Mobile Robots

Differential Kinematics and Go-to-Goal Control

Roberto Marino, PhD

Computational Modeling for Robotics and Human Computer Interaction
University of Messina - FCRLab
roberto.marino@unime.it

April 20, 2026



UniMe
1548

From Serial Manipulators to Mobile Robots

Serial manipulators and mobile robots share joint actuators, but their **configuration space** is fundamentally different.

Serial robot

- Fixed base, moving end-effector
- Config. space: $q \in \mathbb{R}^n$ (joint angles)
- Pose from forward kinematics: $T = f(q)$
- Analytic/numeric IK available

Mobile robot

- Moves freely in the environment
- Config. space: $(x, y, \theta) \in \mathbb{R}^2 \times S^1$
- Pose evolves over time
- **No inverse kinematics** in the classical sense

Control emerges from the **geometry of the task**, not from inverting a kinematic chain.

The Pioneer P3-DX: Differential Drive

Mechanical structure

- 2 independent drive wheels (left / right)
- 1 passive caster wheel (front)
- Non-holonomic: cannot translate sideways

Physical parameters

Parameter	Symbol	Value
Wheel radius	r	0.0975 m
Half-wheelbase	L	0.166 m
Full wheelbase	$2L$	0.332 m



Figure: Pioneer robot

Derivation: From Wheels to Robot Velocity

Rolling constraint (no slip): the velocity of each wheel contact point equals the wheel's peripheral speed.

Let $\dot{\phi}_L$, $\dot{\phi}_R$ be the wheel angular velocities. The contact-point velocities are:

$$v_L = r\,\dot{\phi}_L, \qquad v_R = r\,\dot{\phi}_R$$

The robot **linear** and **angular** velocities:

$$v = \frac{v_R + v_L}{2} = \frac{r}{2}\left(\dot{\phi}_R + \dot{\phi}_L\right)$$

$$\omega = \frac{v_R - v_L}{2L} = \frac{r}{2L}\left(\dot{\phi}_R - \dot{\phi}_L\right)$$

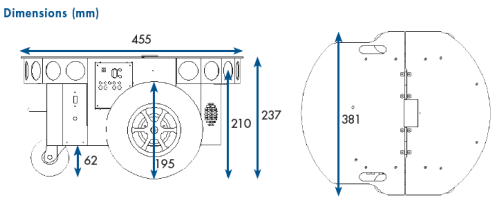


Figure: Pioneer robot

Forward and Inverse Models

$$v = \frac{r}{2}(\dot{\phi}_R + \dot{\phi}_L), \qquad \omega = \frac{r}{2L}(\dot{\phi}_R - \dot{\phi}_L)$$

$$\dot{\phi}_L = \frac{v - \omega L}{r}, \qquad \dot{\phi}_R = \frac{v + \omega L}{r}$$

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

This is a **nonlinear, non-holonomic** system: lateral motion ($\dot{y} \neq 0$ with $\theta = 0$) is not instantaneously achievable.

Odometry: Integrating the Kinematic Model

Odometry estimates the pose (x, y, θ) by integrating wheel velocities over time.

At each time step Δt, given v and ω:

$$\Delta s = v\,\Delta t, \qquad \Delta\theta = \omega\,\Delta t$$

$$x_{k+1} = x_k + \Delta s\,\cos\!\left(\theta_k + \tfrac{\Delta\theta}{2}\right)$$

$$y_{k+1} = y_k + \Delta s\,\sin\!\left(\theta_k + \tfrac{\Delta\theta}{2}\right)$$

$$\theta_{k+1} = \theta_k + \Delta\theta$$

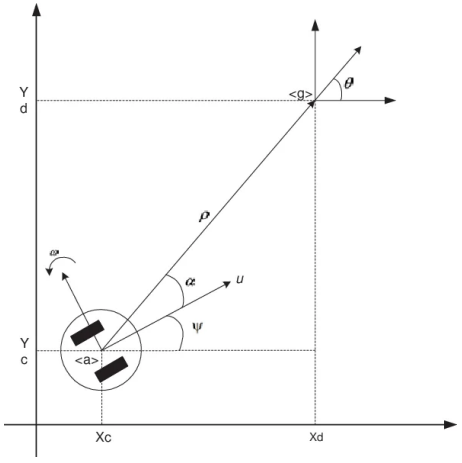
Limitation: odometry is *open-loop* — integration errors accumulate over time (wheel slip, encoder quantisation, parameter uncertainty). External sensors are needed for correction.

