BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

1 – Introduction to Course

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

Course supervisor: Ing. Lukáš Kopečný, Ph.D.

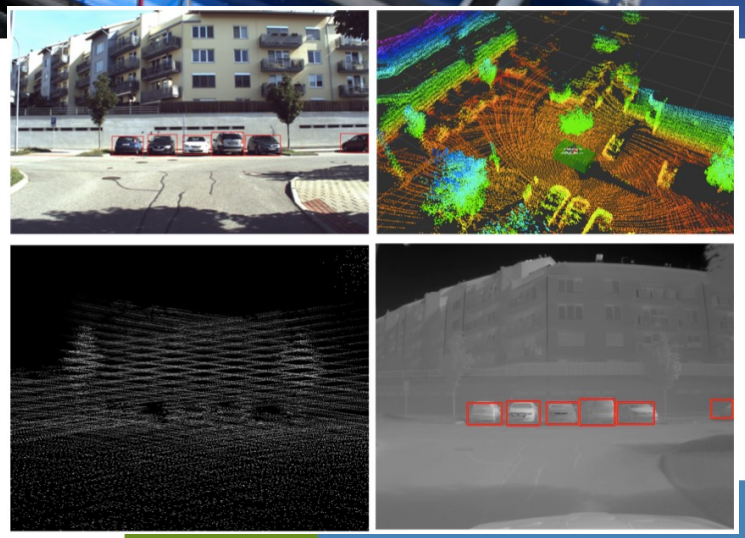
Adam Ligocki

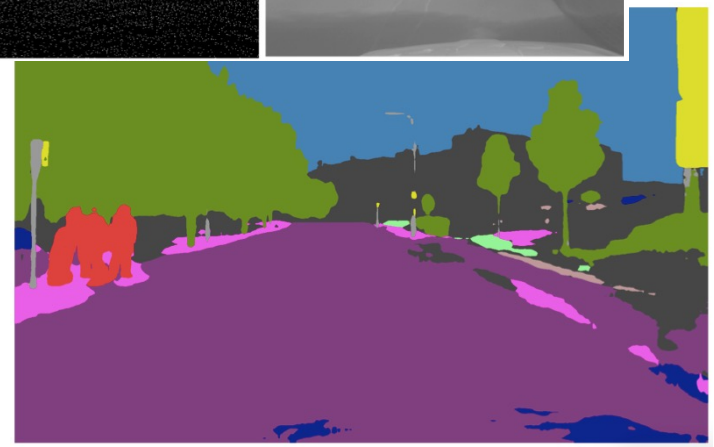
Brno University of Technology
2021

Robotics and AI

Profile

Robotics and AI





Ing. Adam Ligocki, Ph.D.

Position: Research Staff

Research: Data Fusion

Room: SE1.102

Background:

- Artificial Intelligence
- Neural Networks
- Software Development

Web: https://www.vut.cz/lide/adam-ligocki-154791

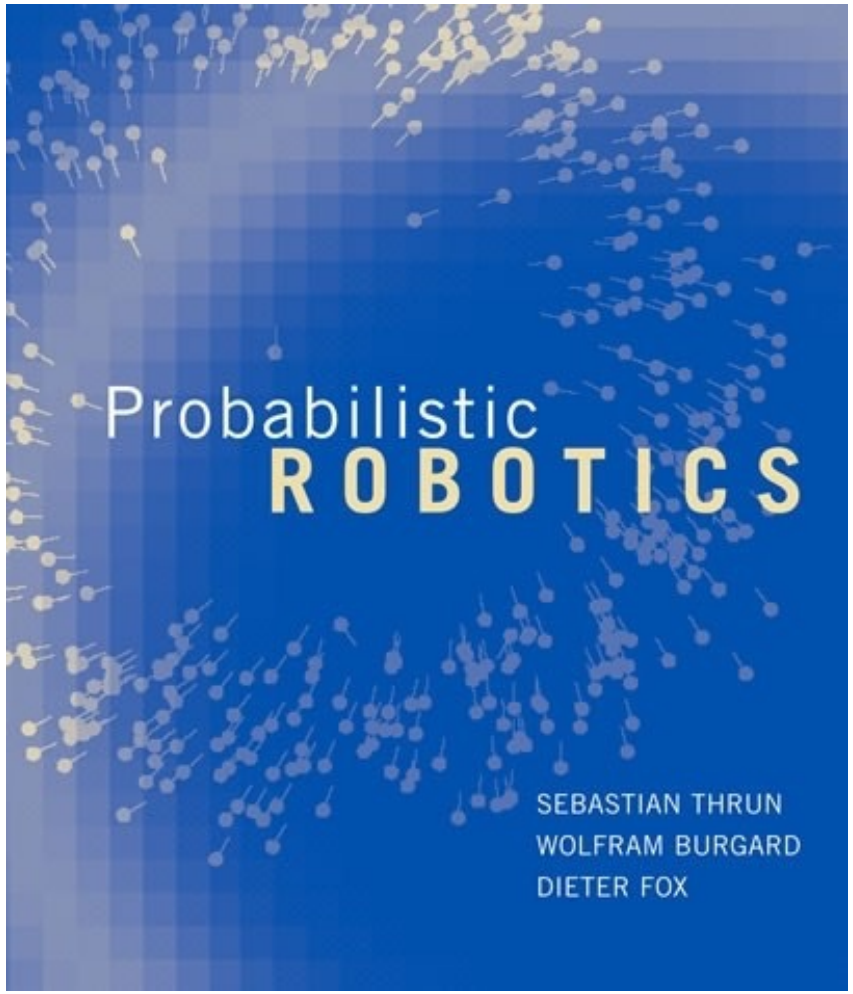
Book – Probabilistic Robotics

Probabilistic Robotics

[Probabilistic Robotics,
Intelligent Robotics and Autonomous Agents series,
Sebastian Thrun, Wolfram Burgard, Diete Fox,
ISBN: 0262201623, MIT Press, 19. 8. 2005, 672p]

Hardcover available in BUT FEKT's library

Early version available from:
https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf



Course Details

About the Course

Motivation

- Introduction to basic sensor modeling, localization and mapping algorithms used in modern robotic systems.

Parameters

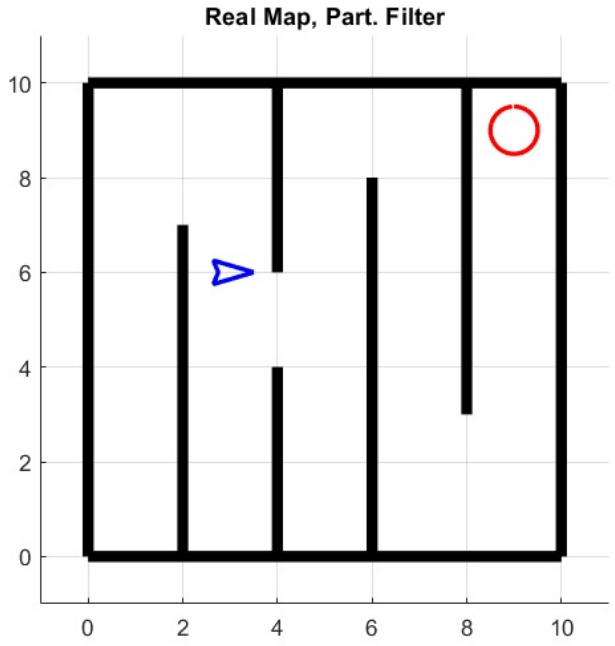
- 3 credits, graded assessment, (ECTS – European Credit Transfare System)
- Total work: **75h** (~25-30h/credit)
- Guided work: 7x 2h lectures, 6x 2h labs
- Individual work: **~49** h
- Labs are in home-office mode.
- Labs hours are optional and dedicated for consultation SD1.51.

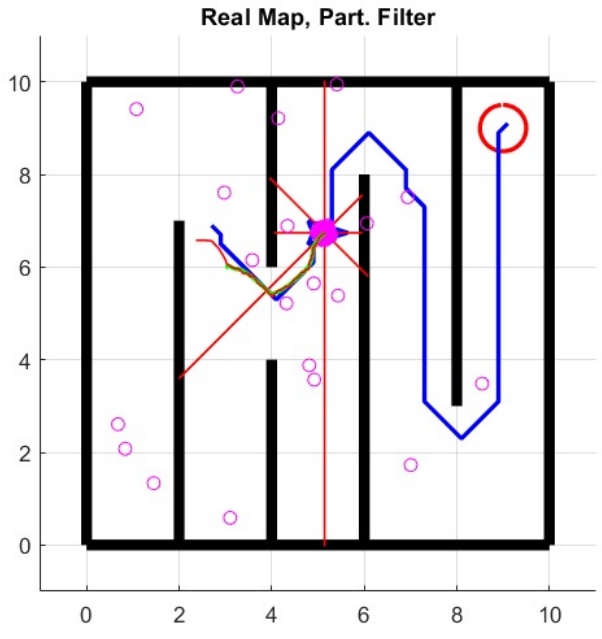
Study sources

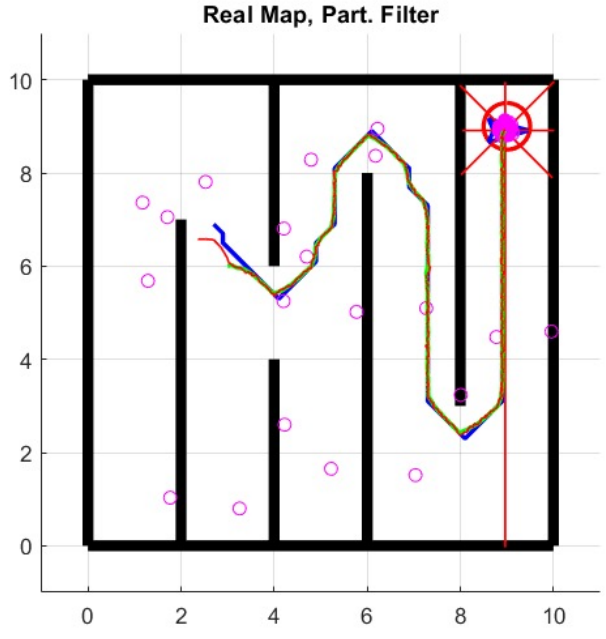
- Lectures
- Labs / Consultations
- Sebastian Thrun: Probabilistic Robotics (Book)
- Sebastian Thrun: Artificial Intelligence for Robotics (Udacity)
- Cyrill Stachniss: SLAM course (Youtube)

Semestral project overview:

- Control robot in the simulated environment
  - Generic differential chassis
- Self-localization in the known map
  - Sensor - simple 2D **LiDAR** (8 rays with fixed-position)
  - Implementing **particle filter** to estimate robot‘s pose
- Path planning (**A***) and trajectory following







Course Structure

	Lecture	Labs	Points
Week 1:	Introduction	-	-
Week 2:	Sensors, Probablility	Sensor model, Simulator	10
Week 3:	Motion Control	Motion Control	10
Week 4:	Particle Filter	PF - implementation	10
Week 5:	Path planning	A* - implementation	10
Week 6:	Kalman Filter	KF - implementation	10
Week 7:	SLAM, EKF SLAM	Individual work on project	50

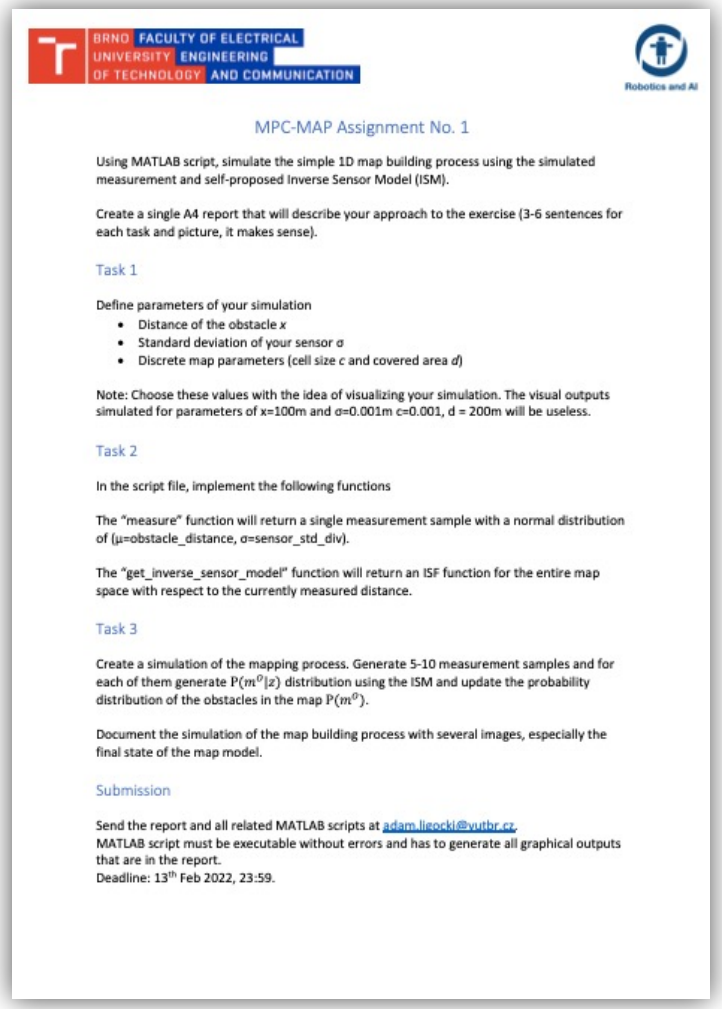
Assignments

A **single A4** assignment for weeks 2 to 6.

