Autonomous Mobile Robotics: Dynamic Control, Global Path Planning, and Trajectory Tracking Across Three Robot Platforms

Mayank Pandey

IIT Palakkad

December 4, 2025

Outline

Introduction

Autonomous mobile robots require reliable path planning and accurate trajectory tracking for safe navigation. This project addresses three core problems: **(i)** generating feasible, collision-free paths, **(ii)** designing controllers for smooth and stable trajectory tracking across different robot models, and **(iii)** tracking a recorded trajectory with a mecanum robot.

The work is divided into three independent tasks:

1. **Task A – Skid-Steer Robot: Modeling & Control**
   - Kinematic–dynamic modeling of a 3-DoF skid-steer platform.
   - Nonlinear two-loop controller (position + dynamic loops).
   - Goal: stable convergence to a desired trajectory.
2. **Task B – Unicycle Robot: Planning & Tracking**
   - A* grid-based planner with obstacle avoidance.
   - Smoothed path followed using Pure Pursuit.
   - Goal: smooth, low-error tracking.
3. **Task C – Mecanum Robot: Trajectory Tracking**
   - Tracking of recorded XY trajectory.
   - Goal: Robot follows the recorded trajectory.

Skid Steer Robot - Objective:

- A four wheel skid steering mobile robot is a differential-drive platform with planar motion (x, y, θ).
- **Objective:** Explore two key aspects of robotic motion
  1. Designing and implementation of PD controller.
  2. Perform trajectory tracking objective.
- These two components are implemented and tested independently in Matlab simulation.

Skid Steer Robot - Kinematic Model

The robot pose and body velocities are related as:

$$\dot{\boldsymbol{\eta}} = J(\psi)\boldsymbol{\zeta}, \quad \boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}, \quad \boldsymbol{\zeta} = \begin{bmatrix} u \\ v \\ r \end{bmatrix}$$

$$J(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Desired trajectory:

$$\boldsymbol{\eta}_d(t) = \begin{bmatrix} x_d(t) \\ y_d(t) \\ \psi_d(t) \end{bmatrix},$$

Skid Steer Robot - Error Definition

Position error in inertial frame:

$$\boldsymbol{e}_p = \boldsymbol{\eta}_d - \boldsymbol{\eta}$$

Convert to body frame to decouple orientation:

$$\boldsymbol{e}_b = J^T(\psi)\boldsymbol{e}_p$$

Differentiating:

$$\dot{\boldsymbol{e}}_b = \dot{J}^T(\psi)\boldsymbol{e}_p + J^T(\psi)\dot{\boldsymbol{e}}_p$$

Since $\dot{\boldsymbol{e}}_p = \dot{\boldsymbol{\eta}}_d - \dot{\boldsymbol{\eta}} = \dot{\boldsymbol{\eta}}_d - J(\psi)\boldsymbol{\zeta}$,

$$\boxed{\dot{\boldsymbol{e}}_b = J^T(\psi)\dot{\boldsymbol{\eta}}_d - S(r)\boldsymbol{e}_b - \boldsymbol{\zeta}}$$

where

$$S(r) = \begin{bmatrix} 0 & -r & 0 \\ r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

represents the coupling between rotation rate r and position error.

Skid Steer Robot - Nonlinear Error Dynamics

From the previous slide:

$$\dot{\boldsymbol{e}}_b = J^T(\psi)\dot{\boldsymbol{\eta}}_d - S(r)\boldsymbol{e}_b - \boldsymbol{\zeta}$$

We define a desired body velocity that includes both feedforward and feedback terms:

$$\boxed{\boldsymbol{\zeta}_{des} = J^T(\psi)\dot{\boldsymbol{\eta}}_d + K_p\boldsymbol{e}_b}$$

The actual velocity tracking error:

$$\tilde{\boldsymbol{\zeta}} = \boldsymbol{\zeta}_{des} - \boldsymbol{\zeta}$$

Substitute into $\dot{\boldsymbol{e}}_b$:

$$\boxed{\dot{\boldsymbol{e}}_b = -S(r)\boldsymbol{e}_b - K_p\boldsymbol{e}_b + \tilde{\boldsymbol{\zeta}}}$$

Thus, the outer loop ensures position convergence provided $\tilde{\boldsymbol{\zeta}} \to 0$.

