한국컴퓨터그래픽스학회
Korea Computer Graphics Society Vol. 29, No. 3, P. 93~103

모바일 환경에서 점 구름 데이터에 대한 효과적인 광선 추적 기반 렌더링 기법

서웅[1] 김영욱[2O] 박기서[2] 김예린[2] 임인성[2*]

삼성전자[1], 서강대학교[2]

{woong.seo}@samsung.com[1], {kimyu7, noeldanny, lin17200, ihm*}@sogang.ac.kr[2]

Effective Ray-tracing based Rendering Methods for Point Cloud Data in Mobile Environments

Woong Seo[1] Youngwook Kim[2O] Kiseo Park[2] Yerin Kim[2] Insung Ihm[2*]

Samsung Electronics[1], Sogang University[2]

요 약

컴퓨터 그래픽스 분야에서 저가의 RGB-D 카메라로 촬영된 색상 및 깊이 영상을 이용한 사람 및 사물을 3차원 모델로 복원하는 문제는 오랫동안 이를 해결하기 위하여 다양한 연구들이 진행되어왔다. 저가의 RGB-D 카메라로 촬영된 색상 및 깊이 영상은 3차원 공간에서 점 구름 형태로 다루어지며, 이는 연속적인 3차원 공간상에 이산적인 값을 대응시키기 때문에 다면체 모델을 이용한 렌더링에 비해 추가적인 표면 재구성 과정이 필요하다. 본 논문에서는 다면체 모델이 아닌 점 구름을 시각화하기 위한 효과적인 광선 추적 기반 렌더링 기법을 제안한다. 특히 프로세서의 발열과 배터리 문제로 인한 모바일 환경에서의 제한적인 성능에서도 효과적인 렌더링 기법으로서의 가능성을 보인다.

Abstract

The problem of reconstructing three-dimensional models of people and objects from color and depth images captured by low-cost RGB-D cameras has long been an active research area in computer graphics. Color and depth images captured by low-cost RGB-D cameras are represented as point clouds in three-dimensional space, which correspond to discrete values in a continuous three-dimensional space and require additional surface reconstruction compared to rendering using polygonal models. In this paper, we propose an effective ray-tracing based technique for visualizing point clouds rather than polygonal models. In particular, our method shows the possibility of an effective rendering method even in mobile environment which has limited performance due to processor heat and lack of battery.

키워드: GPGPU, 광선추적법, 모바일환경, 점 구름.

Keywords: GPGPU, Ray-tracing, Mobile Environments, Point Cloud.

*corresponding author: Insung Ihm / Sogang University (ihm@sogang.ac.kr)

Received : 2023.06.16./ Review completed : 1st 2023.06.30. / Accepted : 2023.07.05.
DOI : 10.15701/kcgs.2023.29.3.93
ISSN : 1975-7883(Print)/2383-529X(Online)

1. 서론

스마트폰으로 대표되는 모바일 기기는 탑재된 프로세서 성능의 비약적인 발전과 보급으로 인하여 단순한 휴대 전화 기능을 초월하여 다양한 고급 작업을 할 수 있도록 그 활용 범위가 확대되었다. 특히, 최근 출시되는 많은 모바일 기기에는 고화질 카메라가 탑재되어 있으며, 이를 활용한 증강현실(augmented reality, AR) 기기로의 활용성이 주목받고 있을 뿐만 아니라 Google Cardboard나 삼성 기어 VR과 같은 스마트폰의 고해상도 디스플레이를 활용한 가상현실(virtual reality, VR) 기기로의 활용성도 주목받고 있다. 최근에는 Meta 사의 Oculus Quest와 같은 안드로이드 기반의 독립형 헤드 마운티드 디스플레이(head mounted display, HMD) 기기가 출시되고 있으며, 혼합현실(mixed reality, MR) 기기인 Microsoft 사의 HoloLens나 Magic Leap 사의 Magic Leap One과 같은 제품은 현실 세계와 가상 세계를 융합하여 사용자에게 새로운 경험을 제공하고 있다. 이러한 상황에서 높은 수준의 몰입감을 사용자에게 제공하기 위해서는 고성능, 고품질의 3차원 렌더링 기술이 필수적이며, 이는 고성능 PC 환경이 아닌 모바일 환경에 탑재된 저사양 프로세서의 성능을 효과적으로 활용하는 3차원 렌더링 기술의 필요성을 의미한다.

컴퓨터 그래픽스 분야에서 효과적인 3차원 렌더링을 위하여 전통적으로 래스터화 기반 기법들이 널리 사용되어왔다. 하지만, 래스터화 기반 기법의 구조적 한계를 극복하고 물리 기반 고품질 영상을 생성하기 위하여 광선 추적 기법이 계속 연구되어왔다. 다양한 광선 추적 응용 기술 개발과 GPU 하드웨어의 성능 향상 결과로 현재 PC 환경에서는 보편적인 GPU 기능을 활용하거나 광선 추적 전용 모듈을 활용하여 실시간 성능 수준으로 렌더링이 가능한 상황이다. 하지만 모바일 환경에서는 프로세서의 발열, 배터리 문제 그리고 고해상도 디스플레이로 인하여 아직 광선 추적 기술을 적용하기에는 많은 어려움이 있다.

한편, Microsoft 사의 Azure Kinect와 같은 저가의 RGB-D 카메라가 보급되면서, 다중 RGB-D 카메라의 색상 정보만이 아닌 깊이 정보를 활용한 실제 세상 인식과 사람이나 사물을 3차원 모델로 복원하는 문제는 그 중요성이 더욱 강조되고 있다. 3차원 모델 복원을 위해서는 다중 RGB-D 카메라로 촬영된 각 영상의 픽셀들을 3차원 정점으로 변환하고, 이를 하나의 좌표계로 정합하여 점 구름(point cloud) 형태로 구성해야 한다. 이때 구성된 점 구름은 저가의 RGB-D 카메라의 낮은 정밀도로 인하여 정합 알고리즘에 의한 오차 이외에도 카메라의 방향과 물체 표면이 이루는 각도에 의한 오차 등 다양한 오차가 발생한다. 이에 본 논문에서는 다중 RGB-D 카메라로 생성되는 점 구름을 활용하여 고품질 렌더링을 위한 효과적인 광선 추적 기반 렌더링 기법을 제안한다. 또한, 매우 제한적인 프로세서가 탑재된 모바일 환경에서 효과적인 렌더링 기법으로서의 가능성을 제시한다.

