보고서

보고서 및 논문 윤리 서약

1. 나는 보고서 및 논문의 내용을 조작하지 않겠습니다.
2. 나는 다른 사람의 보고서 및 논문의 내용을 내 것처럼 무단으로 복사하지 않겠습니다.
3. 나는 다른 사람의 보고서 및 논문의 내용을 참고하거나 인용할 시 참고 및 인용 형식을 갖추고 출처를 반드시 밝히겠습니다.
4. 나는 보고서 및 논문을 대신하여 작성하도록 청탁하지도 청탁받지도 않겠습니다.

나는 보고서 및 논문 작성 시 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.

보고서명 : 캡스톤디자인2 3차보고서

학　　과 : 전기공학과
과　　목 : 캡스톤디자인2
담당교수 : 이범주
마 감 일 : 2023.12.06
제 출 일 : 2023.12.06

팀　　명 : 캡스톤디자인2 7조

팀원1	학번 :	60205095	이름 :	강승원	(서명)
팀원2	학번 :	60195101	이름 :	장원준	(서명)
팀원3	학번 :	60191782	이름 :	임홍균	(서명)
팀원4	학번 :	60205098	이름 :	김민석	(서명)

제안상품명: Shortest path following vehicle

발주자: 명지대학교 전기공학과 이범주 교수
보고기관: 명지대학교 전기공학과
설계팀: 강승원: 명지대학교 소속
경기도 성남시 분당구 장미로 55, 010-3642-7438
장원준: 명지대학교 소속
서울특별시 송파구 마천동 344-5, 010-3734-8415
임홍균: 명지대학교 소속
서울특별시 강북구 도봉로 18길 48, 010-6456-7011
김민석: 명지대학교 소속
경기도 화성시 동탄대로 469-12, 010-9054-1670

작성일자: 2023년 12월 03일
문서버전: Ver 1

목 차

요약문

캡스톤디자인2 7조는 이범주 교수님과의 미팅을 거쳐 트랙에서 출발 위치와 도착 위치가 결정되면 최단 경로로 출발 위치에서 도착 위치까지 이동하는 차량인 '최단 경로 추종 차량'을 제작하기로 하였다. 현실적 제약조건은 야외의 지형을 고려해서 제품을 제작하기가 어려워 제품이 정해진 트랙에서 이동하고 제품이 장애물을 스캔하여 지도를 생성하는 과정에서 라이다 대신 카메라를 사용하는 것이다. 제품의 요구사항은 차량이 출발 위치에서 도착 위치로 이동하는지, 이동할 때 최단 경로로 이동하는지, 트랙의 길이 변화하였을 때 다시 경로를 생성하는 데 걸리는 시간이 최소인지이다. 이 요구사항들을 제품의 제작 목적을 세우기 위해 구체적으로 해석하면 트랙 상의 좌표, 출발 위치와 도착 위치 사이의 좌표들을 지정하는 알고리즘, 지정된 좌표들을 차량이 오차 없이 추종, 알고리즘 소스 코드의 최적화가 필요하다는 것으로 해석할 수 있다. 따라서 앞의 사항들을 만족하기 위해서 제품에 OPENCV 좌표계, A* 알고리즘, Pure Pursuit 알고리즘 이론 등을 적용한다.

Abstract

After a meeting with Professor Bumjoo Lee, Group 7 of Capstone Design 1 decided to produce a 'Shortest path following vehicle', a vehicle that moves from start to arrival in the shortest route when the start and arrival points are determined on the track. The realistic constraint is that it is difficult to manufacture a product considering the outdoor terrain, so the product moves on a set track and the product scans obstacles to create a map, using a camera instead of a lidar. The product's requirements are whether the vehicle moves from the starting point to the arriving point, moves the shortest route, and whether the time taken to generate the path again is the minimum when the track path changes. To set the purpose of production, these requirements can be interpreted to need the coordinates on the track, the algorithm that specifies the coordinates between the starting point and the arrival point, and the vehicle follows the specified coordinates without error, and the optimization of algorithm source code. Therefore, OPENCV coordinate system, A* algorithm, and Pure Pursuit algorithm theory are applied to the product to satisfy the above.

감사의 글

이 프로젝트에 기여해 주신 강승원, 임홍균, 장원준, 김민석 조원과 이범주 교수님께 감사의 말을 전합니다.

1. 서론

캡스톤디자인2 7조 강승원, 장원준, 임홍균, 김민석 4명은 사회적 약자 중에 가장 문제가 심각해 보이는 시각 장애인에 대한 제품인 '시각 장애인을 위한 웨어러블 기기(Wearable devices for the blind)' 를 제작하려고 했었다.

그림 1: 사용자를 따라다니는 캐리어

이후 이범주 교수님과의 1차 미팅에서 피드백의 결과로 주제를 변경하기로 하였다. 그래서 조원들끼리 프로젝트 주제에 대한 브레인스토밍을 통해서 시각 장애인을 위한 웨어러블 기기 대신 사용자가 원하는 위치를 설정하면 해당 위치로 장애물을 회피하고 사용자와 거리를 유지하며 길을 안내하는 로봇을 제작하기로 하였다. 그러나 2차 미팅에서 교수님께서 해당 주제가 너무 어려운 주제라고 피드백을 주셨다. 이를 고려해서 제작하기로 결정한 제품은 다음과 같았다. 로봇을 차량 형태로 제작하여 차량이 트랙 위에서 장애물을 회피하며 정해진 경로를 추종하는 라인 트레이서를 제작한다. 그러나 3차 미팅 이전 중간 미팅에서 교수님이 4명이라는 인원에 비해 라인 트레이서는 제작하기 쉽다는 것을 지적하셨다. 조원들과 새롭게 토의한 결과, 라인 트레이서 트랙이 아닌 도로형 트랙에서 출발 위치와 도착 위치가 결정되면 최단 경로로 출발 위치에서 도착 위치까지 이동하는 일종의 자율주행 차량으로 주제를 결정했다.

