ROS Path Planning & Following

WeGo & WeCAR

WeGo

목 차

WeGo

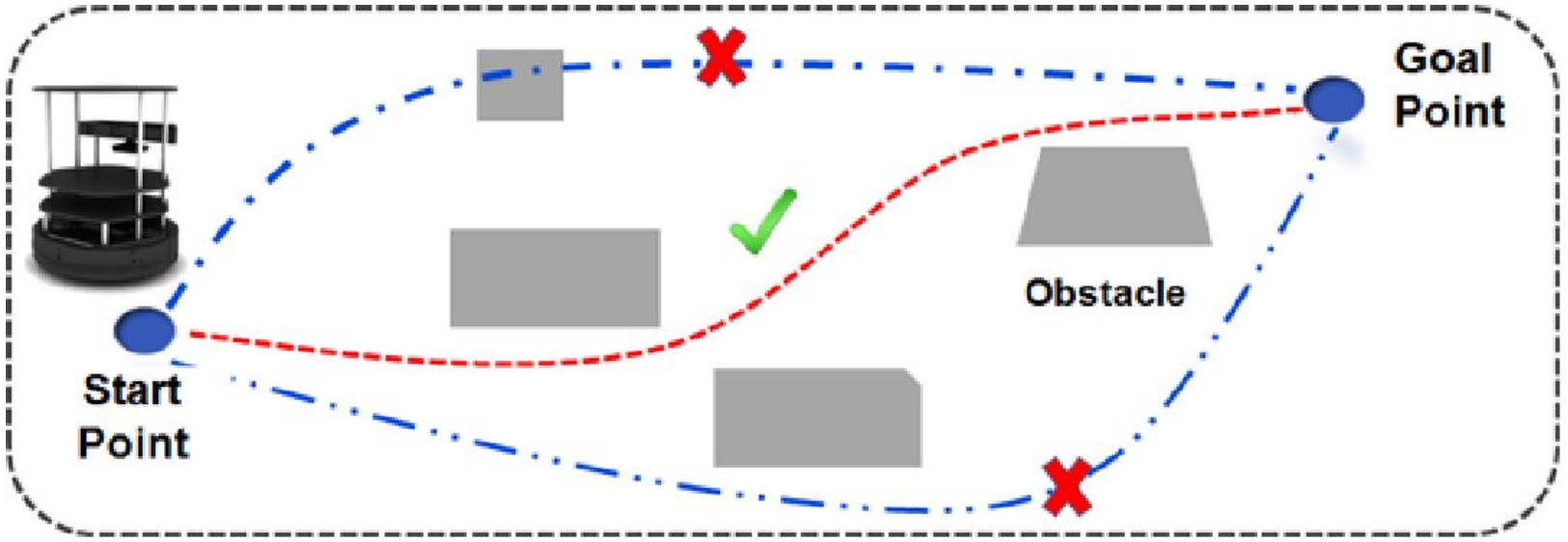
01

Path Planning Introduction

WeGo

01 Path Planning Introduction

- Path Planning
  - Path Planning이란, 지도 상의 로봇의 위치와 목적 장소가 주어졌을 때,
    로봇의 형태 및 크기, 주변 상황을 고려하여, 이동 경로를 생성하는 기술
  - 로봇 외의 다양한 상황에서도 경로 찾기 및 모션 계획에 사용된다.
  - Planner는 크게 Global Planner와 Local Planner로 나뉜다.



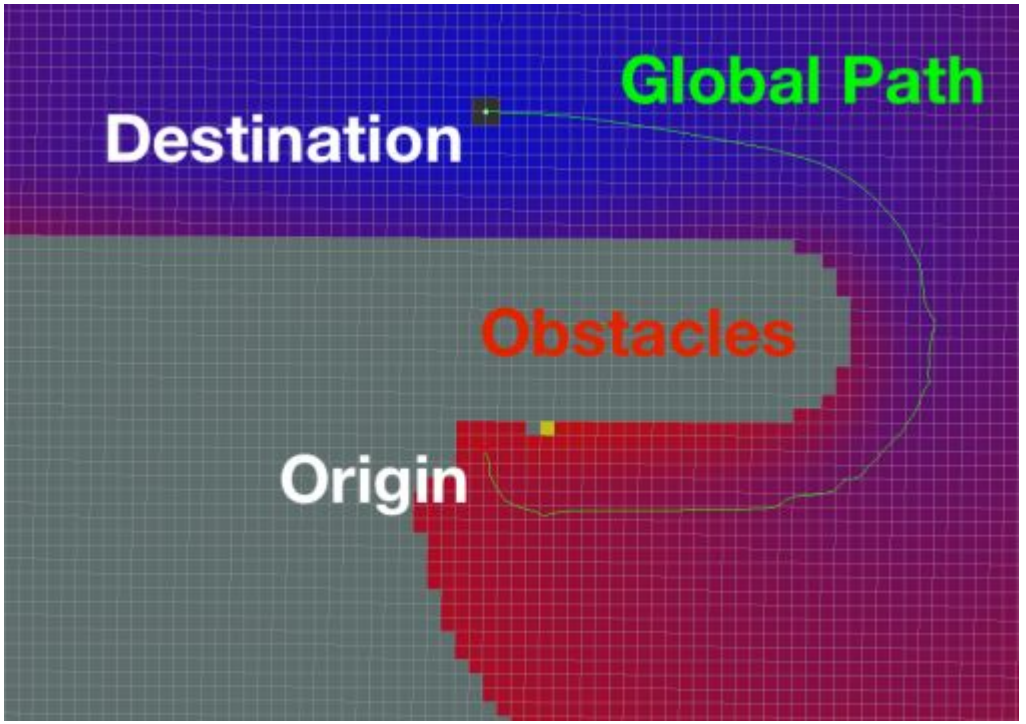
WeGo

01 Path Planning Introduction

- Global Path Planner
  - 전체 지도에서 로봇의 현재 위치와 목표 지점 두 가지 정보를 입력으로.

