

고화질 영상의 다중시점 샘플링을 활용한 실시간 다시점 깊이지도 생성

김선호, 호요성
광주과학기술원 전기전자컴퓨터공학부
sunhokim@gist.ac.kr, hoyo@gist.ac.kr

Real-Time Multi-View Depth Map Generation Using High-Definition Image Downsampling

Sunho Kim, Yo-Sung Ho
School of Electrical Engineering and Computer Science
Gwangju Institute of Science and Technology (GIST)

요 약

실감나는 3차원 콘텐츠를 만들기 위한 방법 중 하나로 많은 시점의 색상 영상과 깊이지도를 활용하는 방법이 사용된다. 그러나 이러한 콘텐츠를 생성할 때 고화질의 영상을 사용하게 된다면 보다 많은 정보를 갖는 영상을 활용하게 되므로 알고리즘의 처리 속도가 매우 느려지게 된다. 본 논문에서는 여러 개의 시점의 색상 영상에 대응되는 다시점 깊이지도를 생성하는 방법에 대해서 설명한다. 이 방법을 수행할 때, 본 논문에서는 다수의 GPU를 활용해 연산을 처리하고, CPU와 GPU간의 영상 전송 속도 단축을 위해 영상의 다운샘플링 방법을 활용한다. 실험 결과에서는 제안된 방법이 실시간에 가까운 속도로 다시점 깊이지도를 생성할 수 있음을 결과 화면과 소요 시간을 통해 보여준다.

1. 서 론

영상의 기술이 발전하면서 이전보다 실감나는 영상 콘텐츠에 대한 요구는 지속적으로 증가하고 있다. 그 중에서 3차원 콘텐츠를 즐기기 위한 안경식 혹은 무안경식 디스플레이는 이미 우리가 많은 곳에서 접할 수 있다. 뿐만 아니라 사용자가 원하는 시점에 대한 영상을 관람할 수 있는 자유 시점 디스플레이에 대한 기술도 지속적으로 연구되고 있고 [1], 나아가서 증강현실과 가상현실에 대한 관심도 지속되고 있다. 특히 증강현실과 가상현실은 정부에서 9대 국가전략 프로젝트 중 하나로 선정하기도 했다.

이와 같은 실감 콘텐츠를 제작하기 위해서는 다수의 시점에 대한 영상뿐만 아니라 이에 상응하는 깊이지도까지 요구된다. 깊이지도를 만들어내는 방법에는 여러 시점의 색상 영상을 기반으로 한 스테레오 정합 방법을 사용할 수도 있다. 그러나 고화질의 영상에 대해 보다 빠르게 깊이 지도를 생성하기 위해서는 깊이 카메라를 통해 촬영된 결과를 직접 활용해 해당 시점에 대한 깊이지도를 생성하는 방법이 보다 효율적이다.

최대한 실시간에 가깝게 다시점 깊이지도를 생성하는 방법 중 하나로 GPU 프로그래밍과 원형 큐를 사용하는 방법이 제안되었다 [2]. 기존의 방법을 통해서 18개의 시점에 대한 깊이지도를 생성할 경우, 평균 26fps 정도의 속도가 나옴을 확인할 수 있었다. 그러나 최대한 실시간에 가깝게 고화질에 대한 깊이지도를 생성하기 위해서는 이보다 빠른 속도의 알고리즘이 요구된다.

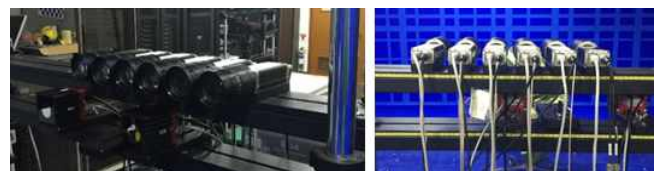
본 논문에서는 기존의 방법보다 빠른 속도로 다시점 깊이 지도를 생성하는 방법에 대해서 제안한다. 기존에 있던 3차원 워핑과 이중 선형 보간법을 이용한 깊이지도 업샘

플링 방법을 기반으로 이 알고리즘의 속도를 개선한다. 메모리를 관리하는 단계에서는 원형 큐 방식이 아닌 비동기식 멀티쓰레딩 방법을 활용해 보다 빠르게 영상을 CPU에서 GPU로 전송한다. 또한 제안된 방법에서는 이러한 영상 전송 과정을 단축시키기 위해 다운샘플링을 통해 보다 적은 전송량을 가지고 영상을 GPU로 전송하도록 한다. 실험 결과에서는 제안된 방법을 통해 얻은 결과가 기존에 제시되었던 방법과 유사한 품질의 깊이지도를 기존 방법에 비해 얼마나 빠르게 얻을 수 있는지를 보여준다.

