Azure Kinect, Open3D, Three.js 를 활용한 3D 모델링 및 WebVR 구현

윤원준, 김중헌
고려대학교

ywjoon95@korea.ac.kr, joongheon@korea.ac.kr

3D Modeling and WebVR Implementation Using Azure Kinect, Open3D, and Three.js

Won Joon Yun, Joongheon Kim
Korea Univ.

요 약

본 논문은 Depth Camera 인 Azure Kinect 를 이용하여 RGBD 이미지를 추출하고, Open3D 를 이용하여 fragment 를 만들고 이를 global registration 하여 point cloud 오브젝트로 만든 뒤, three.js 를 이용하여 WebVR 을 구현하는 방법을 제시하고 동시에 한계점과 발전 가능성을 제시한다.

Ⅰ. 서론

2020 년 세계 VR 시장이 158 억 불을 기록하였고 2027 년에는 621 억 불을 기록할 것이라는 예측이 있다.[1] 2014 년 VR 시장 크기가 6.4 억 불에 불과했던 것에 비하면 성장세가 엄청나다. 2020 년에는 IPad Pro4 가 출시하였는데 Lidar 센서를 적용하여 화두가 되었다. 이는 실제 세계의 정보를 이용하여 가상 현실을 구현하는 혼합 현실의 시대를 여는 것을 시사하는 것이기도 하다.

실제 세계의 정보를 받아 오려면 비전 센서가 필요하다. 비전 센서에는 RealSense, Azure Kinect 등이 있다. 센서로부터 받은 RGBD 데이터를 처리하는 툴은 PCL, Unity, Open3D 등이 있다.[2] 전처리를 마무리 한 데이터를 표현하는 방법으로는 VR, Hologram, Light Field 등이 있다.

본 논문은 Azure Kinect 를 비전 센서로 사용하고, Open3D 를 이용하여 데이터 전처리를 한 후, 이를 WebVR 로 렌더링을 하는 방법을 제시한다.

Ⅱ. 본론

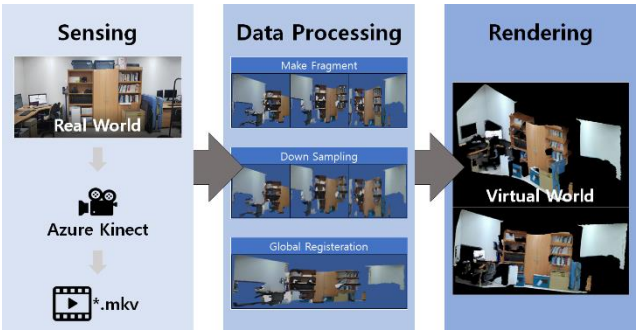


그림 1. 시스템 모델

i. 시스템 모델

실제 세계로 가상 현실을 구현하려면 크게 세 가지 단계를 거쳐야 한다. Fig. 1 은 세 가지 단계를 나타낸 그림이다. 실제 세계의 데이터를 받아 오는 Sensing, 받아 온 데이터의 전처리하는 과정, VR 로 표현하는 렌더링이 바로 그것이다.

ii. Sensing

RGBD 카메라의 문제점은 빈 공간이 많이 생긴다는 점이다. 빈 공간이 생기는 이유는 앞에 가로막는 사물이 존재할 경우에 생긴다. 이러한 문제점을 해결하기 위하여 사각지대가 없게끔 다양한 위치/방향에서 촬영을 한다.

그렇기 때문에 실제 세계(연구실)을 120° 회전하여 총 240 장 프레임의 동영상을 얻었고 이를 color 이미지와 depth 이미지 쌍으로 추출하였다.

