Dynamics and Control Laboratory

# Path Planning of Wheeled Mobile Robot with Occupancy Grid Map


Supervisor: Dr.SRANG Sarot
Co-Supervisor: Mr.SAKAL Morokot
Presenter: Mr.YONRITH Phayuth


Academic Year: 2020-2021

# Table of Contents

# Introduction

Wheeled Mobile Robot is widely used in the field. Autonomous Navigation is one of the main subject in mobile robot study.

Figure 1: Mobile Robot Application

[1]


[2]


[3]

[1] HuskyA200
[2] Nidec AGV
[3] Roomba

To achieve an autonomous navigation functionality, it needs:

- Sensors data
- Algorithms

There are multiple ways of represent the surrounding environment and method of path planning.

Environment Representation

- Graph Representation
- Cell Decomposition
- Roadmap
- Potential Field
- -etc

Path Planning

- Probabilistic Road Maps (PRMs)
- Visibility Graph
- Rapidly Exploring Random Tree (RRTs)
- Generalized Voronoi Diagram
- A*
- -etc

In this research, we aim to:

- Determine **pathway** to move the robot using **Occupancy Grid Map**
- Design a **controller** for the robot to follow the planned pathway

In this research, we aim to:

- Determine **pathway** to move the robot using **Occupancy Grid Map**
- Design a **controller** for the robot to follow the planned pathway

In this research:

- **Differential Drive Mobile Robot** is used
- **2D** environment
- **Simulation** is conducted using **Gazebo** and **ROS** software
- The project produce a **ROS** package

# Introduction

## Scope

In this research:

- **Differential Drive Mobile Robot** is used
- **2D** environment
- **Simulation** is conducted using **Gazebo** and **ROS** software
- The project produce a **ROS** package

In this research:

- **Differential Drive Mobile Robot** is used
- **2D** environment
- **Simulation** is conducted using **Gazebo** and **ROS** software
- The project produce a **ROS** package

In this research:

- **Differential Drive Mobile Robot** is used
- **2D** environment
- **Simulation** is conducted using **Gazebo** and **ROS** software
- The project produce a **ROS** package

# Research Methodology

# Research Methodology

The experiment consists of step below.

- Mobile Robot and Sensor Model
- Occupancy Grid Map
- Path Planning
- Control

# Research Methodology

The experiment consists of step below.

- Mobile Robot and Sensor Model
- Occupancy Grid Map
- Path Planning
- Control

# Research Methodology

The experiment consists of step below.

- Mobile Robot and Sensor Model
- Occupancy Grid Map
- Path Planning
- Control

# Research Methodology

The experiment consists of step below.

- Mobile Robot and Sensor Model
- Occupancy Grid Map
- Path Planning
- Control

# Mobile Robot and Sensor Model

Figure 2: Differential Drive Robot[4]

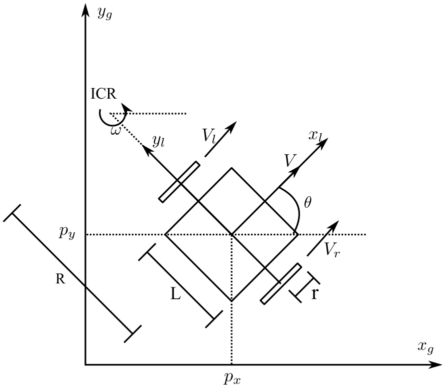
Those sensors are:

- Wheel Encoder
- Light Detecting and Ranging Sensor (LIDAR)
- Inertial Measurement Unit (IMU)

[4] Pioneer P3-DX,https://www.generationrobots.com/en/402395-robot-mobile-pioneer-3-dx.html

Kinematic model describes the robot velocities in the local frame to global frame.