2. 관련연구

프로세서의 성능 향상과 GPGPU(general purpose computing on graphics processing units)의 탑재로 모바일 환경에서 광선 추적 기법을 적용하기 위한 연구가 수행되었다. Kim 등[1]은 모바일 환경을 위한 효율적인 광선 추적 기법을 개발하는 것을 목표로, 계산 비용이 높은 광선 추적 작업을 계산 비용이 낮은 선형 보간 기법으로 효과적으로 대체하여 렌더링 속도를 향상시키는 적응적 언더 샘플링(adaptive under sampling) 기법을 제안하였다. 또한 언더 샘플링으로 인해 불가피하게 발생하는 성가신 아티팩트를 최소화하는 보정 알고리즘을 제안하였다. Nah 등[2]은 모바일 환경에서 보다 반사, 굴절, 그림자 및 동적 장면을 완벽하게 지원하는 사실적인 3차원 시각화를 위해 OpenGL ES 기반 CPU-GPU 하이브리드 광선 추적 기법을 제안하였다. 앞서 기술된 모바일 환경에서 광선 추적 기반 렌더링 기법들은 삼각형 기반 다면체 모델만을 고려하고 있다.

한편, 점 구름 기반 기하 정보를 가시화하기 위하여 삼각형 기반 다면체 모델로 변환하지 않고, 광선 추적 기법을 활용한 가시화 기법[3, 4, 5, 6, 7]들이 제안되었다. 이러한 기법들은 전처리로 음함수 곡면(implicit surface)을 추정하여 점 구름을 재구성하지 않으면 RGB-D 카메라를 통해 생성되는 점 데이터에 대한 정합 오차나 깊이 값 오차로 인해 발생하는 표면 충돌 문제를 해결할 수 없다. 가시화를 위한 다른 기법으로 점 구름을 부호 거리장 형태의 볼륨 데이터로 결합하는 볼류메트릭 퓨전(volumetric fusion) 기반 기법[8, 9, 10, 11, 12]들이 제안되었다. 이러한 기법들은 RGB-D 스트림의 잡음과 정합 오류 등을 효과적으로 극복하였지만, 렌더링 과정까지 상당히 많은 연산량이 필요하여 다수의 고성능 GPU를 사용한다. 따라서 수십 대 이상의 카메라를 활용하는 경우에는 실시간 모델 생성이 어렵다. 다수의 카메라를 활용하는 것이 아닌 단일 카메라를 사용하여 사람의 얼굴이나 신체에 대한 구조(template)를 사전에 만들어두고 이를 활용하여 모델을 생성하는 기법들도 제안되었다[13, 14, 15, 16]. 이러한 기법은 사람만을 대상으로 하며 일반적인 볼륨 캡쳐 응용에는 제한적이다.

Yu 등[17]이 제안한 기법은 기계 학습을 기반으로 사람의 기하 정보를 생성하는데 RGB-D 카메라의 화각의 한계로 인하여 촬영된 영상정보가 없는 경우 기하 정보를 추정하여 생성하더라도 품질이 좋을 수 없다. 특히 색상 정보의 경우 만약 카메라 화각으로 촬영되지 못한 부분이 얼굴이라면 사람의 인지 능력은 얼굴 변화에 매우 민감하기 때문에 이러한 문제점이 부각되어 매우 낮은 품질로 느껴질 수밖에 없

다. 또한 보다 일반적인 상황에서 사람이 임의의 사물을 들고 있거나 사람과 사물을 모두 복원하여 렌더링 해야 하는 경우 사전에 기계 학습에 의한 정보가 없다면 문제가 발생한다.

3. 실시간 점 구름 획득 및 정합

3.1 다수의 RGB-D 카메라를 활용한 점 구름 획득

다수의 RGB-D 카메라를 사용하여 생성되는 점 구름을 획득하기 위하여 Figure 1과 같이 실시간으로 다중 뷰 RGB-D 데이터를 생성하기 위한 시스템을 구축하였다. 시스템에서 사용한 RGB-D 카메라는 Microsoft 사의 Azure Kinect Sensor이다. 각 카메라에서 입력으로 들어오는 매 영상 프레임 데이터에 대하여 CPU 쓰레드를 사용한 병렬 처리를 통해 30fps 성능으로 데이터 획득 가능함을 확인하였다. 또한 원본 RGB-D 데이터의 잡음(noise)을 줄이기 위하여 양방향 필터(bilateral filter) 및 실제 촬영 물체의 경계 부분에서 발생하는 깊이 값 측정 문제를 최소화하기 위하여 경계 제거 필터(outlier removal filter)를 적용하였다. 이때 각 필터는 실시간 성능을 위해서 OpenGL compute shader를 활용하여 구현하였다.

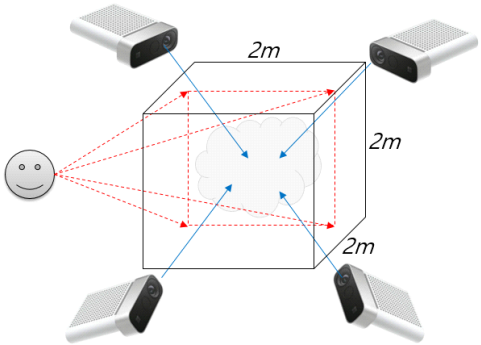


Figure 1. Multi-view real-time volume capture environment

Azure Kinect 기기는 1024 × 1024(WFOV unbinned 모드)의 깊이 영상 해상도까지 지원하지만 최대 해상도로 촬영하면 촬영 프레임 레이트(fps)를 15fps까지만 지원하고, 측정되는 깊이 값에 잡음이 많이 발생하기 때문에 상대적으로 깊이 값에 잡음이 많이 억제되며 30fps까지의 촬영 속도가 지원되는 640 × 576(NFOV unbinned 모드) 해상도로 설정하였다. 또한 기기에서 지원하는 최대 색상 해상도는 4:3 비율로 4096 × 3072 해상도까지 지원이 가능하나, 마찬가지로 촬영 프레임 레이트를 15fps까지만 지원하고 영상 품질에 문제가 있는 것을 확인하여 16:9 비율의 1280 × 720 해상도로 촬영하였다. 현재 시스템에서 해상도를 높이는 경우 단일 PC의 USB 컨트롤러의 한계로 인하여 추가적인 USB 확장 카드를 연결하더라도 이 이상의 해상도에서는 하드웨어적으로 인식이 불안정해지며 촬영 프레임 레이트가 하락하는 등의 문제가 발생하였다. 결과적으로 색상 영상은 1280 × 720 해상도, 깊이 영상은 640 × 576 해상도로 촬영하였으며, 최종 렌더링 품질을 향상하는데 있어 해상도의 증가가 아닌 연결 카메라의 수를 늘리는 것이 실험적으로 보다 효과적이었다. 또한 촬영 공간 측면에서 Azure Kinect 장치에서 깊이 영상 생성 시 카메라로부터 멀어질수록 오차가 커지는 특성이 있다[18]. 따라서 볼륨 캡쳐 공간을 2m × 2m × 2m로 구성하고 카메라는 촬영 공간 주위로 가깝게 배치하였다.

