



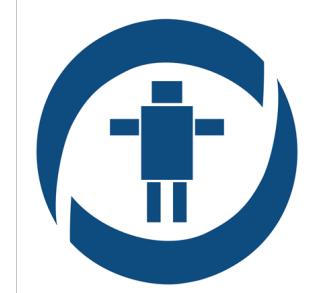
1 – Introduction to Course

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

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

Ing. Adam Ligocki, Ph.D.

Brno University of Technology
2021





Robotics and AI

Ing. Adam Ligocki, Ph.D.

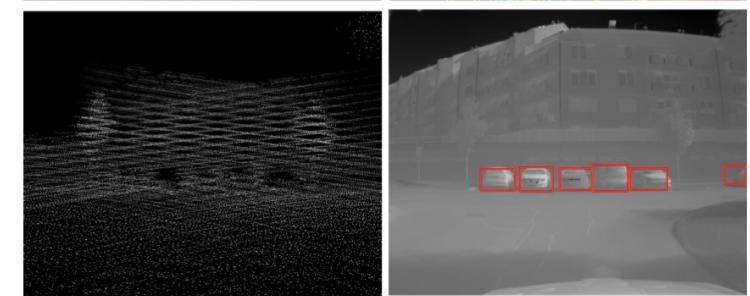
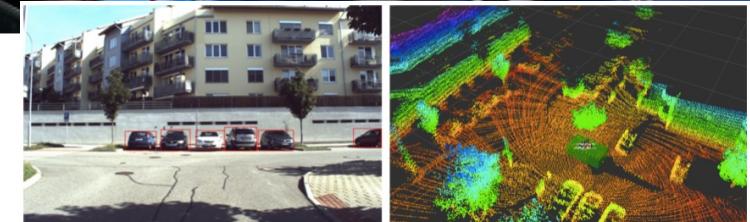
Position: Research Staff

Research: Data Fusion

Room: SE1.102

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

Profile

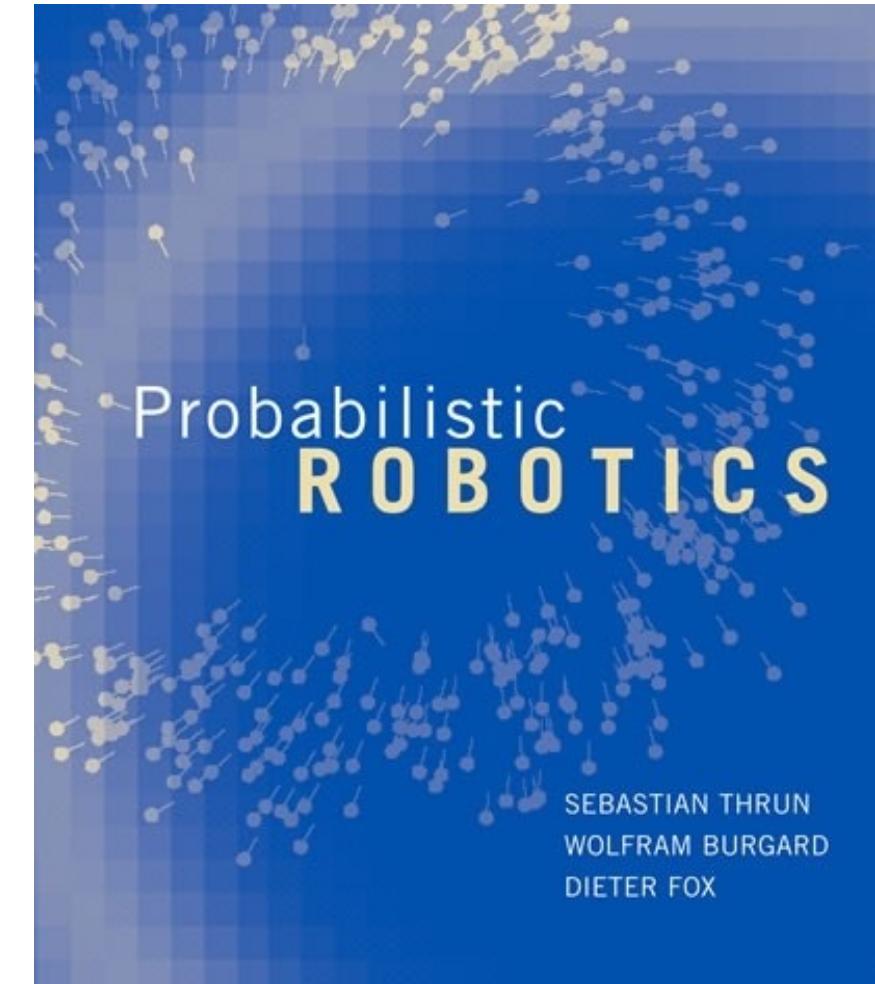


Probabilistic Robotics

[Probabilistic Robotics,
Intelligent Robotics and Autonomous Agents series,
Sebastian Thrun, Wolfram Burgard, Dieter 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



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.gabrlík@vutbr.cz

SE1.112



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 consultations with lector.

Study sources

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



Course Structure

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



Assignments

Assignment in weeks 2 to 6 (5 in total)

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

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

Deadline is by the next Sunday 23:59

During the week you can consult report with lector

Assignments are on GitHub repo

MPC-MAP

Week 2 - Sensor Modeling

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

Create a single A4 report that will describe your approach to the exercise (3-6 sentences for each task and picture, if 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 $\sigma=0.001m$ $c=0.001m$, $d=200m$ 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 $(\mu=\text{obstacle_distance}, \sigma=\text{sensor_std_dev})$.
- The "get_positive_sensor_model"; $p(z | m_o)$; function will return gaussian-based obstacle probability distribution function for the entire map space.
- Complementary implement also the "get_neg_sensor_model"; $p(z | m_e)$.

**BRNO FACULTY OF ELECTRICAL
UNIVERSITY ENGINEERING
OF TECHNOLOGY 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.

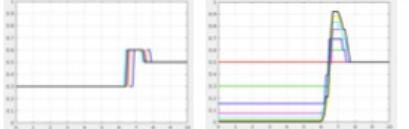
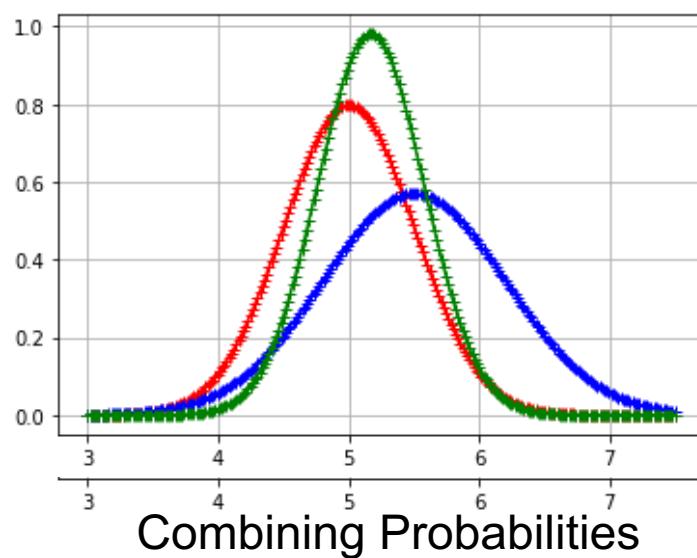
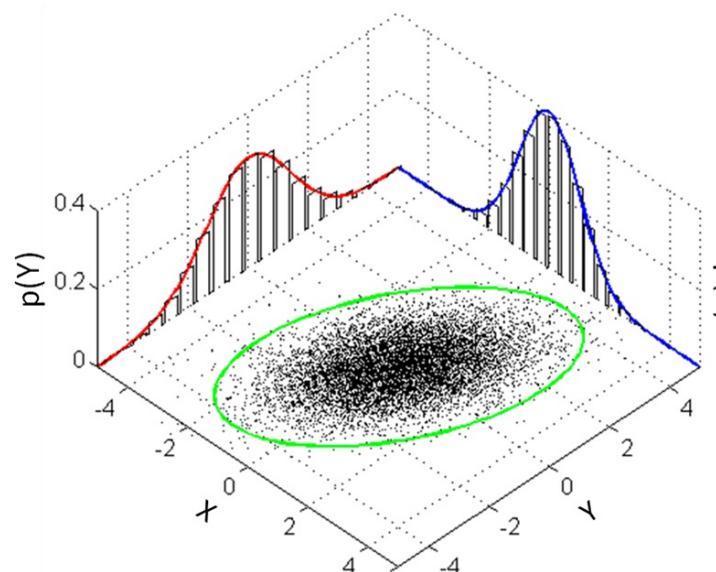
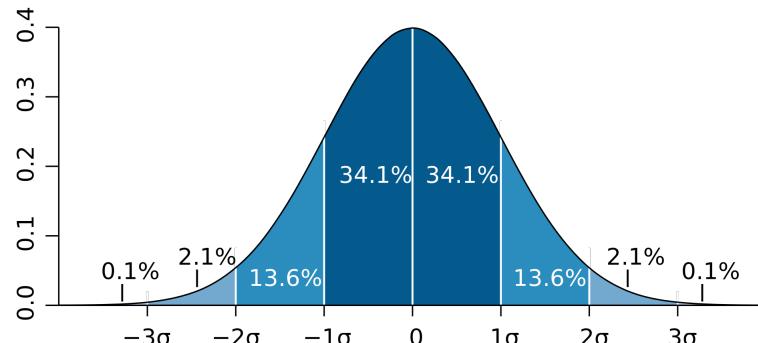


