**Sw 개발계획서**

**- 목 차 -**

1. 개요
   1. 프로젝트 명
   2. 프로젝트 목표
   3. 주요 기능
   4. 기대 효과
2. 프로젝트 조직 (project organization)
   1. 팀 구성
   2. 역할 분담
   3. 의사소통 방법
3. 기술적 접근
   1. 문제 정의
   2. 기술적 목표
   3. 개발 방법론
4. 일정 계획
   1. 개발 단계/일정
5. 자원관리 및 품질관리
   1. 사용 기술 스택
   2. 자원관리 – HW/SW
   3. 테스트 방안
   4. 품질관리 방안
6. 예상되는 어려움
   1. 예상되는 어려움
   2. 대응 방안
7. 결과
   1. 결과물 발표 내용
   2. 결과 보고서

1. **개요**
   1. 프로젝트 명

국민대학교 제8회 자율주행 경진대회

* 1. 프로젝트 목표

본 프로젝트의 목표는 다음과 같다.

* 카메라 하나만을 사용하여 주차구역의 Apriltag 인식
* 차량의 상대위치 추정하여 차량의 이동경로 계획
* 차량을 계획된 경로대로 잘 이동시켜 주차구역에 진입하는 자율주행 시스템 구현

본 프로젝트는 ROS2 기반으로 구성되며 1/10 모형차에서 실행가능한 경량시스템을 목표로 한다.

* 1. 주요 기능
* 카메라 하나를 이용해 주차구역의 Apriltag 인식
* Apriltag를 이용하여 차량과 주자구역 사이의 상대적인 위치와 자세 파악
* 파악한 자량과 주차구역의 상대적인 위치와 자세를 바탕으로 주차 경로 계획
* 시각화 도구 활용해 계획한 주차경로 테스트
  1. 기대 효과
* 본선 대회의 주차미션에 주차 경로 계획 알고리즘 사용 가능
* 본선 대회 차량에 사용되는 ROS2 기반 시스템 설계를 통해 최신 로봇 운영체제 기술의 활용 능력을 강화

1. **팀 구성 및 역할분담**
   1. 팀 구성
   2. 역할 분담
   3. 의사소통 방법
2. **기술적 접근**
   1. 문제 정의

본 프로젝트는 ROS 기반으로 자율주행 차량 시스템을 구현하며 주요 기술적 접근은 다음과 같다

* 차량이 주차구역 인식 후 목표 지점까지 최적 경로 탐색하고 이동해야 한다.
* 차량은 카메라를 통해 데이터를 수집하고 경로 계획 및 제어 알고리즘을 통해 자율적으로 이동한다.
  1. 기술적 목표

ROS2를 사용해 모듈 간의 통신을 구현하고, OpenCV를 통해 카메라 데이터를 처리한다. 또한 처리한 카메라 데이터 활용하여 베지에 곡선 알고리즘으로 구현한다. 구현한 알고리즘의 작동을 시각적 알고리즘으로 구현한다.

* 1. 사용 기술 및 방법론
     1. 사용 기술

ROS2

* 기본 개발 환경 설정
* 모듈 간 통신 및 데이터 처리

OpenCV

* 카메라 데이터 처리하여 Apriltag 인식

Python

* 기본 프로그래밍 언어

Pygame

* Pygame으로 구현한 알고리즘 test

apriltag

* 카메라에 대한 주차구역의 상대적인 위치 측정

(추가해야할 내용: undistold?, 에이프릴 태그 인식 조향각)

* + 1. 알고리즘

베지에 곡선 알고리즘

* 베지에 곡선은 제어점들을 이용해 부드러운 곡선을 생성하는 수학적 알고리즘이다. 주로 n개의 점으로부터 얻어지는 n-1차 곡선으로 표현된다.
* 이번 주차 경로 계획을 위해서는 3차 베지에 곡선을 활용했다. 정해진 시작점과 끝점을 이용해 제어점 2개를 추가적으로 구한 후 베지에 곡선을 구한다.

