유희빈

2020년 12월 1일

3D scanning과 VSLAM기술을 사용힌 CQB 시뮬레이션 시스템에 대하여

Abstract

기 시스템은, 한국 육군의 모의전투 훈련과, 배틀그라운드라는 게임에 영감을 받아. 집에서 할 수 있는 서바이벌 게임을 구현한 것이다. 이의 구현은 제일 현대적이고 사실적인 방법론을 따르기 위해, 3D scanning과 VSLAM기술을 사용하였으며, 제일 핵심적으로 처리한 문제는 실시간성과 소켓 통신을 중심으로 한 pipelining이다. 기 논문에서는 기획의 의도와 장점을 나열하기보다는, 기술적인 핵심과 세부구현 사항들에 대하여 정리를 하고, 이 프로젝트를 수행하면서 누적된 경험들에 대해서 설명하려고 한다.

Introduction

CBQ환경이라고 정의하자, 이는 이 모든 시스템이 동작해야하는 환경이다. 실내 공간에서, 경기자 몇명이 2개의 팀을 구성하여, 상대 팀을 총을 사용해 전멸시키는 경기환경을 말한다고 정의하고자 한다. 이것을 게임 환경으로 만들려면 두 정보가 필요하다. 첫째, 실내 공간이 어떻게 구성되는가? 둘째 각 경기자는 어디에서 어떤 방향으로 총구를 겨누고 있는가? 이 두 문제를 해결하기 위해 두 기술을 사용하였다. 3D scan과 VSLAM기술이다.

3D scan의 경우, 그래픽스 분야의 랜더링 핵심 기술 중 하나로, A volumetric method for building complex models from range image에서 음함수 곡면의 수학적 성질을 이용한 레이저 스케닝을 베이스라인으로 사용한 논문들을 이 프로젝트 진행에 참조되었다. 이의 발전사에 있는 RGBD 센서 바탕 스케닝을 사용한 논문 BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration을 응용하여 만들었다.

VSLAM의 경우, 로보틱스의 핵심 기술이라고 볼 수 있는데, odometry와 비슷한 개념으로 여러가지 기능을 수행하지만, 본 프로젝트에서 집중한 부분을 정리하자면, 카메라가 어디 있고, 어떤 방향으로 이동하는지, 즉 world의 원점과 카메라 사이의 R|t행렬을 구하는 시스템이라고 요약 할 수 있겠다. 이 부분은 ORB SLAM2의 공개 코드를 바탕으로 제작하였다.

Related Work

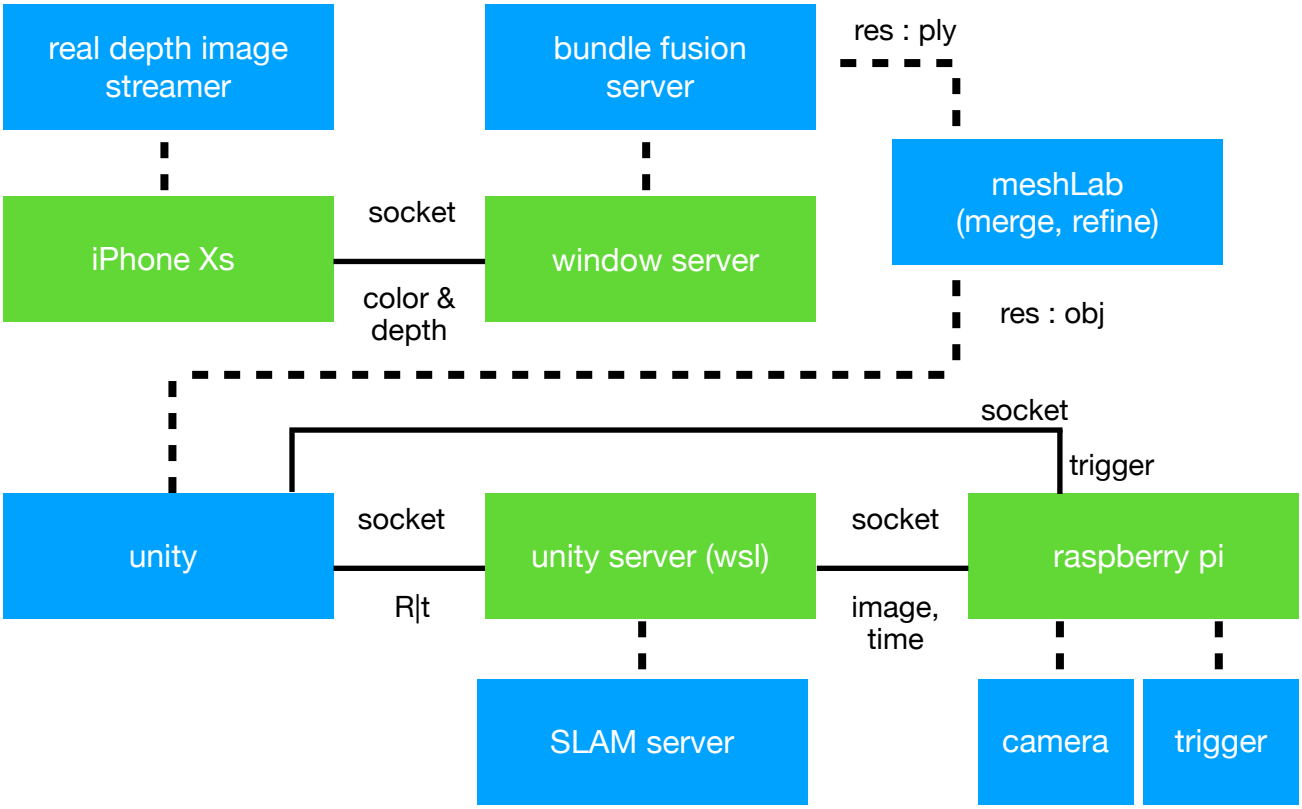
이 시스템의 구현을 위해 체크한 관련 연구를 소개토록 하겠다.