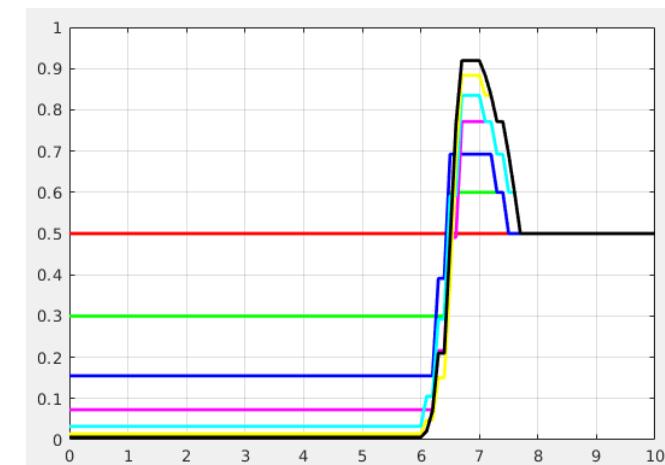
Figure 2 - Lorem ipsum

Assignment (web)

Report (placeholder)



Combining Probabilities



Sensor Modeling

Bayes Probability

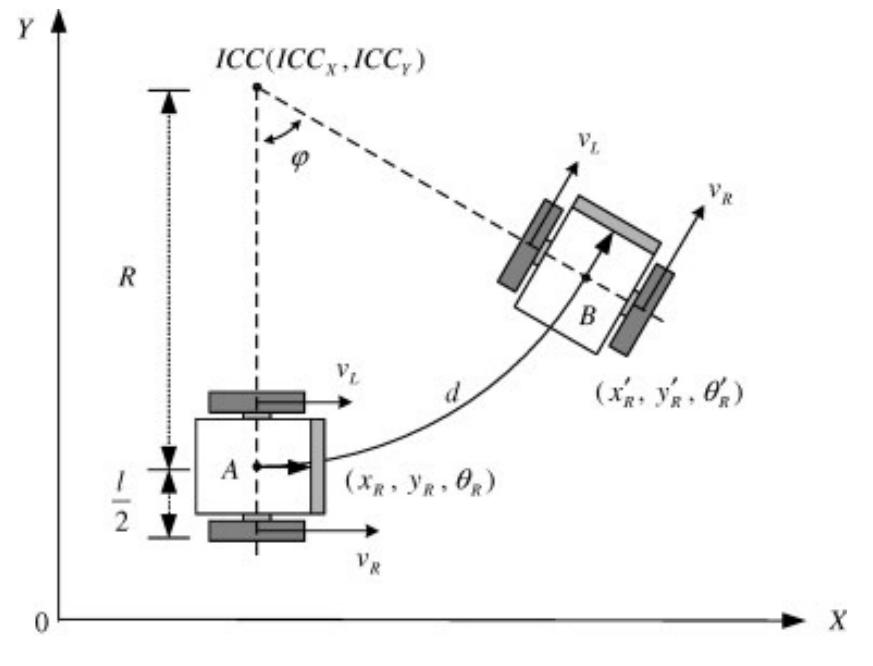
$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

Bayes Filter

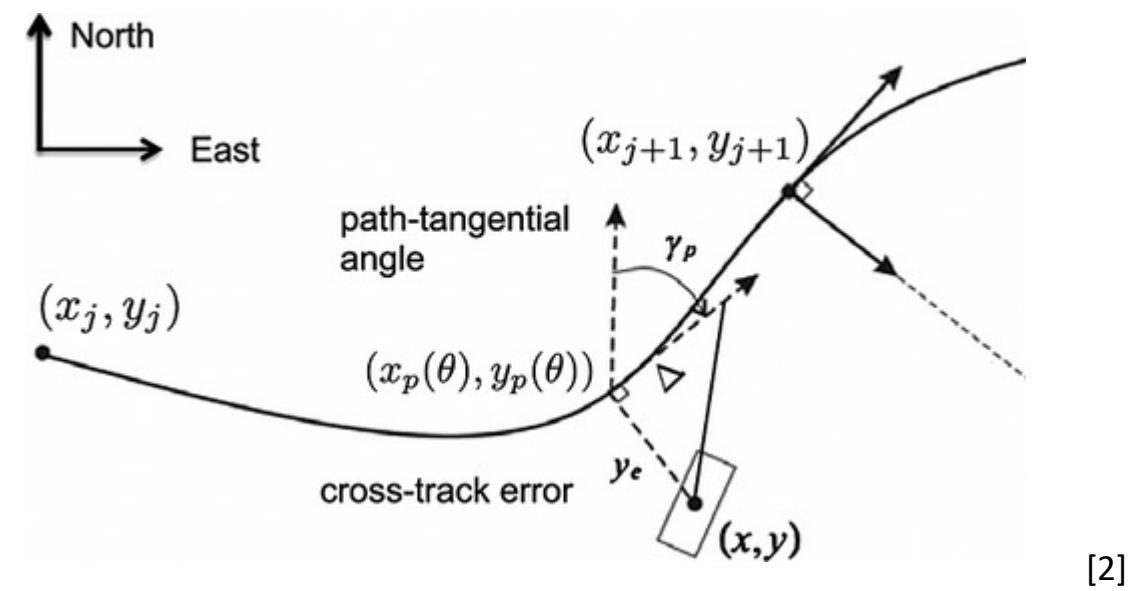
$$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

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

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

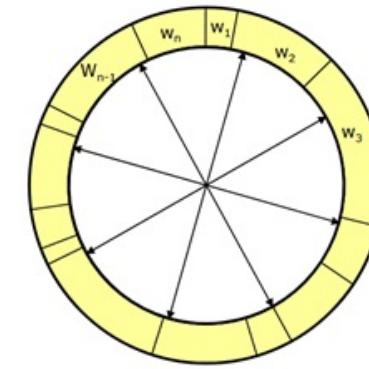
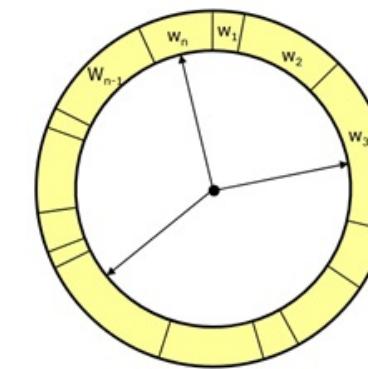
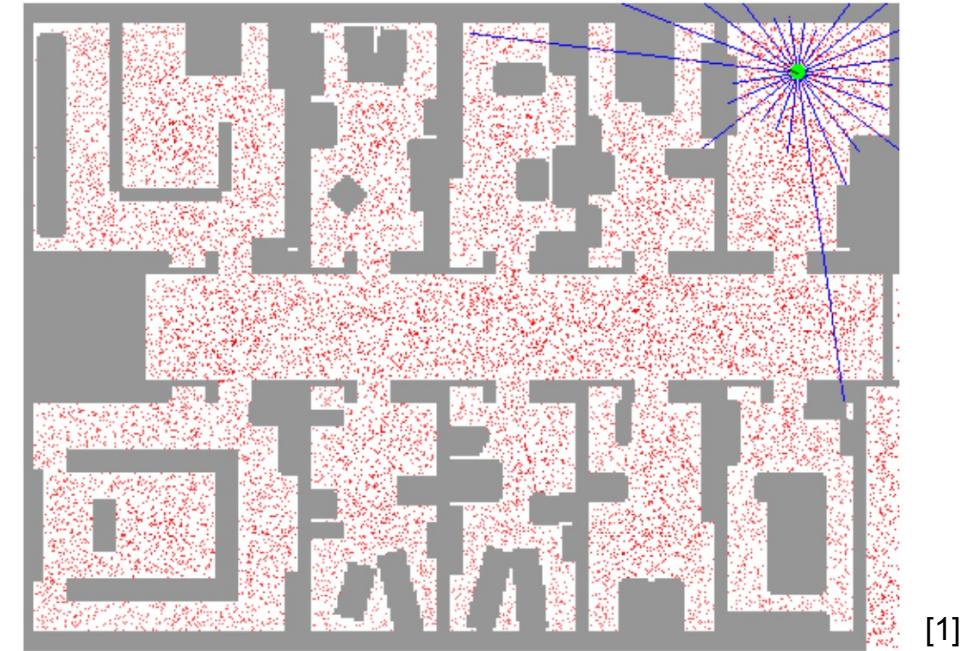
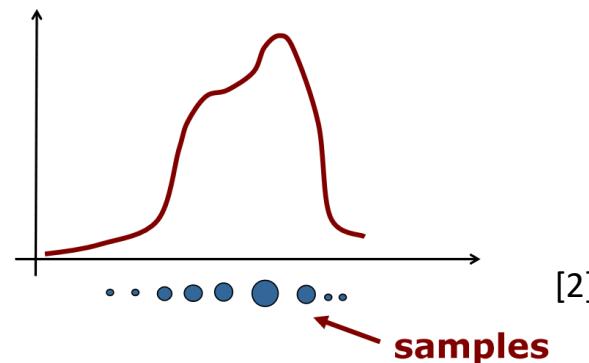
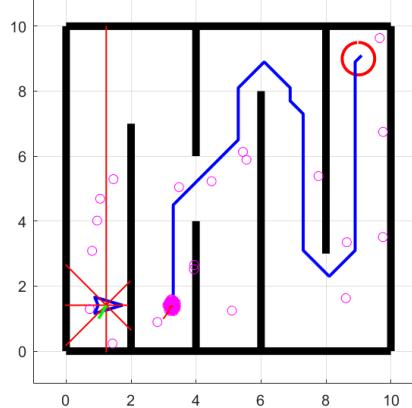


