Navigation2 with Gazebo

CLOiSim+Gazebo+Turtlebot3

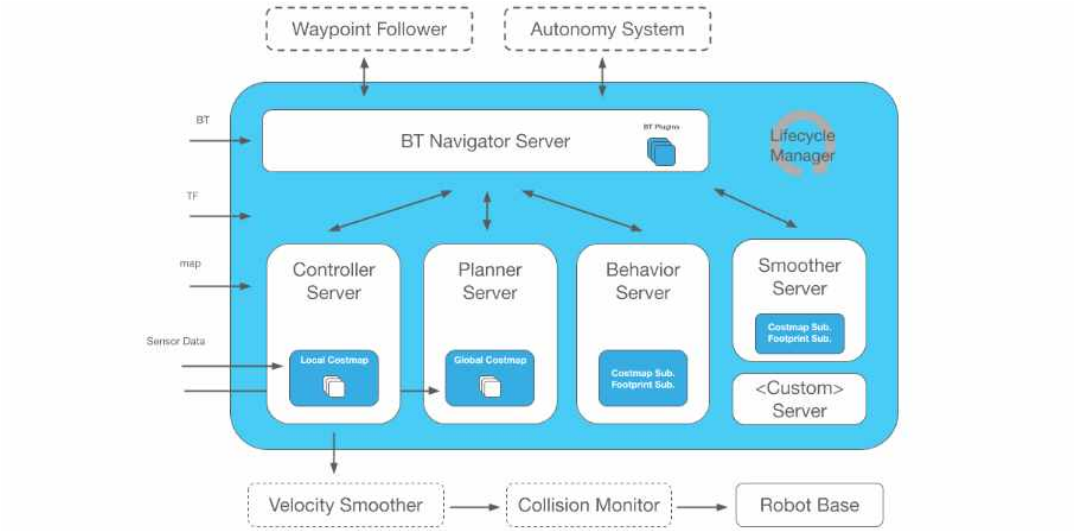
소속	ETRI 시각지능연구실 연구연수생
성명	변민정

목 차

Ⅰ. 서론

1. About Navigation2

특정 로봇이 A 지점(Initial Pose)에서 B 지점(Goal Pose)으로 안전하게 이동할 수 있는 방법을 찾는 Task로, 동적 경로 계획, 모터 속도 계산, 장애물 탐지 등의 과정을 포함한다. 각 노드는 ROS 서버를 통해 행위 트리(BT, Behavior Tree)와 통신하며 아래와 같은 구조를 갖는다.

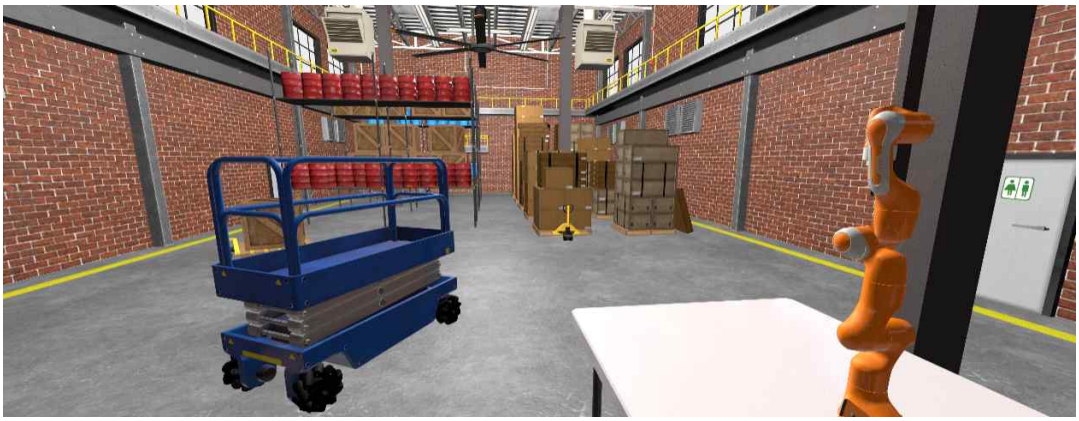


맵 서버를 통한 로드/서비스/저장, 지도 국지화, 경로 계획, 로봇 제어, 센서 데이터의 변환, 로봇 동작의 구축, 장애 발생시 복구, 순차적 경유지 추적, Lifecycle 감시, 알고리즘과 관련된 기타 플러그인 등을 지원한다. Navigation2는 단순 Task로서 다양한 로봇에 적용될 수 있지만, 먼저 Gazebo에서 가장 보편적으로 사용하는 TurtleBot이 Agent로서 동작할 수 있는지 테스트해본다.

2. About Gazebo

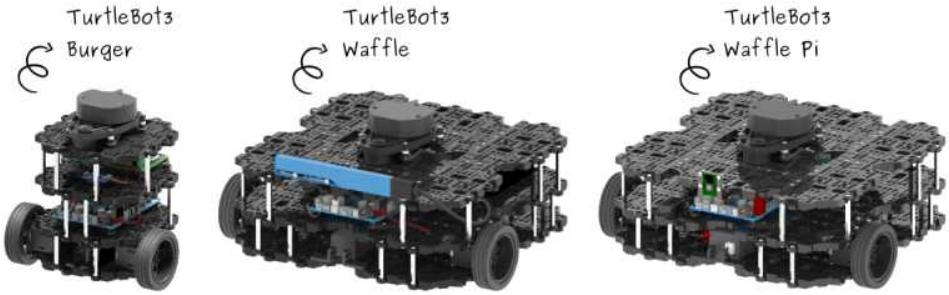
GAZEBO

Gazebo는 개발 라이브러리와 클라우드 서비스 등의 도구를 제공하여 Simulation을 쉽게 만들어주는 가상 시뮬레이터이다. 높은 정확도의 센서 스트림을 가져 현실적인 환경에서 물리적 설계가 용이하며, 연속 통합 테스트 등에 활용할 수 있다. Simulation에 필요한 수학 데이터, 3D Mesh의 관리, 비동기 메시지 전달 등의 요소들은 오픈 소스 라이브러리에 캡슐화된 상태로 제공된다.



3. About TurtleBot3

TurtleBot은 오픈 소스 소프트웨어를 갖춘 저렴한 개인용 로봇 키트이다. 2010년 11월 최초로 개발되었으며, TurtleBot을 사용하면 주변을 운전(탐색)하고 3D로 관찰할 수 있을 뿐만 아니라 다양한 응용 프로그램을 만들 수 있다.
TurtleBot3은 교육, 연구 등 ROS 기반의 프로토타이핑에 적합한 로봇이다. 기능성과 품질을 낮추지 않는 선에서 플랫폼의 크기를 획기적으로 줄였으며, 가격을 낮추는 동시에 확장성을 제공하여 널리 사용되고 있다. 터틀봇3과 4는 현재도 개발이 진행 중이다. 본 Navigation2 Task에서는 TurtleBot3의 Waffle 모델을 사용할 것이다.



TurtleBot3의 핵심 기술은 SLAM, Navigation, Manipulation으로 홈서비스 로봇에 적합하다. SLAM(동시 위치추척과 매핑) 알고리즘을 실행하면 지도를 제작할 수 있으며, 동시에 실내를 운전할 수 있다. 또한 노트북, 조이패드 또는 스마트폰으로 원격 제어가 가능하다. 2절에서 수행할 Navigation의 Bringup Task에는 지도 파일(map.yaml)이 필요한데, TurtleBot3에 내장된 SLAM 기능을 활용하면 Task에 필요한 지도를 제작할 수 있기 때문에 Navigation Task에 적합하게 사용할 수 있다.

4. Final Purpose

로봇을 구동하는 시뮬레이터들은 물리 엔진, 환경, 도구 등에서 모두 차이를 가지고 있다. 같은 환경에서 같은 로봇으로 시뮬레이션을 구동해도 시뮬레이터에 따라 로봇들이 움직이는 방식이나 경로에는 차이가 발생할 수 있기 때문에, 동일한 환경 구성 요소에서 시뮬레이터만 변경하여 로봇의 동작과 경로 계획에 어떤 차이가 있는지 알아본다. 여기에 CLOiSim과 Gazebo를 이용할 예정이며, Gazebo에서 CLOiBot과 TurtleBot에 대해 Navigation을 적용해보고 궁극적으로 CLOiSim과의 차이점을 찾아 문서화한다. 본 문서에서는 Gazebo 환경에 CLOiBot, TurtleBot을 포팅하여 Navigation을 수행하는 과정까지의 내용을 다룬다.

Ⅱ. Navigation with Gazebo

1. Gazebo+CLOiBot

Gazebo 환경에 CLOiBot을 포팅한다. CLOiBot은 CLOiSim에서 구동될 수 있도록 LG에서 제작한 로봇으로, Gazebo 환경에서 SDF 파일을 불러오려면 파일 수정이 필요하다.