2. 기존의 다시점 깊이지도 생성 방법

2.1 촬영 환경

본 논문에서는 다시점 깊이지도 생성에 필요한 영상을 촬영하기 위해 그림 1과 같은 촬영 환경을 하나의 집합으로 하여 총 세 개의 카메라 집합으로 구성했다. 촬영 환경에서는 18대의 색상 카메라(Basler Pilot piA1900-32gc GigE)와 3대의 ToF 카메라(Mesa Imaging SR4000)를 사용했다. 배치된 3대의 ToF 카메라의 촬영 결과물은 각각 6개의 색상 카메라에 대응되는 깊이지도 생성에 사용된다.



(그림 1) 실시간 다시점 깊이지도 생성을 위한 촬영 환경

즉, ToF 1번은 Color 1번부터 6번까지, ToF 2번은 Color 7번부터 12번까지, 그리고 ToF 3번은 Color 13번부터

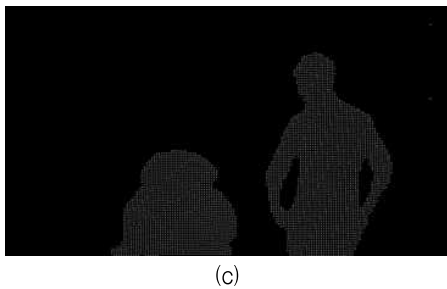
터 18번까지에 대한 깊이지도를 생성한다. 구성된 촬영 환경 내에서 각각의 카메라에 대한 내부 및 외부 파라미터를 구하기 위해 카메라 캘리브레이션 작업을 거치고, 이를 통해 얻은 파라미터를 깊이지도를 만드는 데에 활용된다.

2.2 깊이지도 업샘플링 과정

ToF 카메라를 통해 얻은 깊이지도는 하나의 시점 당 6개의 색상 영상 시점에 대응되는 깊이지도로 변환해야 한다. 깊이지도의 업샘플링을 위한 첫 단계로 3차원 워핑 연산을 수행한다. 3차원 워핑에서는 ToF 카메라와 다시점 색상 카메라의 내부 및 외부 파라미터 행렬을 식 (1)에 대입하여 원하는 시점에 대응되는 위치로 값을 복사한다. 이 과정은 카메라 캘리브레이션 결과를 활용한다.

$$\begin{aligned} m_s &= A_s \cdot R_s \cdot M_W + A_s \cdot t_s \\ M_W &= R_s^{-1} \cdot A_s^{-1} \cdot m_s - R_s^{-1} \cdot t_s \\ m_t &= A_t \cdot R_t \cdot M_W + A_t \cdot t_t \end{aligned} \quad (1)$$

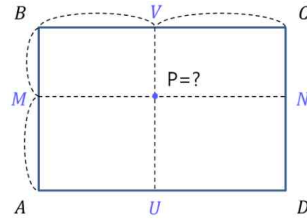
식 (1)에서 A 는 카메라의 내부 파라미터, R 은 회전 행렬, t 는 전이 벡터를 나타낸다. 입력 영상과 관련된 변수에는 첨자 s 를, 결과물과 관련된 변수에는 첨자 t 를 붙였고, 3차원 월드 좌표계와 관련된 변수에는 첨자 W 를 붙였다. 깊이지도에 있는 픽셀 값인 m_s 가 위 공식을 통해 3차원 월드 좌표계인 M_W 로 이동하고, 다시 이 값이 공식을 통해 우리가 얻고자 하는 시점에 대한 공간으로 이동하여 m_t 에 저장된다.



