IA712: Mobile Robotics

Lecture 5: Kinematics

Zhi Yan

ENSTA - Institut Polytechnique de Paris

What is Kinematics?

Definition:

Kinematics is the study of motion without considering the forces and torques that cause it. It describes the robot's position, velocity, and acceleration.

Key assumptions in this lecture:

- The robot is a rigid body.
- The wheels roll without slipping (the "no-slip" condition).
- Motion occurs on a 2D plane.

Objective:

To create a mathematical model that links the actuator speeds (wheel motors) to the velocity of the robot's body.

Forward vs. Inverse Kinematics

For a mobile robot, kinematics helps us answer two fundamental questions:

1. If my wheels turn at a certain speed, how is the robot's body moving through the world?
   Flow: From wheel speeds → to robot velocity.

Forward vs. Inverse Kinematics

For a mobile robot, kinematics helps us answer two fundamental questions:

1. If my wheels turn at a certain speed, how is the robot's body moving through the world?
   Flow: From wheel speeds $\rightarrow$ to robot velocity.
   $\Longrightarrow$ Forward kinematics
2. To make the robot's body move in a desired direction at a certain speed, how fast do I need to turn my individual wheels?
   Flow: From robot velocity $\rightarrow$ to wheel speeds.
   $\Longrightarrow$ Inverse kinematics

Why is this crucial?

Forward kinematics

- It is the basis of odometry (estimating position).
- It allows the robot to know "where it is" based solely on its wheel movements.

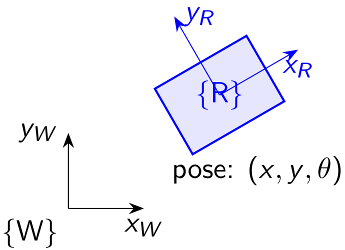
Inverse kinematics

- It is essential for any motor controller that takes velocity commands (like `/cmd_vel`) and translates them into actions.
- It allows the robot to execute a desired trajectory.

Frames of Reference

To describe motion, we must define our coordinate systems, or "frames".

- **Global frame (world frame):** A fixed, external reference frame.
  $\Longrightarrow$ Often called `map` or `odom` in ROS.
- **Local frame (robot frame):** A frame attached to the robot itself, typically at its center of rotation.
  $\Longrightarrow$ Often called `base_link` in ROS.



A robot's **pose** in the world is its pos. (x, y) and ori. θ w.r.t. the global frame.
$\Longrightarrow$ Kinematics allows us to calculate how this pose changes over time.

The Mathematics of Frames: Rotation

The relationship between the robot frame $\{R\}$ and the world frame $\{W\}$ is defined by a **translation** (x, y) and a **rotation** θ.

A point $P_R = (x_R, y_R)$ in the robot frame can be expressed in the world frame $P_W = (x_W, y_W)$ via a rotation.

2D rotation matrix:

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

The transformation is then: $P_W = R(\theta) \cdot P_R$.

Homogeneous Transformation Matrices

To combine rotation AND translation into a single matrix operation, we use homogeneous coordinates.

A point (x, y) becomes a vector $(x, y, 1)^T$.

Homogeneous transformation matrix T:

$$T = \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R(\theta) & \mathbf{p} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

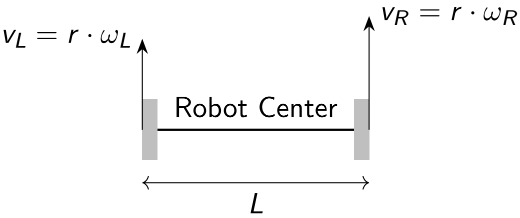
where $\mathbf{p} = (x, y)^T$ is the translation vector.

The full transformation of a point from the robot frame to the world frame is: $P_W = T \cdot P_R$.

The Differential Drive Model

Let's model the robot from Lecture 4:

- Two wheels separated by a distance L.
- Each wheel has a radius r.
- The left and right wheels rotate at angular velocities ω_L and ω_R.



Motion is controlled by varying the relative speed of the two wheels.

Goal: Given the wheel speeds (ω_L, ω_R), what are the linear velocity v and angular velocity ω of the robot's center?

Instantaneous Center of Curvature (ICC)

The motion of any rigid body in a plane can be described as a rotation around a single point, the Instantaneous Center of Curvature (ICC).