    지도 상에 있는 장애물과 충돌 없이 도착하는 경로를 생성
  - 입력으로 필요한 값은 지도. 지도 상의 출발 위치. 지도 상의 도착 위치이며.

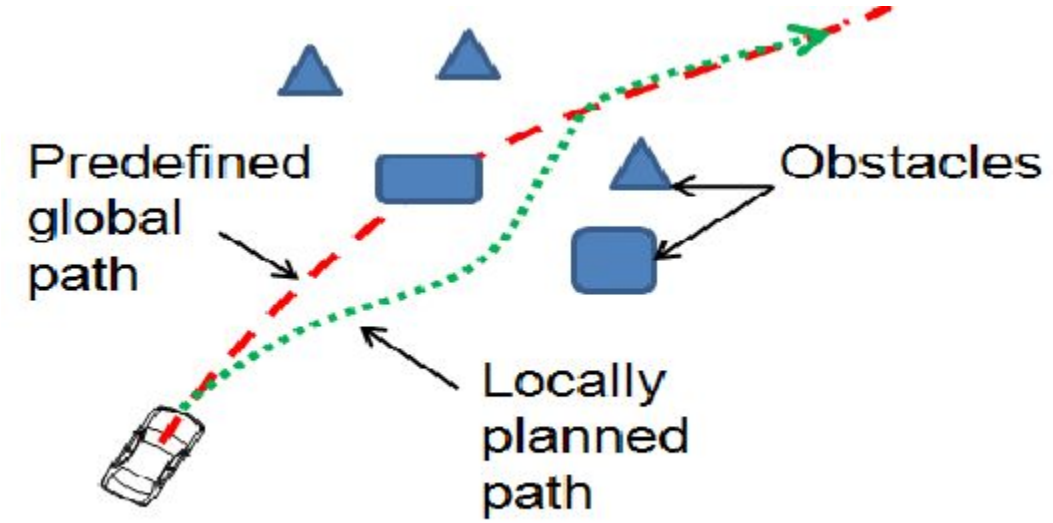
    현재 로봇이 측정하고 있는 주변 환경과는 무관하게 동작한다. (센서 데이터와는 무관)



WeGo

01 Path Planning Introduction

- Local Path Planner
  - 입력 값으로 현재 로봇의 State (위치 및 속도 포함), Sensor Data (주변 상황), Global Path (Global Path Planner가 생성) 를 입력으로 받는다.
  - 현재 로봇의 위치에서 Global Path를 추종하는 최적의 Local Path를 생성



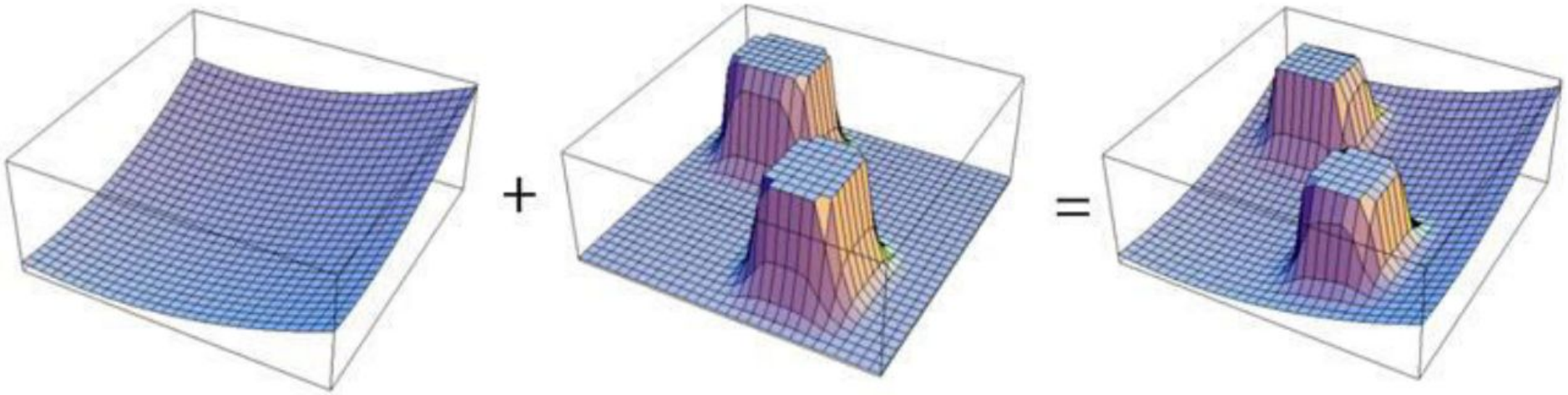
WeGo

01 Path Planning Introduction

- 여기서는 Grid 기반의 Path Planning을 의미하며, 이전에 생성한 Occupancy Grid Map을 기반으로 생성하는 Path를 의미합니다.
- Grid 기반의 Planning 방법은 크게 Potential Field 방식, Graph 기반 Search 알고리즘, RRT 등을 활용할 수 있습니다.