[1]



[2]

- 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] TRIEBEL, Rudolph. The Particle Filter. In: *Machine Learning for Computer Vision* [online]. Technische Universität München, 2017 [cit. 2021-02-19].

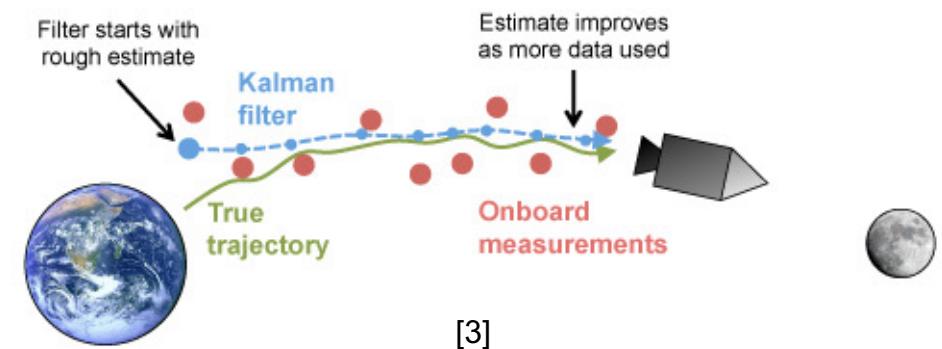
Available at: https://vision.in.tum.de/_media/teaching/ss2017/ml4cv/variationalinference.pdf

[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>

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



[2]

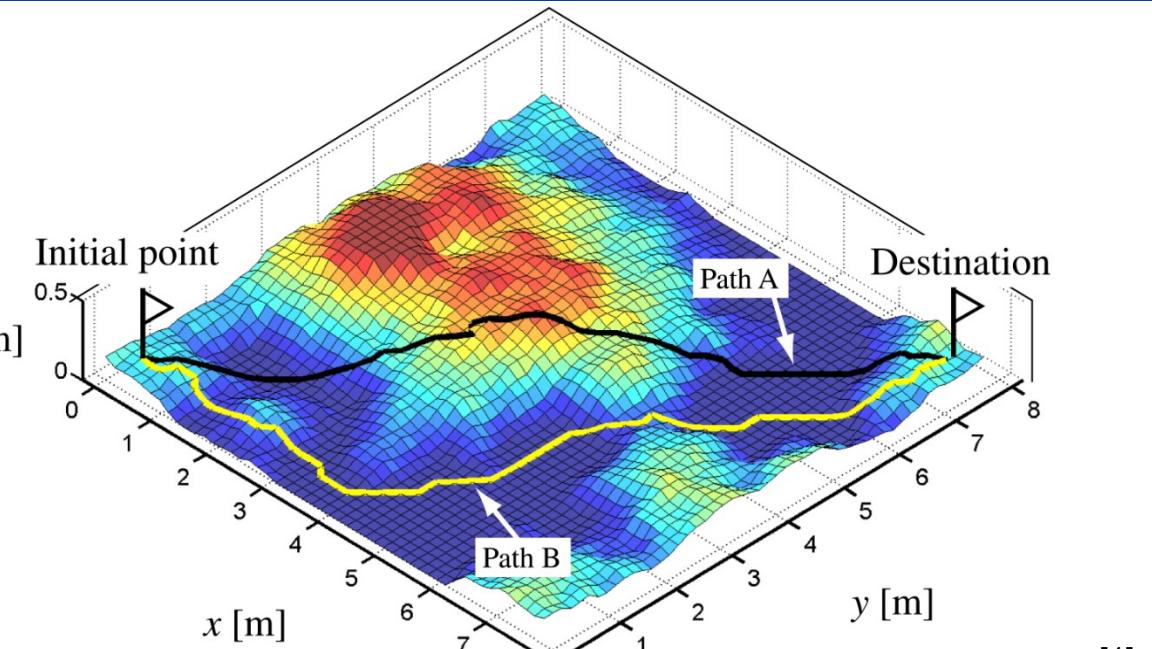
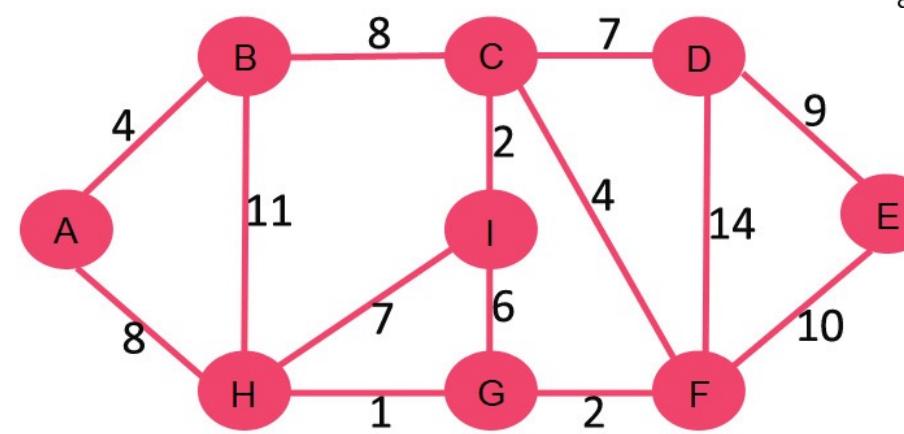
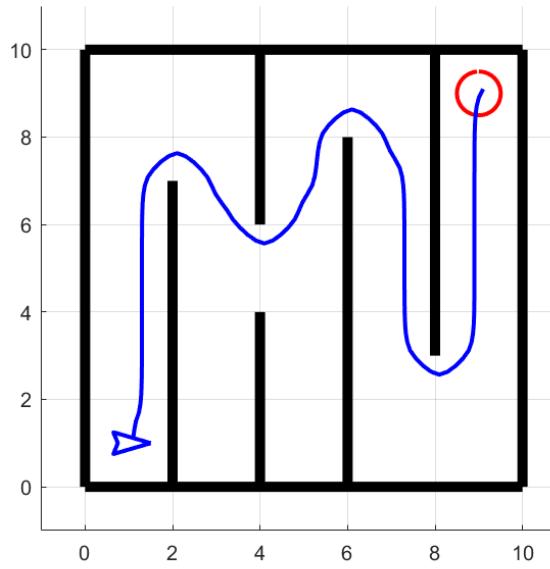


[1] GREWAL, M. S. and ANDREWS, A. P., 2010. Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives]. *IEEE Control Systems Magazine*. June 2010. Vol. 30, no. 3, p. 69–78.
DOI [10.1109/MCS.2010.936465](https://doi.org/10.1109/MCS.2010.936465).