Skid Steer Robot - Dynamic Model

The robot's body dynamics:

$$\boldsymbol{D}(\zeta)\dot{\zeta} + \boldsymbol{n}_v(\zeta) = \boldsymbol{\tau}$$

where

$$\boldsymbol{D} = \begin{bmatrix} m & 0 & -my_{bc} \\ 0 & m & mx_{bc} \\ -my_{bc} & mx_{bc} & I_z + m(x_{bc}^2 + y_{bc}^2) \end{bmatrix}$$

and

$$\boldsymbol{n}_v = \begin{bmatrix} -mr(v + x_{bc}r) \\ mr(u - y_{bc}r) \\ mr(x_{bc}u + y_{bc}v) \end{bmatrix}$$

$\boldsymbol{\tau}$ = generalized body forces (wheel torques)

Skid Steer Robot - Inner Loop Control Law

To drive $\zeta \to \zeta_{des}$, choose

$$\boxed{\boldsymbol{\tau} = \boldsymbol{D}(\zeta)\left[\dot{\zeta}_{des} + K_d(\zeta_{des} - \zeta)\right] + \boldsymbol{n}_v(\zeta)}$$

Then

$$\boldsymbol{D}(\zeta)\dot{\tilde{\zeta}} + K_d\boldsymbol{D}(\zeta)\tilde{\zeta} = 0$$

and $\tilde{\zeta} \to 0$ exponentially.
Substituting into the outer loop:

$$\dot{\boldsymbol{e}}_b = -S(r)\boldsymbol{e}_b - K_p\boldsymbol{e}_b$$

This guarantees asymptotic convergence of $\boldsymbol{e}_b(t) \to 0$.

Using Lyapunov theory we can easily prove this

$$\boxed{\zeta(t) \to \zeta_{des}(t) \quad \Longrightarrow \quad \eta(t) \to \eta_d(t)}$$

Skid Steer Robot - Circular Trajectory Example

Let the desired trajectory be a circle of radius $R = 2$ and angular rate $\omega_d = 0.2$:

$$\begin{aligned} x_d(t) &= R\cos(\omega_d t), \\ y_d(t) &= R\sin(\omega_d t), \\ \psi_d(t) &= \omega_d t + \psi_0 \end{aligned}$$

Then

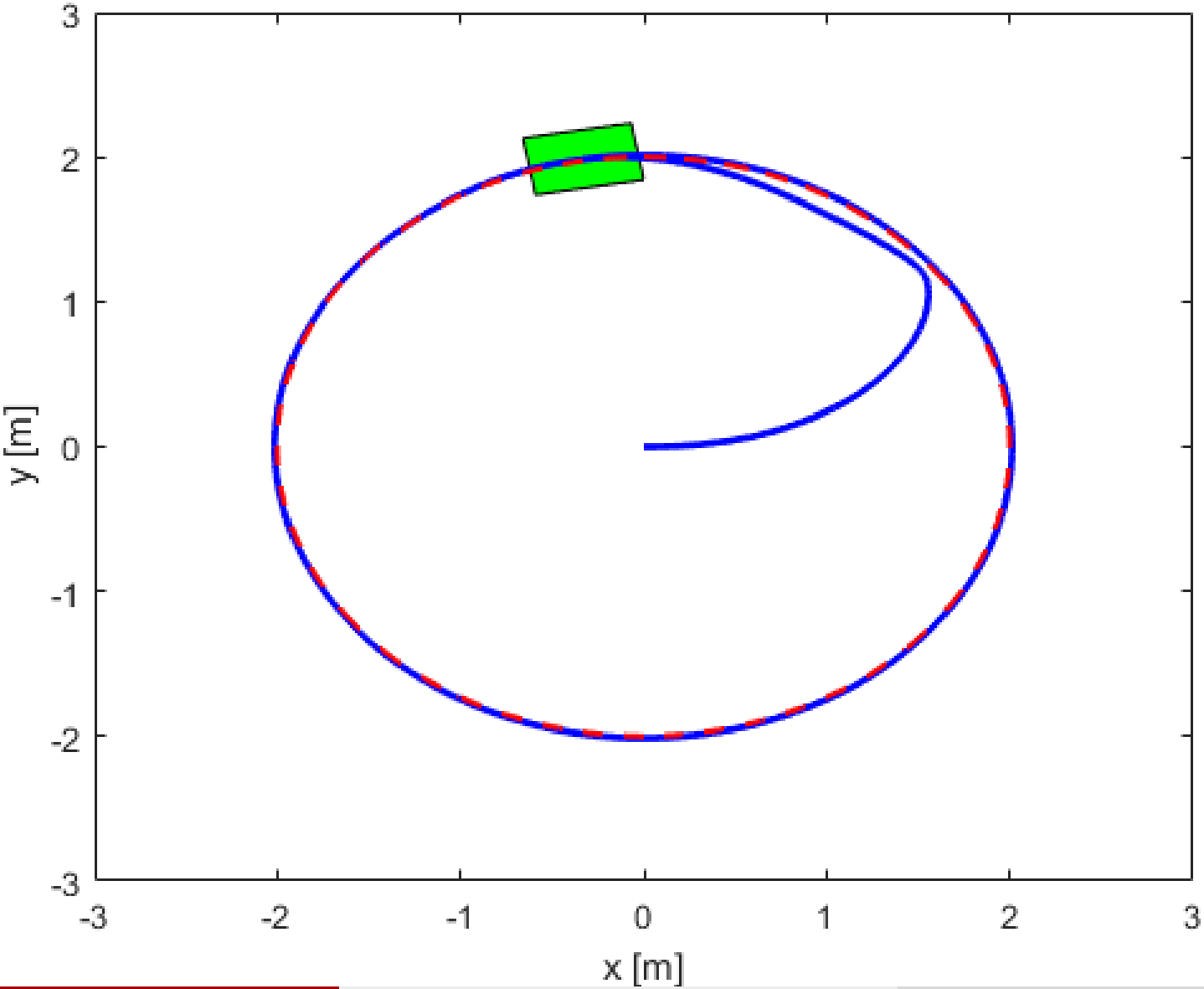
$$\dot{\boldsymbol{\eta}}_d = \begin{bmatrix} -R\omega_d \sin(\omega_d t) \\ R\omega_d \cos(\omega_d t) \\ \omega_d \end{bmatrix}, \quad \ddot{\boldsymbol{\eta}}_d = \begin{bmatrix} -R\omega_d^2 \cos(\omega_d t) \\ -R\omega_d^2 \sin(\omega_d t) \\ 0 \end{bmatrix}$$