1.1. LG Seocho World

먼저 Gazebo, CLOiSim에서 Task 동작에 대해 동일하게 적용될 시뮬레이션 환경인 lg_seocho.world 월드 파일을 살펴보면 다음과 같다.

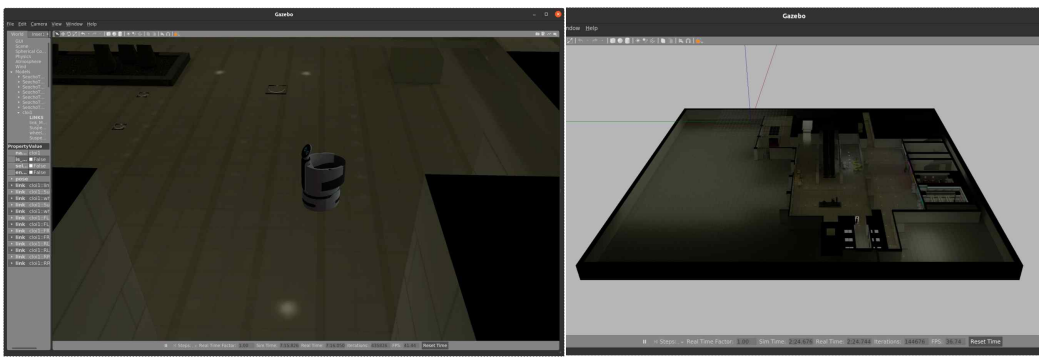
```
<?xml version="1.0" ?>
<sdf version="1.6">
<world name="lg_seocho_world">
  <gui>
    <camera name="user_camera">
      <pose>-47.37 -29.29 -4.5 0 0.523 0</pose>
    </camera>
  </gui>

  <spherical_coordinates>
    <surface_model>EARTH_WGS84</surface_model>
    <world_frame_orientation>ENU</world_frame_orientation>
    <latitude_deg>37.46105</latitude_deg>
    <longitude_deg>127.03418</longitude_deg>
    <elevation>40</elevation>
    <heading_deg>-47</heading_deg>
  </spherical_coordinates>

  <light name='sun' type='directional'>
    <cast_shadows>1</cast_shadows>
    <pose>0 0 500 0 -0 0</pose>
    <diffuse>0.9528302 0.9528302 0.9528302 1</diffuse>
    <direction>0 0 -1</direction>
  </light>

  <include>
    <uri>model://map_lg_seocho</uri>
  </include>

  <include>
    <name>cloi1</name>
    <uri>model://CLOi_porter</uri>
    <pose>-43.81 -28.77 -6.546 0 0 3.14</pose>
  </include>
</world>
</sdf>
```

전체 파일은 SDF 형식으로 작성되어 있고, 광원(sun), 월드(map_lg_seocho), 로봇(cloi1)을 include하여 불러온다. 본 파일을 그대로 사용할 경우 시뮬레이터 자체의 차이로 인해 월드 파일을 읽을 수 없는 오류가 발생하여 아래와 같이 수정하였다.

```
<?xml version="1.0" ?>
<sdf version="1.6">
<world name="lg_seocho_world">
  <gui>
    <camera name="user_camera">
      <pose>-47.37 -29.29 -4.5 0 0.523 0</pose>
    </camera>
  </gui>

  <spherical_coordinates>
    <surface_model>EARTH_WGS84</surface_model>
    <world_frame_orientation>ENU</world_frame_orientation>
    <latitude_deg>37.46105</latitude_deg>
    <longitude_deg>127.03418</longitude_deg>
    <elevation>40</elevation>
    <heading_deg>-47</heading_deg>
  </spherical_coordinates>

  <include>
    <uri>model://map_lg_seocho</uri>
  </include>

  <include>
    <name>cloi1</name>
    <uri>model://CLOi_porter</uri>
    <pose>-43.81 -28.77 -6.546 0 0 3.14</pose>
  </include>
</world>
</sdf>
```

본 파일에서 광원에 대한 부분을 삭제하고, 연계된 파일(map_lg_seocho 등)에서도 같은 부분을 삭제한 뒤 명령어를 통해 Gazebo 시뮬레이터를 실행하였다.

```
$ export GAZEBO_MODEL_PATH="<full_path>/models"
$ cd <lg_seocho.world path>
$ gazebo lg_seocho.world
```



1.2. CLOi Porter

Gazebo에서 LG 서초 사옥과 CLOiBot을 불러오는 데에는 문제가 없지만, 로봇 종속 플러그인의 호환 문제로 인해 CLOiBot의 구동을 담당하는 아래 6개의 바퀴 중 서브 바퀴 4개에서 강한 흔들림이 발생하였고, ROS Teleop을 실행한 결과 ROS로 로봇을 조작할 수 없었다. 로봇의 SDF파일이 위치한 CLOi_Porter 디렉터리를 살펴본 결과,

```
<include>
  <uri>model://CLOi_porter_caster</uri>
  <name>FL_caster</name>
  <pose>0.22 0.1 0.066 0 0 0</pose>
</include>
<joint name='link_MainBody_Caster_FL' type='revolute'>
  <parent>link_MainBody</parent>
  <child>FL_caster::link</child>
  <axis>
    <xyz>0 0 1</xyz>
    <dynamics>
      <friction>0.3</friction>
    </dynamics>
  </axis>
</joint>

<include>
  <uri>model://CLOi_porter_caster</uri>
  <name>FR_caster</name>
  <pose>0.22 -0.1 0.066 0 0 0</pose>
</include>
<joint name='link_MainBody_Caster_FR' type='revolute'>
  <parent>link_MainBody</parent>
  <child>FR_caster::link</child>
  <axis>
    <xyz>0 0 1</xyz>
    <dynamics>
      <friction>0.3</friction>
    </dynamics>
  </axis>
</joint>

<include>
  <uri>model://CLOi_porter_caster</uri>
  <name>RL_caster</name>
  <pose>-0.220 0.1 0.066 0 0 0</pose>
</include>
<joint name='link_MainBody_Caster_RL' type='revolute'>
  <parent>link_MainBody</parent>
  <child>RL_caster::link</child>
  <axis>
    <xyz>0 0 1</xyz>
    <dynamics>
      <friction>0.3</friction>
    </dynamics>
```

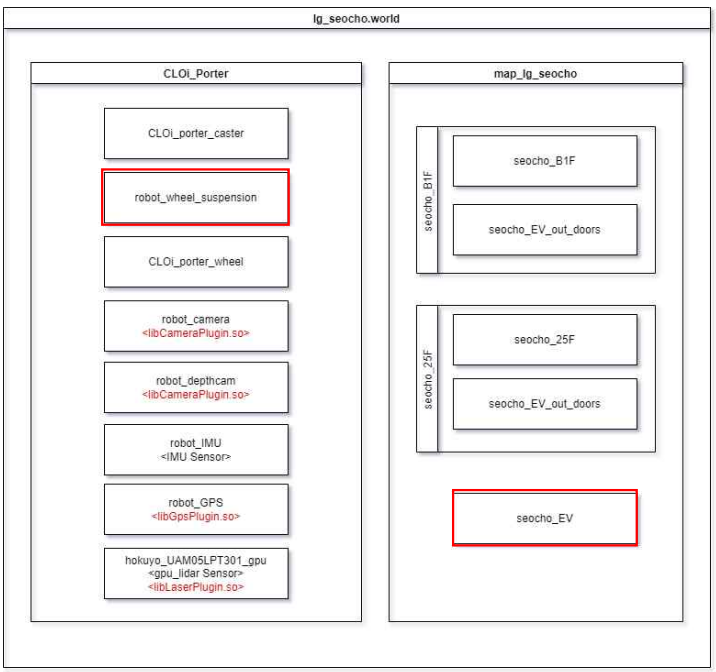
```
    </axis>
  </joint>

  <include>
    <uri>model://CLOi_porter_caster</uri>
    <name>RR_caster</name>
    <pose>-0.220 -0.1 0.066 0 0 0</pose>
  </include>
  <joint name='link_MainBody_Caster_RR' type='revolute'>
    <parent>link_MainBody</parent>
    <child>RR_caster::link</child>
    <axis>
      <xyz>0 0 1</xyz>
      <dynamics>
        <friction>0.3</friction>
      </dynamics>
    </axis>
  </joint>
```

link_MainBody_Caster_RR/RL/FR/FL 4개의 바퀴에 연결된 부분에 대한 오류로 추정되었다. 또한 Gazebo가 별도의 에러 메시지 출력 없이 비정상 종료되거나, Vector 문제로 시뮬레이터가 켜지지 않는 현상이 발생하였다.

gzclient: /build/ogre-1.9-1.9.0+dfsg1/OgreMain/include/OgreAxisAlignedBox.h:252: void Ogre::AxisAlignedBox::setExtents(const Ogre::Vector3&, const Ogre::Vector3&): Assertion `(min.x <= max.x && min.y <= max.y && min.z <= max.z) && "The minimum corner of the box must be less than or equal to maximum corner"' failed.

