GPU와 CPU의 병렬처리를 이용한 실시간 3D 모델링

백운혁[O] 경동욱 한은정 양종렬 정기철
숭실대학교, IT대학, 미디어학과, HCI Lab.
{bwhyuk[O], kiki227, hanej, yjyhorse, kcjung}@ssu.ac.kr

Real-time 3D Modeling using GPU and CPU in parallel processing

Woonhyuk Baek[O], Dongwuk Kyoung, Eunjung Han, Jongyeol Yang, Keechul Jung
HCI Lab., School of Media, College of IT, Soongsil University

요 약

3D 모델링 기술은 가상현실, 실감형 인터랙티브 등에서 많은 연구가 진행되고 있다. 실시간 3D 모델을 생성하는 연구는 많은 계산량으로 인해서 여러 대의 PC를 통합한 PC클러스터를 사용하고 있다. PC클러스터는 여러 대의 PC를 하나의 고성능 컴퓨터로 처리가 가능하지만, 여러 대의 PC를 효율적으로 제어 하는 문제와 고비용의 문제를 안고 있다. 본 논문은 한 대의 PC에서 멀티 코어를 동시에 수행하는 병렬처리 방법과 높은 계산 능력을 자랑하는 GPU와 CPU의 병렬처리 방법을 사용하여 한 대의 컴퓨터로 실시간 3D 모델 생성방법을 제안한다.

1. 서 론

3D 모델링 기술은 3차원 방송, 다시점 비디오, 가상현실, 증강현실, 그리고 실감형 인터랙티브 환경 등 여러 분야에 이용되고 있는 기술이다[1-2].

다양한 환경에서 사용되는 3D 모델링 기술은 실시간을 바탕으로 사용자의 모델을 생성하거나[3-4], 생성된 모델에 텍스쳐 매핑(texture mapping)을 수행하여 실제 분야에 사용되고 있다[4-5]. 이때 실시간영상을 얼마나 신속 정확하게 3D 모델을 생성할 것인가가 중요하다. 현재 빠른 속도로써 3D 모델을 생성하기 위한 기술은 하드웨어 상의 PC 클러스터 기술과 소프트웨어 상의 병렬처리 등으로 나눈다. PC 클러스터는 저사양의 PC 여러 대를 고성능의 PC로 사용가능하게 하는 기술로써, 3D 모델을 생성하는 연구에서 많이 사용하고 있다[6]. 그리고 PC 클러스터 상에서 수함으로서[7] 빠른 처리속도로 처리가 이루어지지만 PC 클러스터는 여러 대의 PC를 사용함으로써, 각 PC의 효율적인 제어문제와 많은 이미지 데이터를 효율적으로 각 PC 간에 전달을 요하는 네트워크 문제 등이 있다.

본 시스템은 한 대의 컴퓨터에서 싱글 및 멀티 코어를 멀티 스레드를 이용하여 효율적인 3D 모델을 생성하는 연구와 계산능력이 탁월한 GPU를 CPU와 병렬처리를 수행하여 멀티 코어를 사용했을 때 보다 빠른 수행을 처리하는 방법을 소개한다.

2. 다시점 카메라를 이용한 3D 모델링

본 연구는 다시점 카메라를 이용하여 3D 모델을 생성하는 연구로써, CPU와 GPU의 병렬처리로 수행속도를 향상시키는 연구이다.

다시점 카메라를 이용한 3D 모델 생성을 위한 연구는 크게 2가지로 분류 된다. 첫 번째는 입력 영상을 제공하는 카메라의 제어와 카메라 보정(camera calibration)이며, 두 번째는 다시점 카메라에서 제공되는 입력영상으로 3D 모델을 정확하게 그리고 빠르게 생성하는 것이다.