Skid Steer Robot - Summary of Control Law

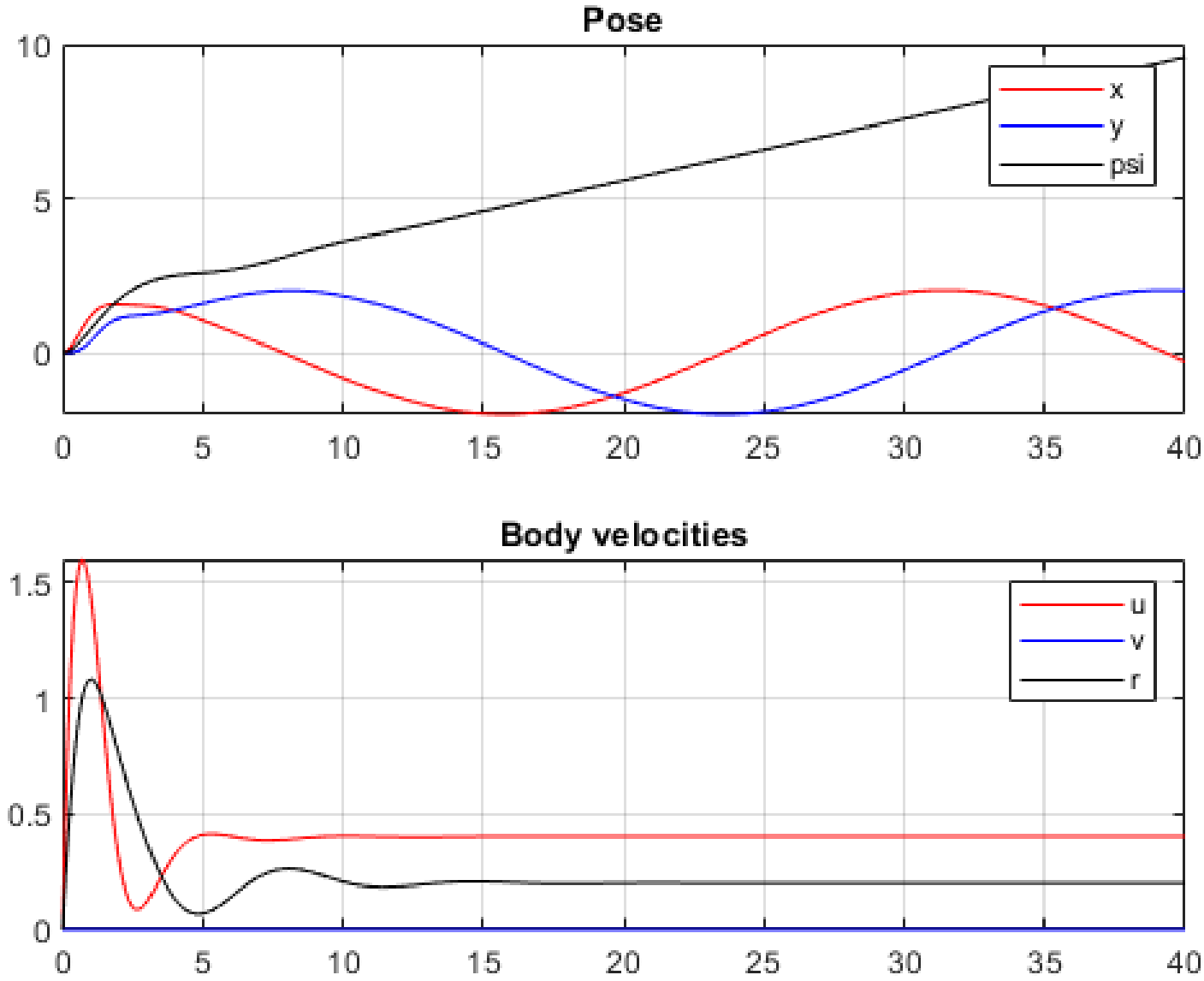
This nonlinear two-loop controller:

- Tracks arbitrary smooth trajectories (including circular motion)
- Compensates for full robot dynamics
- Ensures $\boldsymbol{e}_b, \tilde{\boldsymbol{\zeta}} \to 0$ via Lyapunov stability

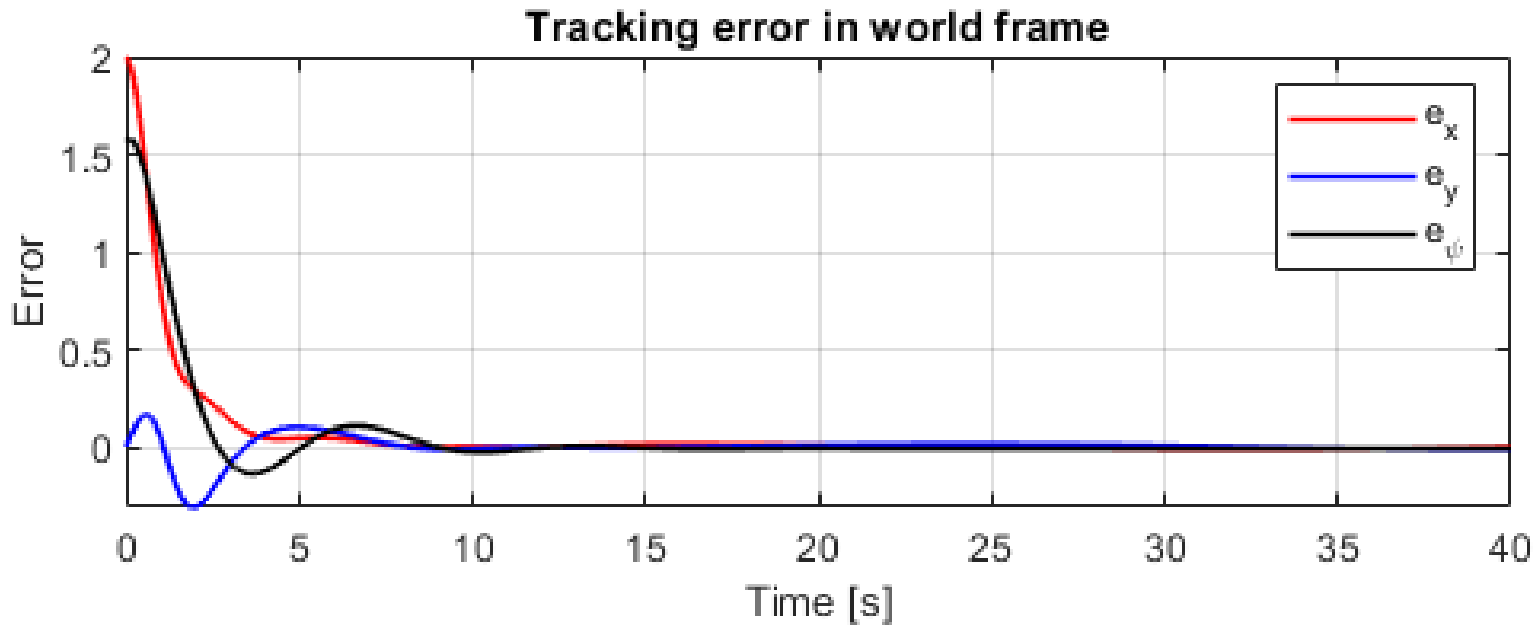
Skid Steer Robot - Robot Pose

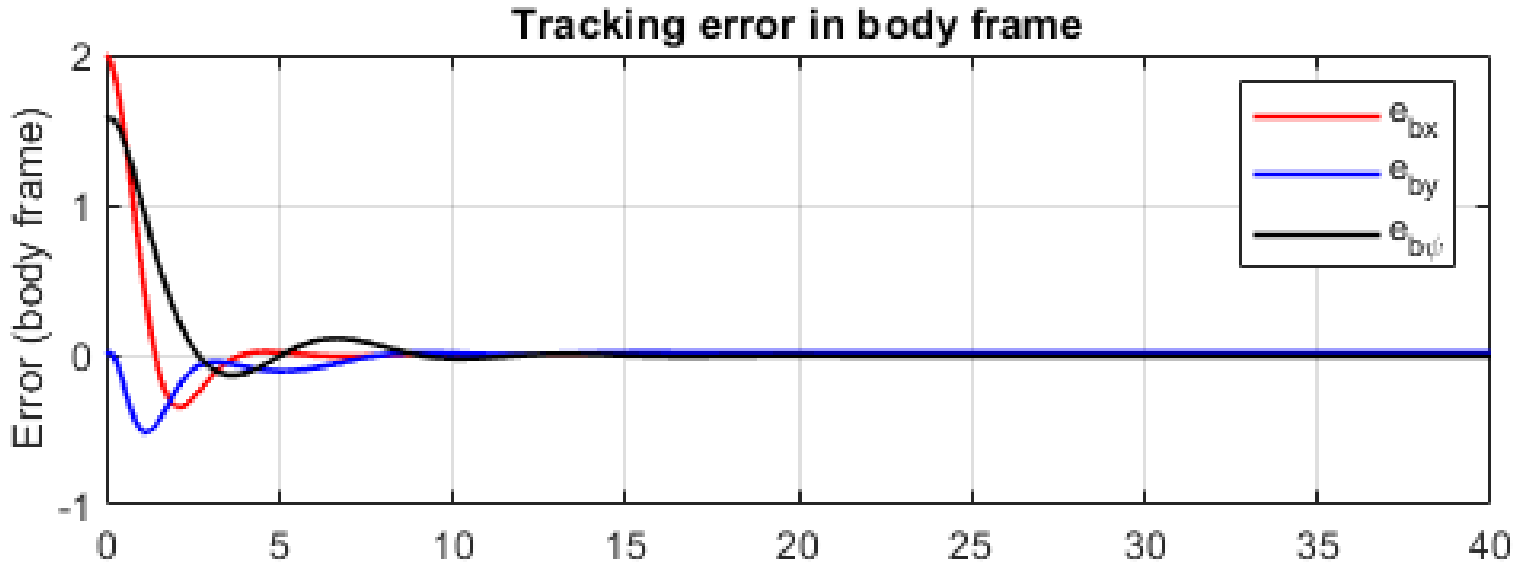


Skid Steer Robot - Robot Pose

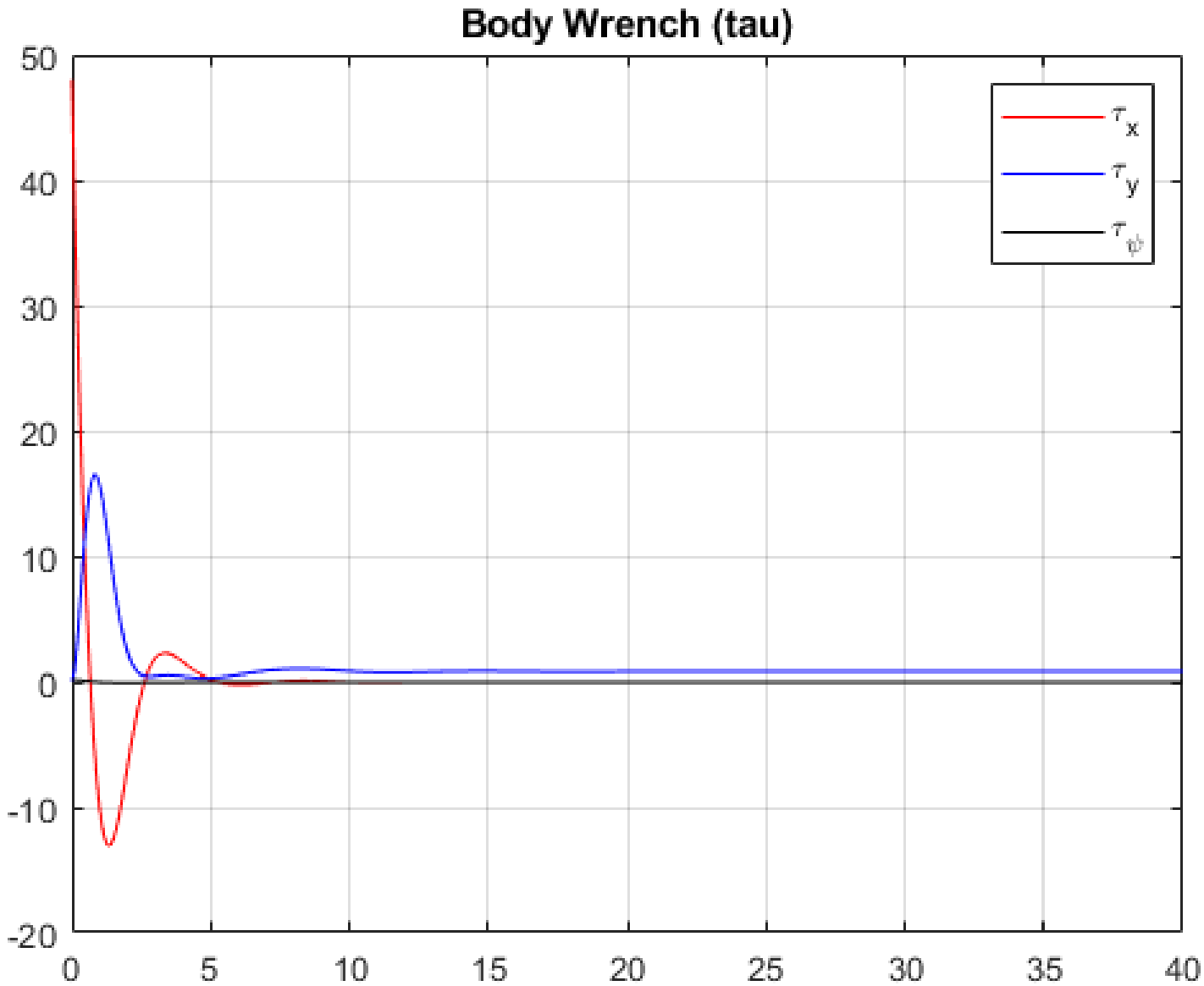


Skid Steer Robot - Tracking Errors





Skid Steer Robot - Control Inputs



Skid Steer Robot - Wheel Inputs