Several lecture related tasks to implement in MATLAB.

For each assignment you will prepare a **single A4** report in which you will document your results and technical approach of your solution.

Each taks documented by **3-6 sentences** and **image**(s) if it make sense.



BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Robotics and AI

MPC-MAP Assignment No. 1

Using MATLAB script, simulate the simple 1D map building process using the simulated measurement and self-proposed Inverse Sensor Model (ISM).

Create a single A4 report that will describe your approach to the exercise (3-6 sentences for each task and picture, it makes sense).

Task 1

Define parameters of your simulation

- Distance of the obstacle *x*
- Standard deviation of your sensor σ
- Discrete map parameters (cell size *c* and covered area *d*)

Note: Choose these values with the idea of visualizing your simulation. The visual outputs simulated for parameters of x=100m and σ=0.001m c=0.001, d = 200m will be useless.

Task 2

In the script file, implement the following functions

The "measure" function will return a single measurement sample with a normal distribution of (μ=obstacle_distance, σ=sensor_std_div).

The "get_inverse_sensor_model" function will return an ISF function for the entire map space with respect to the currently measured distance.

Task 3

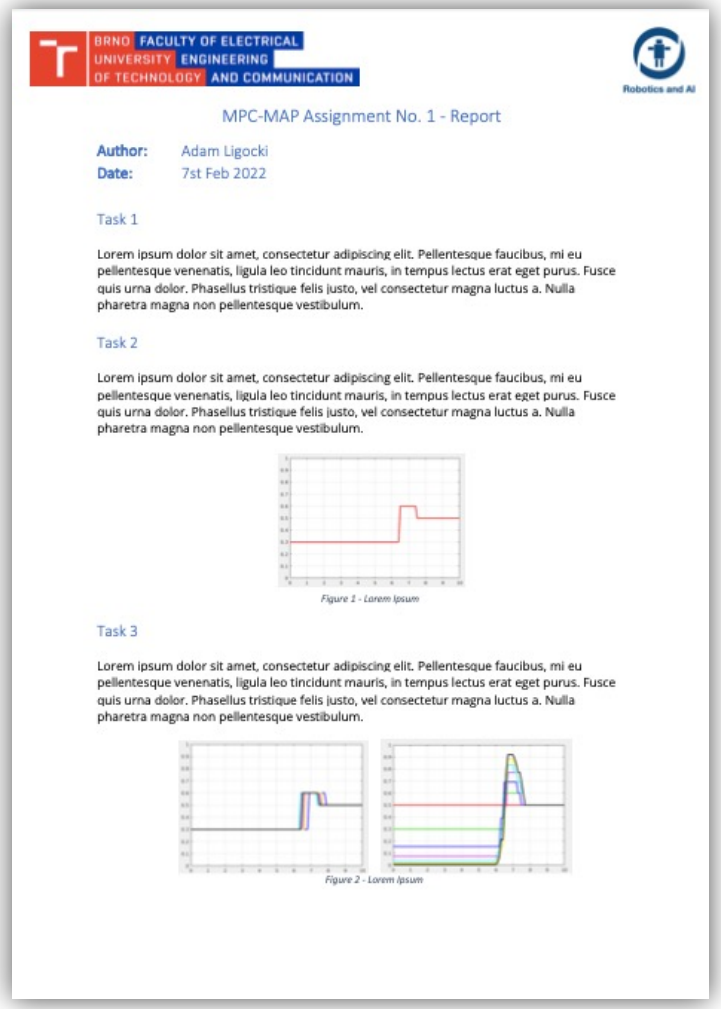
Create a simulation of the mapping process. Generate 5-10 measurement samples and for each of them generate $P(m^0|z)$ distribution using the ISM and update the probability distribution of the obstacles in the map $P(m^0)$.

Document the simulation of the map building process with several images, especially the final state of the map model.

Submission

Send the report and all related MATLAB scripts at adam.ligocki@vutbr.cz.
MATLAB script must be executable without errors and has to generate all graphical outputs that are in the report.
Deadline: 13th Feb 2022, 23:59.

Assignment



BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Robotics and AI

MPC-MAP Assignment No. 1 - Report

Author: Adam Ligocki
Date: 7st Feb 2022

Task 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque faucibus, mi eu pellentesque venenatis, ligula leo tincidunt mauris, in tempus lectus erat eget purus. Fusce quis urna dolor. Phasellus tristique felis justo, vel consectetur magna luctus a. Nulla pharetra magna non pellentesque vestibulum.

Task 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque faucibus, mi eu pellentesque venenatis, ligula leo tincidunt mauris, in tempus lectus erat eget purus. Fusce quis urna dolor. Phasellus tristique felis justo, vel consectetur magna luctus a. Nulla pharetra magna non pellentesque vestibulum.

Figure 1 - Lorem ipsum

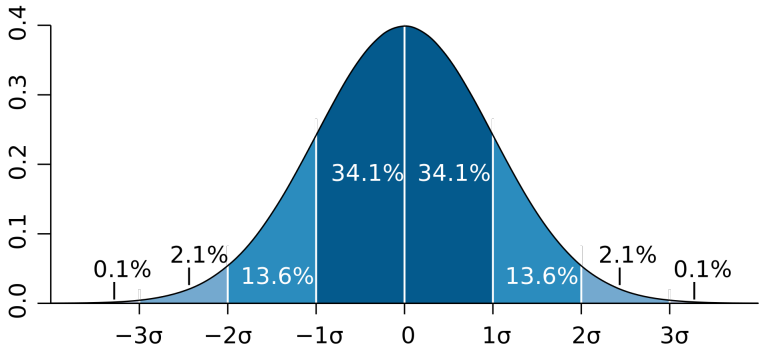
Task 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque faucibus, mi eu pellentesque venenatis, ligula leo tincidunt mauris, in tempus lectus erat eget purus. Fusce quis urna dolor. Phasellus tristique felis justo, vel consectetur magna luctus a. Nulla pharetra magna non pellentesque vestibulum.

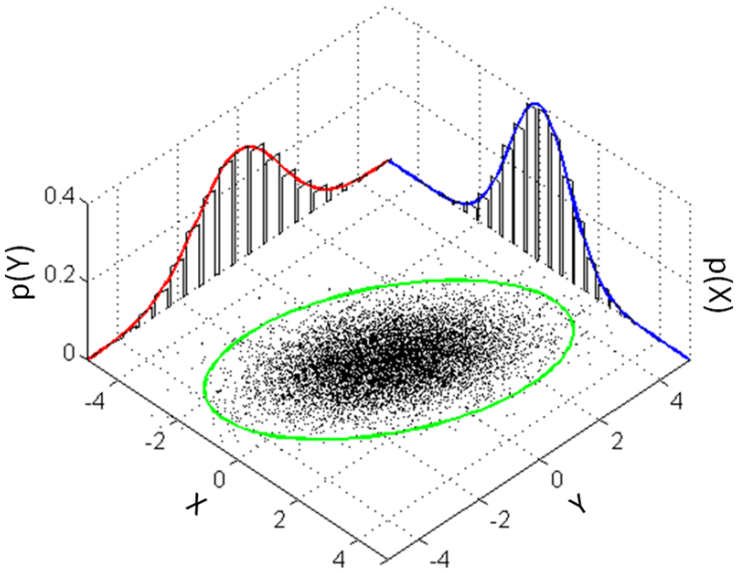
Figure 2 - Lorem ipsum

Report (placeholder)

