Mobile Robot Systems

Lecture 3: Robot Motion & Control

Dr. Amanda Prorok

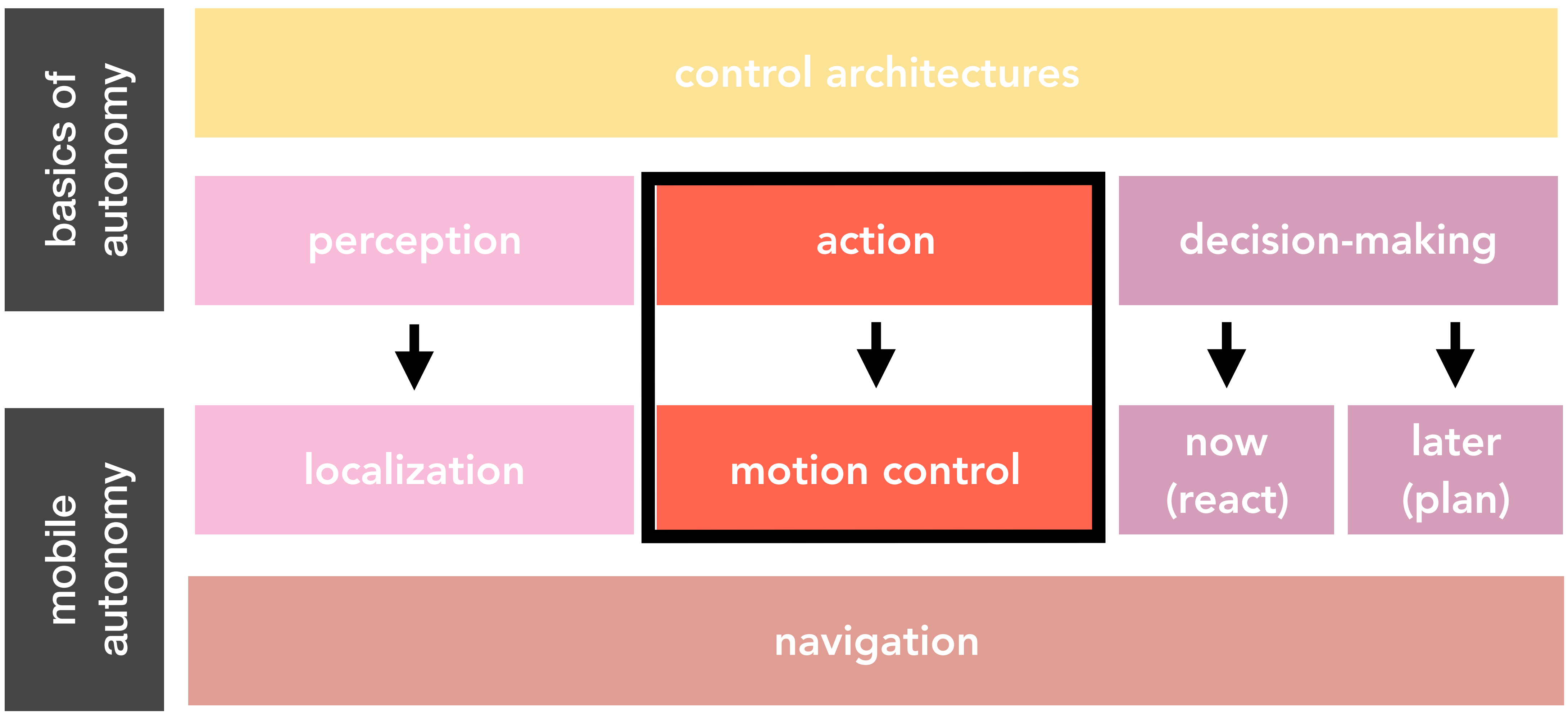
asp45@cam.ac.uk

www.proroklab.org

In this Lecture

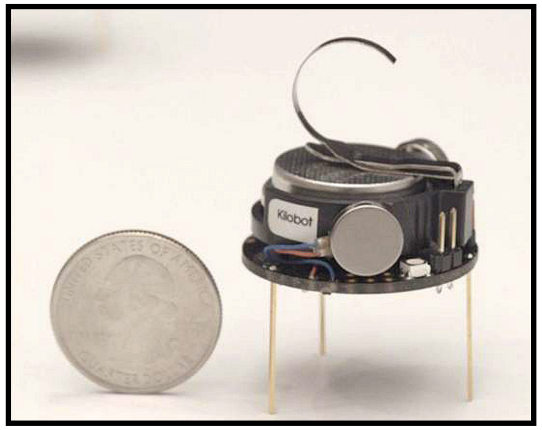
- How can we control mobile robots?
- Motion models
- Forward kinematics; inverse kinematics
- Trajectory tracking
- Open-loop versus closed-loop control
- Introduction to PID control

Control Architectures



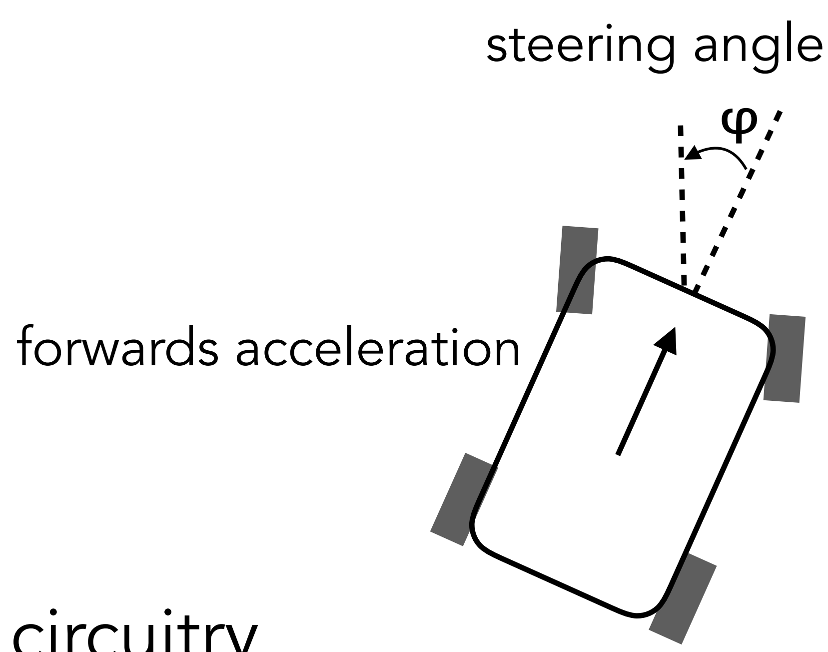
Actuators

- Different purposes
  - Locomotion: e.g., wheeled, legged, slip stick
  - Other motion: e.g., manipulation
  - Other types of actuation: e.g, heating, sound emission



Nagpal et al.: Kilobot

- 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.

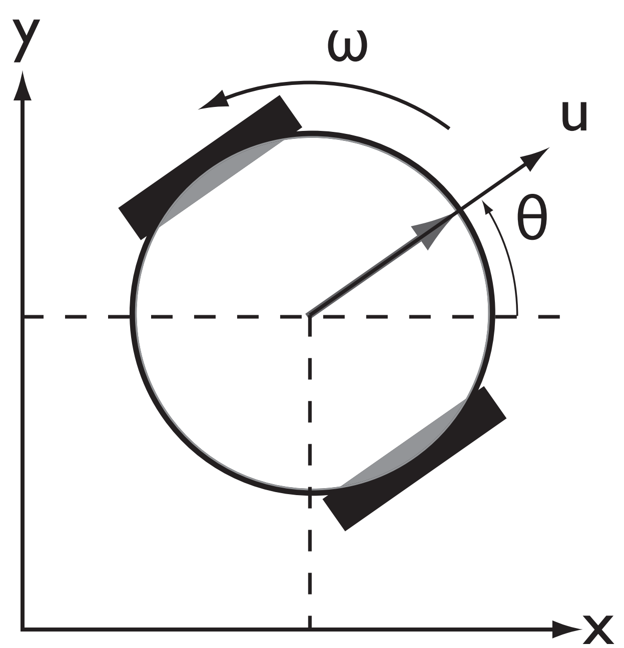


- Uncertainty /disturbances /noise:
  - Examples: wheel *slip*, *slack* in mechanism, *cheap* circuitry with imperfections, *environmental* factors (wind, friction, etc).

Degrees of Freedom

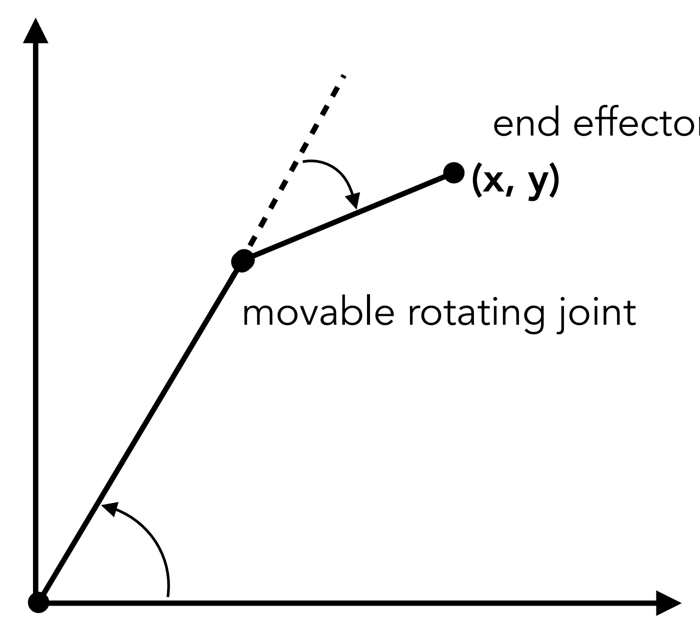
- Most actuators control a single degree of freedom (DOF)
  - a motor shaft controls one rotational DOF
  - a sliding part on a plotter controls one translational DOF
- Every robot has a specific number of DOF
- If there is an actuator for every DOF, then all DOF are controllable

DOF = ?



differential-drive robot

DOF = ?