Error Message: Gazebo OGRE Vector

오류를 해결하기 위해 여러 파일을 수정하였고, 이유를 찾긴 하였으나 Gazebo에서 CLOiBot을 사용하기는 어렵다고 결론지었다.



CLOiBot은 Task를 위해 여러 플러그인 등으로 구성되어 있는데, 왼쪽 다이어그램에 빨간 네모로 표시한 부분이 Gazebo와 호환이 되지 않아 사용이 불가하다. wheel(로봇 바퀴)에는 Micom, EV(엘리베이터)에는 Elevator 시스템이 필요한데 Gazebo에는 내장되어 있지 않아 사용이 불가하다. 로봇의 바퀴 4개가 떨리는 현상은 Micom 플러그인이 호

환되지 않아서 발생하는 문제로 추정되며, EV 시스템의 부재로 Navigation 작업 중 일부가 동작하지 않을 것으로 예상된다. 문제가 발생하는 구문을 *CLOi_Porter/model.sdf* 파일에서 삭제한 후 Gazebo로 월드를 다시 불러오면 정상적으로 실행은 가능하다.

```
<plugin name='RobotControl' filename='libMicomPlugin.so'>
  <wheel>
    <PID>
      <kp>1.0</kp>
      <ki>0.05</ki>
      <kd>0.0</kd>
    </PID>
    <tread>0.449</tread>
    <radius>0.0955</radius>
    <location type="left">wheel_left</location>
    <location type="right">wheel_right</location>
    <friction>
      <motor>0.06</motor>
      <brake>13.0</brake>
    </friction>
  </wheel>
  <update_rate>20</update_rate>
  <ros2>
    <static_transforms>
      <link parent_frame_id="base_link">link_imu</link>
      <link parent_frame_id="base_link">SuspensionLeft::link</link>
      <link parent_frame_id="base_link">SuspensionRight::link</link>
    </static_transforms>
    <transforms>
      <link parent_frame_id="SuspensionLeft::link">wheel_left::link</link>
      <link parent_frame_id="SuspensionRight::link">wheel_right::link</link>
    </transforms>
  </ros2>
</plugin>
```

삭제 또는 주석 처리 (CLOi_Porter/model.sdf)

```
<plugin name="ElevatorSystem" filename="libElevatorSystem.so">
  <system_name>ElevatorSystem_00</system_name>
  <elevator prefix_name="Elevator_" speed="30">
    <floor>floor_collision</floor>
    <doors speed="0.6" closing_timer="7.0">
      <inside open_offset="0.567">
        <door name="left">seocho_EV_door_L_link</door>
        <door name="right">seocho_EV_door_R_link</door>
      </inside>
      <outside open_offset="0.567">
        <door name="left">seocho_EV_out_doors_L_link</door>
        <door name="right">seocho_EV_out_doors_R_link</door>
      </outside>
```

```
        </doors>
      </elevator>
      <floors>
        <floor>
          <name>B1F</name>
          <height>-6.61037</height>
        </floor>
        <floor>
          <name>25F</name>
          <height>150.13362</height>
        </floor>
      </floors>
    </plugin>
```

삭제 또는 주석 처리 (map_lg_seocho/model.sdf)

2. Gazebo+TurtleBot3

1절의 Gazebo+CLOiBot에서 CLOiBot은 바퀴의 떨림 문제, 플러그인 호환 불가 문제로 인해 사용할 수 없다고 판단되어 Gazebo에서 가장 보편적으로 사용되는 TurtleBot을 Seocho World 위에 올려 시뮬레이터를 동작시켜 본다. TurtleBot3에는 Burger, Waffle 두 모델이 존재하며 두 모델의 주요 차이점은 액추에이터, SBC, 센서이다. 사실 가상환경에서 사용하기에는 큰 차이점이 존재하지는 않지만, Waffle 모델은 RVIZ를 별도로 켜지 않아도 Gazebo에서 전면 카메라를 확인할 수 있어 Burger 모델에 비해 편리하다.
먼저, LG 서초 사옥 월드에 터틀봇을 포팅하기 위해 월드 파일을 수정해야 한다.

```
<?xml version="1.0" ?>
<sdf version="1.6">
<world name="lg_seocho_world">
  <gui>
    <camera name="user_camera">
      <pose>-47.37 -29.29 -4.5 0 0.523 0</pose>
    </camera>
    <view_controller>orbit</view_controller>
    <projection_type>perspective</projection_type>
  </gui>

  <include>
    <uri>model://sun</uri>
  </include>

  <spherical_coordinates>
    <surface_model>EARTH_WGS84</surface_model>
    <world_frame_orientation>ENU</world_frame_orientation>
    <latitude_deg>37.46105</latitude_deg>
    <longitude_deg>127.03418</longitude_deg>
    <elevation>40</elevation>
```

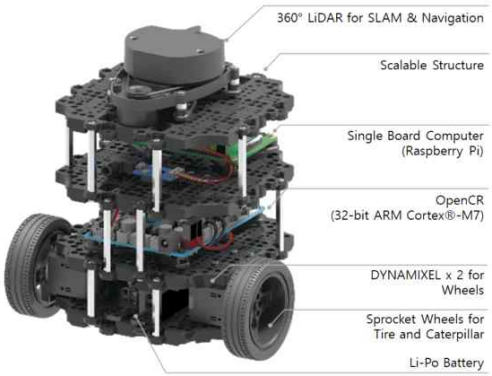
```
      <heading_deg>-47</heading_deg>
    </spherical_coordinates>
    <physics type="ode">
      <real_time_update_rate>1000.0</real_time_update_rate>
      <max_step_size>0.001</max_step_size>
      <real_time_factor>1</real_time_factor>
      <ode>
        <solver>
          <type>quick</type>
          <iters>150</iters>
          <precon_iters>0</precon_iters>
          <sor>1.400000</sor>
          <use_dynamic_moi_rescaling>1</use_dynamic_moi_rescaling>
        </solver>
        <constraints>
          <cfm>0.00001</cfm>
          <erp>0.2</erp>
          <contact_max_correcting_vel>2000.000000</contact_max_correcting_vel>
          <contact_surface_layer>0.01000</contact_surface_layer>
        </constraints>
      </ode>
    </physics>

    <include>
      <uri>model://map_lg_seocho</uri>
    </include>

    <include>
      <name>turtlebot</name>
      <uri>model://<turtlebot_name></uri>
      <pose>-43.81 -28.77 -6.546 0 0 3.14</pose>
    </include>
  </world>
  </sdf>
```

worlds/lg_seocho.world

사용할 Gazebo 환경에 맞게 기타 옵션을 설정해주고, 광원은 1절에서 삭제하였었으나 환경이 어두워 시뮬레이션을 확인하기 어려운 점이 있었으므로 Gazebo에서 제공하는 내장 광원으로 변경하여 적용해주었다. CLOiBot의 include 구문은 TurtleBot으로 변경하고, 원하는 위치에 pose를 지정하면 LG 사옥에 터틀봇을 스폰시킬 수 있다.
Gazebo 내장 광원을 사용하기 위해서는 1절에서 export한 Gazebo model path를 변경해야 한다. *(관련 패키지는 install 되어 있다는 상태로 가정한다.)* 다음으로, Gazebo 시뮬레이터에 사용할 TurtleBot 모델을 지정해주면 초기 세팅은 끝이다.

```
$ export TURTLEBOT3_MODEL=<turtlebot_model_name>
$ export GAZEBO_MODEL_PATH="<full_path>/models"
$ source ~/turtlebot3_ws/install/setup.bash
```

2.1. Burger Model



최대 속도	0.22 m/s, 2.84 rad/s
크기	138 * 178 * 192(mm)
무게	1kg
카메라	X
액추에이터	XL430-W250
컨트롤러	X
최대 하중	15kg

Burger 모델을 사용하여 Gazebo 환경을 세팅하기 위해 export한다.