Figure 3: Differential Drive Kinematic Model




Where:

- $(x_g, y_g)$is global frame
- $(x_l, y_l)$ is local frame
- $r$ is wheel radius
- $L$ is robot base length
- $V_l \& V_r$ is left and right wheel linear velocity
- $V$ is linear velocity
- $\omega$ is angular velocity

# Mobile Robot and Sensor Model

## Sensor Model

### Robot global frame velocity in continuous time step

$$\dot{x}(t) = \begin{bmatrix} \dot{p_x}(t) \\ \dot{p_y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} cos(\theta(t)) & 0 \\ sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V(t) \\ \omega(t) \end{bmatrix} \tag{1}$$

### Robot Velocity

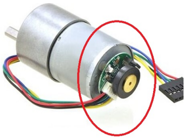
$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{2}$$

### Predicted Robot Pose

$$\hat{x}_k^- = \begin{bmatrix} p_{x,k} \\ p_{y,k} \\ \theta_k \end{bmatrix} = \begin{bmatrix} p_{x,k-1} \\ p_{y,k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} cos(\theta_{k-1})T_s & 0 \\ sin(\theta_{k-1})T_s & 0 \\ 0 & T_s \end{bmatrix} \begin{bmatrix} V_{k-1} \\ \omega_{k-1} \end{bmatrix} \tag{3}$$

**Wheel Encoder** measures the rotation velocity of each wheel denoted by $\omega_r$ and $\omega_l$.

Figure 4: Encoder



# Mobile Robot and Sensor Model

## Sensor Model

**LIDAR** data is input to LidarScanMatch that give the measurement of robot position $p_x$ and $p_y$.

**IMU** measures the robot orientation by integrate IMU gyroscope in z-axis $\omega$.

Figure 5: IMU

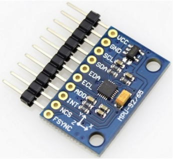



Figure 6: LIDAR

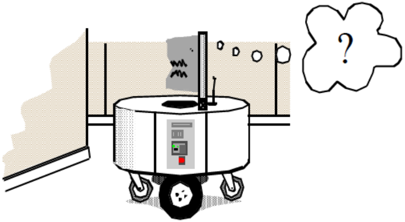
**LIDAR and IMU Measurement**

$$y_k = \begin{bmatrix} p_{x,k} \\ p_{y,k} \\ \omega T_s \end{bmatrix} \quad (4)$$

In this project, we use sensors for **Robot Localization** and building **Occupancy Grid Map**.

- **Robot Localization** is a task of determine location of robot inside the map
- **Occupancy Grid Map** is graph that represents the environment.

Figure 7: Localization[5]

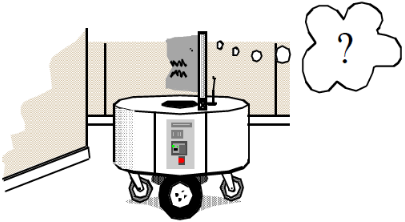



[5] https://docplayer.net/26402340-An-introduction-to-mobile-robotics.html

In this project, we use sensors for **Robot Localization** and building **Occupancy Grid Map**.

- **Robot Localization** is a task of determine location of robot inside the map
- **Occupancy Grid Map** is graph that represents the environment.

Figure 7: Localization[5]




[5]https://docplayer.net/26402340-An-introduction-to-mobile-robotics.html

# Mobile Robot and Sensor Model

## Robot Localization

We use **Extended Kalman Filter(EKF)** as sensor fusion for robot localization.

## Occupancy Grid Map

We use **Hector SLAM** ROS package to create map.

**Extended Kalman Filter(EKF)** is a state estimation algorithm that estimates the unknown or uncertain variable given the observation. In our case, the states of the system is $x$(robot pose)

## Model Equation

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \tag{5}$$

$$y_k = h(x_k) + v_k \tag{6}$$

Where:

- $y$ is the measurement vector
- $u$ is the control input vector
- $w \& v$ are the guassian white noise with covariance $Q$ and $R$ respectively

EKF is divided into 2 steps.

Table 1: Extended Kalman Filter Algorithm

| | |
|---|---|
| **Prediction** | |
| Predicted State estimate | $\boxed{\hat{x}_k^-} = f(\hat{x}_{k-1}^-, u_{k-1})$ |
| Predicted error co-variance | $P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q$ |
| **Correction** | |
| Expected Output | $\hat{y}_k = h(\hat{x}_k^-)$ |
| Measurement residual | $\tilde{y}_k = \boxed{y_k} - \hat{y}_k$ |
| Kalman Gain | $K_k = P_k^- H_k^T(R + H_k P_k^- H_k^T)^{-1}$ |
| Updated state estimate | $\hat{x}_k^+ = \hat{x}_k^- + K_k\tilde{y}$ |
| Updated error co-variance | $P_k^+ = (I - K_k H_k)P_k^-$ |