iii. Data Processing

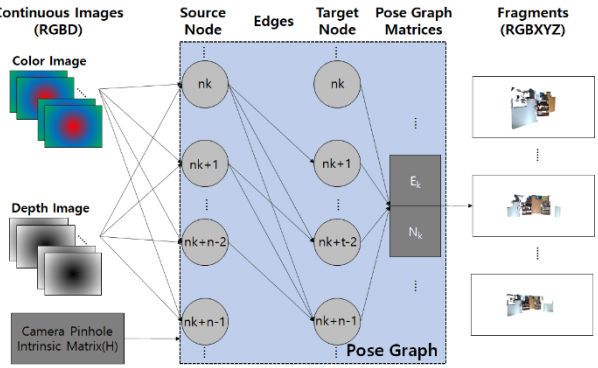


그림 2. Fragment 만드는 과정

Make Fragment. m 쌍의 Color 이미지와 Depth 이미지로 쌍을 만들고, 각각의 쌍에 대하여 RGBD 데이터를 가지

고 있는 M 개의 노드를 만든다. 총 K 개의 fragment(6D)를 만드는 것이 목적이다.
m 개의 노드를 순서대로 $\mathbf{A} = \{\mathbf{a_1}, \mathbf{a_2}, \dots, \mathbf{a_M}\}$ 라고 하자. $1 \le s < t < s + (M \bmod K) \le M$을 만족하는 자연수$s, t$에 대해 $\widehat{\mathbf{a_{s,t}}} \triangleq \mathbf{T}_{s,t} \cdot \mathbf{a_t}$를 만족하는 에지 $\mathbf{T}$가 존재한다.

$$\mathbf{T} = \{\mathbf{T}_{s,t} \mid 1 \le s < t < s + (M \bmod K) \le M,\ s, t \in N^1\} \quad -(1)$$

여기서 $\mathbf{a_s}$는 source 노드, $\mathbf{a_t}$는 target 노드, $\widehat{\mathbf{a_{s,t}}}$ 위 식을 만족하는 최적해이다. $|\mathbf{a_s} - \widehat{\mathbf{a_{s,t}}}|$가 가장 작아지는 방향으로 $\mathbf{T}$에 대하여 Jacobian RGBD odometry 방법으로 포즈 추정을 진행한다[3]. 그 결과 6×6 크기의 pose 에지 $\mathbf{E} \triangleq \cup_{k=0}^{k-1}\{\mathbf{E}_1^{k,k+1}, \mathbf{E}_2^{k,k+1}, \dots, \mathbf{E}_K^{k,k+1}\}$와 4×4 pose 노드 $\mathbf{N} \triangleq \cup_{k=0}^{k-1}\{\mathbf{N}_1^k, \mathbf{N}_2^k, \dots, \mathbf{N}_K^k\}$를 얻을 수 있다. 이를 통해 K 개의 6D(RGBXYZ) Point cloud $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_K\}$를 얻을 수 있다. $\mathbf{P}_k$는 다음과 같이 정의된다.

$$\mathbf{P}_k \triangleq \cup_{l=0}^{len(\mathbf{P}_k^F)}\{\boldsymbol{p}_k^l\} \text{ s.t. } \boldsymbol{p}_k^l \in \mathbf{P}_k \quad -(2)$$

Down Sampling. 모든 $\mathbf{P}_k$에 대해서 point cloud 를 v_{down}로 voxel 의 크기를 down sampling 하고 그리고 총 k 개의 point cloud 에 대하여 인접 점의 최대 이웃이 max_{nn}이고 최대 거리가 r인 조건으로 KD 혼합트리탐색을 진행하여 down sampling 한 모든 점에 대하여 수직 벡터를 구한다.

Global Registration. $0 < k < K$ 인 모든 자연수 k 에 대하여 $(\mathbf{P}_k\ ,\ \mathbf{P}_{k+1})$ 의 가능한 집합을 다음과 같이 정의하자.

$$\mathbf{S}_{k,k+1} \triangleq \{(\boldsymbol{p}_k^l, \boldsymbol{p}_{k+1}^m) \mid \forall \boldsymbol{p}_k^l \in \mathbf{P}_k,\ \forall \boldsymbol{p}_{k+1}^m \in \mathbf{P}_{k+1}\} \quad -(3)$$

그 때의 모든 $\mathbf{S}$의 원소 쌍 $(\boldsymbol{p}_k^l, \boldsymbol{p}_{k+1}^m)$에 대하여, 두 점 사이의 거리의 합이 최소가 되는 변환 행렬 $\mathbf{V}_{k,k+1}$를 구하는 식은 다음과 같다.

$$E(\mathbf{V}_{k,k+1}) = \sum_{(\boldsymbol{p}_k^l, \boldsymbol{p}_{k+1}^m) \in \mathbf{S}_{k,k+1}} \left\| \boldsymbol{p}_k^l - \mathbf{V}_{k,k+1} \cdot \boldsymbol{p}_{k+1}^m \right\|^2 \quad -(4)$$

이 때, $\mathbf{V}_{k,k+1}^* = \text{argmin}_{\mathbf{V}_{k,k+1}}\{E(\mathbf{V}_{k,k+1})\}$ 를 구한다. 이를 ICP registration 이라고 하며, 이를 통해 총 K-1 번 반복을 하면 온전한 1 개의 $\mathbf{P}^*$을 구할 수 있다[4]. 그 결과는 그림 3 과 같다.

iv. Rendering.