Week 2: Probability, Sensors



Probability Distribution



Multi-dimensional Distributions

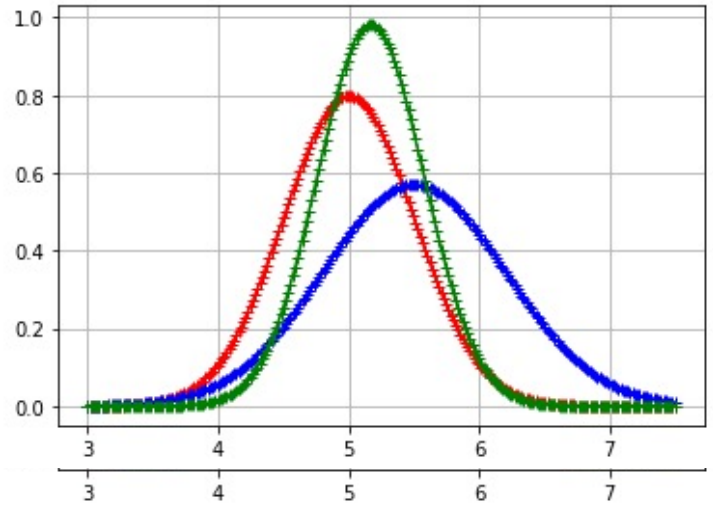
Bayes Probability

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

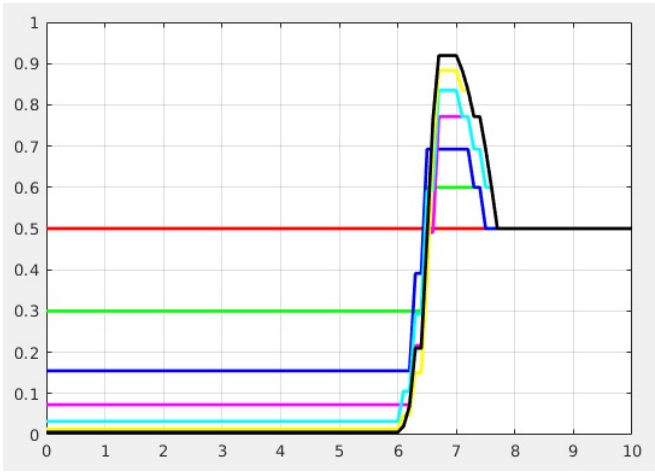
Bayes Filter

$$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$

$$bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)$$

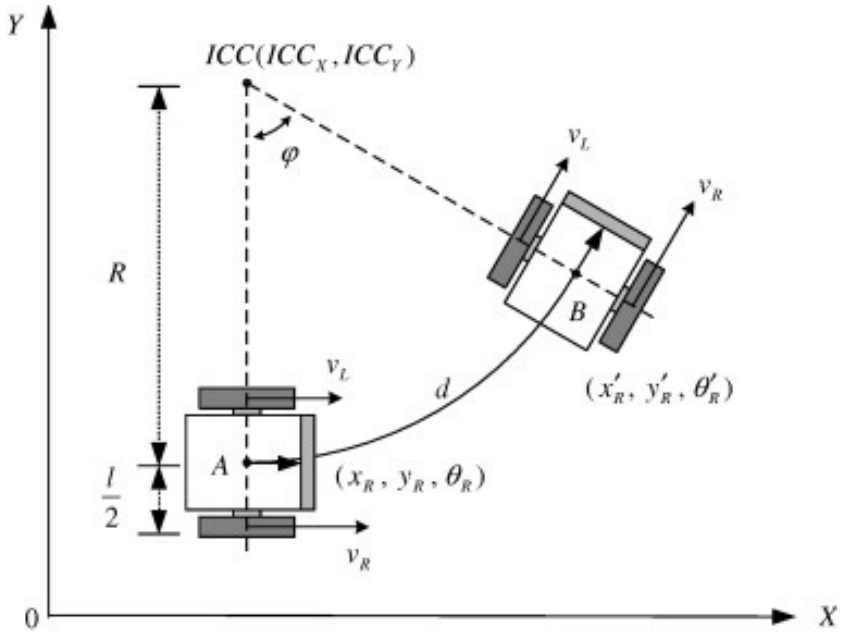


Combining Probabilities

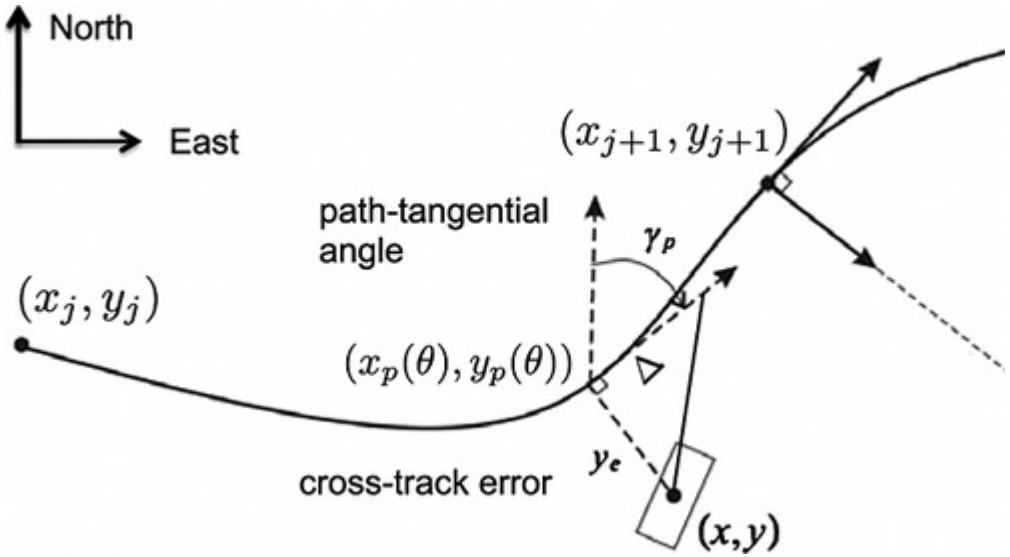


Sensor Modeling

- Kinematics of a differential drive
- Motion model
- How to follow a path?



[1]

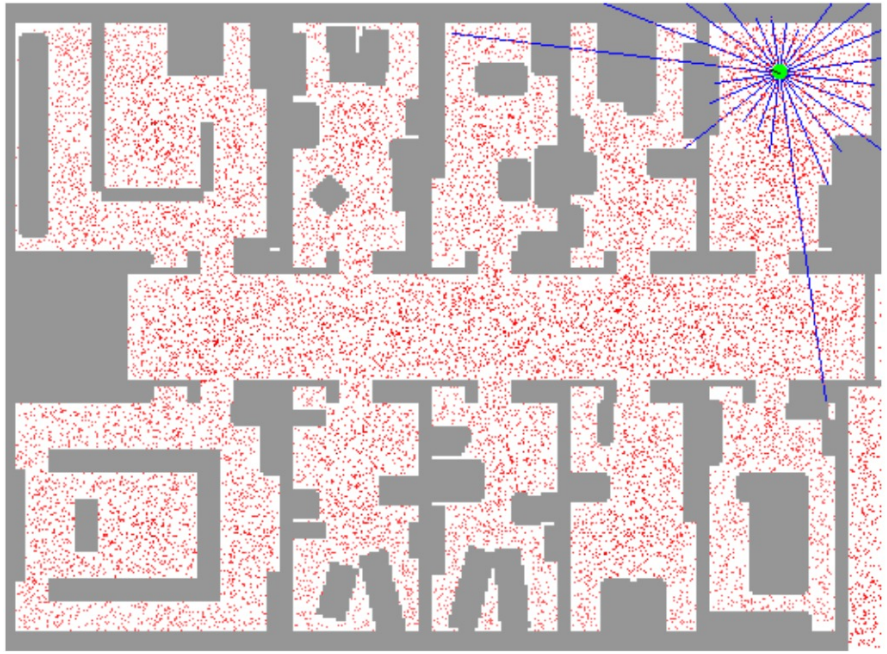


[2]

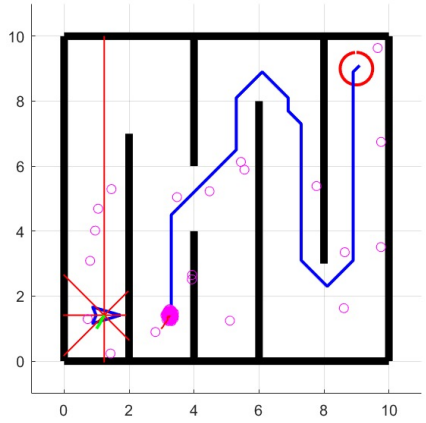
[1] HAN, Soonshin, ByoungSuk CHOI a JangMyung LEE. A precise curved motion planning for a differential driving mobile robot. *Mechatronics* [online]. 2008, **18**(9), 486-494 [cit. 2022-02-04]. DOI: 10.1016/j.mechatronics.2008.04.001
[2] FOSSEN, Thor I. a Anastasios M. LEKKAS. Direct and indirect adaptive integral line-of-sight path-following controllers for marine craft exposed to ocean currents. *International Journal of Adaptive Control and Signal Processing* [online]. 2017, **31**(4), 445-463 [cit. 2022-02-04]. DOI: 10.1002/acs.2550

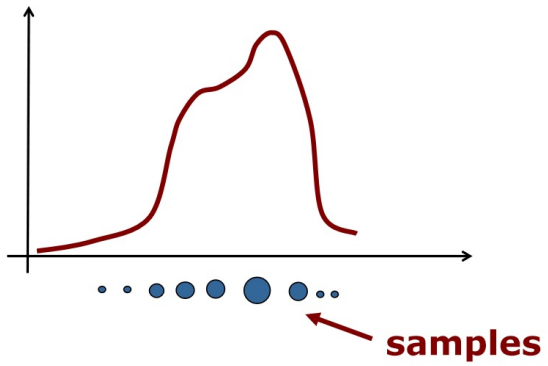
Week 4: Particle Filter

- Introduction to Monte Carlo methods
- What are particles?
- Random numbers sampling – a vital element
- Particle filter algorithm – 3 steps:
  - Prediction
  - Correction
  - Resampling
- Application of the PF to the localization problem
- Typical issues and how to solve them

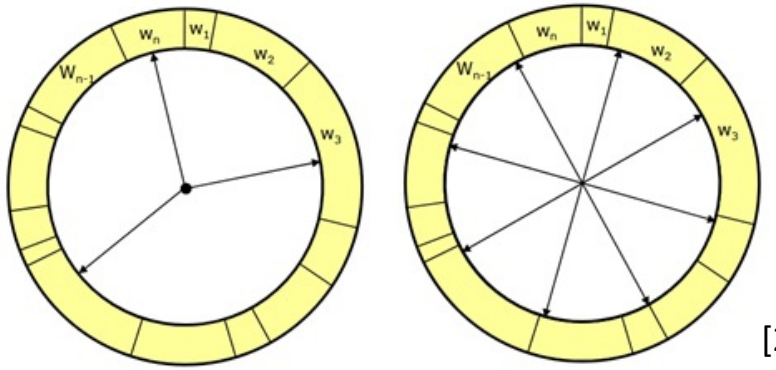


[1]





[2]

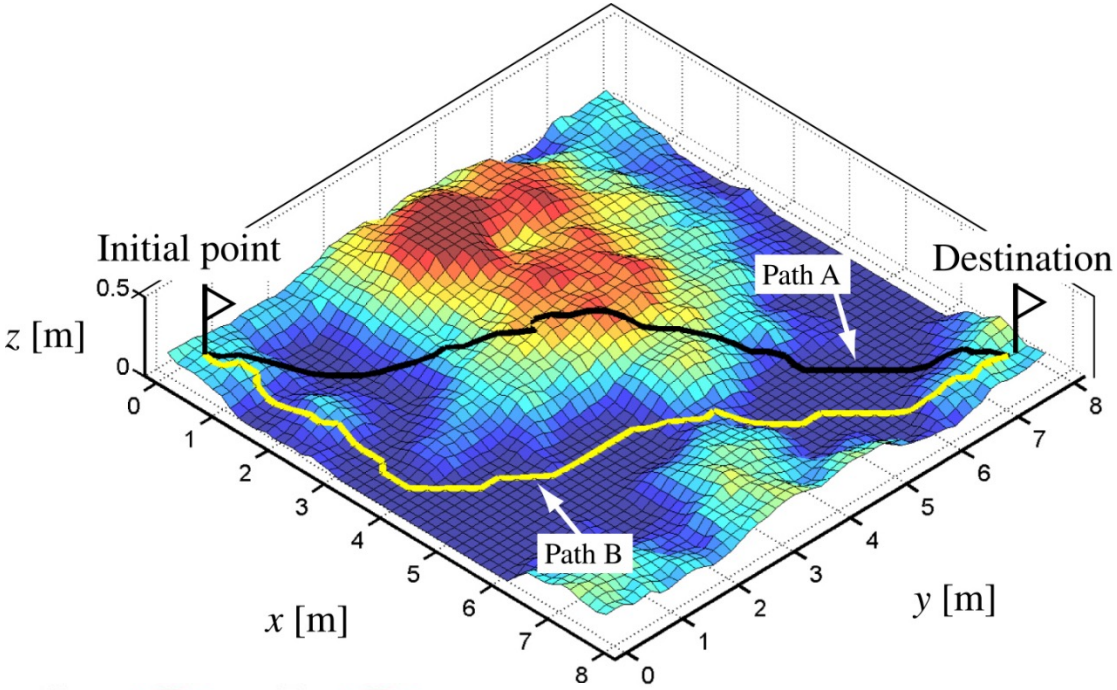


[2]

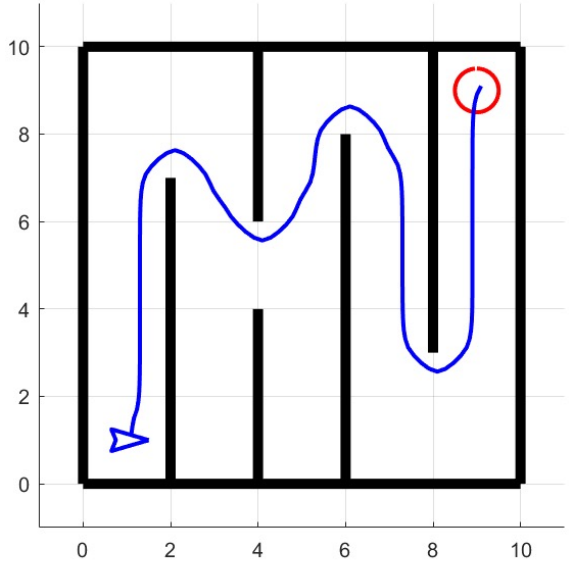
[1] TRIEBEL, Rudolph. The Particle Filter. In: *Machine Learning for Computer Vision* [online]. Technische Universität München, 2017 [cit. 2021-02-19]. Available at:https://vision.in.tum.de/_media/teaching/ss2017/ml4cv/variationalinference.pdf