- $F_{k-1}$

$$F_{k-1} = \frac{\partial f}{\partial x}\Big|_{\hat{x}_{k-1}^+, u_{k-1}}$$

- $H_k$

$$H_k = \frac{\partial h}{\partial x}\Big|_{\hat{x}_k^-}$$

Where:

- $Q$ is the covariance matrix of prediction model
- $R$ is the covariance matrix of measurement model
- $F_{k-1}$ is the Jacobian matrix of the nonlinear state function
- $H_k$ is the Jacobian matrix of the nonlinear measurement function

# Mobile Robot and Sensor Model

## Architecture

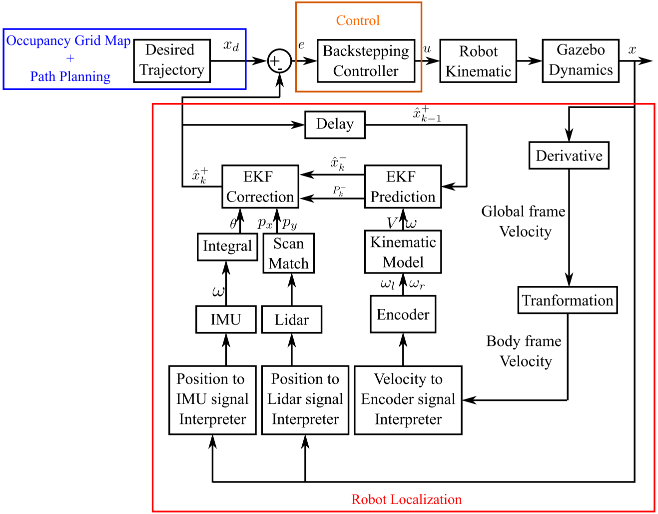
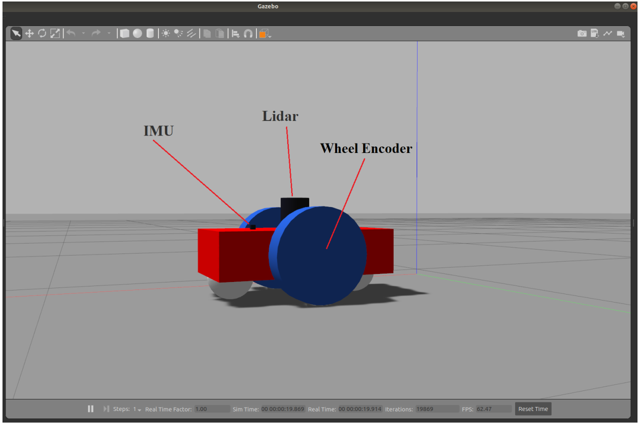


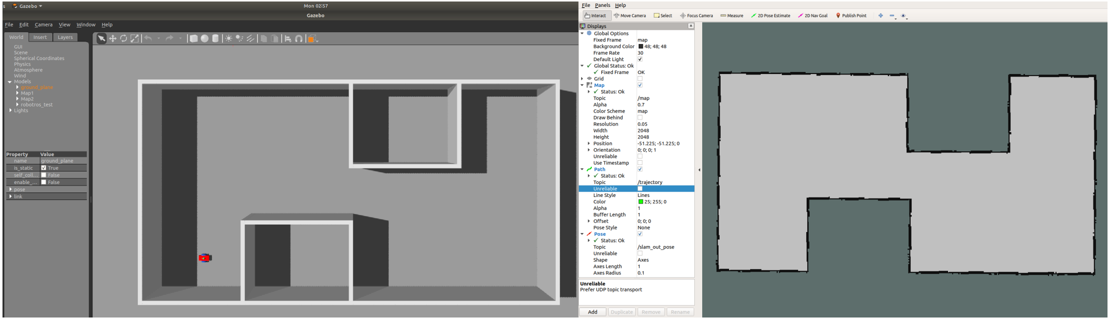
Figure 8: Mobile Robot Model in Gazebo



# Occupancy Grid Map

The Occupancy Grid Map is composed of multiple cell together to represents **Free Space** and **Obstacle Space** of the surrounding environment.