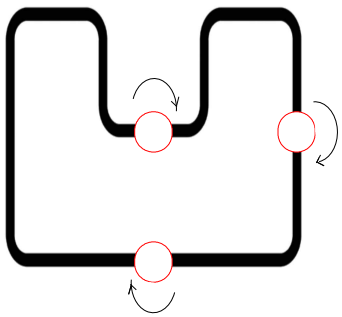


그림 2: 라인 트레이서 트랙

3차 미팅에서 캡스톤디자인2 7조는 앞에서 서술한 제품인 '최단 경로 추종 차량(Shortest path following vehicle)' 을 제작하는 것으로 결정하였다. 그 이후 각 조원 간의 역할 분담을 했는데 그 내용은 다음과 같다. 강승원 조원은 A* 알고리즘 모의실험 및 planning, 임홍균 조원은 노트북을 이용한 A* 알고리즘 동작과 아두이노 차량으로의 데이터 전송, 장원준 조원은 OPENCV 및 SLAM 담당, 김민석 조원은 차량의 속도 및 위치제어와 경로 추종 담당으로 결정되었다.

본 보고서의 구성은 다음과 같다.

2절에서는 현실적 제한조건인 제품이 야외에서 이동하지 않는 이유, 라즈베리파이와 라이다를 사용하지 않는 이유, 모터의 성능 문제에 관해 서술한다.

3절에서는 제품에 대한 사용자의 요구사항을 서술하고 이를 제품의 제작 목적을 세우기 위해 구체적으로 해석한 내용을 서술한다.

4절에서는 제품에 대한 최종 내용을 요약하여 서술한다. 5절에서는 개별요소의 이론적 설명과 시행착오 및 해결과정을 서술한다. 6절에서는 실제 작품의 구동을 확인하여 설계 목적에 부합했는지 평가하고 서술한다. 7절에서는 완성된 작품을 기반으로 프로젝트 결과의 적절성을 평가하고 서술한다.

2. 설계/프로젝트의 현실적 제한조건

최단 경로 추종 차량은 정해진 트랙에서 이동하고 야외에서 이동하지 않는다. 그리고 차량의 현재 좌표, 현재 각도를 계산하는 OPENCV와 현재 좌표로부터 도착 좌표까지의 최단 경로 좌표를 계산하는 A* 알고리즘을 라즈베리파이가 연산하지 않으며 장애물을 스캔하여 지도를 생성하는 과정에서 라이다를 사용하지 않는다. 또한, 모터의 성능 문제도 존재한다. 이에 관해서 서술한다.

최단 경로를 생성하기 위해서는 야외 지도상의 좌표가 있어야 하며 이는 네이버 지도 API, 카카오 지도 API 등을 통해서 좌표 (위도와 경도)를 얻을 수 있다. 하지만 좌표로는 지도상의 지형을 알 수 없다. 지도상의 좌표로는 최단 경로에 산 같은 장애물이 존재할 수 있지만 지형을 고려한다면 산을 돌아가는 경로가 최단 경로일 가능성이 크다.

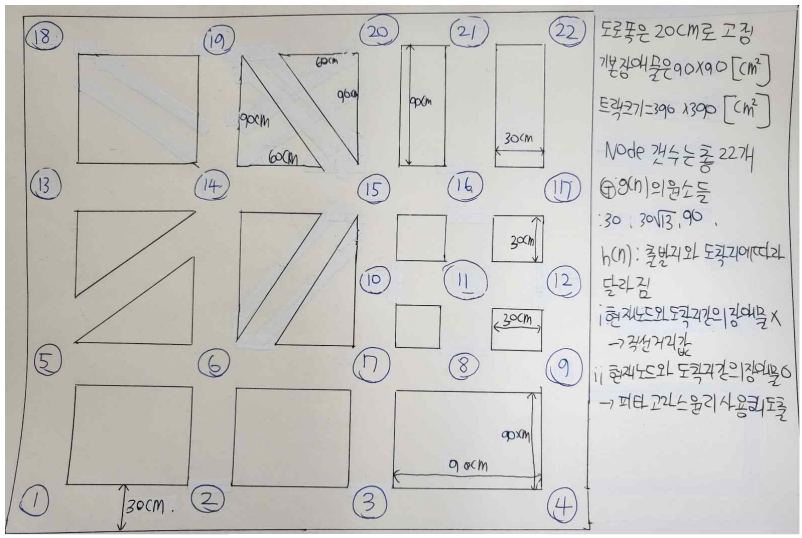


그림 3: 최단 경로 추종 트랙

따라서 캡스톤디자인2 7조는 정해진 트랙에서 이동하는 최단 경로 추종 차량을 제작하기로 하였다. 트랙의 크기는 가변이며, 트랙의 길은 평지에 놓여 있다. 트랙의 예시는 위 그림과 같으며 장애물의 위치는 변경될 수 있다.

차량이 최단 경로 좌표를 추종하기 위해서는 OPENCV를 통해 계산한 차량의 현재 좌표, 현재 각도를 라즈베리파이에서 가능한 빠른 속도로 연산하고 시리얼 통신을 통해 아두이노로 송신해야 한다. 하지만 코드를 수정하였음에도 불구하고 차량이 최단 경로 좌표를 크게 벗어나는 것을 보아 차량이 최단 경로 좌표를 추종하기에는 라즈베리파이의 연산 속도가 충분하지 않다는 결론이 나왔다.

따라서 캡스톤디자인2 7조는 차량의 현재 좌표, 현재 각도를 계산하는 OPENCV와 현재 좌표로부터 도착 좌표까지의 최단 경로 좌표를 계산하는 A* 알고리즘을 연산하는 과정에서 라즈베리파이 대신 노트북을 사용한다.

원래는 라이다를 이용해서 SLAM (Simultaneous Localization And Mapping, 동시적 위치추정 및 지도 생성)을 하려 했었다. 라이다가 장애물을 스캔하면서 지도를 생성하는 과정에서 외부 센서의 도움 없이 라이다가 장애물을 스캔한 정보를 이용해 라이다의 현재 위치도 추정하는 것이다. SLAM에 대해서 학습하는 과정에서 SLAM이 ROS 같은 프로그램의 도움이 없이는 구현하기가 매우 힘들다는 것을 알게 되었다.