스켄쪽 base는 A volumetric method for building complex models from range image이다. 레이저 기술을 사용하여,여러 이미지의 연속과, 깊이 데이터를 사용해, 오브젝트를 가운데 두고, 그를 중심으로 한바퀴 돌아 3D mesh를 생성하는 시스템이다. 이를 scale 관계 없이 스켄할 수 있도록 확장한 연구는 real-time 3d reconstruction at scale using voxel hashing으로, 넓은 공간에서 스켄을 할 수 있는 가능성을 열어주었다. 그 다음에 이를 조금 더 발전시킨 논문은 BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration이다. 같은 기능을 수행하는 시스템이지만, 조금 더 좋은 성능을 내고 있다. 이 구현의 코드는 공개되어 있지만, 너무 방대하여 수정을 가하기는 어렵다는 평을 덧붙인다. 위 논문들에 대한 이해를 도와준 국내 논문 “음함수 곡면을 이용한 모델링” - 박미경, 이의택도 참고할만 하다. 이 논문에는 3D scanning기술의 가정과 모델들에 대해 상세히 설명되어있다. 추가적으로 검토했지만, 본 프로젝트에 첨하지 않은 논문 multi-view convolutional neural networks for 3D shape recognition으로, 한 물체를 돌아가면서 찍을때, 그 물체가 무엇인지 재인하는 기술에 대해 다룬다. 여기서 현대의 3D 그래픽스의 스케닝 연구분야와 3D 비전의 차이점에 대해 엿볼 수 있는 기회가 있는데, 3D 스케닝 연구의 경우 카메라로 찍은 사진을 알고리즘적으로 이어 붙여서 가시의 mesh를 만드는데 집중하고 있다는 것이고, 3D 비전의 경우 카메라로 사진을 찍는 것은 똑같지만, 신경망 등의 판단 기술을 사용하여 구조를 재인한다는데 - 굳이 가시적일 필요는 없다는데 - 집중하고 있다는 것이다.

VSLAM에 대해서는 다음과 같은 논문을 참고하였다. 처음으로는, real-time stereo visual odometry for autonomous ground vehicles를 보았는데, 이 논문에서는 이동 차량에 부착된 카메라를 추적하는 것을 다룬다. 핵심은 SURF특징점을 잡고, 정합을 통해 이동을 확인한다는 것이다. 이런 특징점 기반 추적은 대부분의 odometry, slam 논문에서 수행하게 되는 핵심 작업이다. 하지만, 하드웨어적인 이유로 양안 카메라를 사용 할 수 없게 되어, 직접 사용은 하지 못했다. 다음으로는 실제 구현을 위해 사용된 논문 orb-slam a versatile and accurate monocular slam system이다. 이 논문은 3편까지 나와있는데, 기 시스템에서는 ORB-SLAM2를 사용하였다. 그 이유는 ORB-SLAM2를 바탕으로 한 Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments라는 신경망을 이용한 법선벡터 추출, 점 추정, 선 추정을 사용한 실내 slam 기술의 구현체를 사용하려고 시도했다가, 공개된 소프트웨어가 현 시점 오류가 있어 사용하지 못한 것 때문이다. (둘은 동일한 함수 시그니쳐를 공유하고 있어 호환 가능하다.) 추가적으로 고려한 건 CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction이다. 이것도 신경망을 전처리용으로 사용하는데, 구현체가 제대로 공개되지 않아서 사용하지 못하였다.