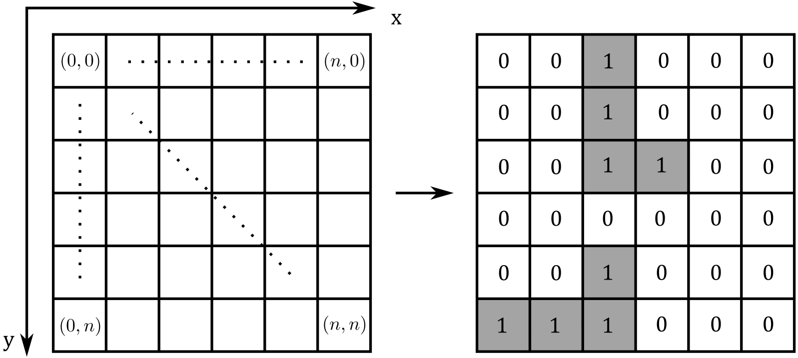
Figure 9: Occupancy Grid Map Example (view in Rviz)



# Occupancy Grid Map

Each cell has its correspond address and a probability value of either 1 or 0 which represents occupied or free respectively.

Figure 10: Occupancy Grid Map Cell



SLAM use Robot and Sensor data that observe from environment to construct the Occupancy Grid Map. In this project, we use Hector SLAM ROS package.

Figure 11: SLAM work flow

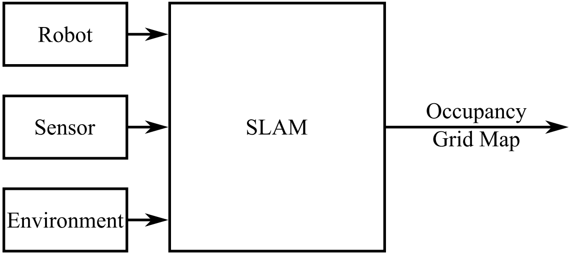
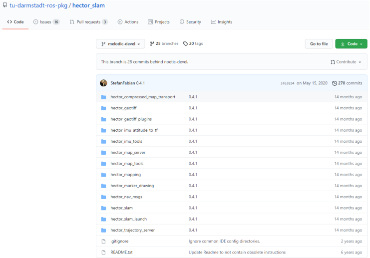



Figure 12: Hector SLAM[6]




[6]https://github.com/tu-darmstadt-ros-pkg/hector_slam

# Path Planning

# Path Planning

- Path Planning is a process of finding the pathway to move from point A to point B.
- In mobile robot navigation, Path Planning is commonly used for finding the pathway to move the robot inside the surrounding environment.

# Path Planning

- Path Planning is a process of finding the pathway to move from point A to point B.
- In mobile robot navigation, Path Planning is commonly used for finding the pathway to move the robot inside the surrounding environment.

A* algorithm is an algorithm that based on the heuristic method. A* is the optimal best-first search algorithm.

### Cost function

$$F(n) = G(n) + H(n) \tag{7}$$

Where:

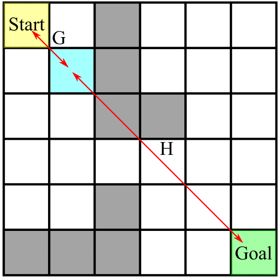
- $F$ is the total cost of node path
- $G$ is the exact distance from the starting node to the current node
- $H$ is the estimation distance from the current node to the ending node
- $n$ is the node

G and H value is calculated using Euclidean distance formula.

## Euclidean Distance

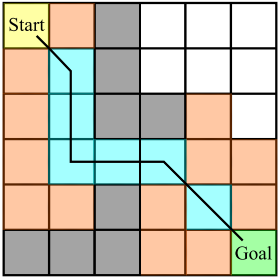
$$d_{ab} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \tag{8}$$

Figure 13: Occupancy Grid Cell in A*



- A* search for pathway from **Start** node to the **Goal** node by expanding node.
- Node Expansion is to calculate the travel cost from the current node to the neighbor node. The lowest travel cost node is chosen.

Figure 14: A* path finding from Start to Goal



# Control

# Control

In Mobile Robot Navigation, controller is used to control robot motion.

For the Differential Drive Mobile Robot, we want to control:

- $V$
- $\omega$

Most commonly known control method are:

- Reference point control
- Segment of line control
- Trajectory control

# Control

In Mobile Robot Navigation, controller is used to control robot motion.

For the Differential Drive Mobile Robot, we want to control:

- $V$
- $\omega$

Most commonly known control method are:

- Reference point control
- Segment of line control
- Trajectory control

# Control

In Mobile Robot Navigation, controller is used to control robot motion.

