다시점 영상과 움직임 기반 구조를 이용한 3차원 점군 생성 방법

고재련, 호요성
광주과학기술원 전자전기컴퓨터공학부
jrko@gist.ac.kr, hoyo@gist.ac.kr

3D Point Cloud Generation Using Structure from Motion with Multiple View Images

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

요 약

움직임 기반 구조는 다양한 시점에서 촬영된 영상 또는 순차적인 비디오 영상으로부터 카메라의 움직임과 3차원 구조의 점군을 생성하는 컴퓨터 비전 기술이다. 이 논문에서는 일반적인 움직임 기반 구조 알고리즘인 점증적 움직임 기반 구조 방법을 설명하고 관련된 공개 소스 소프트웨어를 간략히 소개한다. 실험 결과는 점증적 움직임 기반 구조에서 이상치 제거 과정을 제외한 알고리즘을 실행하여 다시점 영상 수에 따라 다른 결과를 보여준다.

1. 서 론

움직임 기반 구조는 두 장 이상의 다양한 시점에서 촬영한 영상으로 카메라의 움직임과 촬영된 장면이 3차원으로 복원된 구조를 추정하는 방법이다. 입력 영상의 종류에 따라 크게 두 부류로 나눌 수 있는데, 비디오와 같이 단일 카메라를 통해 순차적으로 획득한 영상들로 추정하는 방법과 다른 하나는 순서가 없이 다양한 시점에서 동일한 장면을 촬영한 임의의 영상들로부터 추정하는 방법이다.

이 논문은 다양한 시점에서 촬영한 영상들로 기존의 움직임 기반 구조를 통해 3차원 점군을 생성하는 방법을 다루었다. 다양한 버전의 움직임 기반 구조 방법 중 일반적으로 알려진 알고리즘인 점증적 움직임 기반 구조(Incremental Structure from Motion)를 설명하고 소스 코드가 자유롭게 공개된 관련 소프트웨어들을 소개한다. 또한 공개된 소프트웨어 중 하나인 SfM-Toy-Library를 이용하여 실제 움직임 기반 구조 알고리즘을 실행하여 3차원 점군 생성 결과를 실험하였다.

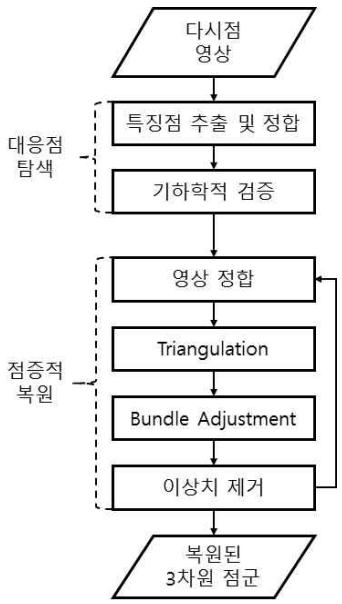
2. 움직임 기반 구조를 이용한 3차원 점군 생성

2.1 움직임 기반 구조 파이프라인

움직임 기반 구조는 그림 1과 같이 각각의 하위 작업이 순차적으로 처리되는 파이프라인 형태의 알고리즘이며, 반복적인 복원 과정이 포함되어 있는 점증적 움직임 기반구조가 가장 많이 사용된다 [1].

대응점 탐색 단계에서는 각각의 영상에 대하여 특징점을 추출한다. 움직임 기반 구조가 다시점 영상들 사이에서 추출된 특징점들을 쉽게 인식할 수 있도록, 크기와 회전에 대하여 불변성을 가진 특징 기술자를 사용한다. 기본적으로 강건한 특징점 정합이 가능한 SIFT 및 SURF가 많이 쓰인다.

특징점 추출 및 정합은 오로지 외형상의 대응점일 뿐 실제 3차원 장면에서의 동일한 대응점을 보장하지 않는다. 따라서 영상 간 평면 사영변환(Homography)을 추정하는 방법으로 대응점 탐색의 정확성을 확인하는 기하학적 검증 과정을 거친다. 충분한 양의 특징점 켤레가 제대로 사영변환 될 경우에만 검증되었다고 할 수 있다. 출력 데이터로 각 영상을 꼭지점으로 대응, 검증된 영상 켤레를 선으로 연결한 장면 그래프(Scene Graph)를 생성한다.



(그림 1) 일반적인 점증적 움직임 기반 구조 알고리즘의 순서도

점증적 복원 단계는 생성된 장면 그래프를 기반으로 카메라의 움직임과 복원된 3차원 점군을 얻어낸다. 먼저 2개 시점 영상을 기반으로 최초의 3차원 복원 모델을 초기화하며, 시점 상의 2차원 특징점과 실제 관측된 3차원 장면에 해당하는 점과의 대응 관계를 포함하고 있어야 한다.

