

SLAM 기술을 활용한 가상 환경 복원 및 드론 레이싱 시뮬레이션 제작

Development of Drone Racing Simulator using SLAM Technology and Reconstruction of Simulated Environments

박 용 희¹·유 승 현²·이 재 광²·정 종 현²·조 준 형²·김 소 연²·오 혜 준²·문 형 필[†]
Yonghee Park¹, Seunghyun Yu², Jaegwang Lee², Jonghyeon Jeong², Junhyeong Jo²,
Soyeon Kim², Hyejun Oh², Hyungpil Moon[†]

Abstract: In this paper, we present novel simulation contents for drone racing and autonomous flight of drone. With Depth camera and SLAM, we conducted mapping 3 dimensional environment through RTAB-map. The 3 dimensional map is represented by point cloud data. After that we recovered this data in Unreal Engine. This recovered raw data reflects real data that includes noise and outlier. Also we built drone racing contents like gate and obstacles for evaluating drone flight in Unreal Engine. Then we implemented both HITL and SITL by using AirSim which offers flight controller and ROS api. Finally we show autonomous flight of drone with ROS and AirSim. Drone can fly in real place and sensor property so drone experiences real flight even in the simulation world. Our simulation framework increases practicality than other common simulation that ignore real environment and sensor.

Keywords: Augmented Reality, SLAM, Drone Racing, Simulation, Autonomous Flight

1. 서 론

최근 로봇의 알고리즘을 정밀하게 검증하기 위해 보다 더 정확한 시뮬레이션의 필요성이 대두되고 있다. 로봇이 실제 산업과 서비스 분야에서 사용되기 위해서는 사전의 안전 실험과 성능 평가가 필수적이다. 대표적인 로봇 시뮬레이션으로 Gazebo^[1]가 사용된다. 이는 ODE 물리 엔진을 제공하여 중력, 마찰과 같은 물리적 특성을 반영하며 로봇의 동역학적 움직임을 관장한다. 이 물리 엔진이 현실성을 더 잘 표현할 수 있어야 만 알고리즘의 정밀한 검증이 가능하고 실제 로봇에 효율적으로 이식할 수 있다. 하지만 위의 시뮬레이션은 로봇의 동역학적 특성과 환경적 특성을 잘 반영하지 못하고 있다. 특히

Gazebo의 드론 시뮬레이션인 PX4 SITL Gazebo는 단순화된 드론 로봇을 구현하였기에 이는 현실과 큰 차이가 있다. 실제 환경의 경우, 센서의 잡음이나 바람과 같은 외란은 드론에게 치명적인 결함으로 작용할 수 있다. 이는 시뮬레이션에서 검증한 알고리즘을 실제 로봇에 바로 이식할 수 없다는 단점을 나타낸다.

이에 우리는 사용자에게 실제 환경에서 드론을 날리는 느낌을 주는 증강 현실 콘텐츠를 제공하기 위해 실제 환경을 가상 환경 속에서 복원하였다. SLAM 및 Depth 카메라의 데이터를 이용하여 주변 3차원 환경을 포인트 클라우드 형태로 mapping하였다. 해당 데이터를 가상 환경 속에서 복원한다면 실제 장소와 똑같은 구조 속에서 드론을 비행하는 경험을 제공할 수 있다.

포인트 클라우드 데이터를 복원하기 위해 Unreal 엔진을 이용하였다. Unreal의 렌더링 엔진(Rendering Engine)은 최고 수준으로 인정받고 있으며 이는 화면을 통해 사용자에게 전달되는 시각 효과뿐만 아니라 가상 환경 속 드론의 카메라 센서에도 좋은 효과를 보인다. 또 Unreal의 물리 엔진을 이용하여 보다 더 정확한 물리적 특성을 반영할 수 있다. Microsoft사의

Received : Jan. 25. 2021; Revised : Mar. 22. 2021; Accepted : Jun. 17. 2021

* This project was funded by Korea Robotics Society (KROS), and is currently supported by the publication grant

1. Principal Researcher, SKKU, Suwon, Korea (qkrdydgm93@naver.com)

2. Researcher, SKKU, Suwon, Korea (littleyu9403@gmail.com, jklee971209@naver.com, jungjh404@g.skku.edu, asdfg3128@gmail.com, encyrine@g.skku.edu, ojune575@naver.com)

† Associate Professor, Corresponding author: Mechanical Engineering, Sungkyunkwan University, Suwon, Korea (hyungpil@skku.edu)

AirSim^[2]을 이용함으로써 드론 Controller의 architecture를 쉽게 구현할 수 있었다. AirSim은 본래 AI 및 딥러닝 연구 목적으로 만들어졌으나, 실제 드론과 똑같은 Interface를 제공한다는 장점을 가지고 있다. AirSim은 연구용 드론에서 가장 많이 사용되는 PX4 firmware를 위한 Interface를 제공하고 있으며 이를 이용하여 HITL (Hardware in the Loop)과 SITL (Software in the Loop) 모두 구현하였다. 또한 AirSim은 로봇프로그래밍을 위한 ROS wrapper를 제공하고 있다. 이 논문 뒤에서 우리는 Unreal 엔진 속에서 ROS 프로그래밍을 이용하여 자율비행으로 드론을 제어한 것을 보일 것이다.