베지에 곡선 구현

(초기점: P0, 끝점: P3, 제어점 1: P1, 제어점 2: P2)

1. 제어점 1 계산

제어점 1의 경우 초기점에서의 방향과 임의로 설정한 초기점에서 제어점 사이 거리를 이용해 계산한다.

P1 = P0 + d0 \* np.array([np.cos(theta0), np.sin(theta0)])

1. 제어점 2 계산

제어점 2의 경우 끝점에서의 방향과 임의로 설정한 초기점에서 제어점 사이 거리를 이용해 계산한다.

P2 = P3 - d1 \* np.array([np.cos(theta1), np.sin(theta1)])

1. 베지에 곡선 그리기
   1. 주요 구현 계획(시스템 아키텍처)

본 시스템은 ROS2 기반으로 설계되며 다음과 같은 노드와 데이터 흐름으로 구성된다.

* + 1. 사용 라이브러리 (ROS1 기준으로 작성되어 있음. ROS2

|  |  |
| --- | --- |
| 라이브러리/모듈 | 역할 |
| numpy | 수학적 연산 및 벡터 계산 |
| cv2 (OpenCV) | 카메라 영상 처리 및 이미지 변환 (예: 흑백 변환, 영상 출력) |
| rospy=>(rclpy: ros2) | ROS 노드 생성, 토픽 발행/구독 및 ROS 메시지 처리 |
| time | 시간 지연 및 루프 제어 |
| os | 운영 체제 관련 작업 (파일 경로, 시스템 호출 등) |
| math | 수학적 계산 (삼각 함수 등) |
| sensor\_msgs.msg | ROS에서 카메라 이미지를 처리하기 위한 메시지 타입 (Image) |
| xycar\_msgs.msg | Xycar의 모터 제어를 위한 ROS 메시지 타입 (XycarMotor) |
| cv\_bridge | ROS 이미지 데이터를 OpenCV 형식으로 변환 |
| apriltag | AprilTag 검출 및 태그의 위치/자세(pose) 계산 |
| scipy.spatial.transform | 회전 행렬을 오일러 각도로 변환하기 위한 도구 (Yaw, Pitch, Roll 계산) |

* + 1. 주요 노드 및 토픽
* 노드

|  |  |
| --- | --- |
| 노드 이름 | Track\_Driver |
| 역할 | 메인 노드로, 카메라 데이터를 처리하고 AprilTag 검출 및 모터 제어를 수행. |

* 구독 토픽

|  |  |
| --- | --- |
| 토픽 이름 | /usb\_cam/image\_raw |
| 메시지 타입 | sensor\_msgs/Image |
| 역할 | 카메라로부터 실시간으로 이미지를 수신하여 OpenCV 형식으로 변환. |

* 발행 토픽

|  |  |
| --- | --- |
| 토픽 이름 | xycar\_motor |
| 메시지 타입 | xycar\_msgs/XycarMotor |
| 역할 | 차량의 조향각(angle)과 속도(speed)를 설정하여 모터를 제어 |

* + 1. 주요 함수
* 콜백 함수

|  |  |
| --- | --- |
| 함수 이름 | usbcam\_callback |
| 역할 | /usb\_cam/image\_raw 토픽에서 수신된 카메라 데이터를 OpenCV 형식으로 변환하여 image 변수에 저장 |

* 모터 제어 함수

|  |  |
| --- | --- |
| 함수 이름 | drive |
| 입력 | angle: 조향각, speed: 속도 |
| 역할 | 차량의 조향각(angle)과 속도(speed)를 설정하여 모터를 제어 |

|  |  |
| --- | --- |
| 함수 이름 | bezier() |
| 입력 | t, 초기점(P0), 제어점1(P1), 제어점2(P2), 끝점(P3) |
| 역할 | 입력 받은 시작점, 제어점, 끝점 이용해 베지에 곡선 생성 |

