



3 – Motion control

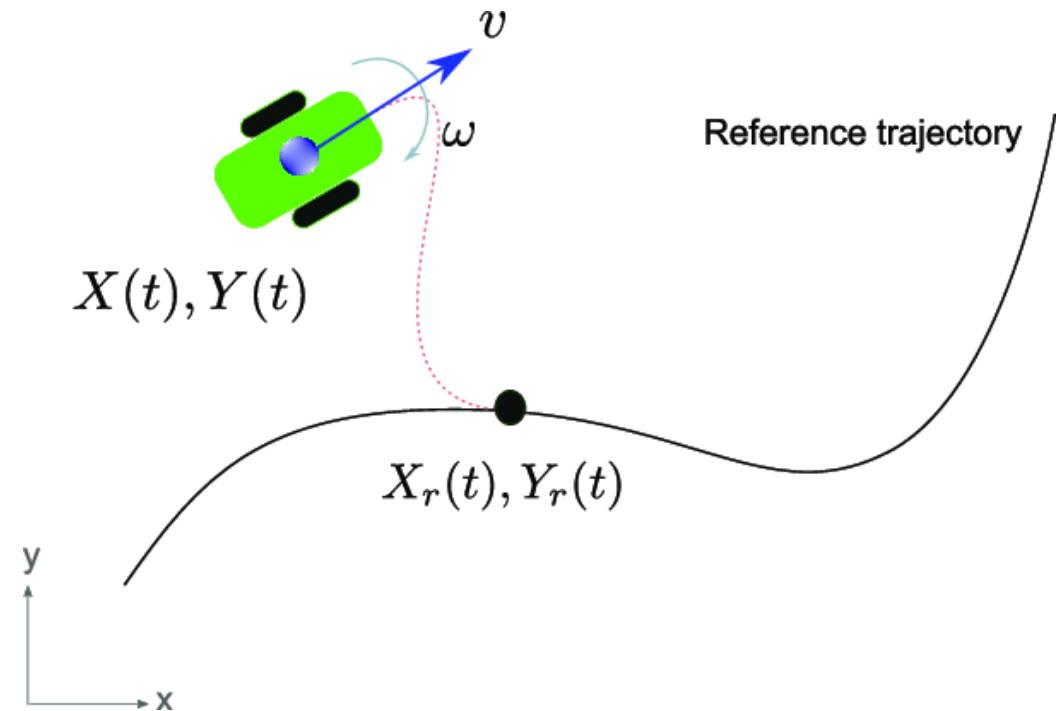
Advanced Methods for Mapping and Self-localization in Robotics (MPC-MAP)

Tomas Lazna

Brno University of Technology
2022

What is motion control for?

- Following known trajectory
- Kinematic model is required – a relation between the wheels speed and the robot motion
- Are there any constraints?
- How to model motion?



[1]

Types of drive:

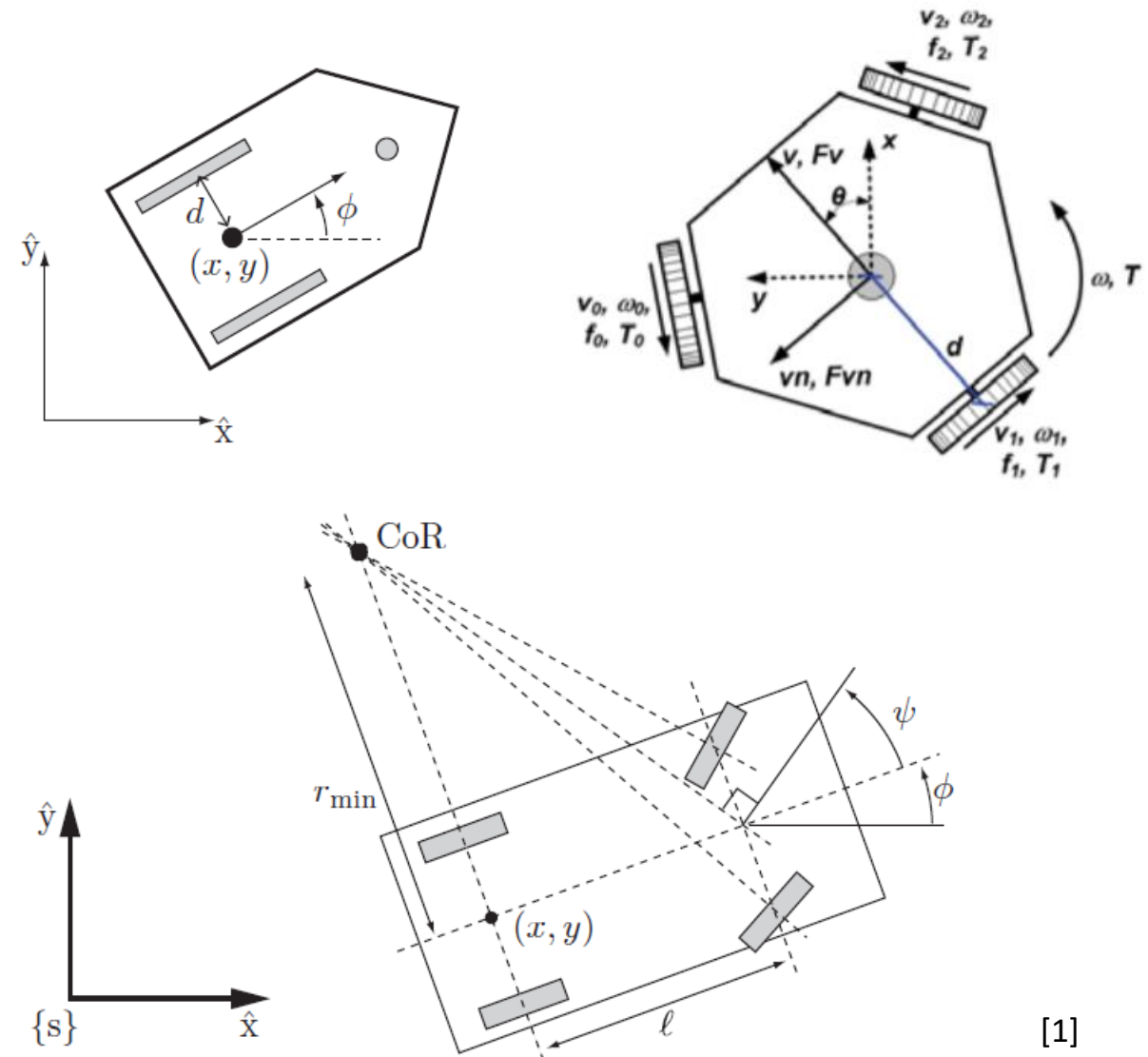
- Differential
- Ackermann
- Omnidirectional
- Others ...

Kinematics

- „Geometry of motion“
- Relation (acceleration –) velocity – position
- Cause of motion is not analyzed

Dynamics

- Focuses on „why is object moving“
- Forces, torques, mass, etc.



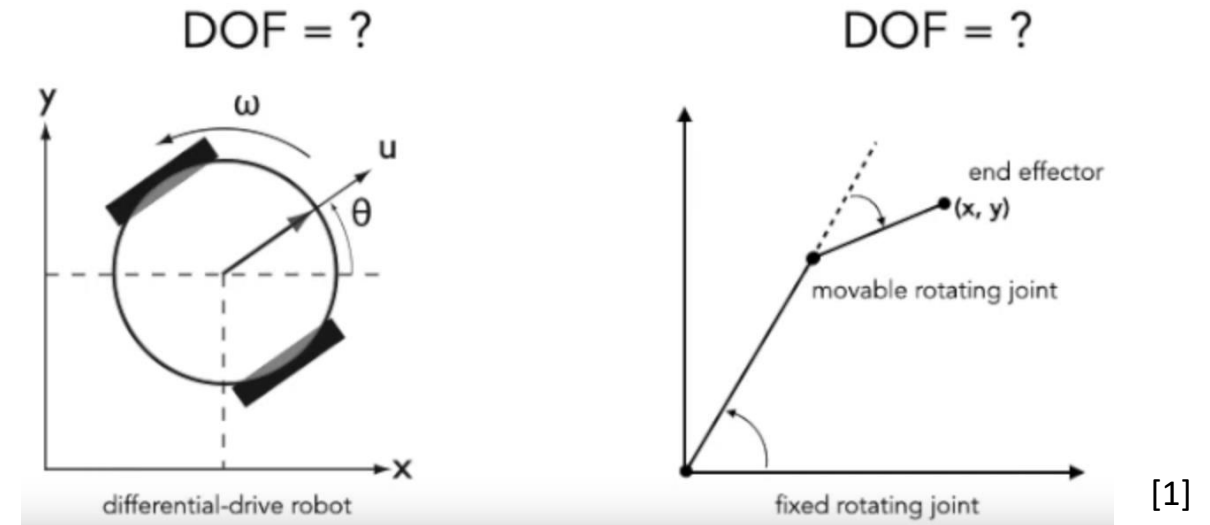
[1]

Degrees of freedom (DOFs)

- Minimum number of real numbers to represent the robot's configuration
- Most actuators control a single DOF
- Either translational or rotational
- Depends on the type of robot

Degrees of motion (DOMs)

- = Differentiable DOFs (DDOFs)
- Number of DOFs that can be directly accessed by the actuators
- Also number of independent motion velocities
 - Unicycle = 2, Bicycle = 1



[1]

Constraints

- Constrain position = holonomic
- Constrain velocity = non-holonomic

Integrability of constraints

- $f(\mathbf{q}, t)$ – depends only on position = integrable = holonomic
- $f(\mathbf{q}, \dot{\mathbf{q}}, t)$ – depends also on velocity = is not integrable = non-holonomic