이후 나머지 다시점 영상들을 하나씩 추가하는 영상 정합 과정을 거친다. 이 과정은 Perspective-n-Points (PnP) 알고리즘을 통해 카메라 움직임 정보를 추정하여 카메라 내부 매개변수를 획득할 수 있으므로 [2], Triangulation 과정에서는 추가된 시점 영상에 대한 새로운 3차원 장면 점을 획득한다 [3].

하지만 앞의 두 과정에서 얻어진 카메라 움직임 정보와 3차원 장면 점들은 촬영 환경 등의 외부 요인으로 이상치(Outlier)에 오염되어 있는 경우가 많다. 이러한 이상치 오류 누적을 완화하기 위하여 Bundle Adjustment 과정을 거치게 된다. 이는 Levenberg-Marquardt 방법과 같은 최적화 알고리즘을 사용, 획득한 카메라 매개변수와 3차원 장면 점의 좌표 값을 개선한다. 마지막으로 이상치를 제거하는 알고리즘이 추가될 수 있으며, 이러한 과정이 모든 다시점 영상들에 적용된 후 움직임 기반 구조 알고리즘은 복원된 3차원 점군을 출력하고 종료된다.

2.2 움직임 기반 구조 관련 공개 소스 소프트웨어

움직임 기반 구조를 구현한 공개 소스 소프트웨어는 종류가 많고 온라인에서 쉽게 구할 수 있다. 하지만 복잡한 연산과 3차원 점군 복원 출력 등의 다양한 기능을 요구하므로, 다수의 하위 라이브러리 종속성을 요구하는 경우가 많다. 따라서 C++로 구현된 소프트웨어가 많으며 잘 알려진 관련 공개 소스 소프트웨어 몇 가지를 소개한다.

실행 버전 및 소스 코드 전부를 제공하는 대표적인 소프트웨어로는 Bundler가 있으며 GUI로 구현되어 다양한 기능을 제공한다. COLMAP은 범용성을 추구한 움직임 기반 구조 알고리즘을 구현하였다. 이 외에도 움직임 기반 구조 알고리즘 구현을 위한 라이브러리인 openMVG는 문서화 작업이 잘 되어 접근하기 편리하다. 가장 잘 알려진 컴퓨터비전 라이브러리인 OpenCV는 3.0 버전부터 외부 기여 모듈로써 SfM 모듈을 제공하며, 설치 방법이 간단하지만 리눅스 환경에서만 사용 가능하다.

3. 실험결과

실험에 사용된 알고리즘은 공개 소스 소프트웨어 중 하나인 SfM - Toy -Library를 기반으로 구현되었다[4].

그림 2는 Middlebury Multi-View Stereo Dataset에서 제공하는 Temple과 TempleRing 세트에 대하여 실험한 결과이다 [5]. 동일한 장면에 대하여 구성되었으나, Temple의 경우 312개 시점, TempleRing은 47개 시점으로 영상 개수에 차이가 있다. 알고리즘 구현에 있어서 이상치 제거 과정을 실험에 포함하지 않았고, 복잡도를 낮춘 버전의 Bundle Adjustment을 적용한 결과, 이상치가 누적되어 312개 시점에서는 카메라 움직임과 3차원 점군 모두 정상을 벗어난 결과가 출력되었다. 오히려 훨씬 적은 수인 47개 시점 사용 시 정상적인 3차원 점군이 생성되었다.

(그림 2) Temple 및 TempleRing 세트에 대한 3차원 점군 생성 결과

4. 결 론

이 논문에서는 다시점 영상을 입력 데이터로 사용하여 움직임 기반 구조를 통해 해당 다시점 영상 장면의 3차원 점군을 생성하는 방법을 설명하고 관련된 공개 소스 소프트웨어를 소개하였다. 기존의 움직임 기반 구조 방법에서 이상치 제거 과정을 생략하고, 다소 간략화 된 버전의 Bundle Adjustment 과정이 포함된 움직임 기반 알고리즘을 통해서 동일한 장면을 대상으로 시점 수가 다른 다시점 영상 세트를 입력으로 실험 한 결과 오류가 누적되는 상황에 따른 결과를 확인할 수 있었다.

감 사 의 글

본 논문은 민·군기술협력사업(Civil-Military Technology Cooperation Program)으로부터 지원을 받아 수행된 연구임.

참 고 문 헌

[1] J. Schonberger, J. Frahm, “Structure-from-Motion Revisited,” Conference on Computer Vision and Pattern Recognition, 2016

[2] V. Lepetit, F. Moreno-Noguer, P. Fua, “EPnP: An Accurate O(n) Solution to the PnP Problem,” International Journal of Computer Vision, vol. 81, pp. 155-166, 2009

[3] R. Hartly, P. Sturm, “Triangulation,” Computer Vision and Image Understanding, vol. 68, pp. 146-157, 1997

[4] http://github.com/royshil/SfM-Toy-Library

[5] http://vision.middlebury.edu/mview/data/