

GOOD MORNING!

早上好!

안녕하세요!

DAY 3

DAY2 RECAP

DAY I

- Welcome
- Project Introduction
- Introduction to Project Development Process
- Business Requirement Development
- System Requirement Development
- System and Development environment Setup

DAY 2 (MINI PROJECT)

- Yolo 객체 인식 모델 활용과 성능 평가 방법 이해
 - Custom Dataset과 Fine Tuning으로 자체 객체 인식 모델 구현 및 평가
 - (Optional) 경량화 모델 등 개별 요구사항에 적합한 모델 탐색 및 성능 검증

DAY 2 (MINI PROJECT)

WEB-CAM 기반 객체 인식 (IF NEEDED)

- YOLOv8 기반 데이터 수집/학습/deploy (Detection Alert)
 - 감시용 데이터 수집(rc_car, dummy, 등)
 - 감시용 데이터 라벨링
 - YOLOv8 기반 학습
 - YOLOv8 Object Detection

AMR-CAM 기반 객체 인식

- AMR(Autonomous Mobile Robot) Turtlebot4 개발 환경 구축
- 로봇 개발 환경에 완성 모델 서빙 및 테스트 / 로봇 H/W, 제반 환경의 한계점 도출
 - Tracking 데이터 수집((rc_car, dummy, 등))
 - Tracking 데이터 라벨링
 - YOLOv8 기반 학습
 - YOLOv8 Object **Tracking**

DAY 3 (MINI PROJECT)

- Auto. Driving 시스템 학습
 - Digital Mapping of environment
 - Operate AMR (Sim. & Real)
 - Tutorial 실행
 - Detection, Depth and AMR 주행
 - 로봇 개발 환경에 적용 및 테스트 / 로봇 H/W, 제반 환경의 한계점 도출

TURTLEBOT4 시뮬레이션 DEMO

- SLAM과 AutoSLAM으로 맵 생성
- Sim.Tutorial 실행
- Detection, Depth and AMR 주행 example

DAY 3 (MINI PROJECT)

REAL ROBOT

- Manually operating the AMR (Teleops)
- autonomous driving 시스템 with obstacle avoidance
 - Digital Mapping of environment
 - Launching Localization, Nav2, and using Rviz to operate a robot
 - Goal Setting and Obstacle Avoidance using Navigation

TUTORIAL

- Turtlebot4 API를 활용한 Initial Pose Navigate_to Pose 구현
- Turtlebot4 API를 활용한 Navigate_Through_pose, Follow Waypoints 구현

DAY 4 (MINI PROJECT)

- System(High Level) Design (Mini Project)
 - System Architectural Diagram
- Detail Design to Acceptance - Agile Development (SPRINTs)
 - Detection
 - AMR Control

DAY 4 (MINI PROJECT)

CODING, TEST & INTEGRATION

- Coding and Test all modules
- Porting to ROS
- And finally, Integration and Test of Detection Alert & AMR Controller

MINI PROJECT DEMO

- Prepare and demo completed project

프로젝트 RULE NUMBER ONE!!!

Have Fun Fun Fun!



DID YOU ACHIEVE SAME **FOV** AND
DIMENSION FOR BOTH DEPTH AND
RGB??

HOMEWORK CHECK

- **Achieve aligned RGB & Depth FOV**
- Object Detection
 - Collect various datasets (i.e. different topics/images sizes)
 - Create various models (i.e. v5, v8, v11, etc; arg: Epoch, Batch, ImgSz, augmentation, etc)
 - Analyze the results
 - Determine using key metrics which model best fit your solution
 - Using .pt file to predict/inference on pc
 - **Successfully publish the annotated image topic**
- Depth
 - **Find and display the distance to the center of the detected objects**
- Update System Requirement

SYSTEM REQUIREMENT PRESENTATION BY EACH TEAM

Using the posted notes and flipchart as needed

KEY SUBSYSTEM (MODULES) TO DEVELOP

-
- | | |
|--|--|
| <ul style="list-style-type: none">• Detection Alert<ul style="list-style-type: none">• Camera Capture• Object Detection• Send messages to other subsystems | <ul style="list-style-type: none">• AMR Controller<ul style="list-style-type: none">• Receive messages and act accordingly• Move using (SLAM) with Obstruction avoidance• Target Acquisition (Obj. Det.) and Tracking• Approach target using camera and motor control |
|--|--|

OPERATING AMR

AMR (TURTLEBOT4)

- [Features · User Manual](#)
- <https://turtlebot.github.io/turtlebot4-user-manual/overview/features.html>
- Review the content



SIMULATION DEMO

TUTORIAL(SIM)

- [TurtleBot 4 Navigator · User Manual](#)

https://turtlebot.github.io/turtlebot4-user-manual/tutorials/turtlebot4_navigator.html

SETUP BASH

- Make sure bashrc has:
 - ROS_DOMAIN_ID = Team Number(i.e. 1 or 2 or 3...6)
- Make sure discovery setup.bash is **not** sourced!
- source ~/.bashrc

OPERATING A ROBOT(SIM) – GAZEBO

TERM1

- ros2 launch turtlebot4_ignition Bringup turtlebot4_ignition.launch.py

TERM2

- ros2 topic list
- ros2 topic echo <topic> --once
 - /oakd/rgb/preview/image_raw
 - /oakd/rgb/preview/depth
 -

OPERATING A ROBOT(SIM)

- Dock/Undock
- Manual Driving
 - Teleops
- Camera Display
 - RGB/Depth
- Navigation with rviz
 - 2D_Pose_Estimate (initial position)
 - Nav2_Goal

DIGITAL MAPPING USING SLAM (SIM)

TERMI

- ros2 launch turtlebot4_ignition Bringup turtlebot4_ignition.launch.py
nav2:=true slam:=true rviz:=true

ON GAZEBO

- Undock the robot
- Use keyboard to operate and complete the map

DIGITAL MAPPING WITH AUTO – SLAM (SIM)

TERM1

- ros2 launch turtlebot4_ignition Bringup turtlebot4_ignition.launch.py
nav2:=true slam:=true rviz:=true
- Undock the robot
- Set init pose from rviz

TERM2

- ros2 launch explore_lite explore.launch.py

TUTORIAL(SIM)

TERMINAL 1

```
$ ros2 launch turtlebot4_ignition_bringup  
turtlebot4_ignition.launch.py nav2:=true  
slam:=false localization:=true rviz:=true
```

- Undock and set init pose

TERMINAL 2

```
$ ros2 run turtlebot4_python_tutorials nav_to_pose  
$ ros2 run turtlebot4_python_tutorials  
nav_through_poses  
$ ros2 run turtlebot4_python_tutorials  
follow_waypoints  
$ ros2 run turtlebot4_python_tutorials create_path  
$ ros2 run turtlebot4_python_tutorials mail_delivery  
$ ros2 run turtlebot4_python_tutorials patrol_loop
```

OPERATING REAL ROBOT (AUTONOMOUSLY)