01 Path Planning Introduction

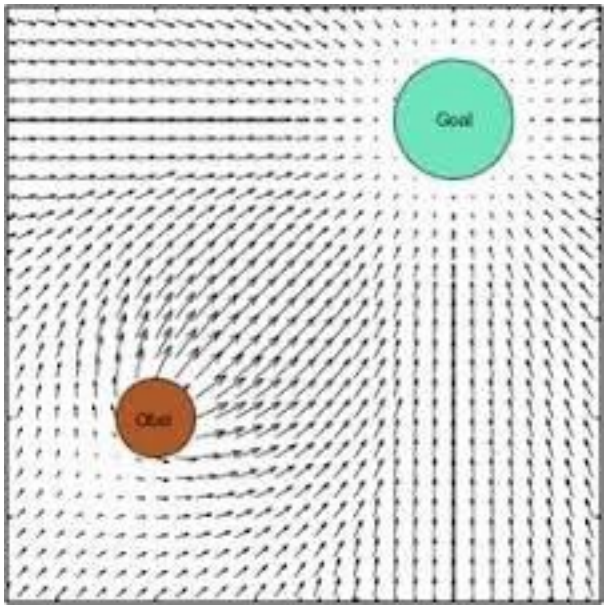
- Potential Field
  - 출발 지점의 Potential을 가장 높게, 도착 지점의 Potential을 가장 낮게 설정한다.
  - 중간에 있는 장애물의 위치에 대해서도 Potential을 Gaussian 함수와 같은 형태로 생성
  - 위 두 개의 Potential을 합쳐서 전체 Potential Field Map을 생성하고, 각 위치에서의 전체 Map에서의 Gradient를 계산하여 이를 연결하는 방법을 통해 경로를 생성

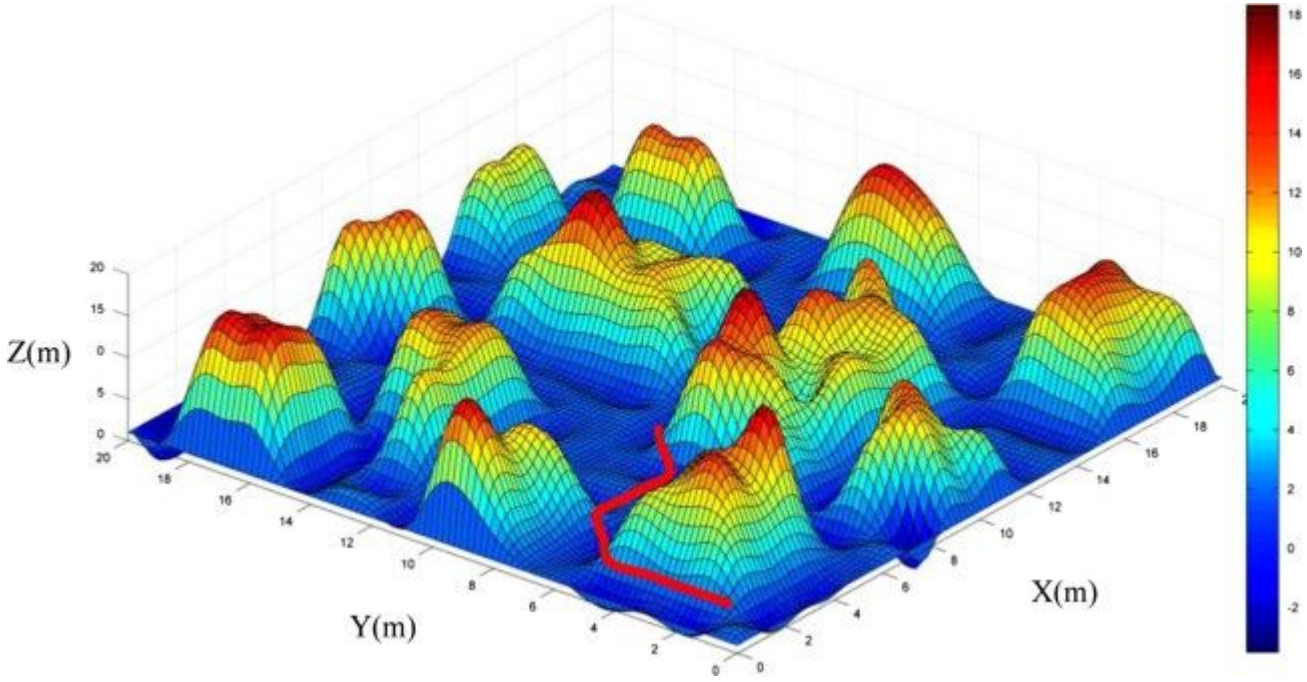


WeGo

01 Path Planning Introduction

- Potential Field
  - Gradient Descent 방식을 통해서 경로를 탐색
  - Gradient를 이용하는 방식이므로. Local Minima, Saddle Point 등으로 인한 문제가 발생할 수 있음