Problem: Navigate to a Goal Point

Given: current pose (x, y, θ) and goal (x_g, y_g).
Find: control inputs $v(t)$, $\omega(t)$ that drive $\rho \to 0$.

Why polar coordinates?

- Working with Δx, Δy separately is *unnatural* for a robot that rotates first, then advances.
- Two variables are sufficient:
  - ρ distance to goal
  - α bearing error (heading mismatch)
- Both go to zero $\Rightarrow$ goal reached.



Go-to-Goal: Polar Error Coordinates

Error variables (from current pose and goal):

$$\rho = \sqrt{(x_g - x)^2 + (y_g - y)^2}$$

$$\alpha = \text{atan2}(y_g - y,\ x_g - x) - \theta \quad \in (-\pi,\ \pi]$$

Proportional control law:

$$v = k_\rho\, \rho \cdot \cos^2\alpha, \qquad \omega = k_\alpha\, \alpha$$

with saturation: $v \leftarrow \text{clip}(v,\ 0,\ v_{\max})$, $\omega \leftarrow \text{clip}(\omega,\ -\omega_{\max},\ \omega_{\max})$

The $\cos^2\alpha$ term damps linear velocity when heading error is large — advancing in the wrong direction worsens the situation. As $\alpha \to 0$ the robot accelerates forward naturally.

Stability Condition and Convergence

Convergence (Lyapunov analysis):

- The system converges globally to the goal provided

$$k_\alpha > k_\rho > 0$$

- If $k_\rho > k_\alpha$: robot oscillates — advances faster than it corrects heading.
- Stopping criterion: $\rho < \rho_{\text{tol}}$ (e.g. 5 cm).

Typical gains

Gain	Value	Role
k_ρ	0.4	speed vs distance
k_α	1.5	turn rate vs angle
$v_{\max}$	0.5 m/s	saturation
$\omega_{\max}$	1.5 rad/s	saturation

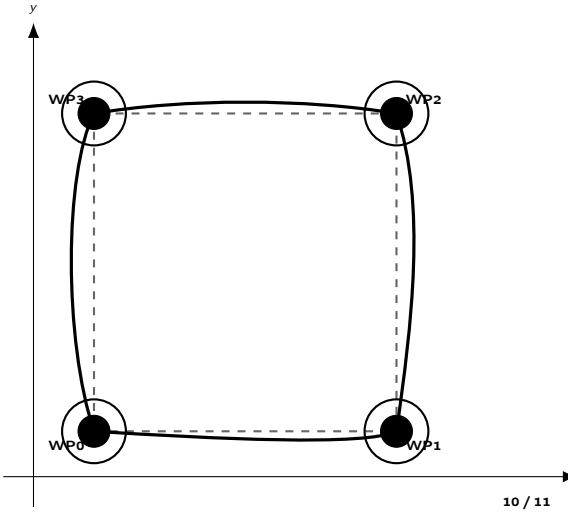
Extension: Multi-Waypoint Trajectory

The go-to-goal controller extends naturally to a sequence of waypoints.

Algorithm

1. Set current goal $= \mathrm{WP}_0$
2. Run go-to-goal until $\rho < \rho_{\mathrm{tol}}$
3. Advance to next waypoint
4. Repeat until last waypoint reached

A smooth velocity profile at waypoint transitions (reduce $v_{\max}$ near each waypoint) avoids abrupt direction changes and reduces odometry drift.



Lab: Pioneer P3-DX in CoppeliaSim

Software stack

CoppeliaSim (scene `Pioneer_p3dx.ttt`) $\xrightarrow{\text{ZMQ Remote API}}$ **Python / Jupyter** (`pioneer_coppelia.ipynb`)

Joint configuration required

In the scene hierarchy: `/PioneerP3DX/leftMotor` and `rightMotor` must have:

- *Dynamic mode*: `Velocity control`
- *Motor enabled*: ✓
- *Maximum torque*: > 0

Lab tasks

1. Verify the kinematic model: measure v and ω for given $\dot{\phi}_L$, $\dot{\phi}_R$.
2. Run the go-to-goal controller to a single point.
3. Navigate a square trajectory; plot odometry vs. ground truth.
4. Vary k_ρ and k_α: observe stability effects.