[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>

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



[1]

[2]

[1] DAHDOUH, Andrew. Graph-Based Path Planning: Dijkstras 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] Dijkstras 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/>

Extended Kalman Filter SLAM

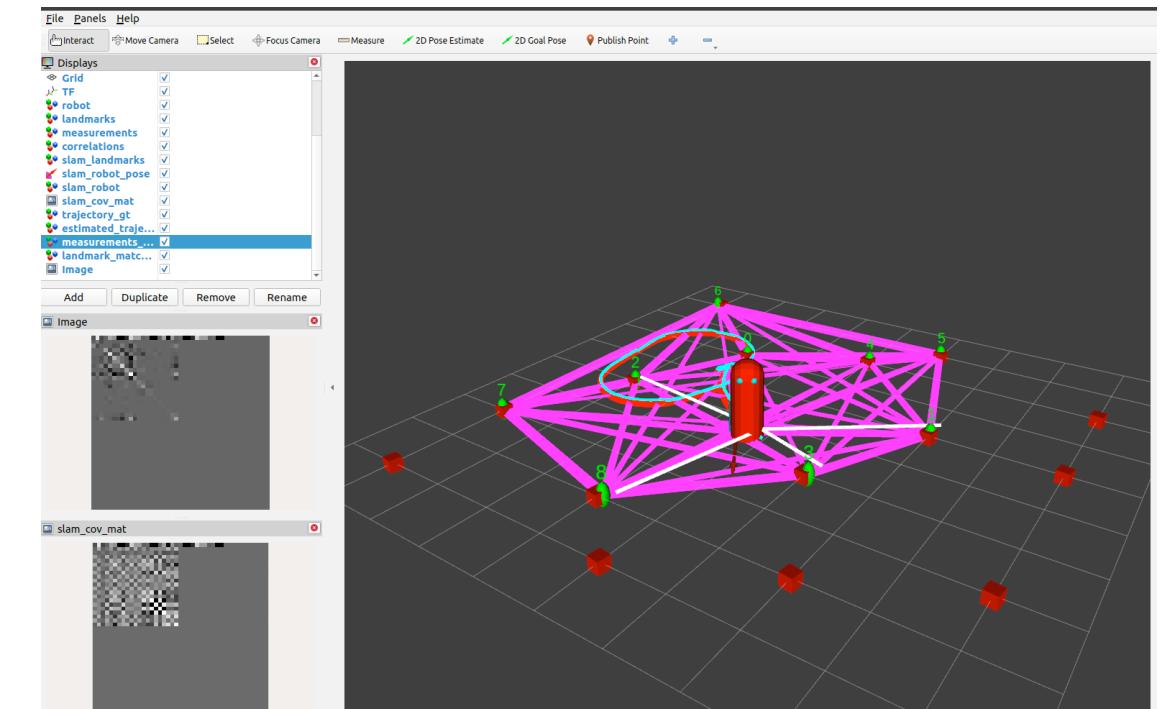
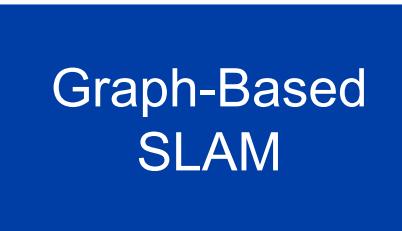
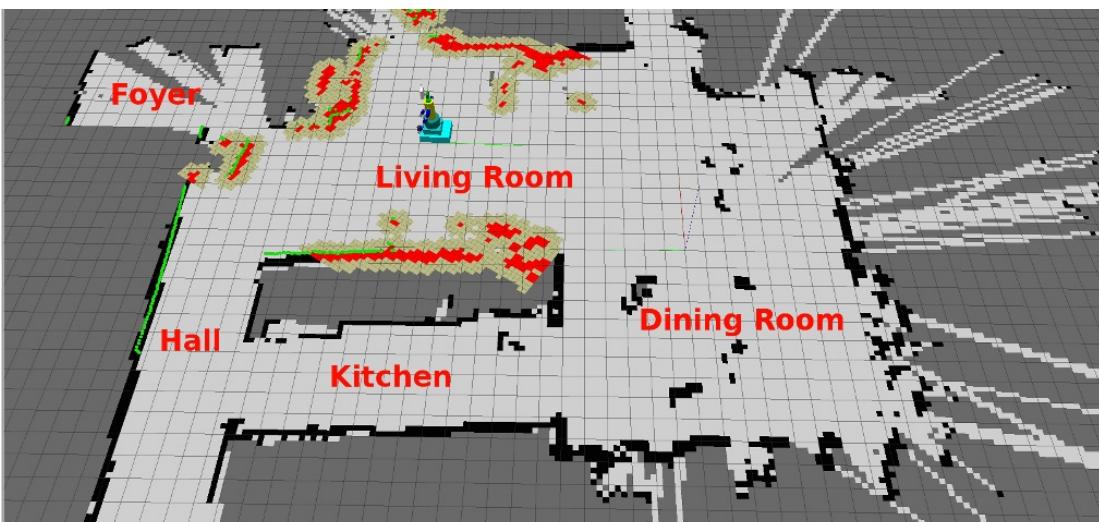
$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$$

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

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

$$\Sigma_t = (I - K_t H_t \bar{\Sigma}_t)$$





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



Course Resources on GitHub

 Robotics-BUT / MPC-MAP-Student Public

Edit Pins Unwatch 7 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main ▾ 3 branches 0 tags Go to file Add file <> Code

File/Folder	Description	Time
algorithms	updated student simulator (gnss-denied zones were added)	3 weeks ago
lab_assignment	added materials for 5th week	10 months ago
lecture_slideshows	added slam slideshow	10 months ago
project_assignment	fixed project file	last year
resources	aktualizovano pro ak. rok 2021/2022	last year
utils	updated student simulator (gnss-denied zones were added)	3 weeks ago
.gitignore	fixed assignment 1	last year
LICENSE	Initial commit	2 years ago
README.md	Initial commit	2 years ago
environment_setup.m	added template for students	2 years ago
main.m	updated student simulator (gnss-denied zones were added)	3 weeks ago

About Course Web Pages robotics-but.github.io/MPC-MAP-Stu...

Readme MIT license 0 stars 7 watching 0 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package

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



Semester Project Overview

Prepared template of the Matlab project with the simulation program.

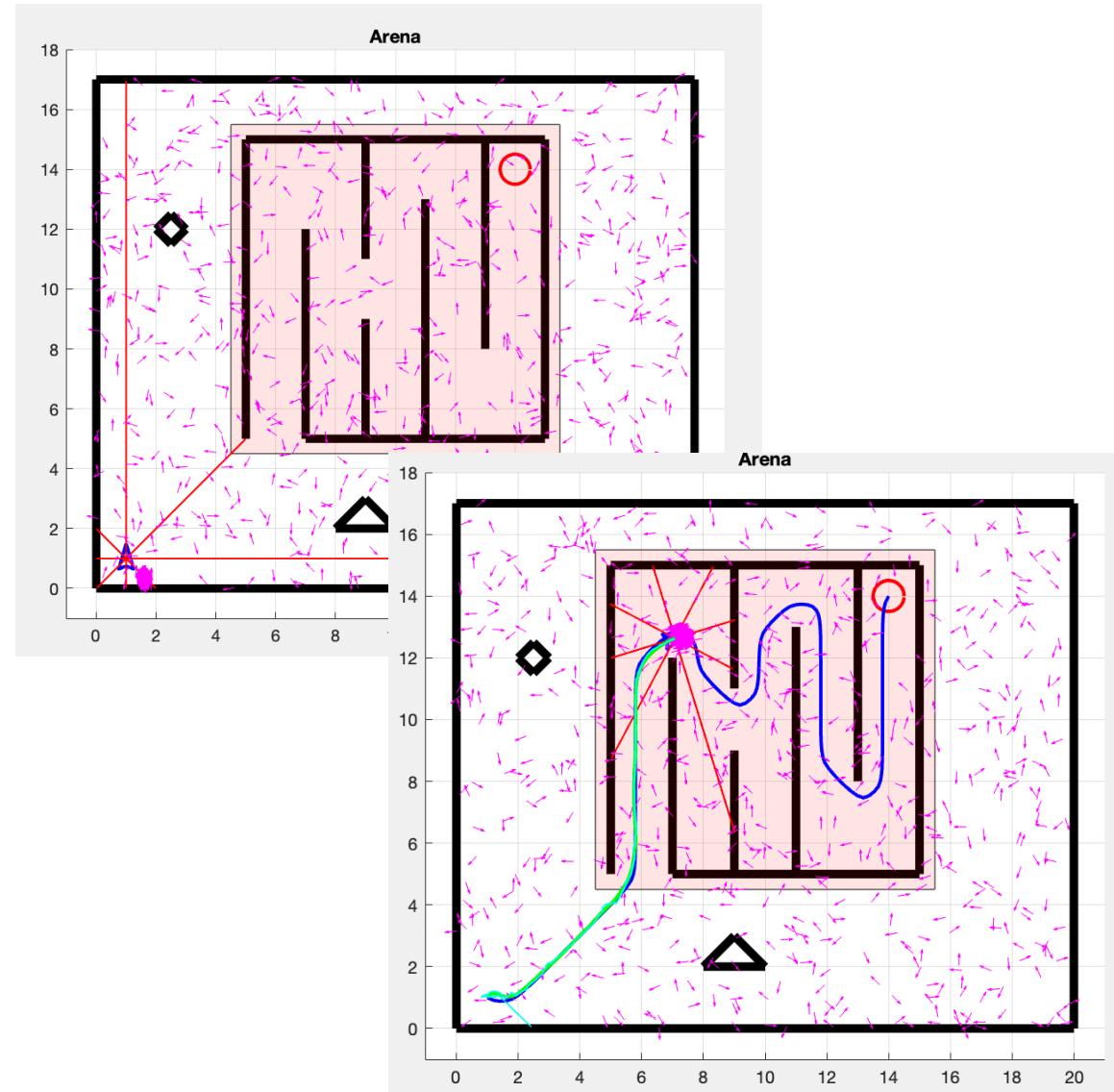
Simulator backend is compiled into .p files

Available:

- 3 DoF (x, y, tau) robot
- map of the arena
- 8 virtual ranging sensors,
- GNSS-like sensor (excluded area)
- start and goal position

