



上海科技大学
ShanghaiTech University

CS283: Robotics Spring 2023: Kinematics

Sören Schwertfeger / 师泽仁

ShanghaiTech University

KINEMATICS

Motivation

- Autonomous mobile robots move around in the environment.

Therefore **ALL** of them:

- They need to know **where** they **are**.
- They need to know **where** their **goal** is.
- They need to know **how** to get there.

- **Odometry!**

- Robot:

- I know how fast the wheels turned =>
- I know how the robot moved =>
- I know where I am 😊

Odometry

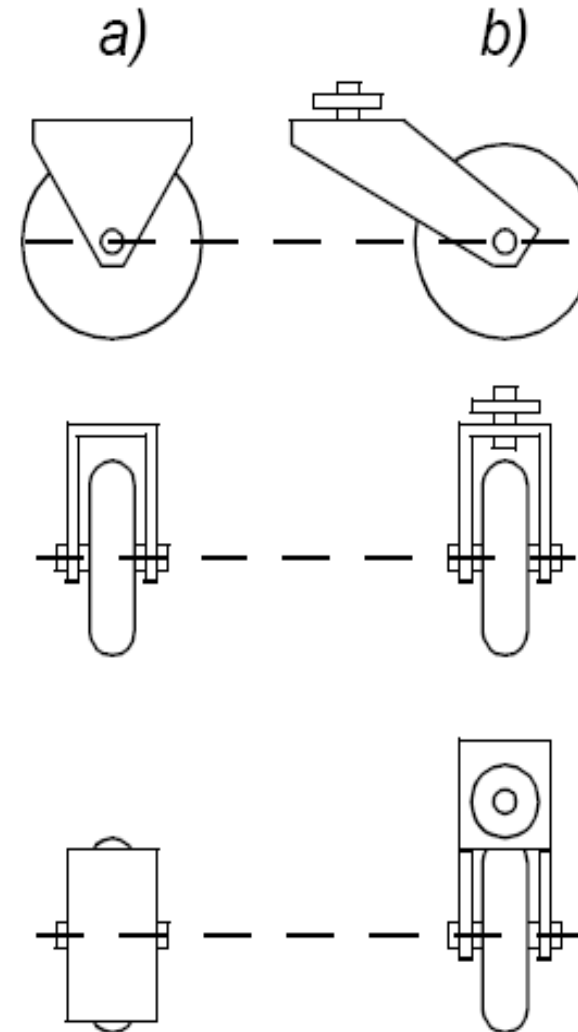
- Robot:
 - I know how fast the wheels turned =>
 - I know how the robot moved =>
 - I know where I am 😊
- Marine Navigation: Dead reckoning (using heading sensor)
- Sources of error (AMR pages 269 - 270):
 - Wheel slip
 - Uneven floor contact (non-planar surface)
 - Robot kinematic: tracked vehicles, 4 wheel differential drive..
 - Integration from speed to position: Limited resolution (time and measurement)
 - Wheel misalignment
 - Wheel diameter uncertainty
 - Variation in contact point of wheel

Mobile Robots with Wheels

- Wheels are the most appropriate solution for most applications
- Three wheels are sufficient to guarantee stability
- With more than three wheels an appropriate suspension is required
- Selection of wheels depends on the application

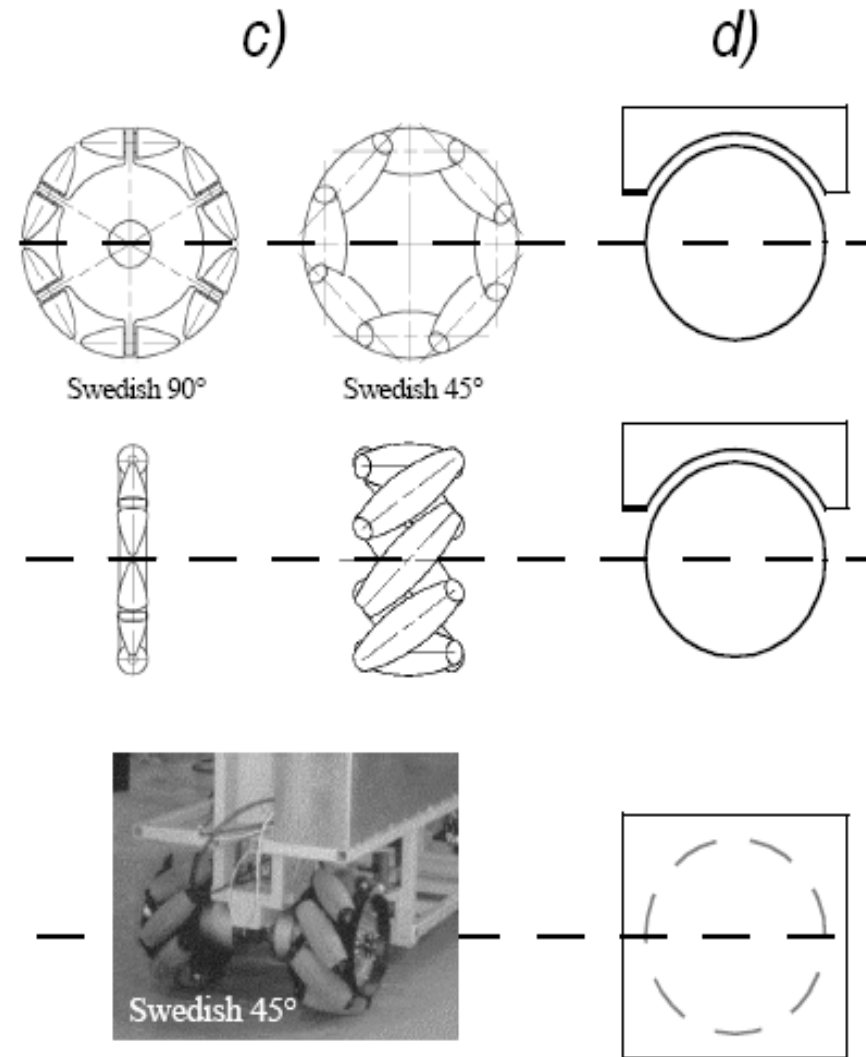
The Four Basic Wheels Types I

- a) Standard wheel: Two degrees of freedom; rotation around the (motorized) wheel axle and the contact point
- b) Castor wheel: Three degrees of freedom; rotation around the wheel axle, the contact point and the castor axle