Unicycle Robot - Introduction & Motivation

A unicycle mobile robot is a differential-drive platform with planar motion (x, y, θ).
Objective: Path planning using the **A* algorithm**.

Unicycle Robot - Degrees of Freedom & Kinematic Model

The robot configuration and control inputs are:

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \qquad \boldsymbol{u} = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

The kinematic equations of motion are:

$$\dot{x} = v\cos\theta, \quad \dot{y} = v\sin\theta, \quad \dot{\theta} = \omega$$

This simplified kinematic model assumes pure rolling contact and forms the basis for trajectory tracking simulations.

Part I: Path Planning using A* Algorithm

Objective: Generate a collision-free path between a start and goal position on a 2D occupancy grid.

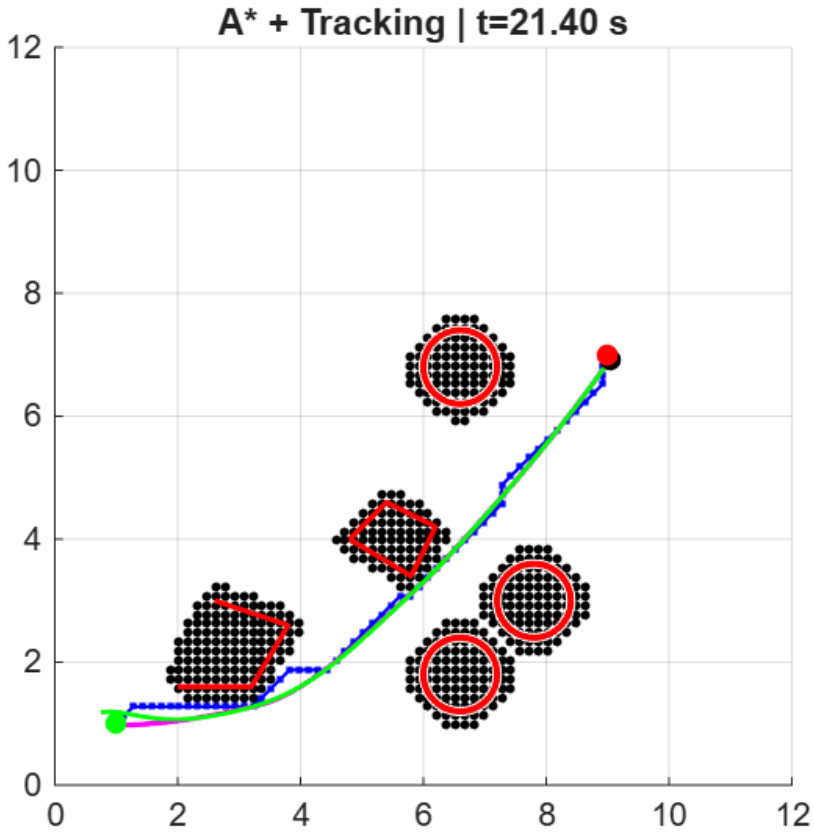
Algorithm:

- The environment is represented as a grid map with obstacles.
- Each node is evaluated using the cost function:

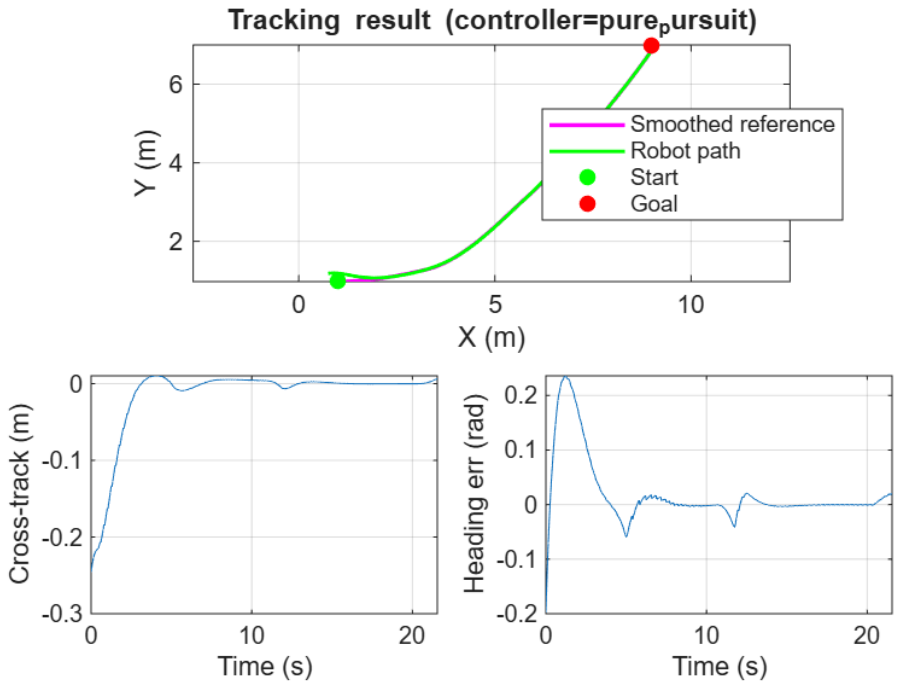
$$f(n) = g(n) + h(n)$$

  where $g(n)$ is the cost from the start, and $h(n)$ is the heuristic (Euclidean distance to goal).
- The node with minimum $f(n)$ is expanded until the goal is reached.

A* Environment and Path Metrics



Environment Map and Paths



Path metrics

Part I: A* Simulation Results & Observations

- The algorithm successfully planned a feasible path around obstacles.
- The number of nodes expanded and computation time were recorded for performance analysis.
- The generated path is discrete and grid-based; further smoothing would be required for physical implementation.
- **Limitations:**
  - Sensitive to map resolution and heuristic scaling.
  - Path not dynamically feasible for a real unicycle robot without smoothing.

Unicycle Robot - Comparison & Discussion

- The A* planner provides a discrete obstacle-free route, suitable for global navigation.
- The Pure Pursuit controller enables smooth local trajectory following when a reference path is known.
- These two components together represent the core of a full autonomous navigation system, even though implemented separately here.
- Future work: integrate both modules by smoothing A* output for tracking and including dynamic constraints.

Unicycle Robot - Implementation Requirements & Conclusion

- **Hardware Requirements:**
  - Sensors: wheel encoders, IMU, GPS or camera for localization.
  - Actuators: two DC motors with differential drive.
  - Controller: Raspberry Pi / Arduino / STM32 for onboard control.
- **Applications:**
  - Warehouse navigation, indoor delivery, and academic robotics.

Problem: Trajectory Tracking with Mecanum Robot

- **Goal**: Make a Mecanum-wheeled robot follow a recorded real-world trajectory.
- **Challenge**:
  - Recorded data is noisy, discontinuous
  - High-frequency acceleration $\rightarrow$ control instability
  - Dynamics mismatch $\rightarrow$ drift
- **Why Mecanum?** Omnidirectional motion, but kinematically complex.

Wheel Kinematics

Wheel angles:

$$\rho = (\theta_1, \theta_2, \theta_3, \theta_4)$$

Relationship between wheel speeds and body velocity:

$$\dot{q}^B = J\dot{\rho},$$

$$J = \frac{r}{4}\begin{bmatrix} -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ \dfrac{1}{l_1+l_2} & -\dfrac{1}{l_1+l_2} & \dfrac{1}{l_1+l_2} & -\dfrac{1}{l_1+l_2} \end{bmatrix}.$$

Wheel speeds from body velocity:

$$\dot{\rho} = J^\dagger \dot{q}^B,$$

$$J^\dagger = \begin{bmatrix} -1 & 1 & l_1+l_2 \\ -1 & 1 & -(l_1+l_2) \\ -1 & -1 & l_1+l_2 \\ 1 & 1 & l_1+l_2 \end{bmatrix}. \tag{5}$$

Dynamic Model

Four-wheel mecanum robot dynamics:

$$M\ddot{\rho} + D\dot{\rho} = \tau + \tau_d. \tag{6}$$

Mass matrix:

$$M = \begin{bmatrix} \Gamma & -m_1 & m_2 - m_1 & m_1 \\ -m_1 & \Gamma & m_2 - m_1 & m_1 \\ m_1 & m_2 - m_1 & \Gamma & -m_1 \\ m_2 - m_1 & m_1 & -m_1 & \Gamma \end{bmatrix},$$

where:

$$m_1 = \frac{I_w}{r^2(l_1 + l_2)}, \quad m_2 = \frac{mr^2}{8}, \quad D = \text{diag}(D_1, D_2, D_3, D_4),$$