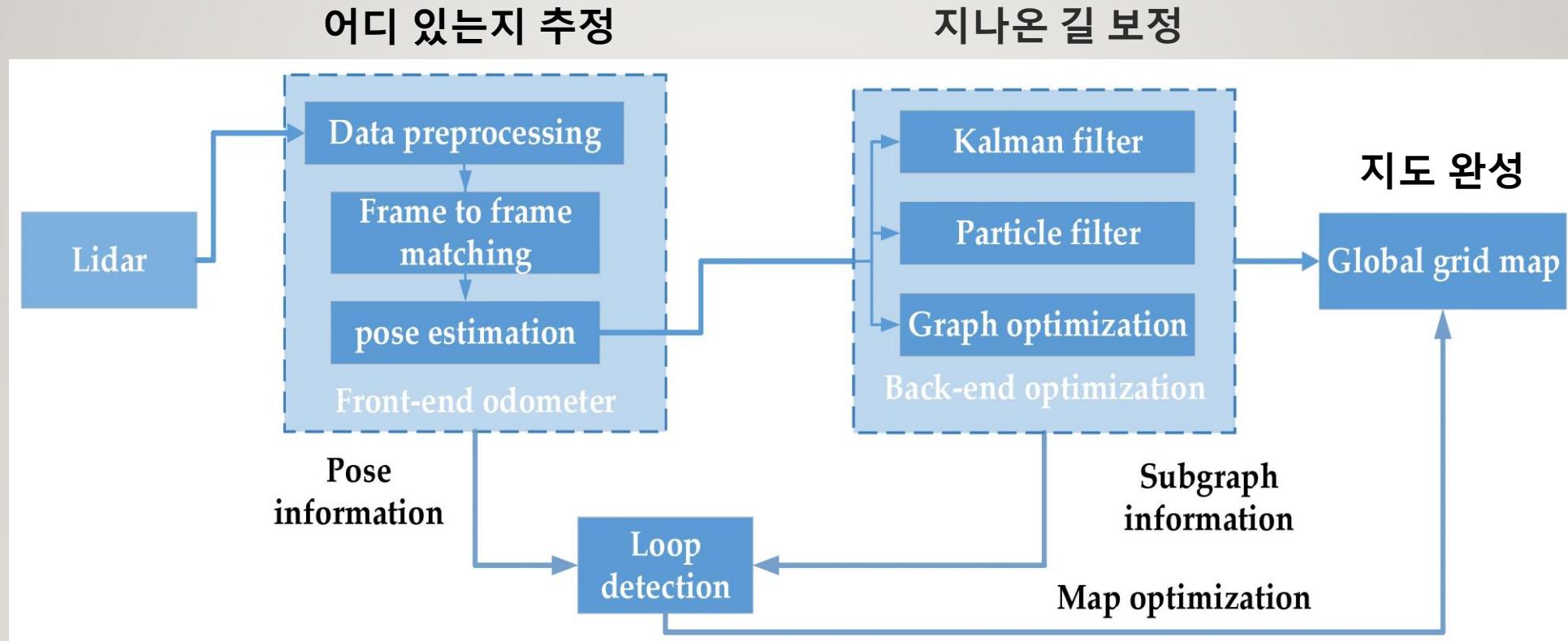
SETUP BASH

- Make sure bashrc has:
 - ROS_DOMAIN_ID = 0
- echo "alias **ros-restart**=‘ros2 daemon stop; ros2 daemon start’" >> ~/.bashrc
- Make sure discovery setup.bash **is** sourced!
- source ~/.bashrc

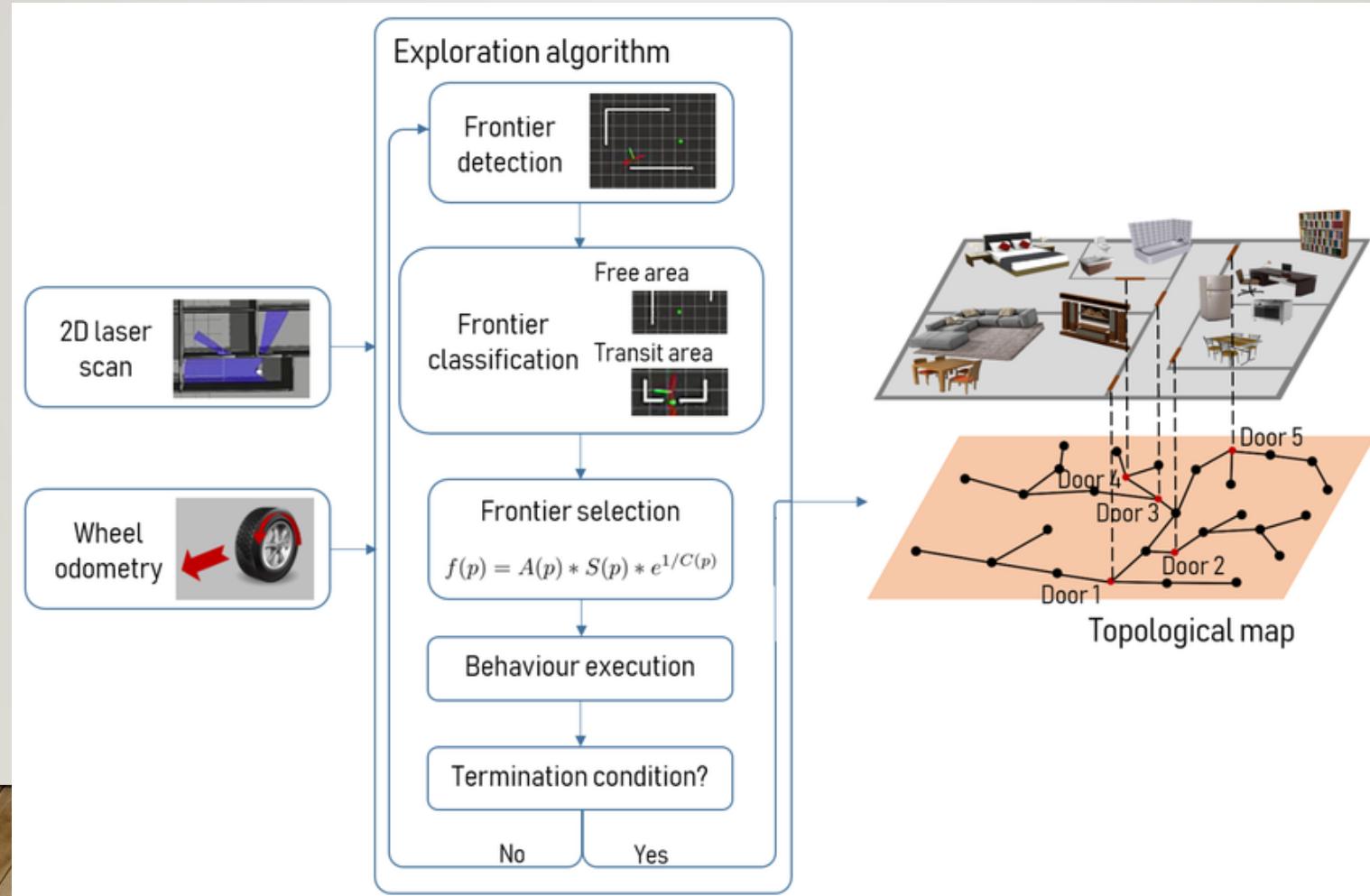
DIGITAL MAPPING (SLAM)

- [Generating a map · User Manual](#)
https://turtlebot.github.io/turtlebot4-user-manual/tutorials/generate_map.html

SLAM 개요



AUTO SLAM CONCEPT/ALGORITHM



ALGORITHM DETAIL

- Map Subscription

`explore_lite` subscribes to the SLAM-generated occupancy grid (`/map topic`) and identifies:

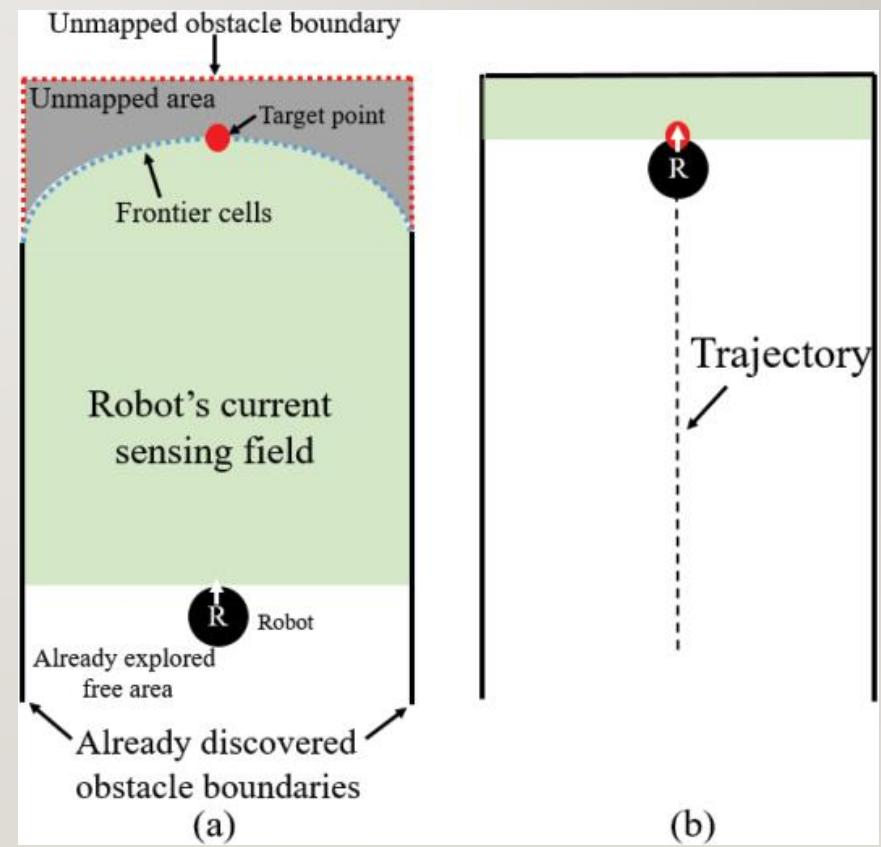
- Free space: known, unoccupied areas
- Occupied space: obstacles
- Unknown space: unexplored