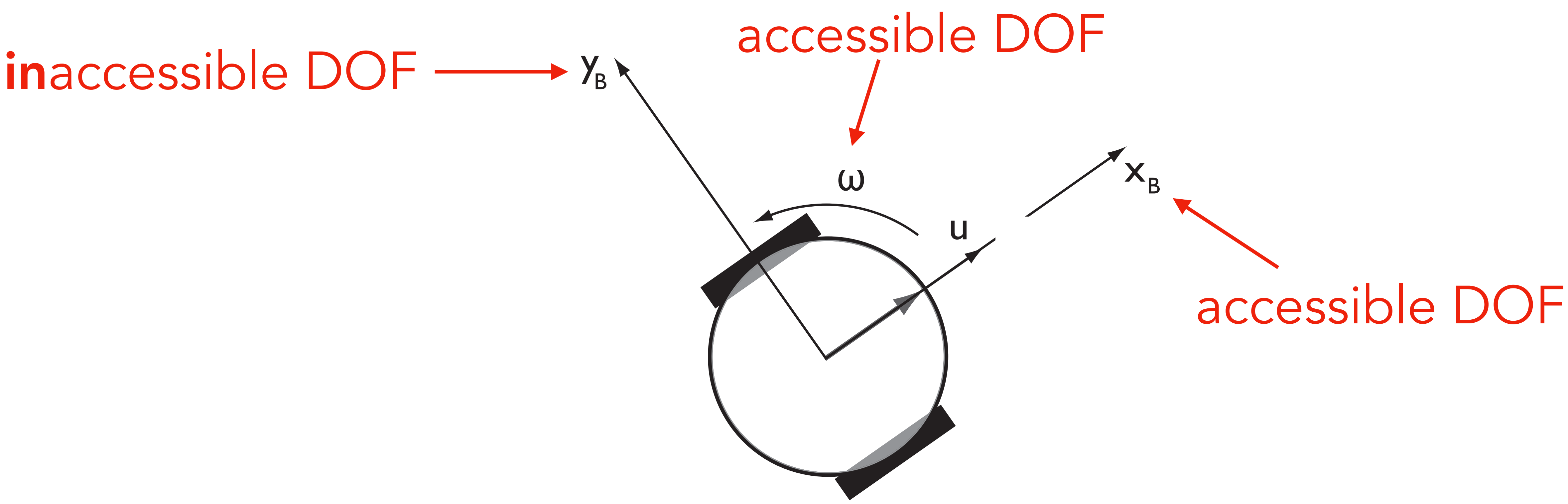
fixed rotating joint

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

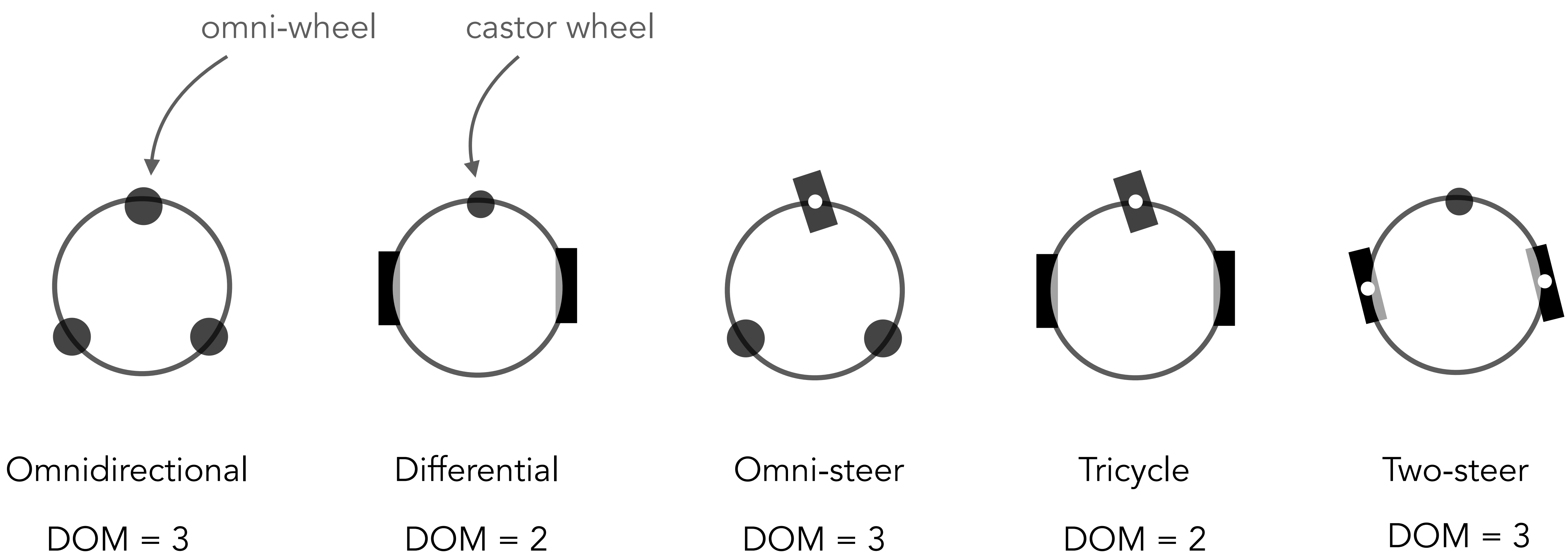
- Differential-drive robots can actuate left and right wheels (independently).

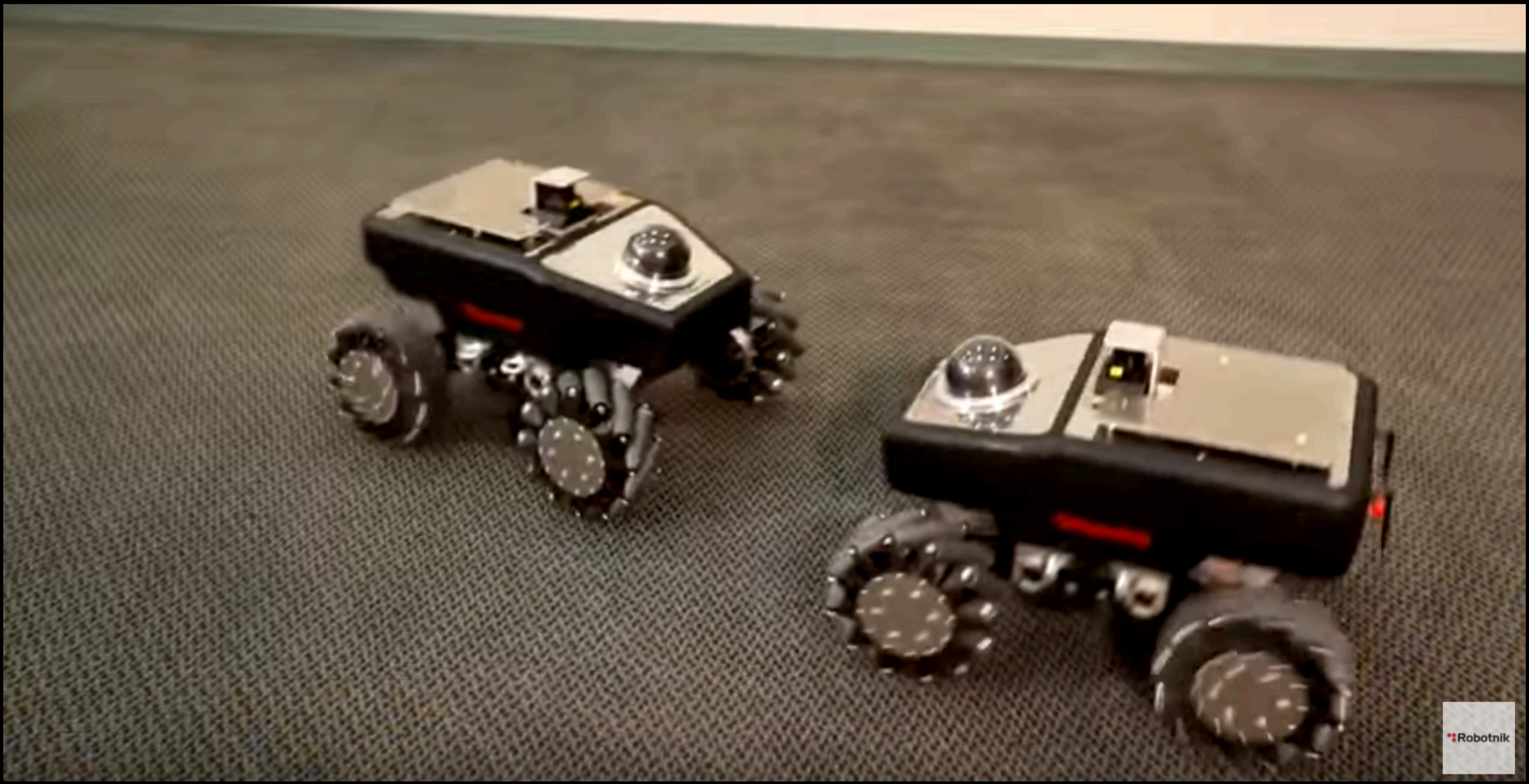


- DOF = 3, but DOM = 2: differential-drive robots are **non**-holonomic.
- Are these robots holonomic: Trains? Cars? Quadrotors?
- Impact of non-holonomicity: motion constraints affect motion planning.