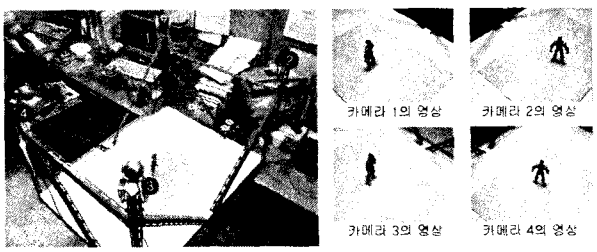

그림 1. 4대의 카메라를 이용한 3D 모델 구축 시스템

2.1 시스템의 초기화 과정

본 논문에서는 4대의 USB 카메라를 사용하여 입력 영상을 획득하며, 카메라의 영상을 쉽게 획득하기 위해서 OpenCV(Open Source computer Vision) 라이브러리를 사용한다[8]. OpenCV는 인텔사의 실시간 컴퓨터 영상 프로그램 라이브러리로써, 객체, 얼굴, 행동 인식, 모션 추적 등에 많이 사용되며, 다수의 카메라의 입력영상을 제어 및 영상을 제공한다. 그리고 영상에서 물리적인 거리를 측정하기 위해서는 카메라의 렌즈와 센서 간의 관계를 파악하는 카메라 내부 파라미터(intrinsic parameters)를 알아내야하며, 카메라 좌표계와 우리가 기준으로 잡고 있는 좌표계(world coordinate)사이의 관계인 외부 파라미터(extrinsic parameter)를 파악해야 한

다. 이때 본 연구에서는 카메라 보정을 위해서 Graphics and Media Lab.에서 제공하는 GML(GML Camera Calibration Toolbox)[9]을 사용하였다. GML은 카메라를 통해서 패턴 이미지를 캡쳐하여 각 사각형의 모서리를 이용하여 내부 및 외부 파라미터를 계산한다(그림 2).

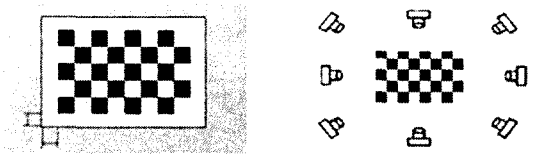

그림 2. 패턴 이미지를 이용한 카메라 보정

파라미터는 m (image projection), R (rotation matrix), M (3D point), A (camera intrinsic matrix) 이다.

$$m = A[R,t]M$$

$$R=\begin{bmatrix} r_{11} r_{12} r_{13} \\ r_{21} r_{22} r_{23} \\ r_{31} r_{32} r \end{bmatrix} \quad A=\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad t=\begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

2.2 다시점 카메라를 이용한 3D 모델 생성 알고리즘

본 연구에서는 3D 모델을 생성하기 위해서 4가지 프로세스를 수행한다. 첫 번째 프로세스는 카메라 입력영상에서 객체(object)와 배경(background)로 나누는 실루엣 추출과정이고, 두 번째 프로세스는 실루엣 추출 이미지에서 3D space 영역으로 이미지 변환, 세 번째 프로세스는 플랜 이미지의 정합 이미지 생성과정이다. 마지막으로 생성된 정합 이미지에서 외각점을 추출하고, 외각점을 기반으로 Voxel 데이터를 만드는 과정이다.

2.2.1 카메라 캡쳐 영상에서의 실루엣 추출

실루엣 추출은 객체와 배경을 분류하는 과정으로써, 입력 받은 영상에서 객체의 영역을 찾기 위해 배경이미지와 차영상을 하게 된다. 이때 보다 빠르고 효과적으로 물체의 영역을 찾기 위해 다음과 같은 방법으로 분류한다. 주어진 background image의 $(i,j)^{th}$ pixel의 색은 RGB벡터 $C_b(i,j)$을 나타내고, 입력 영상의 $(i,j)^{th}$ pixel의 색은 RGB벡터 $C_r(i,j)$을 나타낸다. 이렇게 나타낸 두 벡터 (C_r, C_b)사이의 거리와$(distance)$ 차이$(difference)$를 이용해 물체의 영역을 찾는다(그림 3).