그래서 라이다를 이용해서 SLAM의 M (Mapping)만을 하려 했었다. SL (Simultaneous Localization)은 외부 센서인 카메라를 이용해서 라이다의 현재 위치를 측정하는 방식으로 해결할 수 있기 때문이다. 하지만 이 카메라를 이용한다면 사실 라이다가 없어도 지도를 생성할 수 있다. 카메라를 이용하면 OPENCV를 사용하여 장애물을 스캔할 수 있기 때문이다.

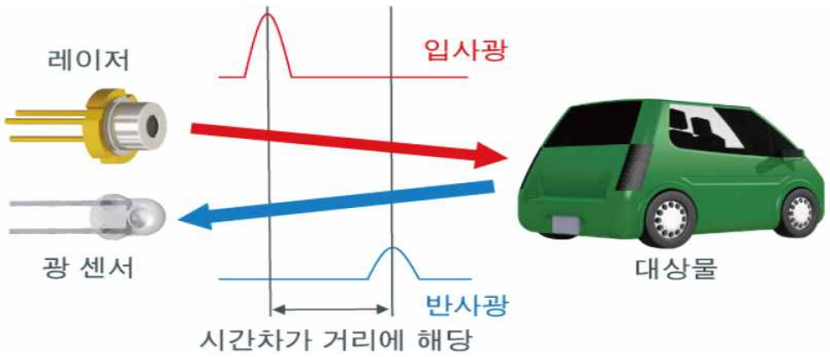


그림 4: 라이다의 원리[1)]

장애물을 스캔하여 지도를 생성하는 것은 비교적 라이다보다 카메라가 쉽다. 카메라를 정해진 트랙 위에 설치하고 OPENCV를 사용해서 장애물을 스캔한다면 장애물 좌표를 계산할 수 있다. 라이다는 장애물에 레이저를 조사하고 반사되는 레이저를 이용해서 장애물 좌표를 계산한다. 하지만 라이다는 장애물에 레이저를 조사해야 하므로 공기 중의 먼지 등도 장애물 좌표로 계산될 수 있다. 그리고 라이다의 현재 좌표도 외부 센서를 통해 알려 줘야 한다.

따라서 캡스톤디자인2 7조는 최단 경로 추종 차량이 장애물을 스캔하여 지도를 생성하는 과정에서 라이다 대신 카메라를 사용한다.

또한, 차량에 사용되는 모터는 저가의 모터이고 판매처에 자세한 모터의 파라미터들도 기재되어 있지 않기 때문에 모터 동작의 신뢰도가 떨어진다는 제한조건이 있다. 그리고 모터에 엔코더가 있지 않아 속도의 피드백 제어가 불가능해 모터의 속도 제어를 오롯이 OPENCV를 통해서 해야 한다는 제한조건도 존재한다.

3. 설계 목적

현실적 제한조건을 생각하였을 때, 캡스톤디자인2 7조 조원들이 판단한 최단 경로 추종 차량의 요구사항은 다음과 같다.

첫 번째 요구사항은 차량이 출발 위치에서 도착 위치로 이동하는지이다. 이를 만족하지 않는다면 최단 경로 추종 차량이 아니므로 이는 반드시 달성해야 할 중요한 요구사항이다. 두 번째 요구사항은 최단 경로 추종 차량이므로 차량이 이동할 때 최단 경로로 이동하는지이다. 세 번째 요구 사항은 트랙의 길이 변화하였을 때 다시 경로를 생성하는 데 걸리는 시간이 최소인지이다. 이 요구사항들을 만족하지 않는다면 차량이 길 안내를 하더라도 도착 위치까지 가는 데 낭비되는 시간이 생긴다.

앞에서 서술한 3가지 요구사항들을 제품의 제작 목적을 세우기 위해 좀 더 구체적으로 해석할 필요성이 있다.

---

1) 로옴 주식회사 - ROHM Semiconductor, “LiDAR (라이다)”, https://www.rohm.co.kr/electronics-basics/laser-diodes/ld_what10.

첫 번째 요구사항을 만족하기 위해서는 트랙 상의 좌표가 필요하다. 출발 위치의 좌표, 도착 위치의 좌표, 출발 위치와 도착 위치 사이의 좌표들이 지정된다면 차량이 이 좌표들을 추종하게 하여 출발 위치에서 도착 위치로 이동할 수 있다. 또한, 주어진 좌표들을 정확하게 따라가도록 하는 경로 추종 알고리즘 또한 필요할 것이다.

두 번째 요구사항을 만족하기 위해서는 출발 위치와 도착 위치 사이의 좌표들을 지정하는 알고리즘이 필요하다. 출발 좌표에서 도착 좌표까지 가는 최단 경로를 나타내는 그래프 탐색 알고리즘인 A* 알고리즘을 사용한다. 또한, 이 요구사항들을 만족하기 위해서는 지정된 좌표들을 차량이 오차 없이 추종하게 하는 것도 필요하다.

세 번째 요구 사항을 만족하기 위해서는 A* 알고리즘 소스 코드의 최적화가 필요하다.

