Real-Time Localization and Mapping in Mobile Robots

Dave Laygo (B.S. MAE '25) • Christopher Gonzales (B.S. MAE '23)

Advisors/Mentors: Prof. Jorge Cortés • Prof. Sonia Martínez • Parth Paritosh • Scott Addams • Brandon Bao

Multi-Agent Robotics (MURO) Lab at University of California, San Diego

UC San Diego JACOBS SCHOOL OF ENGINEERING Mechanical and Aerospace Engineering

GEAR Guided Engineering Apprenticeship in Research

INTRODUCTION

- The purpose of the project is to implement real-time simultaneous localization and mapping algorithms in Turtlebots to demonstrate 2-D mapping capabilities

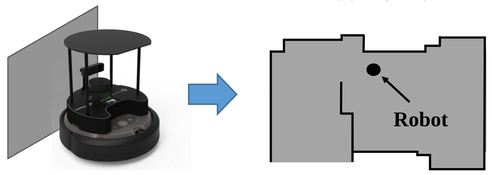


Figure 1: 2D Map Model of MURO Lab from top-down

Our Mobile Robot: TurtleBot 4

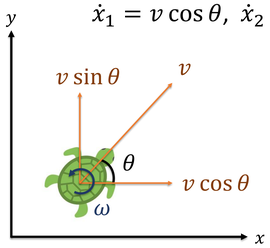
- The TurtleBot 4 is a mobile educational/research robot with advanced computing power, equipped with a multitude of sensors for data collection and modern mobility/control applications
- Operates wirelessly through network connection with a personal computer using ROS and Linux

Figure 2: Image of TurtleBot 4 with visible iRobot® Create® 3 mobile base, Raspberry Pi 4 display using ROS 2, 2D LiDAR, and OAK-D stereo camera

Unicycle Model Dynamics

- Movement of the TurtleBot is modelled through unicycle dynamics, a path-dependent state system
- Given a 2D plane with x-y coordinates, the equations of motion are given by:

$$\dot{x}_1 = v\cos\theta,\ \dot{x}_2 = v\sin\theta,\ \dot{\theta} = \omega$$



Heading: (x, y, θ)
v – linear input velocity
ω – angular input velocity

Figure 3: Model of unicycle dynamics on a 2D x-y plane

Controls

- TurtleBot 4 controls used in the project was a simple open-loop control system
- Wireless control via ROS Keyboard Teleoperation package, displayed on Linux command-line interface

u i o
j k l
m , .

Figure 4: Keyboard keys for moving around the TurtleBot

METHODOLOGY

Light Detection and Ranging (LiDAR) Data

- LiDAR operates on time of flight (TOF) principle
- LiDAR message obtained via /scan topic:
  - I. Ranges [m]: [0.15 (min) – 12 (max)]
  - II. Angles [rad]: angle_increment (0.5 degrees)
  - III. Timestamp and Header
- Data point of observed obstacle calculated by:

$$x = d\cos\theta \qquad y = d\sin\theta \qquad z = 1$$

- Data points are in LiDAR (sensor) frame and need to be transformed in order to obtain a static map

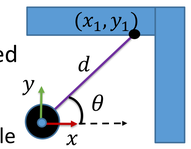


Figure 5: LiDAR model hitting an obstacle

Iterative Closest Point (ICP) Correction

- Iterative Closest Point (ICP) Algorithm
  - I. Compares current dataset to previous dataset

Association: point-to-point correspondence

Transformation: $\tilde{T} = \underset{T}{argmin}\frac{1}{N}\sum_i^N \|\boldsymbol{t_i} - \boldsymbol{Ts_i}\|^2$

Error Evaluation: $\boldsymbol{e} = \frac{1}{N}\sum_i^N \|\boldsymbol{t_i} - \boldsymbol{Rs_i} + \boldsymbol{t}\|^2$

N – # of points **R** – rotation matrix **t** – translation
$\boldsymbol{t_i}$ – target points $\boldsymbol{s_i}$ – source points

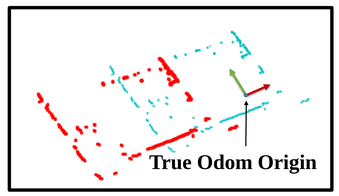


Figure 7: Comparison of transformed data without ICP (red) versus with ICP (blue)

Odometry-Based Localization

- Odometry message obtained via /odom topic:
  - I. Position [m]
  - II. Orientation [quaternions]
- Estimation of robot's change in position and orientation over time based on motion sensors

$$\vec{\boldsymbol{p}}_{\boldsymbol{odom}} = \boldsymbol{R_z} \times \vec{\boldsymbol{p}}_{\boldsymbol{sensor}} + \boldsymbol{d}$$