* 베지 곡선 함수
* 베지 곡선 시각화 함수

|  |  |
| --- | --- |
| 함수 이름 | Bezier\_visualization\_loop() |
| 역할 | 1. Interaction mode on (실시간 업데이트) 2. 초기 설정(화면 플롯, 시작점 설정) 3. 루프(실시간 업데이트 위해) 4. 화면 실행 |

* 메인 함수

|  |  |
| --- | --- |
| 함수 이름 | start |
| 역할 | ROS 노드를 초기화하고, 카메라 데이터를 처리하여 AprilTag을 검출하며, 차량의 조향각과 속도를 제어. |

* + 1. 데이터 흐름

1. 노드 초기화

ROS 노드(Track\_Driver)를 생성하고, 카메라 데이터 (/usb\_cam/image\_raw)를 구독하고 차량 모터 제어 토픽(xycar\_motor)을 발행한다.

1. 카메라 입력

카메라에서 이미지를 수신하고, OpenCV로 처리하여 흑백 이미지 생성.

1. Apriltag 검출

Apriltag 라이브러리를 사용하여 태그의 ID, 중심 좌표, 그리고 자세(pose)를 계산한다. 과정은 다음과 같다.

1. 카메라에서 수신된 이미지를 흑백으로 변환 (cv2.cvtColor).
2. apriltag.Detector를 사용하여 이미지에서 태그를 검출.
3. 검출된 태그의 ID, 중심 좌표, 모서리 좌표를 출력 및 시각화.
4. 태그의 자세(pose)를 계산하여 위치와 회전 행렬을 추출.
5. 거리 및 자세 계산

태그의 위치와 회전 정보를 기반으로 차량과 태그 간의 거리 및 상대적인 자세를 계산하고, Scipy를 활용하여 회전 행렬을 오일러 각도로 변환하여 Yaw(회전 각도)를 추출한다.

1. 모터 제어

차량의 조향각(angle)과 속도(speed)를 계산하여 xycar\_motor 토픽으로 발행한다.

1. 검출한 데이터로 알고리즘 구현

베지에 곡선 이용해 차량 경로 시각적으로 표현할 수 있는 알고리즘 작성

* 1. 시스템 구현 및 Test

베지에 곡선 알고리즘을 활용해 차량의 주차 경로 계산한다. 계산한 주차 경로를 시뮬레이션 할 수 있는 시각적 알고리즘을 작성한다.

* + 1. 사용 라이브러리

|  |  |
| --- | --- |
| 라이브러리 | 역할 |
| numpy | 수치 계산 및 배열 연산 |
| Matplotlib.pyplot | 그래프 시각화 |
| Matplotlib.widgets | GUI 위젯 제공 |

* + 1. 주요 클래스 (DraggableControlPoints)

DraggableControlPoints는 베지에 곡선 이용한 시각적 알고리즘의 핵심 클래스로, 사용자가 마우스 드래그와 슬라이더를 통해 베지에 곡선을 실시간으로 조작할 수 있게 해주는 기능을 제공한다.

* 초기화 메소드

|  |  |
| --- | --- |
| 메소드 이름 | \_\_init\_\_() |
| 입력 | 축 객체, 시작/끝점, 초기각도, 초기거리 |
| 역할 | 객체 초기화 및 그래픽 요소 생성   1. 멤버변수 초기화 2. 곡선 라인 생성 3. 제어점 scatter plot 생성 4. 라벨 텍스트 생성 5. 방향벡터 화살표 생성 6. 마우스 이벤트 연결 |