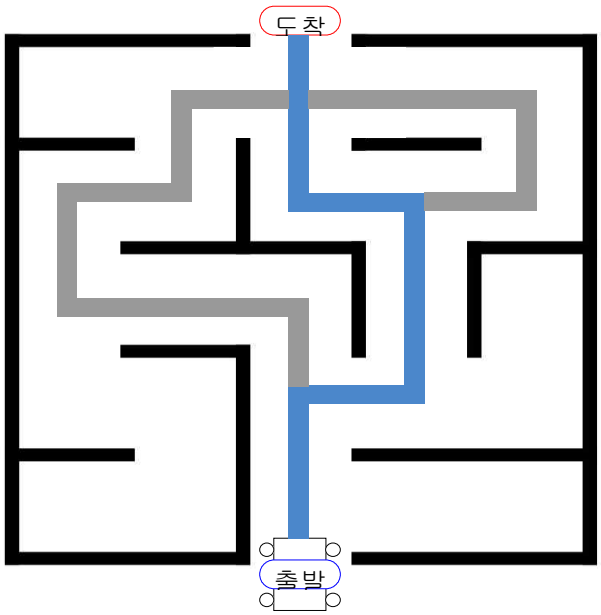


그림 5: 자율주행 차량 동작 그림

그림 5는 앞의 요구사항들과 제한조건들을 적용한 최단 경로 추종 차량의 간단한 동작 그림이다. A* 알고리즘을 기반으로 도착지부터 목적지까지 최적 경로를 탐색하여 이동하는 차량이다. A* 알고리즘 동작에 필요한 좌표와 차량의 현재 위치와 헤딩 각은 직접 카메라로 찍어 OPENCV를 활용하여 동작하며 노트북에서 해당 좌표 데이터들을 전송받은 뒤 A* 알고리즘을 사용하여 최단 경로를 계산 후, Waypoint의 형태로 데이터를 차량의 아두이노에 전송한다. 전송된 경로 데이터와 차량의 현재 위치와 헤딩 각 데이터를 이용한 경로 추종은 Pure Pursuit 알고리즘을 사용한다.

4. 설계 요약

캡스톤디자인 7조는 처음에 시각 장애인을 위한 길 안내 로봇 만들기를 계획했으나 학부생인 점과 시간이 그다지 많지 않은 점을 고려하여 현실에서 구현이 가능한 방안으로 라인 트레이서 등을 고려하다가 결국 최근에 이야기가 많은 최단 경로 생성 및 경로 추종, 즉 자율주행에 중점을 맞추어 제품을 구현하는 것을 선택하였다. 최종적으로 캡스톤디자인 7조가 설계하고자 하는 제품을 요약하면 컴퓨터 비전과 OPENCV 좌표계를 사용해 트랙을 Mapping

한 뒤, A* 알고리즘을 기반으로 출발지부터 목적지까지 최적 경로를 탐색하여 이동하는 제품이다. 이동에 필요한 속도나 위치의 제어는 Pure Pursuit 알고리즘을 사용하여 경로를 추종한다.

5. 설계 결정의 논의

김민석 (Pure Pursuit 담당):
현재 프로젝트에서 차량은 양쪽 바퀴의 속도를 다르게 하여 방향을 전환하는 차동 구동 방식의 차량을 사용하며, 경로 추종은 Pure Pursuit 알고리즘을 통해서 한다.

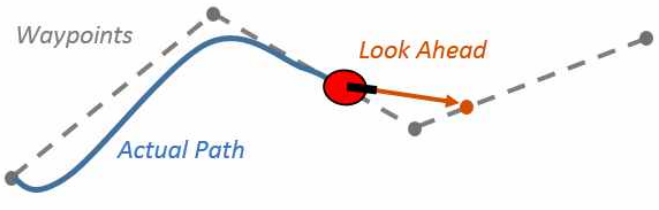


그림 : 간략한 Pure Pursuit 알고리즘 동작 표현

내용을 간략히 소개하면 다음과 같다. 그림을 보면 경로가 일련의 Waypoints로 이루어져 있는 것을 볼 수 있다. 컴퓨터에서 최단 경로(Waypoints), 현재 위치, 차량이 바라보고 있는 방향(Heading Angle)에 대한 데이터들을 받는다. 그리고 차량의 현재 위치에서 Lookahead distance(Ld) 만큼 떨어져 있는 Waypoint를 선택한 뒤, 기존 Waypoints로 이루어진 경로를 따르는 원 모양의 대안 궤적을 추종하게 하는 알고리즘이다. 차량이 원의 궤적으로 동작하도록 먼저 선택한 Waypoint와 Ld에 맞는 원의 곡률을 구한 뒤, 해당 곡률을 가진 원의 궤적을 추종하려면 차량이 얼마나 방향을 틀어야 하는지 나타내는 조향각을 구할 것이다. 조향각이 구해지면 그 조향각으로 동작할 때, 차량이 얼마나 빠르게 회전하는지를 나타내는 각속도를 구한 뒤, 해당 각속도로 회전하도록 하는 양쪽 모터의 속도를 구한다.

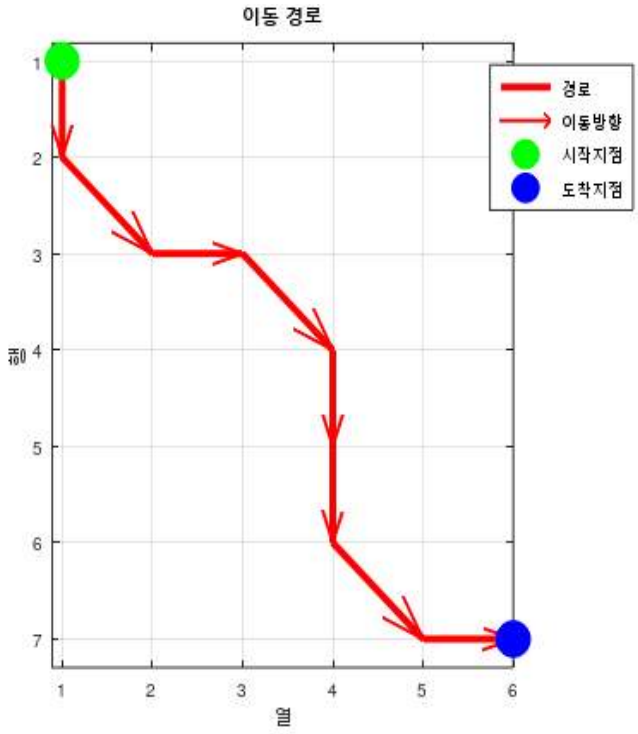
- 강승원: 최단 경로 알고리즘 설계 담당

A^* 알고리즘은 출발 노드와 도착 노드가 정해지면 인접노드와의 거리와 도착노드와의 거리의 비용을 종합적으로 가산하여 최적 경로를 도출하는 알고리즘이다. 출발 노드에서 인접 노드까지의 거리를 $g(n)$이라 한다. 이는 다음 노드간의 직선거리 즉 인접 노드에 대한 맨해튼 거리로 정의한다. 현재 노드에서 도착 노드까지의 거리를 $h(n)$, 이는 도착 노드에 대한 유클리드 거리라고 정의하고 이 함수를 휴리스틱 함수라고 한다. 마지막으로 $g(n)$과 $h(n)$를 더한 값을 $f(n)$이라 정의한다.