3.2 마커 기반 점 구름 정합

보편적으로 2차원 영상을 생성하는 카메라의 3차원 공간상 좌표계를 설정하기 위하여 보정 보드(calibration board)를 활용한다. 전통적으로 체스판 형태의 구조물을 주로 사용하였으나, 체스판 패턴의 경우 좌표계 보정 결과의 정밀도가 높지만 인식이 느린 단점이 있다. 반면에 ArUco 패턴[19]의 경우 인식은 빠르지만 정밀도가 떨어지는 단점이 있다. 따라서 두 기법의 장점을 활용하는 Figure 2와 같이 인식할 영역을 고속으로 탐색하면서 정밀도도 높일 수 있는 ChArUco 기반 패턴을 활용하였다.

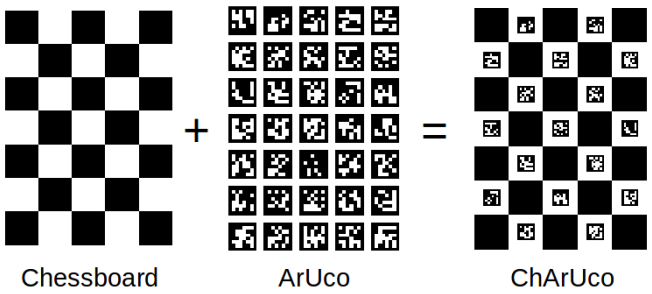


Figure 2. ChArUco pattern example

각 카메라의 좌표계를 일치시키기 위해서 ChArUco 기반 보정 보드를 사용하는데 사방에 배치한 카메라의 특성상 다수의 카메라를 하나의 좌표계로 정합하기 위해서는 좌표계 보정 과정을 여러 번 해야 한다는 문제가 있다. 이를 해결하기 위하여 Figure 3과 같은 3차원 형태의 구조물을 제작하였다.

- 정육면체 구조물 한변의 길이 : 390mm
- 내부 사각형 크기 : 50mm
- 마커의 길이 : 30mm
- 여백의 길이 : 20mm

- 4번 중심에 삼각대를 위한 지름 7mm 홀

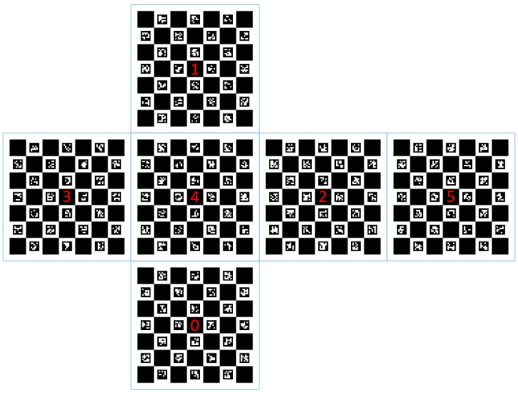


Figure 3. ChArUco pattern based Coordinate system correction structure

본 논문에서는 각 카메라들을 하나의 세상 좌표계(world coordinates, WC)로 변환하기 위해서 ChArUco 패턴 기반 구조물을 활용하여 전처리로 세상 좌표계로 변환하는 카메라 외부 행렬(extrinsic matrix)을 계산하였다. Figure 4는 이렇게 계산된 카메라 외부 행렬을 가지고 실제 사람을 촬영하여 하나의 좌표계로 정합한 결과이다. 그림을 보면 각 카메라에 의해 생성된 표면들이 서로 충돌하여 표면이 뒤섞여 있는 것을 확인할 수 있다. 또한 얼굴의 볼 부분을 보면 다른 카메라에서 정합된 표면의 오차로 인하여 표면이 맞지 않는다. 이러한 문제는 깊이 값 측정 기법에 의한 오차, 각 카메라로부터 생성된 점들을 세상 좌표계로 변환하는 변환 행렬 계산 오차 등으로 발생한다. 해당 문제를 개선하기 위한 방법을 다음 절에서 설명한다.

Figure 4. Point cloud projection results from all cameras

3.3 투영 기반 점 구름 정제

앞선 절에서 설명한 오차들 중에서도 특히 저가의 RGB-D 카메라의 경우 물체의 표면 깊이 값에 오차가 많이 발생하고, 색상 센서와 깊이 센서 간 위치 차이로 인한 보정 오차도 존재하기 때문에 이러한 문제는 완전하게 해결하기 어렵다. 하지만 문제 영역에서 각 카메라의 뷰가 다르더라도 동일한 부분을 촬영하기 때문에 Figure 5와 같이 문제 영역 부분을 연관성 있는 다른 카메라의 영역으로 보정한다면 표면 간 틀어진 정도를 완화할 수 있다.