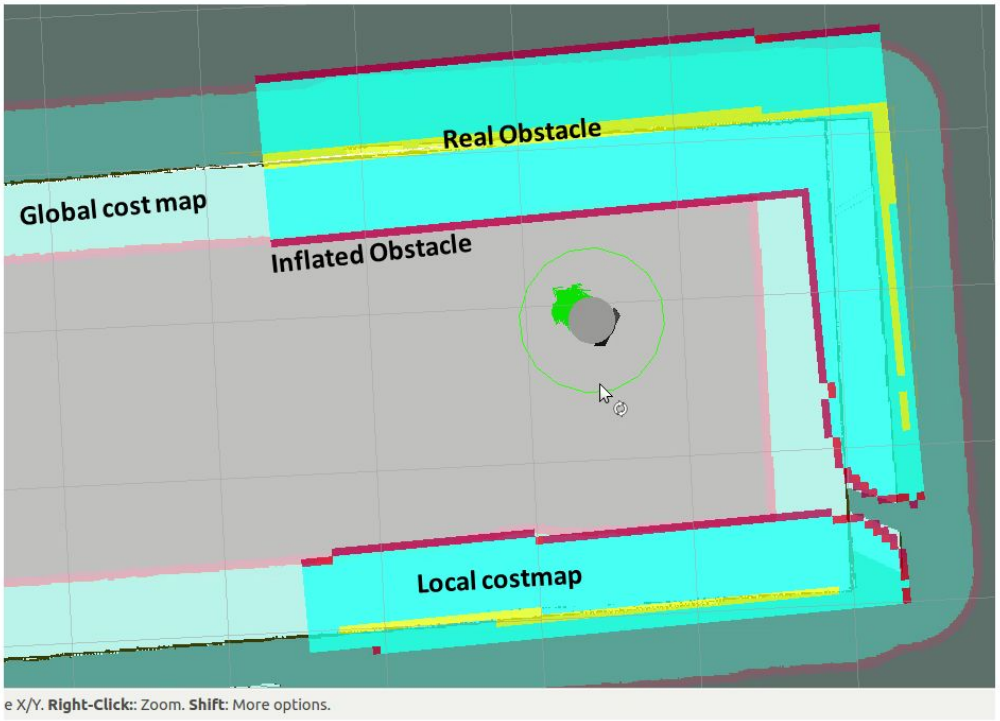




WeGo

01 Path Planning Introduction

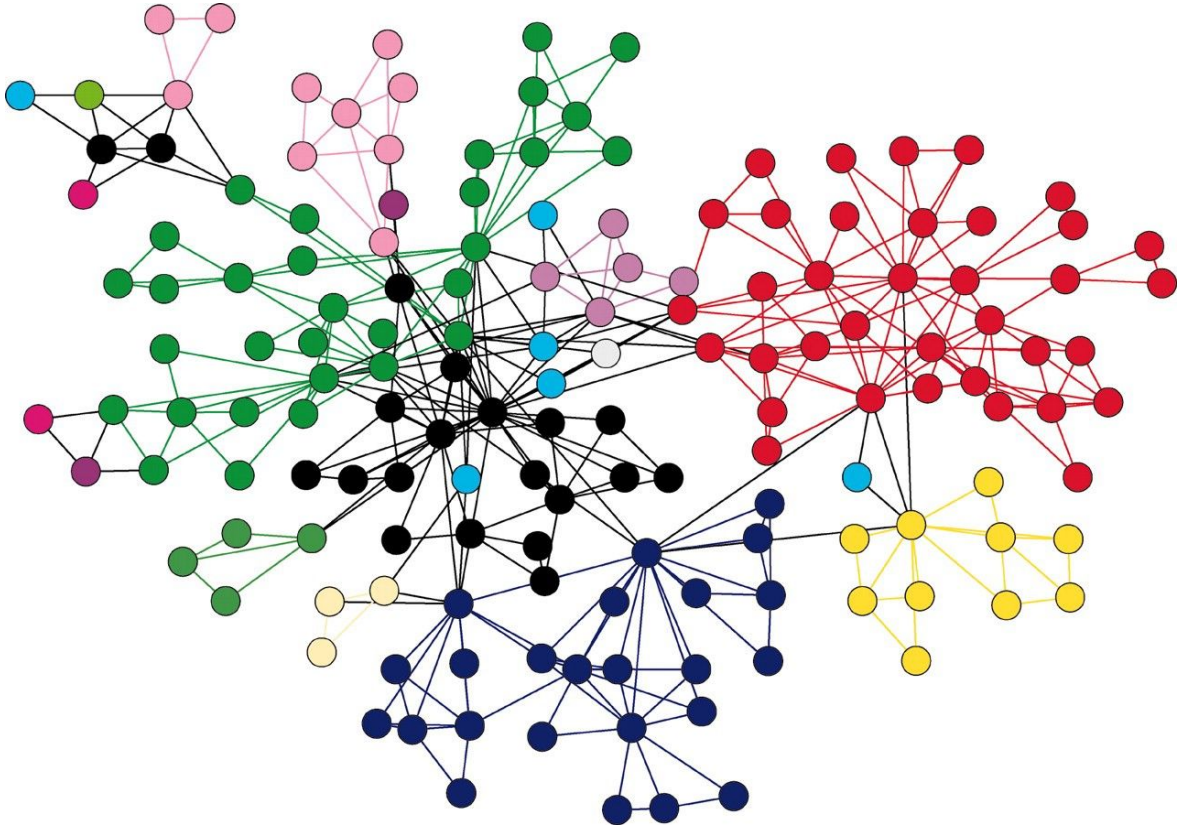
- Graph-Based Search Algorithm
  - Dijkstra's algorithm, A*, Breadth-First Search, Depth-First Search 등
  - 우선적으로 Obstacle Growing을 적용해야하며, Growing Region의 경우, 이동하는 로봇을 원형으로 가정하였을 때, 최소 원의 반지름 이상이 되어야한다.



WeGo

01 Path Planning Introduction

- Graph-Based Search Algorithm
  - Growing 된 결과를 이용하여, 이를 Graph로 변환한다.
  - Graph 변환시에는 4-neighbor인지 8-neighbor인지를 확인하여 변환해야한다.

