



국민대학교
전자정보통신대학
컴퓨터공학부


캡스톤 디자인 I

종합설계 프로젝트

| | |
|--------|--|
| 프로젝트 명 | OLAF |
| 팀 명 | ELSA (Elaborate Localization System Architects) |
| 문서 제목 | 중간보고서 |

| | |
|---------|-------------|
| Version | 1.5 |
| Date | 2020-APR-20 |

| | |
|-----|------------|
| 팀 원 | 김 다 훈 (조장) |
| | 김 명 수 |
| | 김 선 필 |
| | 배 한 울 |
| | 윤 찬 우 |

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤디자인 수강 학생 중 프로젝트 “OLAF”를 수행하는 팀 “ELSA”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “ELSA”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


| | |
|----------|-------------------------|
| Filename | 중간보고서-OLAF.doc |
| 원안작성자 | 김다훈, 김명수, 김선희, 배한울, 윤찬우 |
| 수정작성자 | 김다훈, 김명수, 김선희, 배한울, 윤찬우 |

| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
|------------|-------|----------|----------|------------------|
| 2020-05-24 | 팀 전체 | 1.0 | 추가 | 중간보고서 초안 작성 |
| 2020-05-25 | 윤찬우 | 1.1 | 추가 및 수정 | 기술 부문 내용 추가 및 수정 |
| 2020-05-26 | 배한울 | 1.2 | 추가 및 수정 | 기술 부문 내용 추가 및 수정 |
| 2020-05-26 | 김명수 | 1.3 | 추가 및 수정 | 기술 부문 내용 추가 및 수정 |
| 2020-05-27 | 김다훈 | 1.4 | 추가 및 수정 | 기술 부문 내용 추가 및 수정 |
| 2020-05-28 | 김선희 | 1.5 | 추가 및 수정 | 기술 부문 내용 추가 및 수정 |
| | | | | |
| | | | | |

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

목 차

| | |
|--------------------------------------|----|
| 1 프로젝트 목표 | 4 |
| 1.1 세부 목표 | 4 |
| 2 수행 내용 및 중간 결과 | 5 |
| 2.1 계획서 상의 연구내용 | 5 |
| 2.2 수행 내용 | 5 |
| 2.2.1 서버와 로봇간 데이터 통신 | 5 |
| 2.2.2 Way-Point 알고리즘 | 6 |
| 2.2.3 센서들의 데이터 통합 | 6 |
| 2.2.4 카메라를 통한 타일 Edge Detection 및 제어 | 7 |
| 3 수정된 연구내용 및 추진 방향 | 10 |
| 3.1 수정 사항 | 10 |
| 3.1.1 직진 보정 | 10 |
| 3.1.2 영상처리를 통한 타일인식 | 12 |
| 3.1.3 카메라 장착 | 13 |
| 3.1.4 서버 | 14 |
| 4 향후 추진계획 | 14 |
| 4.1 향후 계획의 세부 내용 | 14 |
| 4.1.1 LiDAR를 이용한 wall following | 14 |
| 4.1.2 장애물 인식 및 회피기동 | 14 |
| 4.1.3 영상처리를 통한 localization 보정 | 14 |
| 5 고충 및 건의사항 | 15 |

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |


1 프로젝트 목표

OLAF는 교내 학생 및 교내를 방문하는 외부인에게 도움을 주기 위해서 고안된 프로젝트로서 국민대학교 각 건물만의 특화된 안내 시스템을 구축하는 것이 이 프로젝트의 목적이다.

학과 건물과 같은 대규모 실내 공간에서 매년 들어오는 신입생 및 외부인들과 같은 초행자들에게 OLAF의 에스코트와 최단경로 안내를 해주는 서비스를 제공하여 사람들은 보다 쉽게 길을 찾을 수 있다. 이를 통해 OLAF는 노동력과 사람들에게 삶의 편의를 제공할 수 있다.

교내 네비게이션이 구축된 후, 더 나아가 누구나 스마트 폰으로 촬영만으로 가상의 3차원 실내공간과 실내공간에 적용될 게임, 콘텐츠의 제작에도 획기적인 발전이나 새로운 시도가 가능하다. 부가적으로, 실내안의 복지시설 광고나 정보 제공의 새로운 플랫폼으로도 확장 가능하다.

기술적인 측면으로는 SLAM으로 생성된 지도와 IMU, Camera를 통한 정확한 실시간 위치추정, LiDAR를 통한 장애물 인식을 통해 보다 빠른 처리 속도의 최단경로를 계산하는 알고리즘 구현을 목표로 한다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

2 수행 내용 및 중간 결과

2.1 계획서 상의 연구내용


OLAF는 안내하게 될 건물의 지도를 획득하여 현재 위치로부터 목표지점까지의 경로를 생성할 수 있어야 한다.