Wheeled Robots

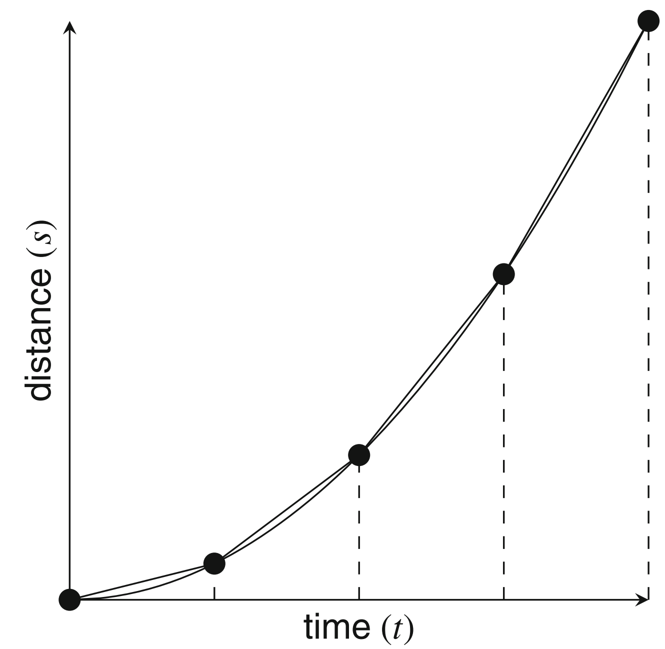
- 5 basic types of 3-wheel configurations:




Robotnik

Distance, Velocity, Time

- Segments:



$$v_i = \frac{\Delta s_i}{\Delta t_i}$$

$$a_i = \frac{\Delta v_i}{\Delta t_i}$$

- Continuous motion: For infinitesimally small segments, we get acceleration and speed at a single point in time (instantaneous), expressed as a derivative.

- Instantaneous speed and acceleration:

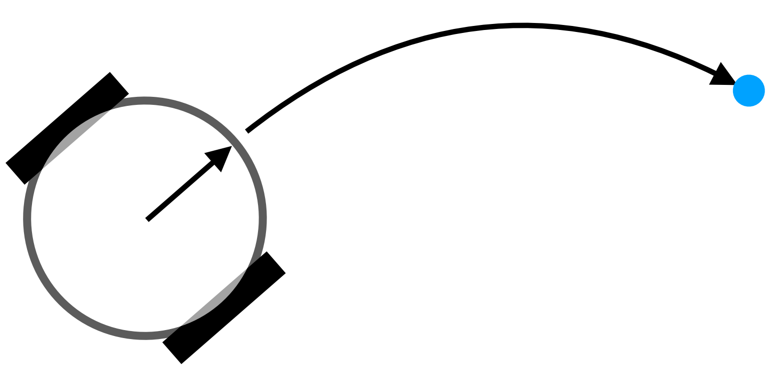
$$v = \frac{ds}{dt} = \dot{s}$$

$$a = \frac{dv}{dt} = \dot{v}$$

* image credit: Elements of Robotics

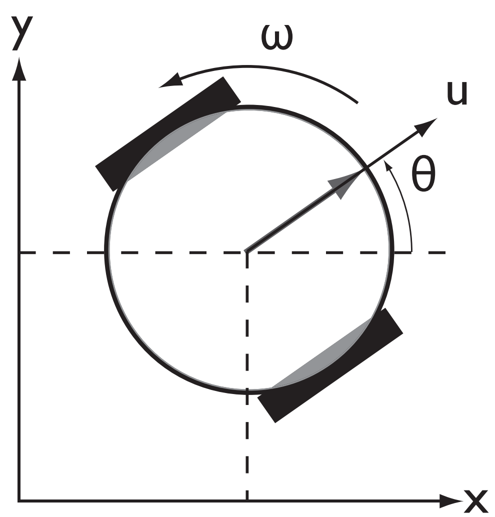
Kinematics

- **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.



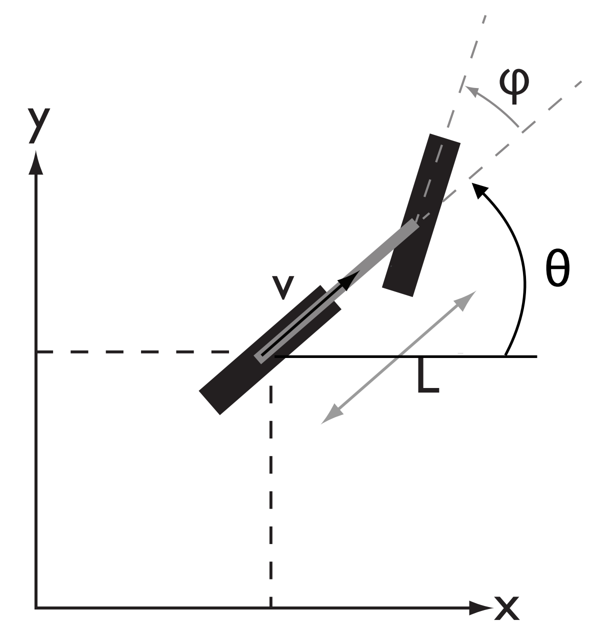
Forward Kinematics

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



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

differential-drive model
3 DOF (2 controllable)

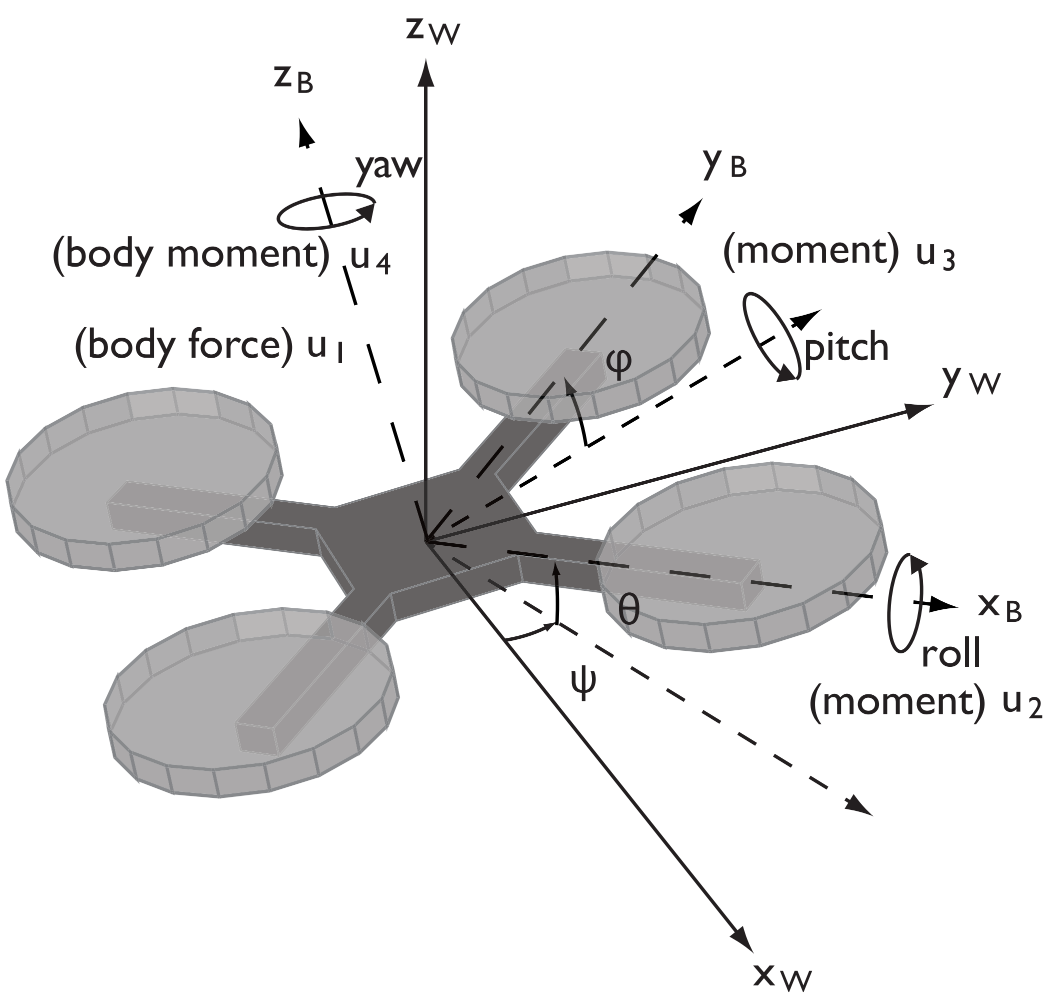


$$\begin{cases} \dot{x} &= v \cdot \cos\theta \\ \dot{y} &= v \cdot \sin\theta \\ \dot{\theta} &= v \cdot \frac{\tan\phi}{L} \end{cases}$$

bicycle model
3 DOF (2 controllable)