(그림 2) 3차원 워핑 결과

그림 2는 3차원 워핑 연산을 수행한 결과를 나타낸다. 그림에서도 알 수 있듯이, 색상 영상보다 작은 해상도에 있는 값들을 옮겨주었기 때문에 원본과 같은 형태의 객체로 복원하기 위해서는 이러한 빈 공간에 대해 보간을 수행해야 한다. 본 논문에서는 이를 위해 이중 선형 보간법을 사용했다. 이 방법에 대한 설명은 그림 3과 식 (2)를 통해 확인할 수 있다.

$$\begin{aligned} P &= \frac{C-V}{C-B} \left(\frac{B-M}{B-A} \times A + \frac{M-A}{B-A} \times B \right) \\ &+ \frac{V-B}{C-B} \left(\frac{B-M}{B-A} \times D + \frac{M-A}{B-A} \times C \right) \end{aligned} \quad (2)$$



(그림 3) 이중 선형 보간법

특정 빈 공간의 픽셀 값을 추정하기 위해 워핑을 통해 얻었던 픽셀 값들을 활용한다. 이중 선형 보간법에서는 가로 방향과 세로 방향에 대한 선형 보간법을 수행한 뒤, 각각을 통해 얻은 결과 값들인 M, N, U, V 를 바탕으로 최종적으로 얻고자 하는 P 값을 구한다. 3차원 워핑을 통해 얻은 결과물은 해상도 차이가 큰 깊이지도로 인해 얻은 결과물이므로, 한 차례 보간법으로 빈 공간을 채우기에는 부족함이 있다. 따라서 이 과정에서는 7차례 혹은 그 이상의 보간 과정을 수행해서 모든 빈 공간을 채워준다.

단순한 보간 방법을 통해서 얻은 결과물은 객체의 경계 부분이 매끄럽지 못하고, 곳곳에 미처 채워지지 못한 홀이 발생할 수 있다. 따라서 이런 부분들을 보완하기 위해 본 논문에서는 결합 양방향 필터링을 활용한다 [3].

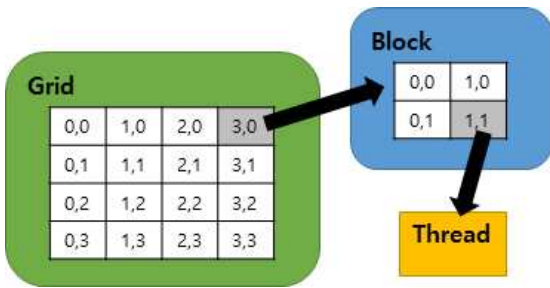
본 논문에서는 최종적으로 얻은 깊이지도에 대해서 객체의 깊이 정보만을 활용한다. 따라서 불필요한 배경 정보를 제거해줄 필요가 있다. 이를 위해 각각의 색상 카메라를 통해 촬영한 배경 영상을 기반으로 배경 제거 연산을 수행하고, 이 연산을 통해 얻은 객체 마스크를 기반으로 배경의 깊이 정보를 제외한다. 색상 영상의 배경을 제거하는 방법에는 RGB 색상 공간을 HSV 색상 공간으로 변경한 뒤, 색상과 채도의 차이로 정보들을 제거하는 방법을 사용한다 [4].

3. GPU 프로그래밍을 통한 실시간 프로세싱

앞서 소개한 연산을 18개 시점의 고화질 영상에 대해서 동시에 수행할 경우, 굉장히 오랜 시간이 소요된다. 따라서 실시간 다시점 콘텐츠를 생성하기 위해서는 이러한 속도 문제를 개선할 필요가 있다. 본 논문에서는 CPU 내에 있는 다수의 스레드와 3개의 GPU를 활용해 보다 빠른 속도로 연산을 수행할 수 있도록 개선했다. 본 논문에서는 NVIDIA GeForce GTX 1080 GPU를 사용했다.

3.1 CUDA GPU 프로그래밍

제한된 방법에서 다수의 GPU를 활용하기 위해서는 별도의 GPU 프로그래밍 방법이 요구된다. 일반적으로는 CUDA(Compute Unified Device Architecture)를 이용해서 GPU 프로그래밍을 구현한다. C 프로그래밍 언어를 기반으로 만들어진 CUDA는 그래픽카드의 API에 관련된 지식과 같은 복잡한 지식 없이도 쉽게 GPU 커널에 접근해서 프로그래밍을 할 수 있게 해주는 방법이다 [5].

