Depth 정보를 이용한 Texturing 의 View Selection 알고리즘

한현덕, 한종기
세종대학교
gusejr0425@sju.ac.kr, hjk@sejong.edu

View Selection Algorithm for Texturing Using Depth Maps

Hyeon-Deok Han, Jong-Ki Han
Sejong University

요 약

2D 이미지로부터 카메라의 위치 정보를 추정할 수 있는 Structure-from-Motion (SfM) 기술과 dense depth map 을 추정하는 Multi-view Stereo (MVS) 기술을 이용하여 2D 이미지에서 point cloud 와 같은 3D data 를 얻을 수 있다. 3D data 는 VR, AR, 메타버스와 같은 컨텐츠에 사용되기 위한 핵심 요소이다. Point cloud 는 보통 VR, AR, 메타버스와 같은 많은 분야에 이용되기 위해 mesh 형태로 변환된 후 texture 를 입히는 Texturing 과정이 필요하다. 기존의 Texturing 방법에서는 mesh 의 face 에 사용될 image 의 outlier 를 제거하기 위해 color 정보만을 이용했다. Color 정보를 이용하는 방법은 mesh 의 face 에 대응되는 image 의 수가 충분히 많고 움직이는 물체에 대한 outlier 에는 효과적이지만 image 의 수가 부족한 경우와 부정확한 카메라 파라미터에 대한 outlier 에는 부족한 성능을 보인다. 본 논문에서는 Texturing 과정의 view selection 에서 depth 정보를 추가로 이용하여 기존 방법의 단점을 보완할 수 있는 방법을 제안한다.

1. 서론[1]

2D 이미지로부터 카메라의 위치 정보를 추정할 수 있는 Structure-from-Motion (SfM) 기술과 dense depth map 을 추정하는 Multi-view Stereo (MVS) 기술을 이용하여 2D 이미지에서 point cloud 와 같은 3D data 를 얻을 수 있다 [1, 2, 3, 4]. 최근 VR, AR, 메타버스와 같은 컨텐츠의 수요가 증가하고 있는데 이러한 컨텐츠의 핵심 요소가 바로 3D data 다. Point cloud 는 point 사이에 빈 공간이 존재하여 완전한 물체로 느껴지지 않는다는 문제점이 있다. VR, AR, 메타버스와 같은 컨텐츠에서 사용되기 위해서는 보다 실제의 물체처럼 표현되어야 한다. 일반적으로 위와 같은 분야에 이용하기 위해 point cloud 로부터 삼각형 면들로 이루어진 mesh 형태를 만들어낸다 [5].

Mesh data 는 포인트들을 이어 삼각형의 face 들로 물체의 표면을 구성한다. Point cloud 와는 다르게 물체 표면을 빈 공간 없이 잘 나타낸 data 지만 texture 정보는 아직 포함되지 않은 상태다. 실제 물체처럼 보이기 위해선 mesh data 에 texture 를 입히는 Texturing 과정이 꼭 필요하다 [6, 7]. Texturing 과정은 mesh 를 구성하는 각각의 face 마다 어떤 image 에서 texture 정보를 가져올지를 결정하는 과정이다. 기존 방법은 흐릿한 영상을 피하기 위하여 영상의 gradient 를 계산하여 이용하고, 움직이는 물체와 같은 outlier 를 판단하기 위해 color 정보를 이용했다 [7]. 하지만 단순히 color 정보만을 이용할 경우

[1] 연락저자: 한종기

outlier 를 판단하지 못하는 경우가 발생하고 이는 성능을 저하시키는 주 원인이 된다. 본 논문에서는 Texturing 과정의 view selection 에서 depth 정보를 추가로 이용하여 기존 방법의 단점을 보완할 수 있는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2 장에서 기존 Texturing 방법에 대한 간단한 설명과 문제점을 보인다. 3 장에서 기존 방법의 문제점을 해결하기 위한 제안하는 방법을 설명하고 4 장에서 실험결과를 보인 후 5 장에서 결론짓는다.

2. 기존 Texturing 방법 및 문제점

Texturing 방법은 mesh 의 각 face F_i 마다 어떤 image l_i 를 이용할 것인지 결정하는 방법이다. [6, 7] 방법들은 이를 위해 식 (1)과 같은 에너지 함수를 사용했다.

$$E(l) = \sum_{F_i \in \text{Faces}} E_{data}(F_i, l_i) + \sum_{(F_i, F_j) \in \text{Edges}} E_{smooth}(F_i, F_j, l_i, l_j) \quad (1)$$

여기서 data term E_{data} 는 mesh 의 face 마다 가장 적합한 image 가 선택되도록 한다. Smoothness term E_{smooth} 는 인접한 face 사이의 seam 이 잘 어우러지도록 하는 term 이다. 식 (1)의 에너지함수는 [8]과 같은 graph-cut 방법을 이용하여 해결할 수 있다.