For the Differential Drive Mobile Robot, we want to control:

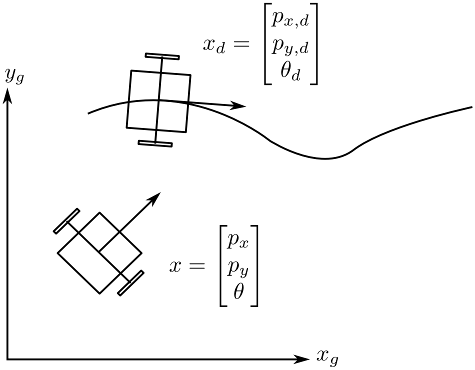
- $V$
- $\omega$

Most commonly known control method are:

- Reference point control
- Segment of line control
- Trajectory control

Backstepping Controller is one type of the trajectory controller.

Figure 15: Differential Drive Control




Controller nullify the error between the $x$ and $x_d$.

### Error Model

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{x,d} - p_x \\ p_{y,d} - p_y \\ \theta_d - \theta \end{bmatrix} \tag{9}$$

Proving by the Lyapunov stability[7], the controllers produce the control input for the robot are:

**Control**

$$\begin{bmatrix} V_c \\ \omega_c \end{bmatrix} = \begin{bmatrix} V_{ref} cos e_3 + k_1 e_1 \\ \omega_{ref} + k_2 V_{ref} e_2 + k_2 sin e_3 \end{bmatrix} \quad (10)$$

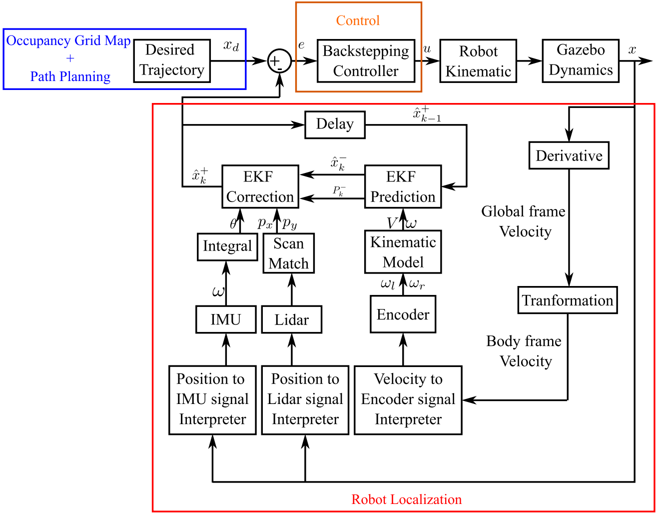
Where:

- $k_1, k_2, k_3$ are positive constants for tuning
- $V_c$ is linear velocity control
- $\omega_c$ is angular velocity control
- $V_{ref}$ is reference linear velocity
- $\omega_{ref}$ is reference angular velocity

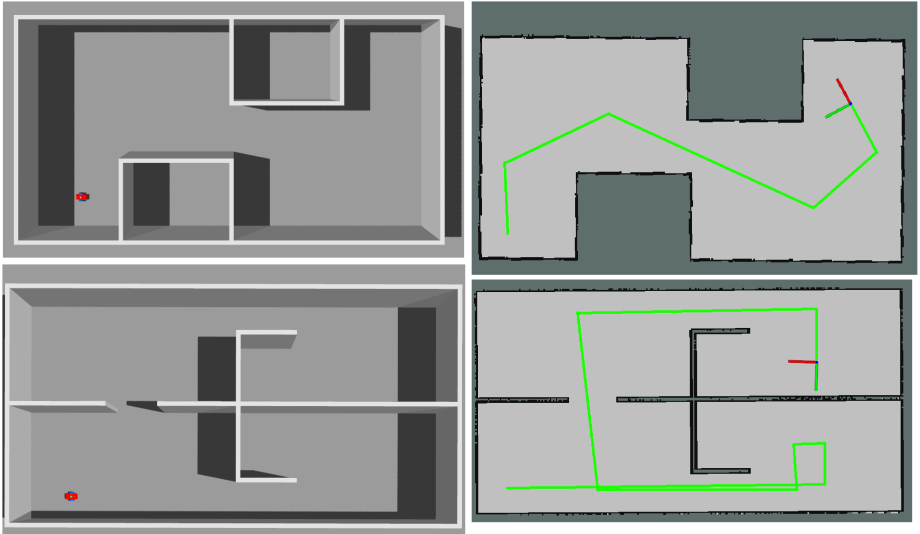
[7] G. Zidani et al., "Backstepping controller for a wheeled mobile robot," 2015 4th International Conference on Systems and Control (ICSC)