- Frontier Detection

The map is scanned for cells that:

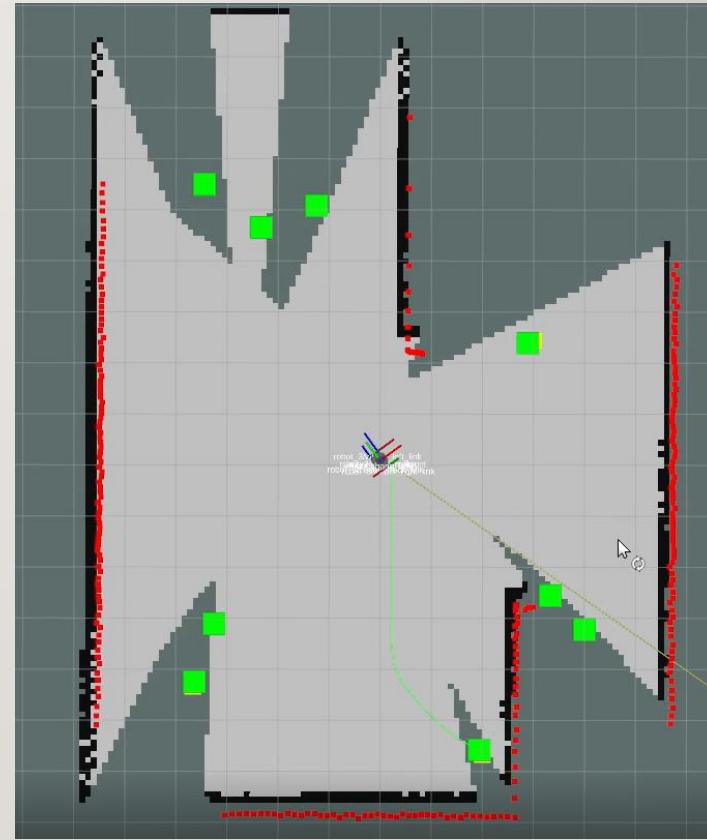
- Are free, and
- Are adjacent to at least one unknown cell.

These are marked as frontier cells.



ALGORITHM DETAIL

- Frontier Grouping
 - Frontier cells are clustered into connected regions.
 - Each group represents a potential exploration target.
- Goal Selection
 - For each frontier group, a representative point (typically the centroid or closest point) is selected.
 - The robot scores each group based on:
 - Distance from the robot
 - Information gain (how much new area might be revealed)
 - The best-scoring frontier is chosen as the next goal.



ALGORITHM DETAIL

- Termination

- While (frontiers exist and reachable)

- Select best frontier

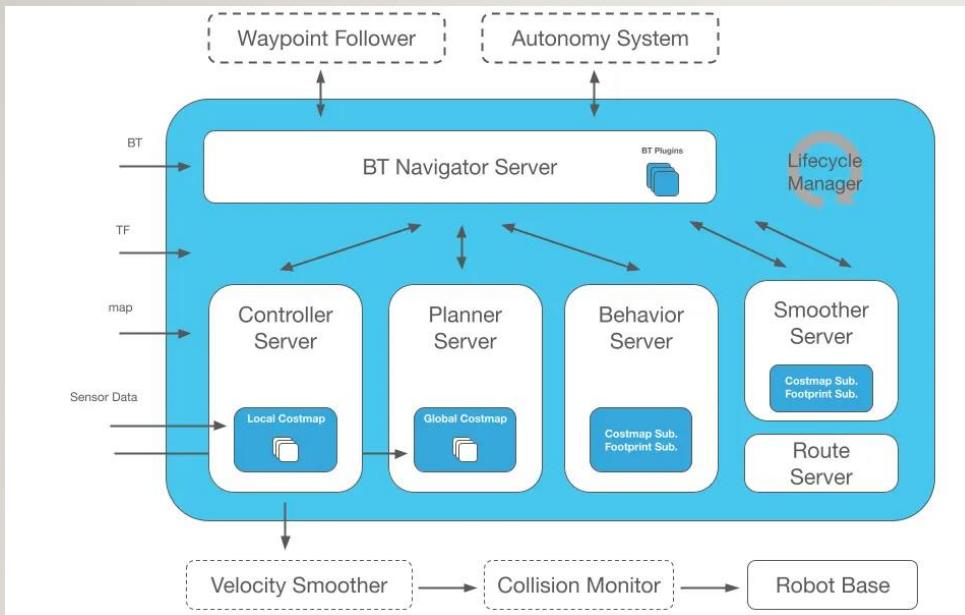
- Send as goal

- If goal fails → blacklist

- If (no frontiers or all blacklisted)

- Terminate exploration

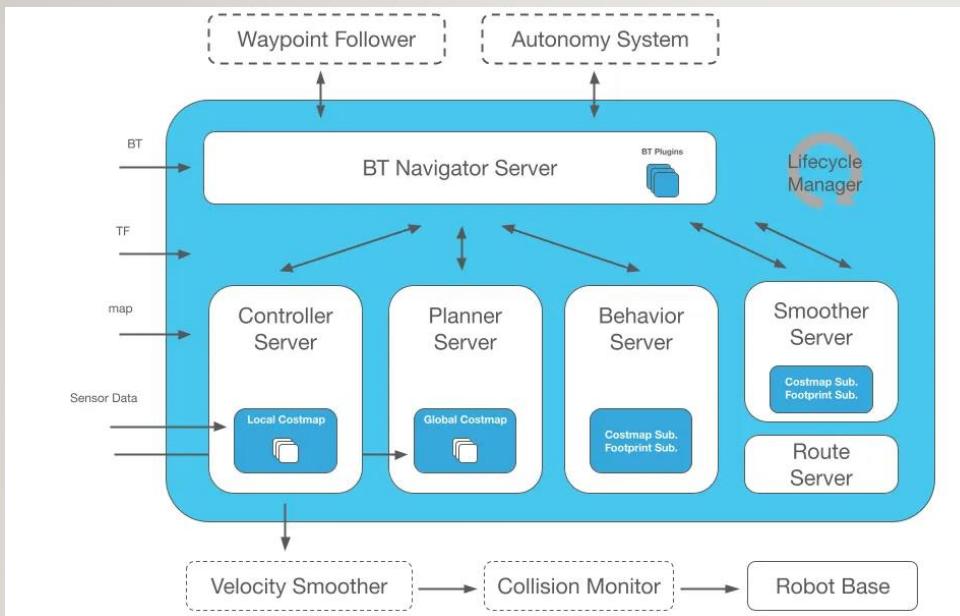
NAV2 IN ONE SENTENCE



- Nav2 turns a goal into safe motion by splitting thinking, reacting, and safety into separate servers.
- One system, many roles
- Behavior Trees coordinate everything
- No single node “drives” the robot

Mental model first. Details later.

