모바일 환경에서 단안시 기반의 Visual Odometry 추정

성홍념*, 박성현**, 한승주***, 이칠우****
*,***전남대학교 소프트웨어공학과, **수학교육과, ****컴퓨터정보통신공학과
e-mail : *6590f1@naver.com, ****leecw@jnu.ac.kr

Visual Odometry Estimation Based on Monocular Vision in a Mobile Environment

Hongnyeom Sung*, Seonghyeon Park**, Seungju Han***, Chilwoo Lee****

요 약

실제 야외 환경에서 플레이가 가능한 AR 슈팅 게임(FPS:First Perspective Shooting) 구현을 위해서는 플레이어의 이동 경로를 추적하는 것이 필요하다. 본 논문에서는 Epipolar 기하 제약을 기반 Monocular Visual Odometry 추정 알고리즘과 이를 모바일 단말기에서 구현한 결과에 대해 기술하고 Odometry 추정치의 정확도를 향상 시킬 수 있는 방법에 대해 소개한다.

1. 서 론

모바일 환경에서 경로, 즉 Odometry를 계산하는 데는 GPS, 관성항법, 컴퓨터 비전을 활용한 방법이 주로 사용되고 있다. GPS는 4미터 정도의 오차가 존재하여 정밀한 움직임을 구현하는 데 한계가 있고, 관성항법은 이동 거리가 커짐에 따라 한쪽 방향으로 에러가 누적된다. 그림1은 Madgwick의 알고리즘[1]을 바탕으로 관성센서를 사용하여 모바일 기기에서 실험했을 때 발생하는 오차를 보여준다. 이 결과는 고도의 정밀도를 요구하는 FPS 게임에서 관성센서 정보를 그대로 사용할 수 없음을 의미한다.

본 논문에서는 모바일 환경에서 Odometry를 계산하기 위해 컴퓨터비전 기술을 사용하여 3차원 경로를 추정하는 알고리즘을 사용한다. 이 방법은 정밀도가 높아 주로 드론[2], 자동차[3]의 경로를 추정하는 방법에 주로 적용되고 있으나 야외, 모바일 환경에서 사용된 예는 드물다. 왜냐하면 Visual Odometry 알고리즘은 계산량이 많고 모바일에서 작동 시 정확도가 낮아서 적용하기에는 무리가 있기 때문이다. 본 논문에서는 Visual Odometry 알고리즘에 대하여 간략히 설명하고, 모바일 기기에서 컴퓨터 비전을 통한 Odometry를 실제로 구현하고 Odometry의 정확도를 향상할 수 있는 실험 결과에 관해서도 설명한다.

2. 모바일 기기를 이용한 Visual Odometry 추정

본 논문에서는 Avi Singh의 레포트[3][4]를 참고하여 모바일 기기에서 Visual Odometry를 구현하였다.

1. 남아있는 특징점 개수가 2000개 이하일 때 FAST[5] 알고리즘으로 특징점 탐색
2. KLT Tracker[6]로 이전 프레임의 특징점을 현재 프레임에 대응
3. 대응된 특징점 쌍들을 RANSAC을 통해 가장 올바르게 대응된 특징점 5개를 계산 후 5-Point 알고리즘으로 Essential Matrix 추정
4. Essential Matrix에서 회전 행렬과 병진벡터를 구해 프레임 사이의 궤적을 계산

위 표는 Odometry 추정을 위한 알고리즘의 개요이다. 본 논문에서는 Visual Odometry 계산 정확도를 높이기 위해 두 가지 방법을 고려하였다. 첫 번째는 크기를 미리 알고 있는 물체를 준비하고 이를 촬영한 영상으로부터 구한 카메라 파라미터의 사용하여 camera calibration의 정확도를 높인 것이다. 두 번째는 알고리즘에서 알고리즘 개요에서 1, 2번 단계를 수정하여 프레임 사이의

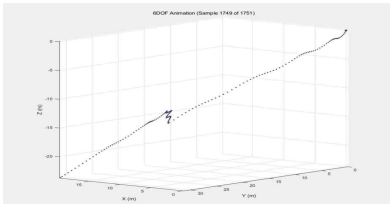

그림 1. 스마트폰의 가속도센서를 이용한 경로추정의 결과

특징점 대응 관계를 더욱 정밀하게 구하는 방법이다.