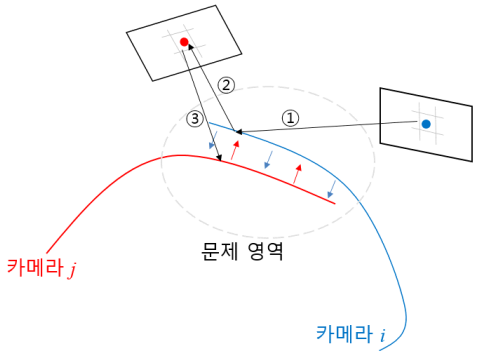


Figure 5. Camera to Camera consistency issues

보정 과정은 다음과 같다. 1)모든 점에 대하여 투영을 기반으로 3차원 공간에서 내 주변 점을 찾는다. 먼저 현재 카메라 i의 문제 영역(Figure 5의 ①)을 탐지하기 위하여, 내 점이 생성된 카메라 이외 카메라 j의 이미지 공간으로 내부행렬과 외부행렬을 사용하여 투영시킨다(Figure 5의 ②). 다른 카메라 j의 이미지 공간으로 투영된 후에 이미지 평면에서 가장 가까운 픽셀과 주변 픽셀(3 × 3)을 후보군으로 설정한다(Figure 5의 ③). 이 과정에서 주변 점들은 최대 3 × 3 × (카메라 수 - 1) 만큼 선택된다. 2)색상, 거리 임계 값을 통해 후보 점들을 간소화 한다. 색상 차이가 크거나 3차원 공간상 거리가 먼 점들은 연관성이 떨어질 확률이 크기 때문에 제외한다. 색상 차이는 각 RGB 값에 대하여 다음과 같이 색상 대비(contrast)를 계산하고, 색상 임계값(본 논문에서는 R=0.4, G=0.3, B=0.6 사용)보다 작은지 확인한다[20].

$$I_\lambda = \frac{I_\lambda^{\max} - I_\lambda^{\min}}{I_\lambda^{\max} + I_\lambda^{\min}}$$

이를 만족하는 경우, 현재 점과 3차원 공간 상 거리가 거리 임계값(본 논문에서는 2cm) 이내인지 확인한다. 3)임계값을 통과한 점들에 대하여 법선 벡터와 눈 벡터(eye vector)를 활용하여 가중치($\omega_{\theta_1}, \omega_{\theta_2}$)를 계산하고, 두 가중평균으로 생성되는 두 개 점의 평균으로 현재 점이 이동할 새로운 점을 정의한다.

$$\omega_{\theta_1} = NORM \bullet NORM_{other}, \sum \omega_{\theta_1} p_i / \sum \omega_{\theta_1}$$
$$\omega_{\theta_2} = eyevector \bullet NORM_{other}, \sum \omega_{\theta_2} p_i / \sum \omega_{\theta_2}$$

새로운 생성된 점은 현재 점과 색상이 비슷하면서 3차원 공간상 거리가 가깝고 법선 벡터가 비슷하며, 촬영 RGB-D 카메라가 표면을 수직에 가깝게 바라보고 있는 점이기 때문에 신뢰도가 높은 점에 해당한다. 4) 새롭게 생성된 점을 향해서 현재 점의 위치를 실험적으로 얻은 τ 만큼 이동시킨다(본 논문에서 τ은 30%). Figure 6에서 빨간색 원으로 표시한 영역을 보면 정제 전의 모습인 Figure 6의 (a)는 표면이 공중에 떠있는 것을 확인할 수 있는데, 정제 후의 모습인 Figure 6의 (b)는 이러한 문제가 어느 정도 완화된 것을 확인할 수 있다.

(a) Before point cloud refinement **(b)** After point cloud refinement

Figure 6. Projection based point cloud data refinement

4. 점 구름에 대한 광선 추적 렌더링

4.1 균일 격자 자료구조 구축

공간 가속 자료구조 중 하나인 균일 격자 구조의 경우 일반적으로 3차원 공간상에서 특정 격자에 기하 물체가 집중되는 경우 성능이 떨어진다. 하지만 본 논문에서 제안하는 시스템에서 RGB-D 카메라에 의해 생성되는 점 구름 데이터는 카메라 영상 평면 상에서 각 정점이 고르게 분포하기 때문에 3차원 공간 상에서도 물체 표면을 따라 각 점들이 비교적 고르게 분포한다. 따라서 구축 속도가 매우 빠르고 매 프레임 기하 정보의 수가 동적으로 변화하는 상황에서는 균일 격자 구조가 적절하다. 따라서 본 논문에서는 광선 추적 성능을 높이기 위한 공간 가속 자료구조로 균일 격자 구조를 적용하였다.

점 구름을 구성하고 있는 각 정점들은 기본적으로 3차원 공간상에서 표면적이 없기 때문에 본 논문에서는 광선 추적 기법 수행 시 광선과 교차 검사를 위해서 점 구름의 각 정점을 Figure 7과 같이 3차원 공간상 부피를 가진 구로 표현하였으며, 해당 구의 축에 정렬된 바운딩 박스를 활용하여 균일 격자 구조에 등록하였다. 이때 각 카메라에서 생성되는 점들에 대해 각각 공간 가속 구조를 생성할 수도 있지만 카메라 수만큼 자료구조를 생성해야 하는 부담이 있기 때문에 하나의 구조에 모든 점 구름을 등록하였다. 이때 각 정점에는 어떤 카메라로부터 생성된 점인지 저장하여 이후 광선과 교차 검사 시에 분류할 수 있도록 구성하였다. 이렇게 구성한 균일 격자 구조를 활용하여 광선 추적 시 고속의 광선-상자 교차 검사[21]를 활용하여 연산 비용을 줄였다.

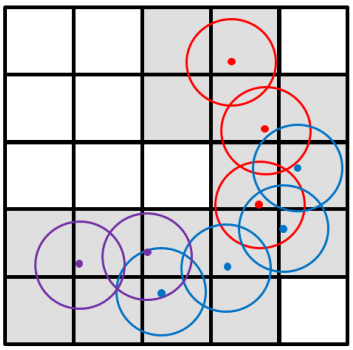


Figure 7. Uniform grid structure creation for point cloud data

4.2 뷰 우선순위 정렬을 이용한 광선 추적 기반 렌더링

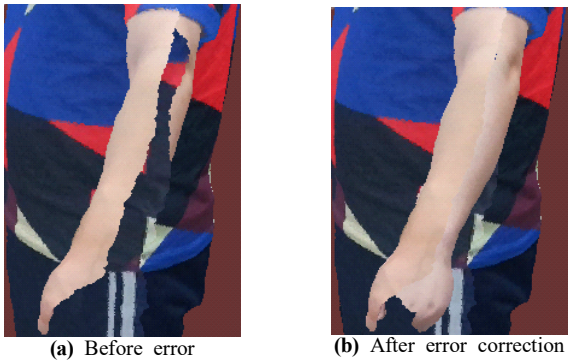
앞서 기술한 바와 같이 점 구름에 대해서 매 프레임 균일 격자 구조를 생성하고, 카메라에서 시작하는 광선은 이를 탐색하여 교차 검사를 할 점을 찾게 된다. 이때 광선이 최초로 교차하는 점 구름에 대해서 픽셀 값을 계산하는 경우 각 카메라에서 촬영된 표면 간 충돌되는 지점에서 오차에 의해 조금이라도 앞에 있는 표면에 충돌할 경우 해당 표면이 우선적으로 렌더링 되기 때문에 모든 표면이 충돌되어 뒤섞인 결과가 생성된다. 이는 점 구름을 정제하더라도 앞서 언급한 바와 같이 RGB-D 카메라의 측정 오차와 좌표계 오차 등에 의해 극복하기 어렵다.

