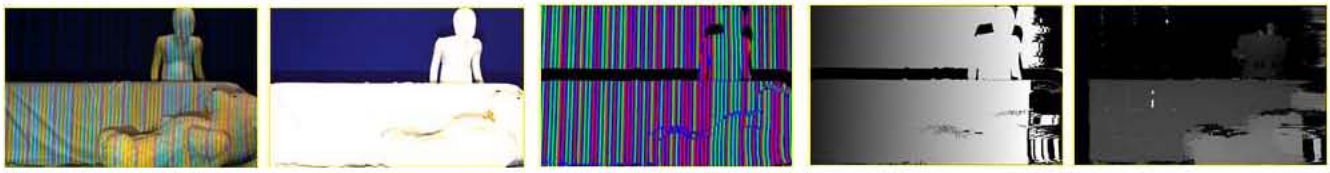
Decoding 된 결과를 통해 depth estimation 하는 과정은 두가지 단계로 나뉘어진다.

먼저 disparity estimation을 수행한다. 이 과정은 decoding 과정과 마찬가지로 가로로 검색하면서 조건부 연산으로 이루어진 연산이다. 때문에 이전과 마찬가지로 세로로 4등분하여 OpenMP로 수행 하였다.

마지막으로 depth map은 앞에서 구한 disparity 값을 통하여 구하게 된다. 실험 이전에 구한 camera projection matrix 값을 이용하여 각각의 depth 값을 구하며 이때 max depth 값을 구한다. Max depth 값은 최종적으로 화면에 나타나게 될 영상의 정규화를 위해 쓰이게 된다. 이때의 연산은 실험적으로 OpenMP를 사용했을때 더 효과적으로 나타났으며 결과는 그림 4(e)와 같다.

4. 실험 결과

여기서는 구현 결과에 대해 설명 한다. 앞에서 언급한대로 각각의 연산에 더 효율적인 병렬처리를 찾아 OpenMP와 CUDA를 사용하여



(a) (b) (c) (d) (e)

그림 4. 구조광을 이용한 depth map 획득 과정의 각 부분별 결과 영상 (a) 구조광이 조사된 입력 영상, (b) Textured image (c) Segmented image, (d) Decoding image, (e) 최종적으로 얻은 depth map image

구현 하였으며, 1280×800 해상도를 가지는 영상을 사용하였다. 실험에 사용된 시스템의 환경은 표 1과 같다.

표 1. 실험 환경

항목	세부사항
CPU	Intel core i7 980 3.33GHz
GPU	GTX 590
Memory	24GB
OS	WINDOWS 7 64bit

표 2는 각 단계 별로 비교한 결과이다. CUDA를 사용한 texture mapping, ROI segmentation, pattern segmentation에 걸리는 시간은 약 10.5ms 로 병렬화를 사용하지 않았을 때와 비교하여 약 530배 정도의 속도 향상이 있었다. Pattern decoding과 estimation 단계는 OpenMP를 사용하였으며 병렬화를 사용하지 않았을 때와 비교하여 약 20배 정도의 속도 향상을 보였다. 이 두과정은 CUDA를 사용했을 때 140ms 보다 7배 정도 향상된 결과를 보여줬다. 전체적으로는 CPU에서 6079ms 걸렸던 것을 multi-thread를 사용함으로써 34.61ms로 약 175배 고속화 성능을 보였다.

표 2. 각 단계별 실험 결과(ms)

단계	CPU	Multi-thread
Image acquisition	17	3.29
Texture mapping + ROI segmentation	5585	7.38
Pattern decoding	411	14.59
Disparity + Depth estimation	66	9.35
Total time	6079	34.61

5. 결론

본 논문에서는 구조광을 이용하여 depth map을 획득하는 방법의 최적화된 고속화 방법에 대하여 제안하였고 구현하였다. 제안한 기법은 depth map정보 획득을 위하여 구조광의 패턴정보를 다루는 과정에서 발생하는 반복적인 연산에 대하여 각각의 수행 동작에 맞는 multi-thread를 사용하여 고속화 하는 것이다. 흔히 사용되어지는 GPU를 통한 병렬 처리 기법은 훌륭한 고속화 방법이다. 하지만 경우에 따라서 GPU에 비해 현저히 낮은 thread를 가진 CPU의 many-core(우리의 실험에선 6개)를 활용하는 것이 더 좋은 결과를 나타낼 수 있음을 보였다. 결과적으로 우리는 병렬화를 수행하지 않았을 때에 비해서 약 175배의 속도 향상을 보였으며, 초당 최소 25 프레임 이상의 속도로 1280×800 해상도에서의 depth map을 획득 하였다.

앞으로 우리는 시간 소모가 가장 큰 pattern decoding에 대해서 좀 더 고속화 할 수 있는 방법을 찾을 것이며, 획득한 depth map의 정확도 향상에 대한 연구를 진행 할 것이다.

감사의 글

본 연구는 방송통신위원회 및 한국방송통신전파진흥원의 방송통신 미디어 원천기술개발 과제의 일환으로 수행되었음. [과제번호 10912-02001, 지상파 양안식 3DTV 방송시스템기술개발 및 표준화]

참고 문헌

[1] K. Sato and S. Inokuchi, "Three-demensional surface measurement by space encoding range imaging," *Journal of Robotic System*, 2:27 - 39, 1985

[2] E. Horn and N. Kiryati, "Toward optimal structured light patterns," *Image and Vision Computing*, 17, (1999).

[3] Daniel Scharstein and Richard Szeliski, "High-accuracy stereo depth map using structured light," *IEEE Computer Society Conferenceon Computer Vision and Pattern Recognition*, Madison, WI, June, (2003)

[4] S. Y. Chen, Y. F. Li, and J. W. Zhang "Vision processing for realtime 3-D data acquisition based on coded structured light", *IEEE Trans. Image Process.*, vol.17, no. 2, pp.167 - 176 , (2008).

[5] Qiang Li, Moyuresh Biswas, Mark R. Pickering, and Michael R. Frater, "Dense depth estimation using adaptive structured light and cooperative algorithm," *Computer Vision and Pattern Recognition Workshops*, June, (2011)

[6] Song Zhang and Peisen S. Huang, "High-resolution, real-time three-dimensional shape measurement," *Optical Engineering*, Vol.45, No.12, (2006)

[7] S. Keerativittayanun, T. Kondo, P. Sira-uksorn, T. Phatrapornant, and M. Sato, "3D data acquisition using active stereo based on spatial neighbourhood technique," *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, May, 17 - 19, (2011)

[8] L. Zhang, B. Curless, and S. M. Seitz, "Rapid shape acquisition using color structured light and multi-pass dynamic programming,"*1st International Symposium on 3D Data processing*, Visualization, and Transmission, Padova, Italy, June, 10-21, (2002).