이 논문의 2장 본론에서 연구의 방법과 결과에 대해서 다룬다. 3장에서 결론, 4장에서 추후 연구를 다루고 마무리한다.

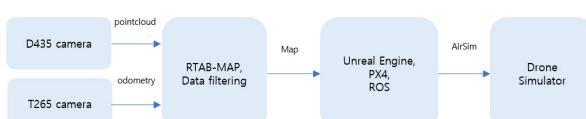
2. 본 론

2.1 방법

이 연구를 진행한 과정들은 [Fig. 1]과 같다. [Fig. 1]은 시뮬레이터를 제작하는 과정을 나타내는 순서이다. 우선 RTAB-Map을 이용하여 포인트 클라우드 데이터를 수집하였으며 데이터 그 자체에는 noise가 있기 때문에 이를 필터링하였다. 필터링된 포인트 클라우드 데이터를 Unreal 엔진 속에서 복원하고 그 환경 안에서 수동 및 자동 조종을 구현하였다. 특히 로봇 프로그래밍인 ROS를 이용한 드론 자율비행을 가능케 하였다.

2.1.1 RTAB-Map 구동

SLAM (Simultaneous Localization and Mapping) 방법 중 RTAB-Map (Real-Time Appearance-Based Mapping)^[3]을 사용하여 mapping을 진행하였다. 카메라는 Intel Realsense의 T265와 D435i 모델을 사용하였다. D435i 모델은 RGB-D 카메라로 카메라 주변의 색상과 거리 정보를 바탕으로 mapping을 하는 것은 수월하지만 IMU 센서가 있음에도 불구하고 위치인식에는 비교적 취약하다. 따라서 회전과 이동에 강건하도록 성능을 높이고 이미지 처리의 최적화를 위해 IMU 센서와 2 종류의 VPU (Vision Processing Unit)가 포함된 트래킹 카메라(tracking camera)인 T265 모델을 추가로 사용하였다. RTAB-Map은 odometry approach 와 같은 설정들을 사용자가 변형할 수 있지만 이 경우에는 별다른 변형 없이 기본 세팅을 사용하였다.



[Fig. 1] Simulator manufacture pipeline using SLAM

Loop closing과 동시에 mapping이 끝나도록 설정할 수 있지만 이 경우 비교적 큰 크기의 공간을 mapping할 경우 드리프트가 많이 쌓이게 되어 정확성이 떨어진다. 따라서 전체 공간을 소규모로 구획하여 mapping을 진행하였고 이후 Unreal 엔진 상에서 맵을 결합하였다.

2.1.2 포인트 클라우드 필터링

필터링은 두 가지 방식을 이용하여 진행하였다. 먼저 Nearest Neighborhood 알고리즘을 통해 필터링을 진행하여 주변과 너무 멀리 떨어진 데이터는 배제하도록 하였다. 또한 포인트 클라우드의 데이터를 새로 구축하는 MLS (Moving Least Squares) 알고리즘을 이용하여 표면을 고르게 하였다.

2.1.3 PX4의 적용

AirSim은 다양한 드론의 컨트롤 방식을 지원하나 이번 프로젝트에서 주로 PX4 Pixhawk의 보드를 이용한 HITL (Hardware In The Loop) 모드와 PX4 Firmware를 이용한 SITL (Software In The Loop)을 이용했다. PX4 Pixhawk의 보드 드론 조종기 이용하여 가상공간에서의 드론을 조종하거나 가상의 PX4 Firmware를 빌드하고 이를 ROS와 연동하여 조작한다.

2.1.4 AirSim의 Data 연동

AirSim은 setting.json 파일을 통해 드론에 장착된 카메라의 개수와 위치 등을 편집할 수 있다. 개개의 카메라에서 받아오는 카메라 데이터의 해상도, FOV 등을 설정해줄 수 있고, 카메라 데이터의 종류도 결정해줄 수 있다. 또 AirSim의 ROS wrapper를 이용해서 MAVROS와 통신이 가능하다. 예를 들어 시뮬레이션에 제공하는 카메라, odometry 좌표 등의 데이터를 ROS topic으로 변화하여 이용할 수 있다. 이를 통해 PX4와 ROS를 연결하여 드론을 조종하게 된다.