$$f(n) = g(n) + h(n)$$

```
>> A_star_example

start point:
   1   1
Goal point:
   7   6
Map Size:
   7   6
Map:
     0     1     1     1     1     1
     1   Inf   Inf     1   Inf     1
     1     1     1     1     1     1
     1   Inf   Inf     1   Inf     1
   Inf     1     1     1     1     1
     1   Inf     1     1   Inf   Inf
     1   Inf   Inf     1     1     1
Path:
   1   1
   2   1
   3   2
   3   3
   4   4
   5   4
   6   4
   7   5
   7   6
Cost (excluding start):
9.2426
```



A^* 알고리즘은 이렇게 도출된 의 값 중 가장 작은 값을 가지는 인접 노드로 이동하여 도착 노드까지의 최단 거리를 탐색하는 알고리즘이다. 작품에서는 상하좌우, 대각선 방향을 고려하는 유클리드 거리 방식의 휴리스틱 함수를 사용한다.

아래는 OCTAVE로 구현한 A^* 알고리즘 코드의 출력값을 나타낸다.

임의의 시작점과 도착점을 설정하면 위에서 설명한 $f(n)$에 따라 최단경로를 탐색하여 이동한다. 이때 start의 비용은 이미 있는 지점임으로 0으로 설정한다. INF는 무한대의 비용으로 장애물을 의미한다. 지도는 임의로 바뀔 수 있으며 이는 시작점과 도착점도 마찬가지다. 실제 작품에서의 구현은 opencv 좌표계에서 받은 좌표값에 변수값을 할당하여 변수값으로 동작하는 A^* 알고리즘을 구현한다. 즉 동작은 변수값으로 하는것처럼 보이나 실제로는 좌표값을 통해 동작하는 형태가 된다.

- 임홍균

장원준의 Python 코드인 장애물을 인식하고 차량의 현재 좌표, 현재 각도를 계산하는 OPENCV와 강승원의 Python 코드인 시작 좌표로부터 도착 좌표까지의 최단 경로 좌표를 계산하는 A* 알고리즘을 라즈베리파이 상에서 병합/보수하였다.

장원준의 Python 코드에서 노란색 객체를 (장애물을) 대표하는 좌표를 노란색 객체의 무게 중심 좌표에서 노란색 객체의 외곽선 좌표로 변경하였으며 객체를 인식하는 주기가 너무 빨라서 카메라 프레임의 변화가 라즈베리파이에 실시간으로 반영이 되지 않는 문제가 있었는데 객체를 인식하는 주기를 늘려서 이를 해결하였다.

강승원의 Python 코드에서는 최단 경로 좌표가 장애물 좌표를 관통하고 지나가는 문제가

있었는데 이를 수정하였다.

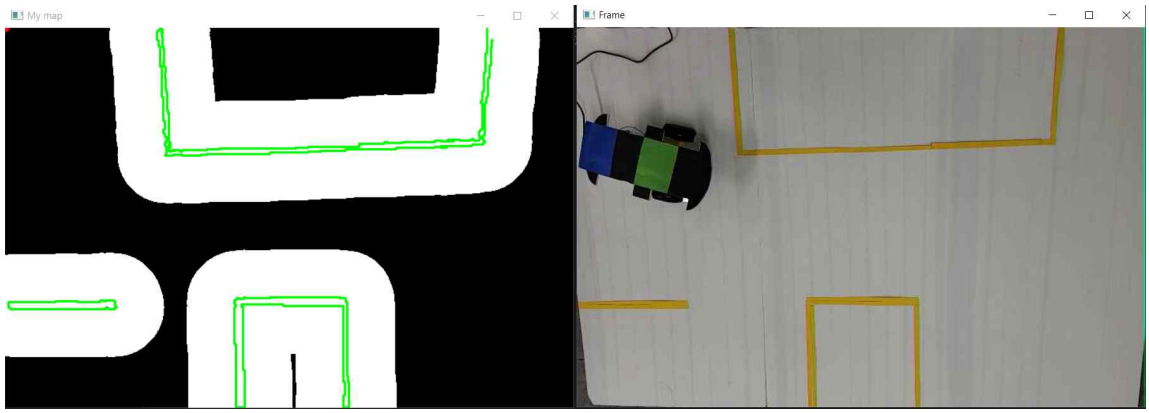
장애물 좌표가 담긴 배열을 my_map, 비용 좌표가 담긴 배열을 cost_map이라고 할 때 장원준과 강승원의 Python 코드를 병합한 방법은 다음과 같다.