* 업데이트 메소드

|  |  |
| --- | --- |
| 메소드 이름 | update\_dirs\_and\_points() |
| 역할 | 방향벡터와 제어점 위치 계산  아래 수식을 통해 계산  • dir = [cos(θ), sin(θ)] • P1 = P0 + d0 × dir0 • P2 = P3 + d1 × dir1 |

|  |  |
| --- | --- |
| 메소드 이름 | update\_curve() |
| 역할 | 베지에 곡선 및 모든 그래픽 요소 갱신   1. 0부터 1까지 100등분한 점에 대해 베지에 메소드를 적용해 점들을 계산 2. 곡선 데이터 업데이트 3. 제어점 위치 업데이트 4. 라벨 위치 업데이트 5. 방향벡터 화살표 재생성 6. 화면 다시 그리기 |

* 이벤트 처리 메소드

|  |  |
| --- | --- |
| 메소드 이름 | on\_press(event) |
| 역할 | 마우스 클릭 시 드래그 대상 결정   1. 클릭 위치와 제어점 간 거리 계산 2. Threshold(0.5) 이하면 드래그 시작   (P1 => P2순으로 검사) |

|  |  |
| --- | --- |
| 메소드 이름 | on\_motion(event) |
| 역할 | 마우스 이동 시 제어점 위치 업데이트   1. 마우스 위치를 방향벡터에 정사영 2. 정사영 길이로 거리 업데이트 3. 실시간 곡선 갱신 |

|  |  |
| --- | --- |
| 메소드 이름 | on\_release(event) |
| 역할 | 드래그 종료 처리  dragging\_point를 None으로 초기화 |

* + 1. 주요 함수
* 메인 함수

|  |  |
| --- | --- |
| 함수 이름 | plot\_bezier\_interactive() |
| 입력 | 시작 좌표, 끝점 좌표, 시작점에서 초기 방향각, 시작점에서 제어점까지 거리, 끝점 제어점까지 거리 |
| 역할 | 1. 시작점 numpy 배열로 변환 2. 메인 화면 생성 & 기본 설정(차트) 3. 베지에 곡선 객체 생성(위에서 정의한 DraggableControlPoints 활용) 4. 슬라이더 UI 생성 (시작점에서 제어점, 끝점에서 제어점, 방향각) 5. 초기 위치 조절 슬라이드 생성(x좌표 조절, y 좌표 조절) 6. 초기 위치 업데이트 함수 사용해 초기 위치 업데이트 7. 화면 실행 |

* + 1. 데이터 흐름

1. 초기화

메인 함수plot\_bezier\_interactive() 실행 후 시작점 numpy 배열로 변환

메인 화면 생성 & 기본 설정(차트)

1. 베지에 곡선 객체 생성

위에서 정의한 주요 클래스 DraggableControlPoints 활용

1. 슬라이더 UI 생성 (시작점에서 제어점, 끝점에서 제어점, 방향각)

초기 위치 조절 슬라이드 생성(x좌표 조절, y 좌표 조절)

1. 초기 위치 업데이트 함수 사용해 초기 위치 업데이트
2. 화면 실행
3. **자원관리 및 품질관리**
   1. 자원 관리 – HW/SW
      1. 하드웨어(HW) 자원

* 개발용 PC
* 170도 어안렌즈 카메라
* 자이카 모터
* 라이다(Lidar)
  + 1. 소프트웨어(SW)
* vIsual studio
* GitHub
* Notion
* ROS2
  1. 테스트 방안

가상 환경 시뮬레이션 사용해 테스트 진행

Pygame 활용해 시각적 알고리즘 구현 후 주차 알고리즘 테스트

* 1. 품질관리 방안
* 기능 정확성 판단 위해 여러 시뮬레이션 가상환경에서 테스트 진행
* 정확한 Apriltag의 거리 및 방향 측정을 위해 실제 카메라 초점거리에 대한 정보 적용해 시뮬레이션 진행(측정값의 안정성 높임)
* Github 이용해 팀의 코드 공유 및 관리
* 차 이동에 대한 시각적 알고리즘 만들어 주차 알고리즘의 동작 확인

디스플레이 연결

도커 이용

1. **예상되는 어려움**
   1. 예상되는 어려움
   2. 대응 방안
2. **결과**