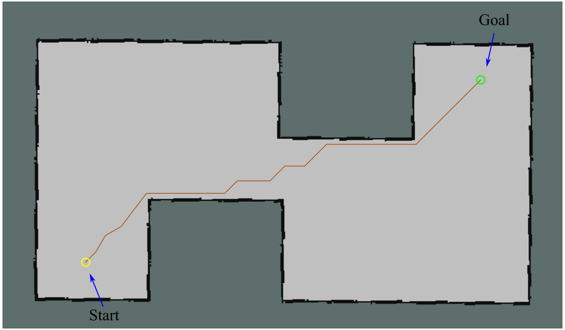
# Control

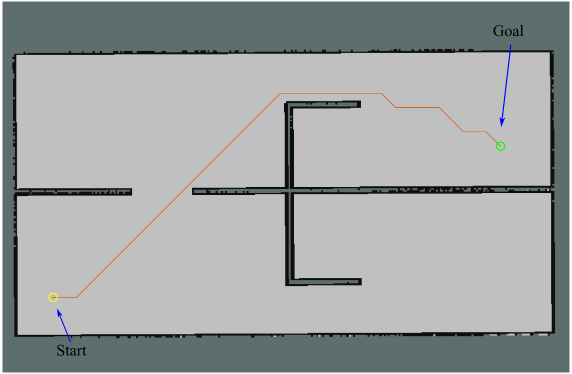
## Architecture



# Result and Discussion

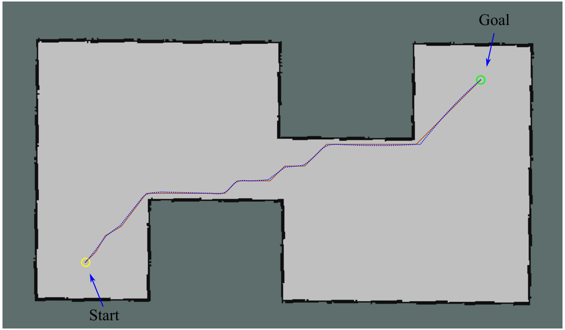


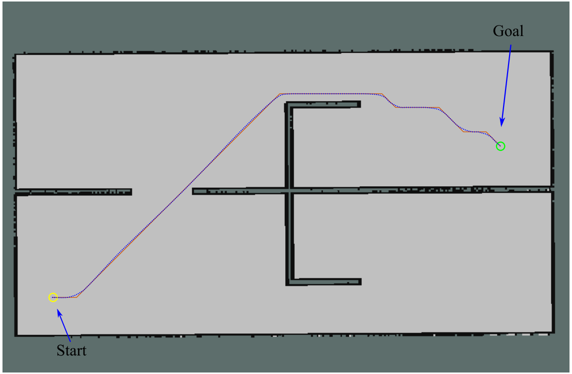

Goal
Start


Goal
Start

# Result and Discussion

## Control



# Conclusion and Recommendation

In the project:

- A* path planning generate pathway using low computational time because the cost function has the heuristic value $H$ as long as its value is not overestimated
- The backsteppig controller results controlled output close to the generated pathway
- EKF give an accurate pose estimation

Contribution : Extended use of ROS package from this project.

- The controller can be further developed or switched
- Add or Switch sensors that have suit algorithm
- Others can use the package to test, simulate, and visualize the robot in the simulated environment.
- Real world implementation as this package has topic ready for data publication.

In the project:

- A* path planning generate pathway using low computational time because the cost function has the heuristic value $H$ as long as its value is not overestimated
- The backsteppig controller results controlled output close to the generated pathway
- EKF give an accurate pose estimation

Contribution : Extended use of ROS package from this project.

- The controller can be further developed or switched
- Add or Switch sensors that have suit algorithm
- Others can use the package to test, simulate, and visualize the robot in the simulated environment.
- Real world implementation as this package has topic ready for data publication.

In the future work, the project will be implemented in hardware.

- Improve accuracy and performance
- Investigate with hardware capability such as computational speed, data publishing rate, sensor noise, interference, unexpected failure -etc
- Disturbance properties -etc.

*THANK YOU*