University of Missouri

Vision-Guided Intelligent Robotics Lab (ViGIR)

EECS 4340/7340 Building Intelligent (Vision-Guided) Robots

Lab 3

Dead Reckoning

Submission Due : Oct 14 - 2025

Objective

The goals of this lab are to:

- Understand the concept of dead reckoning using odometry in ROS 2.
- Learn robot navigation using discrete state feedback control.

Reference Materials (ROS 2)

- ROS 2 Tutorials (Humble): `https://docs.ros.org/en/humble/Tutorials.html`
- *Learning ROS for Robotics Programming* (ROS 2 concepts still applicable).
- *ROS by Example* (conceptual background).
- *Introduction to Autonomous Mobile Robots* (Siegwart et al.), Ch. 3.
- *Robotics, Vision and Control* (Peter Corke), §4.1.1.4.
- Course Lecture Notes 7–11.

Prelab

1. Read about ROS 2 time APIs (rclcpp/rclpy time, clocks, rates). Explain the difference between `ROS time` and `system time` and when simulated time is used.

2. Define **odometry** and explain its importance in mobile robots.

3. Using ROS 2 CLI, give the exact sequence of commands to find the message type of `/odom`.

4. What is a *tf2 broadcaster* in ROS 2? Explain how tf frames (`map`, `odom`, `base_link`) relate, and how transforms are published and queried using `tf2_ros`.

Background

Feedback Control for Dead Reckoning. In this lab you will implement discrete state feedback control to drive base to goal poses using odometry. We formulate the kinematics in the goal-centered polar frame. Let the robot pose be (x, y, θ) in the odom (or map) frame, the goal pose be (x_g, y_g, θ_g), and define the goal-centered polar state vector as in the lecture notes.

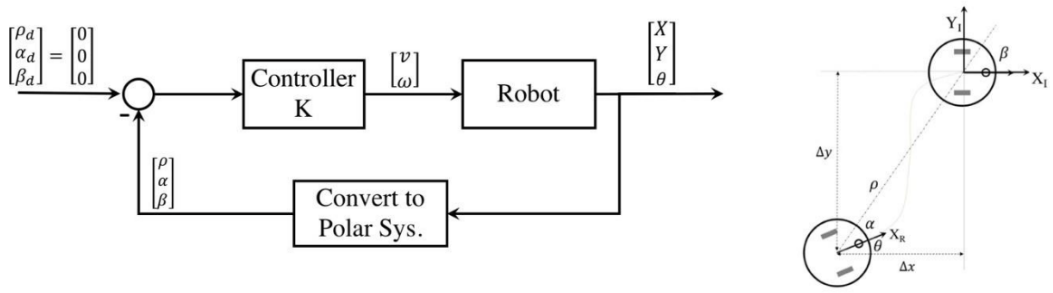

Figure 1: The feedback control loop and robot poses.

Discrete Control Law. A commonly used feedback law (sampled at Δt) is:

$$v_k = k_\rho \, \rho_k, \qquad \omega_k = k_\alpha \, \alpha_k + k_\beta \, \beta_k,$$

with suitable sign conventions and gain constraints for stability of a unicycle model in the local region of the goal (refer to the lecture notes for the stability constraints). Implement saturation on v and ω to respect platform limits. Convert between Cartesian and polar frames when computing errors, and continuously monitor the pose via `/odom`.

Lab Procedure

Implementation of Dead Reckoning (Simulator)

Use Gazebo and TurtleBot simulator, implement a ROS 2 node that:

1. Subscribes to `/odom` and maintains the current pose.
2. Converts Cartesian errors to the goal-centered polar frame (ρ, α, β).
3. Computes discrete feedback commands (v, ω) and publishes `geometry_msgs/msg/Twist` on `/cmd_vel`.
4. Navigates to the four goal poses shown in Fig. 2. For each trial, report the final error between desired and reached goal.
5. Now try to produce the 8 trajectories shown in Figure 3.

Dead Reckoning On Real Robot

1. Deploy the same node on a real robot. You may need to re-tune gains to account for wheel slip, latency, and sensor noise.
2. Answer: Do simulator-tuned gains perform similarly on hardware for the eight-trajectory task and for four-goal task?
3. Save and plot trajectories for inclusion in the final report. For each goal, report the final pose error.