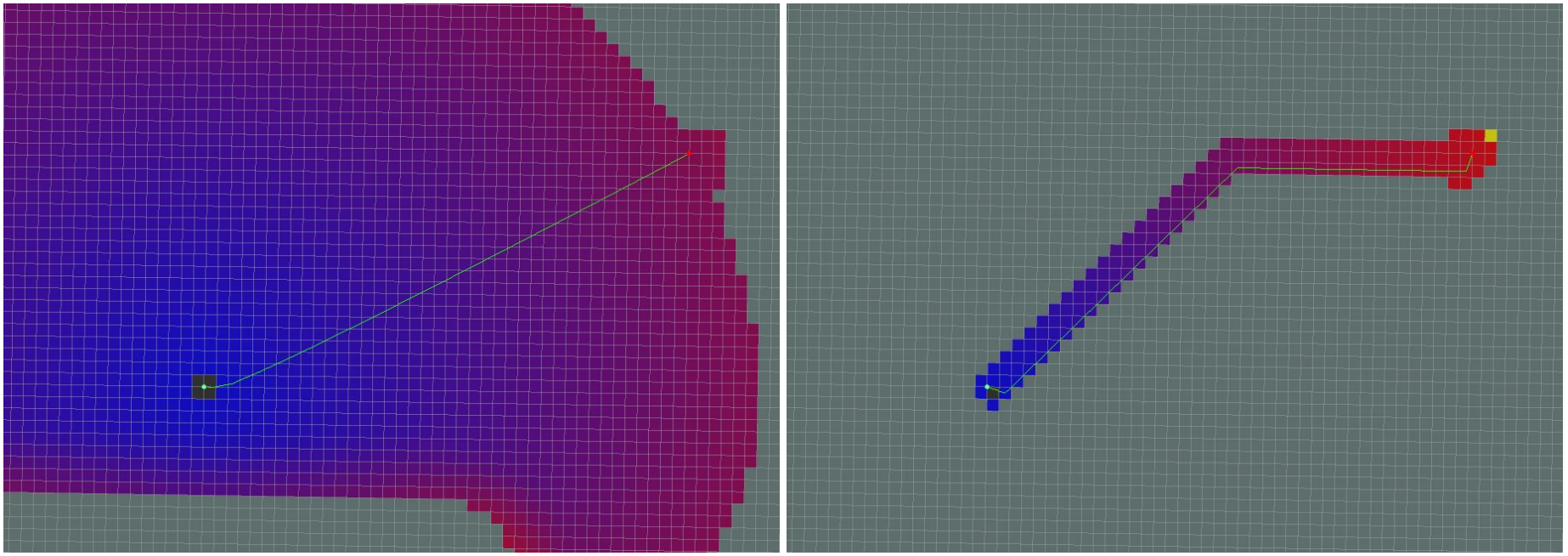


WeGo

01 Path Planning Introduction

- Graph-Based Search Algorithm
  - 변환 후, 입력으로 받은 시작 장소와 목표 장소를 기반으로

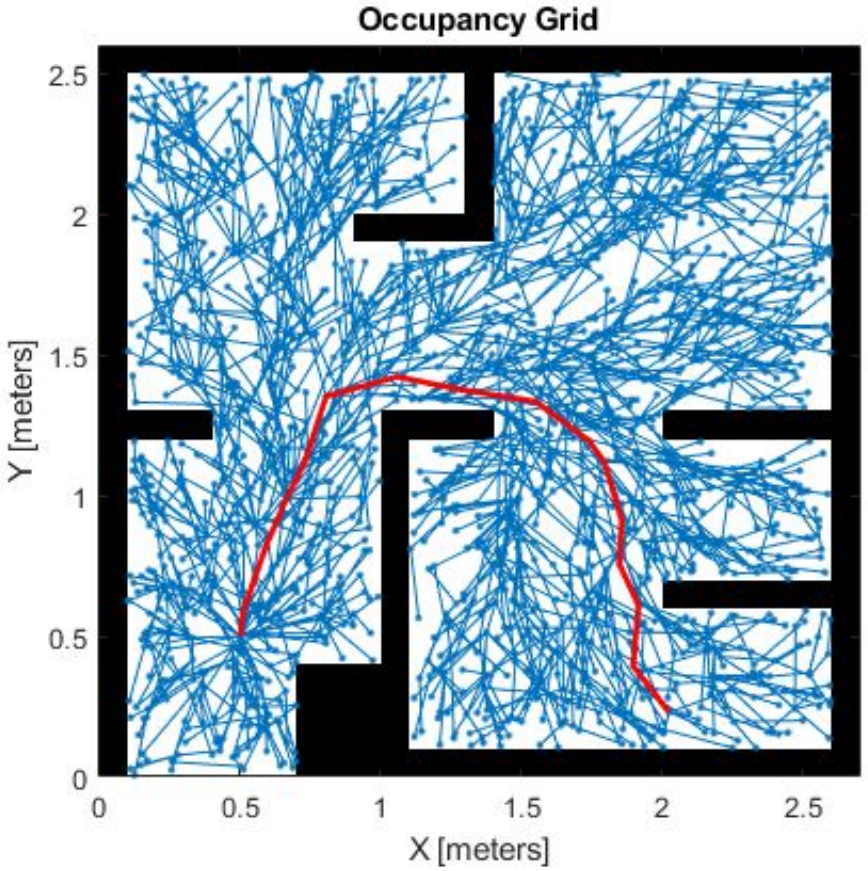
    Graph-based Search Algorithm을 적용하여 경로를 찾는다



WeGo

01 Path Planning Introduction

- RRT (Rapidly-Exploring Random Tree)
  - 시작 장소로부터 Random Tree를 생성하여, 목표 장소에 도착하는 경로를 생성



WeGo

01 Path Planning Introduction

- Global & Local Planner

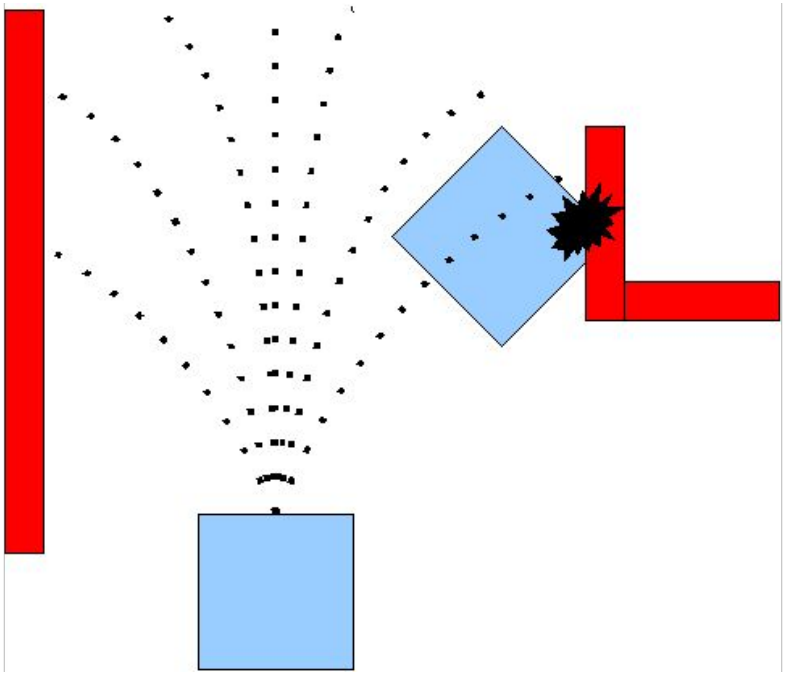
Global Planner	Local Planner
Map-based planning	Sensor-based planning
Deliberative Navigation	Reactive Navigation
Relatively Slower Response	Fast Response
Workspace area is known	Workspace area is incomplete or partially incomplete
Generate a feasible path before moving toward the goal position	Generate the path and moving toward target while avoiding obstacles or objects
Done offline	Done online

