모바일 GPU 를 이용한 Belief Propagation 스테레오 정합 기법의 병렬화

최호열*, 윤제한**, 박인규*

인하대학교 로봇공학전공*, 삼성전자 DMC R&D Center**, 인하대학교 정보통신공학부*

chlghduf314@gmail.com, jehan.yoon@samsung.com, pik@inha.ac.kr

요 약

본 논문에서는 모바일 GPU(graphics processing units)를 이용한 belief propagation 스테레오 정합(stereo matching) 알고리즘의 병렬화 기법을 제안한다. Belief propagation 스테레오 정합 알고리즘은 많은 계산량과 메모리 사용량이 필요한 알고리즘이다. 본 논문에서는 이러한 특성을 지니는 belief propagation 알고리즘을 스마트폰 환경에서 고속 구동하기 위한 GPU 병렬처리 기법을 제안한다. 제안하는 병렬처리 기법은 최신 스마트폰 단말기에 탑재된 POWERVR SGX540 를 이용하여 구현되었으며 같은 단말기의 CPU 에서의 구현 대비 약 6 배의 속도 향상을 보인다.

1. 서론

최근 입체 영상을 이용한 TV 및 영화의 급속한 보급에 따라, 컴퓨터 비젼 분야에서 고속 스테레오 정합(stereo matching) 알고리즘의 필요성이 재조명되고 있다. 입체 영상제작에 사용되는 조밀한 변위 지도(disparity map)은 픽셀 단위의 연산을 필요로 하기 때문에 영상의 해상도와 변위의 범위에 따라 많은 계산량과 메모리 사용량을 요구한다.

또한 스마트폰의 광범위한 보급에 따라 스마트폰상에서의 영상처리 응용기술의 필요성이 증대되었으나 모바일 프로세서에서의 알고리즘 구현에는 메모리 사용량, 하드웨어 제약사항, 프로그래밍 언어적 제약사항 등을 효과적으로 고려해야 한다. 한편 최근 스마트폰에서는 GPU 의 탑재가 일반화되어 이를 응용 소프트웨어의 구동 프로세서로 이용하고자 하는 가능성이 증대되었다.

본 논문은 스마트폰에서의 3 차원 영상처리를 위한 belief propagation 스테레오 정합 알고리즘의 GPU 병렬처리 기법을 제안한다. 제안하는 병렬처리 기법은 OpenGL ES 2.0 을 이용하여 구현하였으며 최신 모바일 GPU 인 POWERVR SGX540 에 이식되어 동작하도록 한다.

2. OpenGL ES 2.0 Shading Language 를 이용한 belief propagation 스테레오 정합 알고리즘의 병렬화

모바일 GPU 를 이용한 병렬처리를 하기 위해서는 OpenGL ES 2.0 Shading Language 의 programmable pipeline 기능을 사용하여 알고리즘을 작성해야 한다. 내장형 기기의 특성상 매우 한정적인 자원을 효율적으로 사용하여 최대한의 성능을 도출해 내는 것이 관건이다. 본 논문에서 제안하는 병렬처리 기법의 흐름은 다음과 같다.

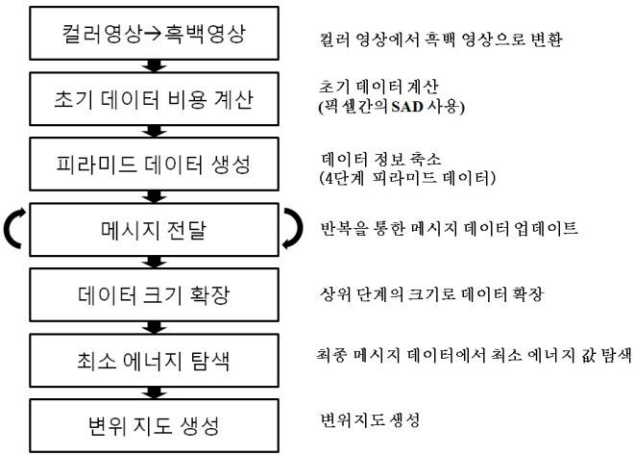

그림 1. 알고리즘의 흐름도

2.1 Shading Language 를 이용한 알고리즘의 병렬화

OpenGL ES 2.0 Shading Language 를 이용하여 알고리즘을 구현하였으며 모든 데이터는 텍스처 형태로 사용한다. 데이터는 rgba 각각 8bit 로 이루어진 텍스처 형태로 사용되었으며 알고리즘의 흐름에 따라 단계별로 작성된 shader program 을 호출하여 픽셀단위 병렬처리를 수행하게 된다. 모든 병렬처리는 fragment shader 에서 다수의 스레드를 발생시켜 수행한다. Belief propagation 스테레오 정합 알고리즘의 경우 각 단계에서의 연산이 데이터 의존성이 적은 특성을 가지고 있어 픽셀단위의 병렬

화에 적합하다. 좌, 우 컬러 영상을 입력으로 각 스레드는 벡터연산을 통하여 픽셀 별로 컬러영상을 gray 영상으로 변환한다. 초기 데이터 비용은 기준 픽셀과 후보픽셀간의 SAD(Sum of absolute differences)를 이용하여 생성하였다. 데이터는 4 장의 텍스처에 rgba 형태로 저장되었으며 피라미드 데이터 생성과 과정은 하드웨어에 의해서 자동으로 처리되었다. 반복적인 메시지 전달 과정에서의 각 스레드는 데이터의 에너지를 수렴시키게 된다. 최종적으로 수렴된 데이터들 중에서 최소의 에너지를 갖는 데이터를 찾아 변위 지도를 생성한다.