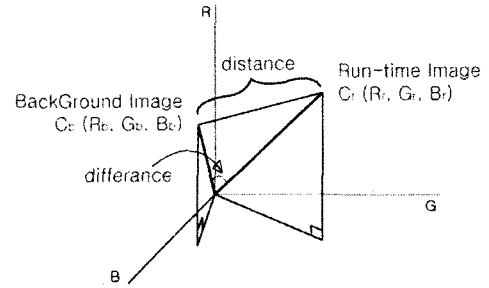

그림 3. RGB 벡터의 표현

이와 같은 정보를 이용하여 배경과 객체를 분리한 결과 영상은 다음과 같다(그림 4).

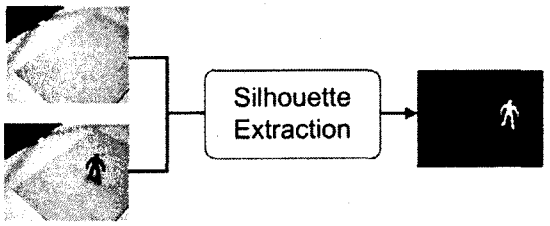

그림 4. 차영상을 이용한 실루엣 추출

2.2.2 3D space 상의 플랜 이미지 생성

3D space상의 이미지 변환 과정은 실루엣 이미지에서 실제 3D space상의 영역으로 변환하는 것을 말한다. 즉, 각 카메라의 보정 파라미터를 통해서 3D space상으로 변환이 이루어진다(그림 5). 식 (1)은 z=0일 때 플랜 이미지를 생성하는 식이다. 이때 사용되는 변환행렬(H_0)는 식 (2)과 같다. 각 Z축에 따른 변환행렬(H_z)는 다음 식 (3)과 같이 생성된다.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A[r_1 r_2 r_3 t]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = A[r_1 r_2 t]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad (1)$$

$$H_0 = \begin{bmatrix} h_{11} h_{12} h_{13} \\ h_{21} h_{22} h_{23} \\ h_{31} h_{32} h_{33} \end{bmatrix} = A[r_1 r_2 t] \quad (2)$$

$$H_z = \begin{bmatrix} h_{11} h_{12} h_{13} \\ h_{21} h_{22} h_{23} \\ h_{31} h_{32} h_{33} \end{bmatrix} = A[r_1 r_2 (r_3 \times z + t)] \quad (3)$$

그림 5는 H_0의 변환행렬을 사용하여 변환한 결과 이미지 이다.

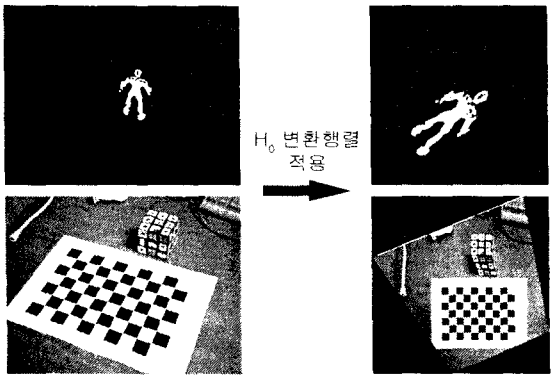

그림 5. 변환행렬을 이용한 플렌 이미지 생성

2.2.3 변환 이미지 정합 및 외각점 추출

변환 이미지의 정합은 각 카메라의 3D space 상의 평면으로 변환된 이미지를 z축의 값이 같은 이미지를 정합하여 3D space상의 3D 모델을 생성한다(그림 6).

정합된 결과 이미지를 z축으로 복셀 간격으로 생성하였을 때 3D 모델이 생성된다. 생성된 각 정합 이미지에서 외각점을 추출 하였을 때 3D 모델의 복셀 데이터로써 사용된다.