2.1.5 비행 평가 시스템 구성

기존의 드론 레이싱 코스와 자율주행의 성능을 평가하기 위한 장애물 코스들을 참고하여 Unreal 엔진의 가상 공간에 코스를 구성했다. 또한 Unreal 엔진의 Blueprint 기능을 이용하여 코스를 주파하는 시간을 측정하기 위한 시스템을 만들었다.

주행시간을 측정하는 Blueprint파트는 사용자가 컨트롤 하는 Mesh가 Starting Collision Box를 이탈하는 시각에 측정을 시작하여, 코스를 완주하고 Starting Collision Box와 다시 접촉한 시각의 완주시간을 체크 한다.

Collision Box와 사용자가 컨트롤 하는 Mesh가 접촉하면 Collision Box가 다음 순서의 체크 포인트의 좌표로 이동하며 이때의 이동한 좌표의 순번으로 사용자의 코스 진행정도를 파악한다.

2.1.6 자율비행 알고리즘 적용

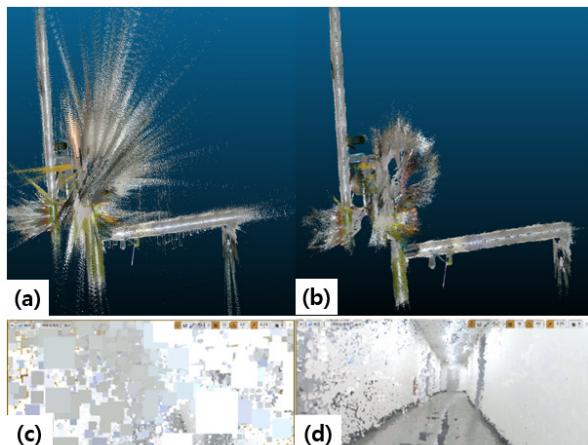
기준에 Gazebo에 적용된 드론 회피 알고리즘 자율비행 알고리즘^[4]을 AirSim에 적용해보았다. AirSim은 ned좌표를 주로 사용하는 반면 PX4는 enu좌표를 주로 사용하기에 알고리즘을 적용할 때 ned나 enu 중 하나로 통일해서 사용해야 한다. 시뮬레이션 환경은 Gazebo를 이용하는 대신 Unreal 엔진을 이용한다.

2.2 결과

2.2.1 Mapping한 결과와 Post-Processing

드론을 날리기 위해 상당히 큰 공간을 mapping 하였다. 성균관대학교 산학협력관을 Mapping한 결과는 다음 [Fig. 2]와 같다. [Fig. 2]의 (a)와(b)는 포인트 클라우드 데이터 전체를 위해서 본 평면도의 필터링 전후 비교 모습이며, [Fig. 2]의 (c)와(d)는 카메라 시점에서 본 복도의 필터링 전후 비교 모습이다.

큰 공간을 처리하다 보니 loop closing 시 drift가 매우 크게 발생하는 것을 여러 사례를 통해 확인할 수 있었다. 이를 방지하기 위해 구역을 세밀하게 나눠 loop closing을 자주 시켜줌으로써 더욱 정확하게 mapping을 실행하였다.



[Fig. 2] (a) Pointcloud mapping before outlier removal, (b) Pointcloud mapping after outlier removal, (c) Corridor before outlier removal, (d) Corridor after outlier removal



[Fig. 3] Corridor pointcloud after smoothing

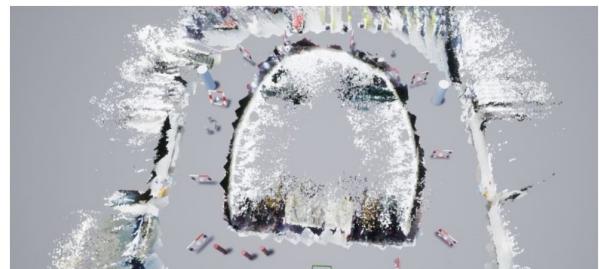
또 [Fig. 2(a)]에서 볼 수 있듯이 불필요한 점들이 많이 있을 뿐만 아니라 [Fig. 2(c)]의 경우 형체를 알아볼 수 없을 정도로 outlier가 많았다. 이를 해결하기 위해 Rtabmap의 Nearest-Neighborhood 알고리즘을 이용하여 [Fig. 2]의 (b)와 (d)와 같이 outlier를 제거하였다.