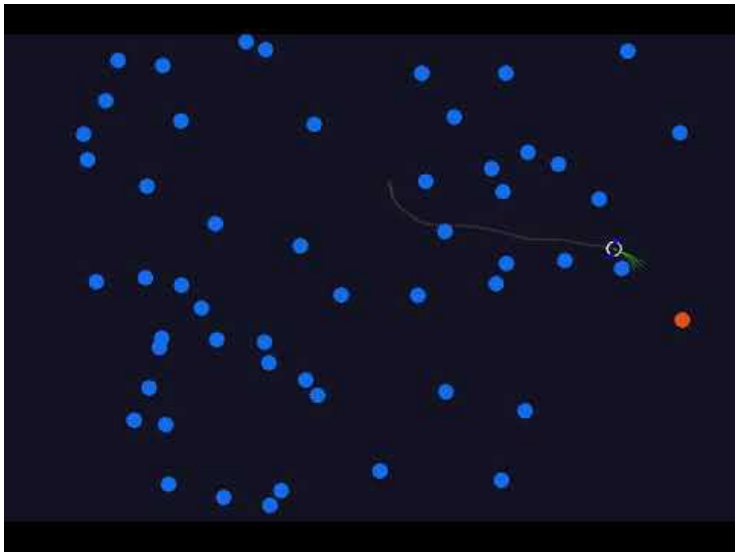
WeGo

01 Path Planning Introduction

- Local Planner
  - Dynamic Window Approach
  - Sampling을 통해 다양한 선속도 및 회전 속도를 생성하고, 이를 기반으로 시뮬레이션을 진행하여, 점수 측정을 통해 높은 점수를 얻은 경로로 주행하는 방식

    http://www.youtube.com/watch?v=Mdq9ElewwA0



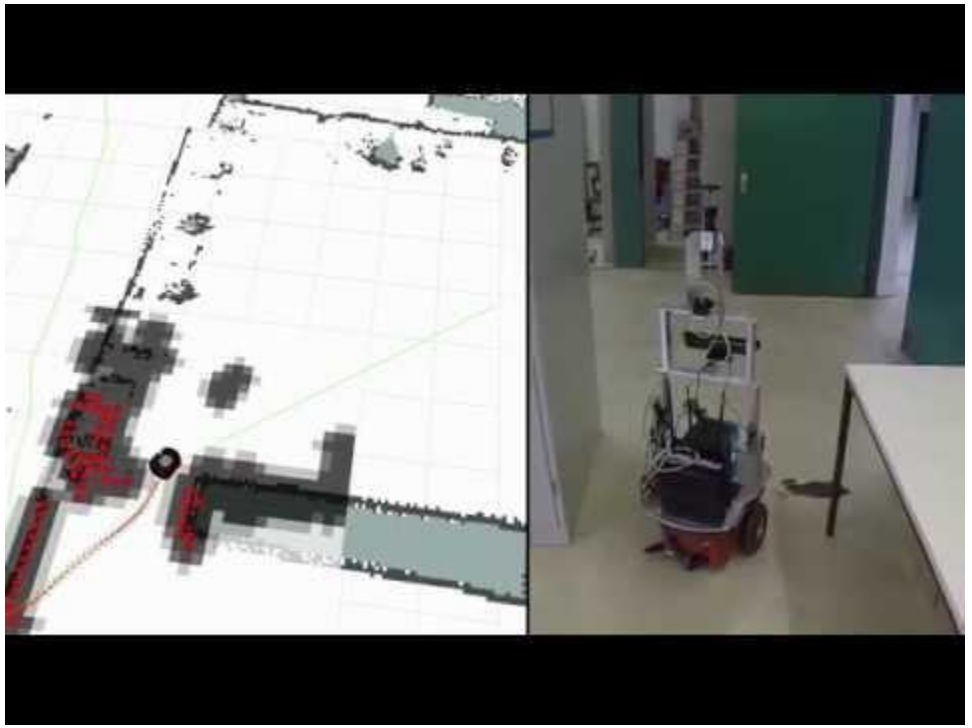


WeGo

01 Path Planning Introduction

- Time Elastic Band Planner
  - 시간, 장애물, 역학적인 제한(Steering angle 등)을 고려하여 최적의 경로를 생성