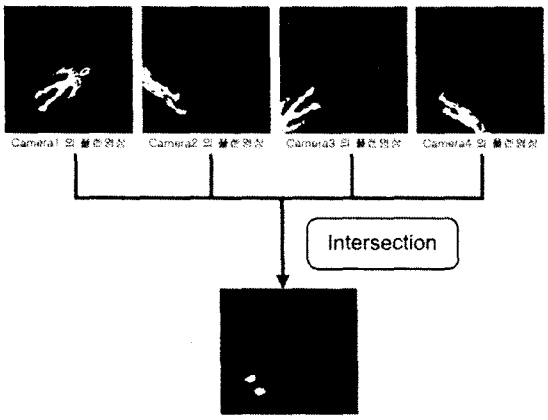

그림 6. 병합 이미지 정합과정

3. CPU 및 GPU의 병렬처리

본 시스템은 다중 코어 CPU 및 GPU의 병렬 처리를 사용하여 기존의 많은 리소스를 사용하는 대신에 한대의 컴퓨터만으로 실시간 3D 모델링 방법에 대해서 제안한다.

3.1 CPU의 병렬 처리

기존의 CPU는 하나의 코어(single-core)로써 명령을 수행하였지만, 최근의 CPU의 발전으로 멀티 코어(multi-core)로써 물리적으로 두개 이상의 CPU가 서로 독립적으로 동시에 두개 이상의 명령어를 처리 할 수 있게 되었다. 멀티 코어를 이용해 여러 대의 PC로 나누어 작업하던 것을 한대의 PC에서 모든 프로세스를 진행 할 수 있도록 하였다(그림 7).

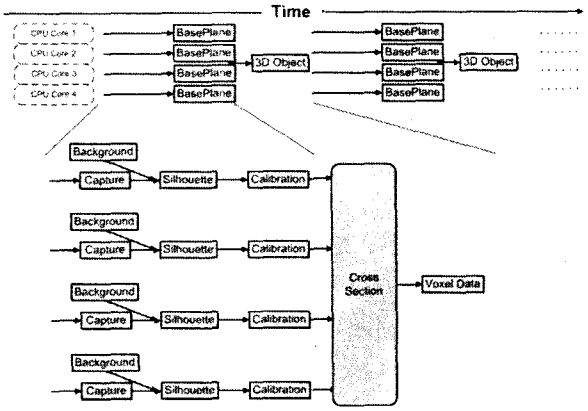

그림 7. 멀티 코어 프로세싱

이미지 캡쳐부터 차영상, 플렌 이미지생성 하는 부분을 각 카메라 별로 스레드를 실행하여 각자 다른 코어에서 전담하여 프로세스 하도록 하였으며, 모든 스레드에서 플렌 이미지 생성이 완료되면, 각 z축마다 병합하는 부분의 스레드를 실행하여 멀티 코어에 분포되어 병합 및 외각점을 추출하도록 하였다.

이렇게 각각의 코어가 카메라 한 개씩 전담하여 플렌 영상을 제작 하며, 해당 결과물을 모든 코어가 공유하여 병합 이미지를 제작하기 때문에, 플렌 영상을 생성할 때에는 각 코어가 독립적으로 병렬로 처리 되며, 플렌 영상을 병합하기 위해 다음 프로세스를 전담하는 PC에 전달을 위해 플렌 영상을 복제 및 전송할 필요 없이 동일한 영상을 공유해서 사용하기 때문에 보다 빠른 속도로 프로세싱이 가능했다.

3.2 GPU의 병렬 처리

우리는 실시간 3D 모델링을 위해 클러스터 대신에 멀티 코어 CPU를 사용하였다. 하지만 각 프로세스에서 발생하는 많은 매트릭스 연산과 실수 연산, 병목현상 등으로 인해 CPU의 처리능력 만으로는 일부 지연되는 경우가 발생했다. 또한 플렌 영상의 병합과정에서는 실제 코어의 개수보다 스레드의 개수가 더욱 많아져 완전한 병렬처리가 불가능하였다. 그래서 우리는 GPU의 그래픽스

파이프라인을 도입해 보다 빠른 계산 처리 및 병렬 처리를 시도하였다(그림 8).