A Second-Order Model

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



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

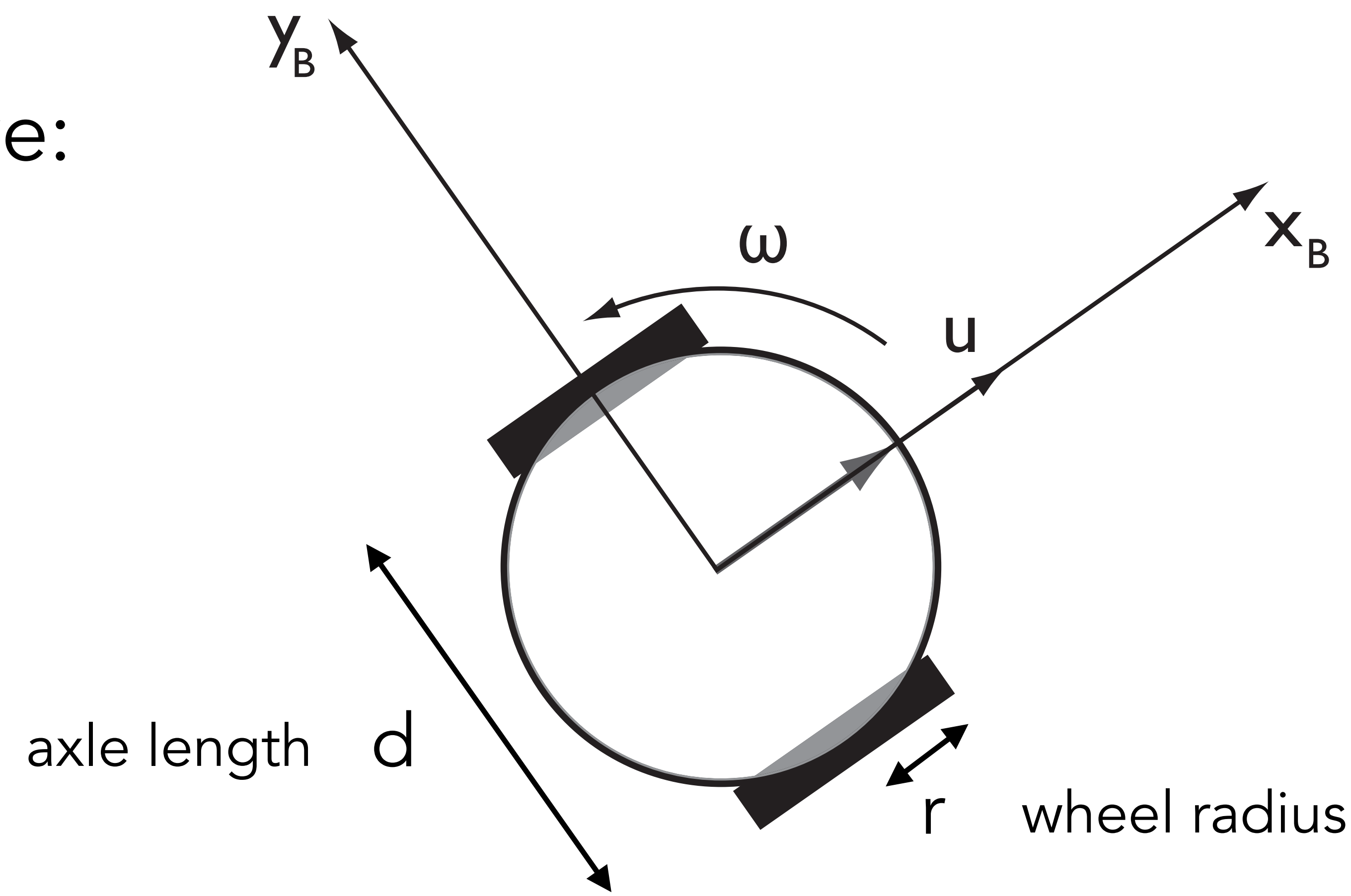
inertia matrix

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:

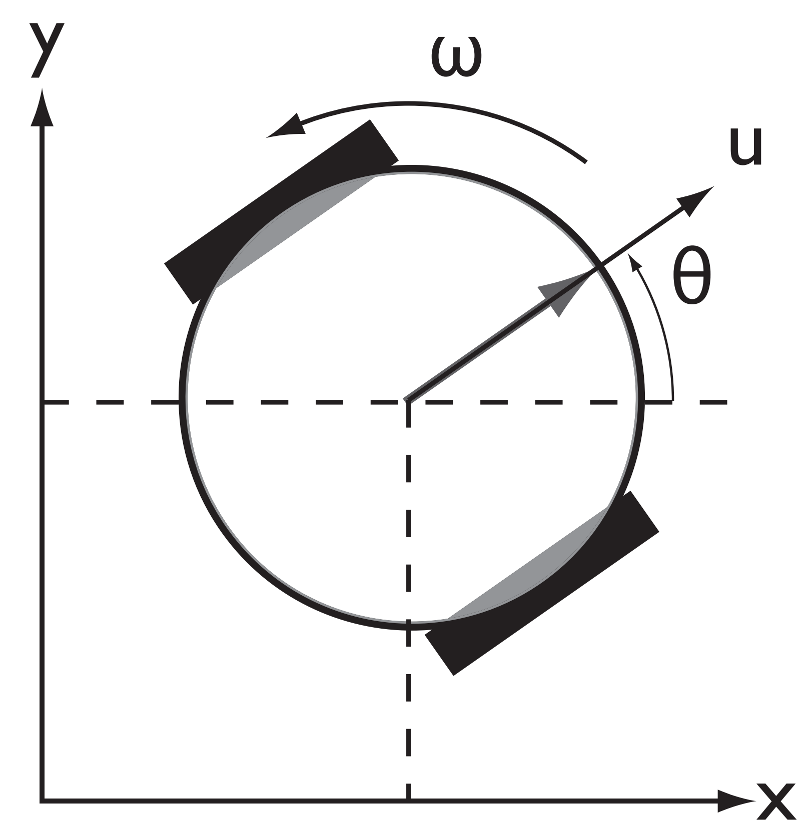
$$\omega = \frac{r\dot{\phi}_r}{d} - \frac{r\dot{\phi}_l}{d}$$

Motion:

$$\dot{x}_B = u$$
$$\dot{y}_B = 0$$
$$\dot{\theta}_B = \omega$$

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



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

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

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 **find control inputs**:

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

- yielding $$\begin{aligned} u &= \dot{x}\cos\theta + \dot{y}\sin\theta \\ \omega &= \dot{\theta} \end{aligned}$$

- under the **constraint** (remember than our robot is non-holonomic):

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

we can now control the wheel speeds!

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

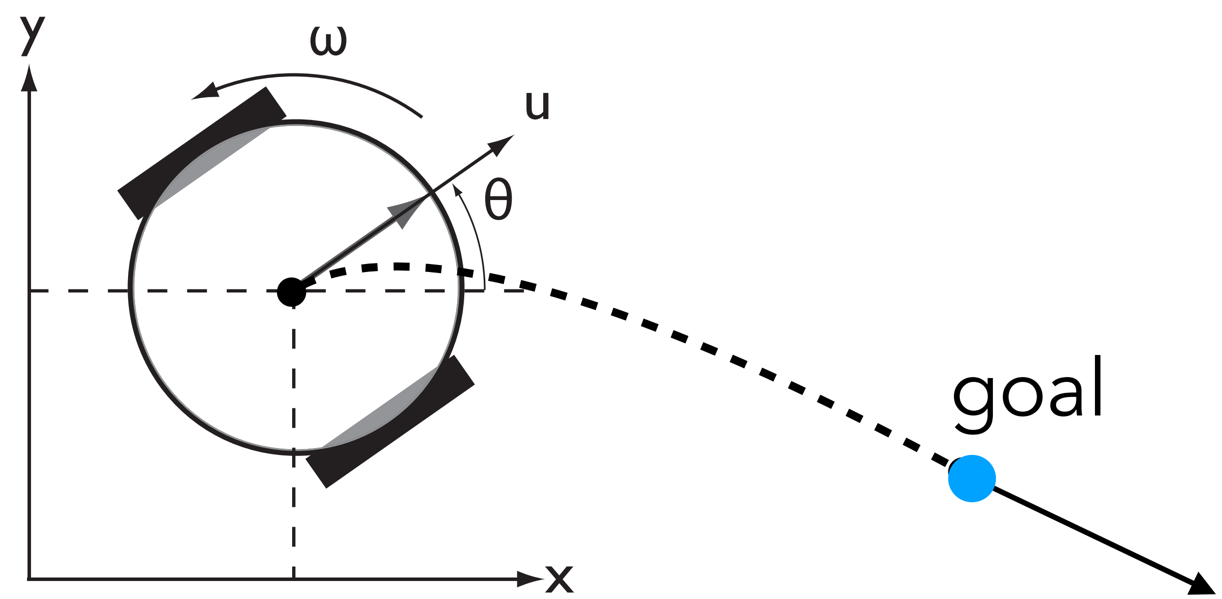
Inverse Kinematics

- 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:

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

control gain

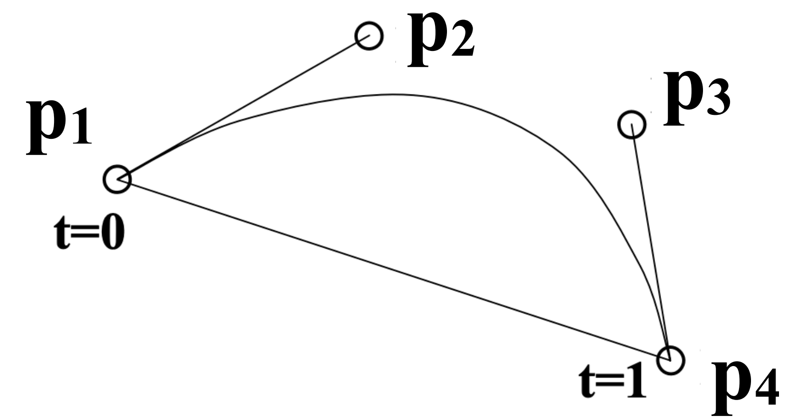


- However, we need to satisfy the non-holonomicity constraint:

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

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:

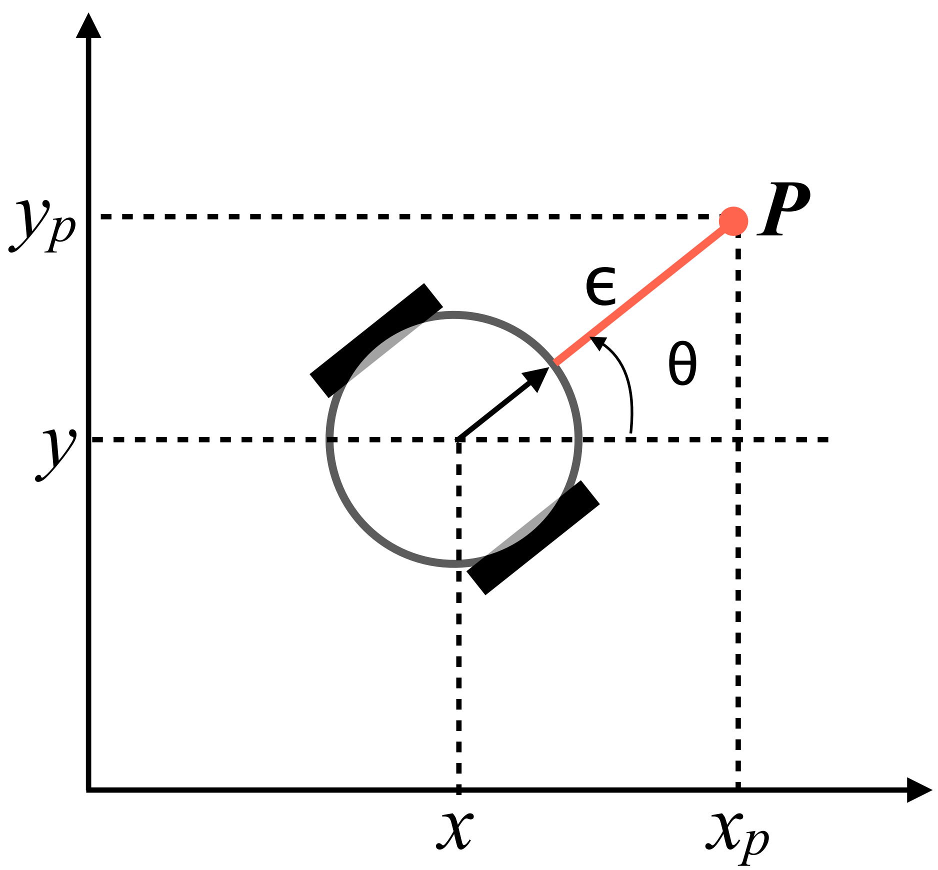


$$\mathbf{p}_1 = \begin{bmatrix} x \\ y \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} x + K_1 \cos\theta \\ y + K_1 \sin\theta \end{bmatrix} \quad \mathbf{p}_3 = \begin{bmatrix} x_G + K_2 \cos\theta_G \\ y_G + K_2 \sin\theta_G \end{bmatrix} \quad \mathbf{p}_4 = \begin{bmatrix} x_G \\ y_G \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{B}(t|\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) \quad \text{with curvature: } \dot{\theta} = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2}$$

Feedback Linearization

- Leverage linear control of a holonomic point P to control a non-holonomic robot.
- Key idea: formulate control inputs u, w as a function of $\dot{x}_p$ and $\dot{y}_p$



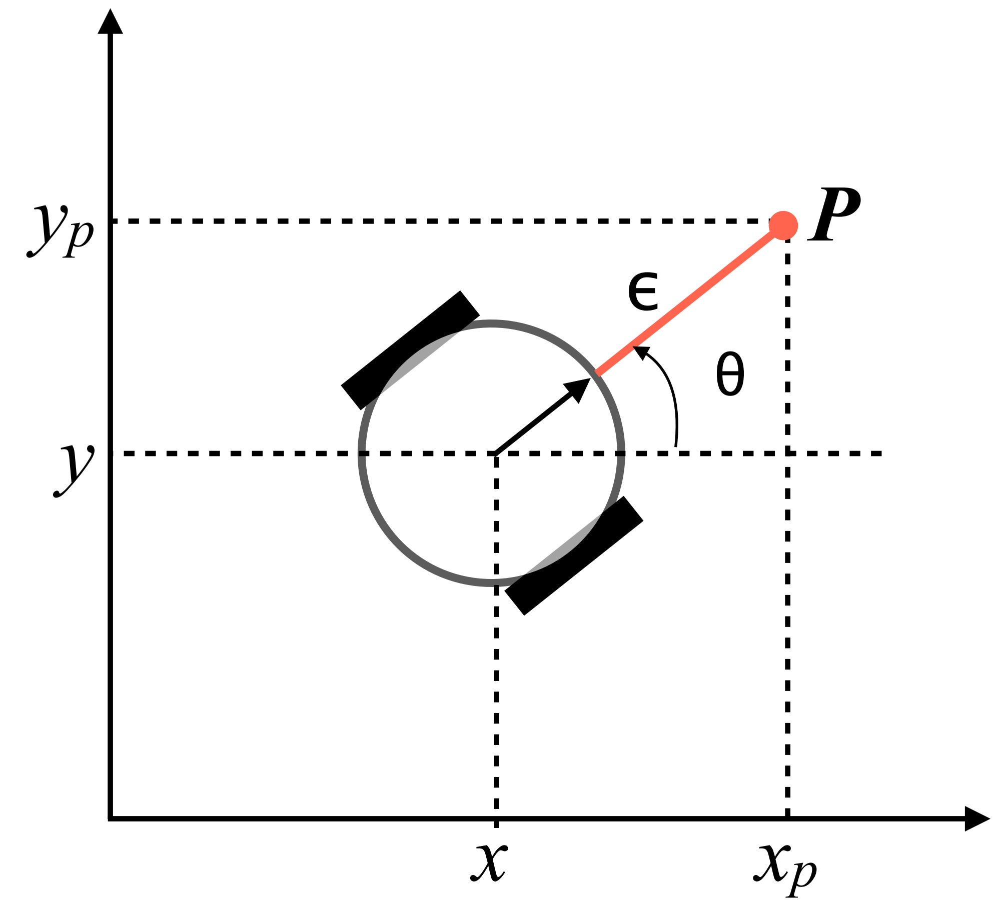
Idea: tie robot to a rod of length ϵ that you hold at point P. Point P can move holonomically; robot is pulled by rod.

Feedback Linearization

- Feedback linearization:

$$x_p = x + \epsilon \cos\theta \qquad \longrightarrow \qquad \dot{x}_p = \dot{x} + \epsilon(-\dot{\theta}\sin\theta)$$
$$y_p = y + \epsilon \sin\theta \qquad \qquad \qquad \dot{y}_p = \dot{y} + \epsilon(\dot{\theta}\cos\theta)$$

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

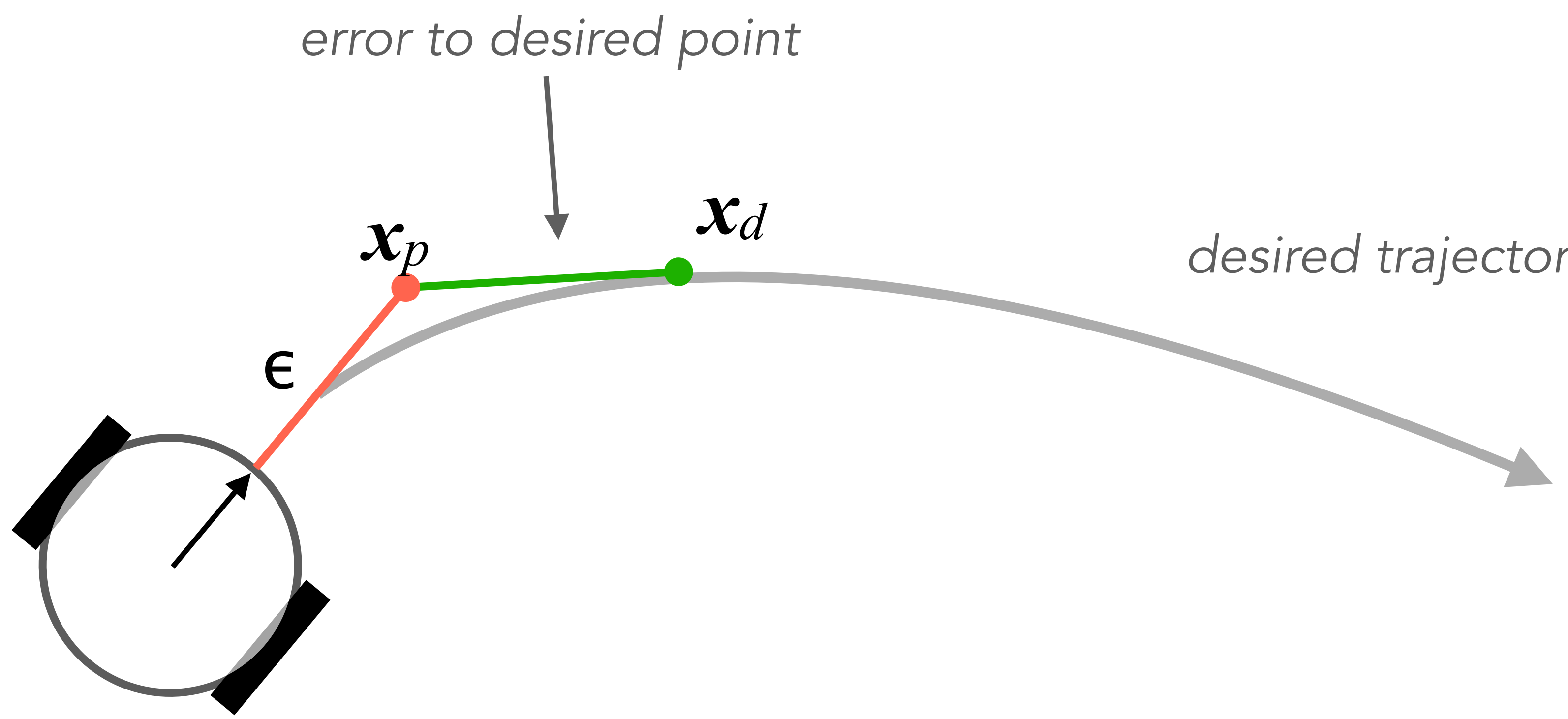


- Isolate control inputs:

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

Feedback Linearization

- Trajectory tracking:

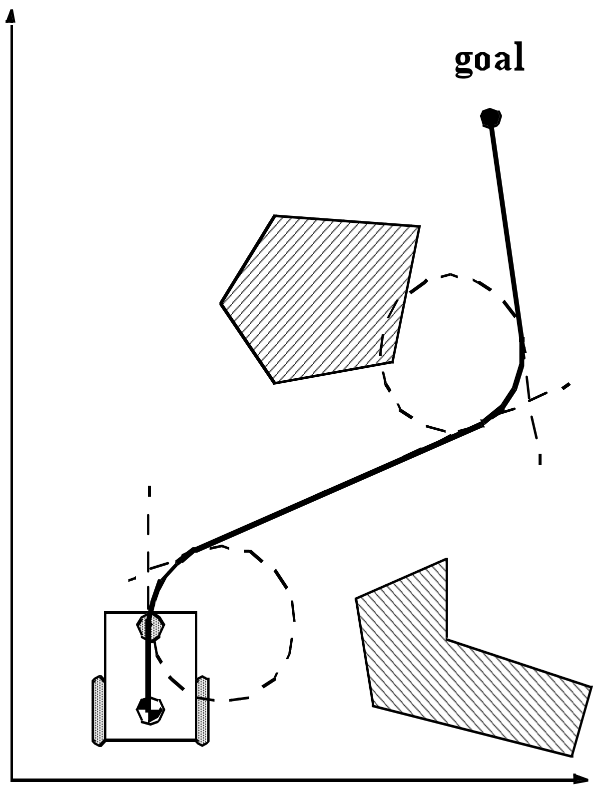


$$\dot{\mathbf{x}}_p = (\mathbf{x}_d - \mathbf{x}_p)\kappa + \dot{\mathbf{x}}_d$$

if desired point $\mathbf{x}_d$ *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!