Your tasks:

- localize robot in the arena
- plan path from current pose to the goal
- follow planned path





All project rules and assignment details are available on:

<https://robotics-but.github.io/MPC-MAP-Student/>

- In weeks 2 to 6, there is always 1 assignment per week to solve
- Deadline for each assignment is by next **Sunday 23:59**
- You can consult your solution with lector (preferred via email) during the week
- Solution will be written on **single A4**; template from official GitHub
- Each solution is rated by 0-10 points.
- Final project deadline: **8th week** of the semester, **Sunday 23:59**
- Final project submission **MUST** be executable without any errors (**penalty up to -20p**)
- Matlab toolboxes are **forbidden**
- Max 50p for project



Prerequisites / Cheat Sheet



Prerequisites / Cheat Sheet: Vector Operations

Vector

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

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

Addition

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

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

Multiplication

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

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

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

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

Hadamard Product (per element)

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

For matrix (vector) multiplication:

$$[k \times l] * [l \times m] = [k \times m]$$



Prerequisites / Cheat Sheet: Matrix Operations

Matrix

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

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

Addition

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

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

Multiplication

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

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

Hadamard Product (per element)

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



Prerequisites / Cheat Sheet: Matrix Operations

Matrix

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

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

Transposition

$$V^T = [v_1 \quad v_2 \quad v_3]$$

$$M^T = \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{bmatrix}$$

Inversion

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

$$M | I \rightarrow I | M^{-1}$$

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

Root

$$M = PP$$

$$P = \sqrt{M}$$

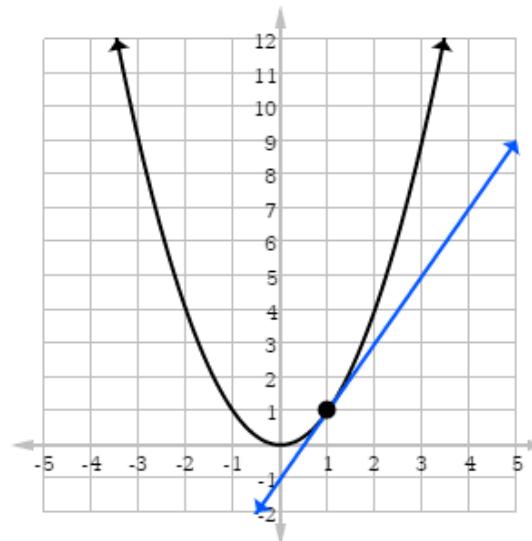
see Cholesky Matrix Decomposition

Matrix Congruence

$$C = ABA^T$$

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

$$\mathbf{J} = \begin{bmatrix} \frac{df}{dx_1} & \dots & \frac{df}{dx_n} \\ \vdots & \ddots & \vdots \\ \frac{df_m}{dx_1} & \dots & \frac{df_m}{dx_n} \end{bmatrix}$$

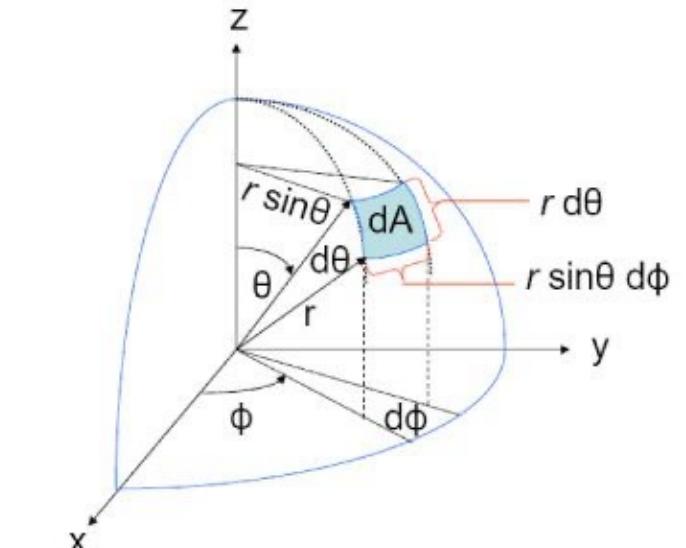


Example:

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}; \quad \mathbf{f}(\mathbf{X}) = \begin{bmatrix} x - \frac{v}{\omega} \sin(\theta) + \frac{v}{\omega} \sin(\theta + \omega \Delta t) \\ y + \frac{v}{\omega} \cos(\theta) - \frac{v}{\omega} \cos(\theta + \omega \Delta t) \\ \theta + \omega \Delta t \end{bmatrix}; \quad \mathbf{X}_{t+1} = \mathbf{f}(\mathbf{X}_t);$$

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

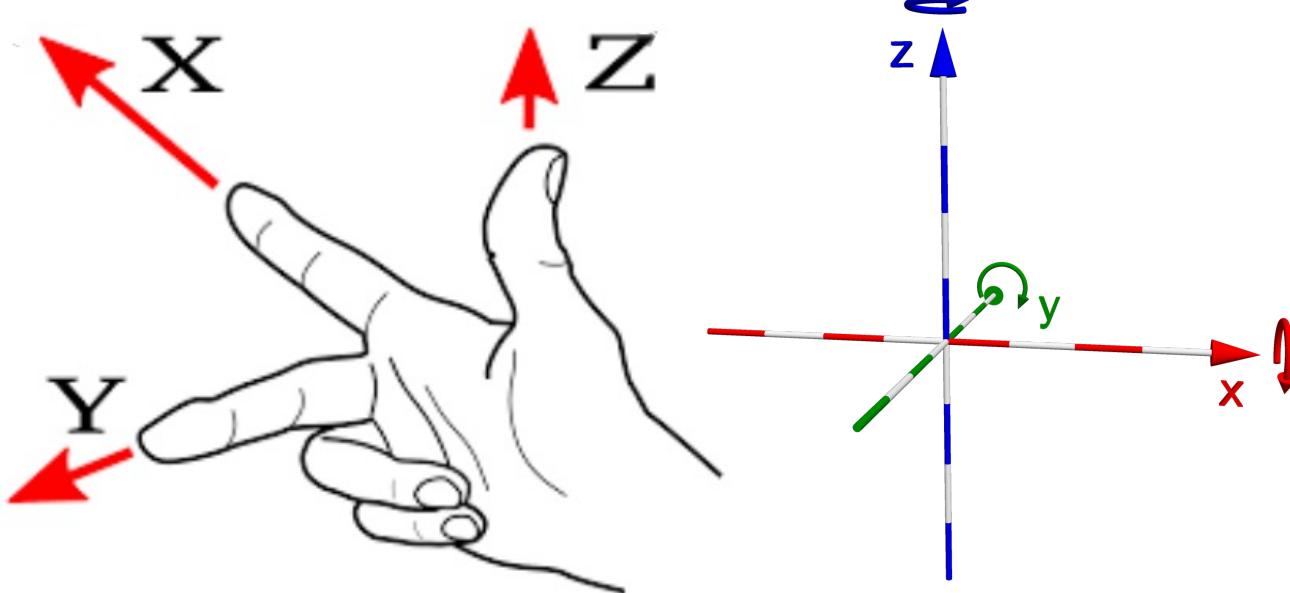
$$\mathbf{X}' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \mathbf{J} * \boldsymbol{\nu}$$



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 independent variables that describes system.

In robotics typically position and rotation.

for 2D: (x, y, θ) ... 3DoF

for 3D: $(x, y, z, \text{roll}, \text{pitch}, \text{yaw})$... 6DoF

Human hand ... 27 DoF

In robotics we use vectors and matrices 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.

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

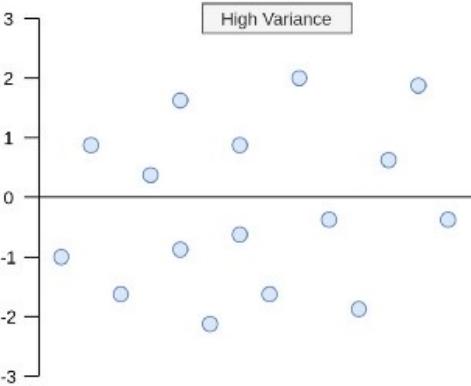
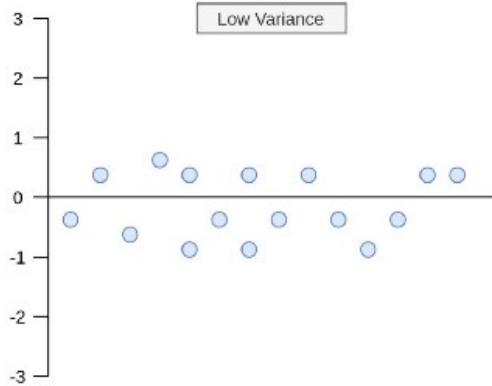
Covariance Matrix ... describes relations between variables in **state vector**.