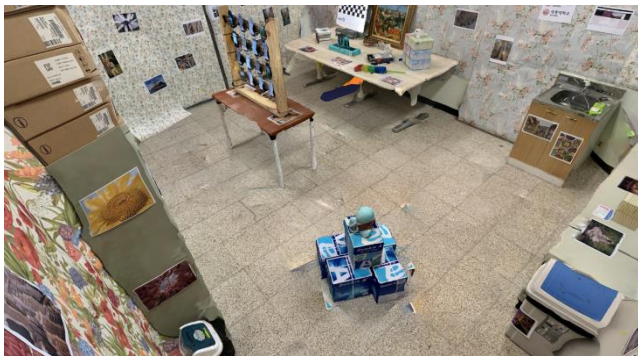
Data term 과 smoothness term 의 수식은 위의 목적에 맞게 설정된다. [6, 7] 방법들은 흐릿한 영상보다 뚜렷한 영상이 각 mesh 의 face 에 적합한 image 라고 판단하기 위해 data term 으로 image 의 gradient 를 계산하여 사용했다. Smoothness term 에서는 이웃한 face 에 같은 image 를 사용하면 seam 이 없다는 점을 이용했다. 즉, 간단히 생각하면 각 face 마다 gradient 가 높은 image 를 선택하되 최대한 이웃한 face 사이에서는 같은 image 를 선택하려고 노력하는 것이다. [7] 방법에서 사용한 Data term 과 smoothness term 은 각각 식 (2), (3) 과 같다.

$$E_{data}(F_i, l_i) = -\int_{P(F_i, l_i)} \left\| \nabla \left(I_{l_i}(p) \right) \right\|_2 dp \quad (2)$$

$$E_{smooth}(F_i, F_j, l_i, l_j) = \begin{cases} 0 & l_i = l_j \\ 1 & l_i \neq l_j \end{cases} \quad (3)$$

식 (2)에서 $P(F_i, l_i)$ 는 F_i 를 image l_i 로 projection 시켰을 때 해당 영역의 픽셀들의 집합이고 p 는 픽셀 위치를 나타낸다. $I_{l_i}(p)$ 는 image l_i 에서 p 위치의 밝기 값을 나타낸다.

단순히 data term 과 smoothness term 을 이용하여 Texturing 을 수행할 경우 outlier 의 영향을 많이 받을 수 있다. Outlier 는 보통 움직이는 물체가 포함되어 있거나 SfM 과정에서 카메라 파라미터가 완벽하게 추정되지 못하기 때문에 발생한다. [7]은 움직이는 물체가 포함된 outlier 를 제거하기 위해 color 정보를 이용한다. [7]은 현재 face F_i 가 선택할 수 있는 image 가 4 장 이상일 때, 각 image 에서 F_i 에 해당하는 patch 의 평균 color 값을 이용하여 이 평균에서 매우 벗어난 image 를 제거하는 방법을 사용했다. 이는 현재 face F_i 에 대한 image 의 수가 충분히 많고 outlier 가 움직이는 물체인 경우에는 효과적이지만 그림 1 과 같이 카메라 파라미터가 완벽하지 못해서 발생하는 outlier 에서는 좋은 성능을 보이지 못한다.



(a)

(b)

그림 1. [7] 방법을 이용한 Texturing 결과

3. 제안하는 알고리즘

제안하는 알고리즘의 목적은 크게 두 가지다. 첫 번째는 현재 face F_i 에 대한 image 의 수가 적은 경우에도 outlier 를 제거하는 것이고 두 번째는 움직이는 물체에 대한 outlier 뿐 아니라 부정확한 카메라 파라미터로 인해 생기는 outlier 도 효과적으로 제거하는 것이다. Texturing 단계에서는 각각의 image 에 대한 depth map 이 주어진 상태이므로 제안하는 방법은 이를 이용해 효과적으로 outlier 를 제거한다.

Mesh 에서 각각의 face 는 보통 물체의 edge 를 넘어가지 않는다. 따라서 각각의 face 를 image 로 projection 시켰을 때 해당 영역에서 depth 값은 변화량이 일정해야 한다. 따라서 제안하는 방법은 depth map 에서 Laplacian 계산을 수행하여 2 차 미분 값을 구한다. 이후 face 를 depth map 으로 projection 시킨 patch 내에 2 차 미분 값이 큰 픽셀이 존재하면 outlier 로 간주한다. Outlier 로 간주되는 경우에 대한 식은 (4)와 같다.