The Four Basic Wheels Types II

- c) Swedish wheel: Three degrees of freedom; rotation around the (motorized) wheel axle, around the rollers and around the contact point
- d) Ball or spherical wheel: Suspension technically not solved

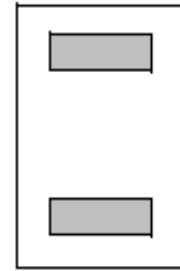


Characteristics of Wheeled Robots and Vehicles

- Stability of a vehicle is be guaranteed with 3 wheels
 - center of gravity is within the triangle with is formed by the ground contact point of the wheels.
- Stability is improved by 4 and more wheel
 - however, this arrangements are hyperstatic and require a flexible suspension system.
- Bigger wheels allow to overcome higher obstacles
 - but they require higher torque or reductions in the gear box.
- Most arrangements are non-holonomic (see chapter 3)
 - require high control effort
- Combining actuation and steering on one wheel makes the design complex and adds additional errors for odometry.

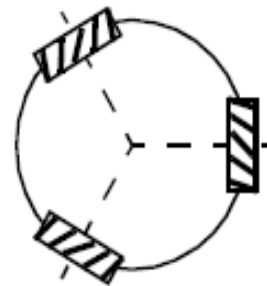
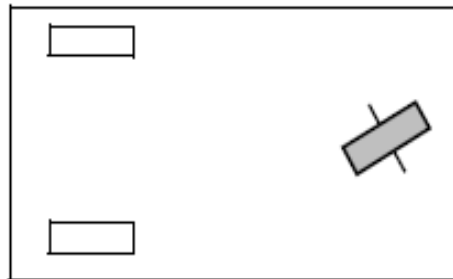
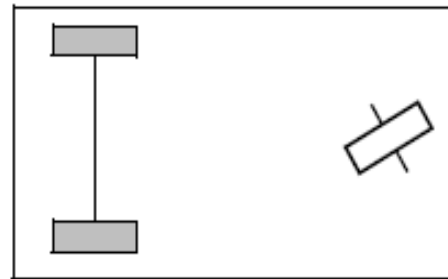
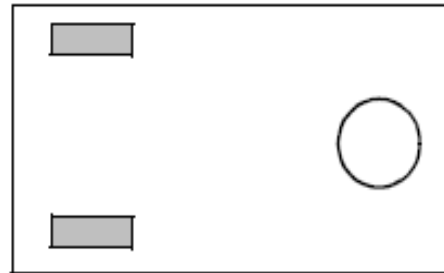
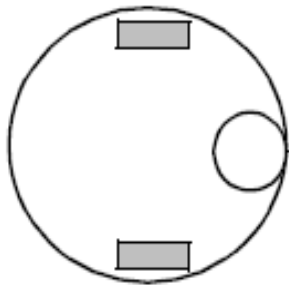
Different Arrangements of Wheels I

- Two wheels

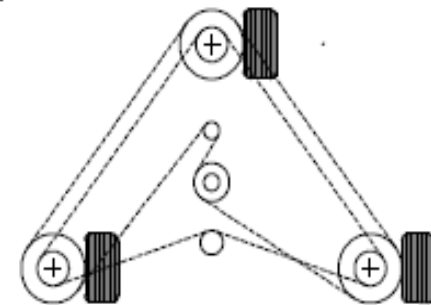


Center of gravity below axle

- Three wheels



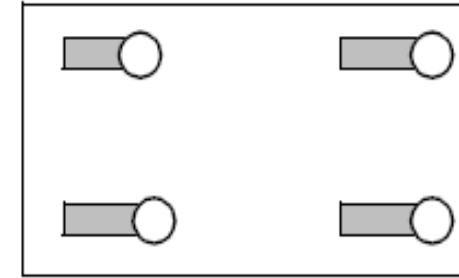
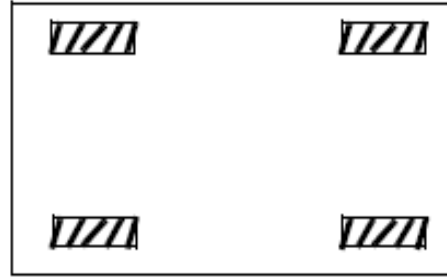
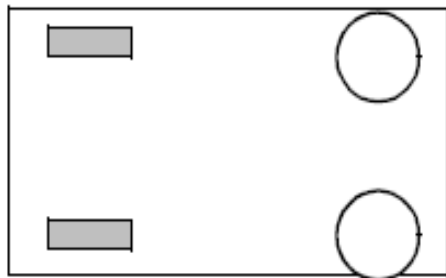
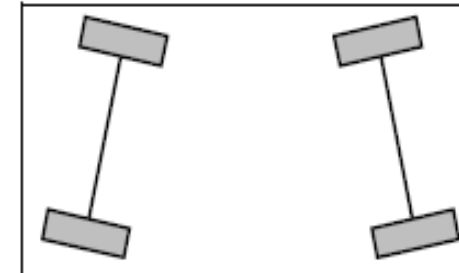
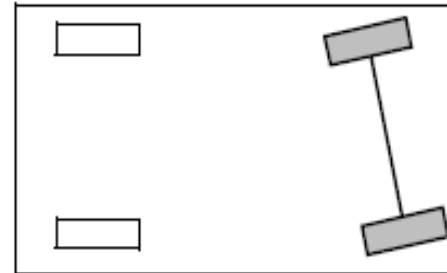
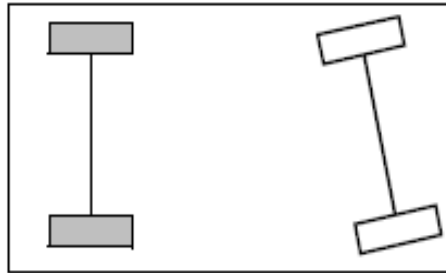
Omnidirectional Drive