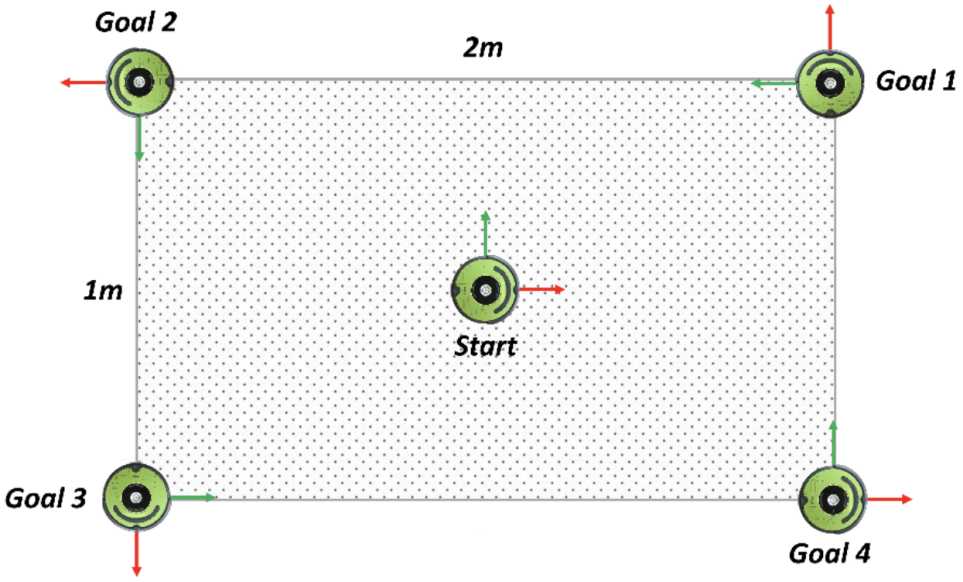

Figure 2: Four goal experiment.

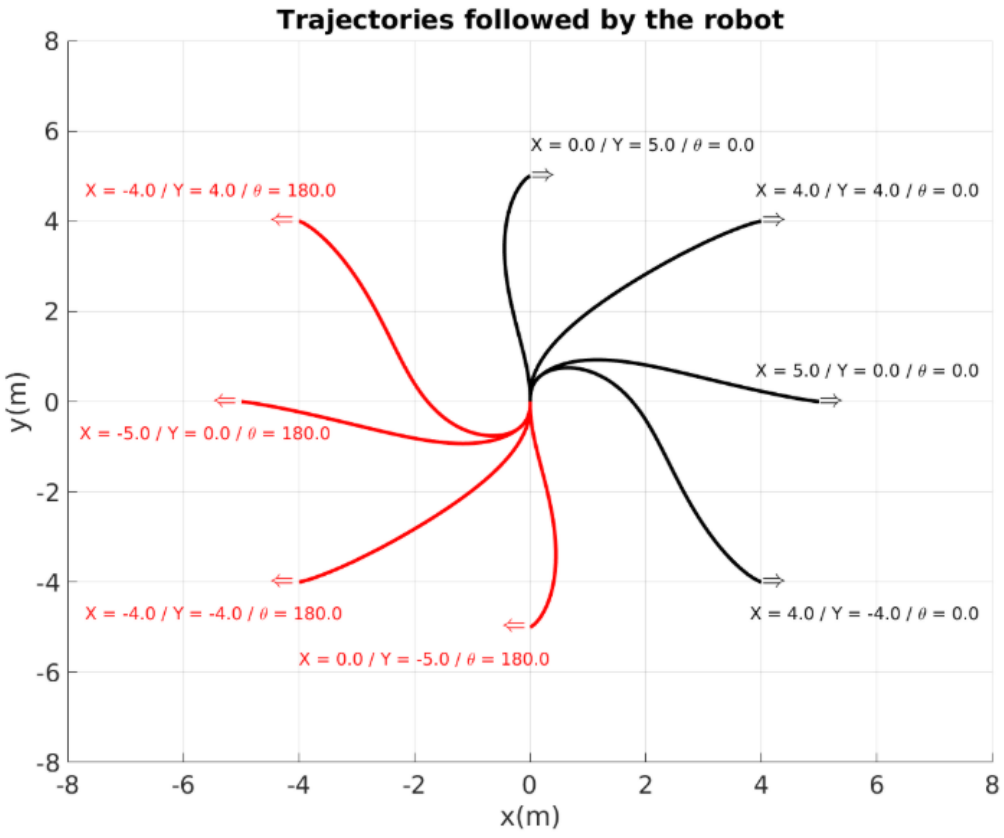

Figure 3: (8 trajectory experiment)

Optional Extension (Bonus points): With IMU

Repeat a subset of experiments while fusing IMU (`sensor_msgs/msg/Imu`) into your pose estimation. Compare convergence speed and steady-state error W/ vs W/O IMU (*More information will be delivered by the TA about this part*).

Post-Lab Questions

1. Derive the discrete feedback equations used in your controller and show where they appear in your code.
2. How does varying each gain $\{k_\rho, k_\alpha, k_\beta\}$ affect motion (overshoot, curvature, convergence)? Support with experiments.
3. Identify any gain values or combinations that lead to instability or oscillation.