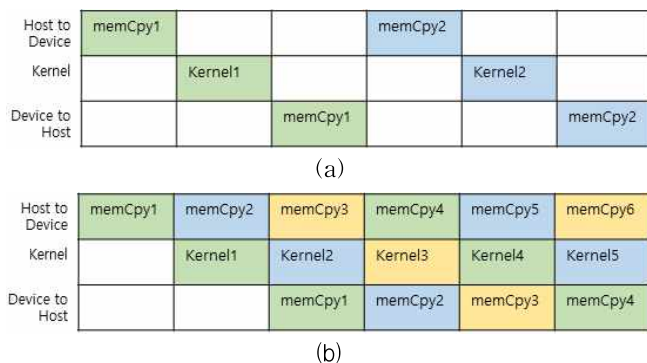


(그림 4) CUDA 프로그래밍의 동작 단위

그림 4는 CUDA 프로그래밍의 동작 단위에 대한 설명을 보여준다. GPU 내에는 많은 수의 쓰레드가 존재하고, 이러한 쓰레드가 모여 하나의 블록을, 그리고 그 블록들이 모여 하나의 그리드를 형성한다. 영상 처리에서 이 방법을 사용할 경우, 쓰레드 단위로 각각의 픽셀에 대해 병렬적으로 연산을 할 수 있기 때문에 단순히 CPU만을 사용한 방법에 비해 빠른 연산을 수행할 수 있다. 제안된 방법은 깊이 지도를 생성하는 알고리즘을 GPU내에서 수행한다.

3.2 비동기식 멀티쓰레딩 방법

높은 해상도를 갖는 색상 마스크를 CPU에서 GPU로 전송할 경우, 일정 수준 이상의 시간이 소요된다. 따라서 이러한 영상 전송 속도를 단축시키는 작업 또한 실시간 다视点 깊이 지도 생성에 있어 중요한 화두다. 제안된 방법에서는 CPU 멀티쓰레딩 방법을 통해 많은 양의 영상들을 전송하는 시간을 단축시켜준다. 멀티쓰레딩 방법을 사용할 수 있게 해주는 라이브러리 중 하나로 Boost 라이브러리가 있다 [6].



(그림 5) 비동기 함수를 통한 메모리 복사 방법

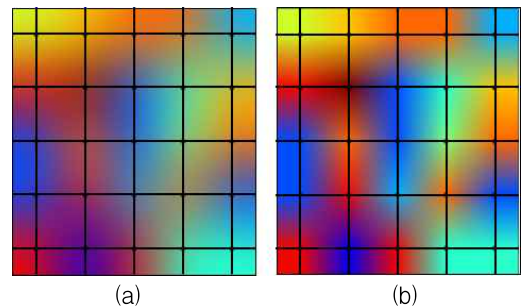
그러나 단순히 멀티쓰레딩 방법만 사용하기에는 메모리 복사 과정에서 걸리는 시간이 많이 단축되지 않는다. 그림 5의 (a)에서처럼 하나의 메모리 복사가 다 끝나는 시점에서야 다음 메모리 복사를 시작하기 때문이다. 제안된 방법에서는 CUDA 내에 있는 비동기 함수를 통해 그림 5의 (b)와 같은 방식으로 메모리를 복사한다. 이 방법에서는 첫 번째 메모리가 복사된 뒤 GPU 커널 내에서 연산을 수행하는 동시에 바로 다음 메모리 복사 작업을 시작한다. 이 방법을 통해 메모리 복사의 시점이 보다 앞당겨져 보다 빠른 메모리 복사 작업을 수행할 수 있다.

3.3 마스크 영상의 해상도 조정 방법

색상 영상의 시점에 맞는 깊이 지도를 만들기 위해 CPU에서는 깊이 지도의 원본과 각 색상 영상들의 객체 마스크를 GPU로 전송한다. GPU로의 전송 속도는 객체 마스크의 해상도에 따라 그 영향을 미치기 때문에 보다 작은 영상일수록 빠르게 GPU로 전송할 수 있다. 제안된 방법에서는 마스크 영상을 CPU에서 우선적으로 1/4배로 다운샘플링을 한 뒤에 GPU로 전송하고, 다운샘플링된 영상을 다시 원본 해상도 크기만큼 업샘플링을 하는 과정을 거친다.