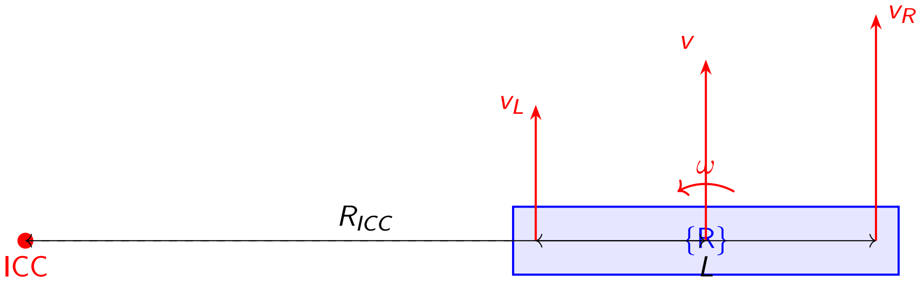
By varying ω_L and ω_R, the robot pivots around this point.

The distance R_{ICC} from the center of the robot to the ICC is given by:

$$\omega = \frac{v}{R_{ICC}} \iff R_{ICC} = \frac{v}{\omega}$$

where ω is the robot's angular velocity and v is its linear velocity.

Instantaneous Center of Curvature (ICC)



Deriving Forward Kinematics

The linear velocity of the robot, v, is the average of the two wheel velocities:

$$v = \frac{v_R + v_L}{2} = \frac{r(\omega_R + \omega_L)}{2}$$

The angular velocity of the robot, ω, is determined by the difference in wheel speeds, causing the robot to rotate around the ICC.

$$\omega = \frac{v_R - v_L}{L} = \frac{r(\omega_R - \omega_L)}{L}$$

Forward kinematic model:

These two equations allow us to map from the wheel velocity space (ω_L, ω_R) to the robot body velocity space (v, ω).

Note: The vector (v, ω) is exactly what a `geometry_msgs/msg/Twist` message in ROS represents!

Forward Kinematics: Special Cases

Moving in a straight line:

If $\omega_L = \omega_R$, then $\omega = 0$.
The robot moves straight ahead with velocity $v = r \cdot \omega_L$.
The ICC is at infinity.

Rotating in place:

If $\omega_L = -\omega_R$, then $v = 0$.
The robot spins about its center point with angular velocity $\omega = \frac{2r \cdot \omega_R}{L}$.
The ICC is at the robot's center.

ENSTA IP PARIS

From Velocity to Pose: Odometry

The forward kinematics model gives us the robot's velocity (v, ω) in its **own frame** $\{R\}$.
To find the new pose (x', y', θ'), we must project this velocity into the **world frame** $\{W\}$ and integrate it.

Odometry equations:

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

Discrete update (Euler approximation):

$$x_{k+1} = x_k + v_k\cos(\theta_k)\Delta t$$
$$y_{k+1} = y_k + v_k\sin(\theta_k)\Delta t$$
$$\theta_{k+1} = \theta_k + \omega_k\Delta t$$

The Problem with Odometry: Drift

Odometry is a form of "dead reckoning".

Sources of error:

- Wheel slippage (uneven ground, rapid acceleration).
- Inaccurate wheel diameter, mechanical backlash.
- Time discretization.

Consequence:

- Errors accumulate without bound over time.
- The position estimate will inevitably drift.
- Odometry is reliable in the short term, but unusable alone in the long term.

Modeling Kinematic Errors & Uncertainty

Professionally, we don't just acknowledge errors, we model them.

Odometry errors are both **systematic** (e.g., one wheel is slightly larger) and **non-systematic** (e.g., random slip).

Error propagation:

The robot's velocity (v, ω) is not known perfectly. We represent this uncertainty with a covariance matrix:

$$\Sigma_v = \begin{pmatrix} \sigma_v^2 & \sigma_{v\omega} \\ \sigma_{v\omega} & \sigma_\omega^2 \end{pmatrix}$$

Σ_v in velocity must be propagated to the robot's pose (x, y, θ).

- Require computing the Jacobian of the motion model.
- The core mathematical basis of the **Extended Kalman Filter (EKF)** used in SLAM.

Deriving Inverse Kinematics

Goal: Given a desired linear velocity v and angular velocity ω, what are the required left and right wheel speeds (ω_L, ω_R)?

We simply need to rearrange our forward kinematics equations:

- $2v = v_R + v_L$
- $\omega L = v_R - v_L$

Solving for v_L and v_R gives us the wheel linear velocities:

$$v_R = v + \frac{\omega L}{2}$$

$$v_L = v - \frac{\omega L}{2}$$

ENSTA IP PARIS

Final Step: To Motor Commands

We have the required linear velocities of the wheels (v_L, v_R). The final step is to convert these back to the angular velocities that a motor controller needs.

Inverse kinematic model:

$$\omega_R = \frac{v_R}{r} = \frac{1}{r}\left(v + \frac{\omega L}{2}\right)$$

$$\omega_L = \frac{v_L}{r} = \frac{1}{r}\left(v - \frac{\omega L}{2}\right)$$

Note: A robot's motor controller in ROS implements exactly these equations to turn a `/cmd_vel` message into motor speeds!