먼저 LiDAR / IMU / Encoder 값을 활용한 SLAM으로 해당 층을 직접 움직이며 지도를 생성한다. 지도를 측량할 때 IMU 센서를 통해 로봇이 향하는 방향을 판단하고, 2D LiDAR는 0도에서 359도까지 총 360개의 거리 정보를 통해 벽 또는 장애물을 판단할 수 있게 한다. Encoder와 IMU를 통해서 이동속도와 현재 위치 등을 판단한다.

지도 생성이 끝난 다음에 사용자로부터 목적지를 입력받아서 현재 위치로부터 목적지까지의 이동 가능한 경로를 생성한다. 사용자가 입력할 수 있는 목적지는 사전에 정의된 목적지만 가능하며, 경로의 생성은 지도 좌표 상에 존재하는 waypoint를 활용해서 최단거리 알고리즘을 통해 생성하게 된다.

경로가 생성되면 OLAF는 보편적인 사람의 걷기 속도로 목적지까지 이동한다. 이동 중에 발생하는 돌발 장애물에 대하여 조건을 나누어서 사람과 같이 속도가 있는 장애물의 경우 OLAF가 제자리에 멈추어서 사람이 지나갈 때까지 대기한다. 만약 속도가 있는 장애물이 지속적으로 이동을 방해할 경우 소리로 알려서 상황을 벗어날 수 있도록 도움을 요청한다. 속도가 없는 정적 장애물의 경우 회피 기동으로 상황을 벗어난다.

목적지의 도착은 지도에서의 위치와 카메라로부터의 입력을 통해 종합적인 상황으로 판단한다. 먼저 지도에서는 목적지 부근의 좌표 범위 설정으로 해당 범위 안에 OLAF가 도착했을 경우에 도착으로 판단할 수 있다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

2.2 수행 내용

2.2.1 사용자와 로봇간 데이터 통신


사용자와 로봇간 통신은 웹을 통해서 하게 된다. 사용자는 웹 페이지에서 목표 장소를 선택하면 서버에서 현재 상태값 및 목적 장소를 json 데이터로 저장한다. 로봇에서는 매초 서버에서 해당 데이터를 받아오면서 운행에 필요한 데이터를 받게 되면 해당 목적 장소로 운행 후 서버에 운행을 완료했다는 상태값을 json 데이터로 저장한다. 서버에서는 운행중일 때 매 초 해당 데이터를 확인하여 운행이 완료 됐다는 상태값을 받아오기 전까지 다른 목적지의 입력을 받지 않으며 알림창으로 표기를 해준다.

2.2.2 Way-Point 알고리즘

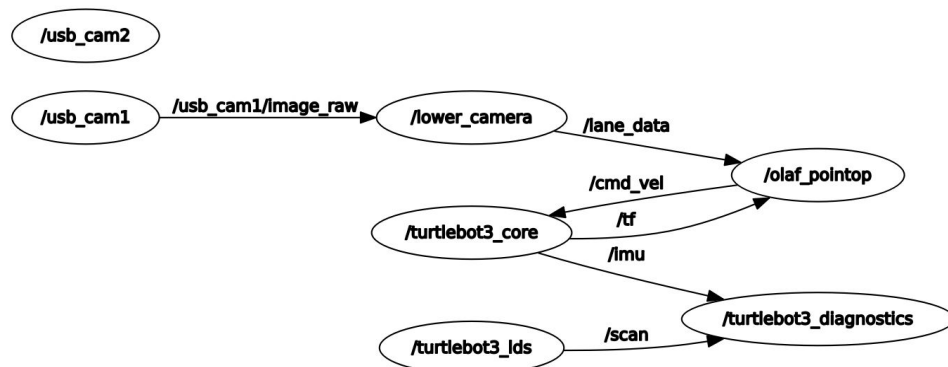
먼저 맵에 강의실, 교차로, 특징점마다 포인트들을 생성한다. 각각의 포인트는 포인트끼리 구별할 수 있도록 인덱스를 가지고 포인트가 어느 지점을 나타내는지 이름을 가진다. 포인트들은 2차원으로 x,y좌표를 가지고 있고 해당 포인트가 교차로 부분인지 아닌지 판단할 수 있도록 하였다. 서버로부터 목적지좌표를 받으면 현재 위치에서 목적지까지 갈 수 있는 알고리즘을 구현했다. 시작지점 (0,0)을 기준으로 서버로부터 목적지 좌표를 받고 목적지 좌표와 시작 좌표를 사용하여 맵에 지정해 둔 포인트들 중 최단경로로 목적지까지 갈 수 있는 포인트들을 배열에 저장한다. 저장한 포인트들의 좌표를 순차적으로 로봇에게 전달하여 로봇이 직각주행만 할 수 있도록 한다. 시작좌표와 포인트좌표의 y좌표 차이를 이용하여 로봇이 90도회전, -90도회전을 판단하고 x좌표값의 차이를 이용하여 로봇이 직진 할 것인지 후진 할 것인지 판단한다. 로봇이 포인트들을 지나 최종 목표지점에 도달하면 길 안내를 종료하고 다시 시작지점으로 돌아 올 수 있도록 하였다.