- Wheel encoder induced drift

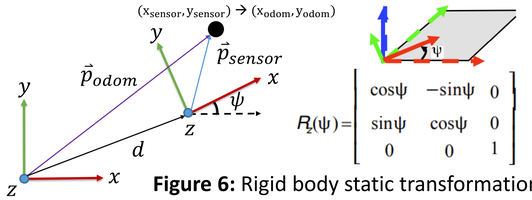


Figure 6: Rigid body static transformation

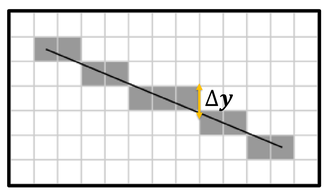
Occupancy Grid Mapping

- Discretization of map space into elements with calculated occupancy probability estimation

$$\boldsymbol{p(m_i|\, z, x) \rightarrow 0: free} \quad \boldsymbol{p(m_i|\, z, x) \rightarrow 0.5: unknown}$$
$$\boldsymbol{p(m_i|\, z, x) \rightarrow 1: occupied}$$

p – occupancy probability $\boldsymbol{m_i}$ – occupancy cell
z – sensor measurements **x** – position of robot

- Bresenham's Line Drawing Algorithm
  - I. Declares free space on occupancy grid map



Decision parameter: $\boldsymbol{\phi_{k+1} = \phi_k + 2\Delta y}$

Figure 8: Bresenham's line algorithm for LiDAR data

IMPLEMENTATION

- Iterative data transfer from TurtleBot to written mapping algorithm via ROS publishers and subscribers

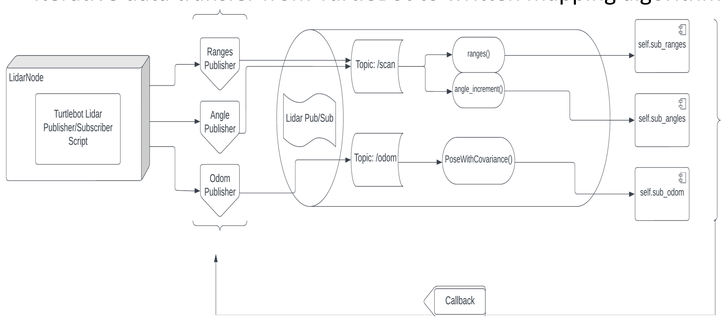


Figure 9: Flowchart documenting real-time mapping procedure

Publisher: Sends data from node to topic
- I. TurtleBot (topic: /odom)
- II. LiDAR (topic: /scan)

Subscriber: Receives data from a topic
- I. Written Mapping Algorithm

RESULTS

- Final maps obtained using mapping algorithm

Figure 10: Raw LiDAR scan data map of MURO Lab

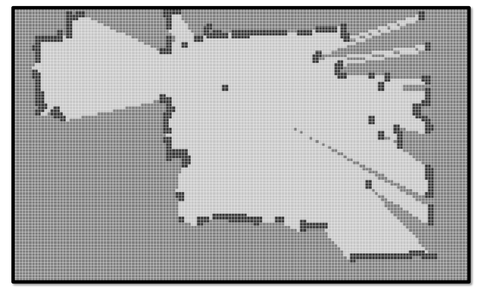


Figure 11: Occupancy Grid Map of MURO Lab

CONCLUSIONS

- Wrote algorithm that reads TurtleBot 4 sensor data and creates a resulting map of environment
- Integrated ROS topic publisher and subscriber to perform localization and mapping in real-time
- Future Work:
  - I. Extend to Multi-Robotic Network
  - II. Extend to Quadcopter collaboration
  - III. Incorporate MarvelMind Beacon localization

REFERENCES

[1] A. Kramer (2019). LIDAR Odometry with ICP. Bot Blog. http://andrewjkramer.net/lidar-odometry-with-icp/

[2] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, A. Paques (2018). PythonRobotics: a Python code collection of robotics algorithms. https://doi.org/10.48550/arXiv.1808.10703

ACKNOWLEDGEMENTS

The authors want to thank Parth, Scott, and Brandon for the mentorship received throughout the project. We are grateful to Prof. Jorge Cortés, Prof. Sonia Martínez, Prof. Evdokimenko, Sergio Godinez, Jessica Baldis, and the GEAR program for such an amazing opportunity.