$$\Gamma = m_1 + m_2 + I_m.$$

Why It Works: Key Fixes

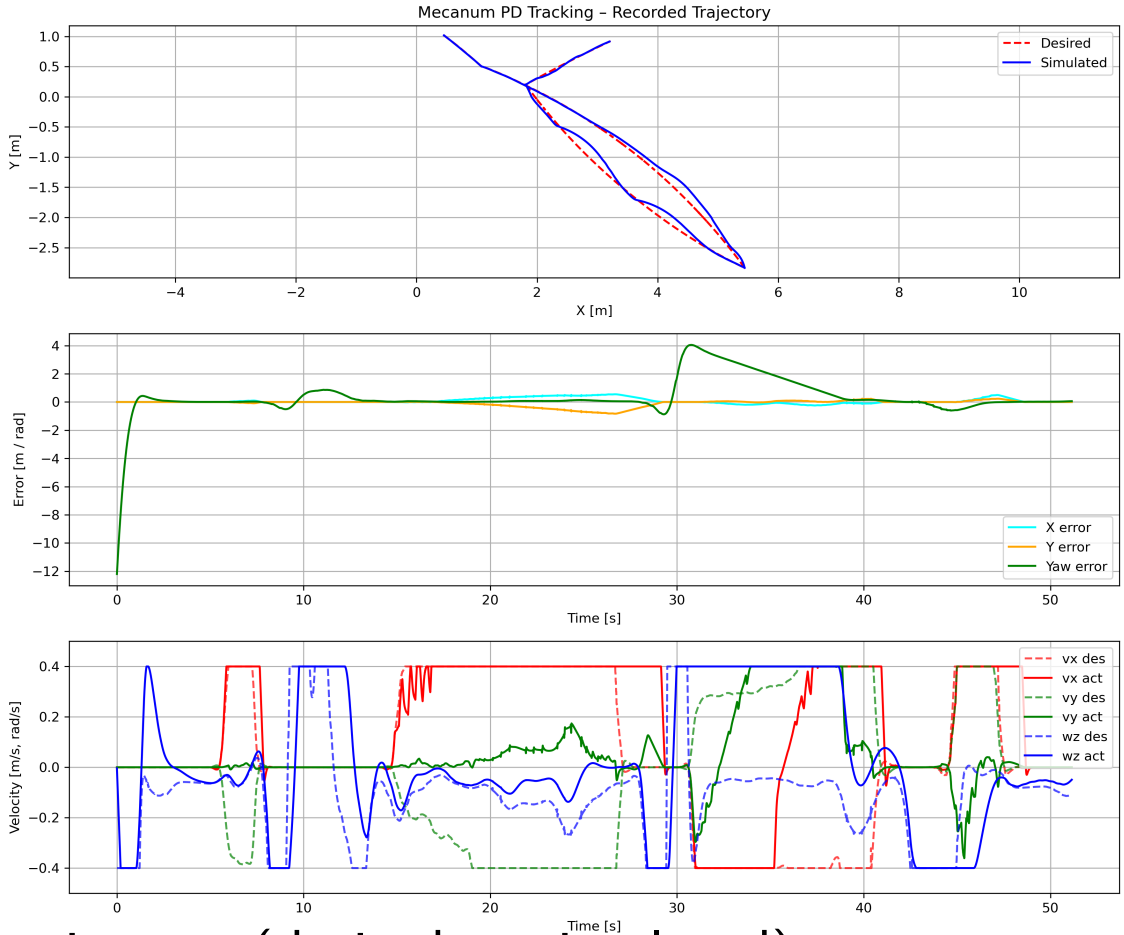
- **Smooth Interpolation**:

$$\text{Butterworth} \rightarrow \text{Cubic Spline}$$

- **Physical Damping**: No artificial decay ($\nu \not= 0.97$)
- **Velocity Clipping**: Respect robot limits

Result: RMS error < 5 cm, stable tracking on real data.

Simulation Results



- **Top**: XY trajectory (desired vs simulated)
- **Middle**: Tracking errors
- **Bottom**: Velocity tracking

Robust tracking despite noisy recorded input.

Overall Conclusion

- The project addressed complementary navigation modules for three robots:
  - **Skid–steer:** dynamic model + nonlinear two–loop control.
  - **Unicycle:** A* global planner + Pure Pursuit tracking.
  - **Mecanum:** reference trajectory and velocity tracking.
- **Key results:**
  - Skid–steer controller ensured stable tracking of the full kinematic–dynamic model.
  - A* generated collision–free paths; Pure Pursuit tracked paths with low error.
  - Mecanum robot accurately followed XY trajectories and maintained acceptable tracking errors despite noise and dynamics mismatch.
- **Takeaway:** The work illustrates the separation between global planning and local motion control which are the core pillars of autonomous navigation.
- **Future work:** integrate planner + controller, enforce dynamic feasibility, and perform hardware validation.

THANK YOU