**RT5402 Robotics Term Project Final Report: Study of local path planning in a dining hall service robot scenario**

Team 1: 박지용, 이승윤, 임준혁

Abstract: 터틀봇3를 이용하여 식당 홀에서 서빙을 하는 로봇을 구현하여, path planning의 동작을 탐구하고자 하였다. 복도는 의자와 같은 unmapped static obstacles와 사람과 같은 dynamic obstacles가 혼재하는 곳이라 가정하여 online dynamic path planning의 구현과 탐구에 집중하였다. 이를 위해 DWA와 TEB local planner를 이용하였으며, DWA local planner를 활용하여 주어진 환경에서 적절히 장애물을 피하여 경로를 생성하는 것을 확인할 수 있었다.

# 1. Overview

## 1.1. Scenario and problem definition

카페 혹은 식당에서 음료를 서빙하는 단순한 시나리오를 구상했다. 아래 그림과 같이 가장자리를 따라 테이블이 배치된 2x4m 복도에서 서빙하며, 테이블을 제외한 의자, 사람 등이 장애물로 존재할 수 있다고 가정했다. 실제 서비스는 다음과 같은 순서로 진행된다.

* GUI를 통해 손님이 주문하면 테이블 넘버와 메뉴의 무게 정보가 로봇으로 전달된다.
* 로봇에 수하물을 올려놓으면 무게를 감지하여, 주문된 메뉴에 따라 일정 무게 이상이 감지될 경우 서빙을 시작한다.
* 고정 장애물(의자, 가방 등)과 동적 장애물(사람)을 피하며 지정된 테이블로 이동한다.
* 손님이 메뉴를 수령하면 무게의 변화를 감지하여 0이 될 경우 카운터의 기본 위치로 돌아온다.
* 식기반납의 경우 손님이 반납을 요청할 경우 지정된 테이블에서 일정 시간 기다린 후 기본 위치로 돌아온다.

Diagram

Description automatically generated실내, 바닥, 벽, 천장이(가) 표시된 사진

자동 생성된 설명

Figure 1. 로봇 서비스 환경의 구성.

## 1.2. Robot configuration

바닥, 실내, 목재의, 목재이(가) 표시된 사진

자동 생성된 설명로봇을 구현하기 위해 다음과 같이 하드웨어 플랫폼을 구성하였다.

Figure 2. 로봇의 하드웨어 구성.

* Turtlebot3 Waffle Pi
* 수하물(음식물) 운반 및 무게 감지를 위한 플랫폼
* 2D LiDAR, IMU, 로드셀 센서

무게를 감지하기 위해 터틀봇의 라즈베리 파이에 연결하여 바로 사용 가능한 로드셀 센서를 이용하였다. 본 프로젝트는 실사용을 위한 목적의 로봇 개발이 아니라, 로봇의 서비스 컨셉을 실증하고 그러한 시나리오에서의 path planning을 탐구하기 위한 것이므로 음료 정도의 수하물만을 운반하기 때문에 5kg 로드셀을 이용하였다.

그림에서 확인할 수 있듯, 로드셀과 로드셀 위에 적재한 쟁반이 SLAM에 방해가 되지 않도록 LiDAR 센서의 높이를 조정하여 위치시켰다.

또한 로봇 소프트웨어는 ROS Kinetic 및 Robotis에서 제공하는 Turtlebot ROS 패키지에 기반하여 구현되었다. 이에 대한 상세 사항은 2.1절 및 4.1절에 서술하였다.

# 2. Methodology

## 2.1. Framework

Diagram

Description automatically generated

Figure 3. 로봇 소프트웨어 프레임워크의 구성.

위 그림은 로봇 서비스를 실행시킬 시에 rqt\_graph로 확인할 수 있는 로봇 프레임워크의 구성이다. /tb3\_weight\_pub\_node는 터틀봇에 탑재되어, 로드셀 센서의 무게 신호를 받아 그램 단위로 환산하여 그 정보를 std\_msgs::Int32로 발행한다. /rosbridge\_websocket은 웹기반 GUI의 정보를 ROS에 연결시켜주는 역할을 하여, 목표점으로 할 테이블 번호 및 운반하게 될 메뉴의 종류에 대한 정보를 발행한다. /service\_controller는 로봇의 서비스 전체를 관장하는 노드로, 상술한 세 가지 토픽에 subscribe하여, 그 정보에 따라 로봇의 행동, 즉 목표 지점의 설정과 navigation 시작 여부 등을 결정하게 한다. 본 framework의 구현에 대한 자세한 사항은 4.1절에 서술하였다.