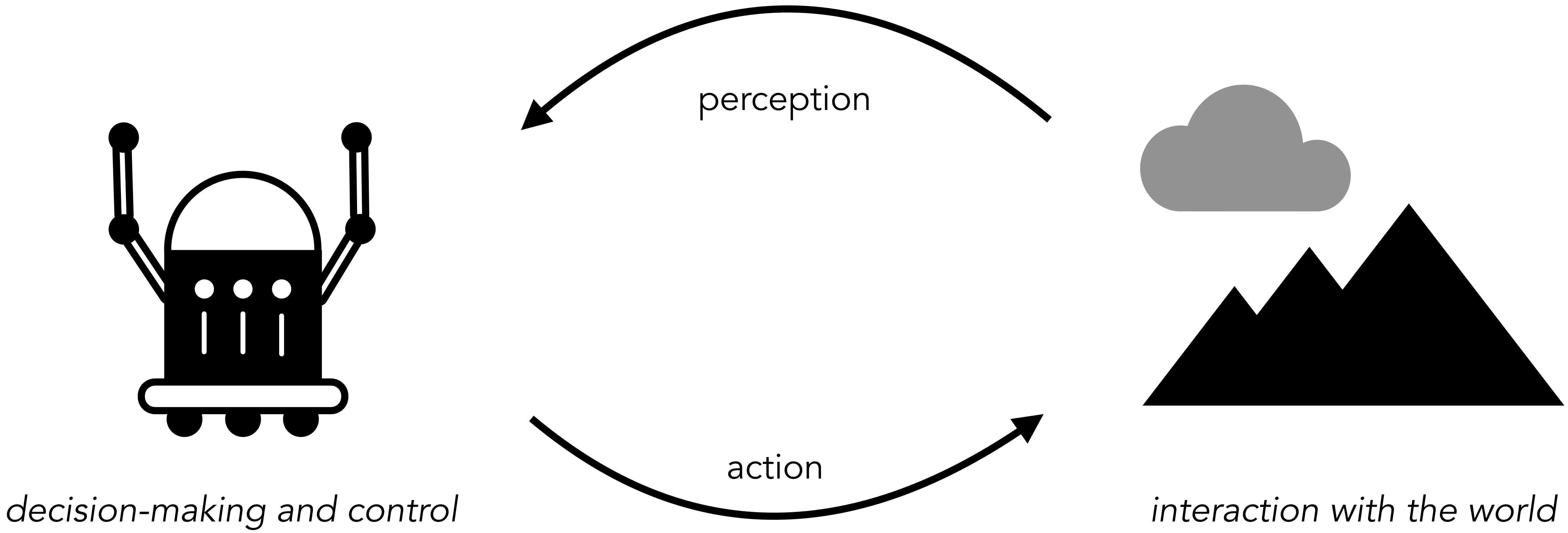
* image: Siegwart et al.

Open-Loop vs 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

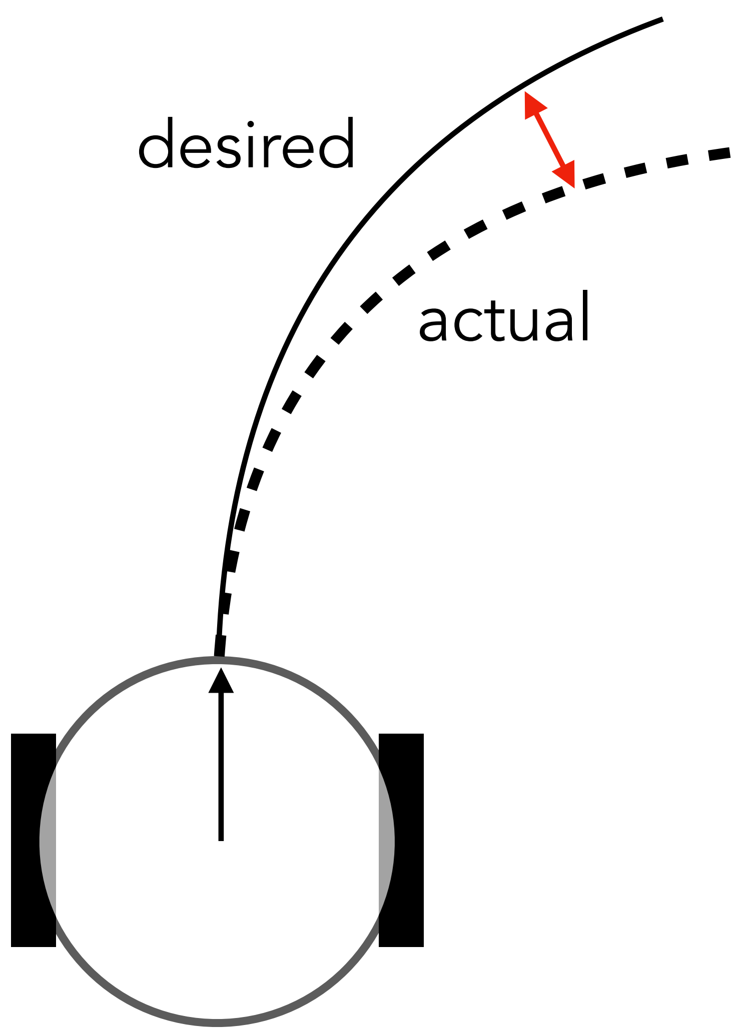
- Basic building block of autonomy



open-loop
vs
closed-loop

Open-Loop

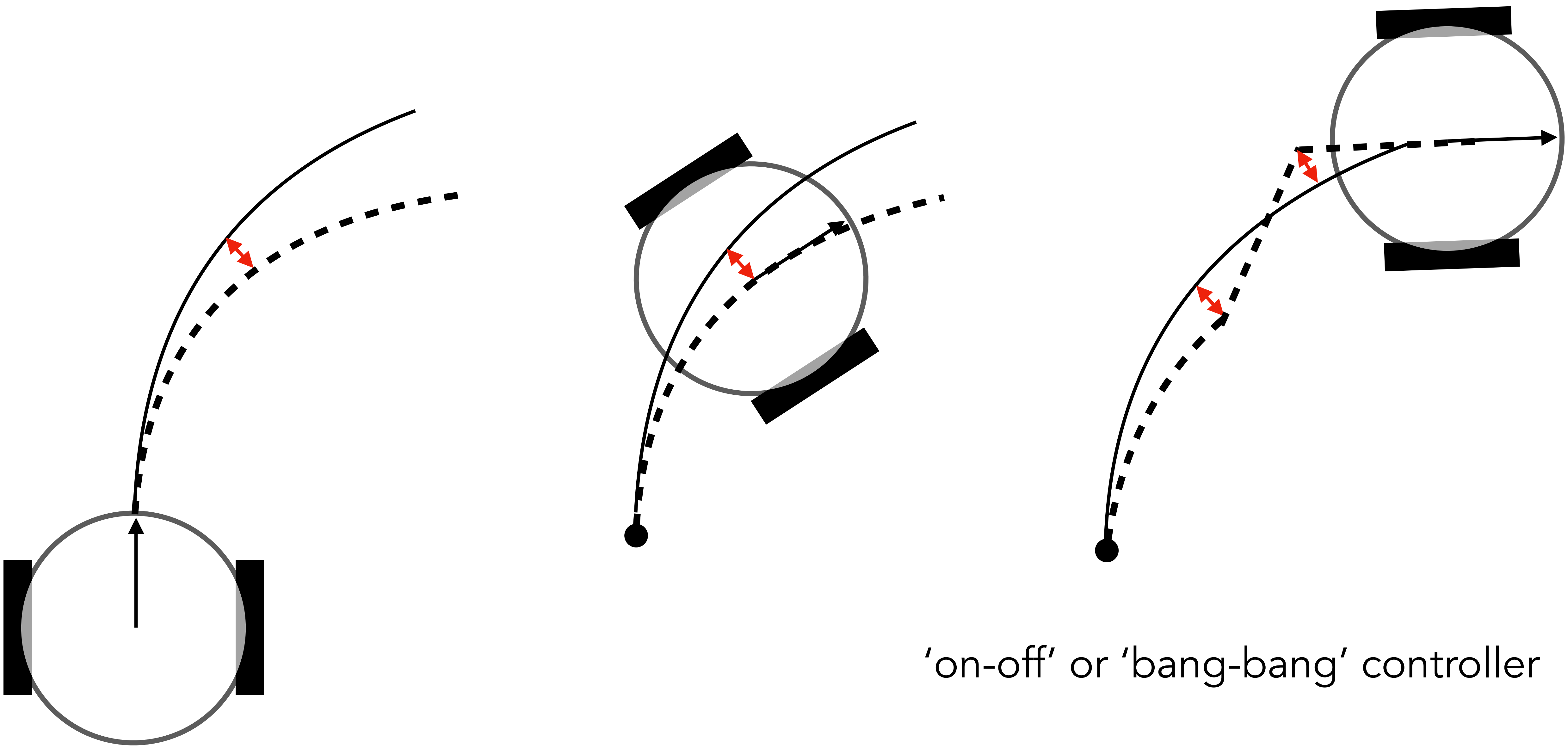
- Example: trajectory tracking
- In open-loop, the robot executes predefined control inputs.



Under imperfect conditions, the robot deviates from desired behavior.

A Simple Closed-Loop Controller

- Example: trajectory tracking
- The robot uses feedback to maintain a desired set-point.
- Assumption: robot receives **feedback** on distance to desired trajectory.