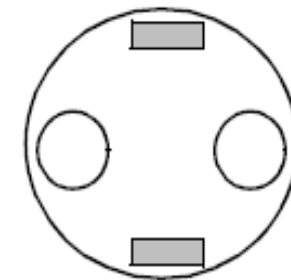
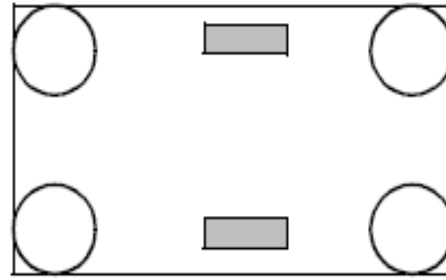
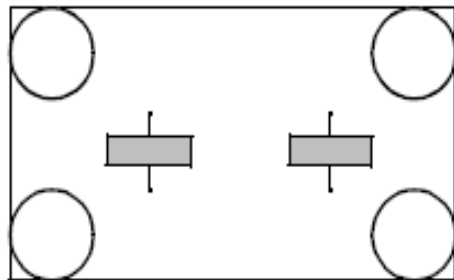
Synchro Drive

Different Arrangements of Wheels II

- Four wheels

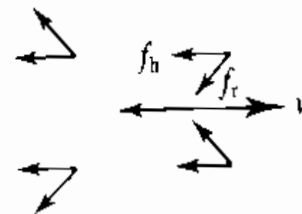
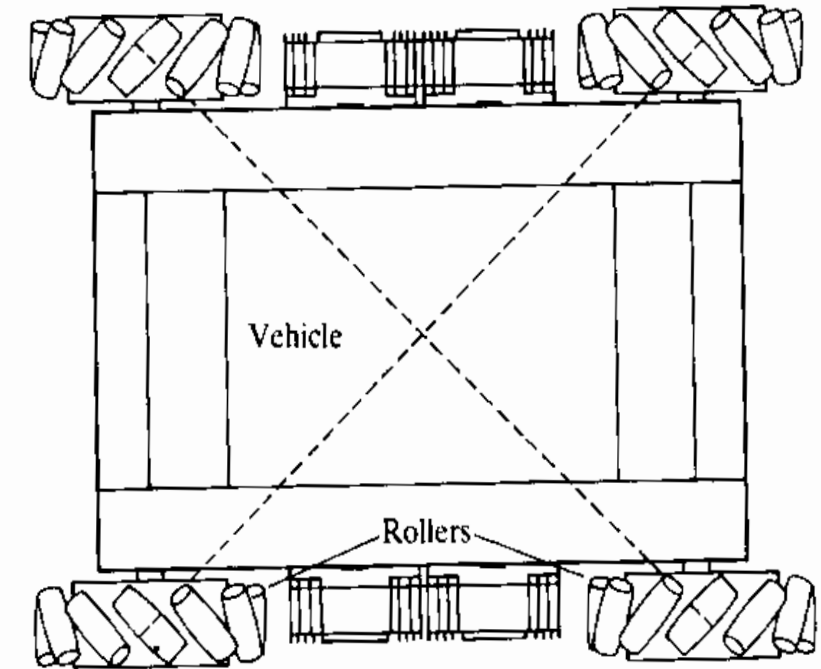
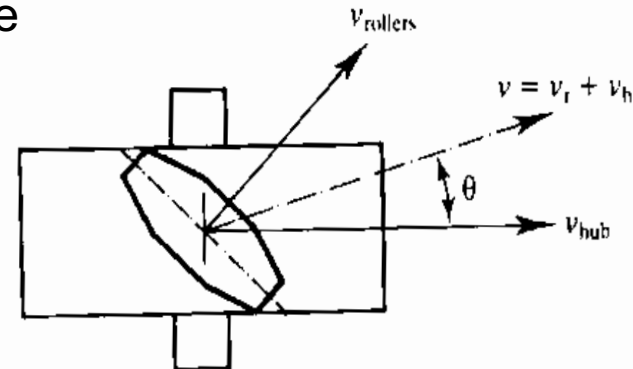
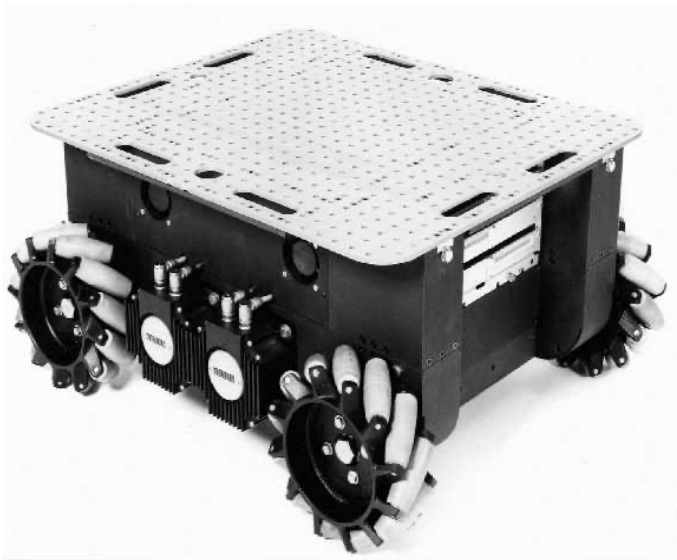


- Six wheels

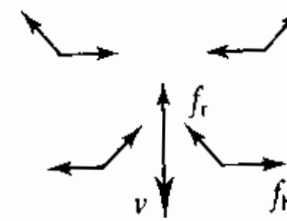


Uranus, CMU: Omnidirectional Drive with 4 Wheels

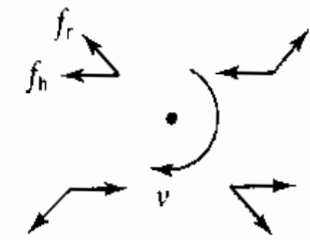
- Movement in the plane has 3 DOF
 - thus only three wheels can be independently controlled
 - It might be better to arrange three swedish wheels in a triangle