본 논문에서는 이와 같은 문제점을 해결하기 위하여 렌더링 카메라와 촬영에 사용되는 RGB-D 카메라 사이의 상관관계를 활용한 뷰 우선순위 정렬 렌더링 기법을 제안한다. 렌더링 카메라와 사이 각도가 가장 가까운 촬영에 사용되는 RGB-D 카메라에서 생성된 점 구름을 기반으로 렌더링 하되, 촬영 카메라로 촬영되지 않은 영역들에 대해서만 그다음으로 렌더링 카메라와 사이 각도가 가까운 촬영 카메라 순서로 점 구름을 선택하여 렌더링 한다면 각 표면들이 충돌하는 문제를 피할 수 있다.

뷰 우선순위를 결정하는 방법은 먼저 렌더링 전에 렌더링 카메라의 뷰 벡터와 모든 RGB-D 카메라의 뷰 벡터 사이각(θ)을 계산하여 정렬한다. 두 벡터의 사이각이 작을수록 렌

더링 카메라와 촬영하는 RGB-D 카메라의 바라보는 방향이 비슷함으로 우선순위를 높게 주었다. 이때 만약 렌더링 카메라와 RGB-D 카메라가 사이각이 너무 크다면, 해당 RGB-D 카메라로 촬영된 부분은 렌더링 카메라에서 보이지 않으므로 광선 계산을 할 필요가 없다. 따라서 뷰 우선순위를 정렬할 때 사이각 임계 각도(T_θ)를 정해서 이를 초과하는 사이각을 가지게 되면 추가적인 광선을 발사하지 않도록 하여 렌더링 성능을 향상하였다. 본 논문에서는 실험적으로 적절한 임계 각도(T_θ)를 120도로 정하였다.

뷰 우선순위 정렬을 이용한 렌더링은 고품질의 렌더링이 가능하지만, Figure 8과 같은 문제가 생기는 경우가 있다. Figure 8의 (a)를 보면 팔 부분에서 구멍으로 인하여 렌더링이 잘못된 것을 확인할 수 있다. 이와 같은 문제가 발생하는 상황을 도시하면 Figure 9와 같다. 카메라 j가 우선순위가 높은 상태에서 최초 교차 지점을 찾아 해당 픽셀 값으로 렌더링하였는데, 실제는 우선순위가 낮은 다른 카메라 i에 의해 촬영된 새로운 표면이 존재하는 상황이다. 이를 해결하기 위하여 렌더링 시 우선순위가 높은 점들에 대해서 최초 교차 지점을 구하되, 추가적으로 우선순위가 낮은 점들에 대해서도 교차하는지 확인하여 임계 값(T_{dist}, 본 논문에서는 3cm로 설정)을 벗어날 정도로 큰 차이가 나는 표면이 존재하는 경우 우선순위가 높은 점이 존재함에도 이를 버리고 우선순위가 낮은 점을 선택하는 방법으로 오류를 개선하였다.



(a) Before error correction

(b) After error correction with additional ray

Figure 8. Error occurrence while view priority alignment

5 실험 결과

5.1 실험환경

실험 환경은 다음과 같이 PC 환경과 모바일 환경에서 수

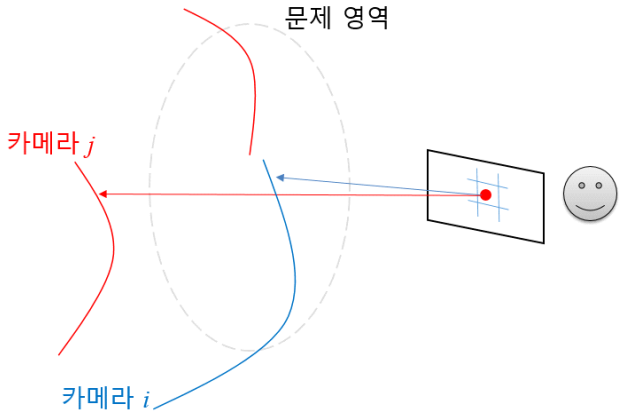


Figure 9. Problem when single ray is used for view priority alignment

행하였으며 광선 추적 계산은 PC에서는 CUDA, 모바일 기기에서는 OpenCL API를 활용하여 구현하였다. 실험 데이터는 실시간으로 다수의 RGB-D 카메라를 통해 촬영되는 점구름 데이터를 저장하여 PC에서는 별도의 렌더링 프로그램에서 데이터를 읽도록 하였고, 모바일 기기 또한 외장 메모리에 저장 후 앱에서 읽도록 하였다. 카메라의 수는 화각을 고려하여 5대로 하였으며, 이를 통하여 각 카메라로부터 생성된 충분한 점 구름 데이터를 하나의 세상 좌표계로 정합하였다.

- **PC환경**
  CPU: AMD Ryzen Threadripper 3960X(24-Core)
  GPU: Nvidia GeForce RTX 2080 Ti
- **모바일 환경**
  Samsung Galaxy Tap S6(SM-T865N)
  AP: Qualcomm Snapdragon 855
  GPU: Qualcomm Adreno 640

5.2 PC에서 실험 결과

Figure 11, 12, 13, 14는 PC 환경에서 렌더링한 결과이다. 각 Figure의 (a)는 다중 RGB-D 카메라로부터 생성된 점 구름을 정합 및 정제한 포인트 렌더링 기법, (b)는 기본적인 스플랫팅 기법, (c)는 본 논문에서 제안한 점 구름에 대한 효과적인 광선 추적 기반 렌더링 기법이다. 스플랫팅 기법의 경우 앞서 언급한 다중 뷰에 의한 표면 충돌 문제로 렌더링 품질이 좋지 않은데 입력 점 구름에 대해서 표면을 완전하게 재구성하는 기법을 적용하지 않는다면 래스터화 기반 그래픽 파이프라인의 특성상 고품질의 렌더링은 어렵다. 하지만 본 논문에서 제안한 기법은 전처리를 하지 않은 점 구름 데이터에 대해서도 뷰 우선순위 정렬 기법을 통해 고품질의 렌더링 결과를 얻을 수 있다. 이때 렌더링 시간은

30FPS 이상의 성능을 보이는 것을 확인하였다.