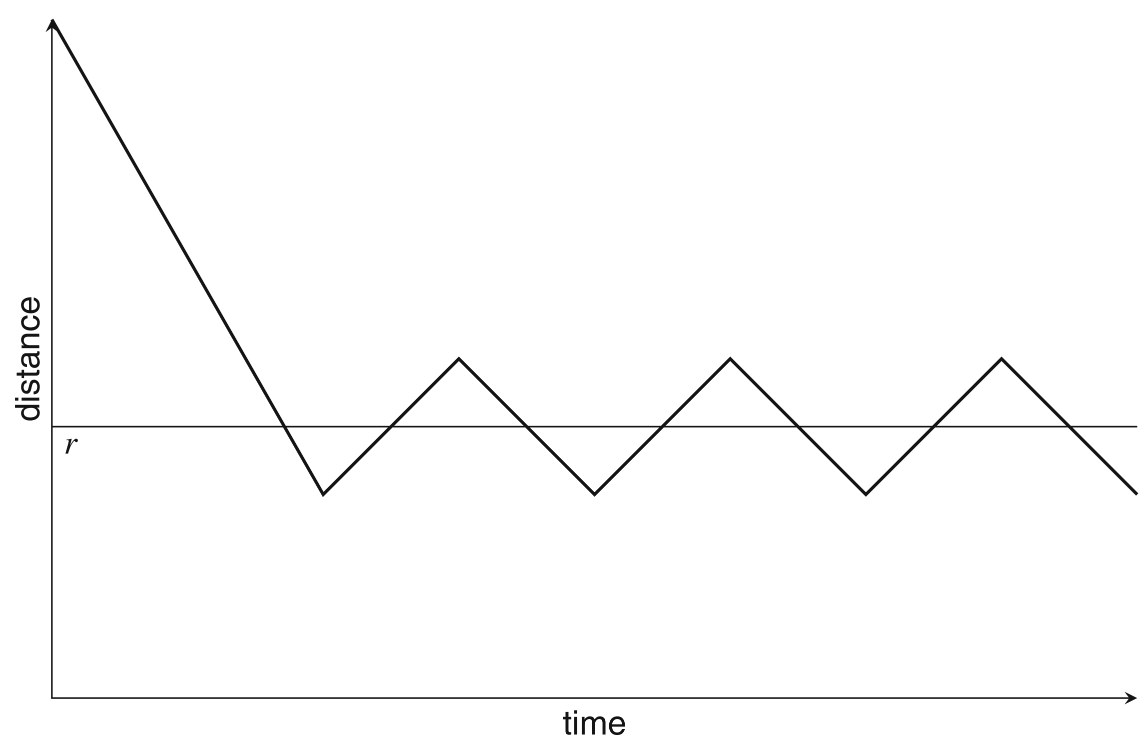
A Simple Closed-Loop Controller

Example pseudo-code for a line-following robot.

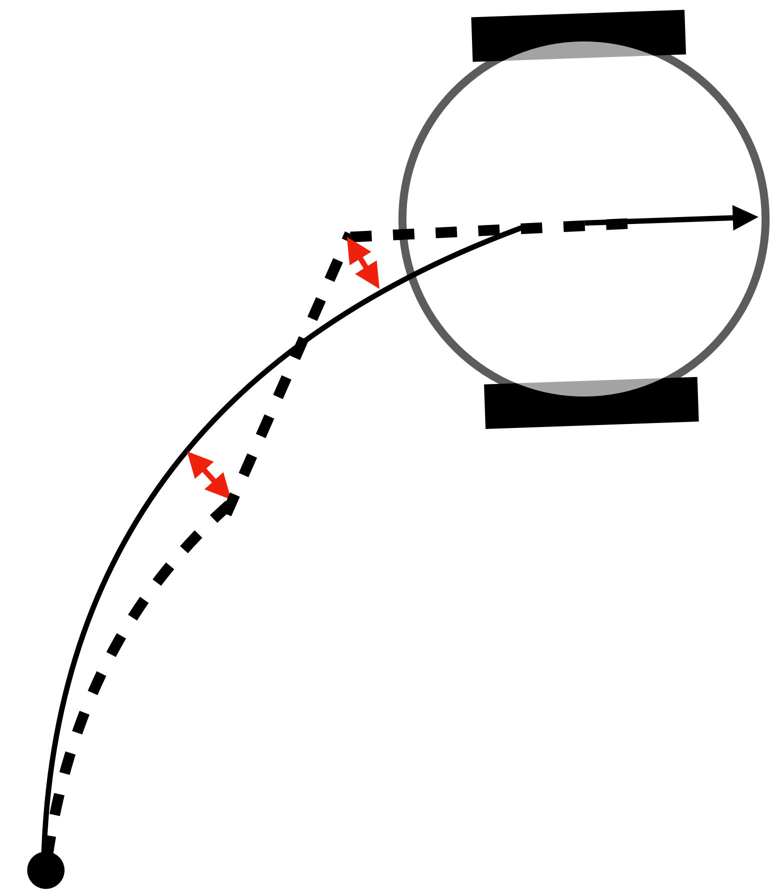
```
Algorithm: Bang-Bang Controller

forever do:
     error ← reference - measured // Distance
     if error < 0                  // Too far left
      left-motor-power ← 100
      right-motor-power ← -100
     if error > 0                  // Too far right
      left-motor-power ← -100
      right-motor-power ← 100
     if error = 0                  // Just right
      left-motor-power ← 100
      right-motor-power ← 100
```

A Simple Closed-Loop Controller

- Example: trajectory tracking
- The robot uses feedback to maintain a desired set-point.
- Assumption: robot receives **feedback** on distance to desired trajectory.



zig-zag behavior: we can do better!

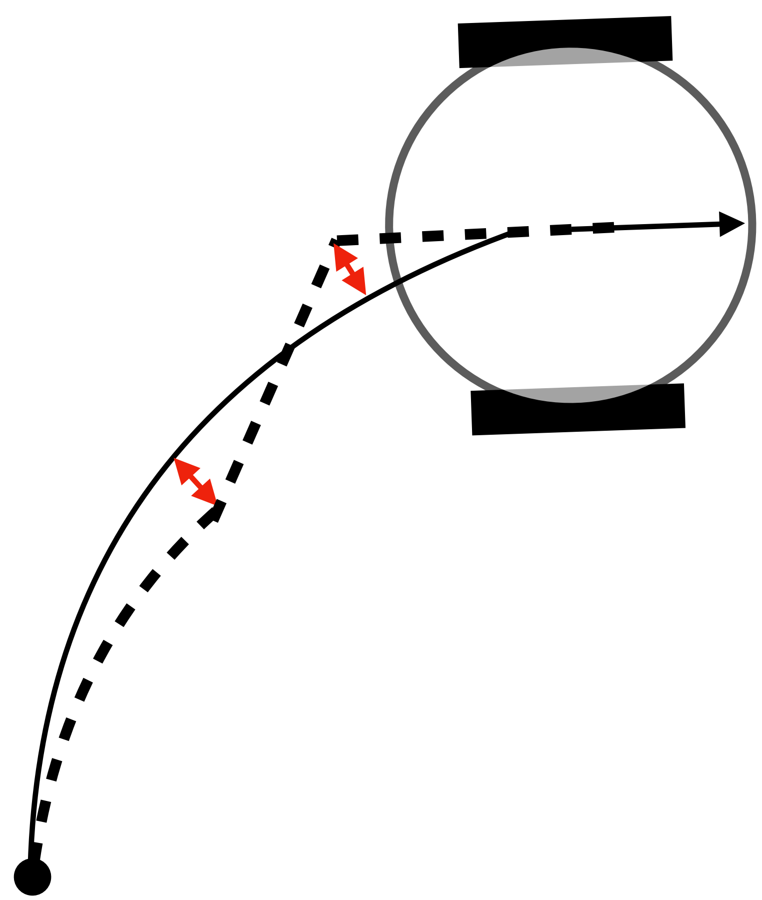


'on-off' or 'bang-bang' controller

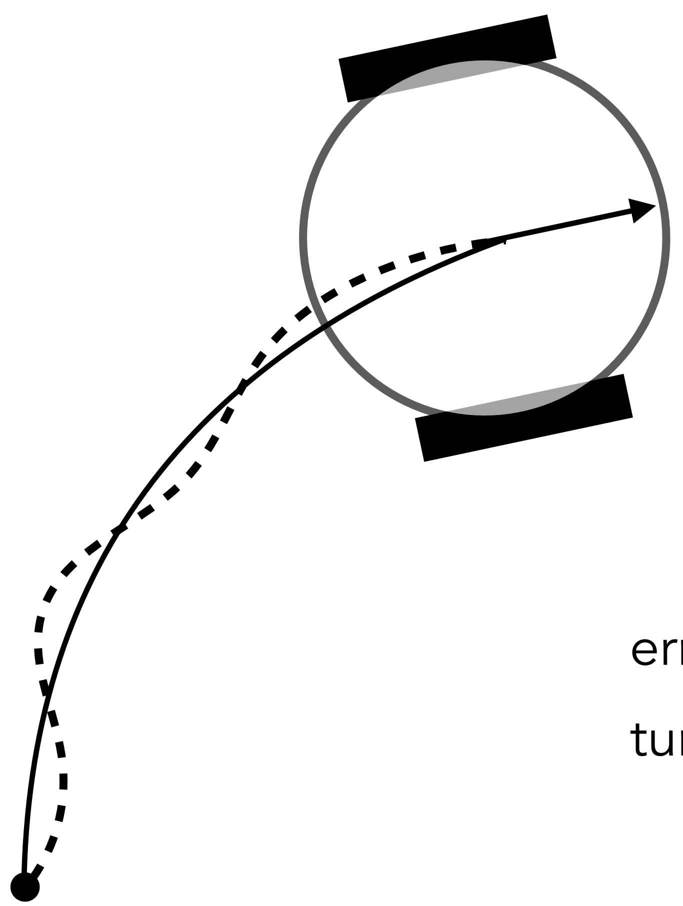
* image credit: Elements of Robotics

Proportional Control (P-Control)

- Example: trajectory tracking
- The robot uses **feedback** to maintain a desired set-point.
- Robot computes error, and adjusts control **as a function of error**



previous slide: oscillatory behavior



error = distance-to-trajectory

turning-control = ***K*** * error

adjustment is proportional to error!