Forward



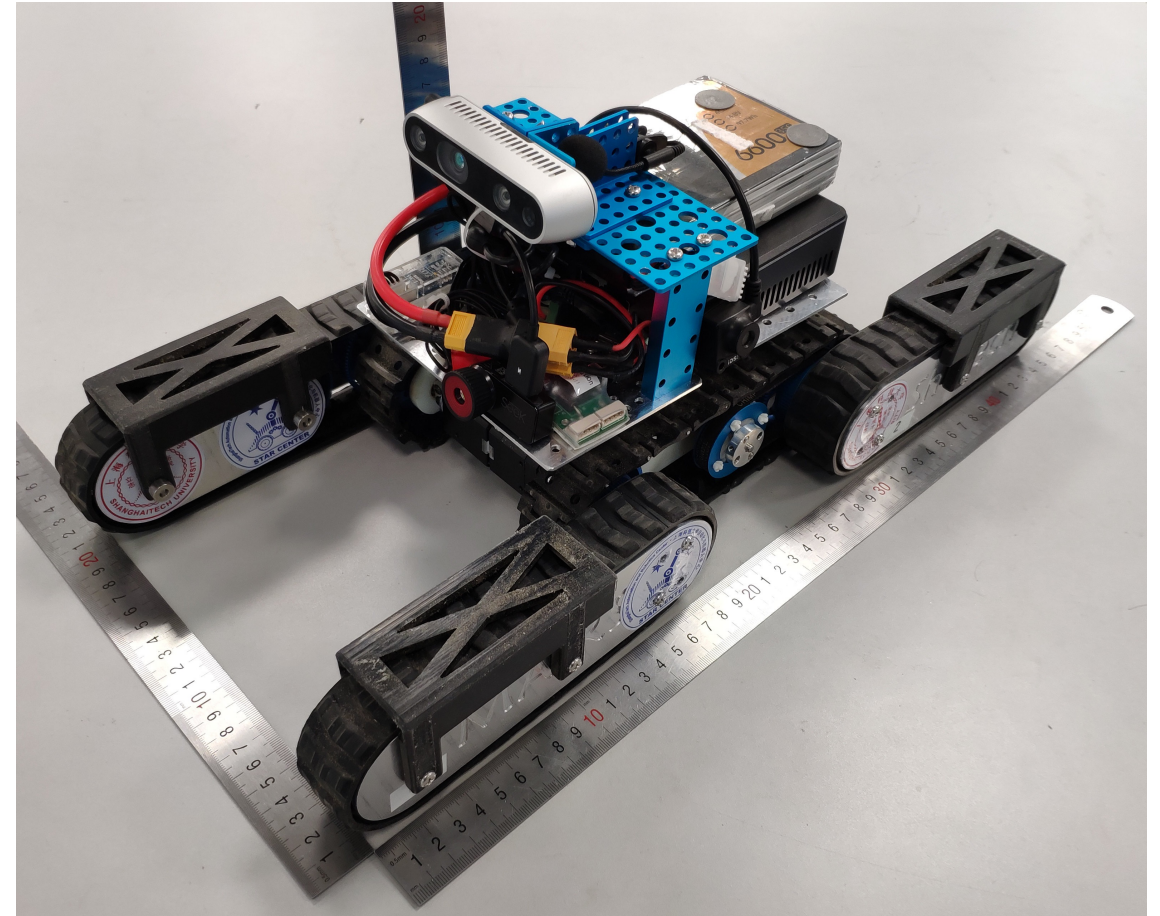
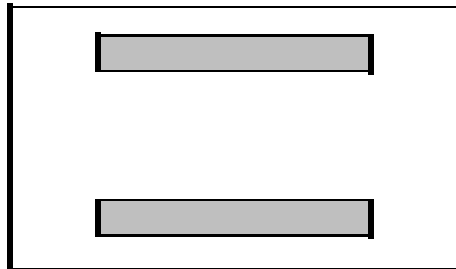
Right



Clockwise

MARS Rescue Robot: Tracked Differential Drive

- Kinematic Simplification:
 - 2 Wheels, located at the center

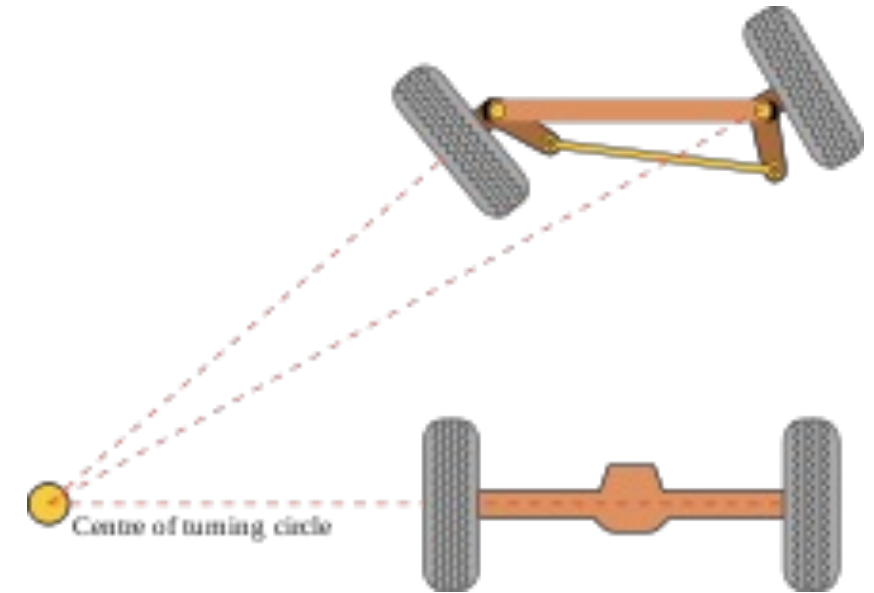


Differential Drive Robots



Ackermann Robot

- No sideways slip than differential drive during turning 😊
- Cannot turn on the spot 😞