2.2 속도 향상을 위한 Shader 최적화

모바일 환경에서의 제한된 자원을 이용하여 최대의 성능을 얻기 위해서는 알고리즘의 구조뿐만 아니라 shader program 의 최적화가 필요하다.

첫째, shader code 의 내부 구조가 간단할수록 좋은 성능을 낸다. 길이가 길어질수록 성능이 저하될 뿐만 아니라 명령어들의 instruction 수가 일정 수준을 넘어가게 되면 프로그램이 실행되지 않는다.

둘째, 적합한 데이터 정밀도(precision)을 사용해야 한다. OpenGL ES 2.0 shading language 는 *highp*, *mediump*, *lowp* 세 가지의 정밀도 한정자를 지원한다. 사용되는 변수 값의 범위에 적합한 정밀도 한정자를 사용하여 효율적인 자원 활용이 필요하다.

셋째, 벡터 연산과 스칼라 연산이 함께 있는 경우 벡터연산의 수를 최소화 하여 사용되는 instruction 의 수를 최소화 한다.

넷째, 텍스처에 대한 접근을 최소화 한다. 동시에 다수의 텍스처에 접근하여 데이터를 읽어오는 경우 급격한 속도 저하가 일어난다.

3. 실험 결과 및 분석

제안하는 기법은 삼성 SHW-M110S(GalaxyS) 스마트폰을 플랫폼으로 하여 구현되었으며 본 기기가 장착한 응용 프로세서 S5PC110 가 내장한 CPU(A8 Cortex 1GHz) 및 GPU(POWERVR SGX540)상에서 구동되었다. 입력 영상은 좌, 우 384×288 크기의 Tsukuba 영상을 사용하였으며 깊이 값의 제약을 16 단계로 설정하였다. CPU 를 이용한 알고리즘은 Android NDK(native development kit)를 이용하여 C 언어로 작성되었으며 고정 소수점 연산화는 수행하지 않았다. GPU 를 이용한 알고리즘은 OpenGL ES 2.0 Shading Language 로 작성되었다. OpenGL ES 2.0 의 FBO(frame buffer object)개수의 제한으로 4 단계의 피라미드로 구현되었으며 실시간에 준하는 성능을 내기 위하여 0~4 단계 중에 1~3 단계의 메시지 전달이 수행되었다.

표 1. 제안된 알고리즘의 성능분석 (단위: 초)

Iteration 수	CPU	GPU	가속비율
2	4.152	0.578	×7.18
4	6.976	1,086	×6.42
8	12.161	2,083	×5.84
12	17.413	3,125	×5.57
20	27.889	5,263	×5.23

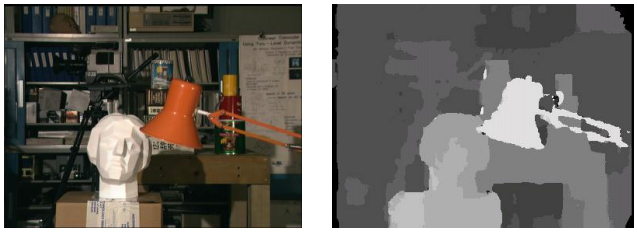

그림 2. 알고리즘 수행 결과 (15 회 메시지 전달 반복) (a) Tsukuba 입력 영상 (b) 생성된 변위 지도

4. 결론

본 논문에서는 모바일 환경에서의 조밀한 스테레오 정합을 하는데 있어서 belief propagation 알고리즘을 사용하였으며 실시간에 근접하는 속도를 얻기 위하여 GPU 를 이용한 알고리즘의 고속화를 수행하였다. 실험결과는 CPU 대비 약 6 배의 속도 향상 효과가 있었다. 모바일 환경과 OpenGL ES 2.0 shading language 는 프로그래머에게 제한된 기능과 한정된 자원 만을 제공해 준다. 이러한 점을 고려하여 효율적인 알고리즘의 설계가 필요하다.

본 논문의 결과로 보아 향후 모바일 환경에서의 GPGPU 사용이 증대될 것이며 알고리즘 고속화에 효과적인 역할을 할 것이라고 예상한다.

감사의 글

본 연구는 삼성전자(주)의 지원을 받아 수행되었음.

참고문헌

[1] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient belief propagation for early vision,” *In Proc. International Journal of Computer Vision*, vol. 70, no. 1, pp. 41-54, October 2006

[2] Q. Yang and L. Wang, “Real-time global stereo Matching Using Hierarchical Belief Propagation,” *In Proc. British Machine Vision Conference*, pp. 989-998, 2006.

[3] Imagination Technologies Ltd, *POWERVR SGX Factsheet*, 2008.

[4] A. Munshi, D. Ginsburg, and D. Shreiner, *OpenGL ES 2.0 Programming Guide*, 1st Edition, Addison Wesley, 2009.