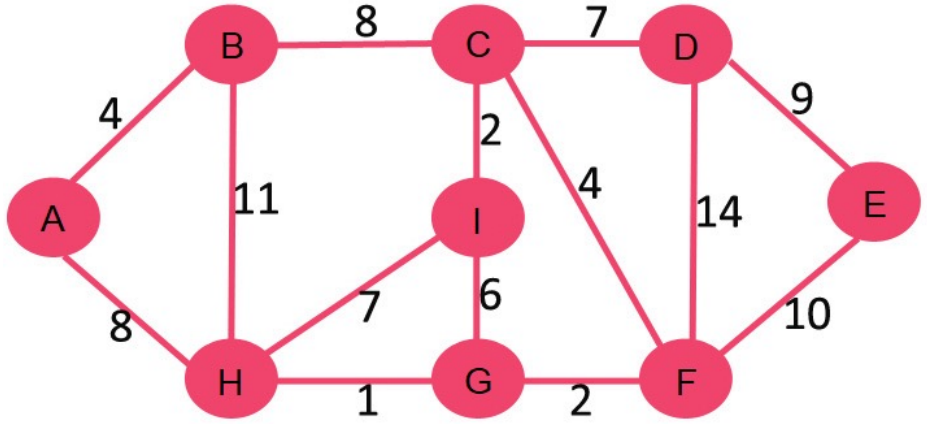
[2] STACHNISS, Cyrill. Short Introduction to Particle Filters and Monte Carlo Localization [online]. Uni Freiburg, 2013 [cit. 2021-02-18]. Available at: http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/pdf/slam10-particle-filter-4.pdf

- Configuration space
- Types of planners
- Workspace representation
- Graph-search algorithms: Dijsktra, A*, …
- Avoiding obstacles
- Path smoothing



[1]



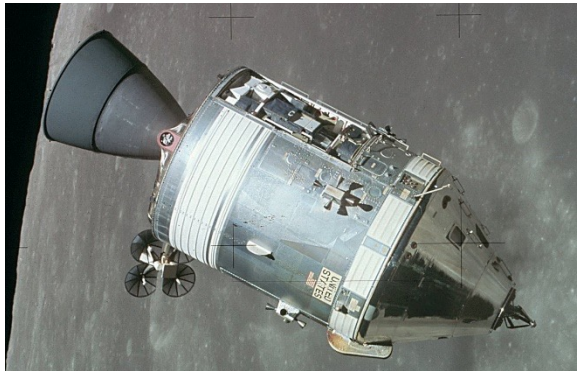


[2]

[1] DAHDOUH, Andrew. Graph-Based Path Planning: Dijkstra's Algorithm. In: *Reality Bytes* [online] 2017 [cit. 2021-02-26]. Available at: https://realitybytes.blog/2017/07/11/graph-based-path-planning-dijkstras-algorithm/
[2] Dijkstra's shortest path algorithm | Greedy Algo-7. In: *GeeksforGeeks* [online]. 2020 [cit. 2021-02-28]. Available at: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

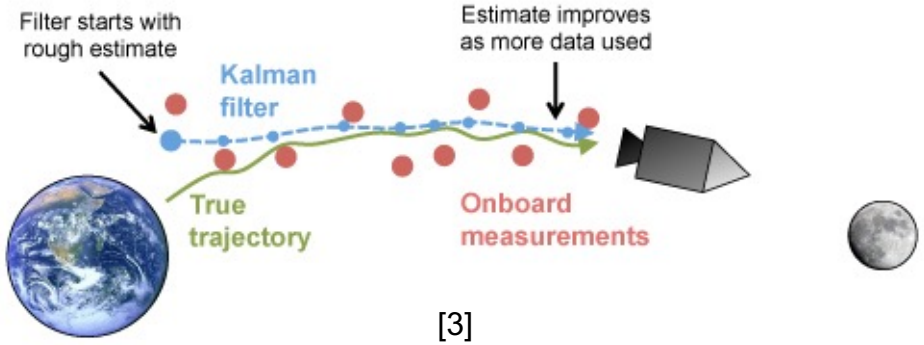
Week 6: Kalman Filter

- An algorithm for ***filtering*** and ***prediction*** in linear systems / ***estimating*** unknown variables.
- Suitable for fusing data from different sensors (different variables and sampling periods).



[2]

- Used for ***trajectory estimation*** for the ***Apollo*** program in the ~1960s – one of the very first applications of the Kalman filter [1].
  - Combination of acceleration data and star position observation.



[3]

[1] GREWAL, M. S. and ANDREWS, A. P., 2010. Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives]. *IEEE Control Systems Magazine*. June 2010. Vol. 30, no. 3, p. 69–78. DOI 10.1109/MCS.2010.936465.
[2] Apollo command and service module, 2021. Wikipedia [online]. [Accessed 4 March 2021]. Available from: https://en.wikipedia.org/wiki/Apollo_command_and_service_module
[3] Implementations of Kalman Filter From Aerospace to Industry. P2 SMTP LIPI [online]. 2018 [cit. 2021-01-18]. Available at: http://smtp.lipi.go.id/berita633-Implementations-of-Kalman-Filter-From-Aerospace-to-Industry.html

Week 7: SLAM Overview

Extended Kalman Filter SLAM

$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$

$$\bar{\sum}_t = G_t \sum_{t-1} G_t^T + R_t$$

$$K_t = \bar{\sum}_t H_t^T (H_t \bar{\sum}_t H_t^T + Q_t)^{-1}$$

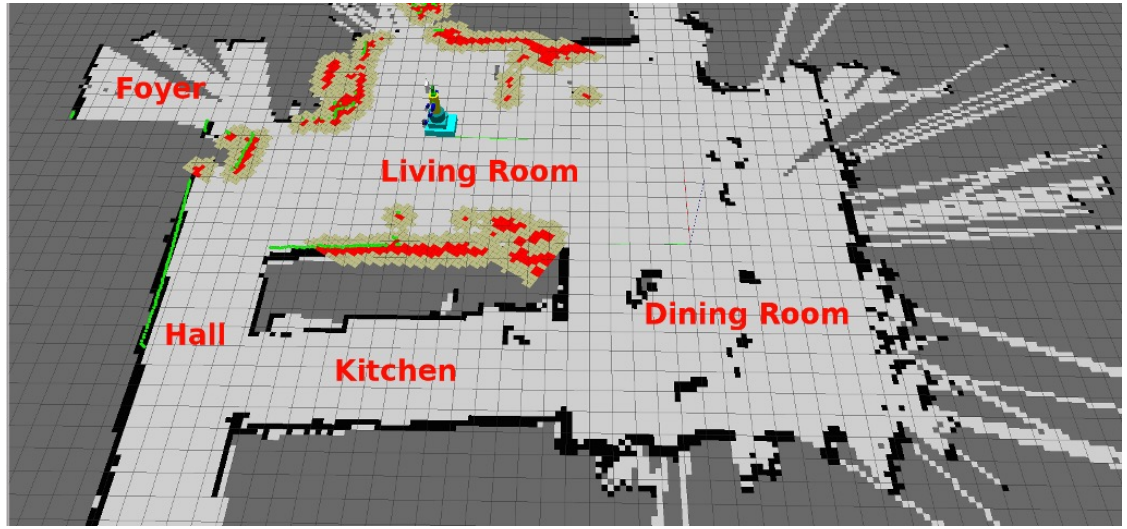
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$

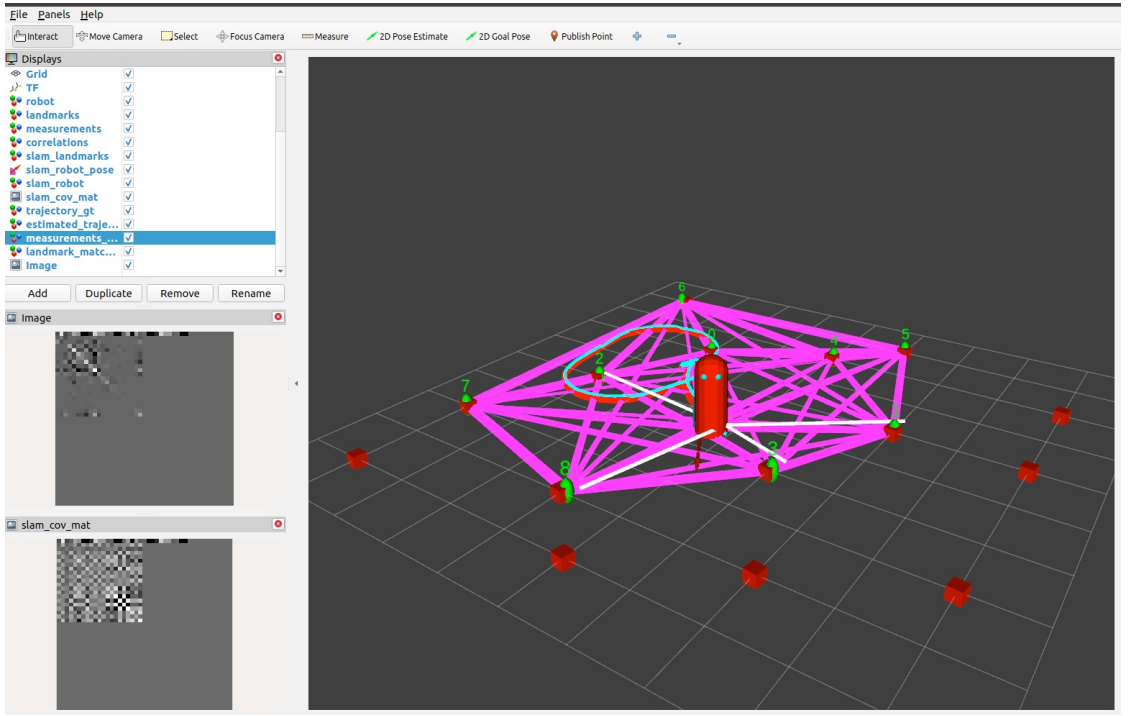
$$\sum_t = (I - K_t H_t \bar{\sum}_t)$$

Kalman-Based SLAM

Particles-Based SLAM

Graph-Based SLAM





https://sifrobot.com/wp-content/uploads/2020/03/Simultaneous-Localization-and-Mapping-SLAM.png

Topics Overview

Motion Models
- Velocity Model
- Rotation-Translation-Rotation Model

Kinematic Models
- Differential
- Omnidirectional
- Ackerman

Probability
- Gaussian distribution

Sensor Models
- Generic range sens.
- Laser sensor
- Range-Bearing model
- Camera model

Filters
- Bayes Filter
- Kalman Filter
- Extended Kalman Filter
- Particle Filter
- Information Filter

Path Planning
- Wavefront
- Dijkstra‘s Algorithm
- A*

SLAM
- EKF-SLAM
- Loop Closure

People

Course Supervisor

Ing. Lukáš Kopečný, Ph.D.