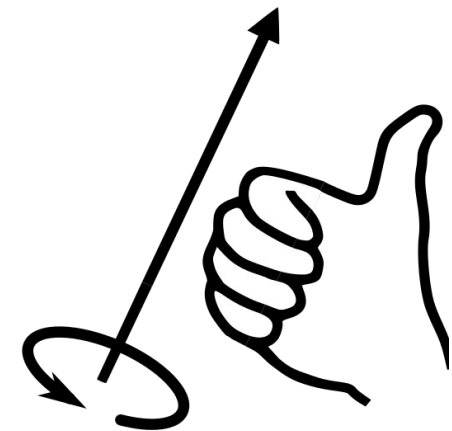
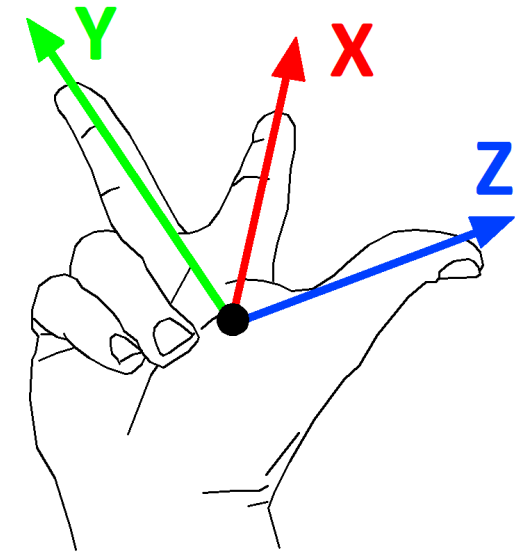
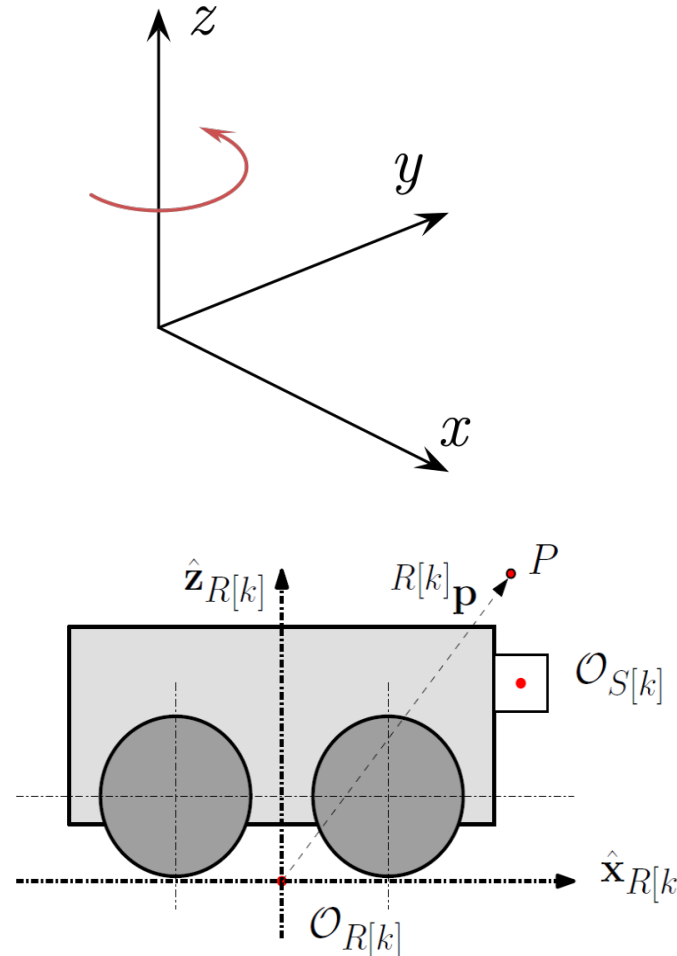
Introduction: Mobile Robot Kinematics

- Aim
 - Description of mechanical behavior of the robot for *design* and *control*
 - Similar to robot manipulator kinematics
 - However, mobile robots can move unbound with respect to its environment
 - there is no direct way to measure the robot's position
 - Position must be integrated over time
 - Leads to inaccuracies of the position (motion) estimate
 - > *the number 1 challenge in mobile robotics*