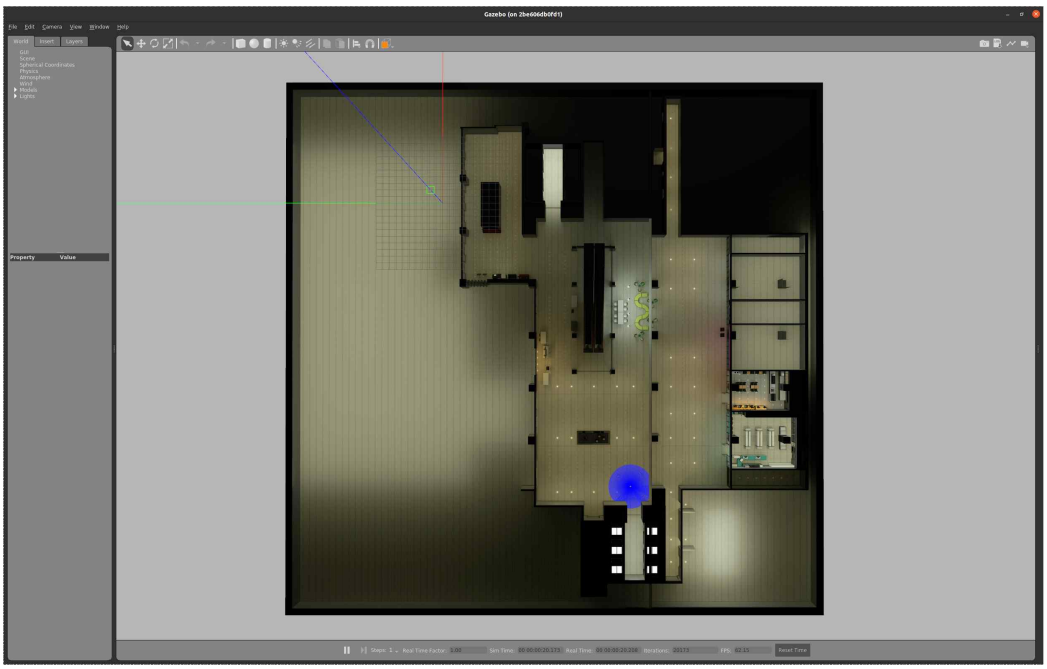
```
$ export TURTLEBOT3_MODEL=burger
```

월드 파일(lg_seocho.world)의 아래 include 구문을 Burger 구문에 맞게 수정한다.

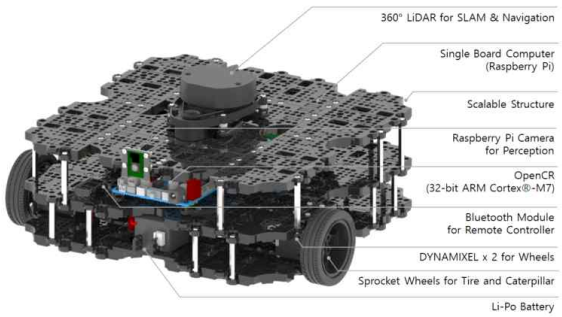
```
<include>
  <name>turtlebot</name>
  <uri>model://turtlebot3_burger</uri>
  <pose>-43.81 -28.77 -6.546 0 0 3.14</pose>
</include>
```

Launch 파일을 실행하여 Gazebo 월드를 불러온다. 푸른색 원은 Burger 모델의 탐색 가능 범위를 나타내며, 진한 원은 장애물이 인식됐을 때 장애물까지의 시야각이다.

```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```



2.2. Waffle Model



최대 속도	0.26 m/s, 1.82 rad/s
크기	281 * 306 * 141(mm)
무게	1.8kg
카메라	O
액추에이터	XM430-W210
컨트롤러	O (BLE)
최대 하중	30kg

Waffle 모델을 사용하여 Gazebo 환경을 세팅하기 위해 export한다.

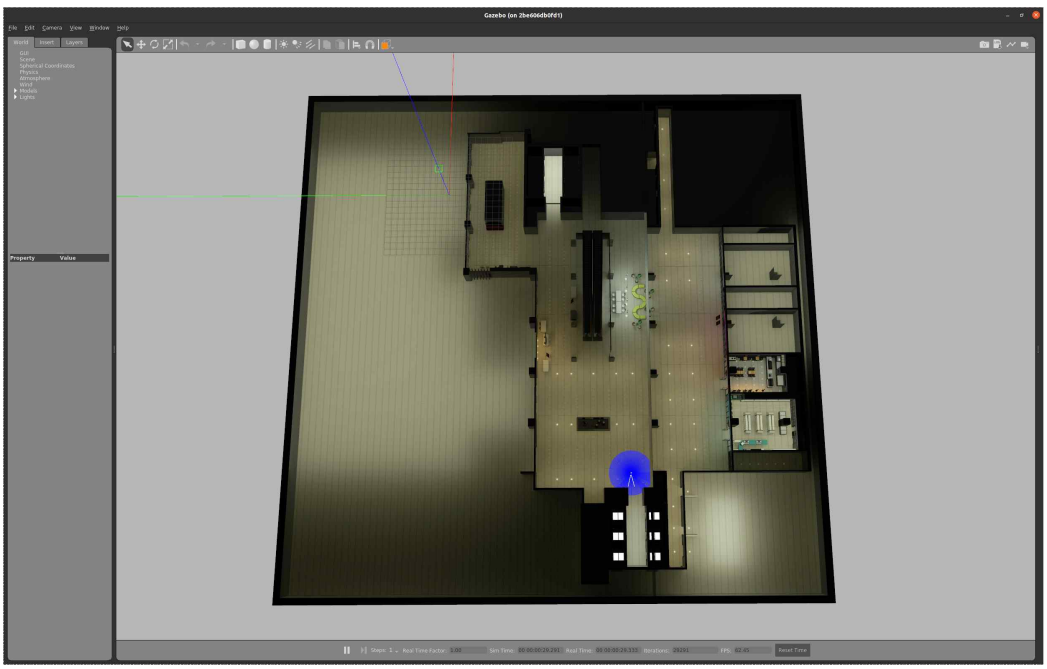
```
$ export TURTLEBOT3_MODEL=waffle
```

월드 파일(lg_seocho.world)의 아래 include 구문을 Waffle 구문에 맞게 수정한다.

```
<include>
  <name>turtlebot</name>
  <uri>model://turtlebot3_waffle</uri>
  <pose>-43.81 -28.77 -6.546 0 0 3.14</pose>
</include>
```

Launch 파일을 실행하여 Gazebo 월드를 불러온다. 푸른색 원은 Waffle 모델의 탐색 가능 범위를 나타내며, 진한 원은 장애물이 인식됐을 때 장애물까지의 시야각이다.

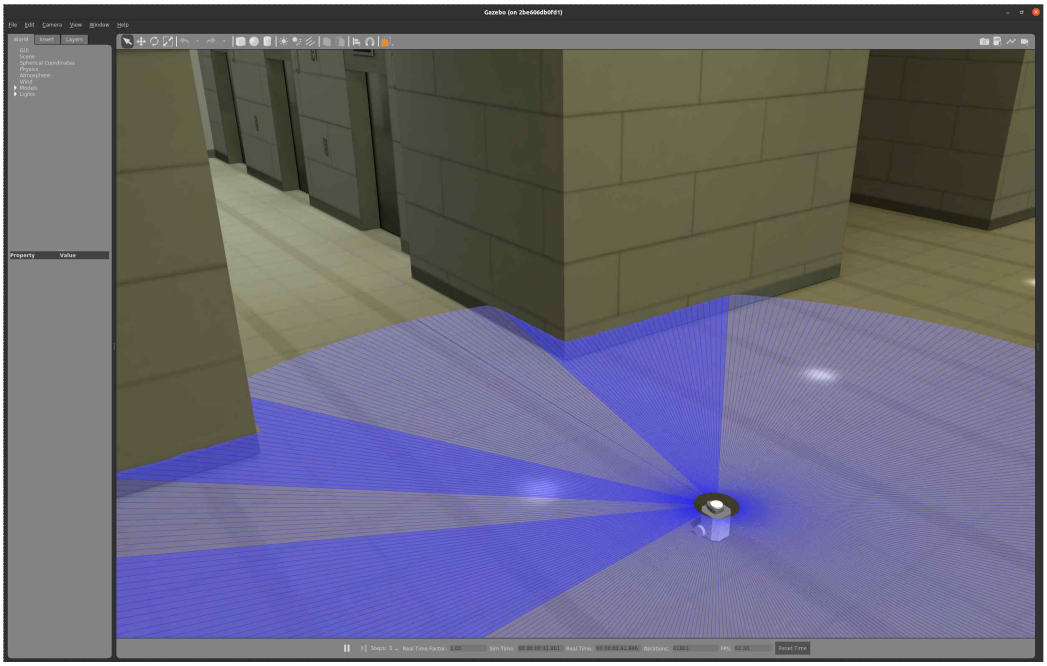
```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```