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

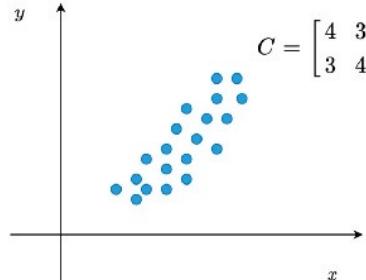
$$\text{var}(x) = \frac{\sum_{i=0}^n (x_i - \mu)^2}{N}$$

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

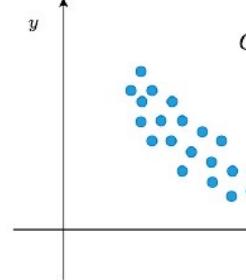
$$\Sigma = \begin{bmatrix} \text{var}(x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, x) & \text{var}(y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{var}(z) \end{bmatrix} \begin{matrix} x \\ y \\ z \end{matrix}$$



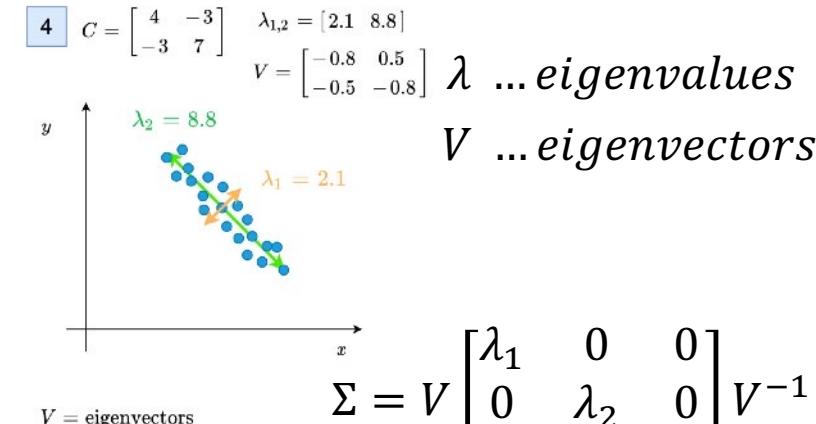
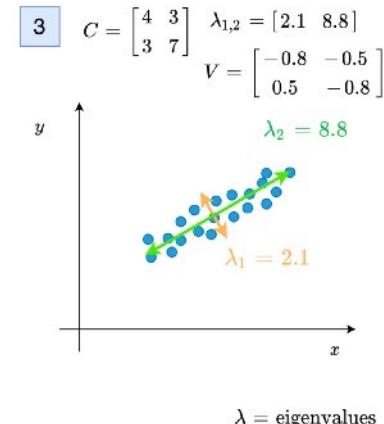
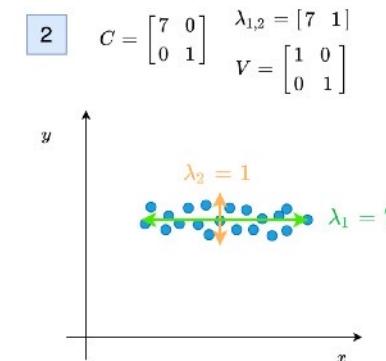
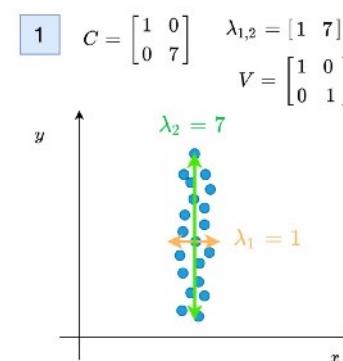
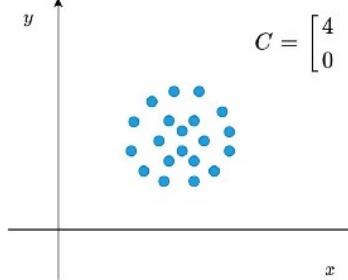
Positive Covariance



Negative Covariance



Zero Covariance





Prerequisites / Cheat Sheet: Linear Model

State Vector and Control Vector

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

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)

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



Process Model (Motion model)

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

Measurement Model

$$z_t = H_t \mu_t + \delta \quad z_t = h(\mu_t) + \delta$$

linear

vs

non-linear

Typically:

non-linear physics models
angular calculations

$$\mathbf{x} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

s ... scale
 R ... rotation
 T ... translation

Homogeneous Euclidian

$$\mathbf{x}_H = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \leftrightarrow \mathbf{x}_E = \begin{bmatrix} x \\ y \end{bmatrix}$$

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

$$\begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \leftrightarrow \begin{bmatrix} \text{inf} \\ \text{inf} \end{bmatrix}$$

Applying Transformation

$$\mathbf{x}' = H\mathbf{x}$$

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

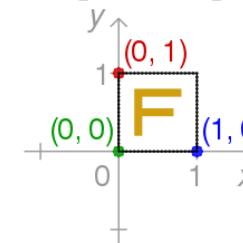
$$\mathbf{x}' = H_1 H_2 \mathbf{x} \neq H_2 H_1 \mathbf{x}$$

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

Note: usually $\lambda=1$

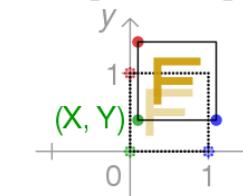
No change

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



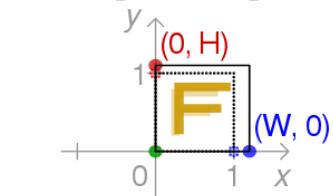
Translate

$$\begin{bmatrix} 1 & 0 & X \\ 0 & 1 & Y \\ 0 & 0 & 1 \end{bmatrix}$$



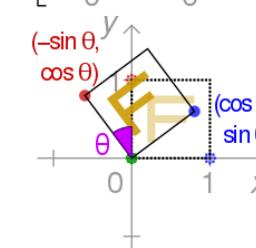
Scale about origin

$$\begin{bmatrix} W & 0 & 0 \\ 0 & H & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



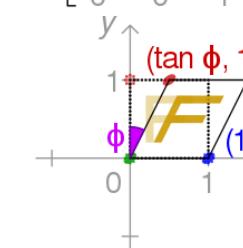
Rotate about origin

$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



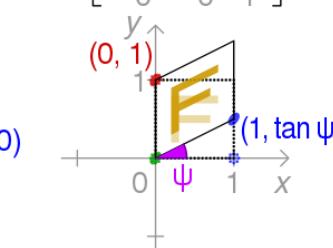
Shear in x direction

$$\begin{bmatrix} 1 & \tan \phi & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



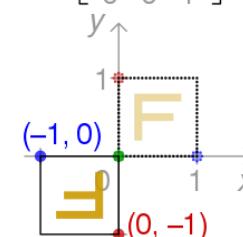
Shear in y direction

$$\begin{bmatrix} 1 & 0 & 0 \\ \tan \psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