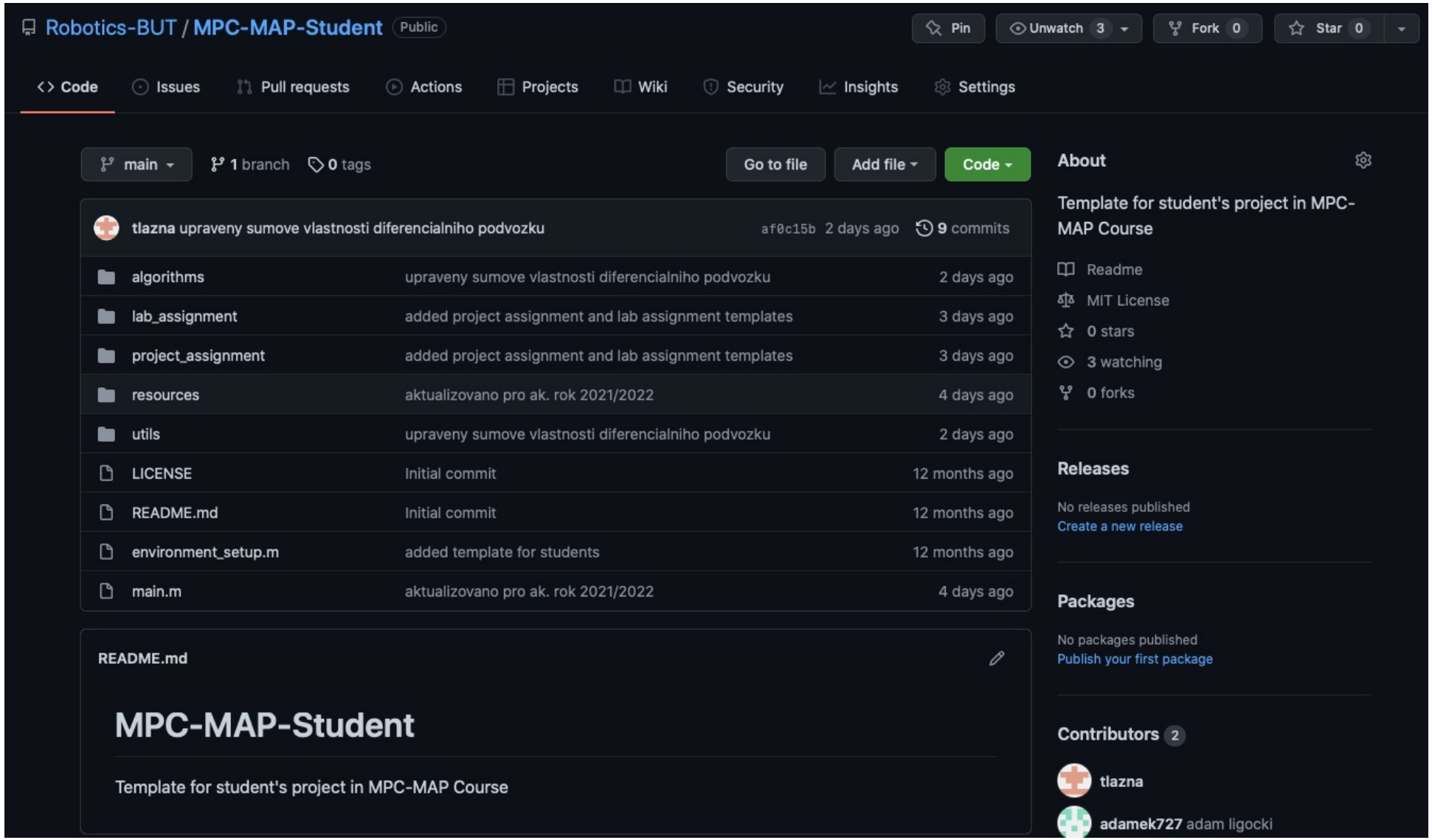
Lecturers

Ing. Adam Ligocki, Ph.D.
adam.ligocki@vutbr.cz
SE1.102

Ing. Tomáš Lázna
tomas.lazna@vutbr.cz
SE1.102

Ing. Petr Gábrlík, Ph.D
petr.gabrlik@vutbr.cz
SE1.112

Course Resources on GitHub



https://github.com/Robotics-BUT/MPC-MAP-Student

Prerequisites

Prerequisites: Vector Operations

Vector

$$\mathrm{V} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$\mathrm{W} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

Addition

$$\mathrm{V} + \mathrm{c} = \begin{bmatrix} v_1 + c \\ v_2 + c \\ v_3 + c \end{bmatrix}$$

$$\mathrm{V} + \mathrm{W} = \begin{bmatrix} v_1 + w_1 \\ v_2 + w_2 \\ v_3 + w_3 \end{bmatrix}$$

Multiplication

$$\mathrm{V} * \mathrm{c} = \begin{bmatrix} v_1 * c \\ v_2 * c \\ v_3 * c \end{bmatrix}$$

$$\mathrm{V} * W^T = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} * [w_1 \quad w_2 \quad w_3] =$$

$$= \begin{bmatrix} v_1 * w_1 & v_1 * w_2 & v_1 * w_3 \\ v_2 * w_1 & v_2 * w_2 & v_2 * w_3 \\ v_3 * w_1 & v_3 * w_2 & v_3 * w_3 \end{bmatrix}$$

$$V^T * W = [v_1 \quad v_2 \quad v_3] * \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} =$$

$$= [v_1 * w_1 + v_2 * w_2 + v_3 * w_3]$$

Hadamard Product (per element)

$$\mathrm{V} \odot \mathrm{W} = \begin{bmatrix} v_1 * w_1 \\ v_2 * w_2 \\ v_3 * w_3 \end{bmatrix}$$

For matrix (vector) multiplication:

$$\boldsymbol{[k\ x\ l] * [l\ x\ m] = [k\ x\ m]}$$

Prerequisites: Matrix Operations

Matrix

$$\mathrm{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{23} & m_{33} \end{bmatrix}$$

$$\mathrm{N} = \begin{bmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{bmatrix}$$

Addition

$$\mathrm{M} + \mathrm{c} = \begin{bmatrix} m_{11} + c & m_{12} + c & m_{13} + c \\ m_{21} + c & m_{22} + c & m_{23} + c \\ m_{31} + c & m_{23} + c & m_{33} + c \end{bmatrix}$$

$$\mathrm{M} + \mathrm{N} = \begin{bmatrix} m_{11} + n_{11} & m_{12} + n_{12} & m_{13} + n_{13} \\ m_{21} + n_{12} & m_{22} + n_{22} & m_{23} + n_{23} \\ m_{31} + n_{13} & m_{23} + n_{23} & m_{33} + n_{33} \end{bmatrix}$$

Multiplication

$$\mathrm{M} * \mathrm{c} = \begin{bmatrix} m_{11} * c & m_{12} * c & m_{13} * c \\ m_{21} * c & m_{22} * c & m_{23} * c \\ m_{31} * c & m_{23} * c & m_{33} * c \end{bmatrix}$$

Hadamard Product (per element)

$$\mathrm{M} \odot \mathrm{N} = \begin{bmatrix} m_{11} + n_{11} & m_{12} + n_{12} & m_{13} + n_{13} \\ m_{21} + n_{21} & m_{22} + n_{22} & m_{23} + n_{23} \\ m_{31} + n_{31} & m_{23} + n_{23} & m_{33} + n_{33} \end{bmatrix}$$

$$\mathrm{M}*\mathrm{N} = \begin{bmatrix} m_{11} * n_{11} + m_{12} * n_{21} + m_{13} * n_{31} & m_{11} * n_{12} + m_{12} * n_{22} + m_{13} * n_{32} & m_{11} * n_{13} + m_{12} * n_{23} + m_{13} * n_{33} \\ m_{21} * n_{11} + m_{22} * n_{21} + m_{23} * n_{31} & m_{21} * n_{12} + m_{22} * n_{22} + m_{23} * n_{32} & m_{21} * n_{13} + m_{22} * n_{23} + m_{23} * n_{33} \\ m_{31} * n_{11} + m_{32} * n_{21} + m_{33} * n_{31} & m_{31} * n_{12} + m_{32} * n_{22} + m_{33} * n_{32} & m_{31} * n_{13} + m_{32} * n_{23} + m_{33} * n_{33} \end{bmatrix}$$

Prerequisites: Matrix Operations

Matrix

$$\mathrm{V} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$\mathrm{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{23} & m_{33} \end{bmatrix}$$

Transposition

$$V^T = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$$

$$M^T = \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{32} & m_{33} \end{bmatrix}$$

Inversion

$$MM^{-1} = M^{-1}M = \mathrm{I}$$

$$M \mid I \quad \rightarrow \quad I \mid M^{-1}$$

$$M^{-1} = \frac{1}{\det(M)} * adj(M)$$

Root

$$M = PP$$

$$P = \sqrt{M}$$

see Cholesky Matrix Decomposition

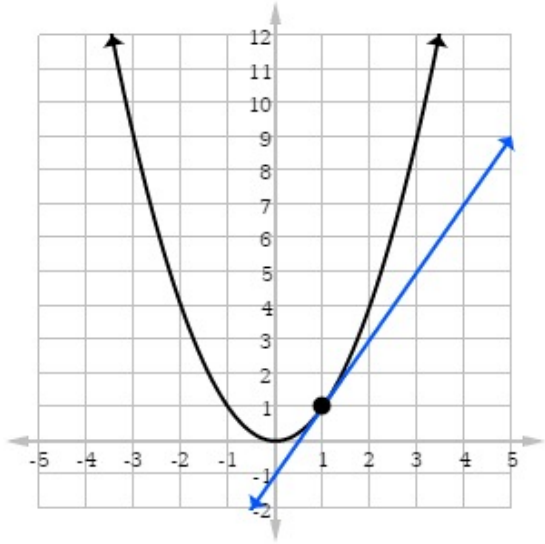
Matrix Congruence

$$\mathrm{C} = ABA^T$$

Prerequisites: Jacobi Matrix (Jacobian)

Matrix that is build from fector function‘s first order partial derivatives of several variables

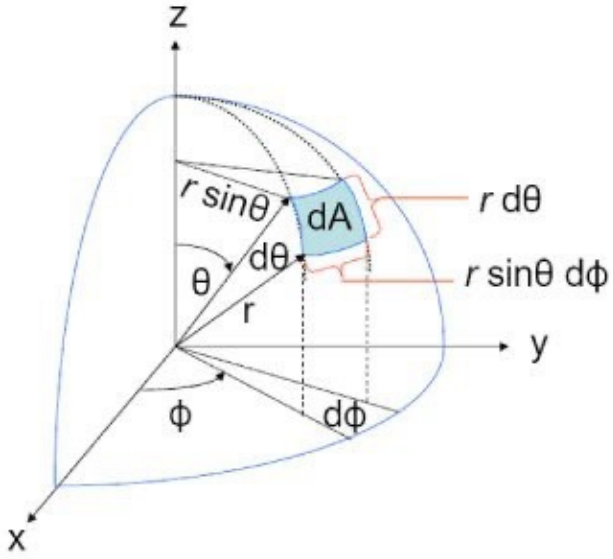
$$\mathbf{J} = \begin{bmatrix} \frac{df}{dx_1} & \cdots & \frac{df}{dx_n} \end{bmatrix} = \begin{bmatrix} \frac{df_1}{dx_1} & \cdots & \frac{df_1}{dx_n} \\ \cdots & \cdots & \cdots \\ \frac{df_m}{dx_1} & \cdots & \frac{df_m}{dx_n} \end{bmatrix}$$



Example:

$$\boldsymbol{X} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}; \qquad \boldsymbol{f}(\boldsymbol{X}) = \begin{bmatrix} x - \frac{v}{\omega}\sin(\theta) + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ y + \frac{v}{\omega}\cos(\theta) - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \theta + \omega\Delta t \end{bmatrix}; \qquad \boldsymbol{X_{t+1}} = \boldsymbol{f}(\boldsymbol{X_t});$$

$$\mathrm{J} = \frac{d}{d(x,y,\theta)} \begin{bmatrix} f_1(\boldsymbol{X}) \\ f_2(\boldsymbol{X}) \\ f_3(\boldsymbol{X}) \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\frac{v}{\omega}\cos(\theta) + \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ 0 & 1 & -\frac{v}{\omega}\sin(\theta) + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ 0 & 0 & 1 \end{bmatrix}$$



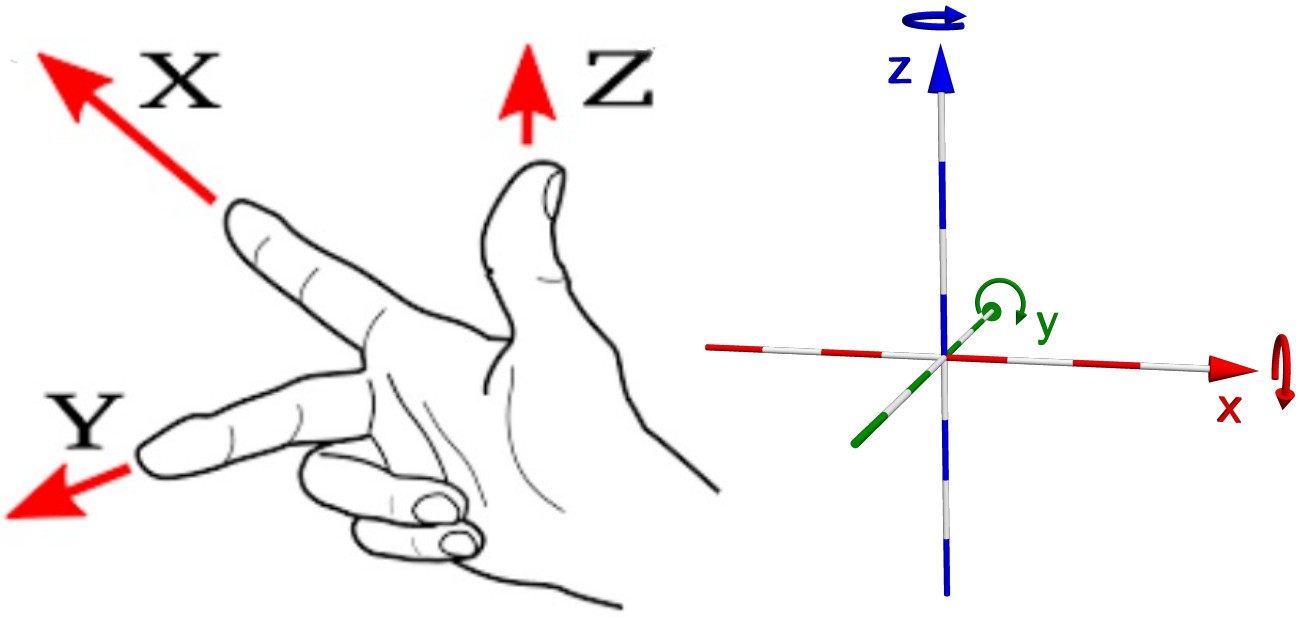
https://www.quora.com/What-is-the-Jacobian-how-does-it-work-and-what-is-an-intuitive-explanation-of-the-Jacobian-and-a-change-of-basis

Prerequisites: Coordinate system, Degrees of Freedom

Right-hand coordinate system

Right hand‘s thumb, forefinger, middle finger defines positive half-axis.