5.3 모바일 기기에서 실험 결과

Figure 10은 모바일 기기 환경에서 본 논문에서 제안하는 점 구름에 대한 광선 추적 응용 렌더링 기법을 적용하여 렌더링 한 결과이다. 모바일 기기 환경에서의 렌더링은 상호작용이 가능한 수준의 렌더링 성능을 보였으며, 이를 통하여 모바일 환경에서의 제한적인 성능에서도 효과적인 렌더링 기법으로서의 가능성을 확인할 수 있었다.

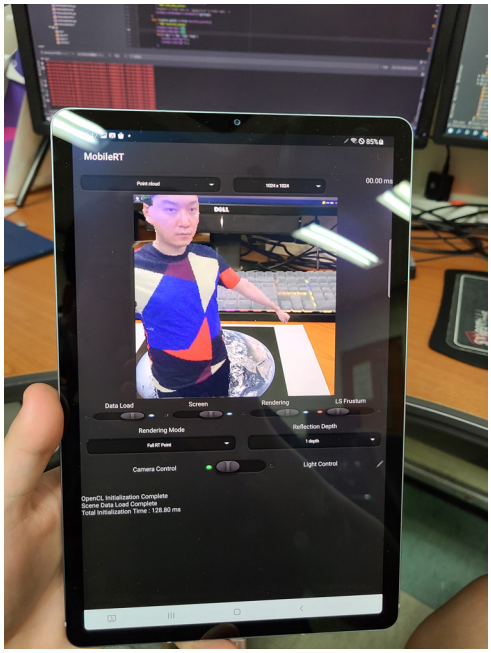


Figure 10. Rendering results on a mobile device

6. 결론

본 논문에서는 모바일 환경에 적합한 점 구름 데이터에 대한 효과적인 광선 추적 기반 렌더링 기법을 제안하였다. 먼저, 동적으로 점 구름 데이터를 생성하기 위하여 다중 RGB-D 카메라를 통해 생성되는 다중 뷰 RGB-D 영상으로 점 구름 데이터를 실시간으로 생성하고, 각 점 구름 데이터를 하나의 좌표계로 정합 및 정제하기 위한 방법을 제시하였다. 그 후, 점 구름 데이터를 전처리 형태로 삼각형 메쉬 형태의 모델을 생성하거나 부호 거리장과 같은 볼륨 데이터를 생성하지 않고, 점 구름의 보정 및 실시간으로 구축되는 균일 격자 구조를 활용하여 광선 추적 기법으로 점과 광선에 대한 교차 검사를 하였다. 이때, 뷰 우선순위 정렬 기법을 통해서 각 뷰 사이에 표면이 모호하게 충돌되는 영역들을 제거할 수 있는 매우 효율적인 고품질 영상 생성 방법을 제안하였다. 끝으로 본 논문에서 제안하는 기법이 비교적 낮은 수준의 성능을 가진 모바일 환경에서도 효과적으로 점 구름 데이터를 가시화할 수 있음을 보였다.

본 논문의 기법은 광선 추적 기법을 기반으로 RGB-D 카메라의 점 구름 데이터를 이용해 3차원 모델을 복원하는 과정에서 생기는 오류를 해결하였다. 또한, 다면체 모델을 생성하지 않고 점 구름 데이터를 고속으로 렌더링하였다. 광선 추적 기반 렌더링 기법은 래스터화 기반 렌더링 기법의 구조적 한계로 인하여 표현하기 쉽지 않은 그림자, 반사 그리고 투과와 같은 렌더링 효과를 표현 가능하다. 본 기법은 광선 추적 기반 렌더링 기법이므로 향후 이와 같은 효과를 쉽게 적용 가능할 것이라고 예상된다.

현재는 점 구름 데이터 획득 시 각 카메라에서 화이트 밸런스 및 노출 시간을 자동 모드로 촬영하였다. 스튜디오가 아닌 일반적인 실내 환경에서는 카메라 위치에 따라 조명 조건이 다름으로 각 카메라 별로 색상 밝기 차이가 발생하였고, 이로 인하여 렌더링 결과에서 경계 부분이 눈에 띈다. 향후 촬영 환경의 조명 조건을 균일하게 적용하거나 각 카메라 RGB 색상에 대해서도 보정을 통해 색상 편차가 줄어든다면 결과 영상의 품질이 향상될 것으로 예상된다.

감사의 글

이 성과는 정부 (과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2020R1A2C2011709).

References

[1] Y. Kim, W. Seo, Y. Kim, Y. Lim, J.-H. Nah, and I. Ihm, “Adaptive Undersampling for Efficient Mobile Ray Tracing,” *The Visual Computer*, vol. 32, no. 6-8, pp. 801-811, 2016.

[2] J.-H. Nah, Y.-S. Kang, K.-J. Lee, S.-J. Lee, T.-D. Han, and S.-B. Yang, “MobiRT: An implementation of OpenGL ES-based CPU-GPU hybrid ray tracer for mobile devices,” *In Proceedings of the ACM SIGGRAPH ASIA 2010 Sketches*, 2010.

[3] B. Adams, R. Keiser, M. Pauly, L. J. Guibas, M. Gross, and P. Dutré, “Efficient Raytracing of Deforming Point-Sampled Surfaces,” *Computer Graphics Forum*, vol. 24, no. 3, pp. 677-684, 2005.

[4] A. Adamson and M. Alexa, “Ray tracing point set surfaces,” *In Proceedings of the 2003 Shape Modeling International*, pp. 272-279, 2003.

[5] A. Beddiaf and M. C. Babahenini, “An improved

splat-based ray tracing for point-based objects," *In Proceedings of the 2015 12th International Symposium on Programming and Systems*, pp. 1-5, 2015.

[6] S. Kashyap, R. Goradia, P. Chaudhuri, and S. Chandran, "Real time ray tracing of point-based models," *In Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2010.