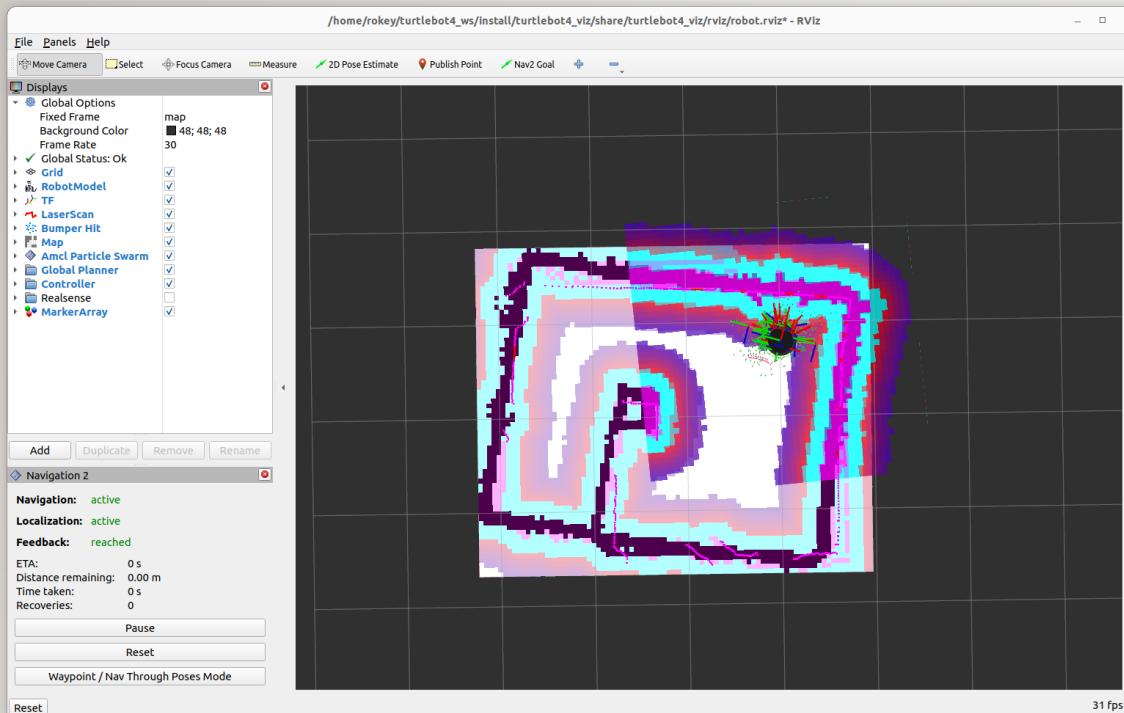
WHO DOES WHAT (CORE SERVERS)



- **BT Navigator**
Orchestrates the mission using a Behavior Tree
- **Planner Server**
Computes a global path using the **global costmap**
- **Controller Server**
Converts the path into velocities using the **local costmap**

Rule: planner thinks far, controller reacts now.

COST MAPS: TWO VIEWS OF THE SAME WORLD



- **Global Costmap**

- Static, slow, conservative
- Used for planning

- **Local Costmap**

- Dynamic, fast, sensor-driven
- Used for control

Same environment. Different time horizons.

SIMPLE NAV2 PARAM ADJUSTMENT

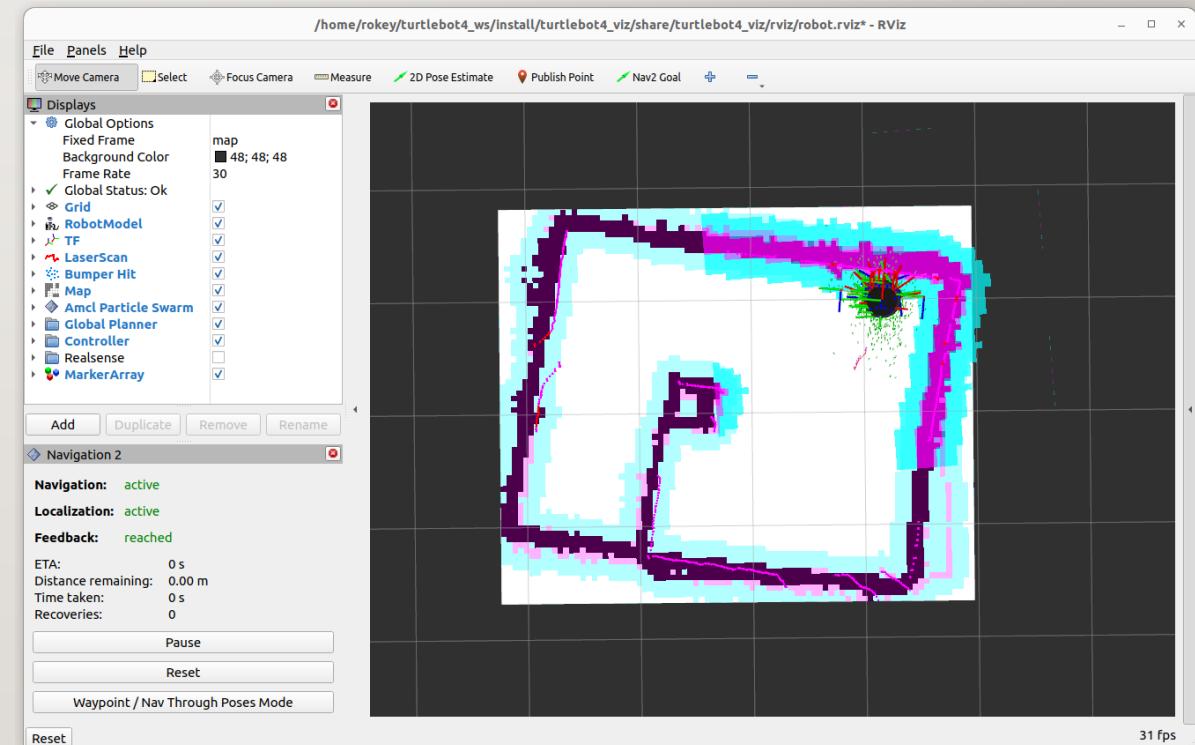
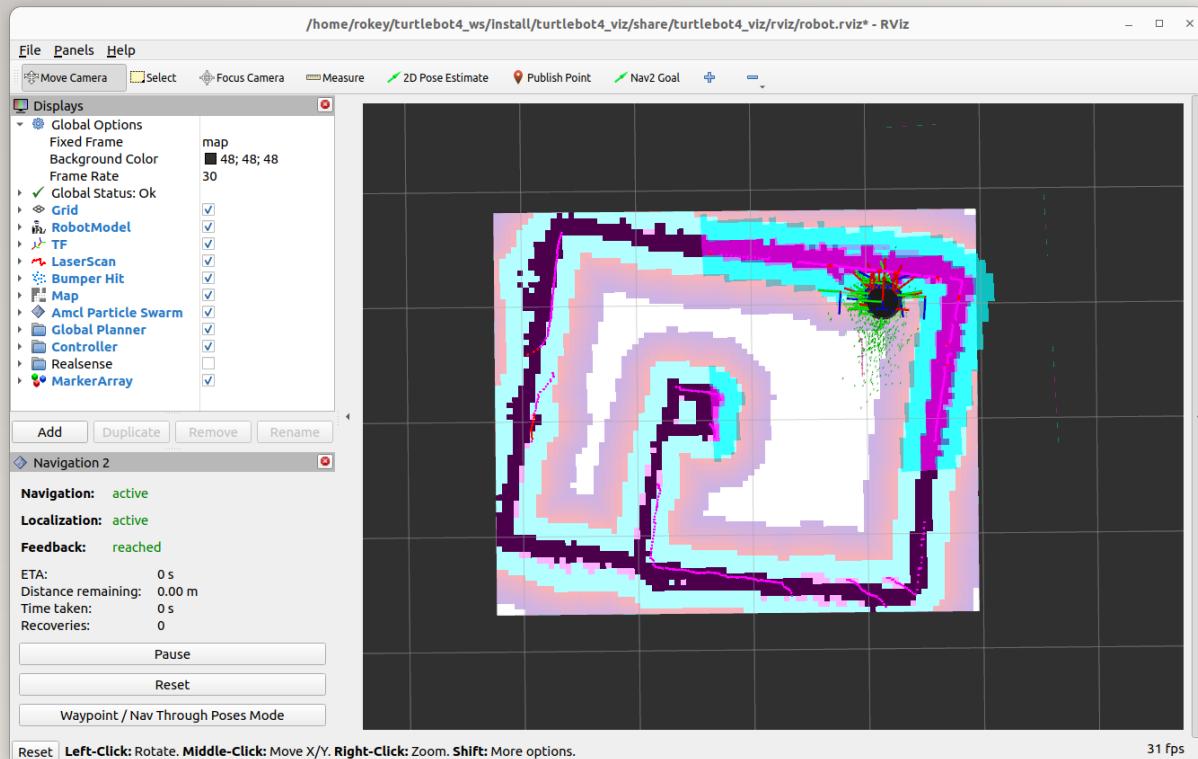
```
$ cd  
~/turtlebot4_ws/src/turtlebot4/turtlebot4_na  
vigation/config
```