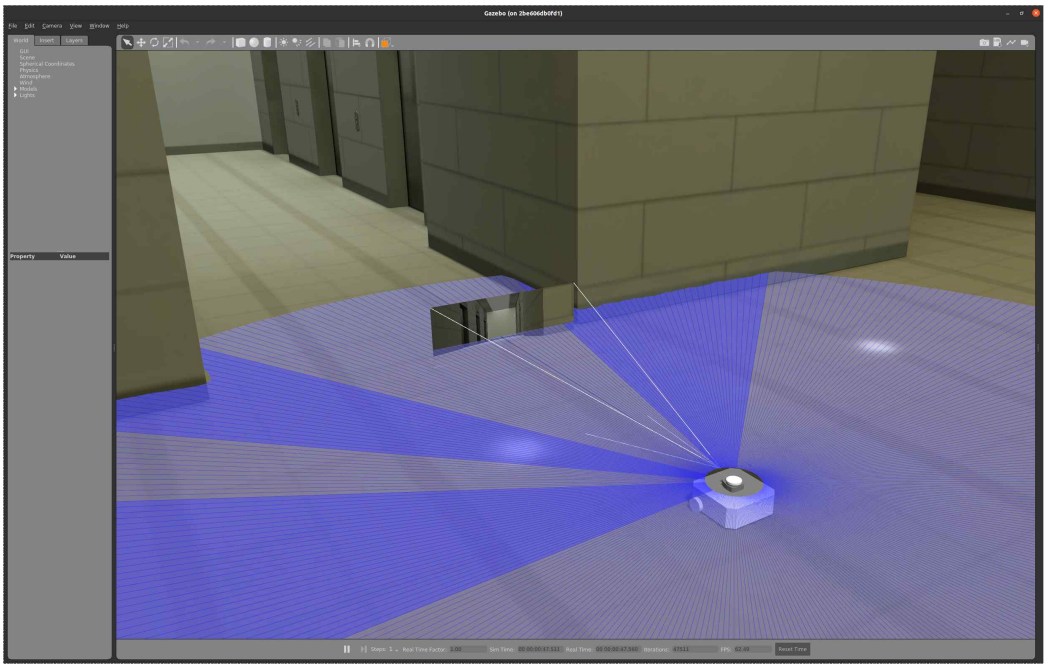
2.3. Burger vs Waffle

Burger 모델과 Waffle 모델을 조금 더 확대해서 보면, Waffle 모델에는 전면 카메라가 부착되어 있어 Gazebo에서 카메라를 확인할 수 있다. RVIZ에서 Image Topic을 Subscribe하면 영상 확인이 가능하긴 하지만, Waffle이 더 우세하다 볼 수 있다.

▶ Burger Model



▶ Waffle Model

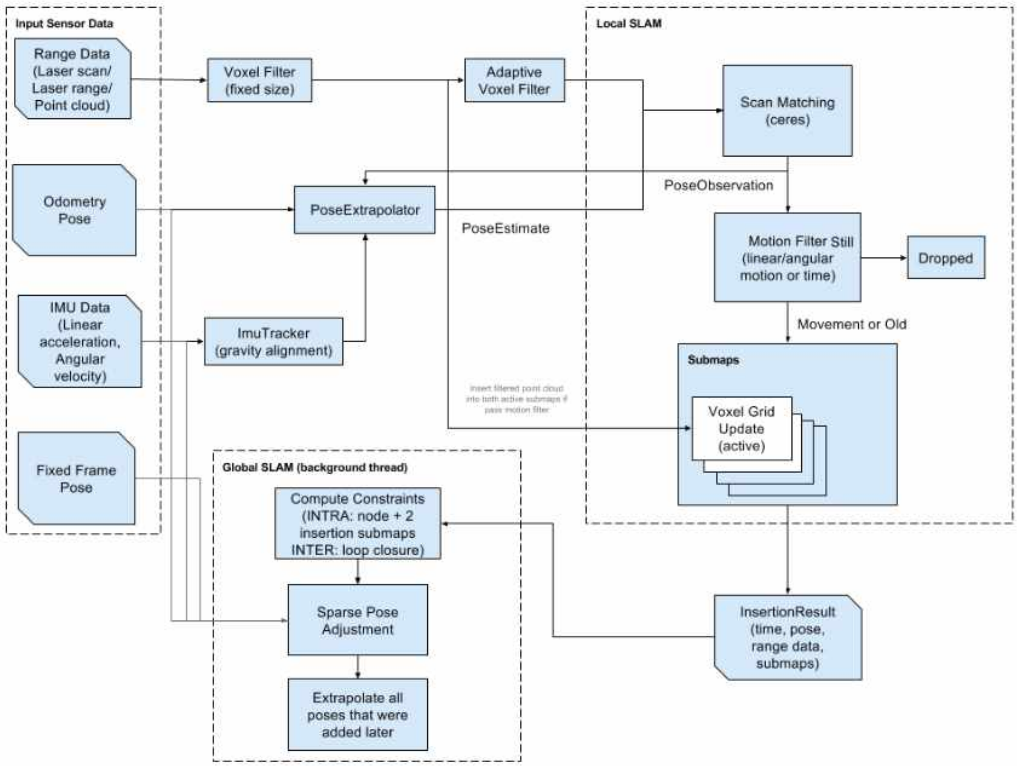


※ 사진에서 보이는 바와 같이, 모델의 렌더링이 되지 않는다. 파일을 여러 번 수정해 보았지만, 세부 디자인이 표현되지 않고 단순 흰색으로 출력되는 문제가 있다. 하지만 주행, 탐색 등의 작업에는 문제가 발생하지 않아 Navigation 수행이 가능하다.

3. SLAM Mapping

3.1. Cartographer

Cartographer는 직역하면 '지도 제작자'로, 시뮬레이션 환경에서 로봇을 이용하여 지도(map) 파일을 제작할 수 있게 만들어주는 SLAM 라이브러리이다. 다양한 플랫폼과 센서(LiDAR, Camera Vision 등) 구성에서 2D, 3D 환경의 SLAM을 실시간으로 제공한다. Cartographer는 독립적인 C++ 라이브러리로 ROS를 위한 바이너리 패키지를 제공하기 때문에 apt 명령을 이용하여 간단히 설치가 가능하다.



SLAM 작업은 Gmapping을 이용할 수도 있는데, Gmapping보다 Cartographer가 지도의 생성이 더 빠르고 부드럽다. Cartographer를 사용하기 위해 관련 패키지를 설치한다. Navigation 패키지가 없다면 함께 설치해주도록 한다.

```
$ apt install ros-foxy-gazebo-ros-pkgs ros-foxy-cartographer ros-foxy-cartographer-ros
ros-foxy-navigation2 ros-foxy-nav2-bringup
```

TurtleBot3을 사용할 것이므로, Update Map Topic을 Cartographer가 Subscribe할 수 있도록 기타 확장 패키지를 추가로 설치해준다.

```
$ apt install ros-foxy-turtlebot3-msgs ros-foxy-dynamixel-sdk ros-foxy-hls-lfcd-lds-driver
```

TurtleBot3의 Cartographer는 로보틱스 Git에 레포지토리가 공개되어 있다. 워크스페이스를 생성하고 Clone한 후 빌드해주면 쉽게 사용이 가능하다.

```
$ mkdir -p ~/turtlebot_ws/src
$ cd ~/turtlebot3_ws/src
$ git clone -b foxy-devel https://github.com/ROBOTIS_GIT/turtlebot3
```

```
$ git clone -b foxy-devel https://github.com/ROBOTIS-GIT/turtlebot3_simulations
$ colcon build --symlink-install --parallel-workers 1
```

colcon build를 수행하면 워크스페이스 아래의 src 파일 내의 install 가능한 패키지가 모두 설치된다. 패키지를 사용할 수 있게 되었으므로, export 해주자.

```
$ export TURTLEBOT3_MODEL=waffle
$ export GAZEBO_MODEL_PATH="~/turtlebot3_ws/src/turtlebot3_simulations/turtlebot3_
  gazebo/models"
```

2절에 Burger, Waffle 모델을 포팅하여 환경을 실행했을 때처럼 Launch한다.

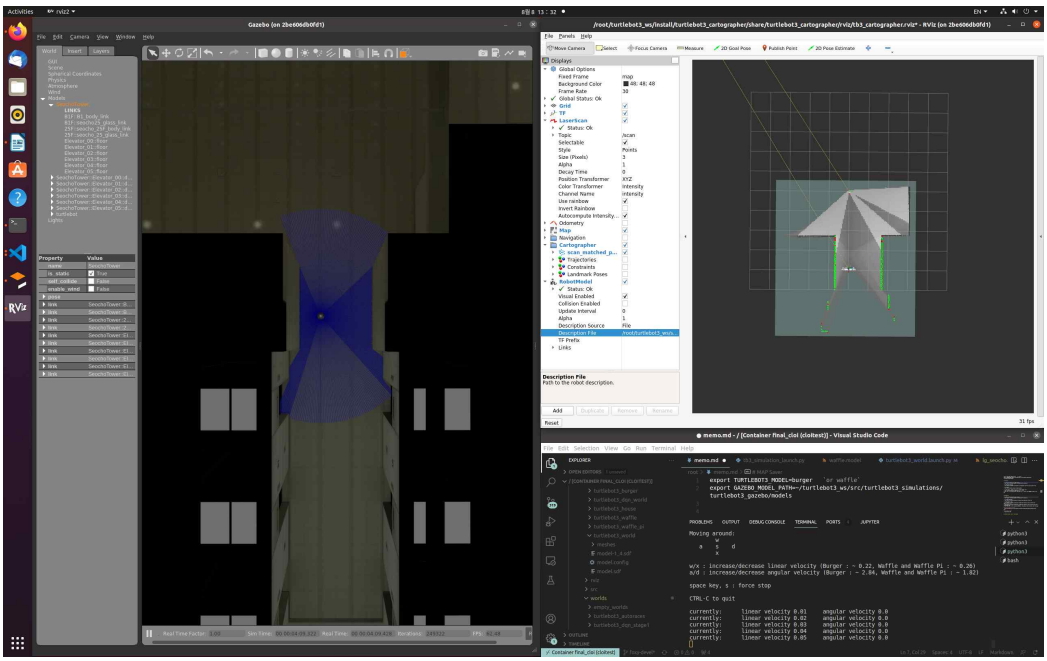
```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

Gazebo가 켜지면, Cartographer를 실행하여 SLAM 작업을 수행한다.