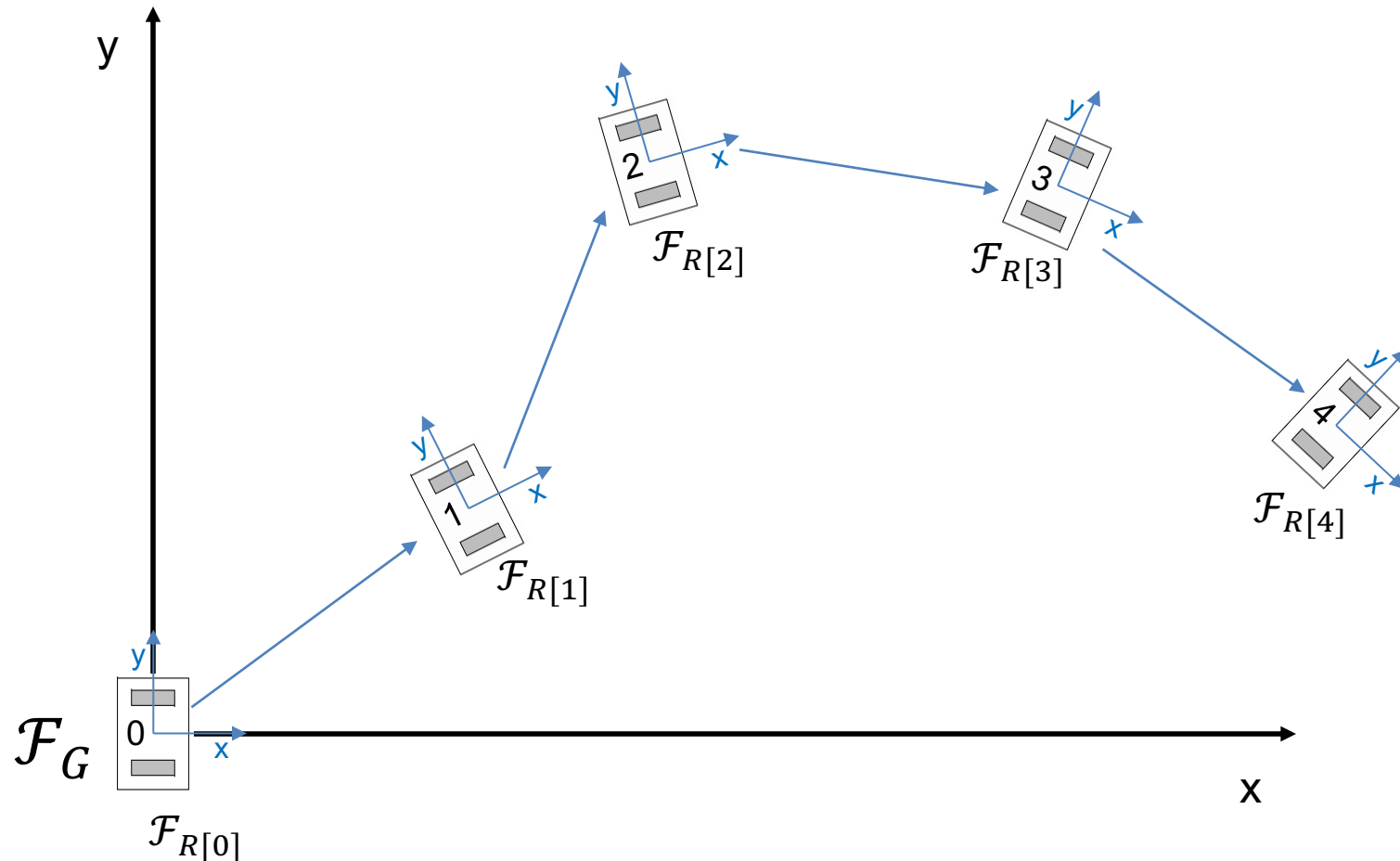
COORDINATE SYSTEM

Right Hand Coordinate System

- Standard in Robotics
- Positive rotation around X is anti-clockwise
- Right-hand rule mnemonic:
 - Thumb: z-axis
 - Index finger: x-axis
 - Second finger: y-axis
 - Rotation: Thumb = rotation axis, positive rotation in finger direction
- Robot Coordinate System:
 - X front
 - Z up (Underwater: Z down)
 - Y ???



Odometry



With respect to the robot start pose:

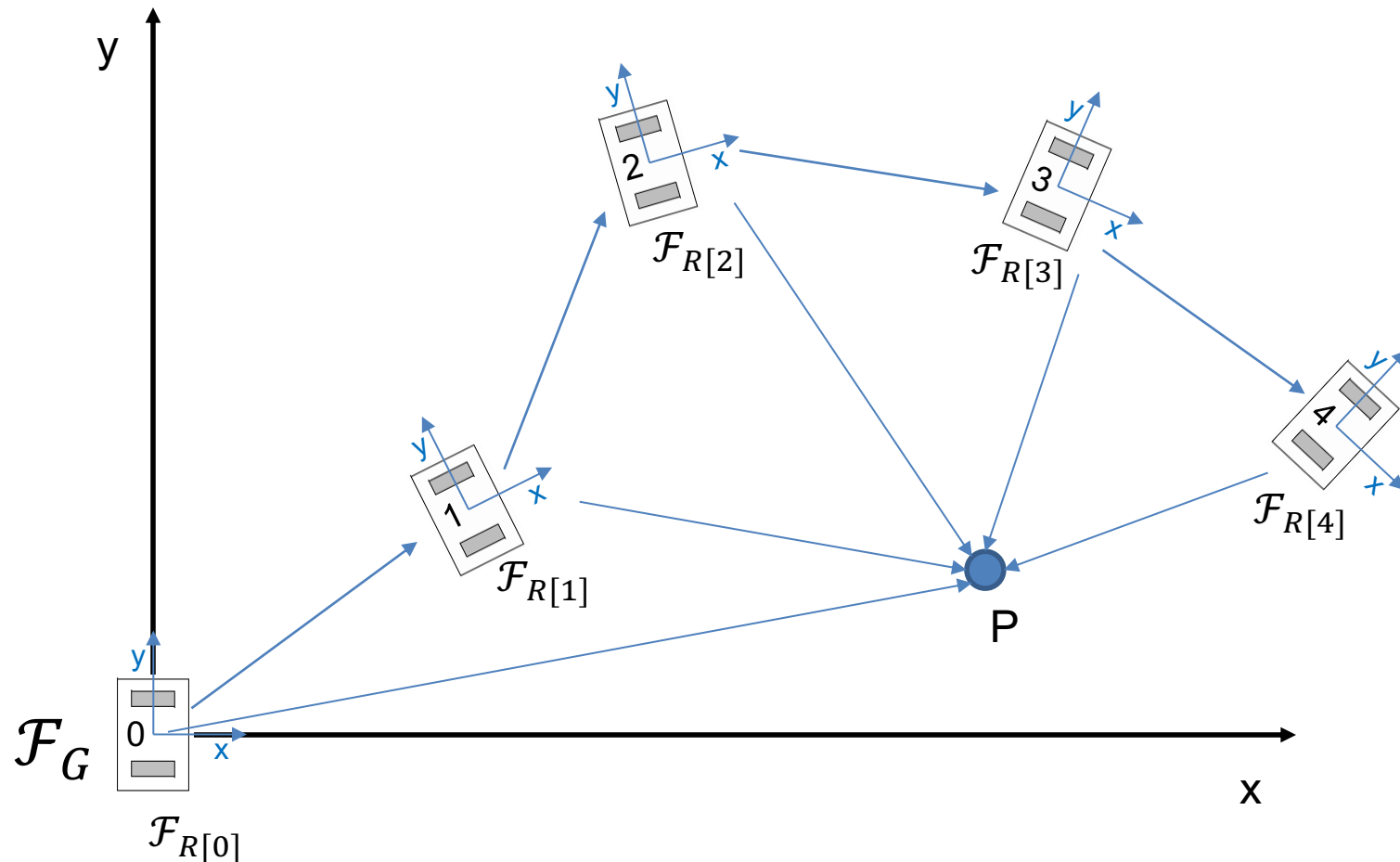
Where is the robot now?