- Change/adjust “inflation_radius” to fit your environment

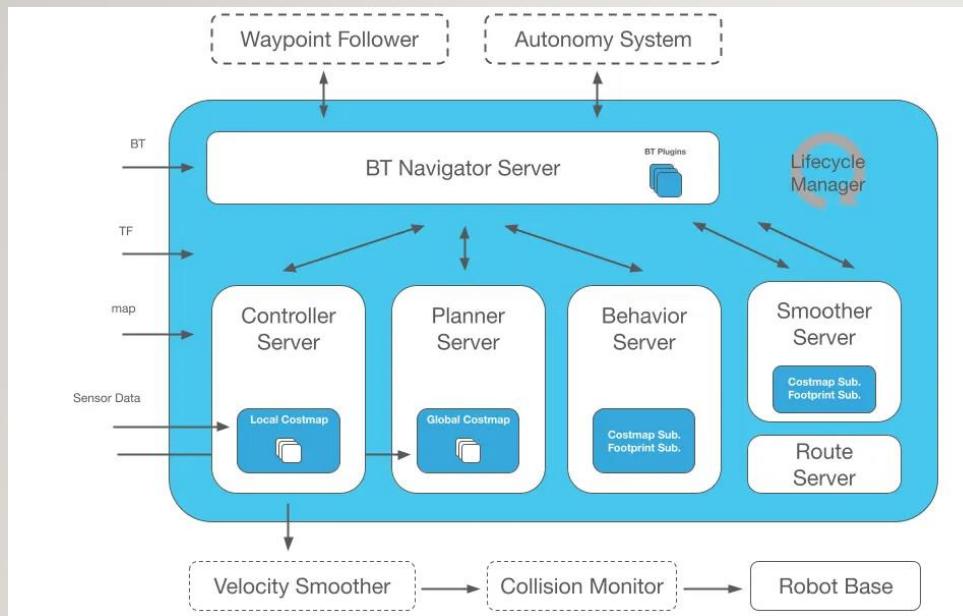
```
139  
140 local_costmap:  
141   local_costmap:  
142     ros_parameters:  
143       update_frequency: 5.0  
144       publish_frequency: 2.0  
145       global_frame: odom  
146       robot_base_frame: base_link  
147       use_sim_time: True  
148       rolling_window: true  
149       width: 3  
150       height: 3  
151       resolution: 0.06  
152       robot_radius: 0.175  
153       plugins: ["static_layer", "voxel_layer", "infl  
154         inflation_layer:  
155           plugin: "nav2_costmap_2d::InflationLayer"  
156           cost_scaling_factor: 4.0  
157  
158           #inflation_radius: 0.45  
159           inflation_radius: 0.25 #changed by aak  
160  
161  
162         voxel_layer:  
163           plugin: "nav2_costmap_2d::VoxelLayer"  
164           enabled: True
```



COST MAPS WITH PARAMETER ADJUSTMENTS



HANDLING REALITY (RECOVERY + SAFETY)



- **Behavior Server**

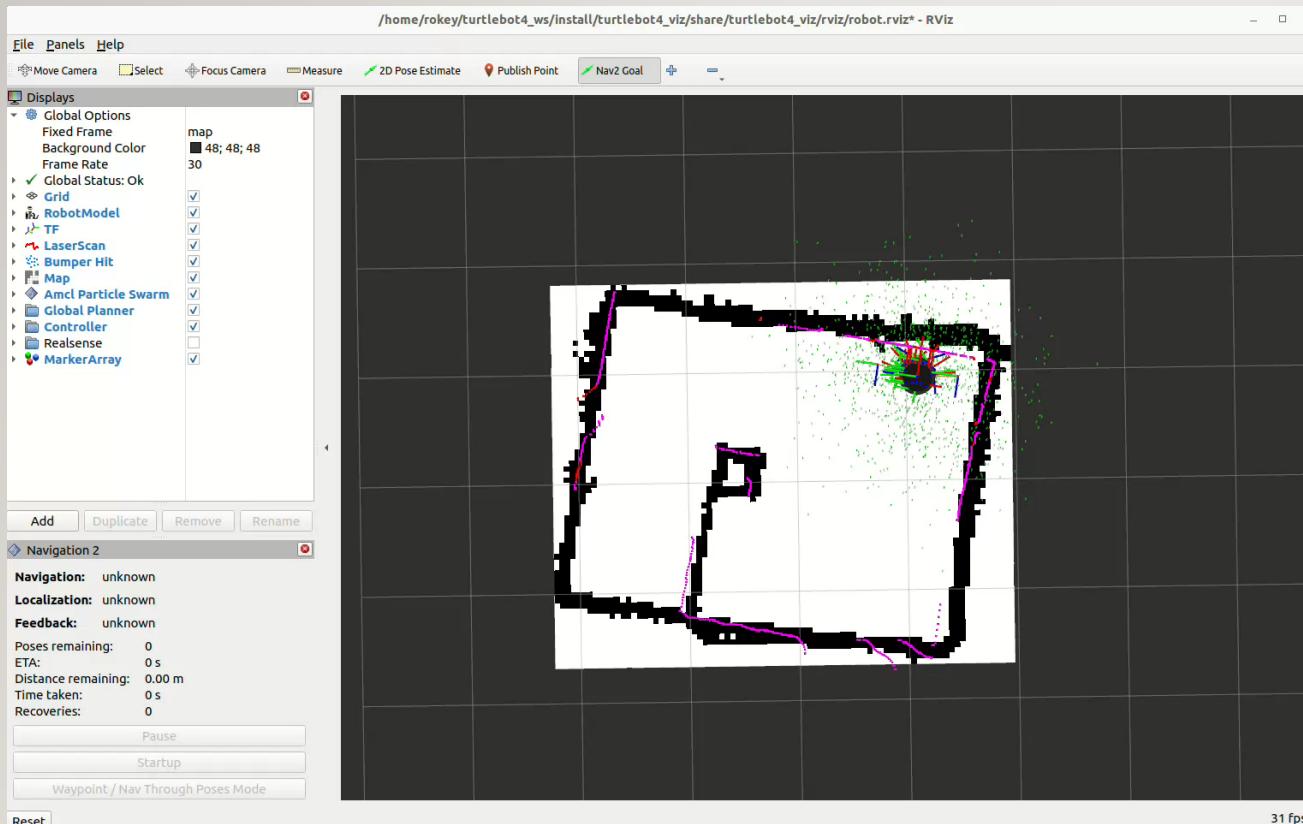
- Spin, back up, wait, clear costmaps
- Triggered when progress fails

- **Safety Chain**

- Velocity smoother
- Collision monitor
- Robot base veto power

Intelligence suggests. Safety decides.