```
$ ros2 launch turtlebot3_cartographer cartographer.launch.py use_sim_time:=True
```

Cartographer는 단순한 지도 제작 프로세스이므로, Teleop과 함께 사용해야 한다.

```
$ ros2 run turtlebot3_teleop teleop_keyboard
```



(좌) Gazebo / (우) RVIZ+Cartographer, Teleop

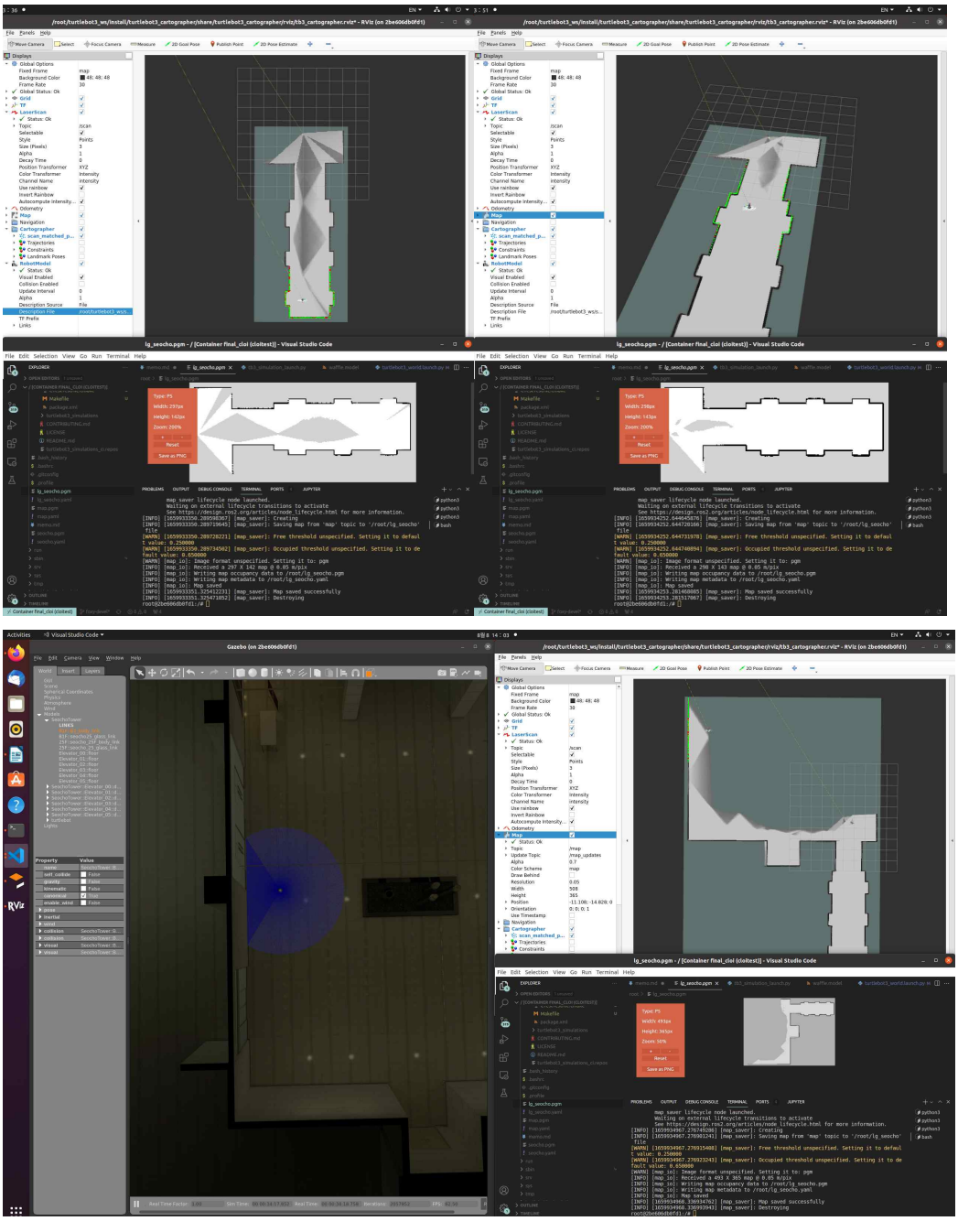
좌측 Gazebo 화면에서 짙은 파란색으로 표시된 부채꼴이 로봇이 탐색 중인 범위이고, 부채꼴 모양으로 시야각이 퍼지다 장애물을 만나게 되는 접점이 우측 RVIZ에 초록색 점으로 표시된다. 점을 기준으로 탐색한 범위는 투명>짙은회색>회색>옅은회색 순서로 점차 바뀌며, 색이 연할수록 완벽하게 탐색된 지역을 의미한다. Teleop을 통해 로봇의 속도를 적당히 조절하며 맵을 탐색하다보면 완성된 지도를 만날 수 있다.

3.2. Map Server(Saver)

Cartographer에서 생성한 지도는 고정된 지도(pgm 형식)로 저장하거나, 확장 가능한 지도(pbstream)로 저장할 수 있다. Cartographer Navigation을 사용하기 위해서는 후자의 pbstream으로 저장해야 하지만 Navigation2에서는 pgm+yaml 형식만으로도 충분하므로 고정된 지도를 제작할 것이다.

```
$ ros2 run nav2_map_server map_saver_cli -f <path>
```

지도를 충분히 제작했다고 판단이 되면, Map Server를 구동한다. 나의 경우 맵이 완벽하지는 않아도 중간 세이브를 위해 자주 저장해주었다. 탐색 작업을 충실히 수행할수록 아래와 같이 흰색 영역으로 채워지게 된다.



TurtleBot3 모델의 최대 선속도(linear velocity) Burger 0.22, Waffle 0.26이고 최대 각속도(angular velocity)는 Burger 2.84, Waffle 1.82이다. 음수로도 같은 값을 가지는데, 최대 속도가 높다 하여 지도 제작에 높은 속도를 이용하면 좌표계가 쉽게 틀어진다. 이유로 추정되는 것은 Cartographer가 지도를 실시간으로 제작하긴 하지만 영상의 형태가 아닌 Topic으로 Update Map을 받아오기 때문에 빠른 속도로 로봇이 이동할 때 Gazebo(로봇의 실제 움직임)과 RVIZ(받아온 토픽)의 Frame에 차이가 발생하면 로봇의 현재 좌표와 시야가 제대로 인식되지 않을 수 있다는 것이다. 지도를 약 이틀간 제작해보았는데, 바깥쪽 90도(270도로 회전하는) 벽면에서 좌표가 자주 틀어지고, 직각 벽을 인식할 때 직각이 아닌 어느정도 벗어난 둔각(비스듬한 벽)으로 인식하는 과정이 반복되어 전체 지도가 깨지는 현상이 발생했다.

4. Navigation2

3.1. Bringup

TurtleBot3 Navigation2의 Bringup을 실행해본다. 먼저 Gazebo부터 Launch한다.