2.2.3 센서들의 데이터 통합

사용하는 센서는 현재 LIDAR, 엔코더, 카메라, IMU를 사용하고 있다. 현재 사용하기 있는 OPENCN Board의 경우 ROBOTIS사에서 제작한 펌웨어가 존재하지만, 대부분 ROBOTIS 사의 로봇에 맞게 제작된 펌웨어이기 때문에 저희 차량에 맞게 Motor Driver를 직접 수정을 했다. 기존에 제공된 Motor Driver 상에서 추가로 직진 보정을 하기위해 엔코더에서 속도 값을 Read하는 함수를 제작하고, Motor Control 함수 내에서 Motor

|  <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div> | 중간보고서 | | |
|---|-------------------------|-------------|-------------|
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

Control Input에 비례하는 Threshold 값에 도달하도록 P control을 통해 입력을 수정하였습니다. 따라서 따로 코드에서 통합하는게 아니라 어느 코드를 돌리든지 직진 주행이 가능하게 했다. IMU 센서값은 로봇이 주행하거나 회전을 할 때 사용한다. 엔코더 값으로 직진 주행을 수정하였지만 방향이 조금이라도 틀어질 시 심각한 오류가 발생하기 때문에 IMU 센서값을 활용해 더욱 정밀하게 목표지점에 도달 할 수 있도록 하였다. 또한 회전을 할 시 90도 180도 -90도 회전을 하는데 로봇이 정확히 회전을 하지 못하더라도 주행하면서 각을 맞추수 있도록 하였다. 마지막으로 카메라를 활용하여 최종적으로 IMU, 엔코더로 잡지 못한 부분을 영상처리를 통해 수정하도록 하였다. 카메라에서 주는 값은 항상 사용하는것이 아니라 보조적인 기능으로 회전을 한 후, 직진 중으로 나누어 로봇이 심각하게 틀어지거나 경로를 이탈할 경우 보정해 줄 수 있는 보조적인 역할을 한다. 최종 시연때까지 LiDAR 센서도 통합 할 예정이다.