Beyond Differential Drive

Two other important models (cf. Lecture 4):

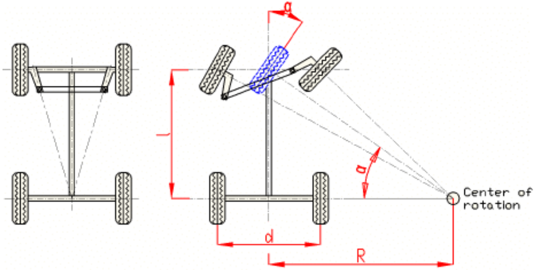
1. **Ackermann steering:** For car-like vehicles with steerable front wheels.
2. **Omnidirectional robots:** Capable of moving instantly in any direction.

Ackermann Kinematic Equations

The ICC lies on the axis of the rear wheel. The distance to the ICC depends on the steering angle α and the wheelbase L_{ack}: $R_{ICC} = \frac{L_{ack}}{\tan \alpha}$.

Forward Kinematics (bicycle model):

$$\dot{x} = v \cos \theta$$
$$\dot{y} = v \sin \theta$$
$$\dot{\theta} = \omega = \frac{v}{R_{ICC}} = \frac{v}{L_{ack}} \tan \alpha$$



$\Longrightarrow$ The command is (v, α), and the state is (x, y, θ).

Mecanum Wheel Kinematics

The kinematics are typically expressed in matrix form.

Inverse kinematics:

Calculates the speeds of the 4 wheels $(\omega_1, \ldots, \omega_4)$ from the body velocity command $(\dot{x}_R, \dot{y}_R, \omega_R)$ in the robot frame:

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} = \frac{1}{r} \begin{pmatrix} 1 & -1 & -(L_x + L_y) \\ 1 & 1 & (L_x + L_y) \\ 1 & 1 & -(L_x + L_y) \\ 1 & -1 & (L_x + L_y) \end{pmatrix} \begin{pmatrix} \dot{x}_R \\ \dot{y}_R \\ \omega_R \end{pmatrix}$$

where L_x and L_y are the half-distances between the wheels.

Legged Robot Kinematics

- **Increased complexity:** Unlike a wheeled robot, a legged robot's "base" is not fixed relative to its actuators.
- **Approach:** Each leg is modeled as a **serial manipulator** (a kinematic chain).
- **Forward kinematics:** Given the angles of each joint (motor) in the leg, where is the foot tip relative to the robot's body?
- **Inverse kinematics:** To place the foot at a desired location, what angles must the joints have?

ENSTA IP PARIS

The Role of Leg Kinematics in Locomotion

Why do we care about the foot position?

The Role of Leg Kinematics in Locomotion

Why do we care about the foot position?
$\Longrightarrow$ Stability!

- **Forward kinematics:** Determine the current locations of all feet on the ground.
  - These ground-contact points form a **Support Polygon**.
  - For the robot to be statically stable (i.e., not fall over when standing still), its **Center of Mass (CoM)** must be projected vertically inside this polygon.

- **Inverse kinematics:** Calculate the joint angles needed to place a swinging foot in a new location, creating a new support polygon and allowing the robot to move forward.

The Need for a Formal Description

So far, the kinematic parameters (r, L) are just numbers in our equations. But how does the entire ROS ecosystem know about the physical structure of our robot?

- How does a visualization tool like RViz know how to draw the robot?
- How does the system know where the LiDAR sensor is relative to the wheels?

Describing the Robot to ROS: URDF

URDF (Unified Robot Description Format):

An XML file format used to describe all the physical elements of a robot: its links and its joints.

- `<link>`: The rigid parts of the robot (chassis, wheel).
- `<joint>`: The kinematic relationship between two `links`.

Example of a wheel joint:

```
<joint name="left_wheel_joint" type="continuous">
<parent link="base_link"/>
<child link="left_wheel_link"/>
<origin xyz="0 0.15 -0.05" rpy="0 0 0"/>
<axis xyz="0 1 0"/>
</joint>
```

Managing Frames in ROS: tf2

Once the robot is described in URDF, ROS needs to know how the pose of each `<link>` changes over time.

tf2: The ROS transform library

- TF2 manages the relationship between all coordinate frames in the system.
- It maintains a tree of all transformations (e.g., `map` → `odom` → `base_link` → `lidar_link`).
- A node like `robot_state_publisher` uses the joint states and the URDF to continuously publish the tf2 transforms.
- This allows any other ROS node to ask "What is the LiDAR's position in the `map` frame?" at any time.

Questions?

Next: Practical Work 5 - tf2 & URDF