[7] I. Wald and H.-P. Seidel, "Interactive ray tracing of point-based models," *In Proceedings of the ACM SIGGRAPH 2005 Sketches*, p. 54-es, 2005.

[8] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Transactions on Graphics*, vol. 34, no. 4, 2015.

[9] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi, "Motion2fusion: Real-time volumetric performance capture," *ACM Transactions on Graphics*, vol. 36, no. 6, 2017.

[10] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi, "Fusion4D: real-time performance capture of challenging scenes," *ACM Transactions on Graphics*, vol. 35, no. 4, 2016.

[11] K. Guo, P. Lincoln, P. Davidson, J. Busch, X. Yu, M. Whalen, G. Harvey, S. Orts-Escolano, R. Pandey, J. Dourgarian, D. Tang, A. Tkach, A. Kowdle, E. Cooper, M. Dou, S. Fanello, G. Fyffe, C. Rhemann, J. Taylor, P. Debevec, and S. Izadi, "The Relightables: Volumetric Performance Capture of Humans with Realistic Relighting," *ACM Transactions on Graphics*, vol. 38, no. 6, 2019.

[12] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, V. Tankovich, C. Loop, Q. Cai, P. A. Chou, S. Mennicken, J. Valentin, V. Pradeep, S. Wang, S. B. Kang, P. Kohli, Y. Lutchyn, C. Keskin, and S. Izadi, "Holoportation: Virtual 3D Teleportation in Real-time," *In Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, p. 741-754, 2016.

[13] M. Habermann, W. Xu, M. Zollh ¨ofer, G. Pons-Moll, and C. Theobalt, "LiveCap: Real-Time Human Performance Capture From Monocular Video," *ACM Transactions on Graphics*, vol. 38, no. 2, 2019.

[14] M. Habermann, W. Xu, M. Zollhofer, G. Pons-Moll, and C. Theobalt, "DeepCap: Monocular Human Performance Capture Using Weak Supervision," *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[15] H. Li, B. Adams, L. J. Guibas, and M. Pauly, "Robust Single-View Geometry and Motion Reconstruction," *ACM Transactions on Graphics*, vol. 28, no. 5, p. 1-10, 2009.

[16] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger, "Real-Time Non-Rigid Reconstruction Using an RGB-D Camera," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.

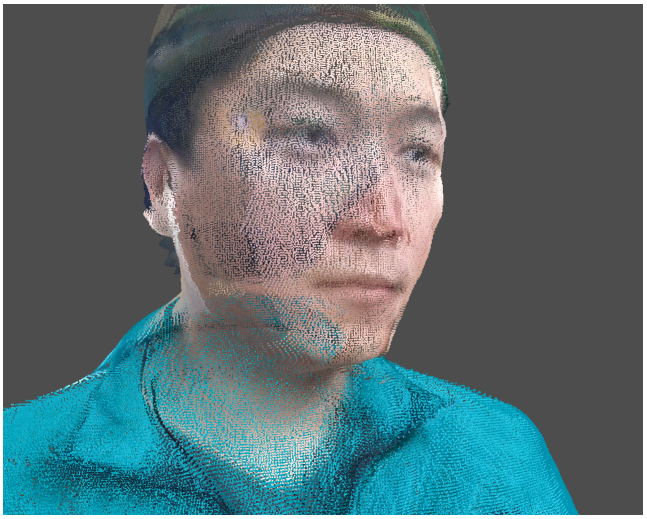
[17] T. Yu, Z. Zheng, K. Guo, P. Liu, Q. Dai, and Y. Liu, "Function4D: Real-Time Human Volumetric Capture From Very Sparse Consumer RGBD Sensors," *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5746-5756, 2021.

[18] M. Tölgyessy, M. Dekan, Chovanec, and P. Hubinský, "Evaluation of the Azure Kinect and Its Comparison to Kinect V1 and Kinect V2," *Sensors*, vol. 21, no. 2, 2021.

[19] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280-2292, 2014.

[20] D. P. Mitchell, "Generating Antialiased Images at Low Sampling Densities," *In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, p. 65-72, 1987.

[21] A. Williams, S. Barrus, R. K. Morley, and P. Shirley, "An Efficient and Robust Ray-Box Intersection Algorithm," *In Proceedings of the ACM SIGGRAPH 2005 Courses*, p. 9-es, 2005.



(a) Point Cloud

(b) Naive splat rendering method

(c) Our method

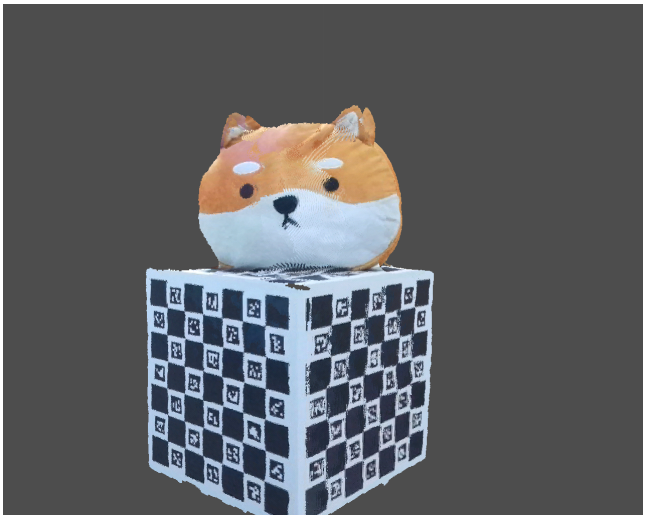
Figure 11. Experimental result (Man1)