본문

이 시스템의 아키텍쳐는 두 부분으로 나뉜다.



환경의 mesh를 위해 공간을 스캔하는 윗쪽 부분과
그 결과인 obj파일을 입력받아서, 사용자의 총이 어디에 있고, 어디를 향하는지, 총알이 상대방에게 맞았는지, 장애물에 맞았는지 확인하는 부분 두개로 나뉜다.
본문에서는 이 두 부분의 구현 방식과 거기서 나온 노하우들에 대해 언급하도록 하겠다.
언급되는 모든 것은 https://github.com/Team-AllyHyeseongKim을 확인하기 바란다.

본문 - scanner

우선, 편의를 위해서 초기에 기획하기에는, 라즈베리 파이에서 들어온 이미지를 바탕으로 스켄을 수행하려고 했으나, 깊이 맵도 여러가지 종류가 있다는 사실을 알게 되어 절대 깊이맵을 얻을 수 있는 iPhone을 파이프라인에 배치하게 되었다.
bundleFusion시스템을 커스템 센서로 구동하기 위한 요구사항은 깊이 이미지와 RGB이미지를 즉, RGBD이미지를 입력하는 것이다. 이때, 유의할 점은 상대 깊이 이미지가 아니라 절대 깊이 이미지 (16-bit, 각 elt가 mm 단위로 되어있는)라는 것이다. 이 사실을 깨닫는 데까지 많은 실험이 있었는데, 초기에 opencv의 block matching으로 나온 깊이맵이나, Hierarchical Deep Stereo Matching on High Resolution Images의 구현체에서 나온 깊이맵으로 실행을 시도하려고 했으나, 상대값이라는 사실을 간과하여 제대로 작동시키지 못했다. 동시에 stereo calibration의 rectify라는 기술도 카메라의 배치가 '—'자이지 않으면, 많은 영상 부분들이 소실된 결과를 얻게 된다. 또한, 추가적으로 남기자면, 많은 이미지는 켈리브레이션의 오차만을 상승시킨다는 점, 최대한 이미지 패턴이 영상에 꽉 차도록 해야 한다는 점, DSLR을 사용하지 않는이상, 끝에서 끝까지 센서를 읽는데 시간이 걸리므로 motion blur문제가 일어날 수 있어, 정지 상태에서 촬영할 것을 당부한다.
결론적으로 절대적인 metric으로 깊이값이 나오는 센서를 확인했더니 IR을 이용한 센서에서만 가능하다고 한다. 이를 위해서 iPhone의 realdepth센서를 사용하게 된 것이다.
(저장소 iphone-realdepth-streammer을 참고하라)

이때 체크할 사항은 아이폰 깊이 이미지는 color이미지를 위해 방사왜곡 - 주변이 더 많이 보이게 - 되어있다는 것이다. 이 방사왜곡은 object-c단에서 수정 가능하며, 처리된 사항을 png로 압축하여 bundle fusion쪽으로 http사용해 보내는 작업을 수행하였다.
bundle fusion에서 처리하기 전에 이를 테스트하기 위해 node.js에서 간단하게 서버를 구현하였다. (real-depth-streamer-server) bundle fusion에서는 cpprest라는 마이크로소프트사의 라이브러리를 사용하여 비동기로 이미지를 수신한다. - 참고할 사항은 수신 queue가 비동기로 작동하는데에 safe하지는 않다, 근데 워낙 이미지 크기가 커서 문제는 없었다. - 물론 이 쪽의 코드에도 test를 위해 HDD에서 이미지를 불러와서 처리할 수 있는 버전을 만들었다. (http-image-server-for-bundle-fusion-scanner) 그리하여 위 아키택쳐의 윗 부분이 완성되었다. 이를 사용하여 공간을 스켄하면 ply파일을 획득 할 수 있다. 다만, 알고리즘의 특성상 오래 캡쳐를 수행하면 카메라의 location 추정이 점점 어려워지는 경향을 확인 할 수 있었다. meshLab이라는 프로그램을 사용해 여러번에 걸쳐 촬영해 merge하는 대안도 있다. 그 후 obj파일로 export하면 unity에서 로드 가능하다.
추가적으로 아이폰을 대체할 수 있는 방법이 있다면 (stereoStreamer-with-rectify) 을 사용하길 바란다. 기타 시도들이 궁금하다면 (vision-utils-calibrator-depth-map-deblur-odometry-)을 확인하길 바란다. 다른 데이터셋이 정상적인지 한번 bundleFusion에 .sens형태로 입력하고자 한다면 (sens-file-compressor-and-decompressor)을 확인하라.