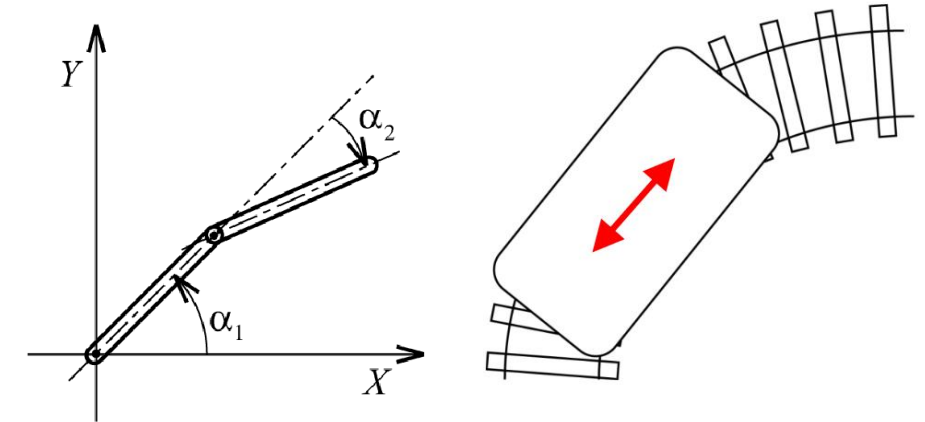
Holonomic robot

- DOF = DOM (= DDOF)
- All constraints are holonomic

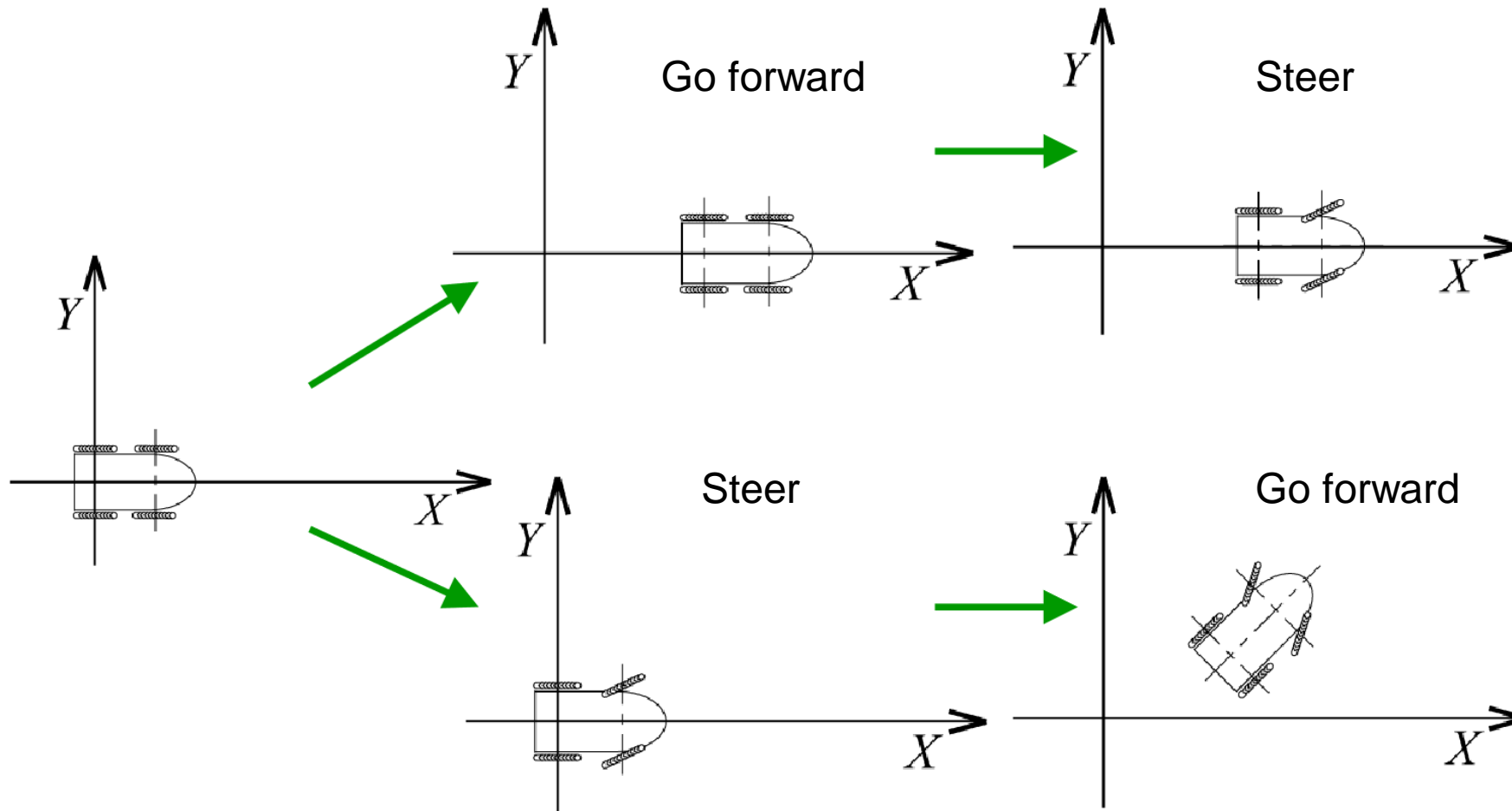
Non-holonomic robot

- DOF > DOM
- Position depends on the order of control inputs!

Examples of holonomic systems:

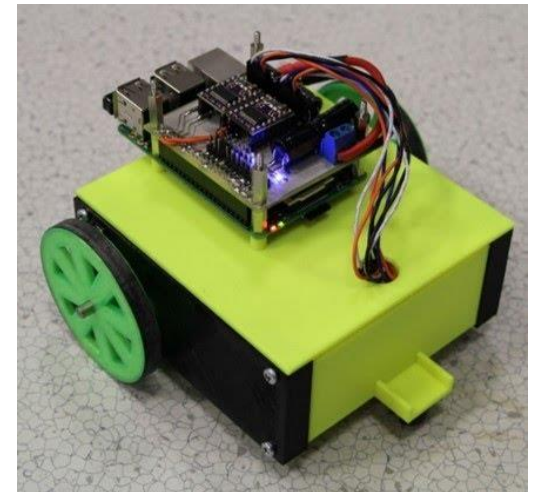


[1]



Depends on the order!

Differential drive =
non-holonomic

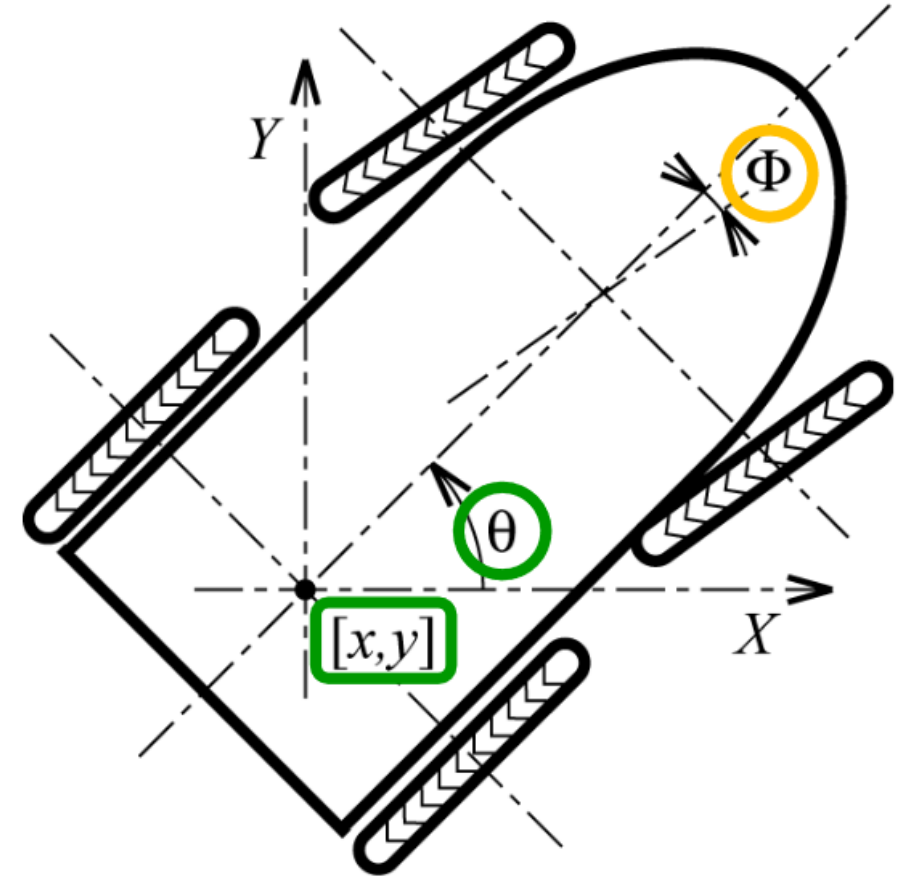


State

- Position of the reference point in the world frame – x, y
- Orientation – θ
- Describes configuration of the robot body
- $\mathbf{x} = (x, y, \theta)$

Extended state

- State + additional parameters
- E. g., steering angle Φ in Ackermann drive
- $\mathbf{x}_{\text{Ack}} = (x, y, \theta, \Phi)$

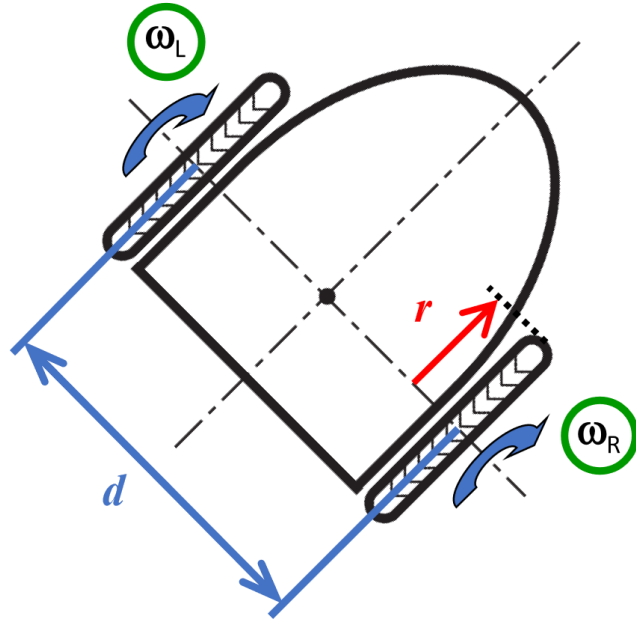


Parameters of the chassis

- Wheel radius r
- Distance between wheels d

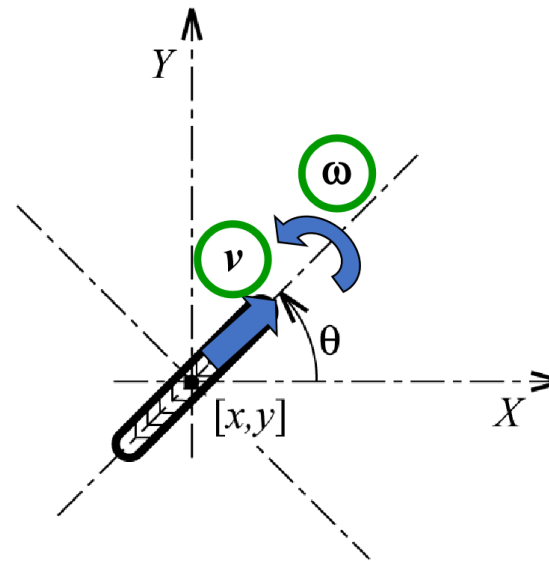
Control inputs

- Speed of wheels ω_R, ω_L

**Simplification to unicycle**

- Forward velocity v
- Angular velocity ω

$$v = \omega R$$



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

$$\omega = \frac{r(\omega_R - \omega_L)}{d}$$

$$\omega_R = \frac{2v + \omega d}{2r}$$

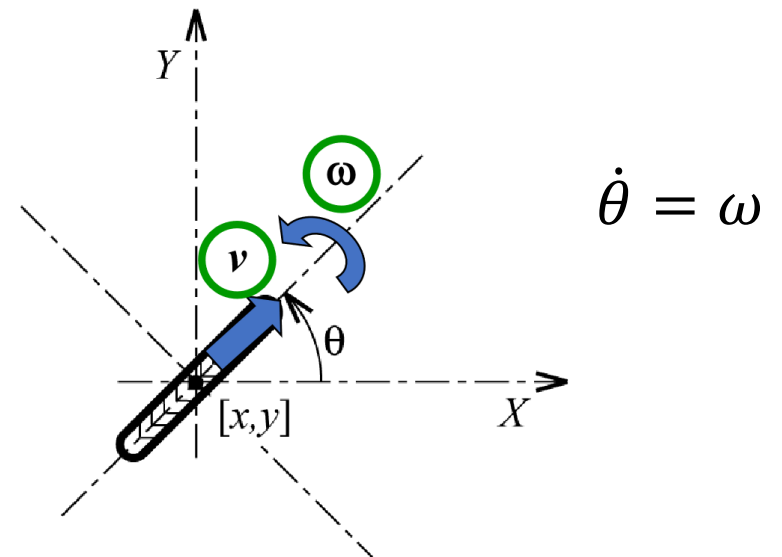
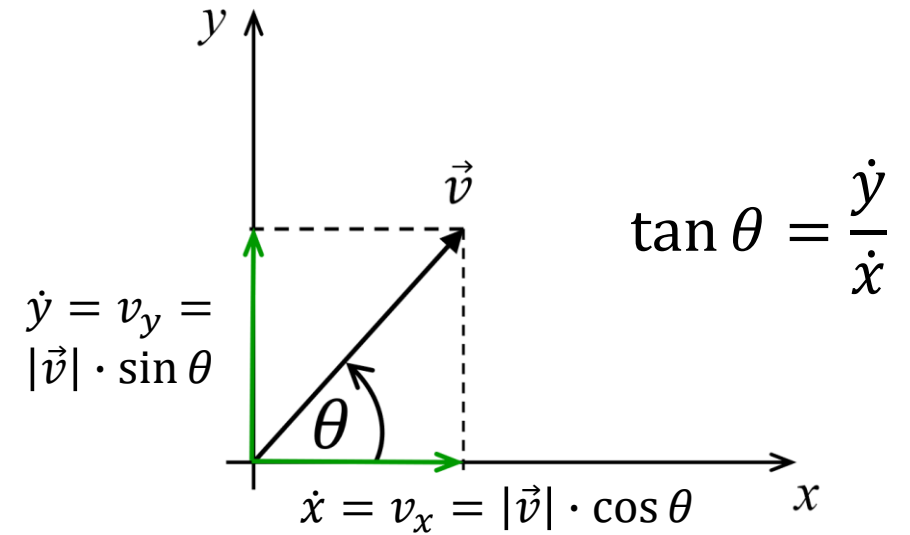
$$\omega_L = \frac{2v - \omega d}{2r}$$

State equations

- 3 states
- 2 control inputs

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos \theta \\ \sin \theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \omega$$