    http://www.youtube.com/watch?v=e1Bw6J0gHME

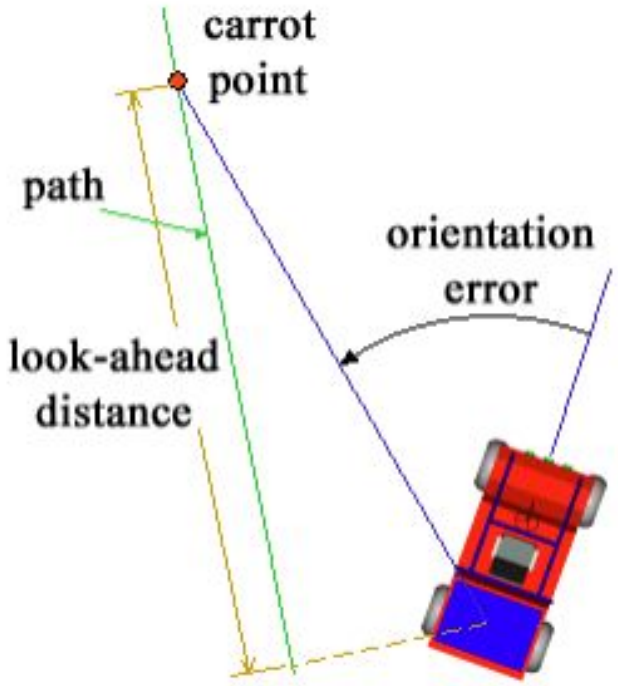


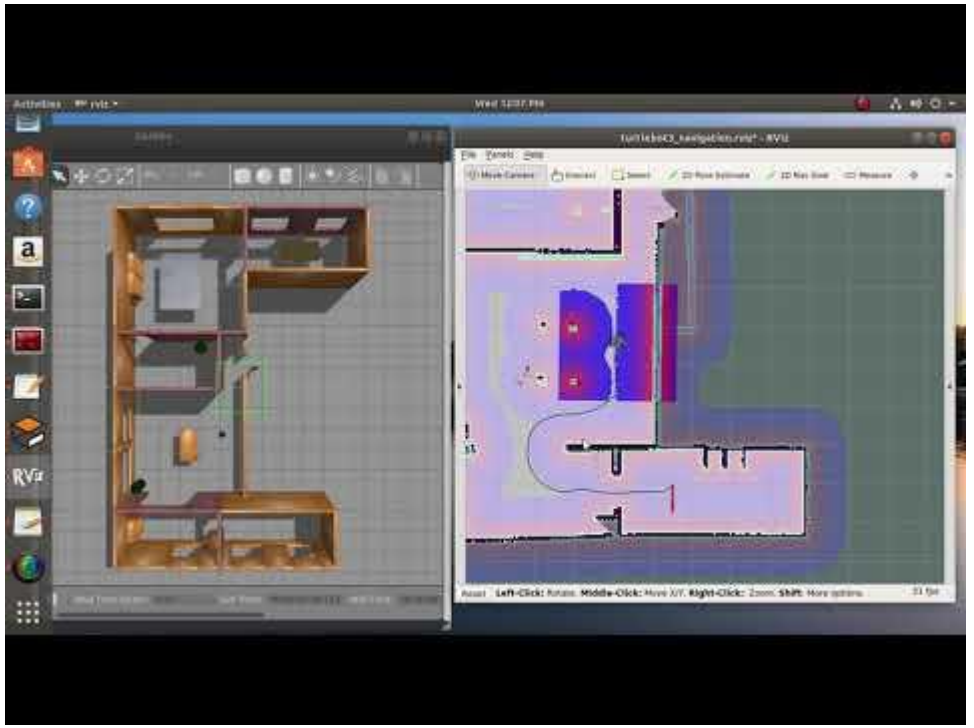
WeGo

01 Path Planning Introduction

- Active Scene Recognition Local Planner
  - Follow the Carrot과 유사한 형태의 Planner
  - Carrot Point를 지정하여, Orientation Error를 최소화하는 Controller

    http://www.youtube.com/watch?v=QvmeDFhwq7Q





02 Path Following Introduction

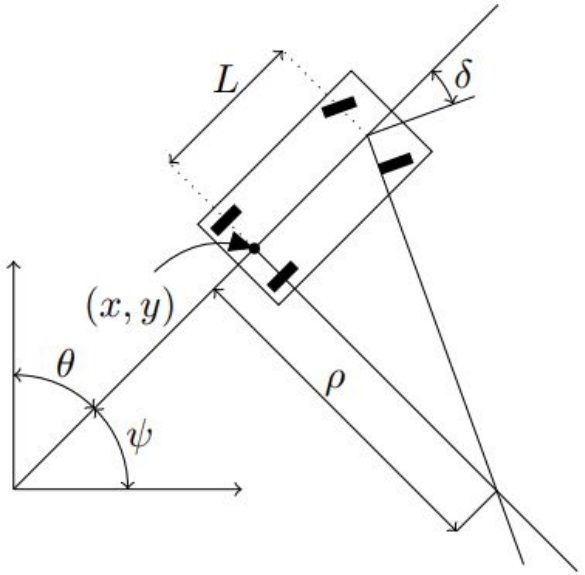
WeGo

02 Path Following Introduction

- Local Planner와 유사하게 Local Planner의 경로를 추종하는 제어 명령을 생성
- 적용하는 알고리즘이 같아도 사용하는 모델에 따라 출력 결과가 달라질 수 있음
- 일반적으로 사용하는 모델은 Ackerman Steering Model (Car-Like Model) 및 Differential-Drive Model 두 가지가 모바일 로봇에서는 일반적인 형태

02 Path Following Introduction

- Ackermann Steering Model
  - 일반적인 후방 구동, 전방 조향의 차량 모델
  - 전방 조향을 통해 원 궤적을 그리는 형태
  - 역으로 계산 시, 원하는 위치에 이동할 때, 어느 정도의 조향각이 필요한지 계산할 수 있음 (Pure Pursuit)



$$\dot{x} = v\cos(\psi) = v\cos\left(\frac{\pi}{2} - \theta\right)$$
$$\dot{y} = v\sin(\psi) = v\sin\left(\frac{\pi}{2} - \theta\right)$$
$$\dot{\theta} = \frac{v}{L}\tan(\delta).$$

WeGo

02 Path Following Introduction

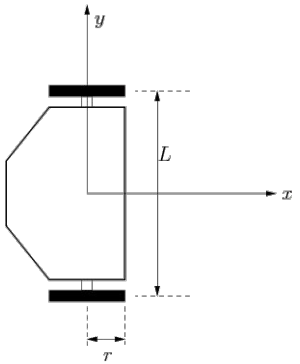
- **Differential Drive Model**
  - 각 바퀴의 제어가 가능하며, 바퀴들의 속도 차이를 이용하여,
    전, 후진 및 회전을 제어할 수 있는 형태

$$\dot{x} = \frac{r}{2}(v_l + v_r)cos(\theta) \qquad \dot{x} = vcos(\phi)$$

$$\dot{y} = \frac{r}{2}(v_l + v_r)sin(\theta) \qquad \dot{y} = vsin(\phi)$$

$$\dot{\theta} = \frac{r}{L}(v_r - v_l) \qquad \dot{\phi} = \omega$$

(a)