마스크 영상의 다운샘플링의 경우 해상도가 작아져도 원본과 같은 형태의 객체가 유지된다. 그러나 이를 다시 업샘플링하기 위해서는 최대한 원본 마스크의 형태와 유사하게 복원을 시켜야만 한다. 제안된 방법에서는 이를 위해 이중 삼차함수 보간법을 활용한다.

삼차함수 보간법은 스플라인 보간법의 한 종류로, 두 픽셀을 이은 곡선에 대한 3차 다항식으로 매끄러운 형태의 결과물을 얻는 방법이다. 제안된 방법에서는 이를 2차원으로 확장한 이중 삼차함수 보간법을 통해 GPU내에서 마스크 영상을 원본에 가깝게 복원하는 알고리즘을 구현한다.



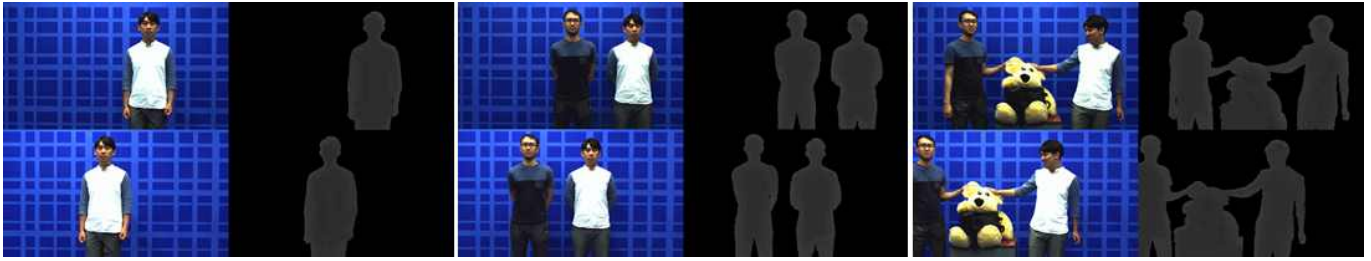
(그림 6) 이중 삼차함수 보간법과 이중 선형 보간법 비교

그림 6은 이중 삼차함수 보간법과 이중 선형 보간법을 수행한 결과를 비교한 모습이다. 그림 6의 (a)는 이중 삼차함수 보간법, (b)는 이중 선형 보간법의 결과다. 결과에서도 알 수 있듯이, 이중 삼차함수 보간법이 이중 선형 보간법보다 색의 경계 부분에서 작은 변화를 일으키며 보다 부드럽게 변화함을 볼 수 있다. 객체 마스크 영상의 경우 원본에 최대한 가까운 형태로 복원이 돼야 만들어진 깊이 지도에서 객체 부분만을 올바르게 잘라낼 수 있다. 따라서 제안된 방법에서는 이중 선형 보간법에 비해 세밀한 부분을 잘 살려줄 수 있는 이중 삼차함수 보간법을 택했다. 이 방법을 통해 얻은 결과를 바탕으로 영상 이진화를 한 차례 수행하고, 최종적으로 원본 객체 마스크와 가장 유사한 결과를 얻어 깊이 지도 생성 과정에서 활용을 한다.

4. 실험 결과

본 논문에서는 기존에 제안되었던 원형 큐를 활용한 방법과 제안된 방법의 결과를 비교한다. 실험에 앞서 각 카메라들의 파라미터를 얻기 위해 카메라 캘리브레이션 작업을 거쳤고, 이를 통해 얻은 카메라 파라미터를 3차원 워핑 작업에서 활용했다.

해당 실험에서는 6대의 색상 카메라와 1대의 ToF 카메라



(그림 7) 실시간 다시점 깊이지도 생성 결과

라를 하나의 집합으로 간주하고 총 세 개의 집합을 촬영 환경으로 구성했고, 연산을 위해 총 세 개의 GPU를 활용하는 실험 환경을 조성했다. 이 세 대의 GPU는 한 집합의 카메라들에 대한 깊이지도 생성 과정을 각각 수행함으로써 최대한 실시간에 가까운 속도로 18개의 시점의 깊이지도를 생성할 수 있도록 했다.