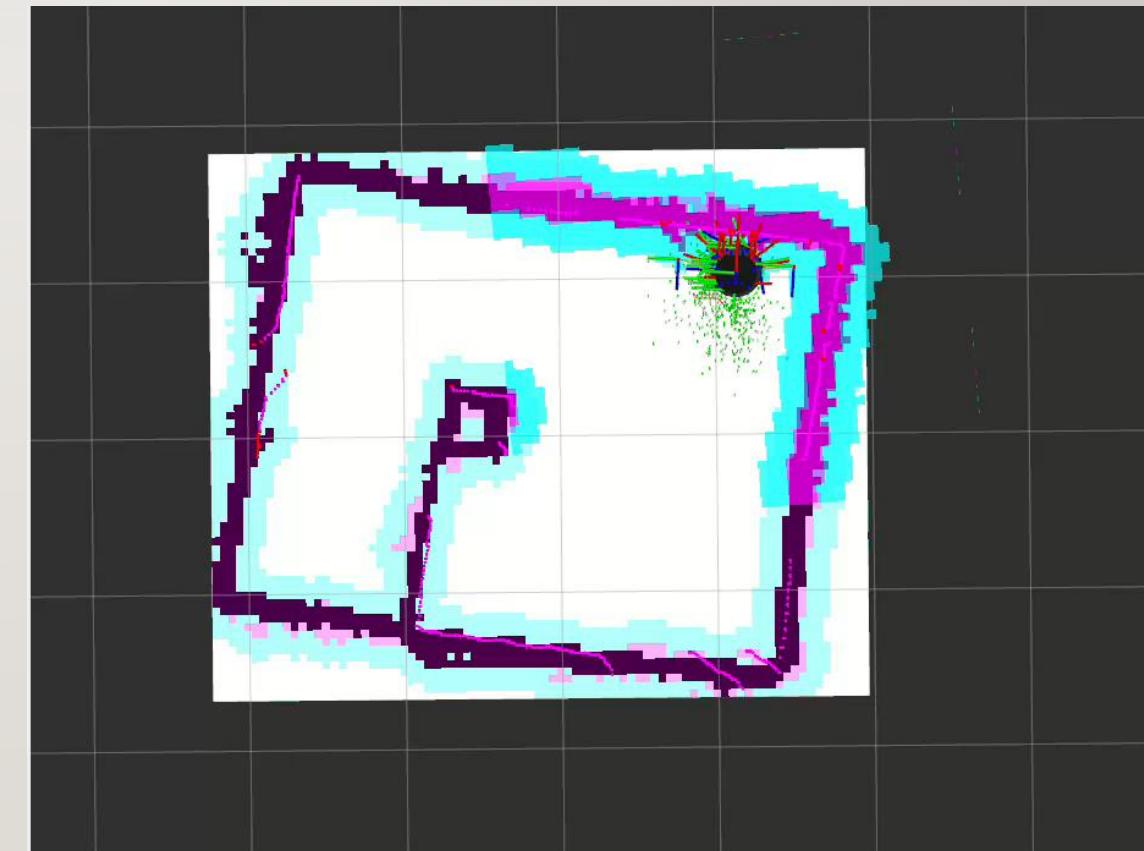
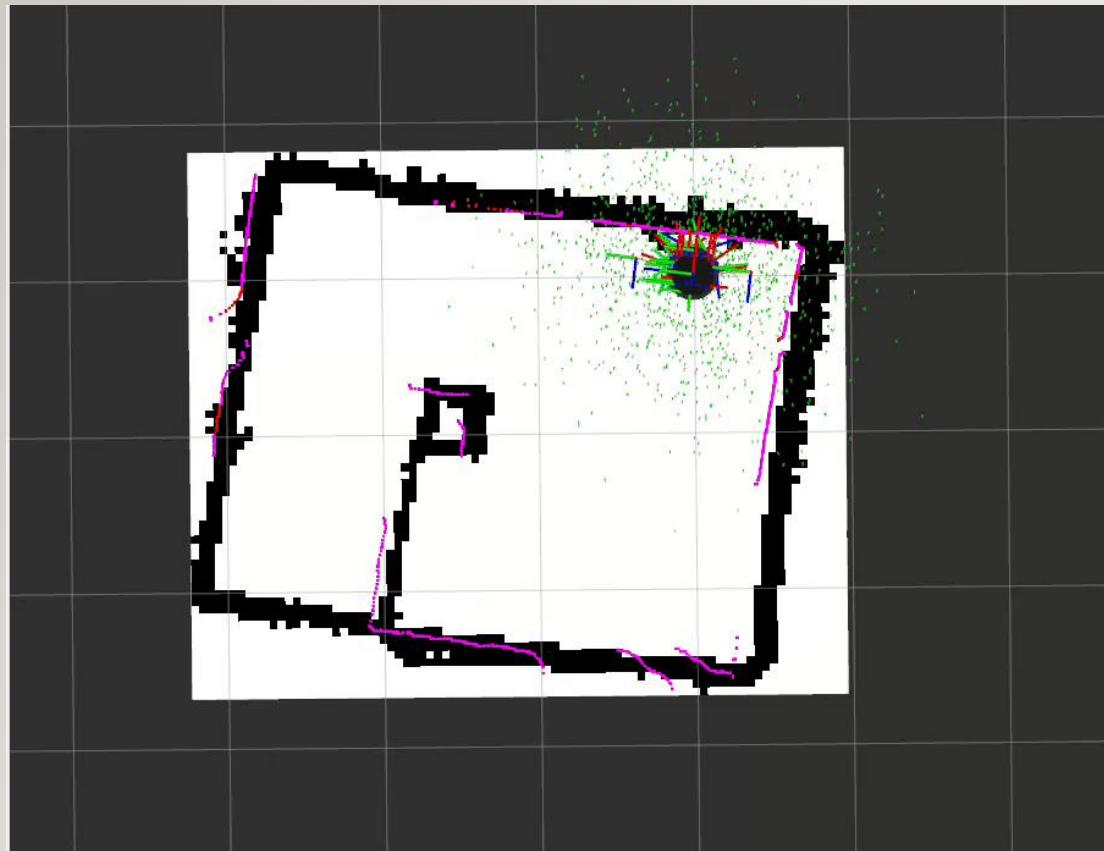
END-TO-END FLOW



1. Goal arrives
2. BT selects actions
3. Planner creates path
4. Controller drives
5. Safety filters motion
6. Behavior recovers if needed

Loop until success or failure.

END-TO-END FLOW WITH PARAMETER ADJUSTMENT



EXERCISE: PERFORM SLAM TO NAVIGATION

DAY 3 - SLAM & Navigation

Aa 이름

상태

● SLAM 개요

● 완료

● Robot_SLAM

● 완료

● Auto SLAM 개요

● 완료

● Robot SLAM_explore_lite

● 완료

● Nav2 개요

● 완료

● Robot_Navigation

● 완료

EXERCISE : RUN TUTORIAL ON REAL ROBOT

Make copy and Update the ***simulation*** tutorial code provided to successfully execute in the project environment with ***real robot***

Tutorial Codes are found in:

\$HOME/turtlebot4_ws/src/turtlebot4_tutorials/turtlebot4_python_tutorials/turtlebot4_python_tutorials

SETUP BASH(ROBOT)

- Make sure bashrc has:
 - ROS_DOMAIN_ID = Team Number(i.e. 1 or 2 or 3...6)
- Make sure discovery setup.bash **is** sourced!
- source ~/.bashrc

TUTORIAL

TERMINAL 1

```
$ ros2 launch turtlebot4_navigation  
localization.launch.py namespace:=/robot<n>  
map:=$HOME/Documents/room/room_map.  
yaml
```

TERMINAL 2

```
$ ros2 launch turtlebot4_viz view_robot.launch.py  
namespace:=/robot <n>
```

- [Set Init Pose using 2D_PoseEstimate](#)
- [Undock Robot](#)

TERMINAL 3

```
$ ros2 launch turtlebot4_navigation nav2.launch.py  
namespace:=/robot <n>
```

TUTORIAL (ROBOT) EXAMPLE CLI

- 3_2_a_nav_to_pose.py
- 3_2_b_nav_through_poses.py
- 3_2_c_follow_waypoints.py
- 3_2_d_create_path.py
- 3_2_e_mail_delivery.py
- 3_2_f_patrol_loop.py

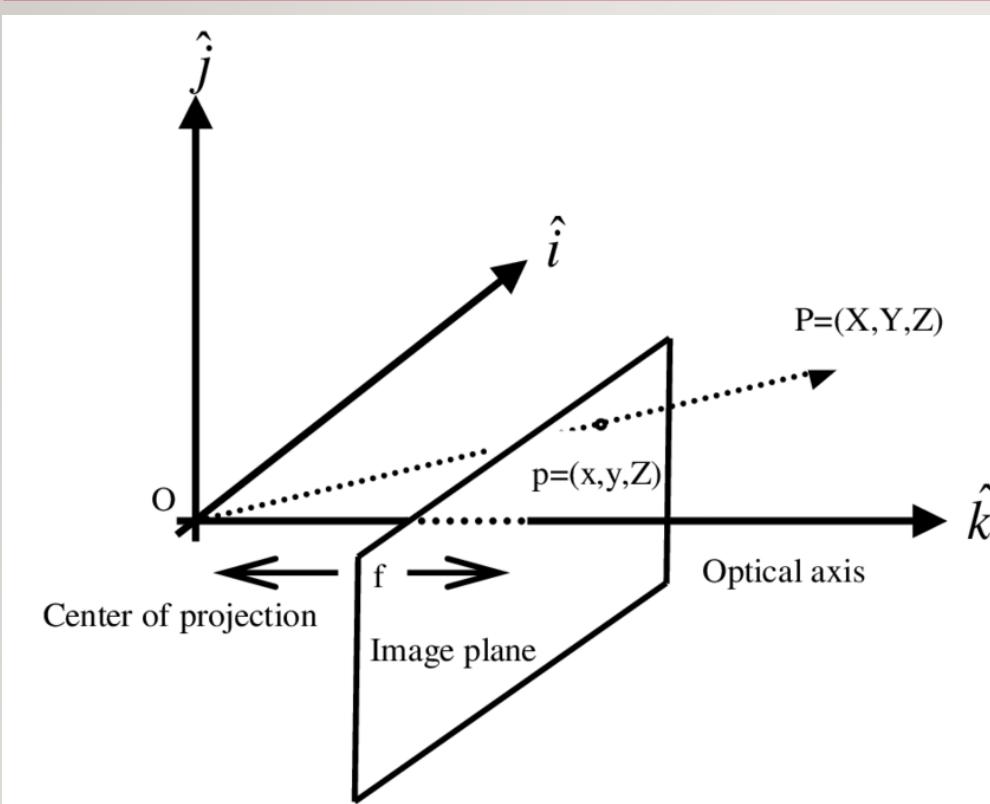
TERMINAL 4

```
$ ros2 run day3 create_path --ros-args -r __ns:=/robot<n>
$ ros2 run day3 nav_to_poses --ros-args -r __ns:=/robot<n>
$ ros2 run day3 follow_waypoints --ros-args -r
  __ns:=/robot<n>
$ ros2 run day3 nav_through_poses --ros-args -r
  __ns:=/robot<n>
$ ros2 run day3 mail_delivery --ros-args -r __ns:=/robot<n>
$ ros2 run day3 patrol_loop --ros-args -r __ns:=/robot<n>
```