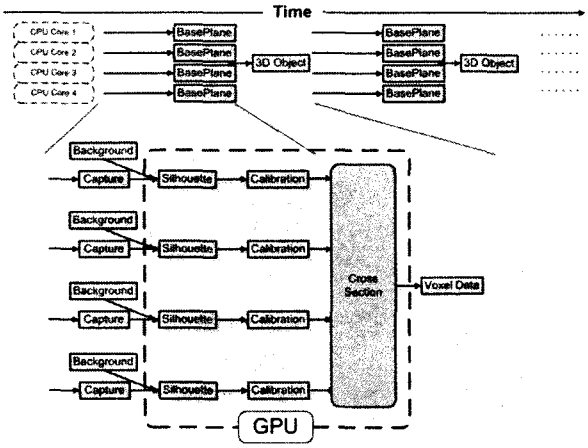

그림 8. GPU 프로세싱

CPU를 사용할 때 각 프로세스에서 가장 많은 프로세스가 필요한 차영상과 플렌 영상 제작, 병목현상이 발생하는 플렌 이미지 병합 부분을 GPU의 그래픽스 파이프라인을 사용하였다.

GPU의 그래픽스 파이프라인의 경우, 실수연산에 최적화 되어 있어 행렬연산을 많이 사용하는 3D 모델생성에서도 빠른 처리가 가능하다. 또한 현재 고급 GPU의 경우 24개의 그래픽스 파이프라인을 제공하여, 병렬처리를 사용하였을 경우에 보다 고속으로 처리가 가능하다.

4. 실험결과

다시점 카메라를 이용해 3D 모델을 실시간으로 생성하기 위해 다수의 PC가 아닌 단일 PC에 멀티 코어를 활용하여 보았다.

CPU 멀티 코어를 사용하여 프로세스를 하였을 때에는 코어의 개수를 늘려감에 따라 등 비율로 프로세스 타임이 줄어들었다(그림 9-11). GPU를 사용하였을 경우에는 그래픽카드내부에서 주요 알고리즘을 모두 처리하여, CPU 코어의 개수에 영향을 받지 않았으며, 멀티 코어보다 많은 수의 그래픽스 파이프라인을 제공하여 CPU에서만 처리하던 것 보다 더욱 좋은 결과를 보여주고 있다.

하지만 실제 결과 프레임의 경우 3프레임 미만으로 각 프로세스에서 사용되는 시간에 비해서 많이 느린 결과를 보여주고 있는데(그림 12), 이는 본 논문에서 사용한 시스템이 저가형, 소형시스템을 지향하여, 웹캠을 사용하였기 때문이다. 실제 OpenCV를 활용하여 4개의 웹캠에서 영상만을 캡쳐하였을 경우에는 4프레임 이하로 이미지를 캡쳐하기 때문에(그림 13), 이미지 캡쳐 부분에서 발생하는 병목현상으로 인해 최종 결과 프레임은 3프레임 미만의 결과를 보여준다. 하지만 각 프로세스에서 사용하는 시간을 기준으로 프레임을 계산하게 된다면, GPU를 사용하였을 때에는 약 8~9프레임정도 까지 나올 것으로 예상된다.