그림 7은 제안된 방법을 통해 다시점 깊이지도를 생성한 결과물을 나타낸다. 제안된 방법과 기존의 방법을 비교했을 때, 깊이지도의 정확도 면에서는 두 방법 모두 원본과 유사함을 보였다. 그림 7에 나와 있듯이 해당 실험에서는 세 개의 다른 시퀀스를 통해 그 결과를 얻었다. 첫 번째 시퀀스에서는 하나의 객체에 대한 깊이지도 생성을, 두 번째와 세 번째 시퀀스에서는 각각 두 객체와 세 객체에 대한 깊이지도 생성을 수행했다. 이는 객체의 수에 따라 처리하는 양이 달라지기 때문에 속도에 얼마나 많은 영향을 끼치는지를 알아보기 위함이다.

(표 1) 실시간 다시점 깊이지도 생성 속도 비교 (단위 : fps)

	Sequence 1	Sequence 2	Sequence 3
Circular Queue	30.293	27.882	22.035
Multi GPU Only	34.881	31.228	27.232
Multi GPU + Resize	36.339	33.001	31.533

표 1은 세 개의 시퀀스를 통해 실험을 했을 경우에 대한 속도 비교 결과들을 나타낸다. 먼저 기존에 제안되었던 원형 큐를 사용하는 방법을 통해 메모리를 관리하는 방법에 대해 실험을 수행했고, 다음으로 다수의 GPU를 통해 메모리를 관리하는 방법에 대해 수행했다. 다수의 GPU를 활용했을 때, 원형 큐를 사용하는 방법에 비해 4~5 fps 정도 시간이 빨라짐을 확인할 수 있었다.

다음으로 다수의 GPU만을 사용한 방법과 본 논문에서 제안된 다수의 GPU 방법과 객체 마스크 해상도 조정을 함께 수행한 방법을 비교했다. 이는 영상의 해상도 조정 방법이 영상의 전송 속도에 얼마나 영향을 끼치는지 알아보기 위함이다. 기존에 제안되었던 원형 큐 방법과 다수의 GPU만을 사용한 방법에서는 이러한 객체 수에 따라 속도의 차이가 보다 크게 발생함을 보여준다. 반면에 객체 마스크의 해상도 조정 방법을 사용했을 때는 줄어든 마스크 영상을 GPU로 전송해줄기 때문에 객체의 수가 많아져도 전송해야 할 픽셀의 수가 상대적으로 적게 증가하기 때문에 전반적으로 속도가 증가했을 뿐만 아니라 객체 수 대

비 속도의 증감 폭이 훨씬 적음을 확인할 수 있었다.

5. 결 론

본 논문에서는 실시간에 가까운 속도로 많은 시점의 고화질 색상 영상에 대응되는 깊이지도를 생성하는 방법을 제안했다. CPU와 GPU를 모두 활용하기 위해 CUDA를 사용했고, boost 라이브러리에 기반을 둔 멀티쓰레딩 방법을 활용했다. 이와 같은 환경을 통해 기존에 있던 깊이지도 업샘플링 알고리즘을 개선했고, 실시간으로 결과물을 얻을 수 있게 되었다.

감사의 글

본 연구는 미래창조과학부 ‘범부처 Giga KOREA 사업’의 일환으로 수행하였음. [GK16C0100, 기가급 대응량 양방향 실감 콘텐츠 기술 개발]

참고 문헌

- [1] Joel Carranza, Christian Theobalt, Marcus A. Magnor, Hans-Peter Seidel, "Free-viewpoint video of human actors," ACM transactions on graphics (TOG). Vol.22 No.3, July 2003. pp.569-577.
- [2] Eunsang Ko, Yunseok Song, Yo-Sung Ho, "Real-time Multi-view Depth Generation Using CUDA Multi-GPU," International Conference on Embedded Systems and Intelligent Technology (ICESIT), September 2014, pp.114-116.
- [3] Johannes Kopf, Michael F. Cohen, Dani Lischinski, Matt Uyttendaele, "Joint bilateral upsampling," ACM Transactions on Graphics (ToG), Vol.26 No.3, August 2007, pp.96-100.
- [4] Jang-Hee Yoo, Doosung Hwang, Jong-Wook Han, Ki-Young Moon, "Extracting Human Body based on Background Estimation in Modified HLS Color Space," World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, Vol.3 No.11, September 2009. pp.1942-1945.
- [5] Shane Cook, "CUDA programming: a developer's guide to parallel computing with GPUs," Newnes, December 2012.
- [6] <http://www.boost.org/>