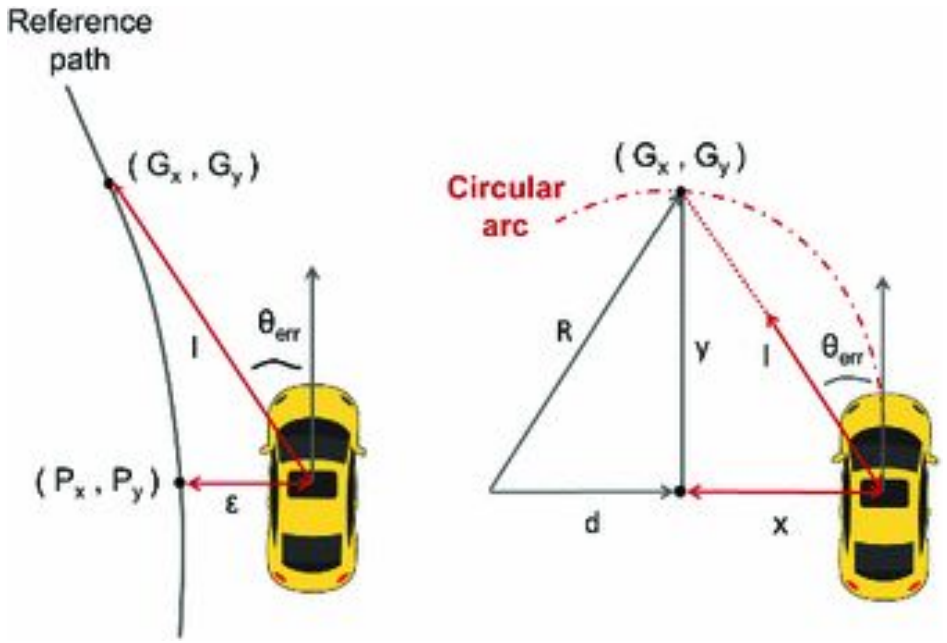


(b)

WeGo

02 Path Following Introduction

- Pure Pursuit 알고리즘
  - 현재 위치에서 목표한 지점까지 이동하기 위한 원궤적을 생성하고, 이 원 궤적을 따라가기 위한 제어값을 계산하는 알고리즘
  - Ackerman Steer Model에서는 바퀴의 각도를 구하는 것이 목적
  - 앞서 설명한 Model을 적용하면 제어값을 계산할 수 있음



WeGo

03

Path Planning & Following Package

WeGo

03 Path Planning & Following Package

- path_planner.py - A* algorithm for grid base search
- Subscribed Topics
  - move_base_simple/goal (geometry_msgs/PoseStamped) - Goal Position
  - /pf/pose/odom or /amcl/pose/odom (nav_msgs/Odometry) - Position of Vehicle
- Published Topics
  - /circle_search/exploration_buffer [nav_msgs/OccupancyGrid]
  - /circle_search/exploration_circles [visualization_msgs/MarkerArray]
  - /circle_search/fast_circles [visulization_msgs/MarkerArray]
  - /fast_trajectory - Fast A* Searched Path
  - /found_trajectory - Result Searched Path
  - /rough_trajectory - Routh A* Searched Path
  - /trajectory/current [geometry_msgs/PolygonStamped

03 Path Planning & Following Package

- pure_pursuit.py - Path Follower based on Pure Pursuit Algorithm
- Subscribed Topics
  - /pf/pose/odom or /amcl/pose/odom (nav_msgs/Odometry) - Position of Vehicle
  - /trajectory/current [geometry_msgs/PolygonStamped] - Followed Path
- Published Topics
  - /followed_trajectory - Followed Path Information
  - /pure_pursuit/lookahead_point [visualization_msgs/Marker] - LookAhead Point of the Path
  - /pure_pursuit/nearest_point [visualization_msgs/Marker] - The Nearest Path Point to the Vehicle
  - /vesc/high_level/ackermann_cmd_mux/input/nav_0 [ackermann_msgs/AckermannDriveStamped]
  - - High Level Control Message
- Parameters
  - odom_topic ("/amcl/pose/odom" or "/pf/pose/odom") - Odometry Topic which to use to follow the trajectory
  - trajectory_topic ("/trajectory/current") - Trajectory to Follow
  - lookahead ("1.5") - Look Ahead Distance (m)
  - max_reacquire ("3.0") - Reacquiring Distance for come back to Trajectory
  - speed ("1.5") - Control Speed
  - wheelbase ("0.335") - Wheel Base of Vehicle

WeGo

03 Path Planning & Following Package

- path_planner.py & pure_pursuit.py
  - $ cd ~/wecar_ws && catkin_make
  - $ source ~/wecar_ws/devel/setup.bash
  - $ roslaunch wecar teleop.launch
  - $ roslaunch wecar amcl_navigation_wecar.launch

    or

    $ roslaunch wecar pf_navigation_wecar.launch

WeGo

03 Path Planning & Following Package

- path_planner.py & pure_pursuit.py
- 실행된 rviz 상단의 2D Pose Estimate 버튼으로 Localization 위치 변경 후, 2D Nav Goal 버튼으로 목표 장소를 전달
- 파란 점 : 경로 상 로봇의 최단 위치, 흰 점 : LookAhead Point, 빨간 점 : 목표 장소
- 조종기 RB 버튼 클릭하여 자율 주행 모드로 동작

WeGo

WeGo Robotics

Tel. 031 – 229 – 3553

Fax. 031 – 229 – 3554

제품 문의: go.sales@wego-robotics.com

기술 문의: go.support@wego-robotics.com