- Constraint: $\dot{x} \sin \theta = \dot{y} \cos \theta$



Forward kinematics

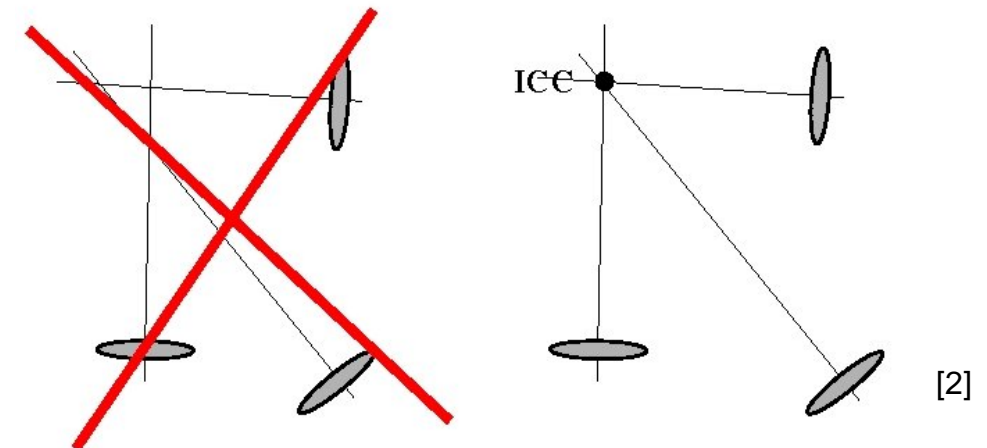
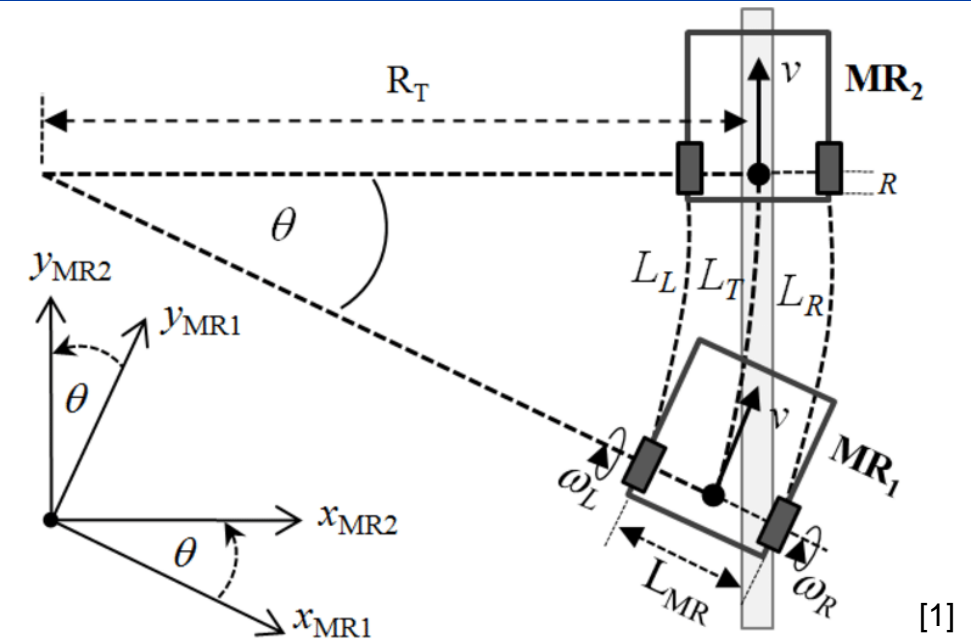
- Initial state and control inputs are known
- What is the final state of the robot?
- Straightforward task

Inverse kinematics

- Initial and final states are known
- What are the control inputs?
- Problematic in case of non-holonomic platforms
- Singular points

Instantaneous center of curvature (rotation)

- Each wheel must rotate along its y -axis
- Wheels move along circular ($R < \infty$) or straight trajectories



- Turning radius

$$R = \frac{v}{\omega} = \frac{\frac{r(\omega_R + \omega_L)}{2}}{\frac{r(\omega_R - \omega_L)}{d}} = \frac{d(\omega_R + \omega_L)}{2(\omega_R - \omega_L)}$$

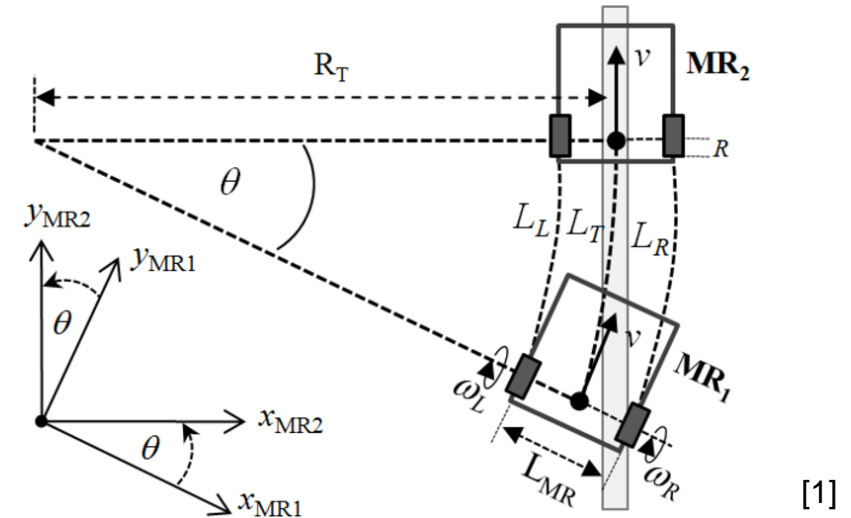
- ICC

$$\begin{pmatrix} ICC_x \\ ICC_y \end{pmatrix} = \begin{pmatrix} x + R \cos(\theta + \pi/2) \\ y + R \sin(\theta + \pi/2) \end{pmatrix} = \begin{pmatrix} x - R \sin \theta \\ y + R \cos \theta \end{pmatrix}$$

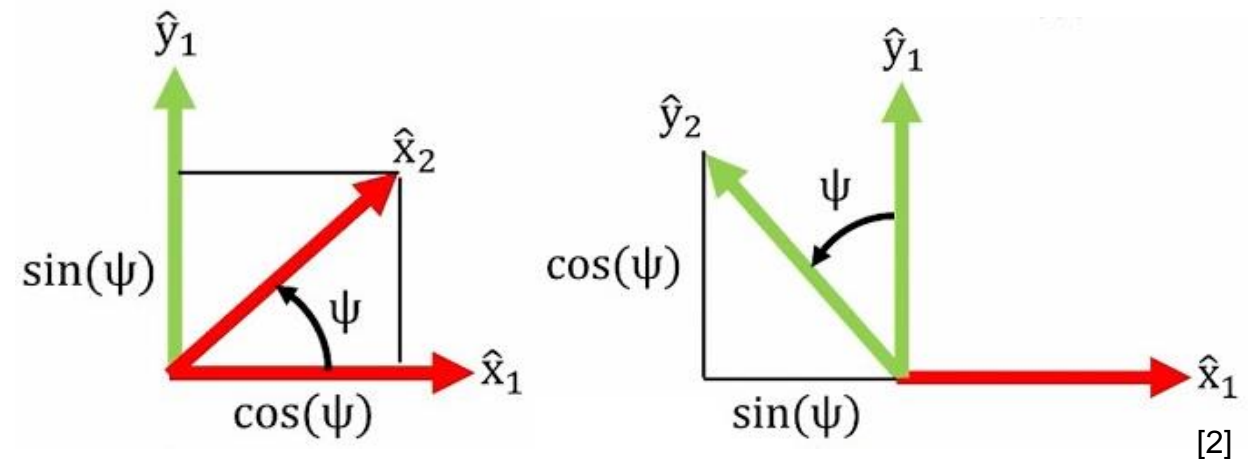
- Rotation matrix

$$\begin{pmatrix} \hat{x}_1 \\ \hat{y}_1 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{x}_2 \\ \hat{y}_2 \\ 1 \end{pmatrix}$$

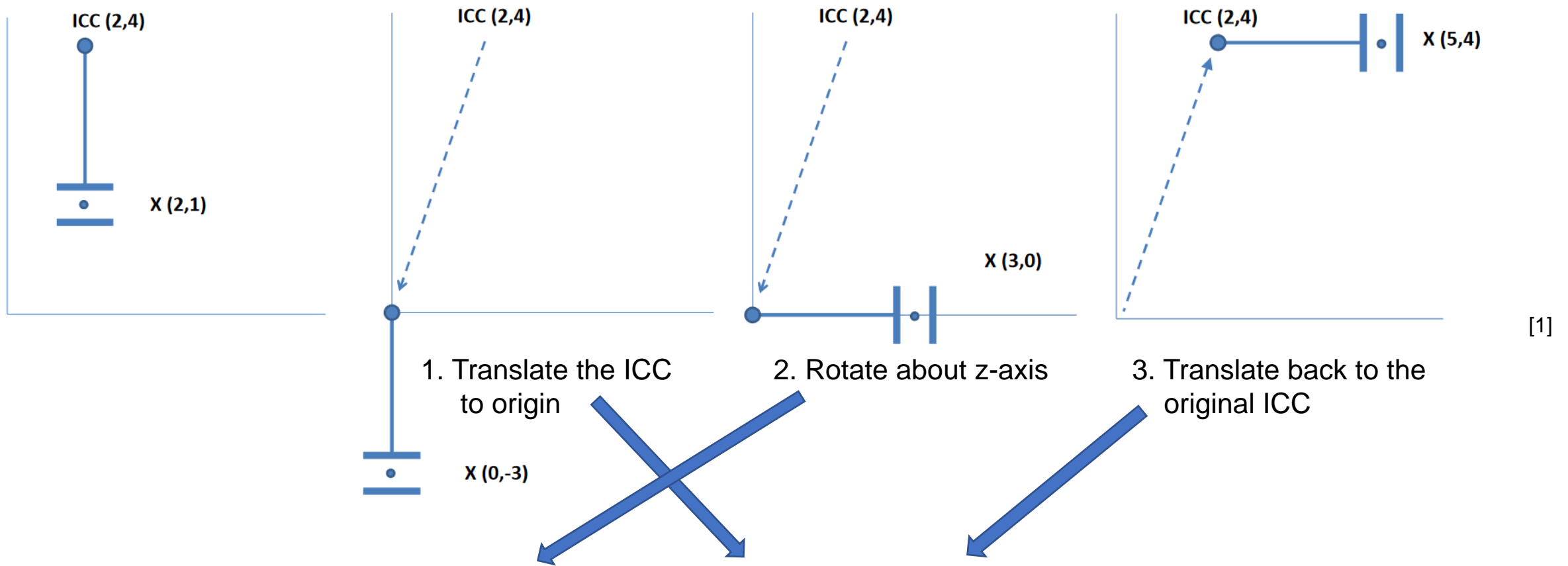
$$\begin{pmatrix} \hat{x}_2 \\ \hat{y}_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{x}_1 \\ \hat{y}_1 \\ 1 \end{pmatrix}$$



[1]



[2]



$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) & 0 \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{pmatrix} + \begin{pmatrix} ICC_x \\ ICC_y \\ \omega\Delta t \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -R \sin \theta + R \sin(\theta + \omega\Delta t) \\ R \cos \theta - R \cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix}$$



- We can express the state equations using the rotation matrix

World frame \rightarrow $\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ 0 \\ \omega \end{pmatrix}$

Robot frame \leftarrow

$= \mathbf{R}(\theta)$

- Let us invert the equations

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

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

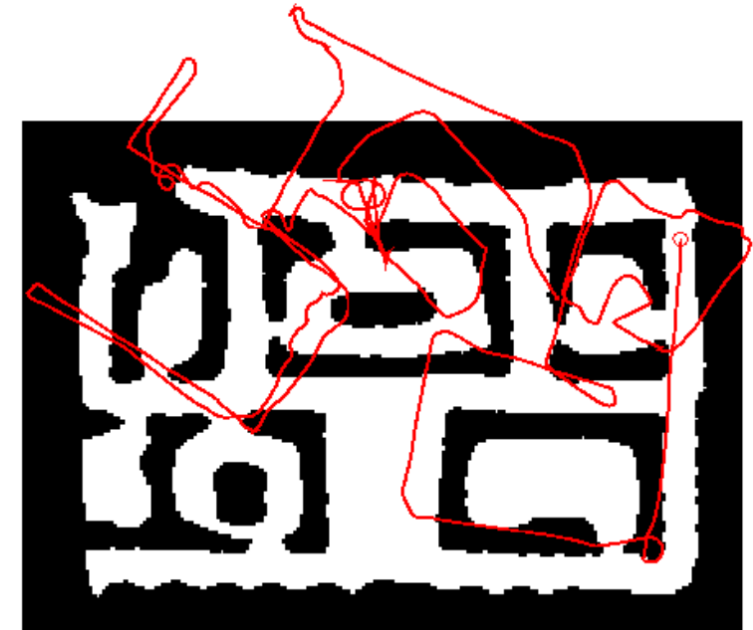
$\longrightarrow \omega = \dot{\theta}$

- Ideally, we would set

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = K \begin{pmatrix} x_G - x \\ y_G - y \\ \theta_G - \theta \end{pmatrix} \quad \text{Goal} = (x_G, y_G, \theta_G)$$