그 다음 [Fig. 3]와 같이 MLS알고리즘을 이용하여 Smoothing 작업을 진행하였다. Voxel size의 4배의 값을 주어 점의 개수가 4배 늘어났지만, 벽이 튀거나 하는 것 없이 좀더 부드럽게 보정하여 처리하였다.

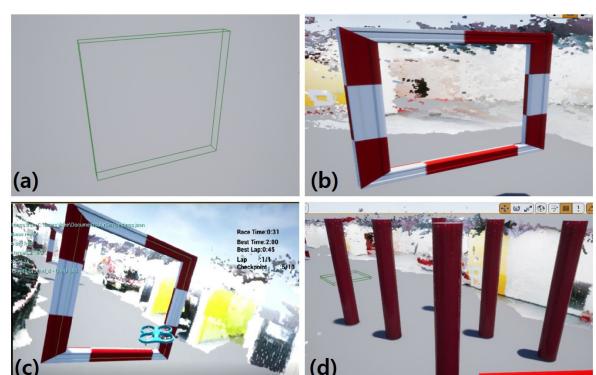
2.2.2 비행 평가 시스템 Unreal상에 적용

Unreal 엔진의 Asset이나 다른 프로그램에서 제작한 Mesh, 포인트 클라우드 등을 이용하여 레이싱 코스를 만들어낼 수 있다. [Fig. 4]는 포인트 클라우드 맵 코스 적용 예이다.

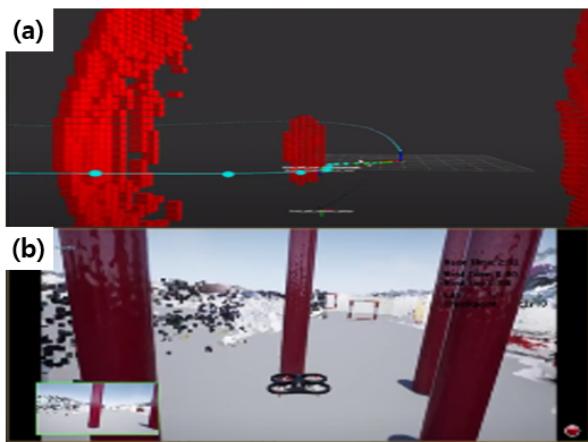
[Fig. 5]은 Unreal 맵 위의 Gate와 장애물을 설치한 것이다. [Fig. 5(a)]의 Collision Box는 사용자의 체크 포인트의 통과 여부를 파악하기 위한 도구이며 Collision Box가 위치한 공간에 사용자가 식별하기 편한 [Fig. 5(b)]의 Gate를 맵 상에 배치하여 사용자의 편의성을 높였다. 또 사용자가 시뮬레이션 진행 상황을 한눈에 파악할 수 있도록 [Fig. 5(c)]와 같은 구성의 HUD (Head Up Display)를 만들었다. HUD에는 경과 시간, 이전까지의 최고 랩타임과 최고 완주시간, 현재 랩과 몇 번째 체크 포인트로 가야 하는지 등을 나타내고 있다.



[Fig. 4] Racing track of pointcloud data in Unreal Engine



[Fig. 5] Unreal Engine system composition. (a) Collision Box, (b) Gate, (c) HUD, (d) obstacles



[Fig. 6] (a) Autonomous flight visualization in Rviz, (b) Autonomous flight in Unreal Engine simulator

2.2.3 자율비행 알고리즘의 AirSim 적용

[Fig. 6]은 자율비행 알고리즘을 적용한 모습이다. 드론이 장애물을 회피하는 과정에서의 AirSim화면과 rviz상 화면이다. 예시에서 occupancy grid를 rviz상에 출력해주고, 새로운 trajectory를 경로상에 그려준다. 이를 따라 Unreal 엔진 상의 드론이 움직인다.

3. 결 론

이번 논문에서는 RGB-D 카메라를 이용한 포인트 클라우드 수집과 이를 Unreal 엔진에서 복원시켜 자율주행 드론 시뮬레이션 환경 구축을 설명한다. 기존의 많은 시뮬레이션들과 다르게 우리는 IMU를 가진 두대의 Depth 카메라를 사용하여 실제환경과 동일한 색과 사이즈, 거리 등의 데이터를 가지고 있는 포인트 클라우드 데이터에 초점을 맞추었고 구역을 세밀하게 나눠 loop closing을 자주 시켜줌으로써 map을 robust하게 만들어주었다. 또한 large scale의 구역을 mapping하기 때문에 outlier가 많음을 확인할 수 있었는데 Nearest-Neighborhood 알고리즘을 이용하여 outlier를 필터링하여 더 정교하게 시뮬레이션 환경을 구축하였다. 구축된 시뮬레이션에 Unreal 엔진의 기능을 활용하여 장애물들을 추가하여 드론의 자율주행을 평가할 수 있는 코스를 구현하였다. 또한 AirSim의 가상 카메라 데이터를 ROS로 연동시켜 시뮬레이션 상의 포인트 클라우드에 대하여 depth데이터를 인식하여 사람의 개입없이 자율주행 알고리즘을 검증해볼 수 있었다. 이에 대해 rviz와 Unreal 엔진 상에서 동시에 출력하여 확인해볼 수 있었다.