## 2.2. Experiment

실험을 수행하기 전에는 벽과 테이블을 제외한 모든 장애물을 배제하고 turtlebot3\_slam을 이용하여 global mapping을 수행한다. 이후 rviz 인터페이스 상에서 initial pose estimation을 실행한 후, turtlebot3\_teleop으로 로봇을 움직여가며 AMCL의 위치추정의 오차가 줄어드는 것을 확인하면 터틀봇의 navigation 준비가 완료된다.

정적 장애물 3개를 회피하여 3가지 목표점으로 왕복하는 경우와 동적 장애물 1개를 회피하며 3가지 목표점으로 왕복하는 경우의 두 가지 실험을 진행하였다. 이에 대한 자세한 구성은 [실험 영상에서](https://drive.google.com/file/d/1Q56xmppxxDOAa4IEUGT_MbdGiu611j9K/view) 확인할 수 있다.

# 3. Results

## 3.1. Observations

실험 결과 아래와 같은 결과를 관찰하였다.

* DWA local planner를 차용하였을 시, 고정 및 동적 장애물을 대체로 잘 피해가는 만족스러운 결과를 확인하였다.
  + 그러나 고정 장애물 간 거리가 로봇의 크기와 비슷할 경우 충돌을 회피하기 위해 멈추거나 일시적인 oscillation이 있는 경우가 있었다.
  + 목표 위치에 도달했을 경우 orientation을 찾기 위해 한 바퀴 이상 도는 경우가 있었다.
  + 동적 장애물이 갑작스럽게 지나치게 가까운 곳에 나타났을 경우 새로운 local path를 찾는데에 다소 시간이 소요되었다.
  + LiDAR 센서가 무릎 높이에 있고, 테이블과 충분히 가까운 위치에 로봇을 주차하기 위해 충돌 회피 거리를 짧게 설정하였기 때문에 발에 부딪히는 경우가 발생했다.
* TEB local planner는 사람 한 명과 같이 장애물이 하나만 있을 경우 경로 생성에 지장이 없었으나 장애물이 두 개 이상일 경우 장애물 사이의 거리가 충분함에도 심한 oscillation을 관측하였다.
* 서비스 수행 후 기본 위치로 돌아올 때에 대체로 10cm 이내의 오차로 목표 지점에 도달하는 것을 확인했다. 이로써 LiDAR와 IMU 센서의 fusion만으로 수행하는 AMCL 위치 추정이 본 프로젝트의 목표에 적당함을 알 수 있었다.
* 쟁반에 올린 수하물의 무게가 1kg을 초과할 경우 바퀴에 slip이 일어나는 것을 관측했다.\

## 3.2. Challenges

이 프로젝트를 진행함에 있어서 기술적인 어려움과 시간적 제한으로 몇 가지 어려움을 마주하였다. 그 하나로, 초기에는 navigation stack 상의 local path planner를 직접 설계하여 fuzzy logic 및 artificial potential field 기반 local planner를 구현하고자 하였으나, C++ 및 ROS Plugin을 다루는 데에 어려움이 있었으며, navigation stack의 제어기는 위치 제어가 아닌 속도 제어로 구현되어 있어 경로계획의 출력을 도출하는 것이 쉽지 않아 기존에 공개되어있는 local path planner plugin을 사용하게 되었다.

TEB local planner를 사용했을 경우, obstacle 주변에서 심한 oscillation이 일어났는데, 그 원인으로는 다음과 같이 추측된다.

* 로봇 모델의 parameter가 부정확하여 constrained optimal control의 일종인 TEB가 좋지 않은 결과를 도출한다.
* 장애물과의 최소 거리, cost gain값 등 local costmap 생성을 위한 parameter가 적절치 않게 설정되었다.
* 경로 최적화는 10Hz인데 비해, 컴퓨터의 성능이 이를 따라가지 못했다.

MPC (model predictive control) local planner를 이용한 실험도 진행하고자 하였으나 설치를 할 수 없었다.