2.1 Camera Calibration

카메라로 촬영한 이미지는 실세계의 3차원 공간상의 점들을 2차원 평면에 투영하여 얻어진다. 이때, 3차원 공간상의 점들이 이미지의 어느 부분에 맺히는지는 이론적으로는 이미지 센서의 일직선상에 위치해야 하지만 실제로는 사용한 카메라의 렌즈나 이미지 센서와의 초점거리 등 카메라의 고유한 특성에 따라 달라진다. 따라서 초점거리(focal length) fx, fy와 광학 중심(optical center) cx, cy를 실제 영상으로부터 구할 필요가 있다. 본 연구에서는 크기를 미리 알고 있는 물체를 카메라로 촬영하여 특징점 대응 관계로부터 내부 파라미터를 구하여 사용하였다. cx, cy 는 영상의 폭과 높이의 절반 값으로 설정하여 사용하였다. 이 값들을 카메라 내부변수 매트릭스에 대입하면 다음 수식을 얻을 수 있다.

$$\begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

2.1 Kanade - Lucas - Tomasi feature tracker[6]

Kanade - Lucas - Tomasi feature tracker(이하 KLT Tracker)는 영상에서 detection된 물체, 특징점을 다음 프레임에서 찾는 알고리즘이다. KLT Tracker는 Optical Flow 방식으로 특징점을 추적하는 알고리즘으로 간단한 원리는 다음과 같다. 영상의 인접한 프레임에서 물체의 밝기는 변하지 않고 인접 픽셀들의 움직임은 서로 유사하다는 가정을 하면 수식(2)을 세울 수 있고, 테일러 급수를 적용하면 수식(3)와 같이 미분식의 합이 0이 되는 원리를 적용하여 인접한 프레임에서 특징점을 추적할 수 있게 된다.

$$I(x,y,t) = I(x+\triangle x, y+\triangle y, t+\triangle t) \quad (2)$$

$$I(x+\triangle x, y+\triangle y, t+\triangle t) = I(x,y,t) + \frac{\partial I}{\partial x}\triangle x + \frac{\partial I}{\partial y}\triangle y + \frac{\partial I}{\partial t}\triangle t \quad (3)$$
$$\frac{\partial I}{\partial x}\triangle x + \frac{\partial I}{\partial y}\triangle y + \frac{\partial I}{\partial t}\triangle t = 0$$

3. 실험 및 결과

3.1 Camera Calibration에 따른 실험 결과

크기를 알고 있는 격자무늬 사물을 이용 하여 실험을 진행하였다. 실험 결과는 그림3과 같다. 그림에서 빨간 선은 궤적 계산 결과, 파란색 선은 실제 측정 결과를 나타낸다. 그림2와 그림3을 비교했을 때 그림 3은 카메라 왜곡 보정 파라미터로 보정하지 않았을 때와 마찬가지로 회전 움직임 오차가 없지는 않지만, 보정하지 않았을 때보다 움직임이 정확히 추정되었음을 알 수 있다.

2.2 특징점 탐색 방식에 따른 실험 결과

기존 방법은 특징점을 대응시키기 위해 특징점을 한번 탐색 하고 프레임이 변할 때마다 이전 프레임의 특징점에서 다음 프레임의 특징점을 검색하기 때문에 프레임마다 FAST 알고리즘을 사용하지 않고도 특징점을 구할 수 있다. 하지만 이상적인 특징점 추출 및 대응 방식은 매 프레임마다 특징점을 추출하고 다음 프레임의 특징점과 매칭하는 방식이다. 그렇기 때문에 기존의 방식은 그림 3과 같이 매 프레임마다 특징점에 대응하는 과정에서 미세한 오차가 FAST 알고리즘을 통해 특징점을 다시 탐색하기 전까지 누적되게 된다. 그 결과 그림 3과 같이 오차가 누적되어 잘못된 움직임을 구하는 결과가 나타나게 되는 것으로 추정된다. 이런 오차를 줄이기 위해 본 논문에서는 특징점을 탐색하는 시점을 수정하여 오차가 누적되기 전에 재탐색이 되도록 특징점 탐색 기준을 특징점 개수가 아닌 특징점 탐색으로부터 지난 프레임 개수로 변경하여 실험하였다.