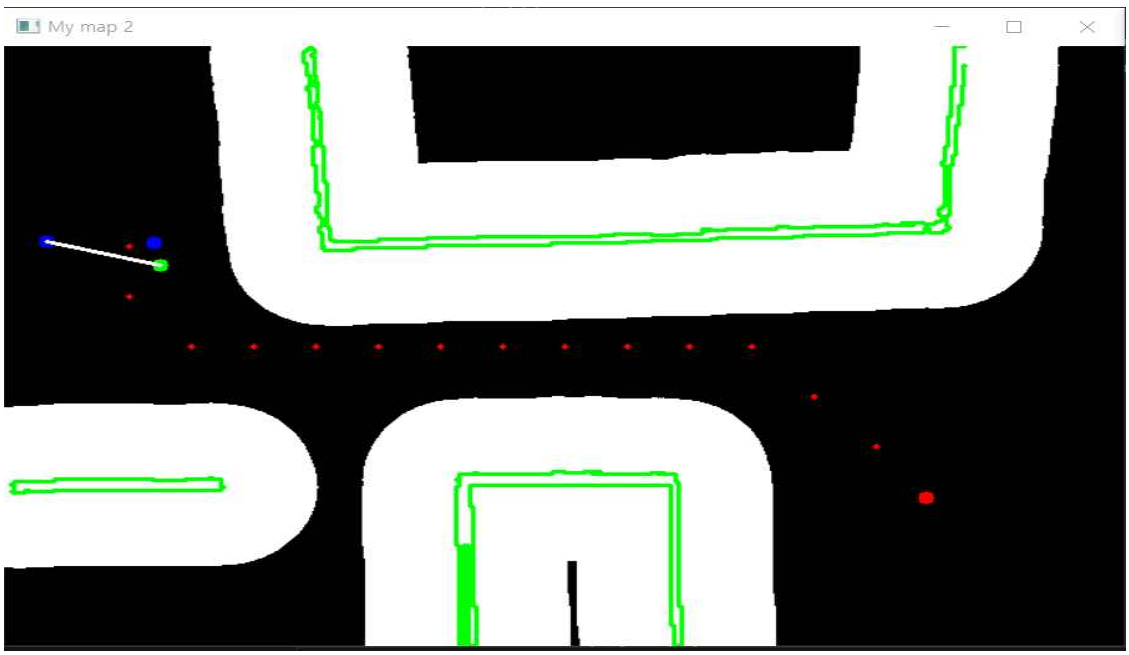
$$mymap = \begin{bmatrix} (0,0,0) & (0,0,0) & (0,0,0) & \cdots \\ (0,0,0) & (254,255,255) & (254,255,255) & \\ (0,0,0) & (254,255,255) & (0,255,0) & \\ \vdots & & & \ddots \end{bmatrix} \quad costmap = \begin{bmatrix} (1) & (1) & (1) & \cdots \\ (1) & (255) & (255) & \\ (1) & (255) & (1) & \\ \vdots & & & \ddots \end{bmatrix}$$

왼쪽 위 그림과 같이 my_map의 흰색 픽셀 좌표 (장애물 좌표)를 (254, 255, 255) 원소로 만들어 보자. 그리고 cost_map을 my_map의 첫 번째 채널로 만든 뒤에 모든 원소에 1을 더하면 my_map의 장애물 좌표를 cost_map에 옮길 수 있다. 현재 좌표는 비용이 0 이기 때문에 cost_map에서 (0) 원소로 만든다.

또한, 라즈베리파이를 사용하여 차량의 현재 좌표와 현재 각도, 최단 경로 좌표, 장애물의 현재 좌표를 계산하는 데 약간의 시간이 필요해서 라즈베리파이보다 성능이 좋은 노트북을 사용할 것이다. 그리고 김민석과 토의를 해서 노트북이 차량이 이동하기 전에 시작 좌표에서 도착 좌표까지의 최단 경로 하나만을 아두이노로 송신하는 것으로 결정하였다.

장원준 : 컴퓨터비전(OPENCV) 담당



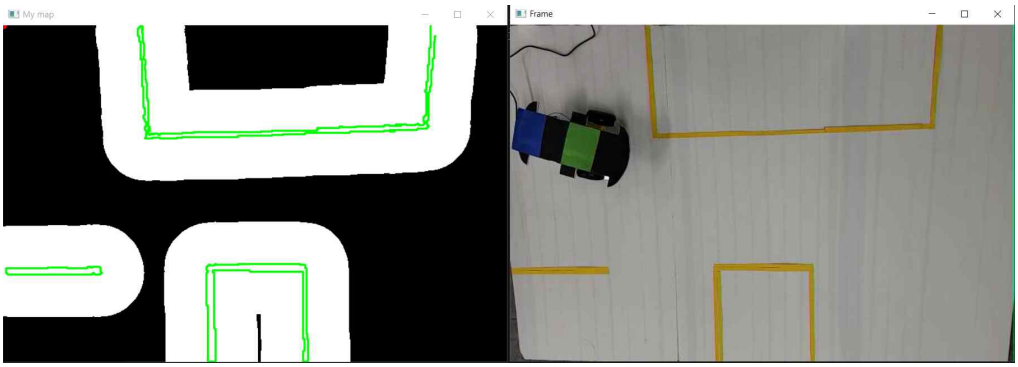


트랙 위 천장에 카메라를 설치하여 위에서 아래로 트랙 전체를 촬영하여 카메라가 트랙 전체를 촬영할 수 있게 세팅을 한 뒤 카메라가 촬영하는 화면에 좌표계를 추가하기 위해서 numpy 프레임워크를 통하여 실제로는 보이지 않는 행렬을 카메라 화면에 깔아두어 촬영화면에 좌표계를 생성하였으며 촬영 화면을 클릭하게 되면 출발점에 대한 좌표가 생성되며 한 번 더 클릭하게 될 시 도착점에 대한 좌표가 생성되며 이 좌표를 기반으로 A* 알고리즘과 Pure Pursuit 알고리즘이 작성이 되며 또한 Pure pursuit 알고리즘에서 필요로 하는 차량은 얼마나 기울어졌는지에 대한 각도 값을 차량 위의 파란색과 초록색 종이 사이의 각도를 통해 각도 값이 구해지며,
A* 알고리즘이나 Pure Pursuit 알고리즘 모두 트랙에서 차량이 현재 어느 위치에 있는지를 알아야 하므로
파란색이 인식된 지점의 좌푯값을 지속해서 터미널을 통해 위치를 알려주며 차량이 이동 시에 벽을 피해 가야 하므로
벽의 경우 해당 프로젝트에서는 노란색으로 지정하였으며
노란색을 최대한 피해서 A* 알고리즘이나 Pure Pursuit 알고리즘이 작성이 될 수 있도록
앞에 나온 각도나 좌표의 경우 한 지점을 의미하지만,
벽의 경우 범위가 있기에 무게중심 코드를 통해 노란색의 중심을 찾은 다음
그 중심으로부터 노란색의 범위를 보정으로 벽이 있는 좌표 부분을 리스트에 담아 저장한 다음 실행을 하게 되면 화면에 벽을 인식한 화면이 나오게 되며 리스트 안에 들어간 좌푯값들은 A* 알고리즘과 Pure Pursuit 알고리즘에서 비용값을 높게 측정하여 노란색을 피해서 경로생성을 하여 이동하며 앞의 설명의 실제 적용 화면은 위의 사진과 같다.