## 3.3. Lessons and Takeaways

먼저 이번 프로젝트를 통해 ROS와 ROS의 활용에 대한 이해도를 키울 수 있었다. 학부생 팀원들은 이번 강의에 있어서 우분투를 처음 접해보았을 정도로 로봇 개발 및 활용에 대한 경험 및 이해가 부족한 상태에서 시작하여, 각자 터틀봇에 원하는 센서를 연결하여 이의 정보를 발행하는 노드와 로봇에 명령을 보내는 GUI 및 서비스 전반을 구현하는 노드를 만들어 이를 ROS package로 편리하게 구현할 수 있었다. 이를 통해 ROS topic 및 action client에 대한 학습을 할 수 있었다. 또한, 기본 ROS package로 제공되는 것이 아닌 local planner를 이용해보고자 함으로써 navigation stack의 구조와 ROS plugin 등에 대한 공부를 할 수 있었다.

추가로, 이론적으로는 동적 환경에서 압도적으로 부드러운 주행을 보여주어야 할 TEB planner가 보다 간단한 DWA planner보다 적용하기 어려움을 경험하였다. 상황에 따라서는 보다 단순한 솔루션이 문제를 일으킬 가능성이 적고 상황에 맞추어 빠르게 조정할 수 있어 더욱 효과적일 수 있다는 것을 알게 되었다.

# 4. Appendix

## 4.1. Code implementation

### 4.1.1. Weight node

[tatobari의 Github repository](https://github.com/tatobari/hx711py)에서 무게 감지 코드를 받아 python example.py를 라즈베리 파이에 입력하여 실행시켰다. 이후 실제 무게를 측정한 물건을 올려보면서 코드 상 reference unit을 적절히 수정하여 무게값의 calibration을 진행한다. 코드가 산출한 무게값은 rospy.publish를 이용하여 발행한다.

### 4.1.2. GUI implementation

인간과 터틀봇의 상호작용을 구현하기 위해 ROS의 rosbridge package를 이용했다. Rosbridge package는 메뉴의 주문을 담당하는 html기반 페이지와 ros사이의 통신을 담당한다. 기본적으로 주문 페이지의 구성은 order와 return으로 이루어진다. 메뉴 주문시에는 table, menu를 선택한 후 order버튼을 누르고, 빈 그릇을 반납할 때는 table만 선택하고 return버튼을 눌러주어 정보를 ros로 전달한다. 메뉴 주문 페이지에서 발행하는 정보는 주문 테이블의 위치, 메뉴이며, 각각 gui\_table, menu라는 이름의 topic으로 발행된다. Table과 menu를 주문자가 입력한 후 order버튼을 누르면 service node에서 위의 토픽을 구독하여 터틀봇이 작동하도록 한다. 이후의 작동은 service node의 명령으로 이루어진다. 만약 빈 그릇을 return하려면 table number만 입력하고 return버튼을 누른다. 그러면 rosbridge가 table number와 return(message of menu topic)을 발행하고, service node가 이를 구독한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Figure 4. GUI 실행화면

상업적으로 확장한다면 각 테이블마다 이 페이지를 구현한 디스플레이가 구비되어있으며, Table number는 각 디바이스에 내장되어 있는 정보로 전달되므로 따로 입력을 필요로 하지 않는다. 빈 그릇의 반납을 질량 측정을 통해 판단하기는 어려우므로 주문자가 빈 그릇을 모두 서비스로봇에 올려두었다는 메세지를 전달하는 기능이 필요할 것으로 보인다. 예를 들어 Return을 한 번 눌렀을 때 서비스로봇이 오고, 빈 그릇을 모두 입력한 후 Return버튼을 눌러 반납을 하는 방식을 구상할 수 있다.

## 4.2. Execution instruction

2.2절의 navigation 준비 과정이 끝난 경우 teleop을 중단하고 service package를 실행시킨다.

$rosrun service\_pkg service.py

GUI노드를 실행시킨다.

$node ~/GUIserver/app.js

터틀봇의 라즈베리 파이에 SSH로 액세스하여 weight node를 실행시킨다.

$cd hx711py & $python example.py

이후 웹브라우저에 localhost:3000으로 접속하여 GUI를 실행시키고 주문을 넣어 메뉴 무게에 맞추어 적재물을 로봇에 탑재하면 서비스가 실행된다.