HINT: Just need to find new coordinates for the goal(s)

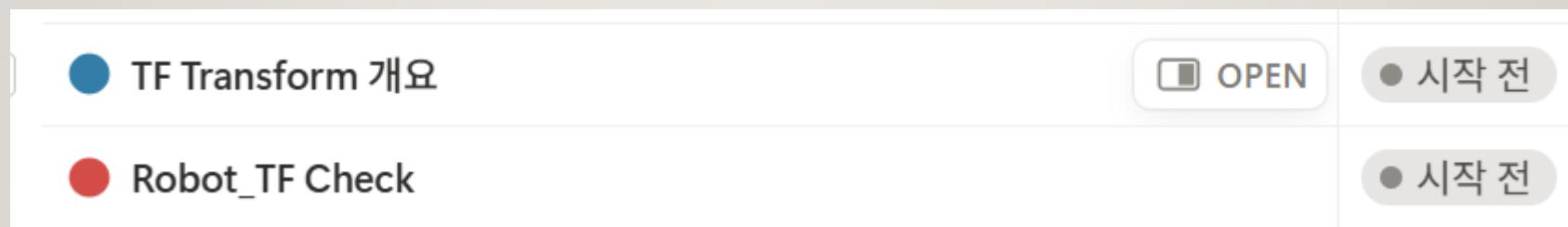
USING DEPTH TO GET MAP COORDINATES (TF TRANSFORM)

CAMERA INTRINSIC AND REPROJECTION

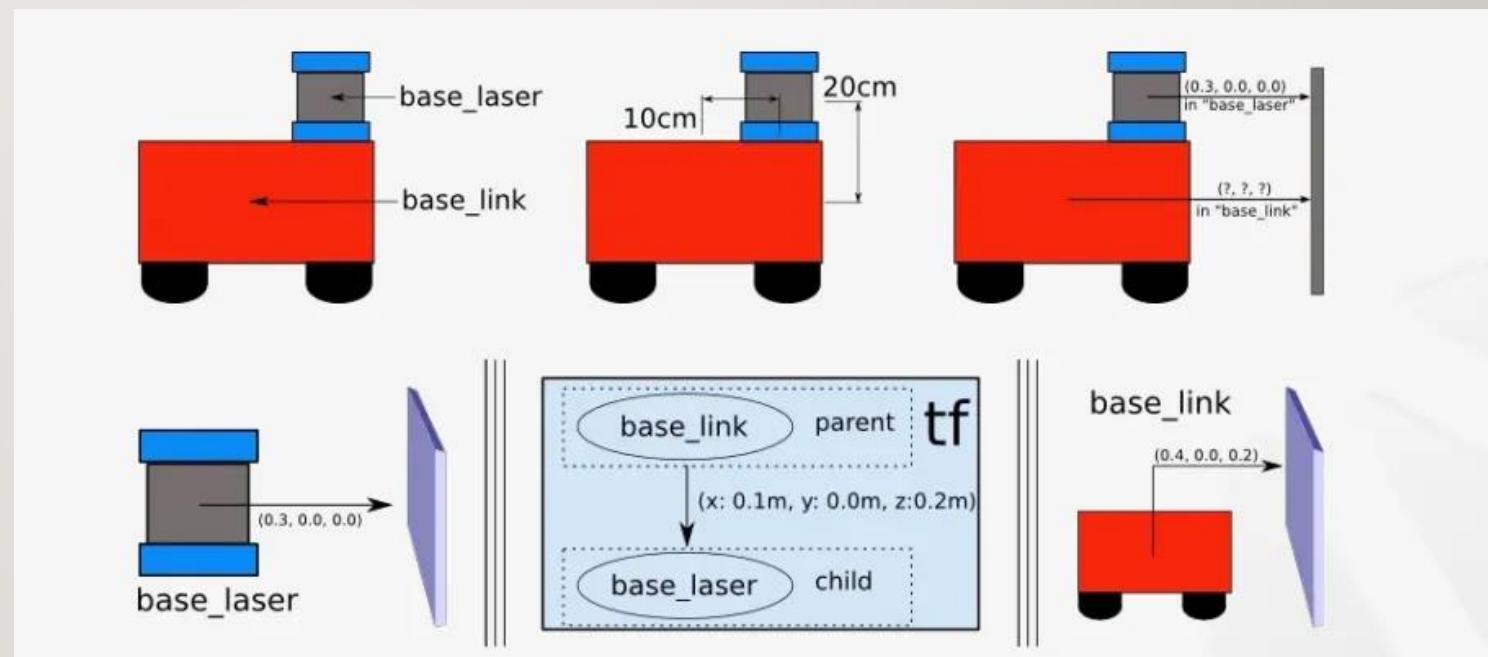
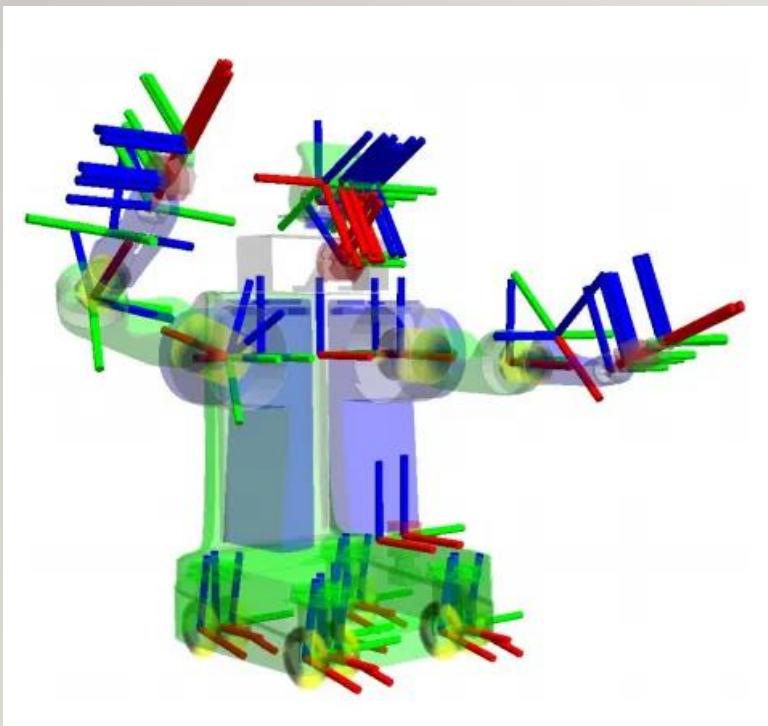


$$X = \frac{(u - c_x) \cdot Z}{f_x}, \quad Y = \frac{(v - c_y) \cdot Z}{f_y}, \quad Z = Z$$