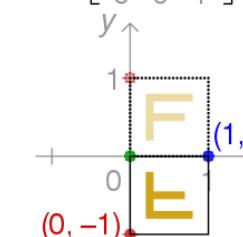
Reflect about origin

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



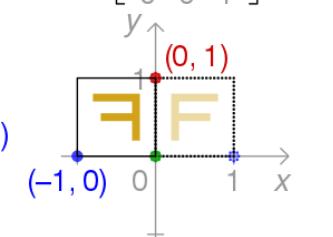
Reflect about x-axis

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

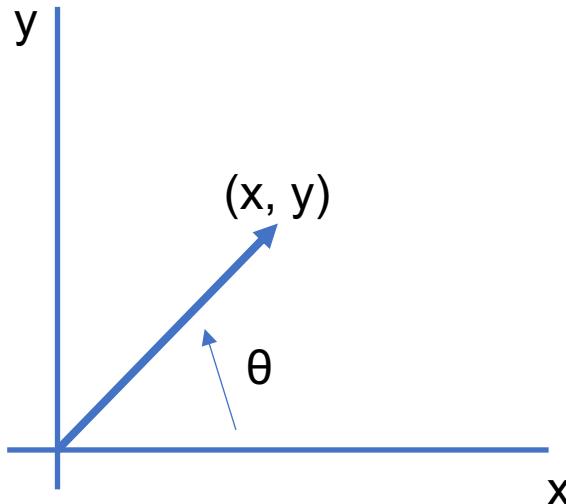


Reflect about y-axis

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

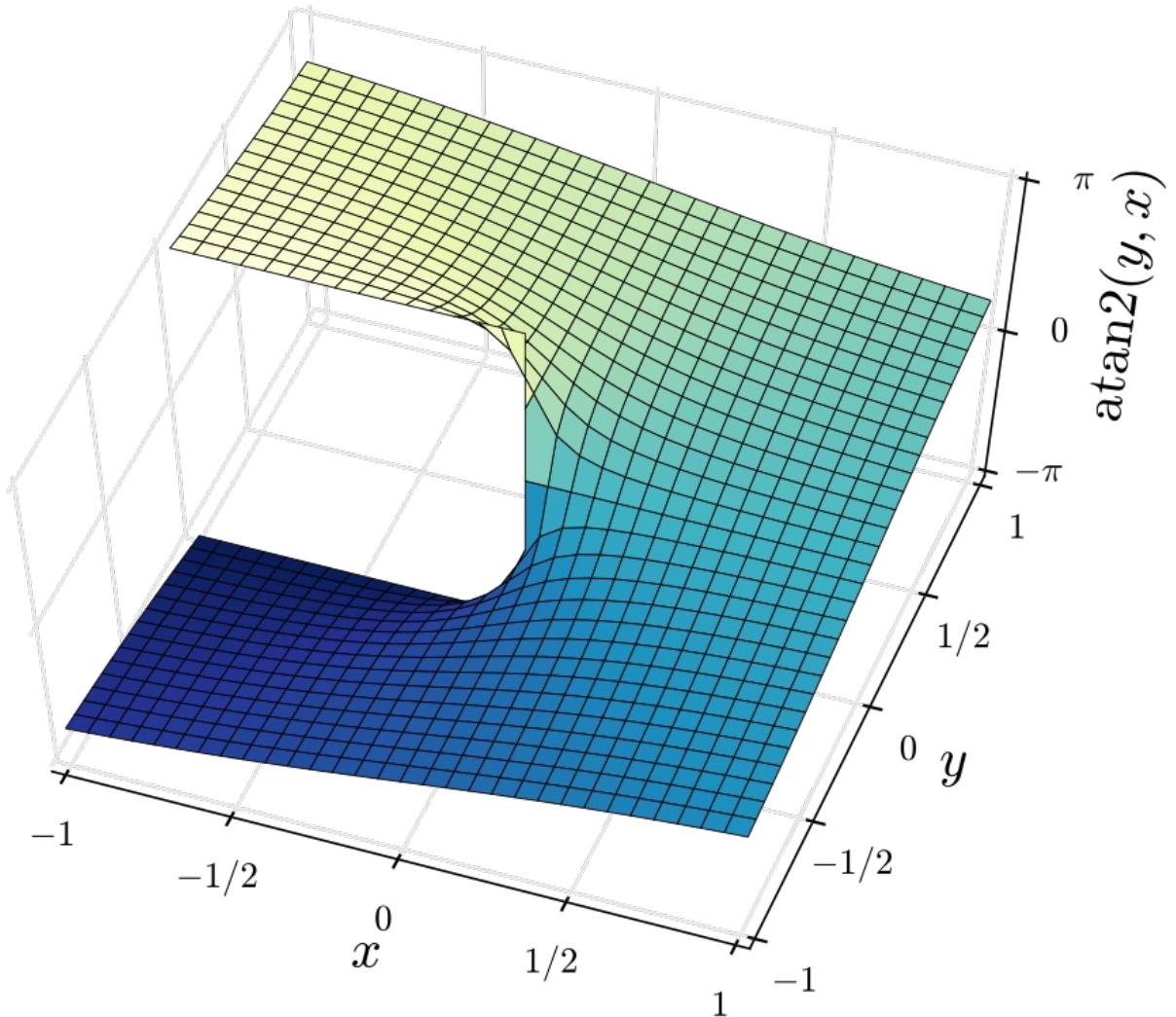


Function resolves the angle of (x,y) vector.

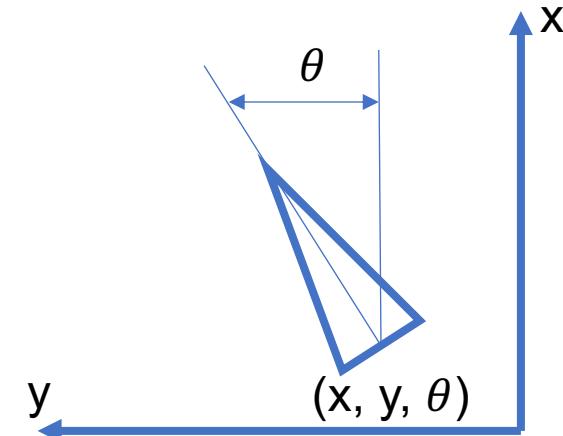
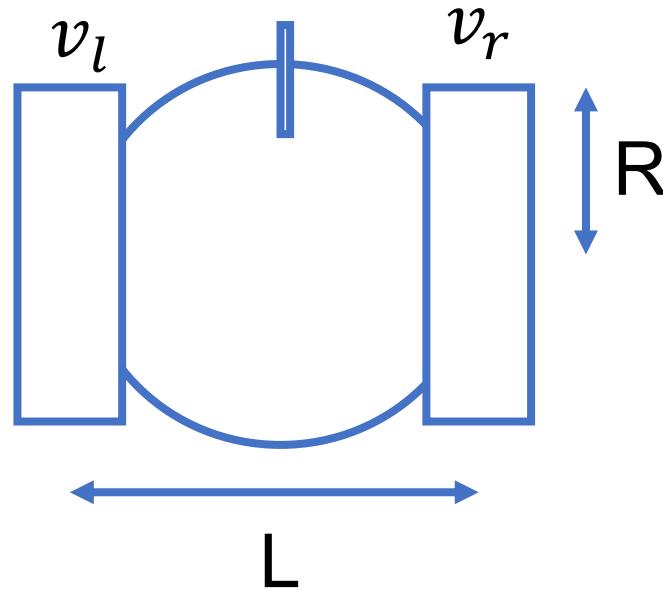


$$\theta = \text{atan2}(y, x)$$

Tip: in robotics, always normalize angle $<-\pi, \pi>$



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

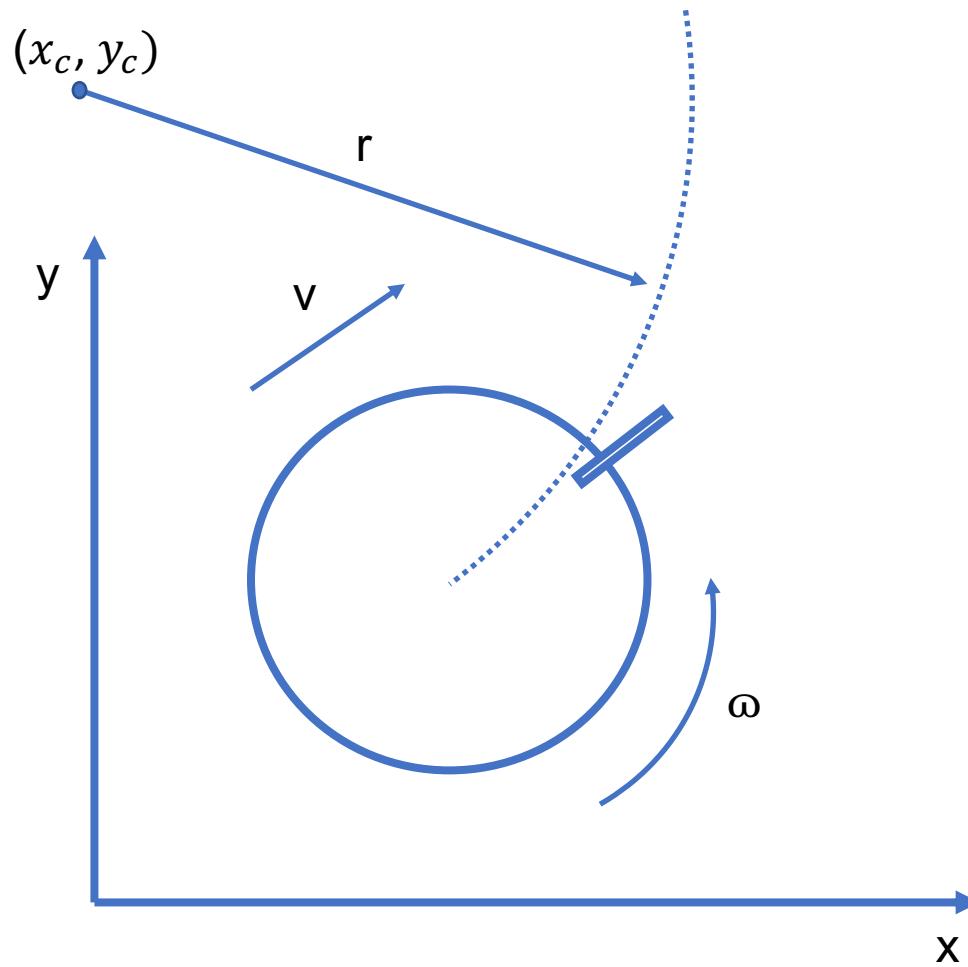


$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{R}{2}(v_l + v_r) * \cos(\theta) \\ \frac{R}{2}(v_l + v_r) * \sin(\theta) \\ \frac{R}{L}(v_r - v_l) \end{bmatrix}$$