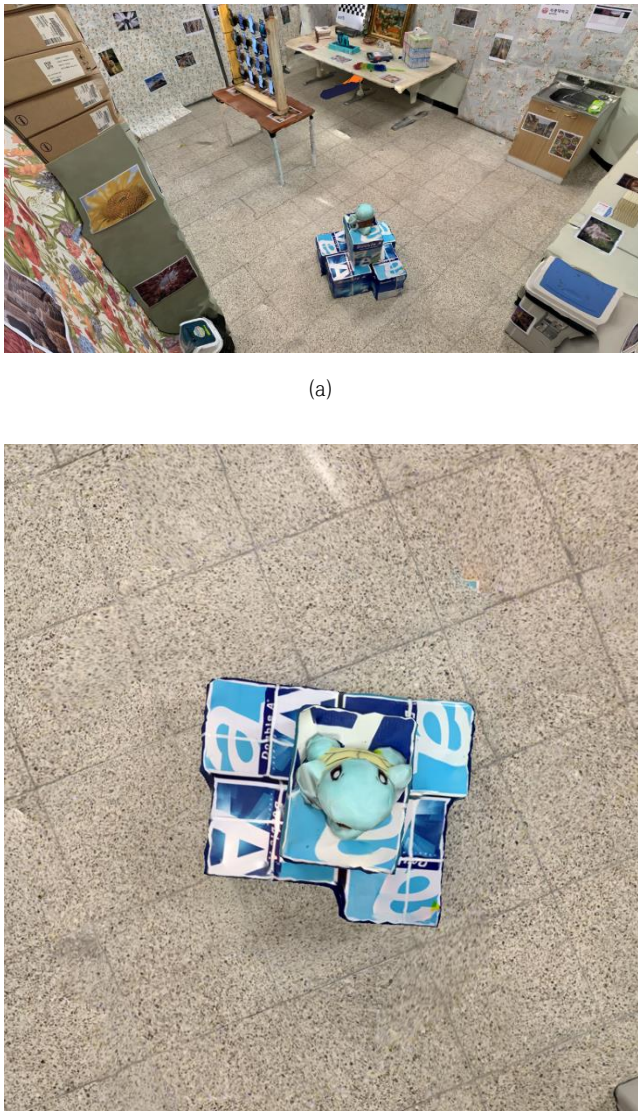
$$\max_{p \in P(F_i, I_i)} \left| \nabla^2 \left(D_{I_i}(p) \right) \right| > \tau \tag{4}$$

식 (4)에서 $D_{I_i}(p)$ 는 p 위치의 depth 값을 나타낸다.

Depth map 에는 hole 이 존재할 수 있다. 특히 hole 은 특징이 없는 평평한 영역에 존재한다. Hole 이 존재하는 영역에 대해선 (4)의 값을 0 으로 설정한다. Outlier 가 존재하는 부분은 보통 특징이 있는 영역이기 때문에 hole 이 존재하는 영역은 outlier 가 아니라고 볼 수 있다. 만약 face F_i 에 대해 모든 image 가 outlier 로 판별된다면 그 중 가장 작은 식 (4)의 max 값을 갖는 image 만을 inlier 로 남긴다.

4. 실험결과

제안하는 방법은 수식 (4)에서 1 개의 파라미터가 존재한다. 실험 결과에서는 τ 값으로 0.0005 를 사용했고 이는 실험적으로 정해졌다. 그림 2 는 제안하는 방법을 이용한 Texturing 결과를 나타낸다. 그림 2 의 (a)와 (b)를 보면 [7] 방법에서는 제거하지 못했던 부정확한 카메라 파라미터로 인한 outlier 들을 제안하는 방법이 잘 제거하는 것을 보인다.



(a)

(b)

그림 2. 제안하는 방법을 이용한 결과.

5. 결론

본 논문은 Texturing 과정에서 depth map 을 이용하여 효과적으로 outlier 를 제거하는 알고리즘을 제안한다. 제안하는 방법은 단순히 움직이는 물체가 존재하는 경우의 outlier 만 효과적으로 제거하는 것이 아니라 카메라 파라미터가 부정확할 때 발생하는 outlier 도 효과적으로 제거한다. 또한 기존의 방법과 다르게 각각의 face 에 대응하는 image 의 수가 매우 적어도 제안하는 알고리즘을 적용할 수 있다.

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) under Grant (2017-0-00486) funded by the Korea government (MSIT).

참 고 문 헌 (References)

[1] P. Moulon, P. Monasse, and R. Marlet, "Adaptive structure from motion with a contrario model estimation," *In Asian Conf. Comput.Vision*, pp.257-270, Springer, 2013. Available: github.com/openMVG/openMVG/.

[2] P. Moulon, P. Monasse, and R. Marlet, "Global Fusion of Relative Motions for Robust, Accurate and Scalable Structure from Motion," *In International Conference on Computer Vision*, December 2013.

[3] Shuhan Shen, "Accurate Multiple View 3D Reconstruction Using Patch-Based Stereo for Large-Scale Scenes," *IEEE transactions on image processing*, 22(5):1901-1914, 2013.

[4] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler,
M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," *in Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Y. Zhou, S.H. Shen, Z.Y. Hu, "Detail preserved surface reconstruction from point cloud," *Sensors,* 2019, 19(6): 1278, doi:10.3390/s19061278

[6] V. Lempitsky, D. Ivanov, "Seamless mosaicing of image-based texture maps," *In CVPR,* 2007.

[7] M. Waechter, N. Moehrle, and M. Goesele, "Let there be color! Large-scale texturing of 3D reconstructions," *in Proc. Eur. Conf. Comput. Vis*, pp. 836-850, 2014.

[8] Y. Boykov and O. Veksler, "Graph Cuts in Vision and Graphics: Theories and Applications," *The Handbook of Math. Models in Computer Vision*, Springer, 2006.