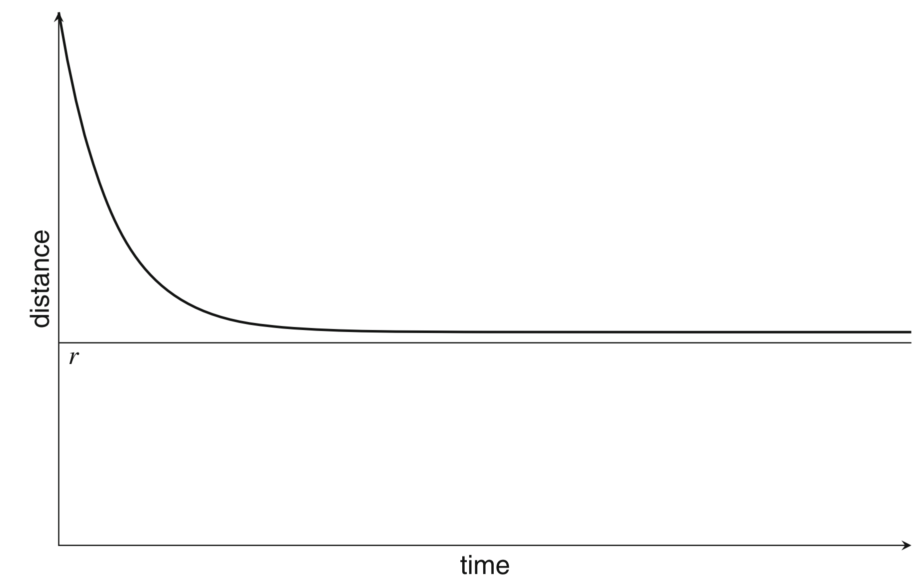
Proportional Control (P-Control)

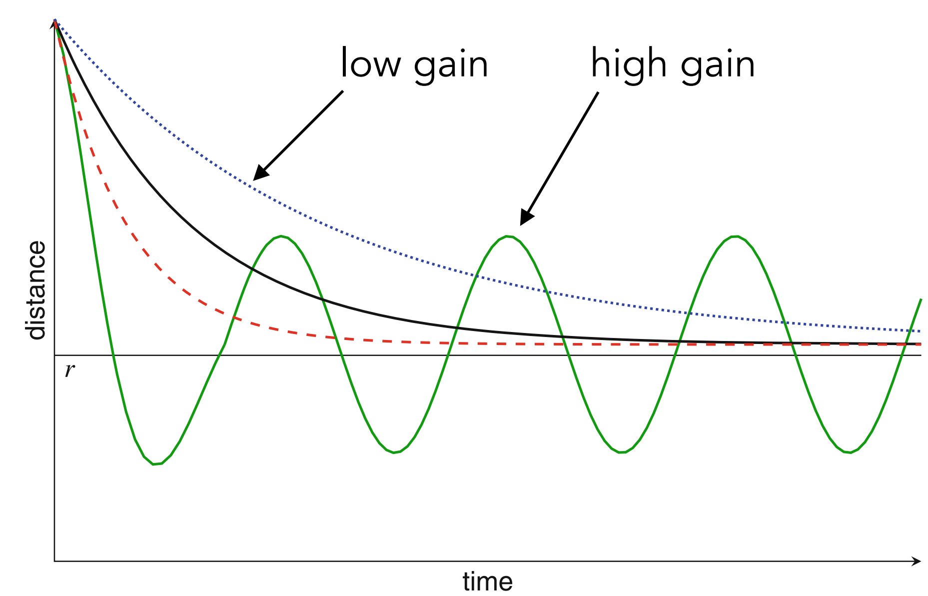
Example pseudo-code for a line-following robot.

```
Algorithm: P-Controller

forever do:
     error ← reference – measured   // Distance
     power ← gain * error           // Control value
     left-motor-power ← power_left
     right-motor-power ← power_right
```

Proportional Control (P-Control)

- Behavior of P-control:
  - Adapt control proportionally to your perceived error to set-point.
  - $$u(t) = \kappa_p e(t)$$
- Why is the target distance not reached?
  - E.g., what if motors have friction?
- Behavior for varying gain values
- High gains not desirable! We call this an **unstable** controller.





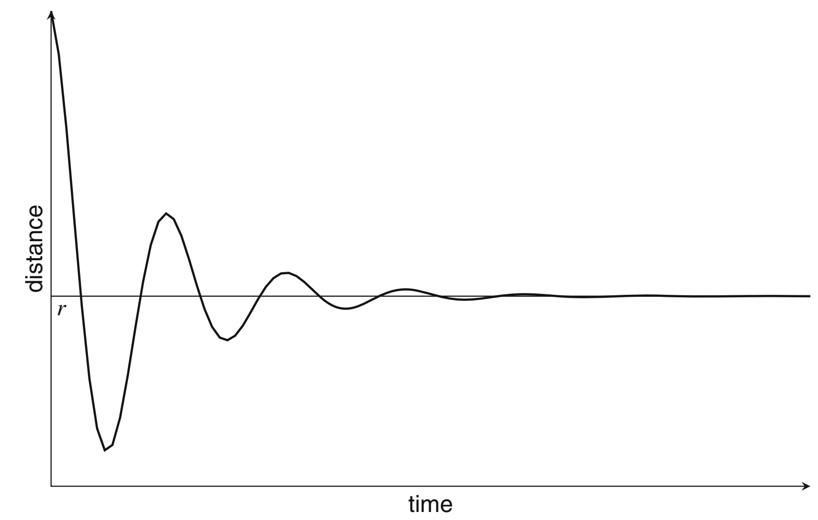
* image credit: Elements of Robotics

PID Control (Advanced)

- PI-controller:
    - takes into account **accumulated error** over time

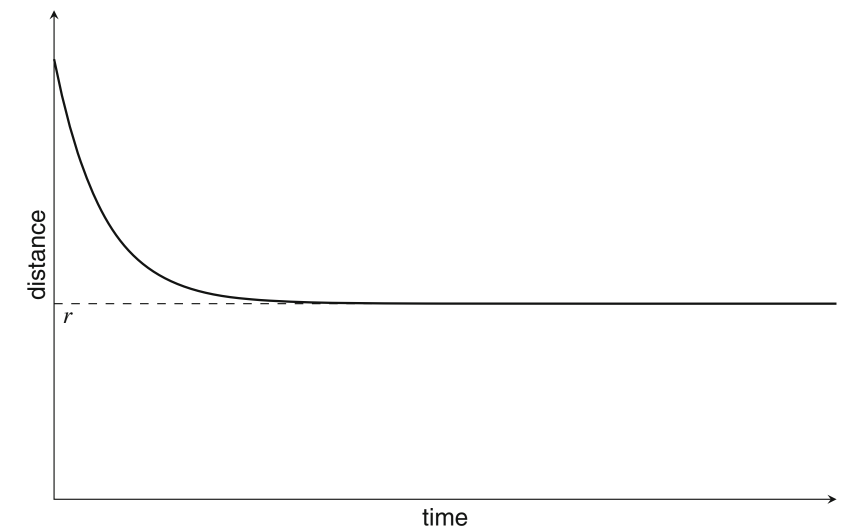
$$u(t) = \kappa_p e(t) + \kappa_i \int_0^t e(\tau)\, d\tau$$

    - E.g., in presence of friction, error will be integrated causing higher motor setting to overcome remaining delta.



- PID-controller:
    - take into account **future error** by computing rate of change of error.
    - acts as a *'dampener'* on control effort.

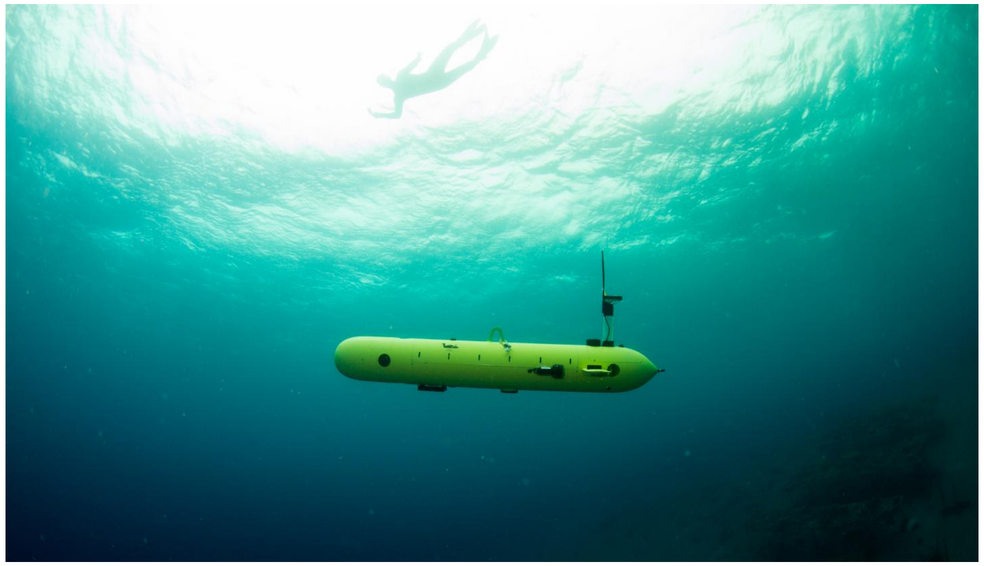
$$u(t) = \kappa_p e(t) + \kappa_i \int_0^t e(\tau)\, d\tau + \kappa_d \frac{de(t)}{dt}$$



* image credit: Elements of Robotics

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



Further Reading

Books that cover fundamental concepts:

- Elements of Robotics, F Mondada et al., 2018
- Autonomous Mobile Robots, R Siegwart et al., 2004