TF TRANSFORM 개요



TF (ROBOT TRANSFORM) 개요



USING DEPTH (SIM)

```
🐍 3_1_a_depth_checker.py  
🐍 3_1_b_depth_to_3d.py  
🐍 3_1_c_depth_to_nav_goal.py  
🐍 3_1_d_nav_to_person.py
```

TERMINAL 2 (

```
$ ros2 run <pkg_name> <exec_name>
```

For example,

```
$ ros2 run day3 nav_to_person
```

DEPTH/TRANSFORM EXERCISE

Make copy and Update the ***simulation*** depth code provided to successfully execute in the project environment with ***real robot***

Simulation Depth Codes are found on **[Github](#)**

USING DEPTH (ROBOT)

TERMINAL 1

```
$ ros2 launch turtlebot4_navigation  
localization.launch.py namespace:=/robot<n>  
map:=$HOME/Documents/room/room_map.ya  
ml
```

TERMINAL 2

```
$ ros2 launch turtlebot4_viz view_robot.launch.py  
namespace:=/robot <n>
```

- [Set Init Pose using 2D_PoseEstimate](#)
- [Undock Robot](#)

TERMINAL 3

```
$ ros2 launch turtlebot4_navigation nav2.launch.py  
namespace:=/robot <n>
```

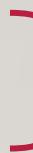
USING DEPTH (ROBOT)

```
↳ 3_1_a_depth_checker.py  
↳ 3_1_b_depth_to_3d.py  
↳ 3_1_c_depth_to_nav_goal.py  
↳ 3_1_d_nav_to_person.py  
↳ 3_2_a_nav_to_pose.py  
↳ 3_2_b_nav_through_poses.py  
↳ 3_2_c_follow_waypoints.py  
↳ 3_2_d_create_path.py  
↳ 3_2_e_mail_delivery.py  
↳ 3_2_f_patrol_loop.py  
↳ 3_3_a_depth_checker.py  
↳ 3_3_b_depth_to_3d.py  
↳ 3_3_c_depth_to_nav_goal.py
```



Simulation Code

```
↳ 3_3_a_depth_checker.py  
↳ 3_3_b_depth_to_3d.py  
↳ 3_3_c_depth_to_nav_goal.py
```



Provided Code

Provided code can be found on Github

USING DEPTH/TRANSFORM (ROBOT) EXAMPLE CLI

- ⚡ 3_3_a_depth_checker.py
- ⚡ 3_3_b_depth_to_3d.py
- ⚡ 3_3_c_depth_to_nav_goal.py

```
$ ros2 run day4 depth_checker --ros-args -r __ns:=/robot<n>
```



```
$ ros2 run day4 depth_to_3d --ros-args -r __ns:=/robot<n> -r /tf:=/robot<n>/tf -r /tf_static:=/robot<n>/tf_static
```



```
$ ros2 run day4 depth_to_goal --ros-args -r __ns:=/robot<n> -r /tf:=/robot<n>/tf -r /tf_static:=/robot<n>/tf_static
```

The image shows a Linux desktop environment with five terminal windows open in the Terminator window manager. The windows are arranged in two columns and one row.

- Top Left Terminal:** A standard Ubuntu terminal window titled "ubuntu@ubuntu: ~ 100x8". It displays a message about enabling ESM Apps service and upgrading to a new release.
- Top Right Terminal:** A terminal window titled "rokey@rokey-Victus-by-HP-Laptop-16-d1xxx: ~ 101x18". It runs a ping command to 192.168.107.57, showing four successful responses.
- Middle Left Terminal:** A terminal window titled "rokey@rokey-Victus-by-HP-Laptop-16-d1xxx: ~ 100x8". It shows ROS2 activation and the execution of a launch command for turtlebot4_navigation, specifying a map file.
- Middle Right Terminal:** A terminal window titled "rokey@rokey-Victus-by-HP-Laptop-16-d1xxx: ~ 101x18". It shows ROS2 activation and the execution of a launch command for turtlebot4_viz, specifying a namespace and Rviz.
- Bottom Left Terminal:** A terminal window titled "rokey@rokey-Victus-by-HP-Laptop-16-d1xxx: ~ 100x8". It shows ROS2 activation and the execution of a launch command for turtlebot4_navigation, specifying a namespace and a navigation parameters file.

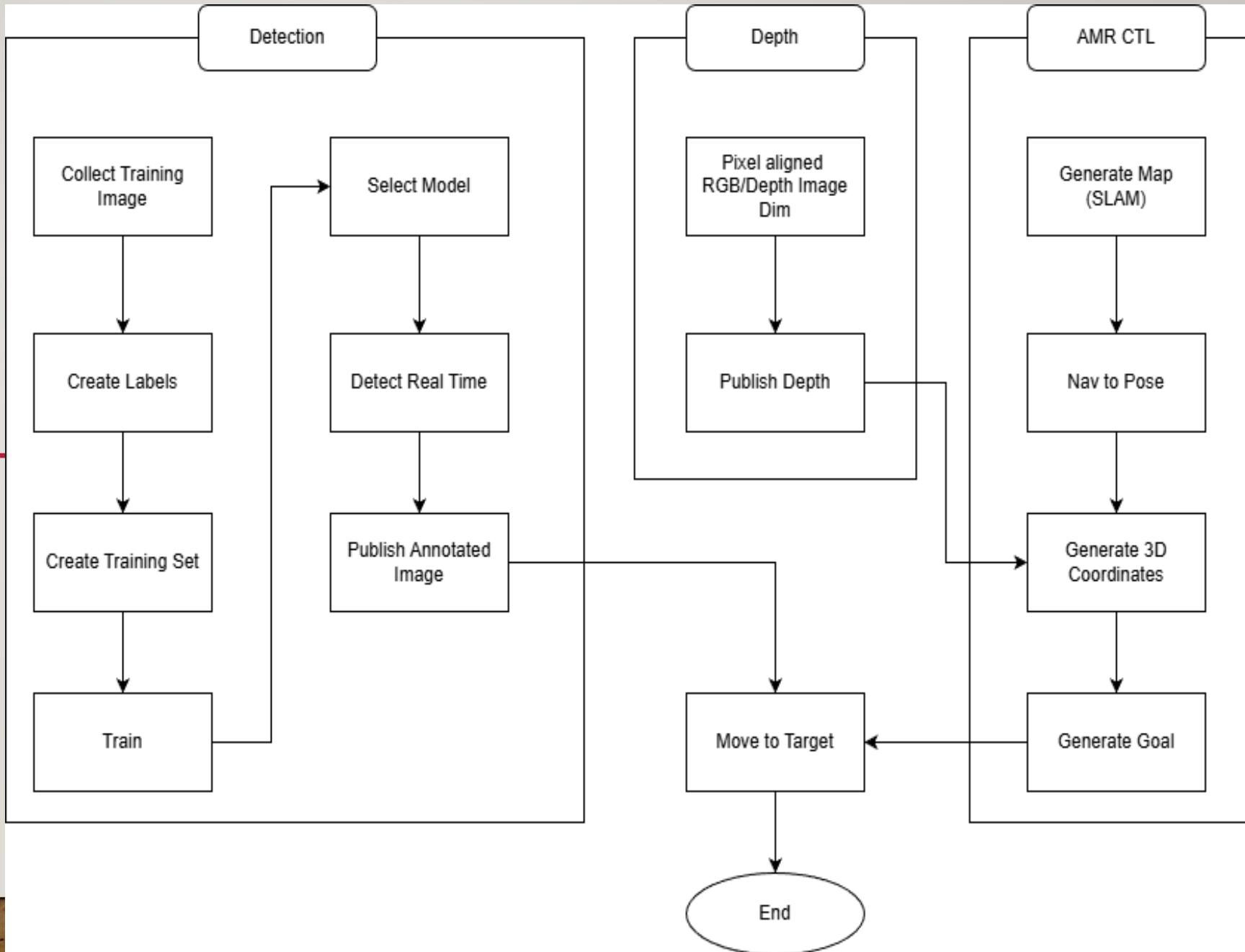
Bottom Right Content:

Recommended Terminator Layout

HOMEWORK

- Create an AMR control code
 - AMR receives an event and undocks
 - Init Pose is set
 - AMR moves to a goal position
 - AMR able to approach a target
 - **Design a way to get information about the target**
 - **Design an approach algorithm**
 - **Test**
- Update System Requirement

TASKS TO BE PERFORMED



프로젝트 RULE NUMBER ONE!!!

Are we still having
FUN!