- But we have to follow the non-holonomic constraint $\dot{x} \sin \theta = \dot{y} \cos \theta$

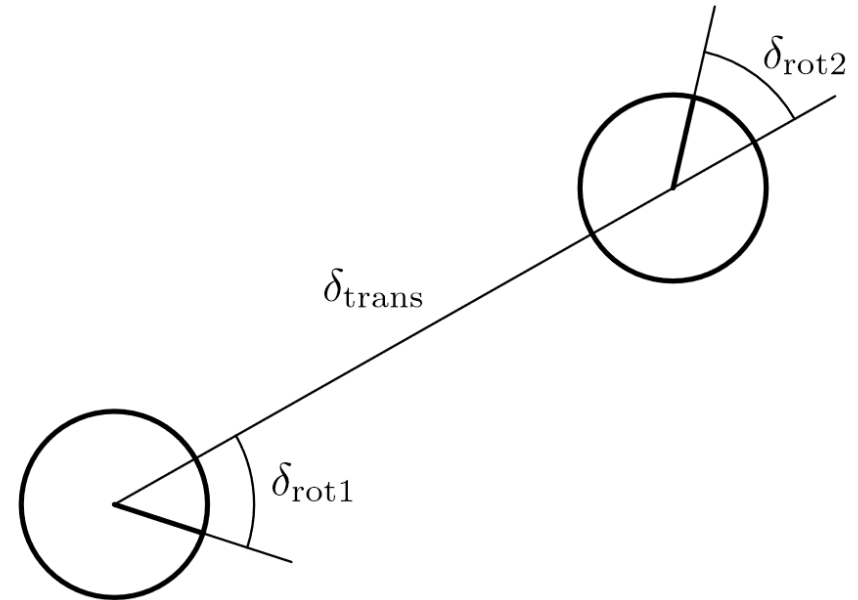
- Motion is inherently uncertain – how to model it?
- *Probabilistic* models are needed
- To implement Bayes filters (particle, Kalman, ...) we need to know the posterior probability $p(x_t|x_{t-1}, u_t)$
= what is the probability that action u_t takes the system from state x_{t-1} to state x_t
- Two major types of motion models:
 - odometry-based
 - velocity-based (dead reckoning)



Odometry-based models:

- Systems equipped with wheel encoders
- Suitable for estimation
- Generally more accurate
- $u_t = \begin{pmatrix} \bar{x}_{t-1} \\ \bar{x}_t \end{pmatrix} \rightarrow (\delta_{\text{rot1}}, \delta_{\text{trans}}, \delta_{\text{rot2}})$

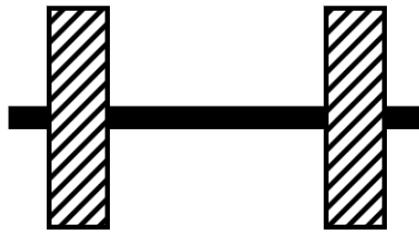
(decomposition to rotation-translation-rotation)



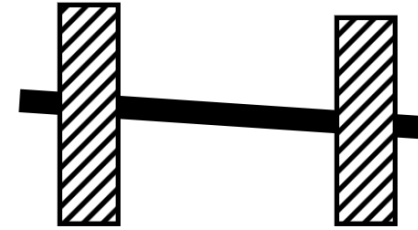
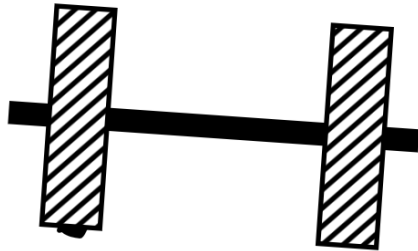
[1]

Velocity-based models:

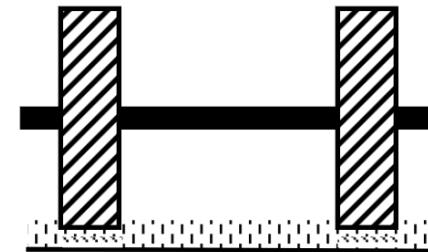
- New pose is calculated from velocities and time elapsed
- Dead reckoning = deduced reckoning
- Suitable for prediction
- $u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix}$



ideal case

different wheel
diameters

bump



carpet

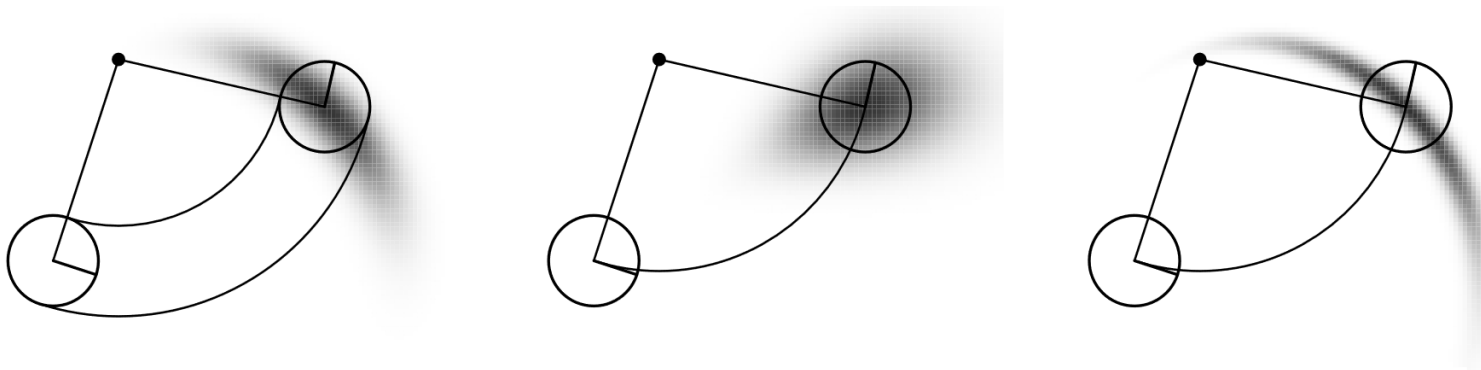
and many more ...

[1]

- We are looking for $p(x_t|x_{t-1}, u_t)$ for a differential drive
- Naive approach: Normal distribution (3D) centered in $x_t =$ not very realistic
- Let us assume that actual control inputs consist of desired velocities and a noise:

$$\begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \begin{pmatrix} v \\ \omega \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha_1 v^2 + \alpha_2 \omega^2} \\ \varepsilon_{\alpha_3 v^2 + \alpha_4 \omega^2} \end{pmatrix}$$

where ε_{b^2} is a zero-mean error variable with variance b^2 ,
parameters α_i are robot-specific error coefficients



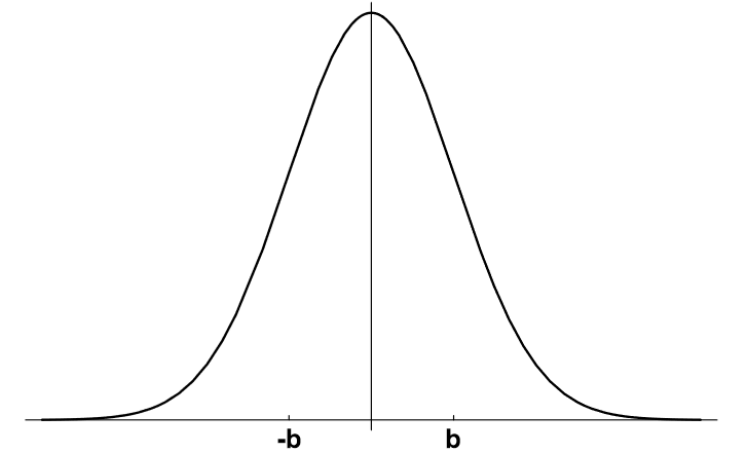
[1]

- Possible error distributions:

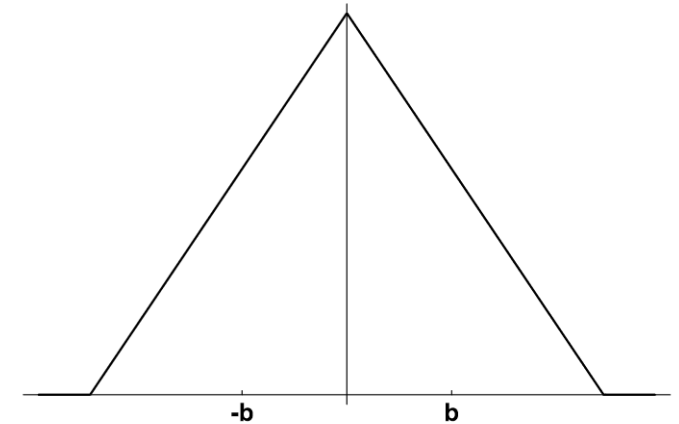
- Normal $\varepsilon_{b^2}(a) = \frac{1}{\sqrt{2\pi b^2}} e^{-\frac{1a^2}{2b^2}}$
- Triangular $\varepsilon_{b^2}(a) = \max\left\{0, \frac{1}{\sqrt{6}b} - \frac{|a|}{6b^2}\right\}$

- Assuming $x_{t-1} = (x, y, \theta)$, $x_t = (x', y', \theta')$, we have:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\hat{v}/\hat{\omega} \sin \theta + \hat{v}/\hat{\omega} \sin(\theta + \hat{\omega}\Delta t) \\ \hat{v}/\hat{\omega} \cos \theta - \hat{v}/\hat{\omega} \cos(\theta + \hat{\omega}\Delta t) \\ \hat{\omega}\Delta t \end{pmatrix}$$



Normal distribution



Triangular distribution

[1]

- 2 velocities control 3 states \rightarrow all posterior poses are located on a 2D manifold within a 3D space \rightarrow degeneracy of the motion model
- We need to add third variable, a final rotation $\hat{\gamma}$

$$\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t \qquad \hat{\gamma} = \varepsilon_{\alpha_5} v^2 + \alpha_6 \omega^2$$

- Resulting motion model:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\hat{v}/\hat{\omega} \sin \theta + \hat{v}/\hat{\omega} \sin(\theta + \hat{\omega}\Delta t) \\ \hat{v}/\hat{\omega} \cos \theta - \hat{v}/\hat{\omega} \cos(\theta + \hat{\omega}\Delta t) \\ \hat{\omega}\Delta t + \hat{\gamma}\Delta t \end{pmatrix}$$

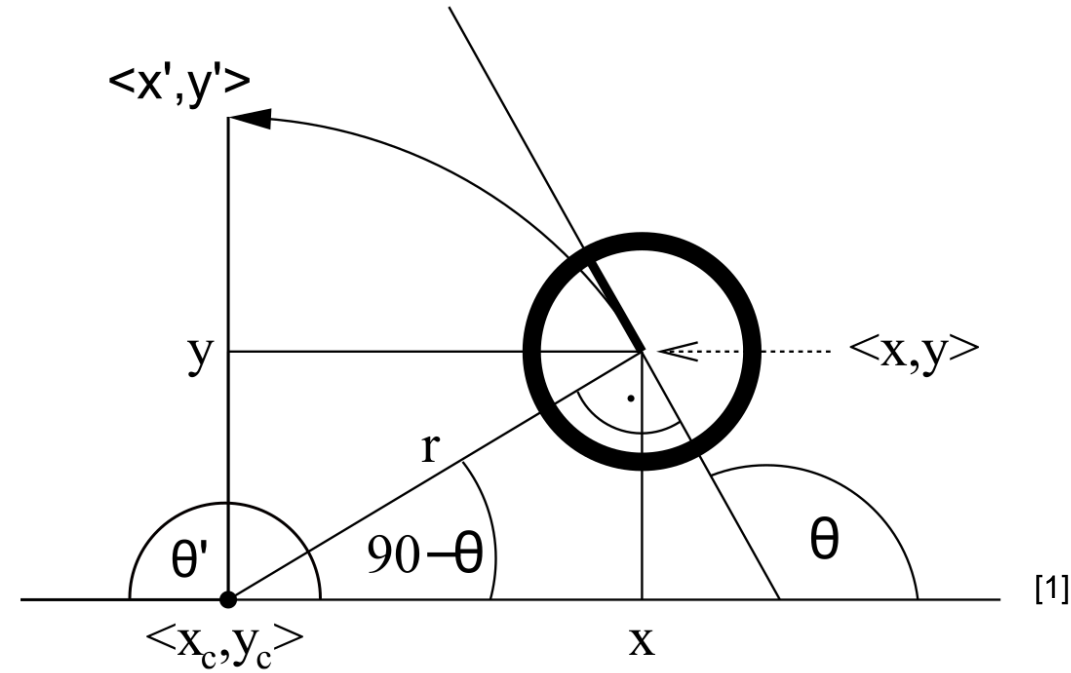
- To compute value $p(x_t|x_{t-1}, u_t)$ for arbitrary combination of arguments, we need an *inverse motion model*