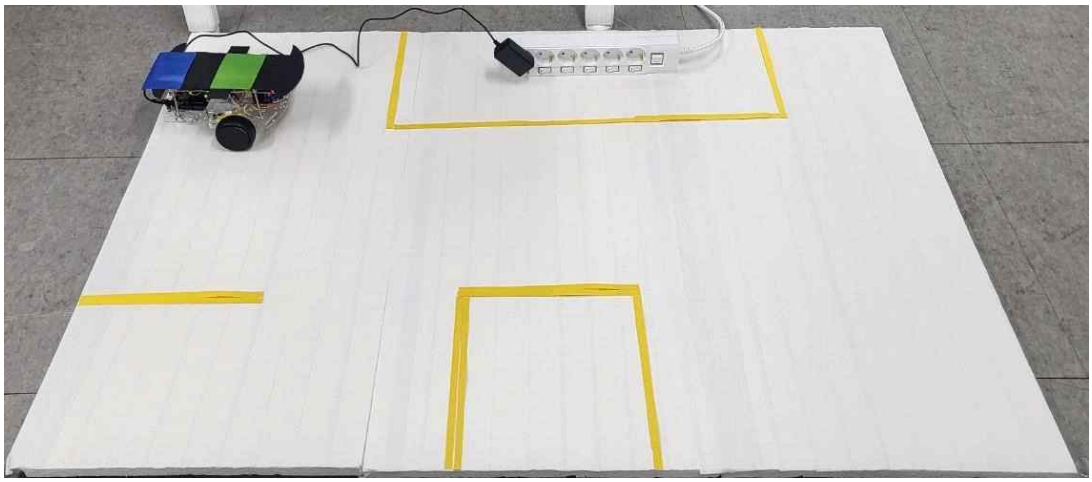
프로젝트 중 시행착오

처음에 차량 위에 색깔을 인식할 때 스티커를 활용하여 진행하려고 하였으나 스티커 겉면에 코팅으로 인하여 조명에 반사가 되어서 빛 번짐 현상이 발생하였고
그로 인하여 정밀한 측정이 어려워 결국 스티커를 포기하고 그림판의 RGB 값을 입력하여 해당 색깔을 컬러프린터로 출력 후 차량 위에 설치하여 빛 번짐 현상을 줄이고 색 인식을 보다 더 정확하게 인식하도록 바꾸었으며,
카메라를 천장에 설치하여 아래로 촬영하는 부분에서 높이에 따라 차량의 범위가 달라 차량의 범위를 보정해야 하는 크기가 다르며,
맵 크기 또한 코드에서 설정한 크기에 맞게 조절해야 하는데
처음 실험한 2층 열람실과 두 번째 실험한 계단강의실 각각의 높이가 달라 맵의 크기를 지정하는데 문제가 생겼으며
실제 전시의 경우 335호에서 진행할 때의 높이가 앞에 두 장소보다 낮아 맵 전체가 카메라에 들어오지 않아 335호 높이에 맞게 전시 전날 급하게 카메라 설치 방식을 수정하였으며 코드 또한 픽셀 수 제한을 낮춰 335호 전시하는 공간에 맞춰 수정하였으며,
또한 조명이 가까이 있어 색깔 인식 범위를 재조정하고 열람실이나 계단강의실에 비해 높이가 낮아 트랙 인식 범위 또한 다시 조절해 프로젝트를 진행하였습니다.

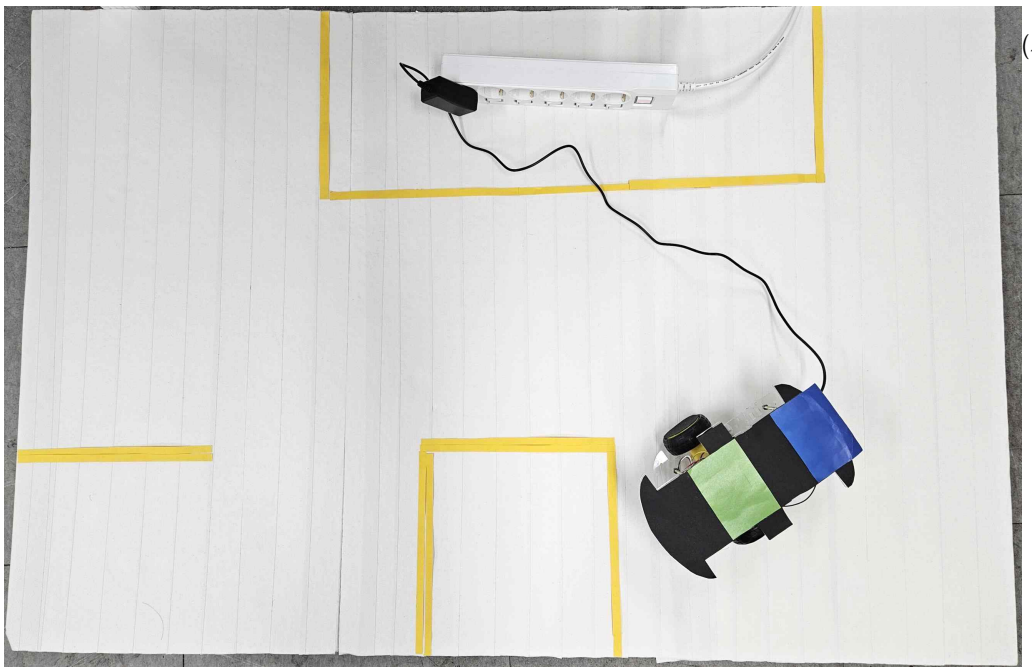
6. 설계 평가



(사진 1)