4. Future Work

본 연구는 추후 실제 환경과 더 유사한 환경을 mapping할 수 있는 기술로 발전시킬 예정이다. 현재 로봇 시뮬레이션의 경우 매우 이상적인 환경 속에서 사용자가 센서 특성에 임의의 분산을 적용하여 사용하고 있다. 하지만 이는 실제 센서의 특성을 완벽히 반영하지 못한다는 한계가 있으며 이로 인해 로봇 시뮬레이션 겸증 성능이 떨어진다고 할 수 있다. 하지만 본 연구에서는 실제 카메라로 얻어낸 Depth의 포인트 클라우드 데이터를 그대로 복원하는 동시에 RGB 이미지의 데이터는 깔끔하게 렌더링하여 실제 환경에서 로봇이 얻어낼 수 있는 데이터를 가상 환경 속에서도 똑같이 얻어낼 수 있도록 할 예정이다. 현재 포인트 클라우드 수집단계에 있어, 유리창과 거울의 depth를 제대로 표현하지 못하거나 비슷한 환경의 다른 장소에서 loop closing이 자동적으로 발생하는 한계점이 존재한다. 향후 두개의 depth 카메라 이외에 라이더 및 초음파 센서를 추가하여 더욱 정교한 포인트 클라우드를 수집할 계획이다. 또한 mapping하는 과정에서 등장하는 사람의 경우 필터링으로 제거 되지 못하고 있다. 하지만 사물인식이나 image segmentation을 이용하여 추후 이러한 outlier 요소들을 제거할 수 있는 방안을 개발할 예정이다.

사 사

본 연구는 정보통신기획평가원의 대학ICT연구센터지원사업의 지원을 받아 수행하였습니다(IITP-2020-0-01460, Simulated Reality 콘텐츠 구현을 위한 핵심기술 연구 및 인력양성).

References

- [1] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, DOI: 10.1109/IROS.2004.1389727.
- [2] S. Shah and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," *Field and Service Robotics 2017*, [Online], <https://arxiv.org/abs/1705.05065>.
- [3] M. Labbe and F. Michaud "RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416-446, 2019, DOI: 10.1002/rob.21831.
- [4] V. Usenko and D. Cremers. "Real-Time Trajectory Replanning for MAVs using Uniform B-splines and 3D Circular Buffer," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, DOI: 10.1109/IROS.2017.8202160.

**박 용 희**

2019 성균관대학교 기계공부(학사)
2019~현재 성균관대학교 기계공학과(석사)

관심분야: Robotics, Drone, Autonomous flight

**조 준 형**

2018~현재 성균관대학교 기계공학부(학사)
2019 서울 하드웨어 해커톤 2등
2020~현재 성균관대학교 소프트웨어공학
복수전공(학사)

관심분야: Robotics, SLAM, Autonomous Driving, Computer Vision &
Deep Learning

**유 승 현**

2014~현재 성균관대학교 전기전자공학부
(학사)
2020 사회문제해결 메이커톤 2등

관심분야: Visual Recognition, Artificial intelligence, Computer
Simulation

**김 소 연**

2018~현재 성균관대학교 기계공학부(학사)

관심분야: Robotics, SLAM, Autonomous Driving, Computer Vision &
Deep Learning

**이 재 광**

2016~현재 성균관대학교 기계공학부(학사)

관심분야: Robotics, Drone, SLAM

**오 혜 준**

2016~현재 성균관대학교 기계공학부(학사)

관심분야: Robotics, Visual Recognition

**정 종 현**

2016~현재 성균관대학교 기계공학부(학사)
2020 판교 자율주행모빌리티쇼 3등

관심분야: Robotics, Autonomous Driving

**문 형 필**

1996 포항공과대학교 기계공학과(공학사)
1998 포항공과대학교 기계공학과(공학석사)
2005 Mechanical Engineering, University of Michigan, Ann Arbor(공학박사)

관심분야: Robotic Manipulation, Polymer-based sensor and
actuators, Visual recognition