Multi-Robot Systems

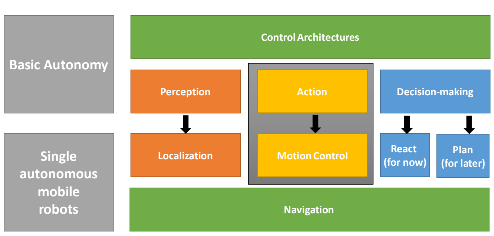
Lecture 3: Motion Control

Mr. Liangyawei Kuang
lkuang@connect.ust.hk
www.deadsecond.com

In this Lecture

- How to control?
- How to model?
- Kinematics
- Trajectory tracking
- Open loop and close loop

Control Architectures



Actuators

- Different purposes
  - Locomotion: e.g., wheeled, legged, slip stick
  - Other motion: e.g., manipulation
  - Other types of actuation: e.g, heating, sound emission
- Examples of electrical-to-mechanical actuators:
  - DC motors, stepper motors, servos, loudspeakers.
  - **Control input** example:

A driver can steer and accelerate (or decelerate), so there are **2** control inputs.

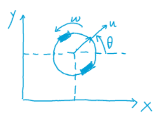
- Uncertainty /disturbances /noise:
  - Examples: wheel *slip*, *slack* in mechanism, *cheap* circuitry with imperfections, *environmental* factors (wind, friction, etc.).

An example of wheeled robot

Degrees of Freedom (DOF)

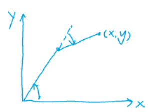
- Most actuators control a single degree of freedom (DOF)
  - a motor shaft controls one rotational DOF
  - a sliding part on a plotter controls one translational DOF
- Every robot has a specific number of DOF
- If there is an actuator for every DOF, then all DOF are controllable

DOF = ?



differential-drive robot

DOF = ?



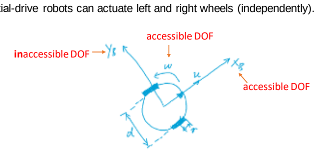
fixed rotating joint

Holonomic Motion

- Degree of mobility: DOM (*differentiable* DOF)
  - Number of DOF that can be **directly accessed** by the actuators
  - A robot in the plane has at most 3 DOMs (position and heading)
- Holonomic motion:
  - **Holonomic robot:** When the number of DOF is equal to robot's DOM
  - **Non-holonomic robot:** When the number of DOF is greater than robot's DOM
  - When a robot's DOM it is larger than is DOF, the robot has 'redundant' actuation

Differential-Drive Robot

- Differential-drive robots can actuate left and right wheels (independently).



- DOF = 3, but DOM = 2: differential-drive robots are **non**-holonomic.
- Are these robots holonomic: Trains? Cars? Quadrotors?
- Impact of non-holonomicity: motion constraints affect motion planning.

Wheeled Robots

- 5 basic types of 3-wheel configurations:

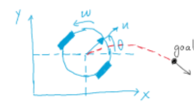
Omnidirectional DOM = 3 | Differential DOM = 2 | Omni-steer DOM = 3 | Tricycle DOM = 2 | Two-steer DOM = 3

An example

https://www.youtube.com/watch?v=_tmiu1wpp_E

Kinematics

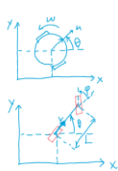
- **Forward** kinematics:
  - Given the control parameters (e.g., wheel velocities), and the time of movement *t*, **find the pose** (*x, y,* θ) reached by the robots.
- **Inverse** kinematics:
  - Given the final desired pose (*x, y,* θ), **find the control parameters** to move the robot there at a given time *t*.