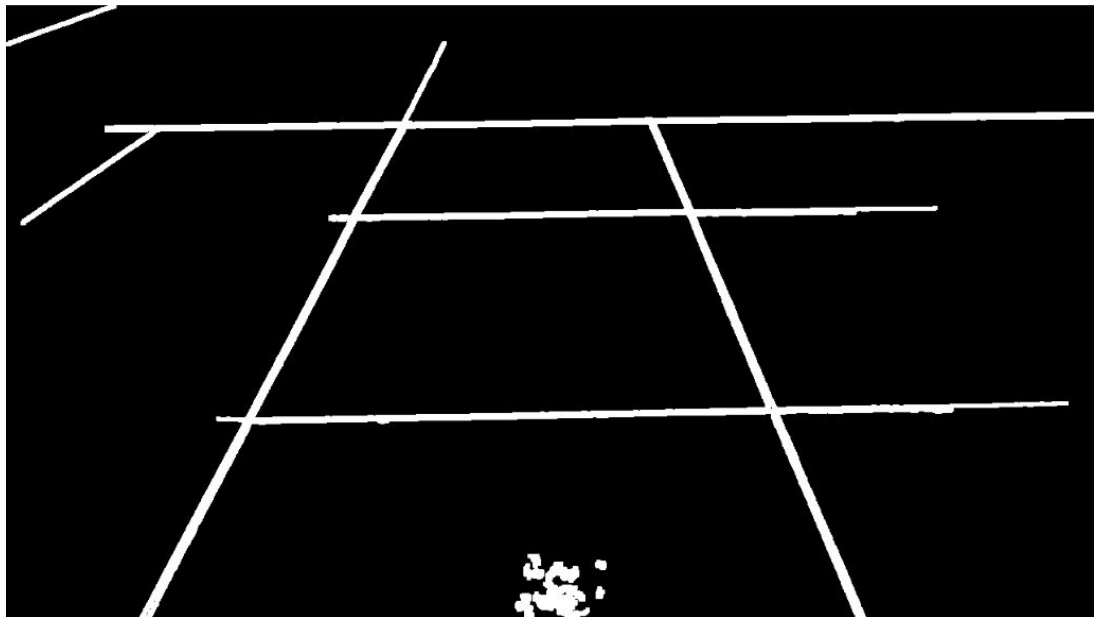


| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

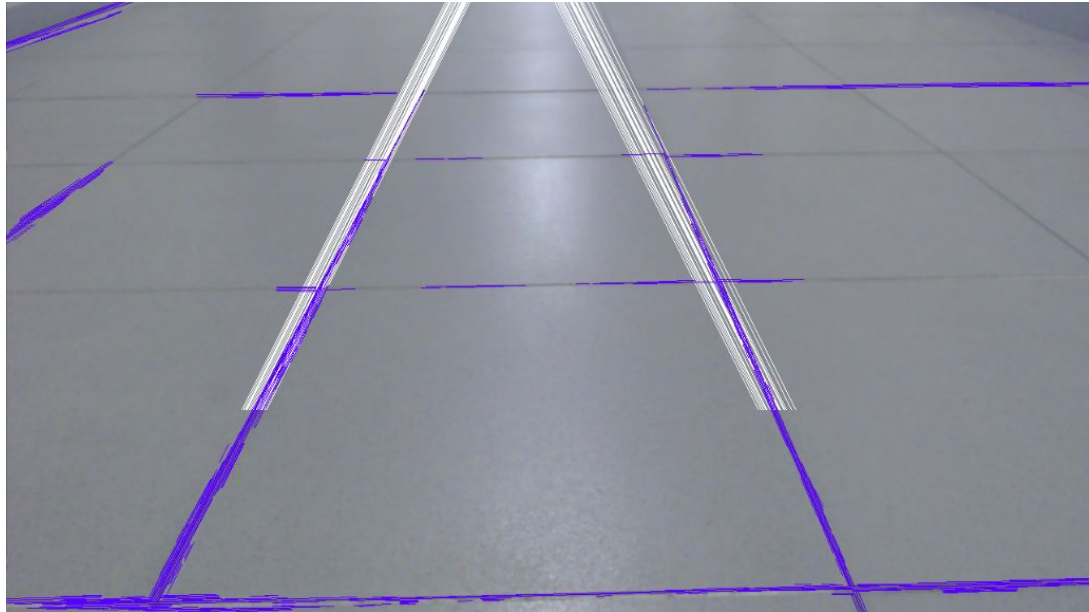
2.2.4 카메라를 통한 타일 Edge Detection 및 제어

(VER 1)

1차 중간발표의 알고리즘을 토대로 하여 더욱 안정적으로 Lane을 추출하는 알고리즘을 채택하였다. 하지만 여전히 빛의 영향에 따른 노이즈가 문제였다. 노이즈를 제거하기 위해 Gaussian Filter와 HSV색공간의 동적 인자와 Canny Edge함수의 동적인자를 사용하였고 추가적으로 히스토그램으로 후보가 되기 위한 임계값을 주어서 빛이 아닌 직선만 추출하였다. Hough Transform 알고리즘을 통해 직선들의 좌표와 기울기를 통해 수평에 가까운 기울기를 배제 하고 차선을 구해 가야할 방향의 직선을 구하였다.



[그림 1] 전처리에서 빛에 따른 노이즈 Edge(해상도 1280 X 720)




[그림 2] 후처리에서 빛에 따른 노이즈 제거 (해상도 1280 X 720)

위의 [그림 1]과 같이 빛에 의한 Edge를 [그림 2]와 같이 제거하였다. 연산량은 보다 많았지만, 더욱 안정적으로 차선을 추출하였다.

(VER 2)