(a) Point Cloud

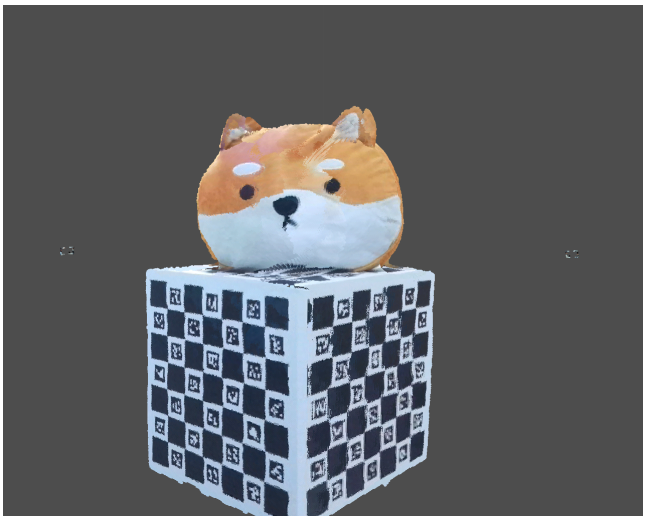
(b) Naive splat rendering method

(c) Our method

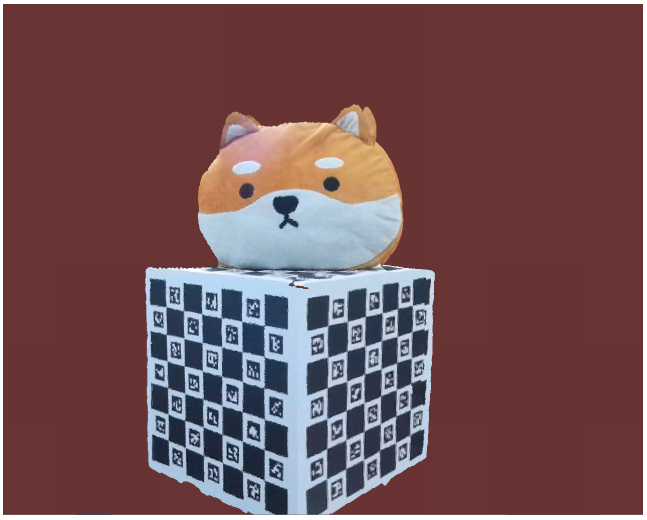
Figure 12. Experimental result (Man2)



(a) Point cloud

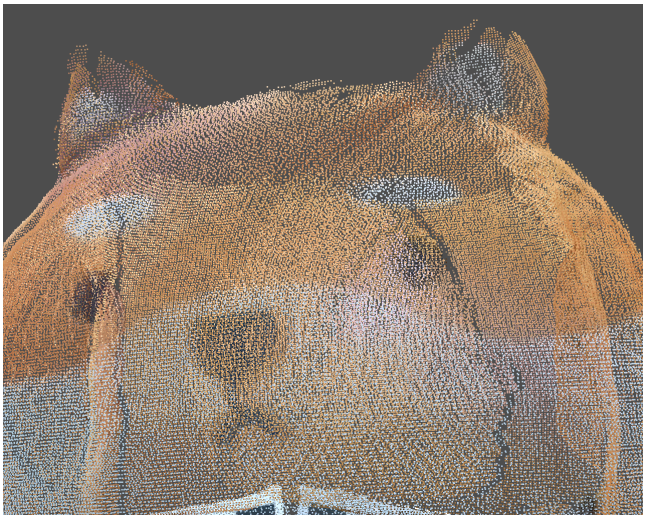


(b) Naive splat rendering method

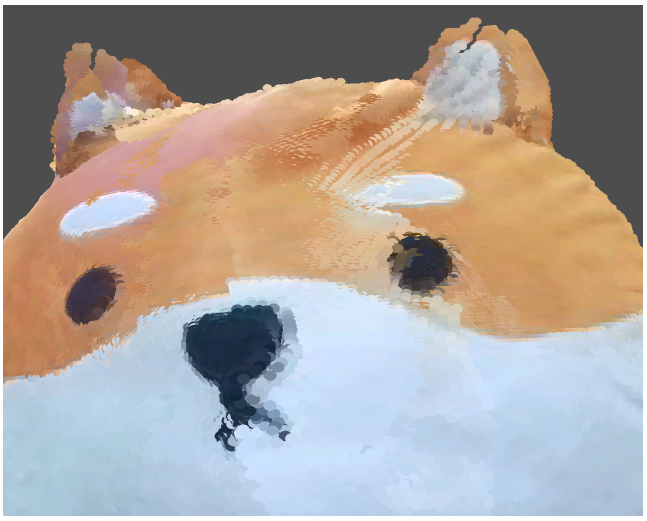


(c) Our method

Figure 13. Experimental result (Object)



(a) Point cloud



(b) Naive splat rendering method

(c) Our method

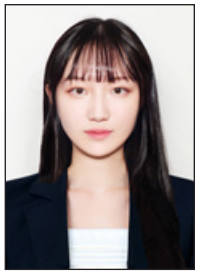
Figure 14. Experimental result (Object zoomed in)

〈저자소개〉

서 웅
- 2012년 한국외국어대학교 공과대학 전자공학과 학사
- 2014년 서강대학교 공과대학 컴퓨터공학과 석사
- 2021년 서강대학교 공과대학 컴퓨터공학과 박사
- 2021년~2022년 LG전자 연구원
- 2022년~현재 삼성전자 연구원
- 관심 분야: 컴퓨터 그래픽스, 모바일 렌더링, 광선 추적법
- https://orcid.org/0000-0001-8272-9169

김 영 욱
- 2012년 한국외국어대학교 공과대학 컴퓨터공학과 학사
- 2012년~현재 서강대학교 공과대학 컴퓨터공학과 석박통합과정
- 2016년~2019년 스타십벤딩머신(주) 연구원
- 관심 분야: 실시간 렌더링, 모바일 환경 렌더링, 가상/증강 현실
- https://orcid.org/0000-0002-4024-5841

박 기 서
- 2022년 서강대학교 공과대학 컴퓨터공학과 학사
- 2022년~현재 서강대학교 공과대학 컴퓨터공학과 석사과정
- 관심 분야: 컴퓨터 그래픽스, GPGPU
- https://orcid.org/0009-0008-5166-7733



김 예 린
- 2019년 서강대학교 자연과학대학 수학과 학사
- 2019년~2022년 SK 하이닉스 근무
- 2022년~현재 서강대학교 공과대학 컴퓨터공학과 석사과정
- 관심 분야: 컴퓨터 그래픽스, GPGPU
- https://orcid.org/0009-0002-4194-4916

임 인 성
- 1985년 서울대학교 자연과학대학 계산통계학과 학사
- 1987년 Rutgers - The State University of New Jersey 컴퓨터학과 석사
- 1991년 Purdue University 컴퓨터학과 박사
- 1999년~2000년 University of Texas at Austin 연구 교수
- 1993년~현재 서강대학교 공과대학 컴퓨터공학과 교수
- 관심 분야: 실시간 렌더링, 가상/증강 현실, GPGPU 컴퓨팅
- https://orcid.org/0000-0002-5611-925X