If you grab positive half axis by right hand, fingers show direction of positive angle.



Degrees of Freedom (DoF)

Maximum number of independen variables that describes system.

In robotics typically position and rotation.

for 2D: (x, y, θ) ... 3DoF

for 3D: (x, y, z, roll, pitch, yaw) ... 6DoF

Human hand ... 27 DoF

Prerequisites: State Vector, Covariance Matrix

In robotics we use vectors and matries to express fundamental structures that describes state of modeled systems (same as control theory).

State vector ... usually express robot‘s states (x pose, y pose, rotation) or position and rotation of objects around the robot.

$$\mu = \begin{bmatrix} x_{robot} \\ y_{robot} \\ \theta_{robot} \\ m_{1x} \\ m_{1y} \\ \dots \end{bmatrix}$$

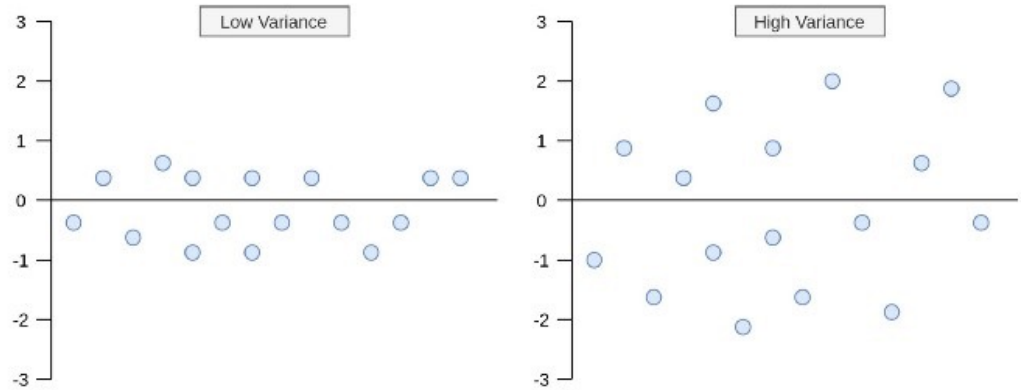
Covariance Matrix ... describes relations between variables in **state vector**.

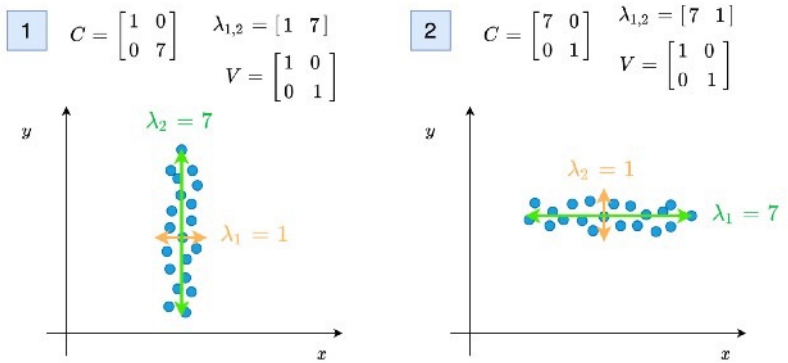
$$\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{xr,xr} & \Sigma_{xr,yr} & \dots & \Sigma_{xr,m1y} \\ \Sigma_{yr,xr} & \Sigma_{yr,yr} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \Sigma_{m1yr,xr} & \dots & \dots & \Sigma_{m1y,m1y} \end{bmatrix}$$

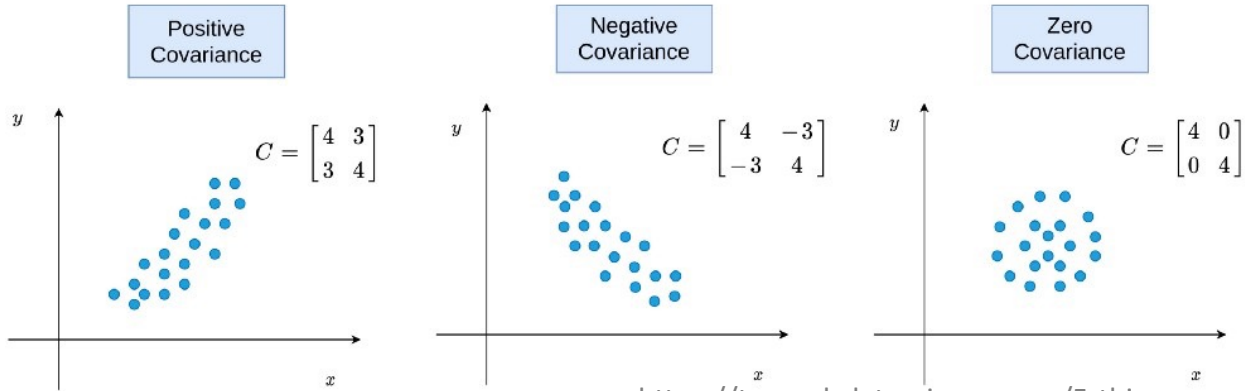
Prerequisites: Covariance Matrix

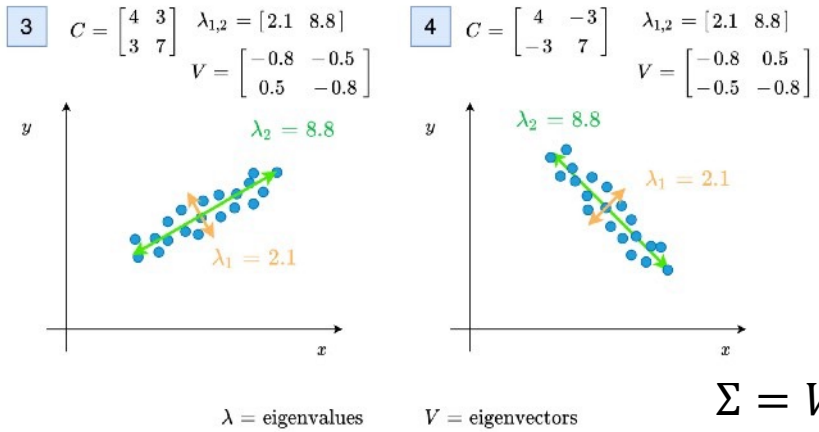
$$var(x) = \frac{\sum_{i=0}^{n}(x_i - \mu)^2}{N} \qquad \text{cov}(x, y) = \frac{\sum_{i=0}^{n}(x_i - \mu_x)(y_i - \mu_y)}{N}$$

$$\Sigma = \begin{matrix} x & y & z \\ \begin{bmatrix} var(x) & cov(x,y) & cov(x,z) \\ cov(y,x) & var(y) & cov(y,z) \\ cov(z,x) & cov(z,y) & var(z) \end{bmatrix} & \begin{matrix} x \\ y \\ z \end{matrix} \end{matrix}$$









λ … *eigenvalues*

V … *eigenvectors*

$$\Sigma = V \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \dots \end{bmatrix} V^{-1}$$

https://towardsdatascience.com/5-things-you-should-know-about-covariance-26b12a0516f1

Prerequisites: Linear Model

State Vector and **Control Vector**

$$\mu = \begin{bmatrix} x \\ y \\ v_x \\ v_y \\ \theta \end{bmatrix} \qquad u = \begin{bmatrix} a_x \\ a_y \\ \omega \end{bmatrix}$$

State Transition Matrix

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Control Matrix

$$B = \begin{bmatrix} 0.5\Delta t^2 & 0 & 0 \\ 0 & 0.5\Delta t^2 & 0 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix}$$

State Transition (Prediction)

$$\mu_t = A_t * \mu_{t-1} + B_t * u_t$$

Prerequisites: Transformation Matrix, Homogeneous Coordinates

Homogeneous **Euclidian**

$$\boldsymbol{x} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \boldsymbol{x_H} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \leftrightarrow \boldsymbol{x_E} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\boldsymbol{H_{3D}} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{yz} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{sR} & \boldsymbol{T} \\ \boldsymbol{0} & 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ c \end{bmatrix} \leftrightarrow \begin{bmatrix} u/c \\ v/c \end{bmatrix} \qquad \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \leftrightarrow \begin{bmatrix} inf \\ inf \end{bmatrix}$$

$\boldsymbol{s}$ ... scale
$\boldsymbol{R}$ ... rotation
$\boldsymbol{T}$ ... translation

Applying Transformation

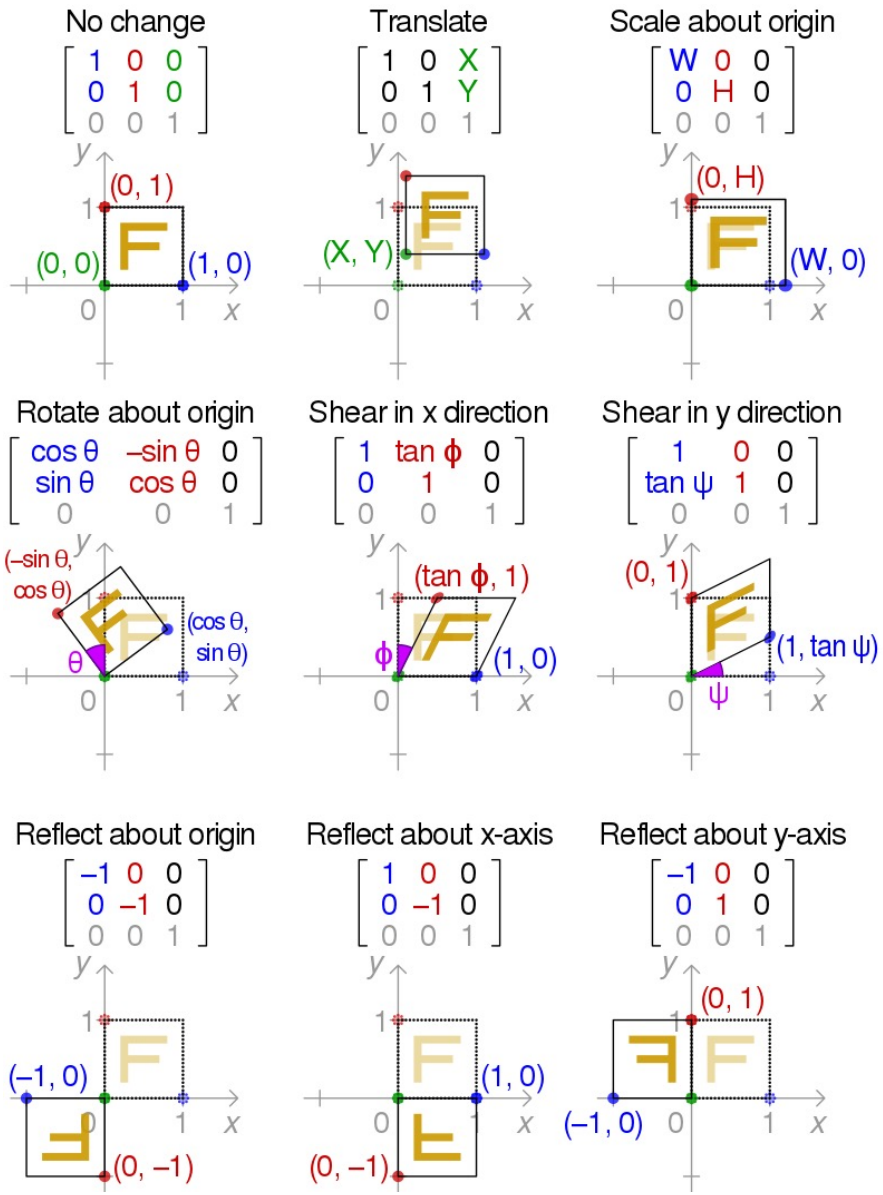
$$\mathbf{x}' = \boldsymbol{Hx}$$

$$\mathbf{x} = \boldsymbol{H^{-1}x'} \qquad \mathbf{x} = \boldsymbol{HH^{-1}x}$$

$$\mathbf{x}' = \boldsymbol{H_1 H_2 x} \neq \boldsymbol{H_2 H_1 x}$$

$$\begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda z' \\ \lambda \end{bmatrix} = \lambda \boldsymbol{H_{3D}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Note: usually λ=1