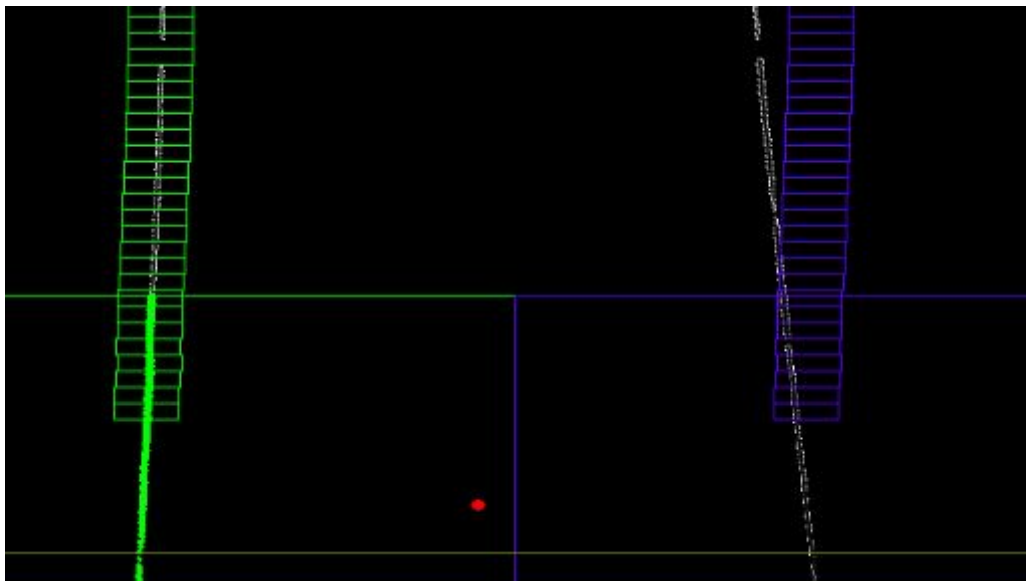
하지만 Nvidia Jetson TX2 Board에서의 환경에서 많은 연산에 따른 Frame수 저하(8~10)로 인한 제약이 있음을 인지하였다. 이러한 문제는 실시간으로 차량이 주행하기에 중요한 이슈였고, 연산이 적으면서 빛의 영향에 덜 받을 수 있는 새로운 알고리즘을 도입할 수 있게 연구를 하였다.

새로운 알고리즘의 이미지 전처리 부분으로 연산량을 줄이기 위하여 RGB 이미지를 Grayscale하여 3차원의 채널을 1차원으로 합친 후, Perspective Transform을 하여 연산하는 이미지의 크기를 1280X720에서 640X360 의 크기로 변환하였다. 타일 인식의 주 목표는 차량이 직진시 특정 타일상에서 주행하는 목표를 세웠기 때문에, 현재 차량 기준으로 수직인 Edge를 검출하는 것이 좋겠다는 판단하에 Sobel Edge의 X filter를 이용하여 수직에 대한 Edge를 검출하였습니다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |




[그림 3] Gaussian Blur와 Sobel Edge, Histogram 분석을 통한 유동적으로 노이즈 제거 및 Perspective Transform 을 이용한 연산량 감소 (해상도 640 X 360)




[그림 4] Sliding Window에 따른 Lane Detection 및 제어점 추출 (해상도 640 X 360)

[그림 3]에서 Gaussian Filtering를 통해 노이즈를 제거하고, Sobel Edge를 이용하여 수직인 직선을 추출하였다. 매 Frame마다 Histogram 분석을 통해 유동적인 후보군들은 통하여 통계적 이산점들을 제거하였다. [그림 4]에서 앞에서 구한 히스토그램 후보군을

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

직사각형의 ROI 영역에서 예지로 추정되는 좌표값을 얻어냈다. 위의 과정을 통해 얻어진 좌표값들을 Polyfitting 하여 직선의 기울기 및 Control Plane 상의 Control Point를 구하였다.

Ver1 알고리즘에 비해 영상에서 특정 부분을 분석하여 Control Point를 추출하기 때문에, 정확성 측면에서는 Control Point의 분포가 진동하는 현상이 종종 발견되었다. 이러한 문제를 해결하기 위해 Control Point에 대한 Filtering이 필요하다고 판단되어, Low pass filter를 도입하였고, 보다 안정적인 Control Point를 제공할 수 있게 되었다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |


3 수정된 연구내용 및 추진 방향

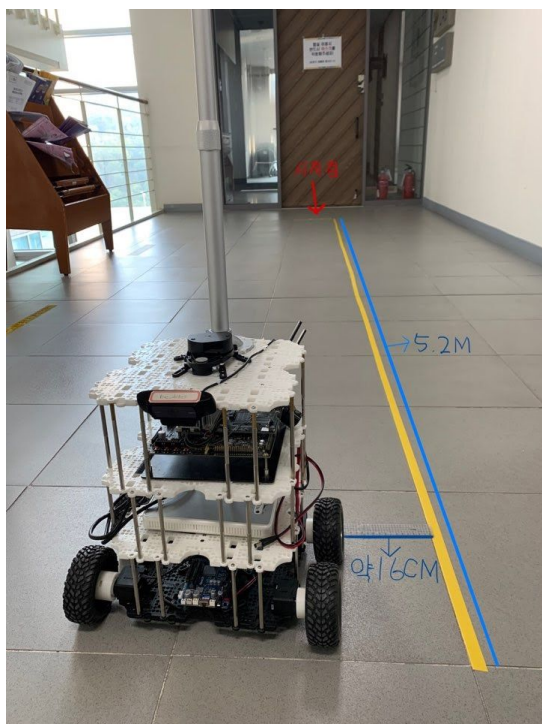
3.1 수정 사항

3.1.1 직진 보정

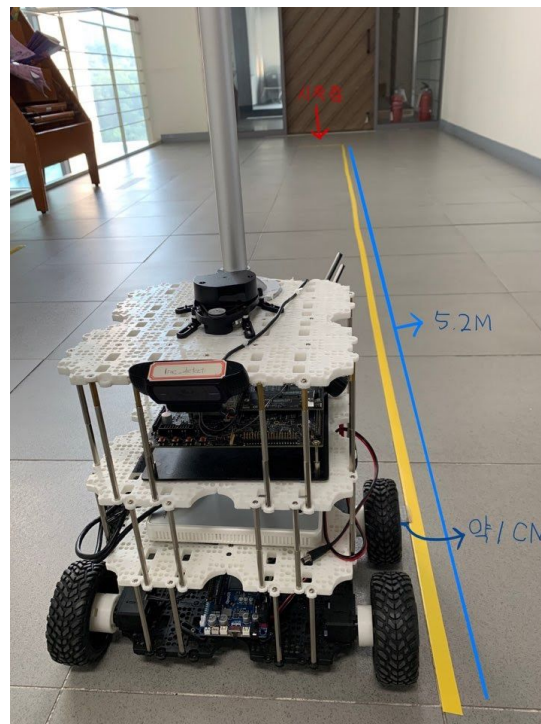
기존에 직진보정을 하는 방법은 OpenCR이 Dynamixel의 엔코더값을 읽고 엔코더 값을 master node로 publish 한다. 전달된 메시지를 Nvidia Jetson TX2 Board의 master node에서 subscribe하고 master node에서 조향값을 OpenCR Board에 시리얼 통신으로 전달한 뒤, OpenCR Board에서 subscribe한 조향값을 가지고 Dynamixel Motor에 명령을 내리는 방식이다. 이와 같은 보정방법은 엔코더로부터 양쪽 모터의 속도를 읽어 차이값을 계산하고 차이값을 스케일링하여 조향값으로 제어하는 방법이다.

이 방법의 문제점은 조향값을 덮어 씌우는 방식이기 때문에 회전 조향값을 주어도 직진하는 문제가 발생하였다. 또한 모터를 개별 제어하지 못하기 때문에 보정 방식이 한정적이었다. 마지막으로 불필요하게 거쳐야 하는 소프트웨어 구조들이 있었기 때문에 피드백 시간이 긴 단점이 있었다. 이를 해결하기 위하여 ROS 메시지 통신을 거치지 않고 OpenCR 펌웨어 상에서 직진 보정이 가능하도록 수정했다. 수정 방법으로는 시리얼 통신을 통해 OpenCR Board에게 Dynamixel Motor의 제어값이 들어오면 각 제어값에 맞는 Threshold를 설정하여 Dynamixel에서 읽은 엔코더값과 Threshold 값의 차이를 계산하여 계산된 차이만큼 새로 각각의 Dynamixel에게 제어값을 주는 방식으로 수정하였다. 이러한 방법의 장점으로 각각의 모터를 개별 보정이 가능해 졌고 설정한 Threshold에 가까워 지도록 P 제어를 하기 때문에 직진주행 뿐만 아니라 회전주행까지 보정이 가능해 졌다. 마지막으로 불필요한 소프트웨어 스택을 거치지 않게 되었다. 수정된 방법을 통해 직진주행의 성능을 테스트한 결과 기존의 방식은 5.2m 주행시 약 16cm의 오차가 발생하였지만 현재 방식으로 5.2m 주행시 약 1cm의 오차가 발생하는것으로 확인했다. 테스트 중에는 엔코더 센서만을 사용하였다.


| | | | |
|---|-------------------------|-------------|-------------|
|  <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div> | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |



[그림1] 기존의 보정방식

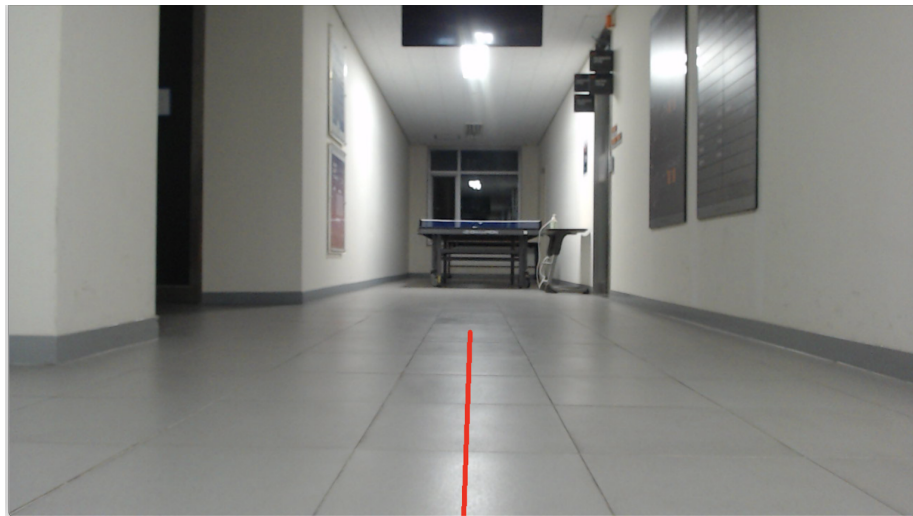


[그림2] 새로운 보정방식

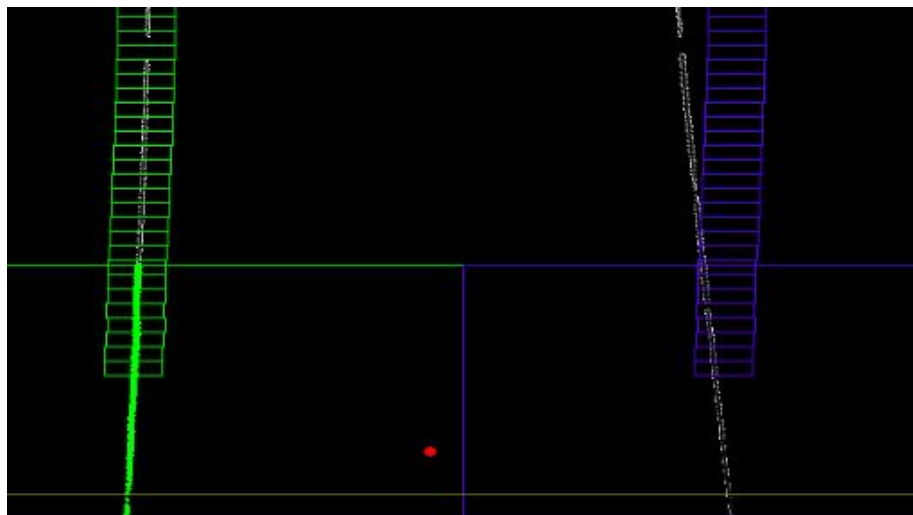
| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |


3.1.2 영상처리를 통한 타일인식

2.2.4의 ver 1 알고리즘의 경우 안정적으로 차선을 추출하나 많은 연산으로 인하여 frame 수가 저하되는 문제가 있었다. 따라서 정확도와 시간적인 Trade Off 현상을 느끼고, 정확성과 실시간성 사이에 적당한 기준을 두어 2.2.4의 ver 2 알고리즘을 새로 고안하였다. 새로운 알고리즘은 Perspective 변환 및 Sliding Window 알고리즘을 도입하여 일정한 ROI(Region Of Interest)들을 이어 연산량을 40배가량 줄였으며, 또한 비교적 안정적으로 차선을 추출하게 되었다.



[그림 7] 기존의 고해상도에서의 Lane Detection



| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |


[그림 8] 새로운 알고리즘을 도입한 Lane Detection

3.1.3 카메라 장착

기존에 하드웨어에 카메라 1대를 추가로 장착하였다. 추가된 카메라는 방번호 인식, 특징점 추출을 통해 Localization에 힌트를 주는 방법으로 사용된다.




[그림] olaf 완성본

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

3.1.4 서버

웹 / 앱 페이지를 이용하여 사용자와 로봇간 데이터 통신을 생각하고 있었다. 하지만 로봇이 사용자를 목적지 까지 데려다주기 때문에 굳이 앱으로 만들 필요없이 로봇에 해당 웹페이지로 이동하는 무언가 장치를 뒤서 사용자가 웹페이지로 이동 후 로봇에게 목적지를 입력하는 방식으로 바꿨다. 이 방법으로는 간단하게 QR 코드를 이용하여 사용자가 로봇에서 QR 코드를 이용하여 웹페이지로 이동하는 방식으로 수정하였다.