Two approaches – same result:

- Geometry (easy in 2D)
- Transforms (better for 3D)

$\mathcal{F}_{R[X]}$: The **F**rame of reference (the local coordinate system) of the **R**obot at the time **X**

Use of robot frames $\mathcal{F}_{R[X]}$

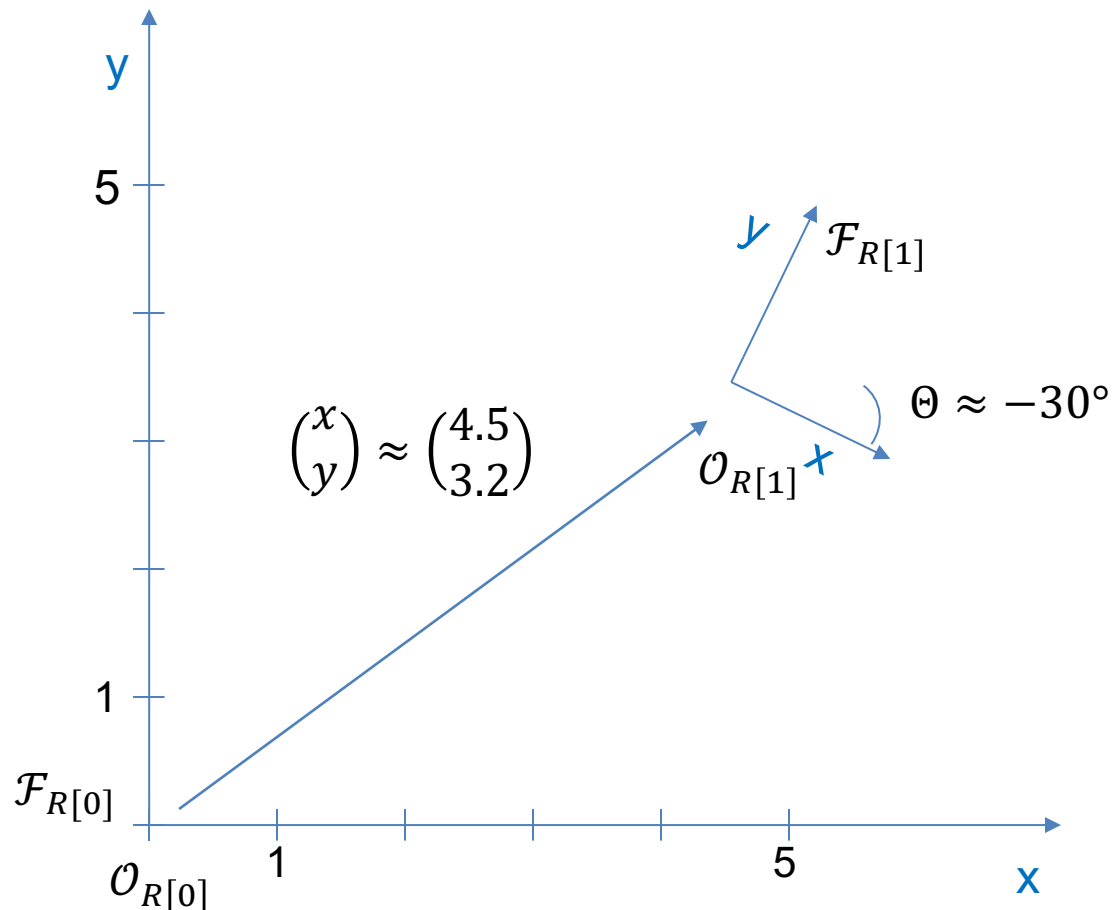


$\mathcal{O}_{R[X]}$: Origin of $\mathcal{F}_{R[X]}$
(coordinates (0, 0))

$\overrightarrow{\mathcal{O}_{R[X]}P}$: position vector from $\mathcal{O}_{R[X]}$ to point P - $\begin{pmatrix} x \\ y \end{pmatrix}$

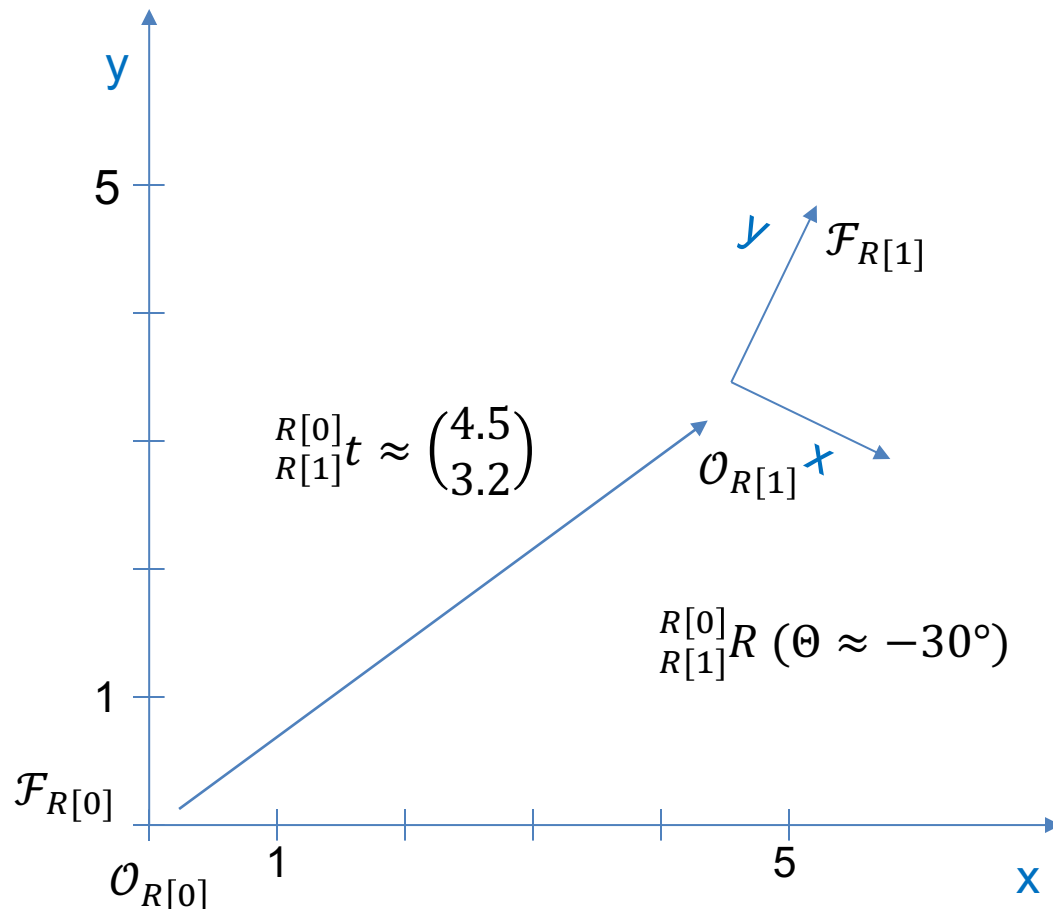
- Object P is observed at times 0 to 4
- Object P is static (does not move)
- The Robot moves (e.g. $\mathcal{F}_{R[0]} \neq \mathcal{F}_{R[1]}$)
- \Rightarrow (x, y) coordinates of P are different in all frames, for example:
 - $\overrightarrow{\mathcal{O}_{R[0]}P} \neq \overrightarrow{\mathcal{O}_{R[1]}P}$