https://en.wikipedia.org/wiki/Transformation_matrix

Process Model (Motion model)

$$\mu_t = A_t\,\mu_{t-1} + B_t\,u_t + \varepsilon \qquad\qquad \mu_t = g(\mu_{t-1}, u_t) + \varepsilon$$

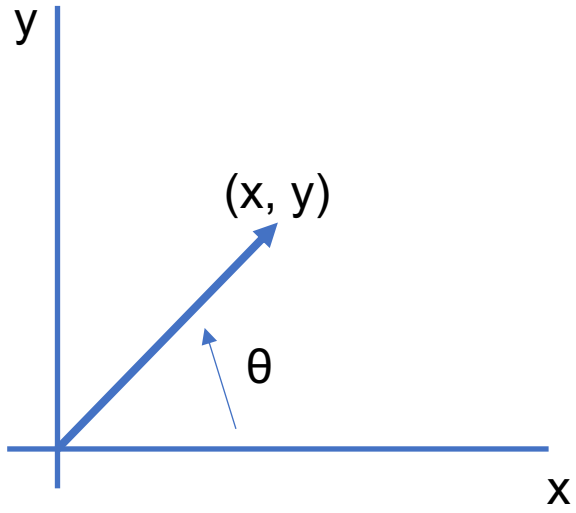
Measurement Model

$$z_t = H_t\,\mu_t + \delta \qquad\qquad z_t = h(\mu_t) + \delta$$

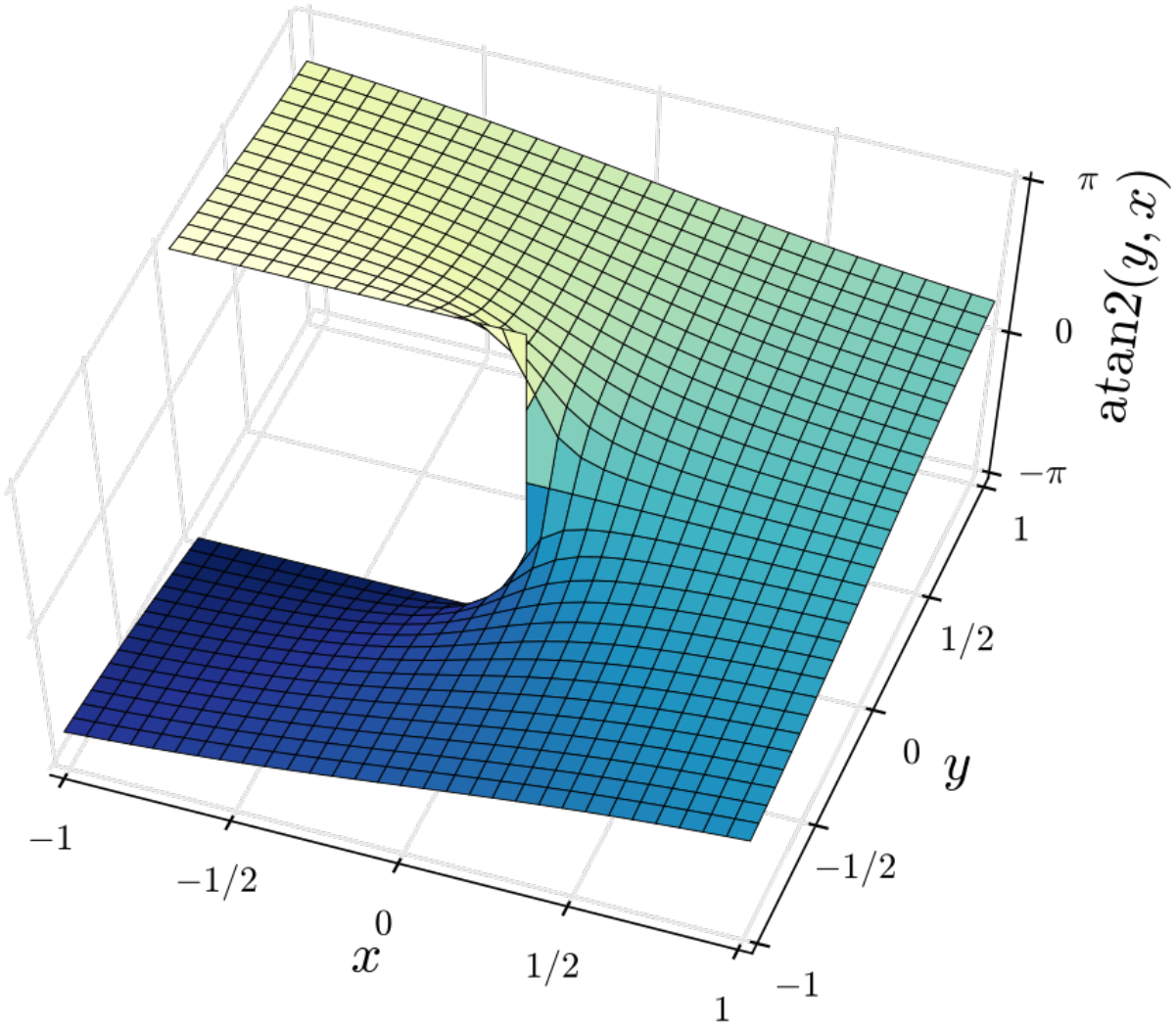
linear vs **non-linear**

Prerequisites: atan2(...)

Funciton resolves the angle of (x,y) vector.

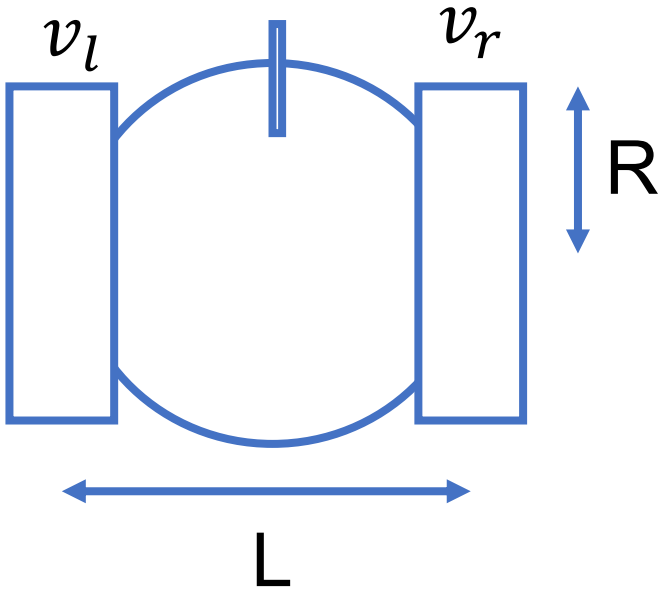


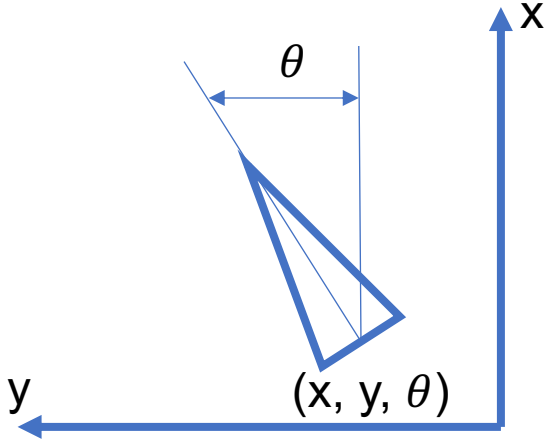
θ = atan2(y, x)



https://en.wikipedia.org/wiki/Atan2

Prerequisites: Differential Chassis Kinematic Model





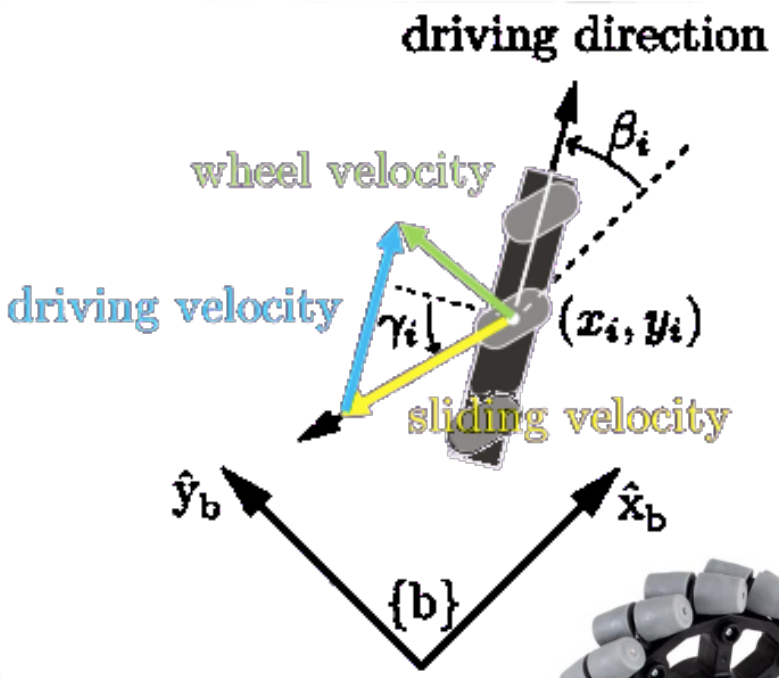
$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\frac{R}{2}(v_l + v_r) * \cos(\theta)\\ \frac{R}{2}(v_l + v_r) * sin(\theta)\\ \frac{R}{L}(v_r - v_l)\end{bmatrix}$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}v * \cos(\theta)\\ v * sin(\theta)\\ \omega\end{bmatrix}$$

$$\begin{bmatrix}v\\ \omega\end{bmatrix} = \begin{bmatrix}\frac{R}{2}(v_l + v_r)\\ \frac{R}{L}(v_r - v_l)\end{bmatrix}$$

$$\begin{bmatrix}v_r\\ v_l\end{bmatrix} = \begin{bmatrix}\frac{2v + \omega L}{2R}\\ \frac{2v - \omega L}{2R}\end{bmatrix}$$

Prerequisites: Omnidirectional Chassis Kinematic Model



$$u_i = \frac{1}{r}[1 \quad \tan(\gamma_i)] \begin{bmatrix} \cos(\beta_i) & \sin(\beta_i) \\ -sin(\beta_i) & cos(\beta_i) \end{bmatrix} \begin{bmatrix} -y_i & 1 & 0 \\ x_i & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_b \\ v_{xb} \\ v_{yb} \end{bmatrix}$$

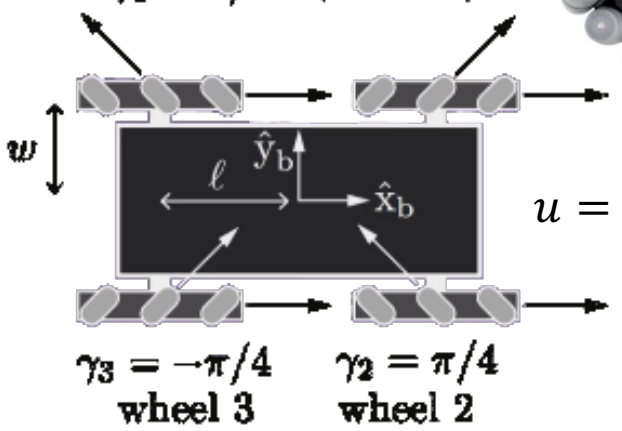
Motor speed

components in driving dir.