실험 결과 기존의 방식이 재탐색 과정에서 특징점이 평균보다 많이 잡히면 다음 재탐색까지 걸리는 프레임이 오래 걸려 오차가 많이 누적되는 문제점이 있었는데 새로운 방식은 오차가 쌓이기 전에 재탐색을 하므로 기존방식과 비교하였을 때 평균 재탐색 프레임 횟수가 비슷할 때 더 높은 정확도를 갖는 것을 알 수 있었다.

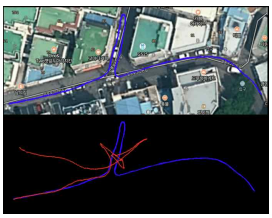

그림 2. 모바일 기기의 실측 데이터 실험 결과와 실제 이동 경로

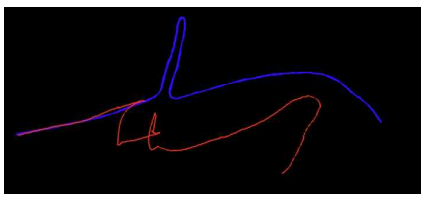

그림 3 카메라 보정값 측정후 적용한 실험 결과

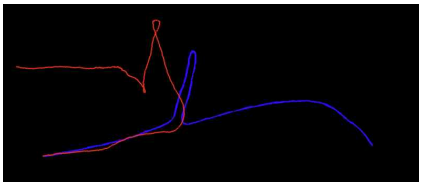

그림 4. 10프레임마다 특징점을 재탐색하여 계산한 이동 경로

그림 5 카메라 보정, 특징점 재탐색 기준 변경을 적용하여 계산한 이동 경로

4. 결 론

본 논문은 모바일 환경의 한정적인 리소스에서 다양한 방법을 통해 Visual Odometry를 계산하고, 그 결과에 대하여 비교하였다. 실험 결과 Camera Calibration을 바탕으로 Odometry를 추정할 경우 보정을 하지 않았을 때보다 정확한 결과를 얻을 수 있었다. 하지만 보정을 하지 않았을 때와 마찬가지로 계산 중 특정 프레임에서 급격한 회전 움직임 오차가 발생하는 문제가 존재했다. 하지만 해당 문제는 특징점 재탐색 기준을 변경하여 대부분 없앨 수 있어 그림 5와 같이 정확도를 상당히 높일 수 있었다. 하지만 여러 방법을 통해 정확도를 향상시켜도 여전히 직선 움직임에서는 비교적 정확하게 계산하지만, 회전 움직임에서 오차가 생기는 문제가 있기 때문에 IMU 센서나 GPS 정보와 결합하여 보정하는 방식으로 나머지 오차를 잡을 수 있을 것으로 기대된다.

참 고 문 헌

[1]Sebastian O.H. Madgwick, “Estimation of IMU and MARG orientation using a gradient descent algorithm”, 2011

[2]Forster, Christian, Matia Pizzoli, and Davide Scaramuzza. "SVO: Fast semi-direct monocular visual odometry." 2014 IEEE international conference on robotics and automation (ICRA). IEEE, 2014.

[3]http://avisingh599.github.io/assets/ugp2-report.pdf

[4]https://github.com/avisingh599/mono-vo

[5]E. Rosten and T. Drummond, “Machine learning for high speed corner detection,” in 9th Euproean Conference on Computer Vision, vol. 1, 430 - 443p, 2006.

[6] C. Tomasi. T. Kanade, "Detection and tracking of point features.", Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.

본 연구는 문화체육관광부 및 한국컨텐츠진흥원의 2022년도 문화 기술연구개발 지원사업으로 수행되었음. (R2020060002)