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -\lambda \sin \theta \\ \lambda \cos \theta \end{pmatrix} = \begin{pmatrix} \frac{x + x'}{2} + \mu(y - y') \\ \frac{y + y'}{2} + \mu(x' - x) \end{pmatrix}$$

$$\mu = \frac{1}{2} \frac{(x - x') \cos \theta + (y - y') \sin \theta}{(y - y') \cos \theta - (x - x') \sin \theta}$$

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2} = \sqrt{(x' - x_c)^2 + (y' - y_c)^2}$$

$$\Delta \theta = \text{atan2}(y' - y_c, x' - x_c) - \text{atan2}(y - y_c, x - x_c)$$



- Inverse motion model – continuation

$$\begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \Delta t^{-1} \begin{pmatrix} r \cdot \Delta\theta \\ \Delta\theta \end{pmatrix}$$

$$\hat{\gamma} = \Delta t^{-1}(\theta' - \theta) - \hat{\omega}$$

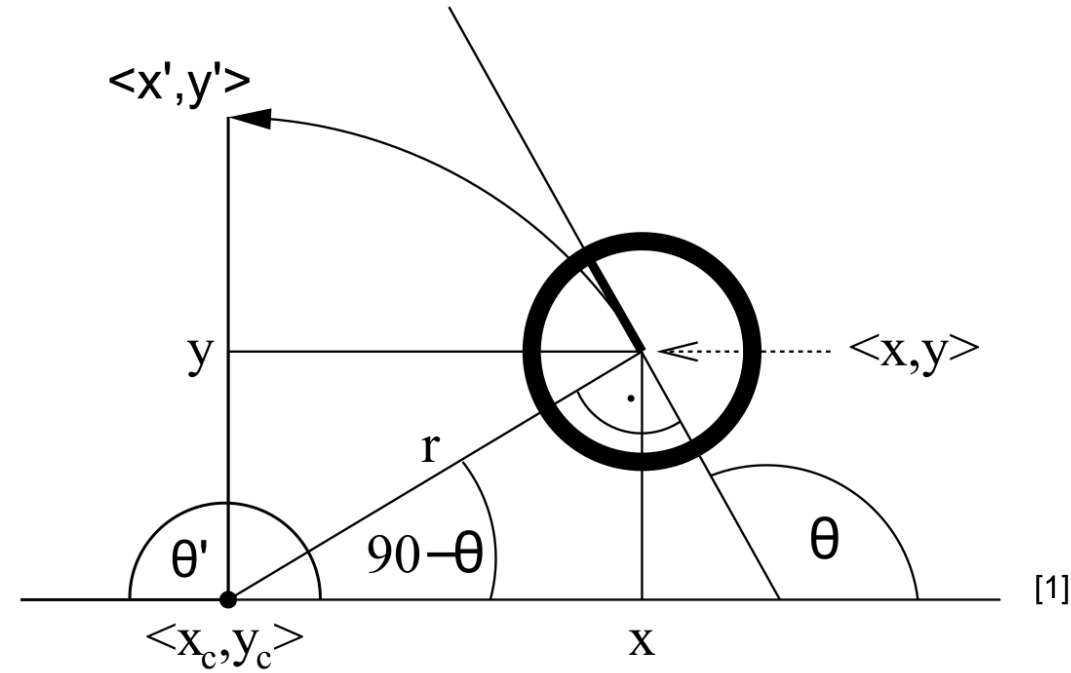
$$\varepsilon_1 = \varepsilon_{\alpha_1 v^2 + \alpha_2 \omega^2} (v - \hat{v})$$

$$\varepsilon_2 = \varepsilon_{\alpha_3 v^2 + \alpha_4 \omega^2} (\omega - \hat{\omega})$$

$$\varepsilon_3 = \varepsilon_{\alpha_5 v^2 + \alpha_6 \omega^2} (\hat{\gamma})$$

- Finally:

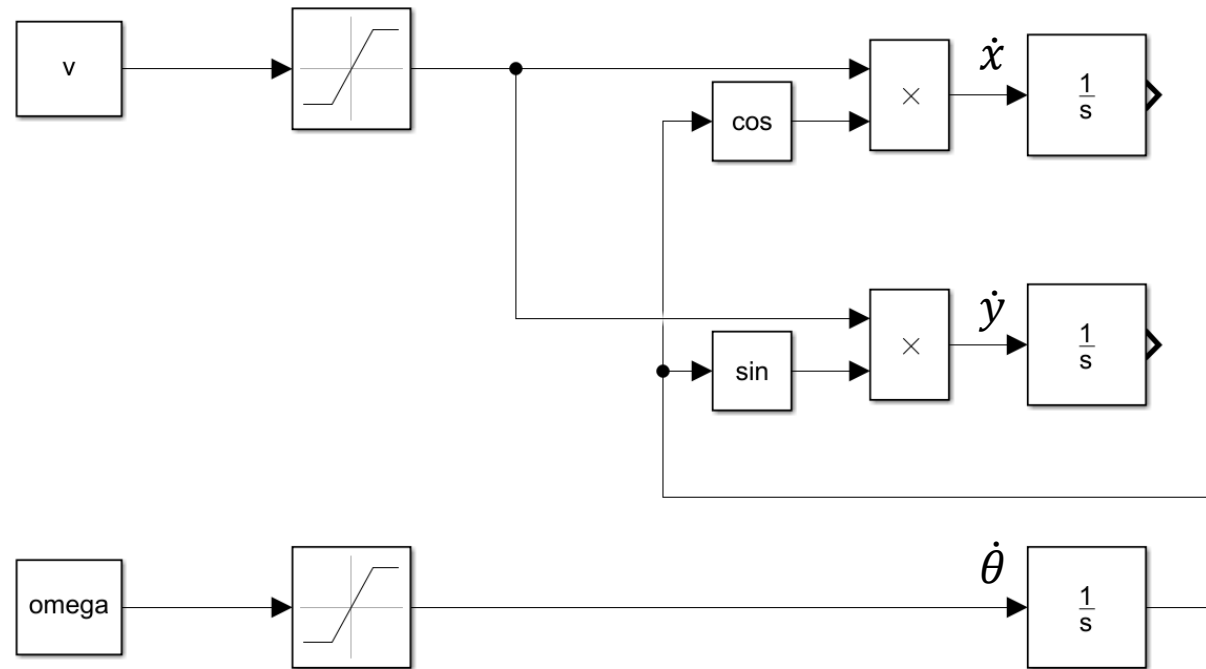
$$p(x_t | x_{t-1}, u_t) = \varepsilon_1 \cdot \varepsilon_2 \cdot \varepsilon_3$$





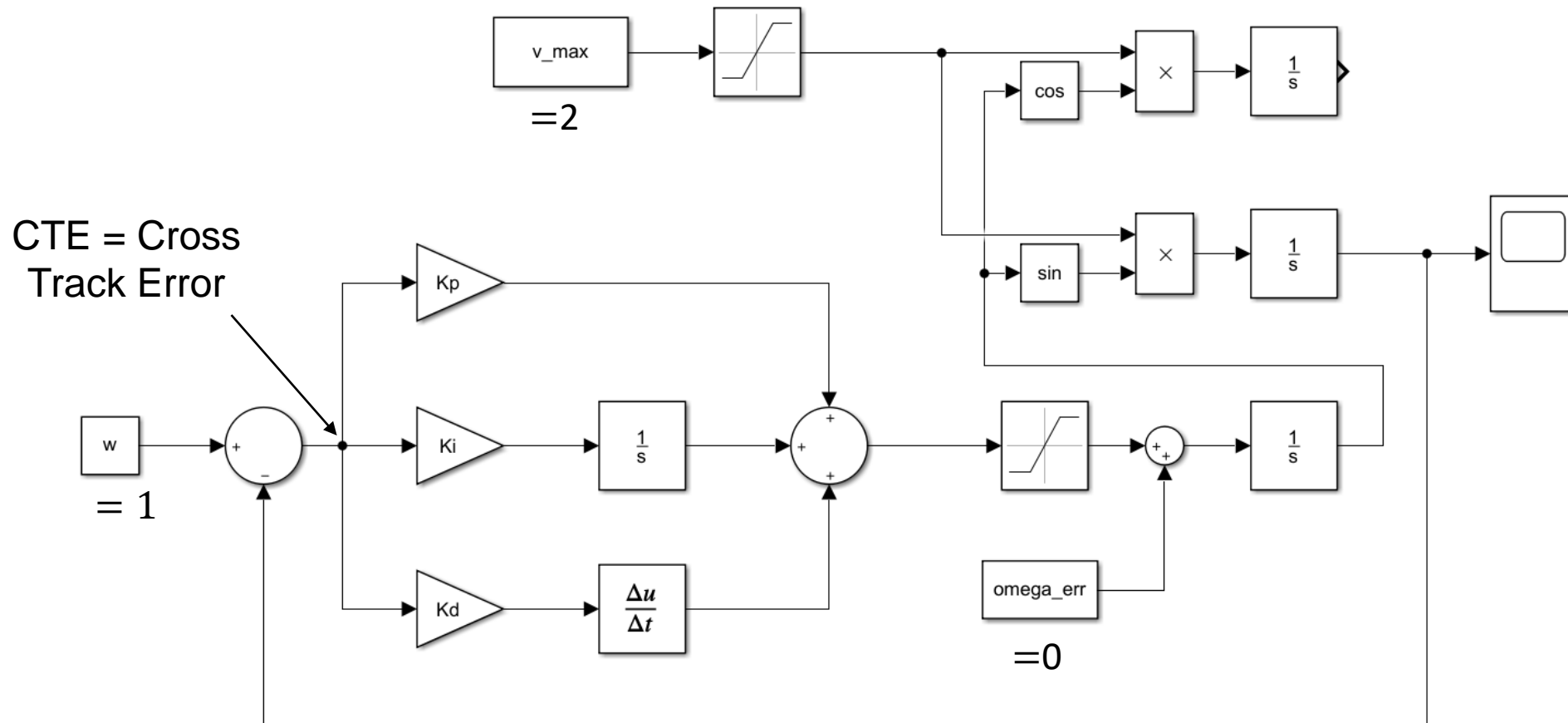
- Trajectory description
 - Closed-form expression
 - E. g., Bézier curves
 - Sequence of waypoints
 - Equivalent vs. adaptive spacing
- Global planning
 - Optimal path to goal
 - Outlined in Lecture 6 – Path planning
- **Local planning**
 - *Navigate* robot to the pre-planned path
 - Do not hit any obstacles!
 - Inputs: (estimated) pose, desired path, sensor readings
 - Parameters: kinematic model of the robot
 - Outputs: robot control = velocities

- State model of the differential drive:

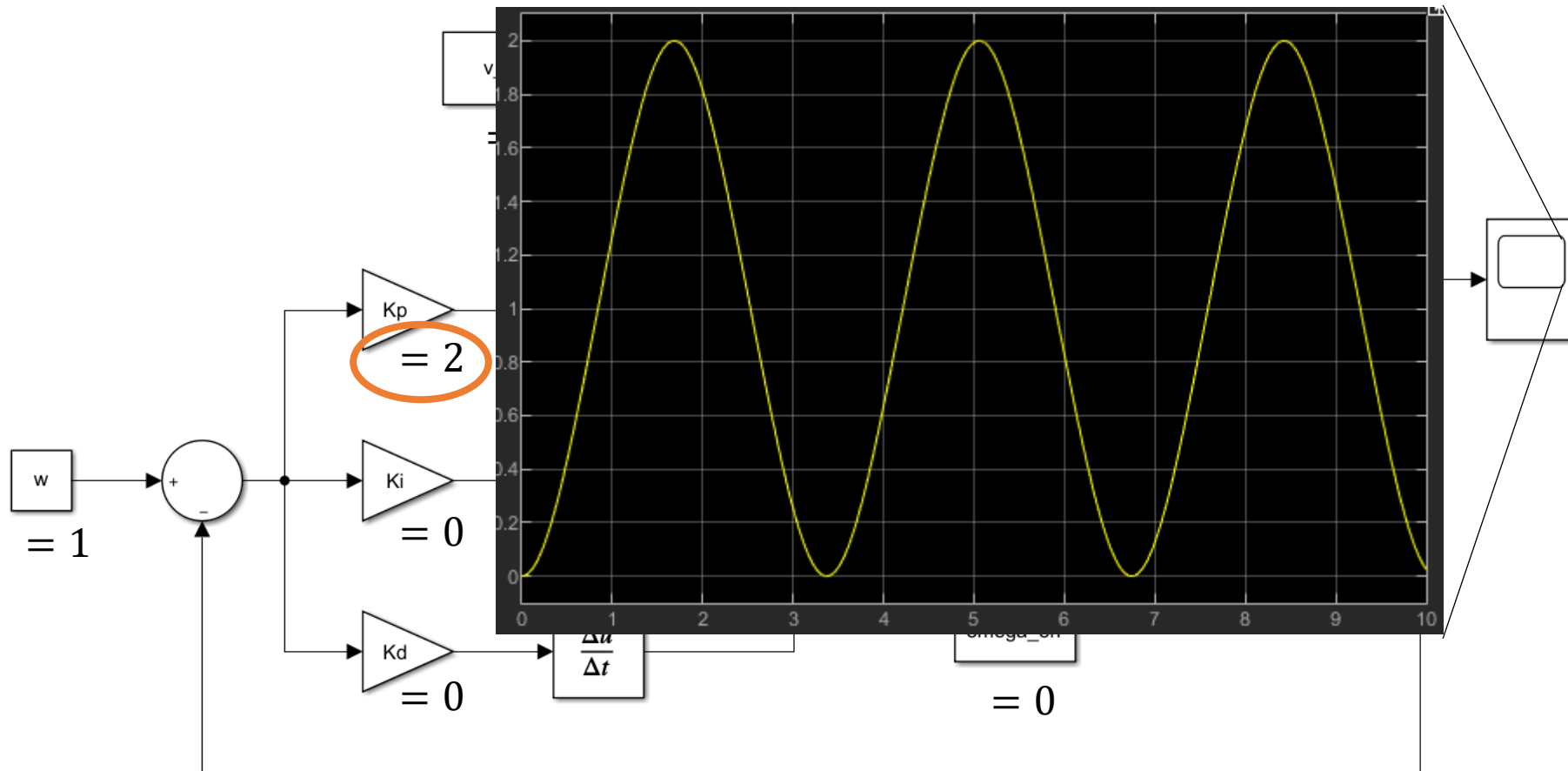




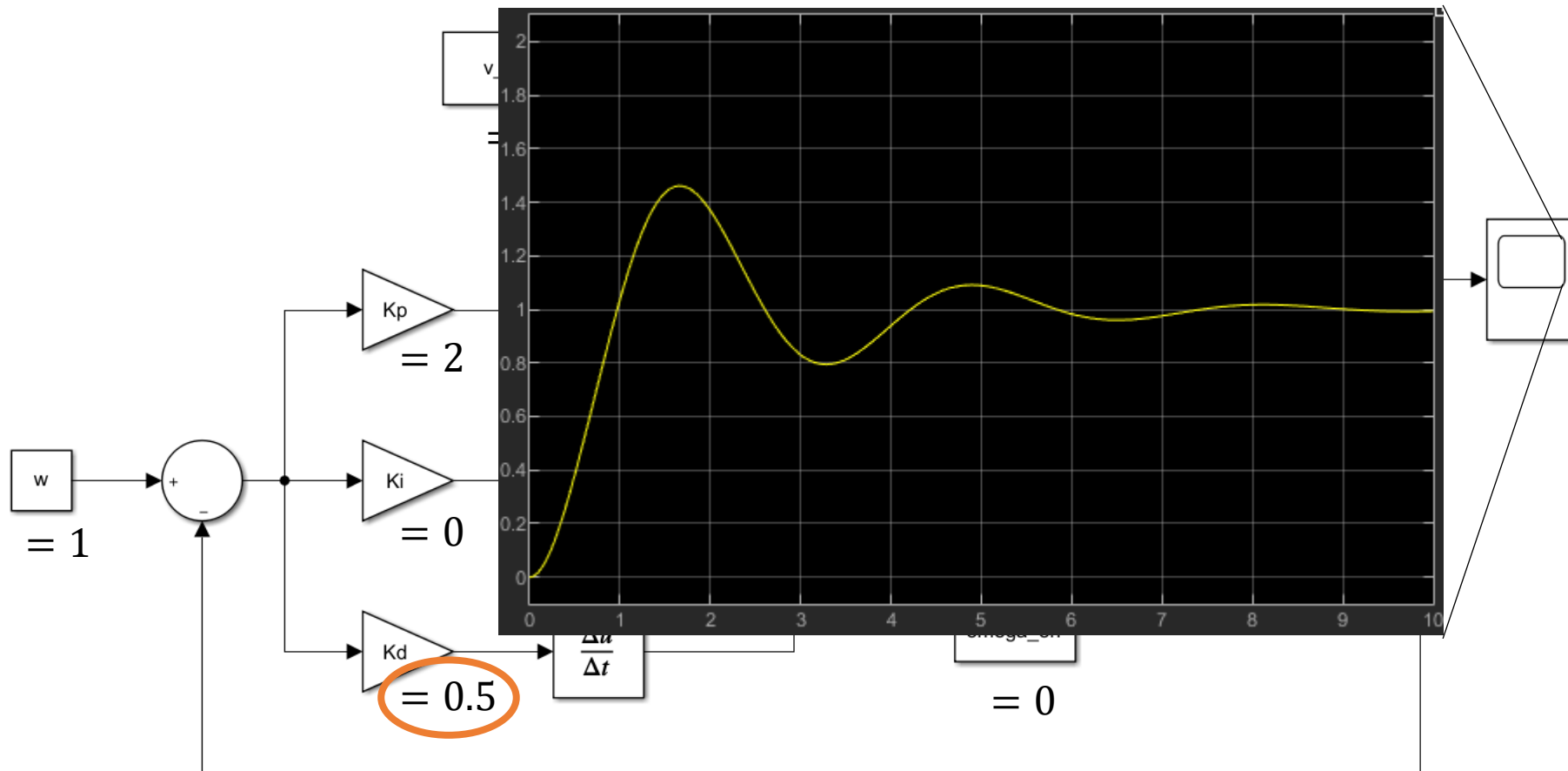
- Let us add a PID structure to control the y state