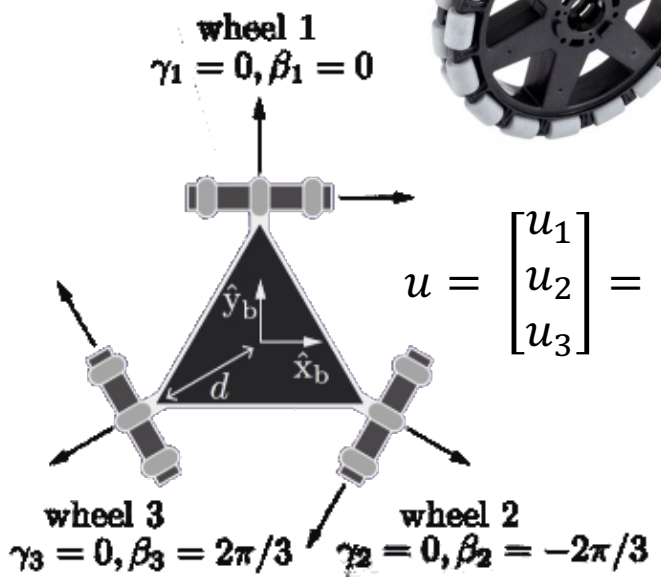
to lin. vel. in wheel frame

to wheel's velocities in chassis frame

chassis velocities



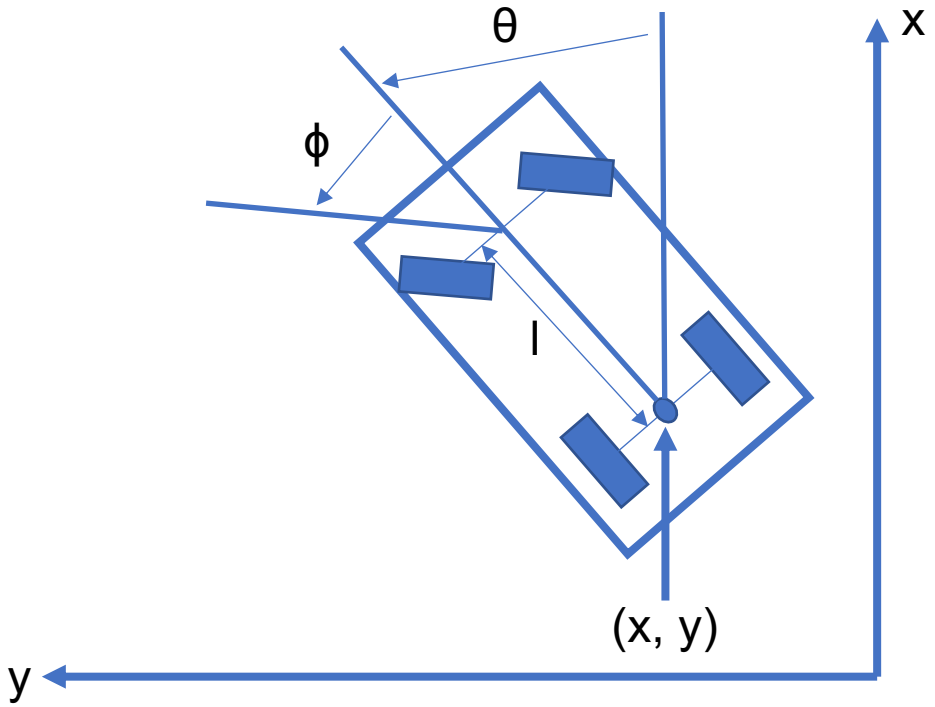
$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -l-w \\ 1 & 1 & l+w \\ 1 & -1 & l+w \\ 1 & 1 & -l-w \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$



$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & 0 & -d \\ -\frac{1}{2} & -\sin(\frac{\pi}{3}) & -d \\ -\frac{1}{2} & \sin(\frac{\pi}{3}) & -d \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

http://hades.mech.northwestern.edu/images/7/7f/MR.pdf; Chapter 13

Prerequisites: Ackerman Kinematic Model

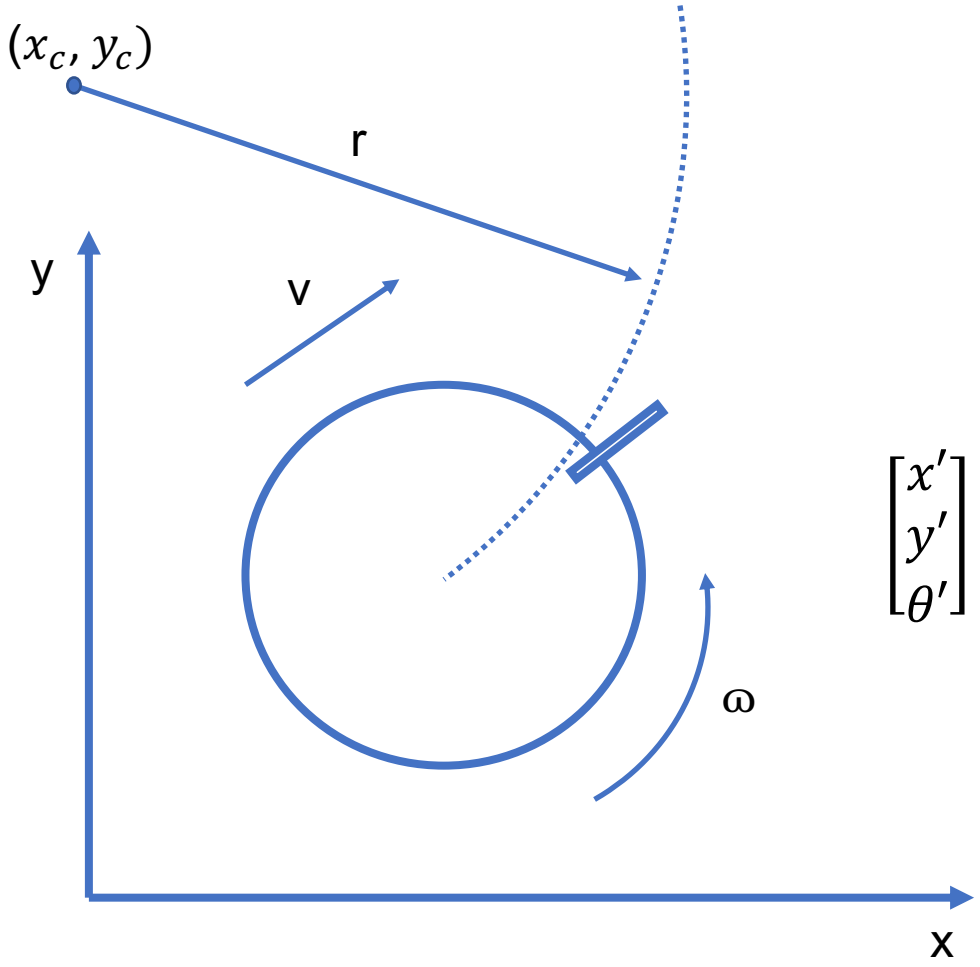


Control inputs: $\begin{bmatrix} v_{lin\ speed} \\ \omega_{stearing\ speed} \end{bmatrix}$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} v_{lin\ speed} * \cos(\theta) \\ v_{lin\ speed} * \sin(\theta) \\ \frac{v_{lin\ speed}}{l} \tan(\phi) \\ \omega_{stearing\ speed} \end{bmatrix}$$

! Simplified model ! – assuming, same orientation of both front wheels

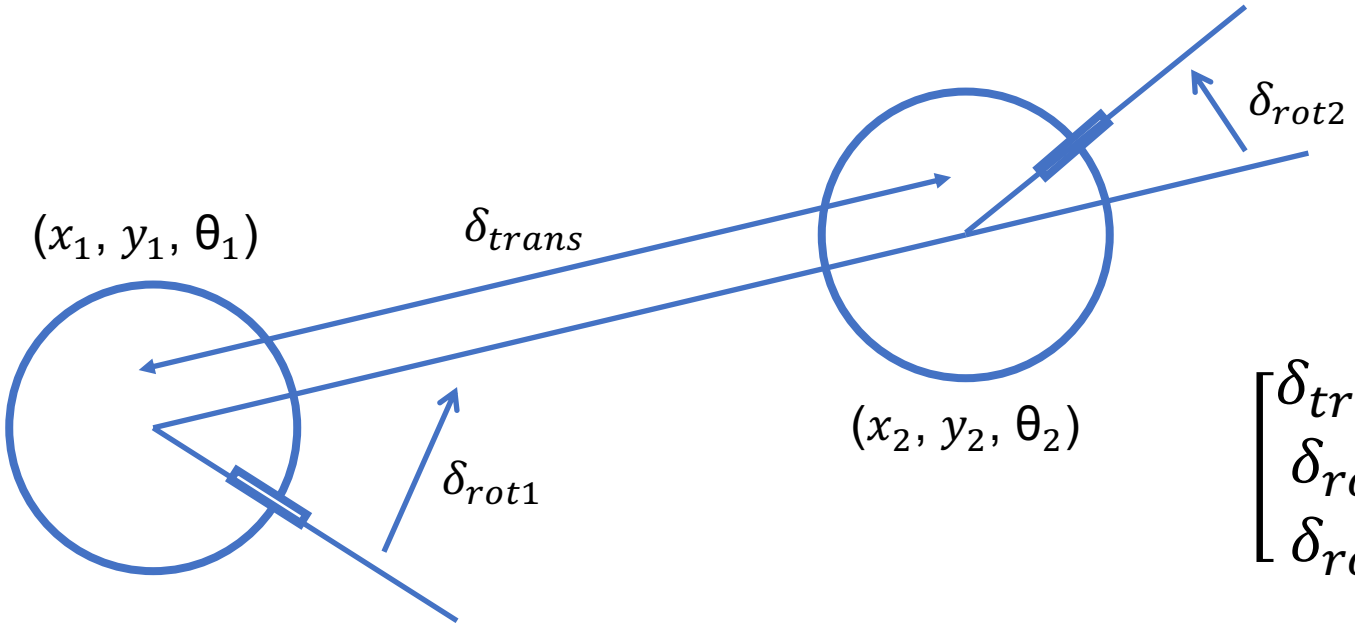
Prerequisites: Velocity Model (Arc Trajectory)



$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x_c + r\sin(\theta + \omega\Delta t) \\ y_c - r\cos(\theta + \omega\Delta t) \\ \theta + \omega\Delta t \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{v}{\omega}\sin(\theta) + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos(\theta) - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{bmatrix}$$

Robot moves on arc trajectory (constant v and ω; for ω=0 reduced to eq. mentioned in differential chassis model)

Prerequisites: Rotaiton-Translation-Rotation Model



$$\begin{bmatrix} \delta_{trans} \\ \delta_{rot1} \\ \delta_{rot2} \end{bmatrix} = \begin{bmatrix} \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2} \\ atan2(y_2 - y_1, x_2 - x_1) - \theta_1 \\ \theta_2 - \theta_1 - \delta_{rot1} \end{bmatrix}$$

Robot moves from (x_1, y_1, θ_1) to (x_2, y_2, θ_2) using three separated control motions, rotation, translation and rotation.

Adam Ligocki

adam.ligocki@vutbr.cz

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

Robotics and AI
Research Group