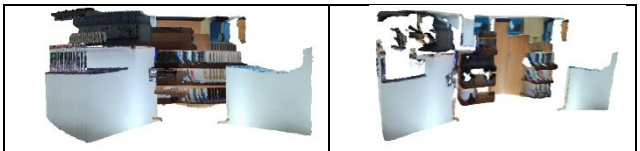
과정 II-iii.을 마친 pcd 파일 $\mathbf{P}^*$를 Three.js 의 내장 함수인 PCDLoader 을 통하여 불러온다. 또한 내장 함수인 VRButton 을 이용하면 WebVR 이 구현된다. 그 결과는 그림 4 와 같다.

v. 실험 환경

윈도우 10 home 운영 체제에서 python 3.6.5 버전 개발 환경, Azure Kinect 1.2.0 버전을 사용하였다. 그리고 python 라이브러리는 Open3D 1.0.0 버전, OpenCV 4.2.0 버전, numpy 1.19 버전을 사용하였다. Python 라이브러리는 모두 바이너리(pip)로 설치하였다. Three.js 는 r118 버전으로 사용하였다. 실험에 사용된 파라미터 값은 표 1 과 같다.

Ⅲ. 결론

3D 카메라로부터 실제 세계의 데이터를 RGBD 형태로 받아 오고, 이를 Pose Graph 방법을 이용하여 RGBXYZ 형태의 다수의 fragment 를 만들었다. 다수의 fragment 에 대하여 global registration 을 하였을 때, fragment 가 퍼즐이 맞춰 지는 것과 같은 결과를 얻을 수 있었다. 그렇게 global registraion 을 마친 pcd 파일을 three.js



(a) Global Registration 전 | **(b) Global Registration 후**

그림 3. Global Registration 전후 비교

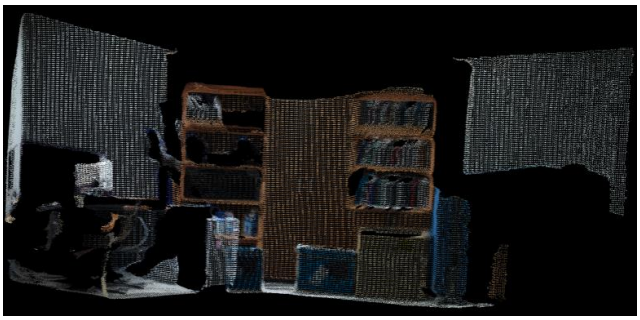


그림 4. Three.js 를 통한 WebVR 구현

Parameter	Value
M(이미지의 개수)	240
K(Fragment 개수)	6
K(Fragment 개수)	6

표 1. 실험 파라미터

자체 내장 함수를 이용하여 불러오고 WebVR 을 구현할 수 있었다. 한계점도 존재하였는데 그림 3 - (b), 그림 4 를 보면 위의 과정을 거쳤음에도 불구하고 빈 공간이 많다는 것이다.
본 논문이 제시하는 발전 방향성은 다음과 같다. 첫 째, 빈 공간을 메우는 과정을 통하여 온전한 객체를 이루는 것이다. 둘 째, pcd 파일에 빛의 간섭무늬를 집어넣는 작업을 OpenHolo 라이브러리를 통해 진행한다면 손쉽게 홀로그램 혹은 라이트 필드 기술을 구현할 수 있다.[5]

ACKNOWLEDGMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2017-0-01637) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

참 고 문 헌

[1] "Virtual Reality Market Size, Share & Trends Analysis Report By Device (HMD, GTD), By Technology (Semi & Fully Immersive, Non-immersive), By Component, By Application, By Region, And Segment Forecasts"
[2] Q. Zhou, "Open3D: A Modern Library for 3D Data Processing", arXiv, 1801-09847
[3] C. Kerl, "Colored Point Cloud Registration Revisited", *IEEE International Conference on Computer Vision(ICCV)*, 2017, DOI: 10.1109/ICCV.2017.25
[4] P. J. Besl, "A method for registration of 3-D shapes", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, NO.2, 1992
[5] OpenHolo, url : http://openholo.org/