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

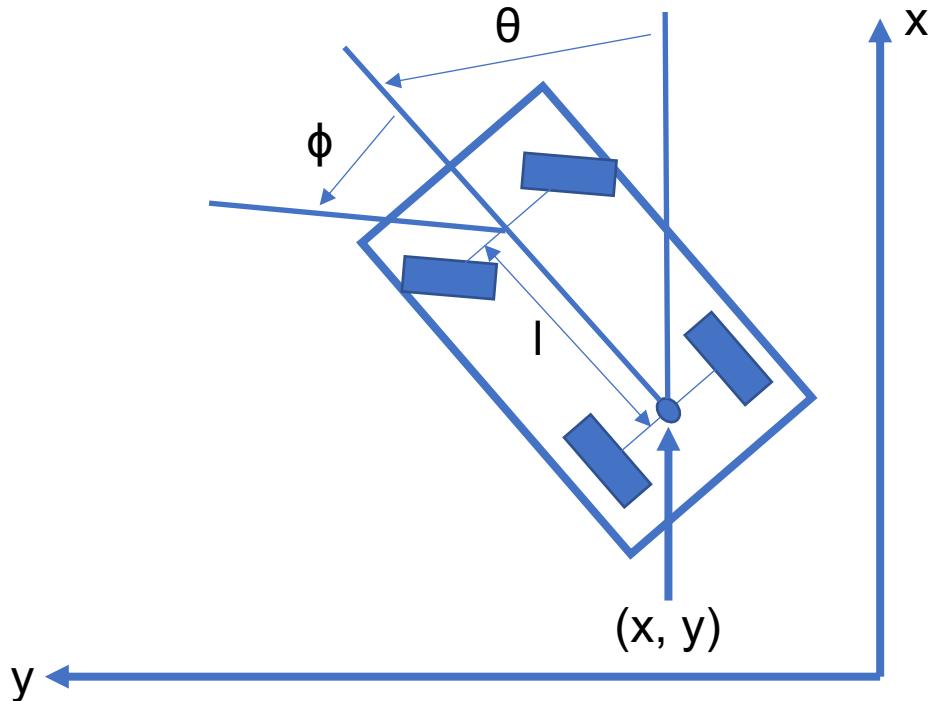
$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{R}{2}(v_l + v_r) \\ \frac{R}{L}(v_r - v_l) \end{bmatrix}$$

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



$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} x_c + r \sin(\theta + \omega\Delta t) \\ y_c - r \cos(\theta + \omega\Delta t) \\ \theta + \omega\Delta t \end{bmatrix}$$

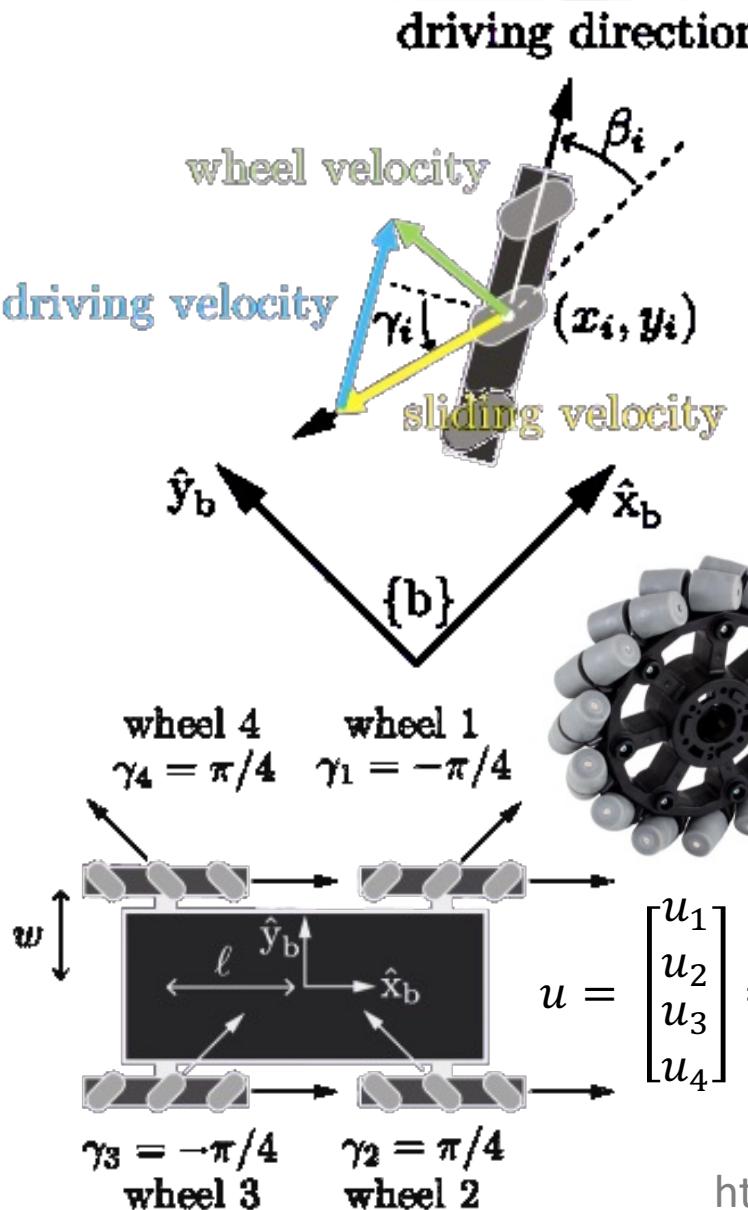
$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} -\frac{v}{\omega} \sin(\theta) + \frac{v}{\omega} \sin(\theta + \omega\Delta t) \\ \frac{v}{\omega} \cos(\theta) - \frac{v}{\omega} \cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{bmatrix}$$



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

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

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



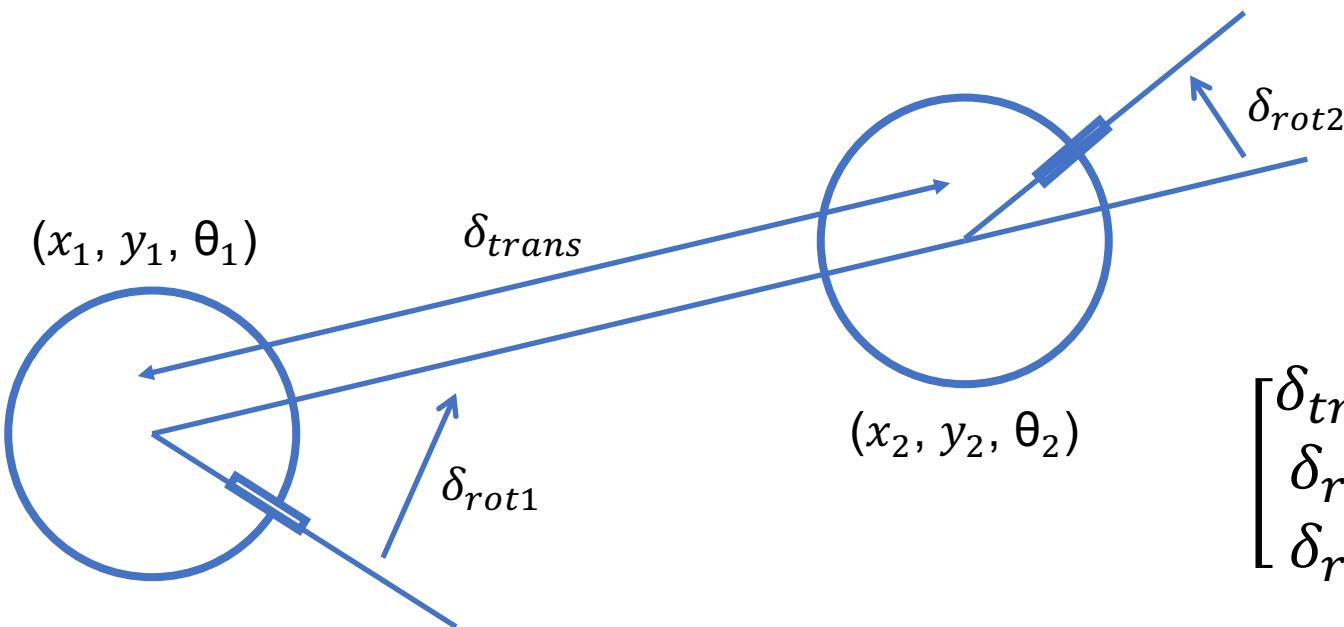
$$u_i = \frac{1}{r} [1 \quad \tan(\gamma_i)] \begin{bmatrix} \cos(\beta_i) & \sin(\beta_i) \\ -\sin(\beta_i) & \cos(\beta_i) \end{bmatrix} \begin{bmatrix} -y_i & 1 & 0 \\ x_i & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_b \\ v_{xb} \\ v_{yb} \end{bmatrix}$$

Motor speed components in driving dir. to lin. vel. in wheel frame to wheel's velocities in chassis frame chassis velocities

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

Diagram of a triangular omnidirectional chassis with three wheels. Wheel 1 is at the top with $\gamma_1 = 0, \beta_1 = 0$. Wheel 2 is at the bottom-right with $\gamma_2 = 0, \beta_2 = -2\pi/3$. Wheel 3 is at the bottom-left with $\gamma_3 = 0, \beta_3 = 2\pi/3$. The chassis frame $\{b\}$ has axes \hat{x}_b and \hat{y}_b . The distance between the centers of wheel 2 and wheel 3 is d .

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



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

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