Forward Kinematics

- Differential equations describe robot motion
- How does robot state change over time as a function of control inputs?

differential-drive model
3 DOF (2 controllable)



$$\begin{cases} \dot{x} = u \cdot \cos\theta \\ \dot{y} = u \cdot \sin\theta \\ \dot{\theta} = w \end{cases}$$

bicycle model
3 DOF (2 controllable)

$$\begin{cases} \dot{x} = v \cdot \cos\theta \\ \dot{y} = v \cdot \sin\theta \\ \dot{\theta} = v \cdot \frac{\tan\varphi}{L} \end{cases}$$

A Second-Order Model

- When a first-order model (kinematics) is not enough…
- Differential equations for modeling the dynamics of a quadrotor

$$\begin{cases} \ddot{r} = -g Z_W + \frac{u_1}{m} Z_B \\ \dot{\omega} = I^{-1}\left(-\omega \times I\omega + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}\right) \end{cases}$$

inertia matrix

quadrotor model
6 DOF (4 controllable)

Forward Kinematics (body frame)

Actuators of differential-drive:

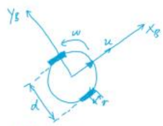
- Left wheel speed $\dot{\phi}_l$
- Right wheel speed $\dot{\phi}_r$

Forward velocity:

$$u = \frac{r\dot{\phi}_r}{2} + \frac{r\dot{\phi}_l}{2}$$

Rotational velocity:

$$w = \frac{r\dot{\phi}_r}{d} - \frac{r\dot{\phi}_l}{d}$$

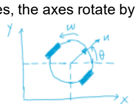


Motion: $\dot{x}_B = u$
$\dot{y}_B = 0$
$\dot{\theta}_B = w$

Forward Kinematics (world frame)

- Given known control inputs, how does the robot move w.r.t. a **global coordinate system**?
- Use a **rotation matrix:**
  - From body to world frames, the axes rotate by θ

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{T(\theta)} \begin{bmatrix} \dot{x}_B \\ \dot{y}_B \\ \dot{\theta}_B \end{bmatrix}$$



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ 0 \\ w \end{bmatrix} = \begin{bmatrix} u\cos\theta \\ u\sin\theta \\ w \end{bmatrix}$$

Inverse Kinematics

- We would like to control the robot motion in the world frame: $\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$
- We **invert** the previous equations to **fifind control inputs**:

$$\begin{bmatrix} u \\ 0 \\ w \end{bmatrix} = T^{-1}(\theta) \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

- yielding $\begin{cases} u = \dot{x}\cos\theta + \dot{y}\sin\theta \\ w = \dot{\theta} \end{cases}$
- under the **constraint** (remember than our robot is non-holonomic):

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

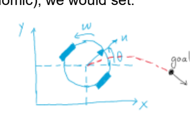
- and finally

$$\begin{aligned} \dot{\phi}_l &= u - \frac{wd}{2r} \\ \dot{\phi}_r &= u + \frac{wd}{2r} \end{aligned} \Rightarrow \begin{aligned} \dot{\phi}_l &= \dot{x}\cos\theta + \dot{y}\sin\theta - \frac{\dot{\theta}d}{2r} \\ \dot{\phi}_r &= \dot{x}\cos\theta + \dot{y}\sin\theta + \frac{\dot{\theta}d}{2r} \end{aligned}$$

Inverse Kinematics

- We would like to control the robot to reach a goal pose: $\begin{bmatrix} x_G \\ y_G \\ \theta_G \end{bmatrix}$
- Ideally (if the robot would be holonomic), we would set:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \underset{\text{control gain}}{K} \begin{bmatrix} x_G - x \\ y_G - y \\ \theta_G - \theta \end{bmatrix}$$

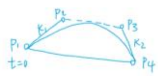


- However, we need to satisfy the non-holonomic constraint:

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

Example of Trajectory Generation

- To satisfy our constraint, we need to be creative. There are various ways of solving this (e.g., differential flatness).
- Cubic Bézier curves, for example, would satisfy our differential drive constraint
- Ensure that robot waypoints lie on a feasible trajectory.
- We set:

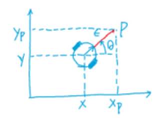


$$P_1 = \begin{bmatrix} x \\ y \end{bmatrix} \quad P_2 = \begin{bmatrix} x + K_1 \cos\theta \\ y + K_1 \sin\theta \end{bmatrix} \quad P_3 = \begin{bmatrix} x_G + K_2 \cos\theta_G \\ y_G + K_2 \sin\theta_G \end{bmatrix} \quad P_4 = \begin{bmatrix} x_G \\ y_G \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbb{B}(t \mid P_1, P_2, P_3, P_4) \text{ with curvature: } \theta = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2}$$

Feedback Linearization

- Leverage linear control of a holonomic point *P* to control a nonholonomic robot.



idea: formulate control inputs u, w as a function of $\dot{x}_p$ & $\dot{y}_p$

Feedback Linearization

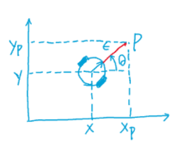
- Feedback linearization:

$$\begin{cases} x_p = x + \epsilon\cos\theta \\ y_p = y + \epsilon\sin\theta \end{cases} \Longrightarrow \begin{cases} \dot{x}_p = \dot{x} + \epsilon(-\dot{\theta}\sin\theta) \\ \dot{y}_p = \dot{y} + \epsilon(\dot{\theta}\cos\theta) \end{cases}$$

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = u \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} + \epsilon w \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

- Isolated control inputs:

$$u = \dot{x}_p\cos\theta + \dot{y}_p\sin\theta$$
$$w = \epsilon^{-1}(-\dot{x}_p\sin\theta + \dot{y}_p\cos\theta)$$



Feedback Linearization

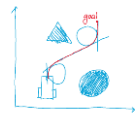
- Trajectory tracking:

$$\dot{x}_p = (x_d - x_p)\underbrace{k}_{\text{constant number}} + \dot{x}_d$$

if desired point x_d is NOT stationary

Trajectory Tracking

- Trajectory tracking:
  1. Pre-compute a smooth trajectory
  2. Follow trajectory (in open-loop or closed-loop)



- Challenges:
  - Feasibility of trajectory given motion constraints
  - Adaptation of trajectory in dynamical environments
  - Must guarantee smoothness of resulting trajectories (kinematic / dynamic feasibility):

E.g., continuity of 1st derivative for 1st order control!

Open-Loop and Closed-Loop

- Once we have a trajectory that enables the robot to reach its goal, we need to follow that trajectory.
- There are two ways of doing this:
  - **Open-loop control:**

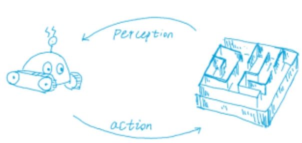
Robot follows path blindly by applying the pre-computed control inputs

- **Closed-loop control:**

Robot can follow path for a small duration, then observe if anything changed in the world, recompute a new adapted path (repeatedly)

Perception-Action Loop

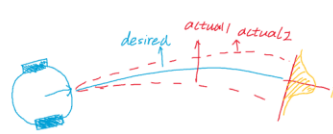
- Basic building block of autonomy



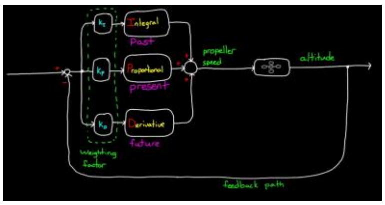
open-loop
vs
closed-loop

Open Loop

- Example: trajectory tracking
- In open-loop, the robot executes predefined control inputs.



Close Loop: a PID example



https://www.youtube.com/watch?v=wkfEZmsQqiA

Open-Loop vs Closed-Loop

- Closed-loop is much more robust to external perturbation:
  - Noisy sensors: wrong estimate of the goal position, wrong estimate of the robot position.
  - Noisy actuation: robot does not move precisely.
  - Unforeseen events
  - Dynamic obstacles
- Open-loop is only useful when feedback is not possible:
  - Sensors cannot operate in certain circumstances
  - Limited bandwidth
  - Limited computational resources