본문 - SLAM

이 시스템은 우분투에서 작동한다. 윈도우에서도 빌드 가능하다고 하지만, 생각보다 많이 복잡하여 우분투를 실험 환경으로 골랐다. 다만, 윈도우 pc에서 작업을 완료하기 위해서 wsl에서 작업하였다. 이때 문제는 네트워크 연결이 가상 NIC를 이용해 연걸되므로 호스트의 네트워크와 분리되어있다는 점이다. 이때 forwarding작업을 수행하길 바란다. (orb_slam2-for-socket-streaming)저장소의 setup파일을 사용하라, 이 파일에서 port 번호를 더 지정하여 포워딩 시킬 수 있는데, 처리를 수행하기 전에 확인하길 바란다. 또한, 디버깅을 위해 ssh display프로그램 - xming을 사용하였다. 그 후 위 그림과 같은 식으로 소켓 연결, 큐잉 구조를 만들었다. (rasp-socket-trigger-and-image-sender)저장소가 라즈베리 파이에서 작동하는 스크립트이고, (unity-game)가 최종 게임이다. 추가적으로 간단한 튜토리얼로 (unity-python-socket), (socket-receiver-sample-cpp) 저장소를 만들었다.

이로서 전반적인 처리구조에 대한 설명을 마친다.

고찰

이후로부터는 프로젝트를 진행하면서 나온 노하우들과 경험에 대해 정리하겠다.
사실 이 시스템은 정확도와 안정성이 그리 높지 않다. 전혀 고려할 시간이 없었기도 하고, 쇼케이스를 위해 만든 것이 아닐까 라는 생각도 있다.

고찰 - 소프트웨어 개발이 해야 할 일 - 사용 기술의 이해

이는 초기에 했던 착각 때문인데, 논문의 구현체는 이해 가능하고, 어떤 데이터에도 수이 적용되며, 어느 곳에서도 실행 가능한 컴포넌트, 믿을 수 있는 덩어리라고 생각했다. 하지만, 모든 코드들은 happy 시나리오에 대한 구현만을 다루고 있었다. 즉, 수학적인 증명을 제공하는 것이 최상이라고 생각하고 만들어진 논문을 데모하려고 만든 구현체이기 때문에, 증명에 들어간 가정 외의 상황에서는 작동 할 수 없기 때문이다. e.g. structure SLAM에서는 맨하탄 world 가정을 하고, 모델을 구성하였는데, 만약 구불거리는 것들이 가득한 곳이라면 이 시스템은 작동 할 수 없을 것이다. 물론, 여러가지 정리들이 말하듯이, 공학은 널리 퍼트려진 엔트로피를 원하는 한 곳으로 모으는 것이기 때문에 어쩔 수 없는 것이지만, 소프트웨어로 만들어서 최종 사용자에게 전달할 때에는 그런 가정들에 대해서 재고 할 필요가 있겠다.
즉, 만들어진 것이 언제 왜 무엇을 위해 만들어졌는지 생각하고, 우리의 시스템의 커버리지에 부합하는지 확인하여 여러 대안을 가지고 있어야 한다고 생각한다.

고찰 - 소프트웨어 개발이 해야 할 일 - 사용 하드웨어의 이해

소프트웨어는 사용하는 하드웨어를 정확하게 이해하고, 하드웨어의 오류 경우의 수에 대해서 모두 파악하고 어떤 예상되는 경우에 대해서도 문제 없게 대처하여 강하고 튼튼한 integration을 제공해야 한다. 하지만, 하드웨어의 한계에 대해서도 잘 알고 있어야겠다. 우리는 몇개의 논문과 쇼케이스를 바탕으로 IMU가 위치를 추정해 줄 것을 믿고 있었다. 다만, 구현체가 없다는 사실이 조금 걸렸고, 논문들을 바탕으로 이를 구현해서 사용하려고 했다. 하지만, 정말 많은 필터링이 필요했고, 그럼에도 불구하고 정상 작동을 거의 보장 할 수 없어 도입을 포기하였다. 앞으로의 과제에서는 두세개의 구현이 된 라이브러리 수준의 무언가가 없으면 그것이 조사되었고, 검증되어 믿을 수 있다고 말하기 어렵다는 사실을 알아야 한다.