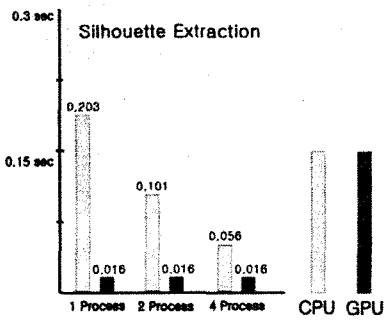

그림 9. 차영상 이미지 생성 수행 시간

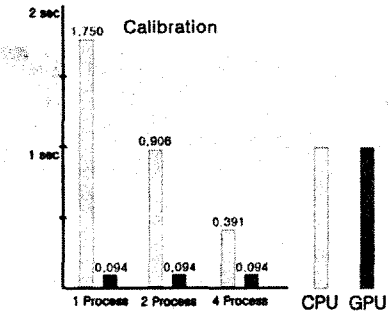

그림 10. 플렌 이미지 생성 수행 시간

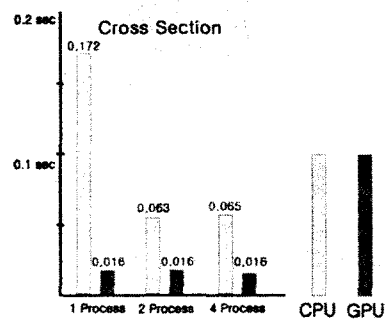

그림 11. 병합 이미지 생성 수행 시간

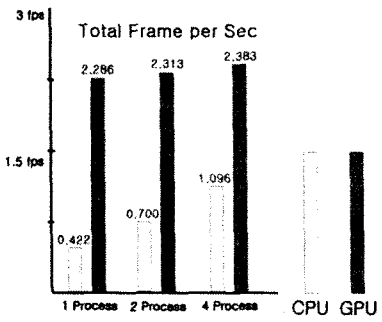

그림 12. 단위 시간동안 수행 횟수

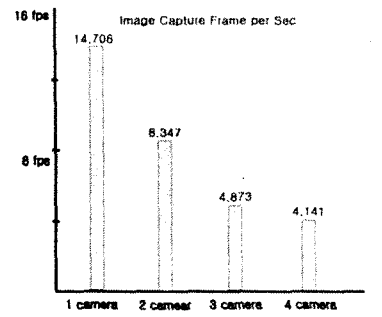

그림 13. 단위 시간동안 캡쳐한 이미지 수

5. 결론

본 논문에서는 멀티 코어를 갖는 PC한대를 사용하여 실시간으로 3D 모델을 생성하는 시스템과 GPU와 CPU를 사용하여 실시간으로 3D 모델을 생성하는 시스템을 구축하였다. 여러 실험 및 테스트 결과 멀티 코어 CPU를 사용하는 것 보다 GPU를 사용하는 것이 월등한 결과를 나타내었으며, 이렇게 GPU를 사용하여 남는 CPU자원을 제스쳐 인식과 모션캡쳐 등에 사용해볼 예정이다.

참고문헌

[1]H. Saito, S. Baba, and T. Kanade, "Appearance-based Virtual View Generation of Temporally-Varying Events from Multi-Camera Images in the 3D Room," IEEE Trans. On Multimedia, Vol.5, No3, Sept. 2003, pp.303-316.

[2]J. Schimidt, H. Niemann, and S. Vogt, "Dense Disparity Maps in Real-Time with an Application to Augmented Reality," Proceedings of IEEE Workshop on Applications of Computer Vision(WACV 2002), Orlando, FL USA, Dec., 2002, pp.225-230.

[3]Takashi Matsuyama, Xiaojun Wu, Takeshi Takai, Shohei Nobuhara,"Real-time 3D shape reconstruction, dynamic 3D mesh deformation, and high fidelity visualization for 3D video," Computer Vision and Image Understanding vol.96, no.3, pp.393-434, 2004.

[4]GKM Cheung, T. Kanade, JY. Bouguet, and M. Holler. A real time system for robust 3D voxel reconstruction of human motions. Proc. Computer Vision and Pattern Recognition Conf., vol.2, pp.714~720, 2000.

[5]S. Moezzi, L. Tai, and P. Gerard,"Virtual View Generation for 3D Voxel Reconstruction of Human Motions," proc. of CVPR, pp.714-720, 2000.

[6]H. Tezuka, A. Hori, Y. Ishikawa, and M. Sato,"PM: An Operating System Coordinated High Performance Communication Library," High-Performance Computing and Networking, Lecture Notes in Computer Science, Vol.1225, pp.708-717, 1997.

[7]Xiaojun Wu, Osamu Takizawa, Takashi Matsuyama,"Parallel Pipeline Volume Intersection for Real-Time 3D Shape Reconstruction on a PC Cluster," proc. IEEE International Conference on Computer Vision Systems, 2006

[8] "Open Source Computer Vision Library", Intel, http://www.intel.com/technology/computing/opencv/.

[9] GML C++ Camera Clibration Toolbox, http://research.graphicon.ru/calibration/gml-c++-camera calibration-toolbox-3.html.