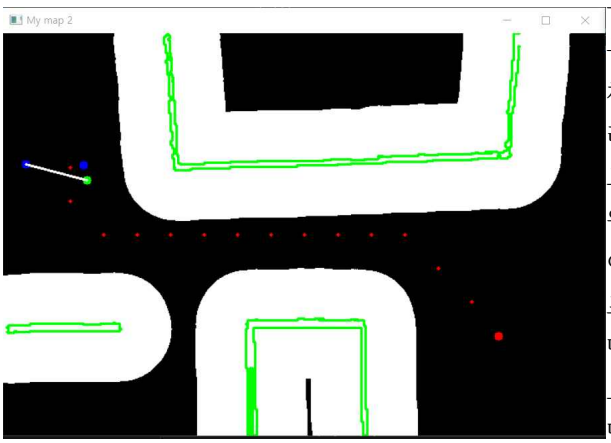
(사진2)



(사진3)

(사진4)

앞에서 이론설명과 모의실험 및 실험을 통해서 최단 경로 추종 차량을 어떻게 동작시킬 것 인지 자세히 살펴보았다. 이제 실제 환경에서 어느 정도의 추종 성능을 보이는지 확인해 볼 것이다. 실제 차량을 주행시켜 볼 트랙은 사진1과 같다.
또한 사진 2를 통해서, 트랙에 매핑이 잘 되었고 장애물 인식도 만족스러운 수준으로 잘하는 것도 확인할 수 있다.
또한 사진 3을 봤을 때, 최소 비용을 가진 경로 역시 A*알고리즘을 통해 잘 생성된 것을 확인 할 수 있다.
사진 4는 차량이 주어진 경로를 추종하는 과정의 일부분이다. 차량의 이동 역시 A*알고리즘을 통해서 생성된 경로에 맞게 추종하며 목적지에 도달 후 멈추는 것을 알 수 있다.



차량에 사용된 아두이노 우노의 버퍼, 메모리, 계산속도 그리고 모터의 성능 등의 하드웨어적인 문제로 인해, 제어 주기 또한 원래 계획보다 길게하고 모터에 피드백 제어도 하지 못했지만, 주어진 환경 내에서 프로젝트의 목표를 달성했다고 평가한다.
만약 더 좋은 하드웨어 장비로 성능을 업그레이드해 프로젝트를 재구성하면 그것에 맞게 더 높은 추종 성능을 보일 것으로 예

상된다.

7. 결론

주어진 트랙을 매핑 후 장애물을 올바르게 인식하는 것을 보였으며, A*알고리즘을 통해 생성된 경로 역시 최단 거리가 생성되는 것을 볼 수 있었다. 그렇게 해서 주어진 경로를 Pure Pursuit 알고리즘을 통해 올바르게 추종하는 것을 확인할 수 있다. 따라서 앞 장에서 설명한 이론들과 설계들이 최단 거리 경로 추종에 유효하게 적용되었다고 결론 내릴 수 있다.
해당 프로젝트는 한정적인 비용 안에서 수행하다보니
사용된 아두이노의 버퍼 용량 한계와 모터의 엔코더나 기본 성능에 있어서 해당 문제를 좀 더 성능이 뛰어난 아트메가나 모터를 이용했다면 스캔을 통해 한번에 입력하는 방식이 아닌 장애물과 최단 경로의 데이터를 변화에 맞춰 실시간 데이터로 받으며 Waypoint 의 개수를 대폭 늘려, 오차범위를 최소화 시킬 수 있었으며, 모터 역시 엔코더를 통한 피드백 제어를 통해서 현재 PWM을 통한 모터 자체를 멈추는 방식이 아닌 PID를 통해 좀 더 정밀하게 속도를 올바르게 제어하고, 더 빠른 제어 주기를 통하여 실시간 제어를 통해 좀 더 퀄리티 높은 작품을 만들었을 것으로 생각되며 보다 더 현재 위치와 차량의 기울어진 정도의 방향을 빠르게 업데이트해 주면 이번에 제출한 캡스톤 디자인 보다 더 높은 수준의 추종 성능을 보였을 것 같다는 결론과 또한 한정적인 자원으로 최단거리 알고리즘, Pure Pursuit 알고리즘, OPENCV 에 대한 코드들을 모두 성공적으로 동작시켰으며,
이를 통해 프로젝트의 목적에 부합하는 동작을 하였으며 최종적으로 캡스톤디자인7조의 프로젝트는 성공적으로 마무리 되었다고 평가된다.

참고자료

Tistory, “Pure Pursuit<Path Tracking Algorithm”
https://iridescentboy.tistory.com/69.

네이버 블로그, “Pure Pursuit Alorithm”,
https://blog.naver.com/thoon1113/222887453151.

연세대학교, “자동차에 대한 나의 비전‘
https://web.yonsei.ac.kr/hgjung/Lectures/DME427/2014-1/%EC%9E%90%EB%8F%99%EC%B0%A8%EC%97%90%20%EB%8C%80%ED%95%9C%20%EB%82%98%EC%9D%98%20%EB%B9%84%EC%A0%84%20--%20Embedded%20System%20&%20Path%20Tracking.pdf.

위키피디아, ”Ackermann steering geometry“
https://en.wikipedia.org/wiki/Ackermann_steering_geometry.

KROS,“차동 구동 로봇의 경로 추종”
https://jkros.org/_common/do.php?a=full&b=12&bidx=2192&aidx=26078.

LUNCHBALLER, “[C++] A^* 알고리즘의 구현”, https://lunchballer.com/archives/793.

OPENCV 기울기 측정, 중심점
https://junworld.tistory.com/25