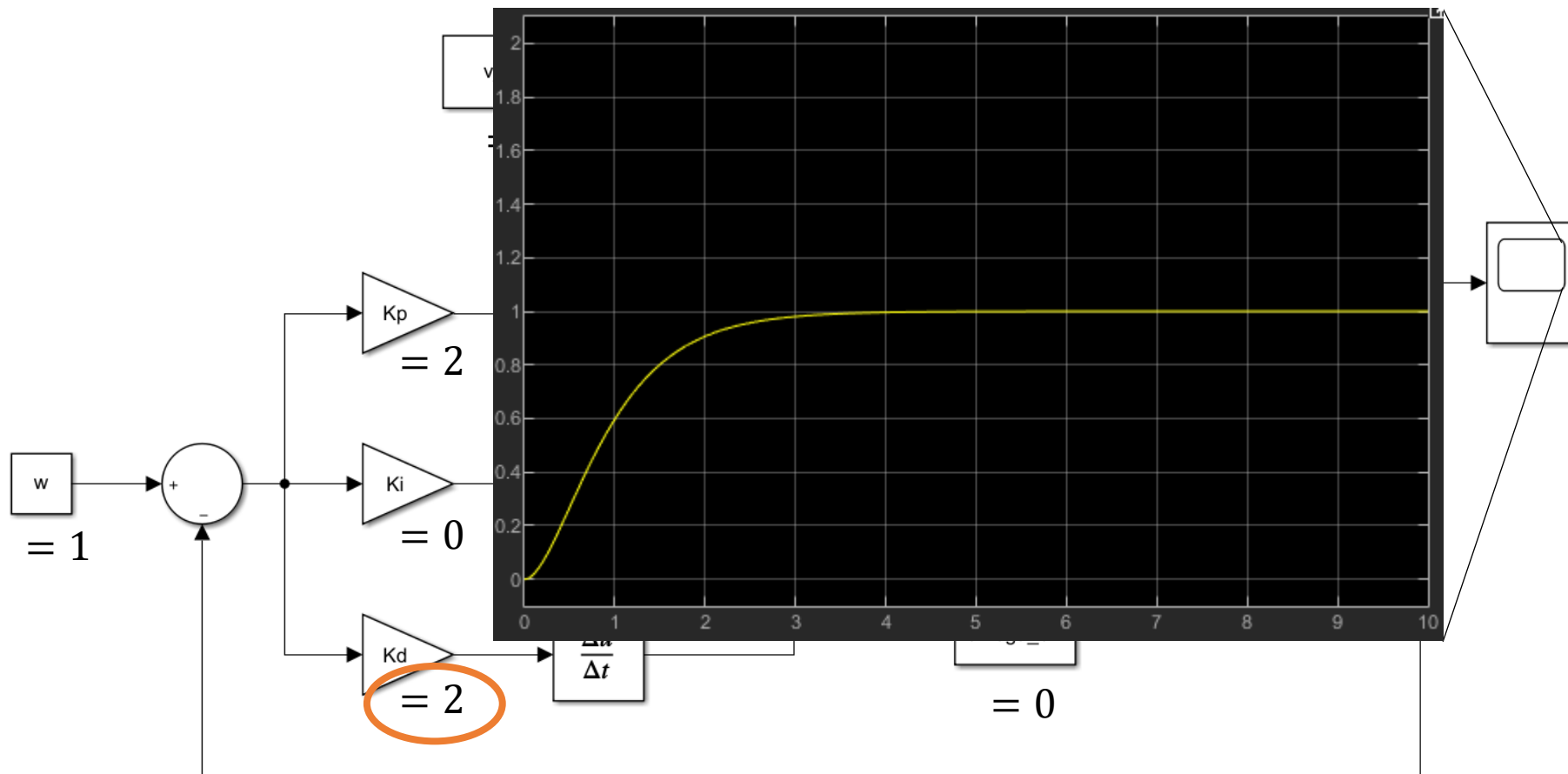
- P controller



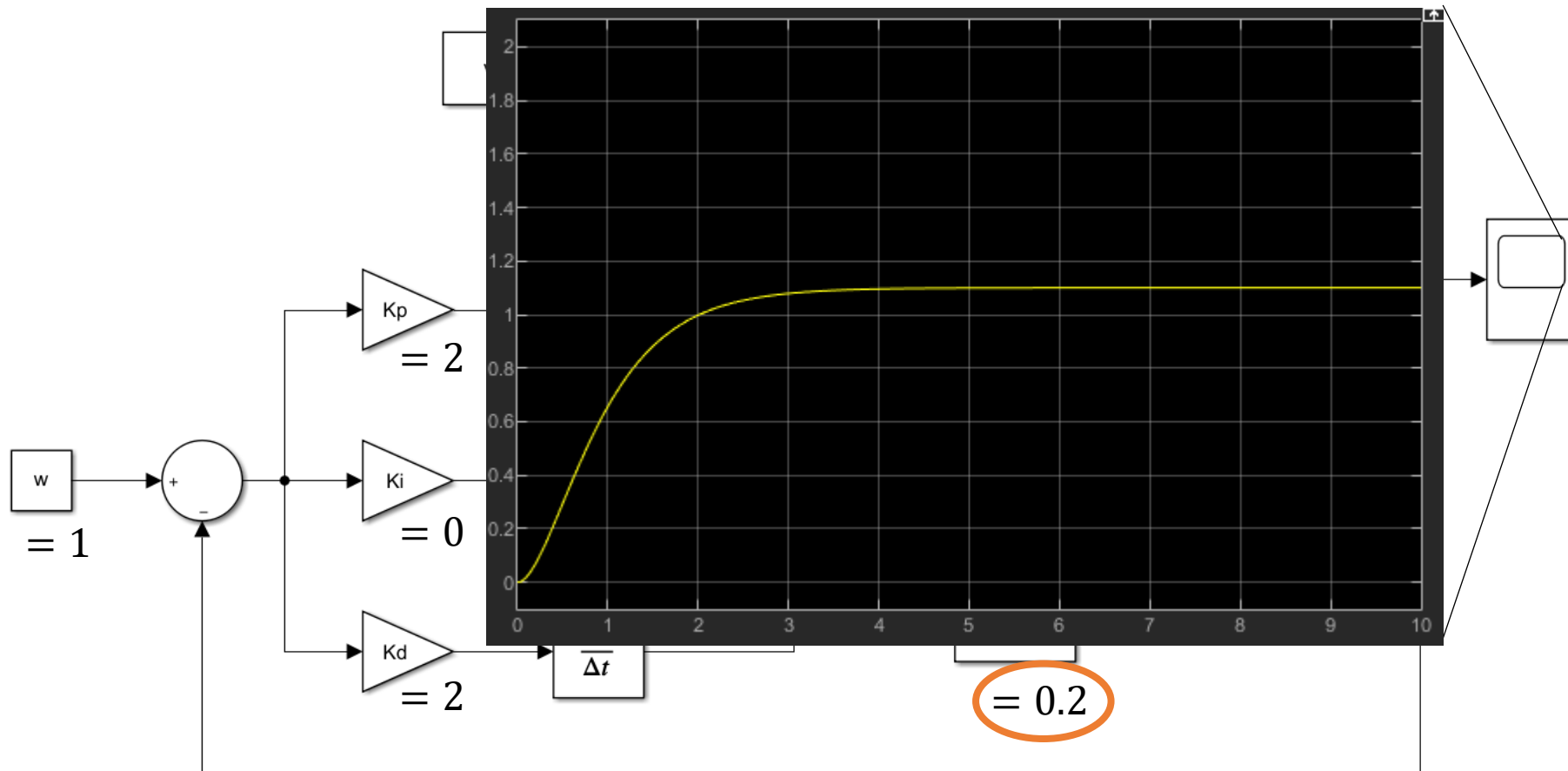
- PD controller



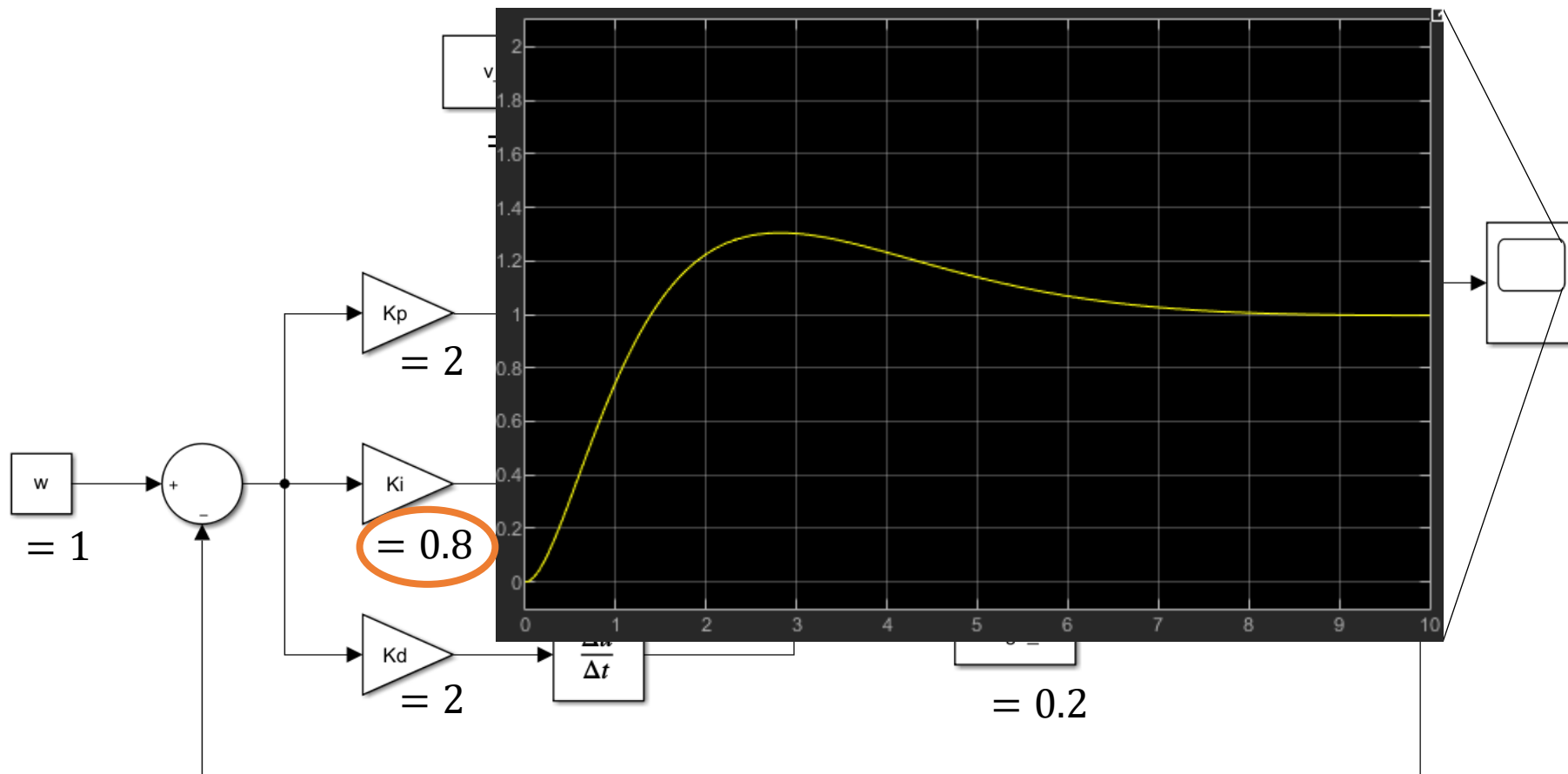
- PD controller – **higher D gain**



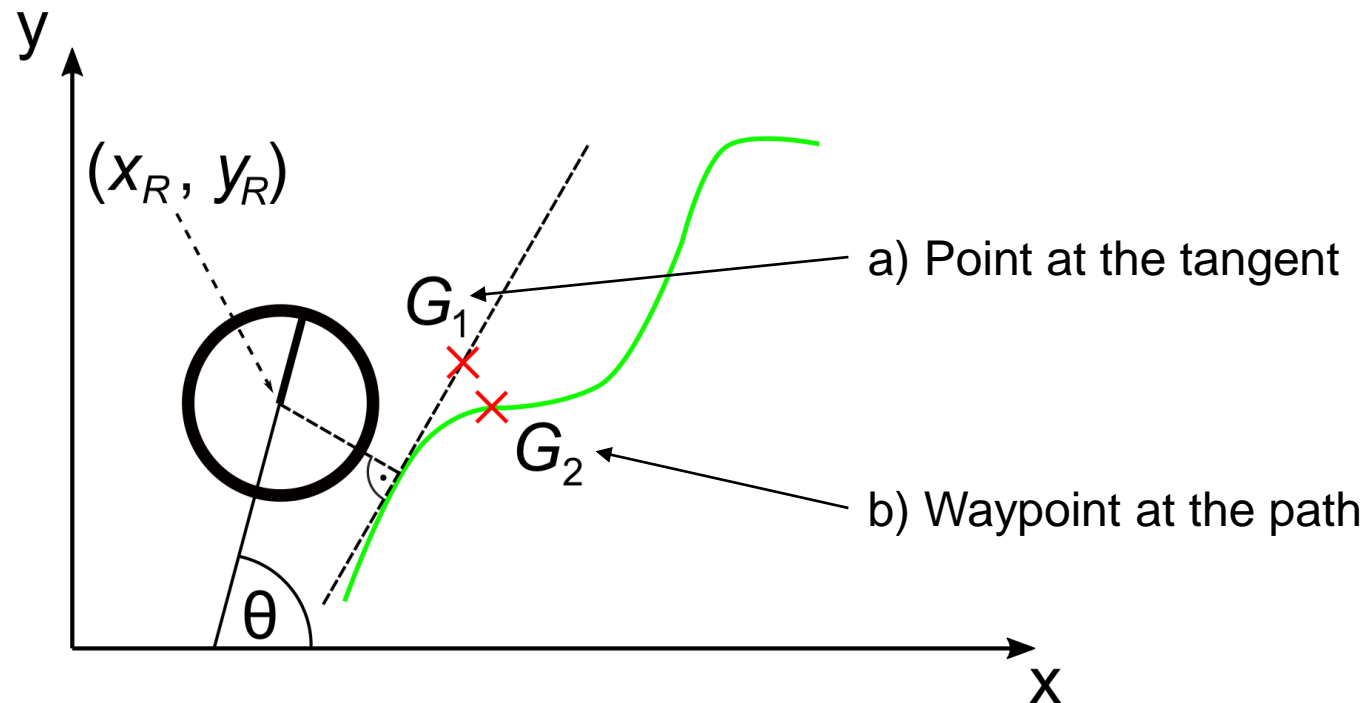
- PD controller – **with added error**



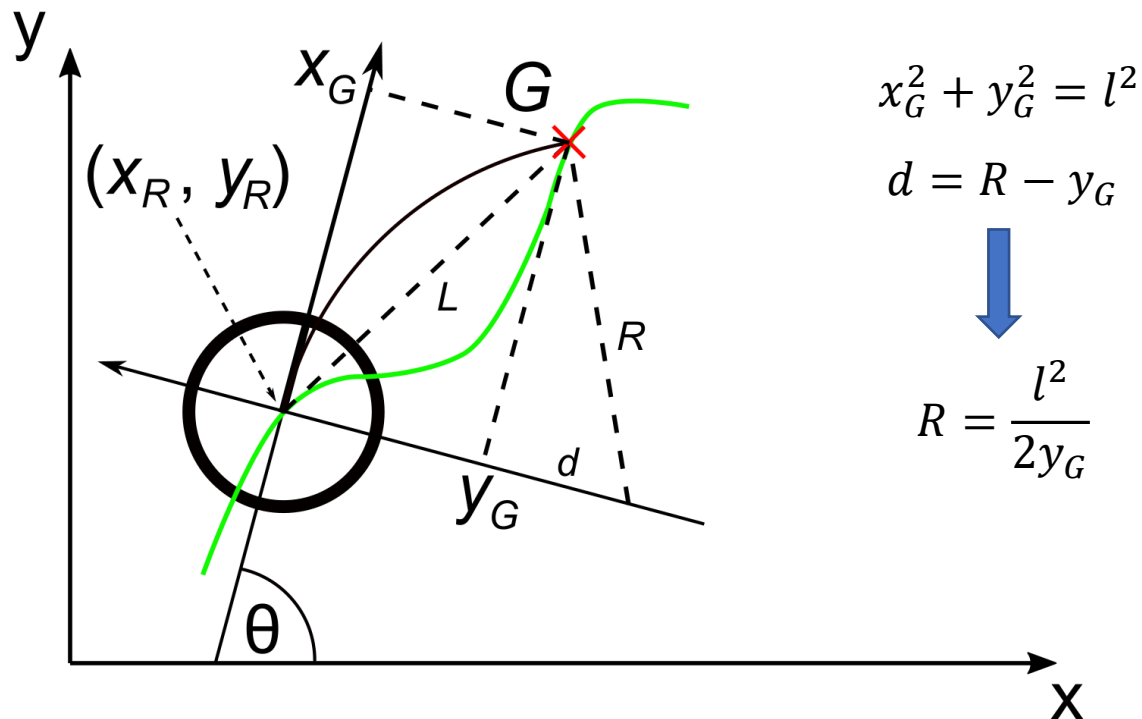
- PID controller – with added error



- How to navigate to trajectory in 2D?
- We will introduce two simple algorithms
 - Pure pursuit path tracking
 - Feedback linearization
- It is necessary to identify target/goal coordinates (x_G, y_G)



- Determine the curvature that will drive the vehicle to the chosen target
- The target is at *lookahead distance* l from the robot
- Constraints: v_{\max} , ω_{\max} , R_{\min} (minimal turning radius)