```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

TurtleBot3의 상태를 주기적으로 Publish해주는 Publisher 프로세스를 실행한다.

```
$ ros2 launch turtlebot3_bringup turtlebot3_state_publisher.launch.py
```

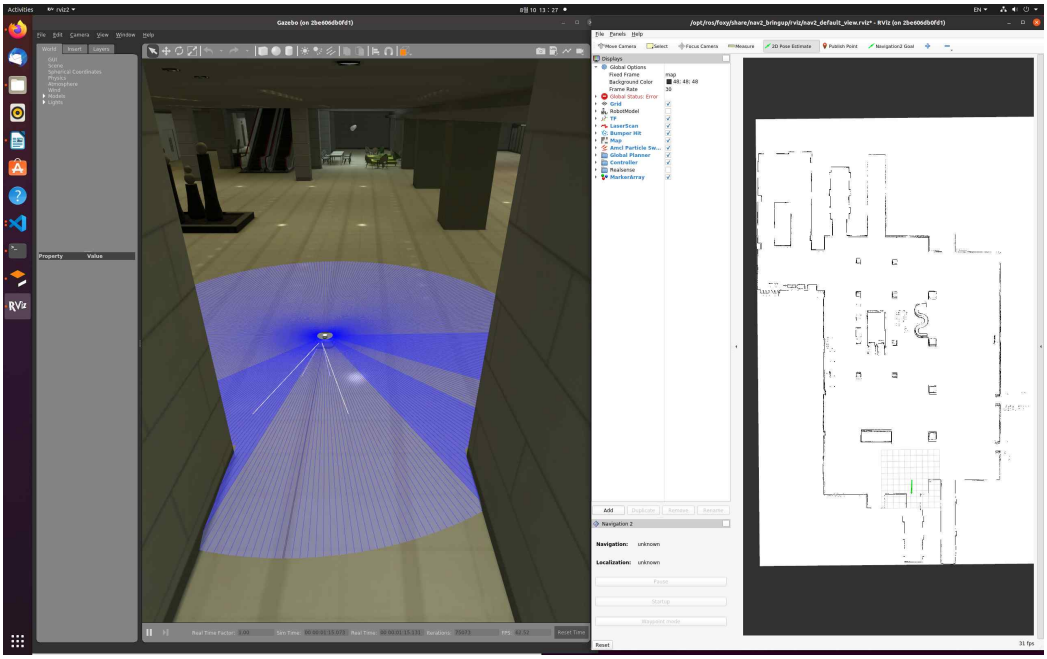
Navigation Task를 수행시키기 위해 Launch File을 실행한다. 본 Task는 이미 탐색된 지역에 대해 목표(Goal) 좌표를 설정하여 로봇이 이동할 수 있으므로, 탐색된 지역과 현재 지역의 장애물 등의 정보를 갖는 지도(map) 파일을 이용해야 한다. 여기에 사용될 지도가 바로 3절 SLAM 작업의 결과물이다.

```
$ ros2 launch turtlebot3_navigation2 navigation2.launch.py use_sim_time:=True
  map:=<path/map.yaml>
```

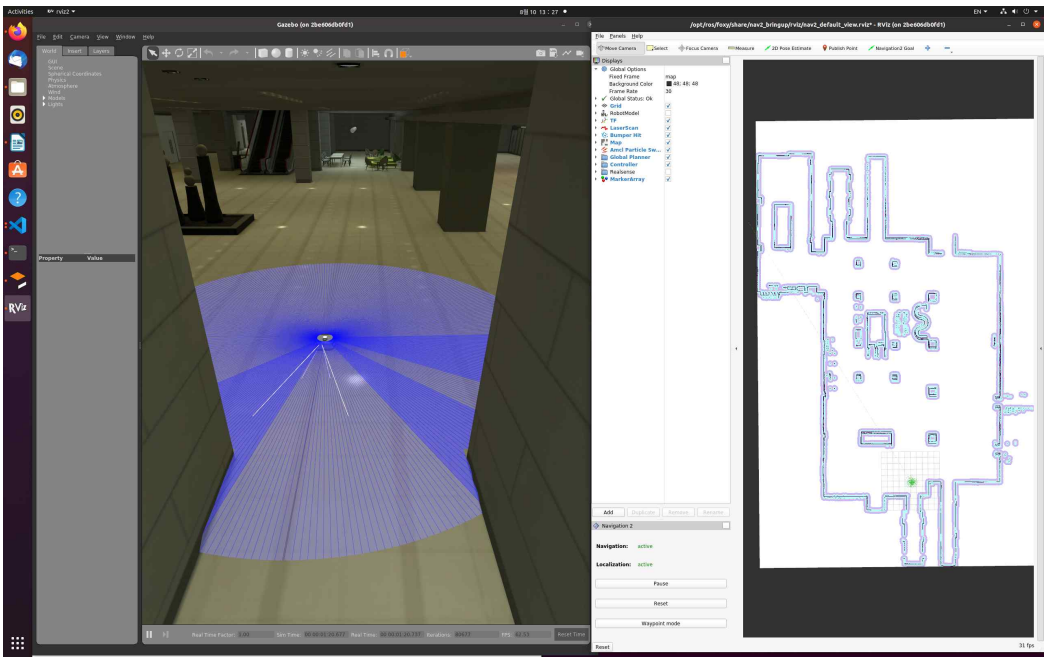
지도(map) 파일의 경로를 지정하여 Navigation2를 Launch하면 RVIZ가 켜진다. 좌측에 Gazebo, 우측에 RVIZ를 세팅하고 다음과 같은 순서로 조작하면 TurtleBot3으로 Navigation2를 수행할 수 있다.

1) 2D Pose Estimate

: 로봇의 초기 좌표를 지정한다. 시작점을 기준으로 바라보는 방향으로 드래그하면 화살표 모양으로 Pose를 지정할 수 있다. 초기 좌표 설정이 이상할 경우, 목표 지점까지의 Navigation 수행이 원활하지 않을 수 있다.

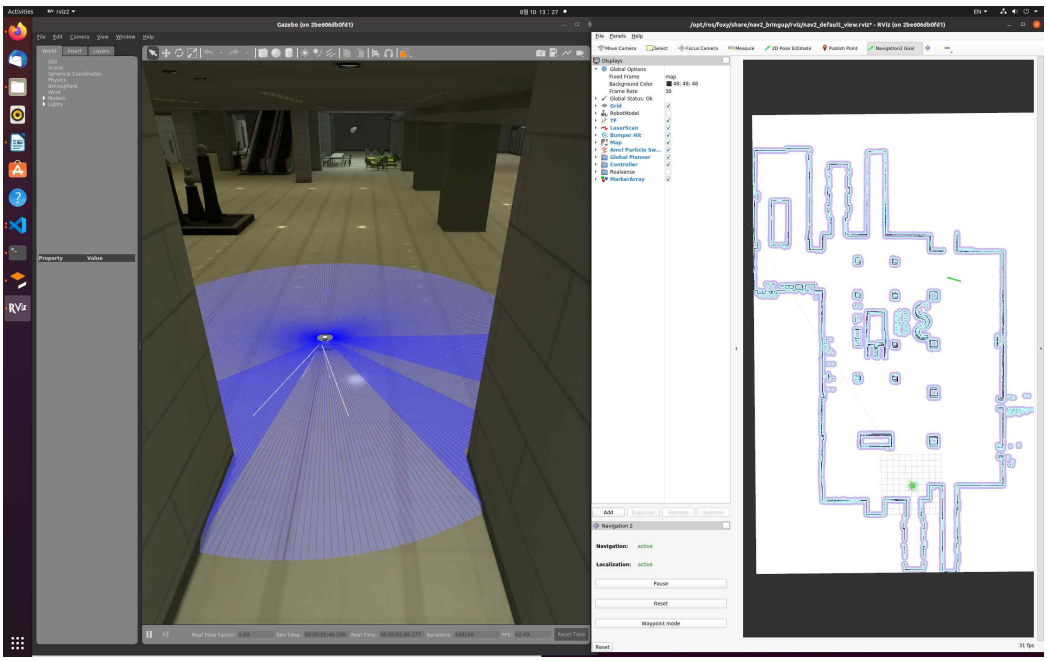


2D Pose가 지정되면 지도가 활성화되고 탐색 작업을 수행할 수 있다.