로봇과 서버(웹) 간에 데이터 통신에서 json 데이터 파일을 이용하여 데이터 통신을 하려고 했다. 이때 json-server 를 이용하여 json 파일을 새롭게 서버로 띄우고 api 통신을 하여 json 파일을 읽고 쓰는 방식으로 변경하였다. 이 경우 로봇과 서버간 네트워크 통신 상태에 따른 데이터 통신이 원활한지에 대한 문제 해결과 1차 중간보고 피드백 답변을 위해 로봇에서의 데이터 통신 테스트를 진행하였고, 미래관 내부에서 로봇과 서버간 데이터 통신은 이상이 없다는 결과를 얻었다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

4 향후 추진계획

4.1 향후 계획의 세부 내용

4.1.1 LiDAR를 이용한 wall following


LiDAR를 통해 얻는 데이터 정보를 활용하여 wall-following 알고리즘을 구현한다. Wall-following 알고리즘이 필요한 이유는 로봇이 예상치 못하게 벽에 붙어서 가는 경우 벽과의 충돌을 방지하고자 LiDAR 데이터를 사용하여 벽과 일정거리를 유지 할 수 있게 하기 위함이다. 현재 7호관 내부환경이 계단, 난간, 유리벽 등 LiDAR 데이터를 적절히 사용하기 힘든 환경이지만 판단할 수 있는 벽의 위치는 대부분 존재하기 때문에 한쪽 벽을 인식하고 벽과의 간격을 유지 하게 한다.

4.1.2 장애물 인식 및 회피기동


LiDAR 데이터 정보를 활용하여 장애물을 인식하도록 한다. Wall-following에서는 측면 각 90도를 활용하고 장애물 인식은 정면각 30도를 활용한다. 로봇 정면에 장애물이 지나가 LiDAR에 잡히게 되면 일단 정지 후, LiDAR의 데이터 값으로 3초 이상 장애물을 인지 할 경우 정적 장애물로 판단하여 회피기동을 실행한다. 3초 이내에 장애물이 없어진다면 동적 장애물로 인식하고 회피기동을 하지 않고 다시 출발하도록 한다.

4.1.3 영상처리를 통한 localization 보정

현재 실내환경이 LiDAR 센서를 활용하여 Localization하기에는 다소 무리가 있는 환경이다. 라이다 센서가 인지하지 못하는 계단, 난간 등이 있고 유리로 된 벽은 LiDAR센서 빛이 투과하여 오차가 발생한다. 현재 Localization 부분에 있어서 LiDAR, 엔코더값과 IMU센서 값에 지나치게 의존하는 부분이 있다. 실외인 경우 GPS를 통해 로봇의 위치를 절대좌표화 할 수 있지만, 실내에서는 GPS를 사용하지 못하므로 맵에서 절대적인 위치를 추정하기 어렵다. 따라서 이를 해결하기 위하여 영상처리를 통해 로봇이 현재 어디를 지나고 있는지 확인을 할 수 있는 방법호, 정수기, 자판기 등을 활용하여 현재 로봇이 어디를 지나고 있는지 정보를 얻게 한다. 예를들어 현재 로봇이 사람이라고 생각을 하면 지금 현재 로봇은 눈을 감고 균형 감각과 걸음걸이 만을 계산하여 이동하고 있는

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

셈이다. 목적지까지 제대로 걸어가기 위해서는 중간중간 눈을 떠서 현재 어디를 지나고 있는지 파악해야한다. 이것을 다시 로봇에게 적용하면 균형감각은 IMU, 걸음걸이는 엔코더에 해당한다. 영상처리를 통해 현재 로봇이 현재 어느 지점을 지나는지 파악할 수 있도록 구현한다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 컴퓨터공학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | OLAF | |
| | 팀 명 | ELSA | |
| | Confidential Restricted | Version 1.5 | 2020-May-28 |

5 고충 및 건의사항

현재 코로나19 바이러스로 인해 공간제약이 있습니다. 저희의 프로젝트는 미래관(구 7호관) 네비게이션 로봇입니다. 즉, 구동체의 하드웨어와 소프트웨어 둘 다 설계하고 구현해야 합니다. 하드웨어적인 부분에서는 직접 구동체를 설계해 두었지만, 소프트웨어적인 부분에서는 직진주행을 위한 영상처리 및 하드웨어 제어, 데이터셋이 필요합니다. 현재 코로나19가 빨리 마무리 되어졌으면 하는 바람이 있지만 언제 안정화가 될지 아무도 모르는 것이기 때문에 하드웨어랑 소프트웨어를 가진 프로젝트인 저희에게는 소프트웨어를 구현하여도 설계한 하드웨어를 국민대에서 구동을 해야 하기 때문에 불편함이 있습니다. 소프트웨어를 구현하면 하드웨어와 함께 테스트를 해야 확인이 가능하고 현재는 미래관(7호관)이라는 장소에서 구현하는 것을 목표로 두어 팀원끼리 온라인으로만 진행하기에는 어려움이 있으며 많은 제약이 따릅니다. 그렇기에 미래관 전체가 아닌 교내에 팀원들과 프로젝트를 진행할 수 있는 장소와 공간을 마련하거나 대처방안이 필요하다고 생각합니다.