Ideal case

$$x_G^2 + y_G^2 = l^2$$

$$d = R - y_G$$

$$R = \frac{l^2}{2y_G}$$

$$\text{a) } |R| > R_{\min}$$

$$v = v_{\max}$$

$$\omega = \frac{v}{R}$$

$$|\omega| = \omega_{\max}$$

OR

$$v = |R| \cdot \omega_{\max}$$

$$\text{b) } |R| \leq R_{\min}$$

$$|\omega| = \omega_{\max}$$

$$v = R_{\min} \cdot \omega_{\max}$$

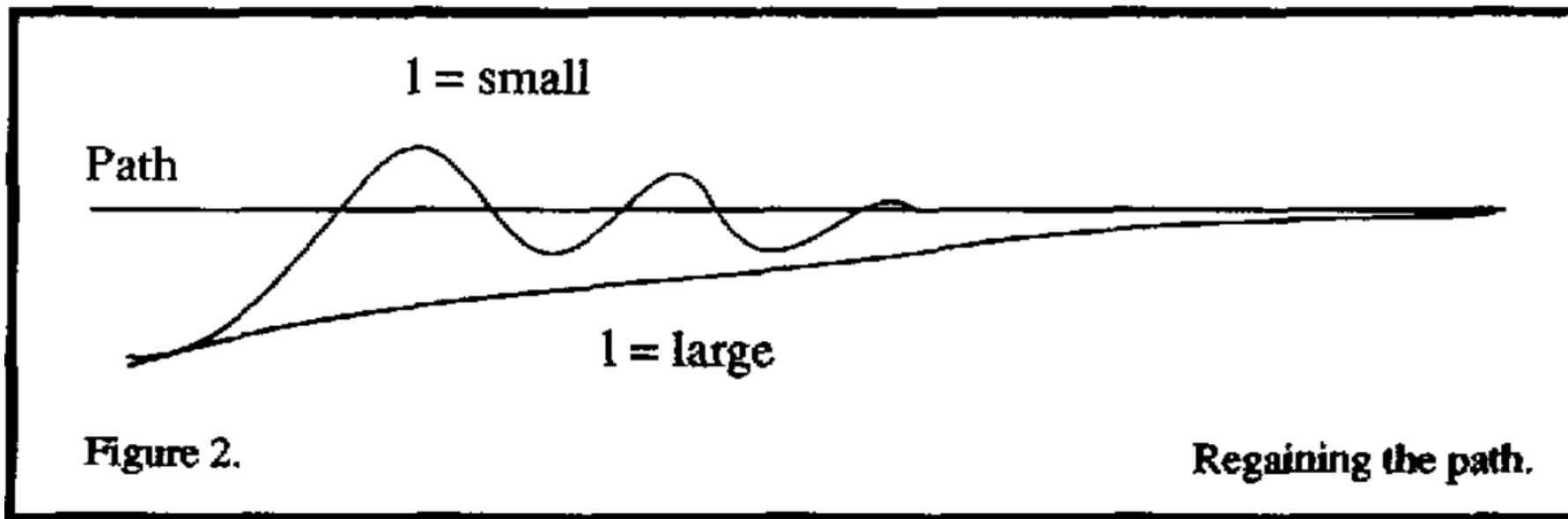
$$v = v_{\max}$$

OR

$$|\omega| = \frac{v_{\max}}{R_{\min}}$$

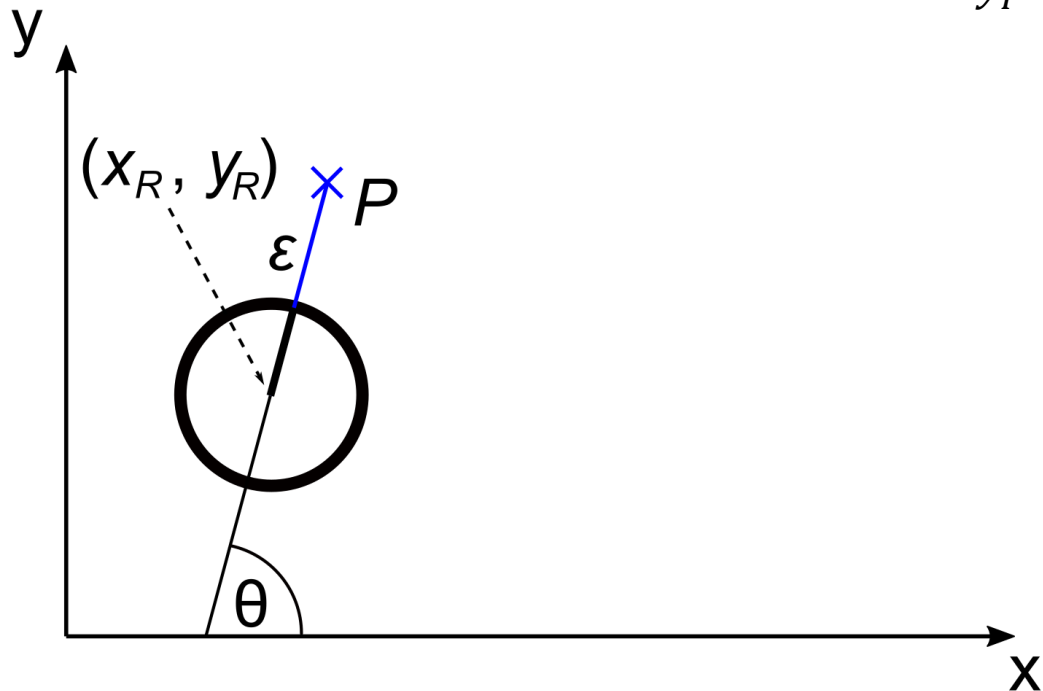
The goal is *not* reached

- The effect of changing the lookahead distance
- Similar to second order dynamic system (l acts as a damping factor)



[1]

- Linear control of holonomic point P to control a non-holonomic robot
- Rigid connection of the robot and P
- Key idea: $\begin{pmatrix} v \\ \omega \end{pmatrix} = f(\dot{x}_P, \dot{y}_P)$



$$\begin{aligned}
 x_P &= x_R + \epsilon \cos \theta \\
 y_P &= y_R + \epsilon \sin \theta
 \end{aligned}
 \xrightarrow{\frac{d}{dt}}
 \begin{aligned}
 \dot{x}_P &= \dot{x}_R + \epsilon(-\dot{\theta} \cos \theta) \\
 \dot{y}_P &= \dot{y}_R + \epsilon(\dot{\theta} \sin \theta)
 \end{aligned}
 \downarrow$$

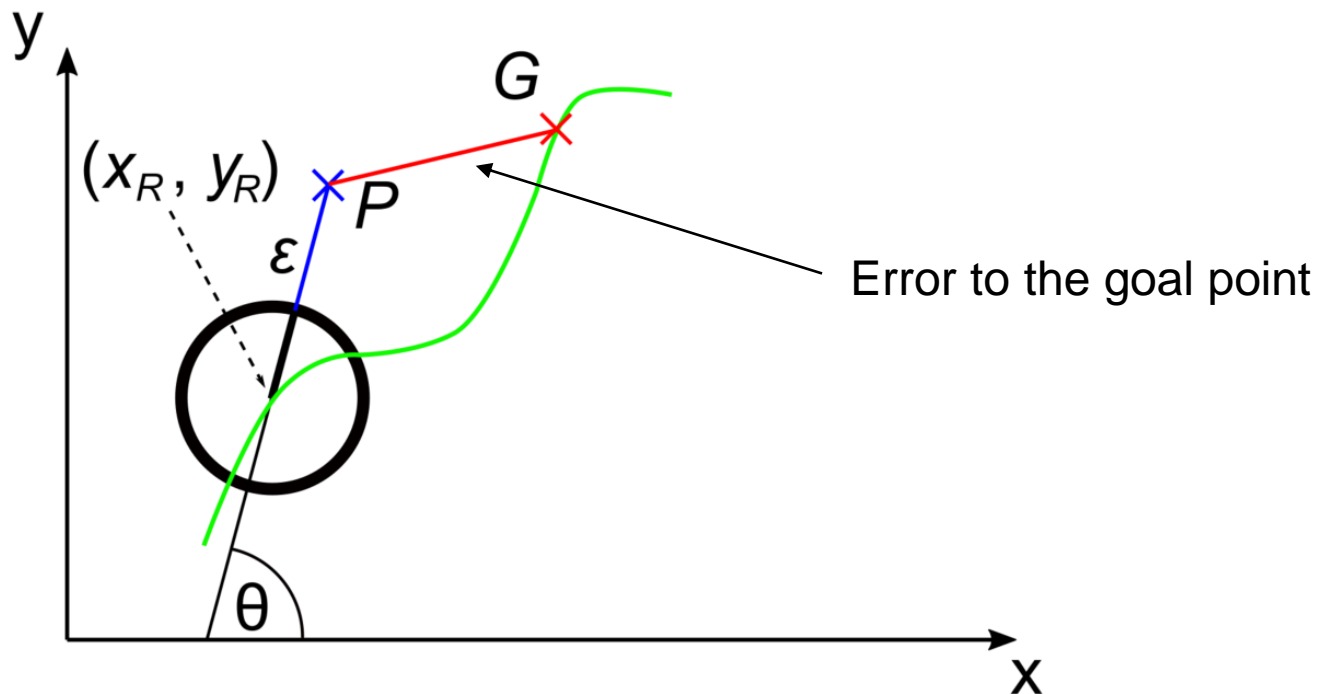
$$\begin{pmatrix} \dot{x}_P \\ \dot{y}_P \end{pmatrix} = v \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} + \epsilon \omega \begin{pmatrix} -\sin \theta \\ \cos \theta \end{pmatrix}$$

$$\downarrow$$

$$\begin{aligned}
 v &= \dot{x}_P \cos \theta + \dot{y}_P \sin \theta \\
 \omega &= \frac{1}{\epsilon} (-\dot{x}_P \sin \theta + \dot{y}_P \cos \theta)
 \end{aligned}$$

$$\begin{pmatrix} \dot{x}_P \\ \dot{y}_P \end{pmatrix} = \kappa \begin{pmatrix} x_G - x_P \\ y_G - y_P \end{pmatrix} + \begin{pmatrix} \dot{x}_G \\ \dot{y}_G \end{pmatrix}$$

← If goal is not stationary



Kinematics

- Types of drive: differential, Ackermann, omnidirectional, ...
- Holonomic vs. non-holonomic systems/constraints
- Computing forward and angular velocity from speed of wheels (and vice versa)
- Forward and inverse kinematic
- Probabilistic motion models

Following trajectory

- Trajectory description
- Selection of target point
- PID controller
- Pure pursuit tracking algorithm
- Feedback linearization



[1]



Profile

Tomas Lazna

Position: Ph.D. Student @ FEEC,
Junior Researcher @CEITEC

Research Topic: Radiation mapping via
robotic platforms

Room: SE1.102

Contact: Chat @ MS Teams,
tomas.lazna@ceitec.vutbr.cz

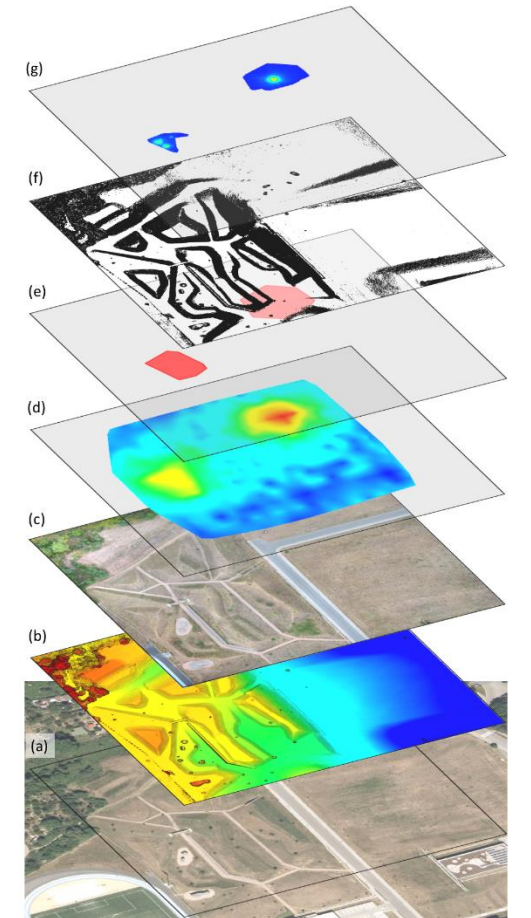
Background:

- Motion planning in mobile robotics
- Radiation data processing
- Cooperation of UASs and UGVs
- Estimation problems



Hobbies and interests:

- Star Wars & science fiction
- LEGO
- Politics





Tomas Lazna

tomas.lazna@ceitec.vutbr.cz

Brno University of Technology
Faculty of Electrical Engineering and Communication
Department of Control and Instrumentation



Robotics and AI Research Group