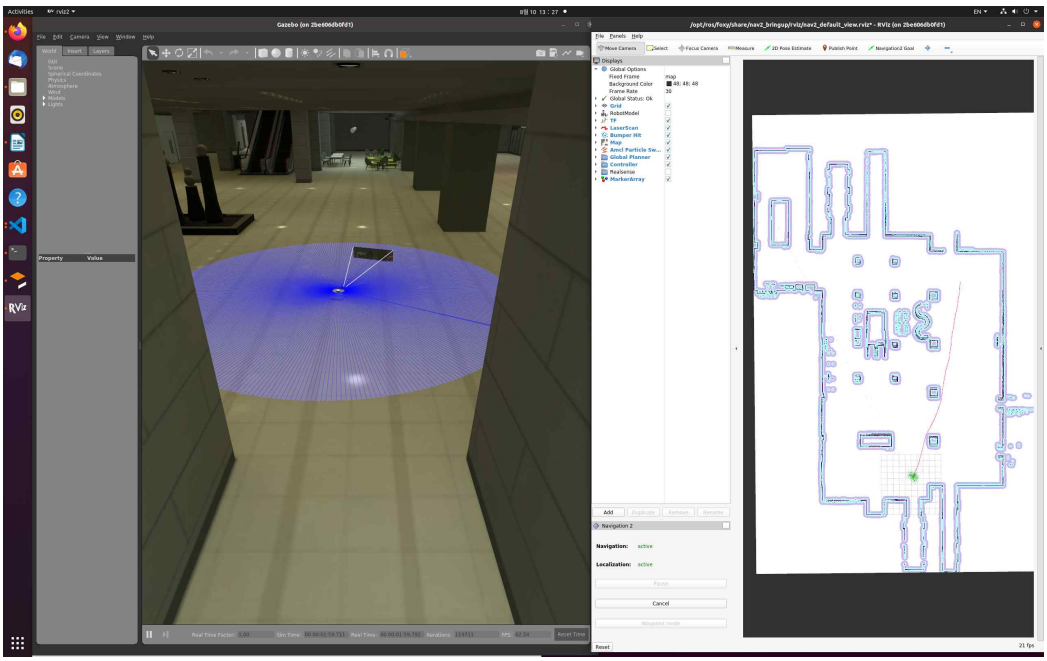


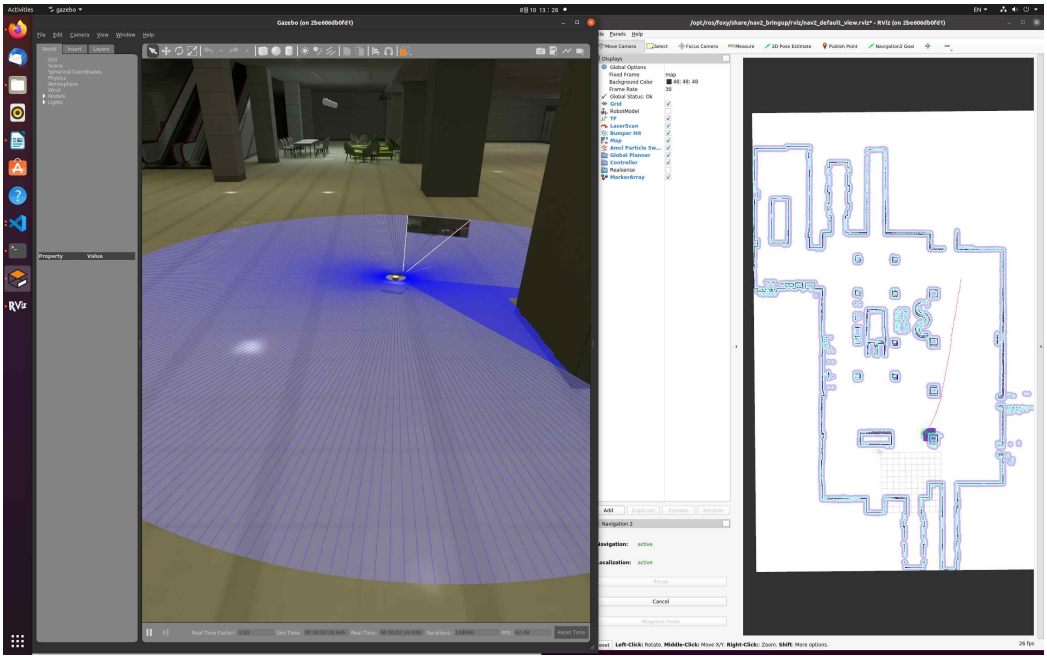
2) Navigation2 Goal

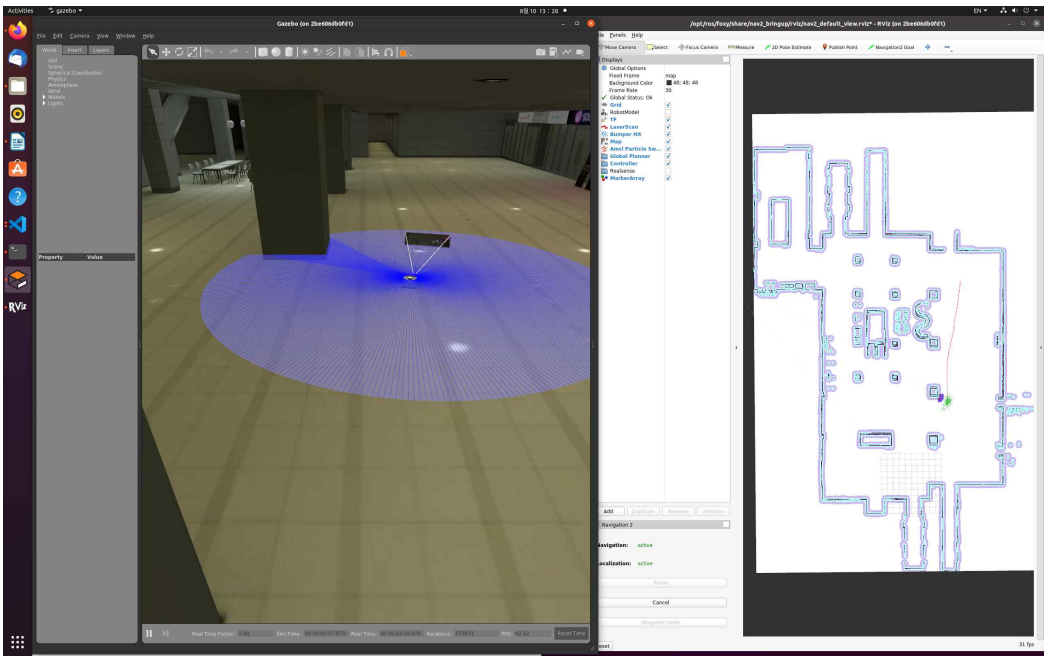
상단의 Navigation2 Goal 버튼을 클릭하여 목표 좌표를 설정한다. 목표 좌표는 사용자가 로봇을 보내고 싶은 위치와 방향으로, 2D Pose의 지정과 같은 방식으로 드래그하면 로봇이 해당 위치까지 지도를 기반으로 탐사하며 이동한다.



목표 지점과 반대 방향을 바라보도록 스폰된 로봇은 목표 좌표를 입력받은 후 360도 회전하여 목표 좌표까지 주행하는 모습이다.







Ⅲ. 결론

1. Defect & Warning

해결한 오류는 제외하고, 현재 해결되지 않은 오류만 나열하였다.

1) Gazebo 환경에서 TurtleBot3을 불러올 때, LG 서초 사옥과 함께 사용하며 로봇의 렌더링이 되지 않는 문제가 있다. → CLOiSim Model, World를 사용하지 않고 터틀봇에서 제공하는 공식 Model, World를 사용했을 때는 모델이 정상적으로 불러와진다. 하지만 렌더링만 되지 않을 뿐, 탐색 작업이나 Teleop에는 문제가 없어 그냥 사용하였다.

2) CLOiSim, CLOiSim ROS, Sample Resources, Gazebo11, TurtleBot3 모두 한 Docker에 Install할 수 있도록 DockerFile을 만들어 사용하였는데, CLOiSim, TurtleBot3 모두 SDF 형식을 지원하고 Sample 모델도 SDF로 작성되어 있어 그대로 사용하였으나 형식 에러가 발생한다. 완벽한 SDF 형식이 아니어서 Warning이 발생하는 것 같은데, 이것 또한 실행 자체에는 문제가 없기 때문에 무시하고 Navigation을 수행하였다.

```
[gzserver-1] Warning [parser.cc:833] XML Attribute[version] in element[sdf] not defined in
SDF, ignoring.
[gzserver-1] Warning [parser.cc:833] XML Attribute[version] in element[sdf] not defined in
SDF, ignoring.
[gzserver-1] Warning [parser.cc:833] XML Attribute[version] in element[sdf] not defined in
SDF, ignoring.
[gzserver-1] Warning [parser.cc:833] XML Attribute[version] in element[sdf] not defined in
SDF, ignoring.
[gzserver-1] Warning [parser.cc:833] XML Attribute[version] in element[sdf] not defined in
SDF, ignoring.
```

3) Gazebo 월드와 Publisher, RVIZ 등을 실행할 때 Port를 3개 정도 사용하는데, 가끔 Process Died 에러가 발생하며 Navigation 수행이 불가한 경우가 있다. 원인을 찾지는 못하였으나 모든 프로세스 또는 터미널을 종료한 뒤 다시 실행하면 정상 작동이 가능하기에 크게 해결하려 하지 않았다.

2. 참고문헌

Navigation2

- https://navigation.ros.org/

Gazebo

- https://gazebosim.org/home

TurtleBot3

- TB: https://www.turtlebot.com/
- TB3: https://www.turtlebot.com/turtlebot3/
- TB Manual: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

SLAM

- Gmapping(참고): https://omorobot.com/docs/gmapping-%ec%82%ac%ec%9a%a9%ed%95%98%ec%97%ac-slam%ed%95%98%ea%b8%b0/
- Cartographer: https://omorobot.com/docs/cartographer-%EC%82%AC%EC%9A%A9%ED%95%98%EC%97%AC-slam%ED%95%98%EA%B8%B0/