고찰 - 데이터셋의 중요성

소프트웨어라고 하면, 최종 사용자가 어디서든 사용 할 수 있다는 것이 매우 중요한 우선순위로 고려되어야 한다. 근데, 데이터셋으로 테스트한 것들을 사용하다보면, 실제 카메라 입력 데이터가 돌아가지 않는 경우가 꽤 생겼다. 이런 것들에 대해서 라이브러리가 아닌 것들을 사용할 때 명확하게 인지하고 있어야겠다. 반대로, 데이터셋이 있다는 것이 매우 중요할 때가 있는데 실제 카메라 및 기타 장비를 당장 구성하기가 어려운 경우나 HW 구성이 틀린건지 SW구성이 틀린건지 확인하고자 할 때, mock객체를 잠시 배치하고 돌리려고 할 때가 될 것 같다. 이 둘의 벨런스는 상당히 중요하다.

고찰 - 저수준 언어에 대한 생각

c++에 대한 환상이 만연하다. “실력있는 프로그래머라면 c++을 할 줄 알아야 한다”, “c++은 속도에 좋은 언어이다.” 등등. 이 두 문장에 대해 비판을 제기하고자 한다. 이 문장들은 아주 특수한 경우에만 일어나는 가정에 기반하는데, 특정 기법, 기술 하나를 직접 구현할 때에 그것에 대한 명확한 노하우가 있어야 한다는 것이다. 그렇지 않은 이상 어떤 상황에서도 저수준 언어는 적합하지 못하다. 더욱이 여러 처음 수행하는 것들을 구현할 때, python이 아니라 c++을 쓴다는 것은 ‘바보짓’에 가까울 것이다. - 아무리 boost, eigen등의 라이브러리가 있다고 해도, 빌드부터 메모리 관리 실수까지 너무 많은 리스크가 있다. - 구현 후 병목 부분만 최적화하는 접근법이 제일 적합한 판단이라고 생각한다.

그런 의미에서 numpy & conda의 python환경이 논문을 구현하거나, 새로운 실험적인 기술들을 구현할 때 더 효율적인 env.라고 말하고 싶다. 물론, 어떤 저수준의 하드웨어를 제어하며 연결하는 작업 등은 할 수 없겠지만, 그런 부분은 c++라이브러리를 쉽게 가져올 수 있으므로 문제는 없다고 생각한다. 기본 베이스들이 통일되어 있기 때문이다. c++의 경우 배열, eigen의 행렬, opencv의 행렬이 모두 다르고, 변환에 꽤 많은 비용이 들지만, pythondml ruddn eoqnqnsdl numpy로 구현되고 있어 더 좋은 통일성을 가졌다고 말하고자 한다.

또한, 추상화의 입장에서도 그렇다. 예전 자동차와 다르게 현대의 자동차는 수많은 전자장비(트랙션 컨트롤, ABS etc.)가 들어간다. 이런 시스템들은 직접 작동하기에는 운전자의 수많은 이해를 요구한다. 그렇기에 직접 작동시킬 수 없는 것이다. (모두가 전문가가 될 수 없다.) 단지 컴퓨터를 테스트베드로서 사용하고자 하는 사람들(기기의 특성에 대해 관심가지고 싶지 않은 사람들)에게는 cuda보다는 자동화된 cupy가 더 효율적이라고 말하고 싶다.

결론

결국 이 프로젝트에서 낼 수 있는 결론은, 연구와 R&D와 개발이라는 것들로 공학자들이 하는 일을 크게 나누면, 3개는 다음과 같이 정리되며, 한 프로젝트는 이 3개 중 하나만 할 수 있다는 것이다. 연구는 원 연구를 바탕으로 새로운 가정 혹은 발견을 추가하여 성능을 높이고, 그것의 happy시나리오를 증명 - 간단한 구현과 실험을 하는 것을 목적으로 해야 한다. R&D는 원 연구를 바탕으로 시연을 하기에 신뢰할 수 있는 소프트웨어적 컴포넌트를 만들어 최소 수준으로 integration하여 통제된 환경에서 쇼케이스 수준의 데모를 하는 것을 목적으로 해야 하는 것이다. 개발은 이미 있는, 신뢰할 수 있는 컴포넌트들을 좋은 설계 행위를 통해 완벽하고 robust하게 integration하는 것을 목적으로 해야 한다. - 아키텍쳐는 신뢰 가능한 컴포넌트들만을 배치한 것을 말한다고 정의할 수 있겠다.

이 프로젝트가 원 계획대로 성공되지 못하고, 일부의 데모를 성공했다는 것에 아쉽다. 그 이유는 세 부분을 모두 수행하려고 했기 때문이지 않을까 라고 생각한다. 나쁘지 않은 R&D 수준의 결과를 내었다고 판단한다.

글을 마친다.