Position, Orientation & Pose



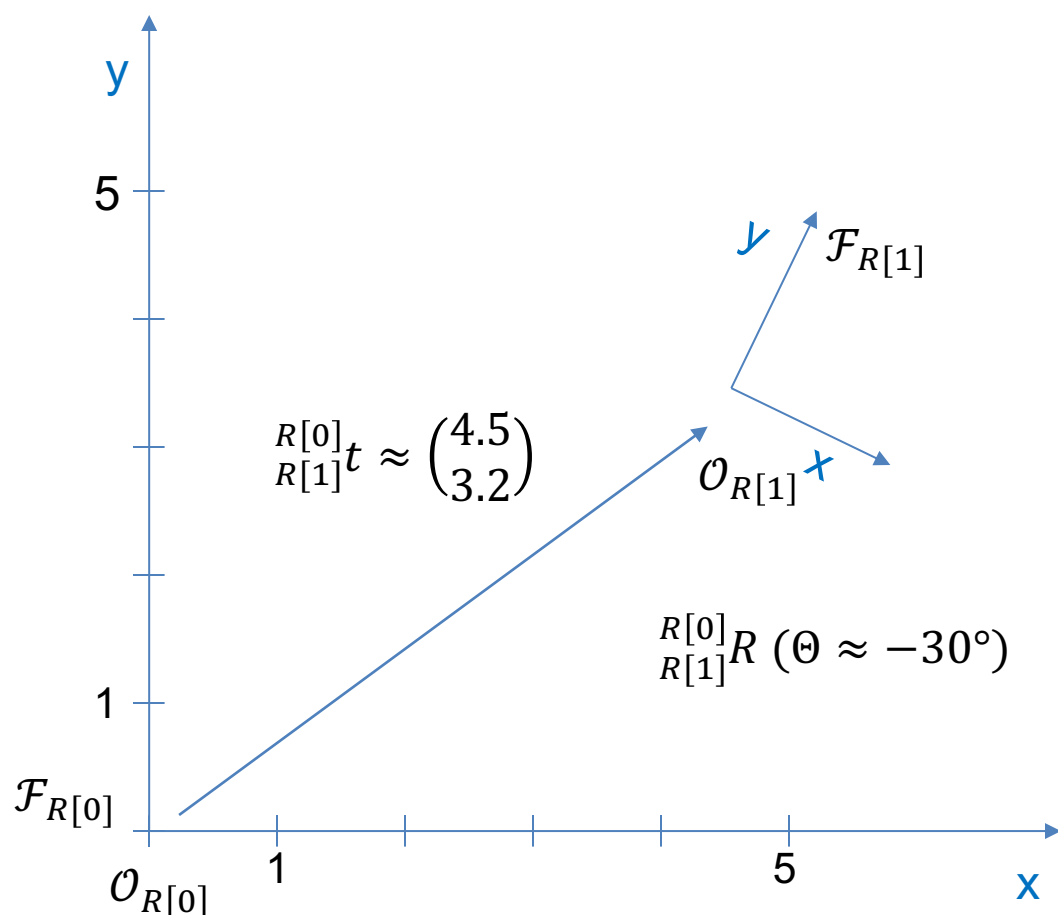
- **Position:**
 - $\begin{pmatrix} x \\ y \end{pmatrix}$ coordinates of any object or point (or another frame)
 - with respect to (wrt.) a specified frame
- **Orientation:**
 - (θ) angle of any oriented object (or another frame)
 - with respect to (wrt.) a specified frame
- **Pose:**
 - $\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$ position and orientation of any oriented object
 - with respect to (wrt.) a specified frame

Translation, Rotation & Transform



- **Translation:**
 - $\begin{pmatrix} x \\ y \end{pmatrix}$ difference, change, motion from one reference frame to another reference frame
- **Rotation:**
 - (Θ) difference in angle, rotation between one reference frame and another reference frame
- **Transform:**
 - $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$ difference, motion between one reference frame and another reference frame

Position & Translation, Orientation & Rotation



- $\mathcal{F}_{R[X]}$: Frame of reference of the robot at time X
- Where is that frame $\mathcal{F}_{R[X]}$?
 - Can only be expressed with respect to (wrt.) another frame (e.g. global Frame \mathcal{F}_G) =>
 - Pose of $\mathcal{F}_{R[X]}$ wrt. \mathcal{F}_G

- $O_{R[X]}$: Origin of $\mathcal{F}_{R[X]}$
 - $\overrightarrow{O_{R[X]}O_{R[X+1]}}$: **Position** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

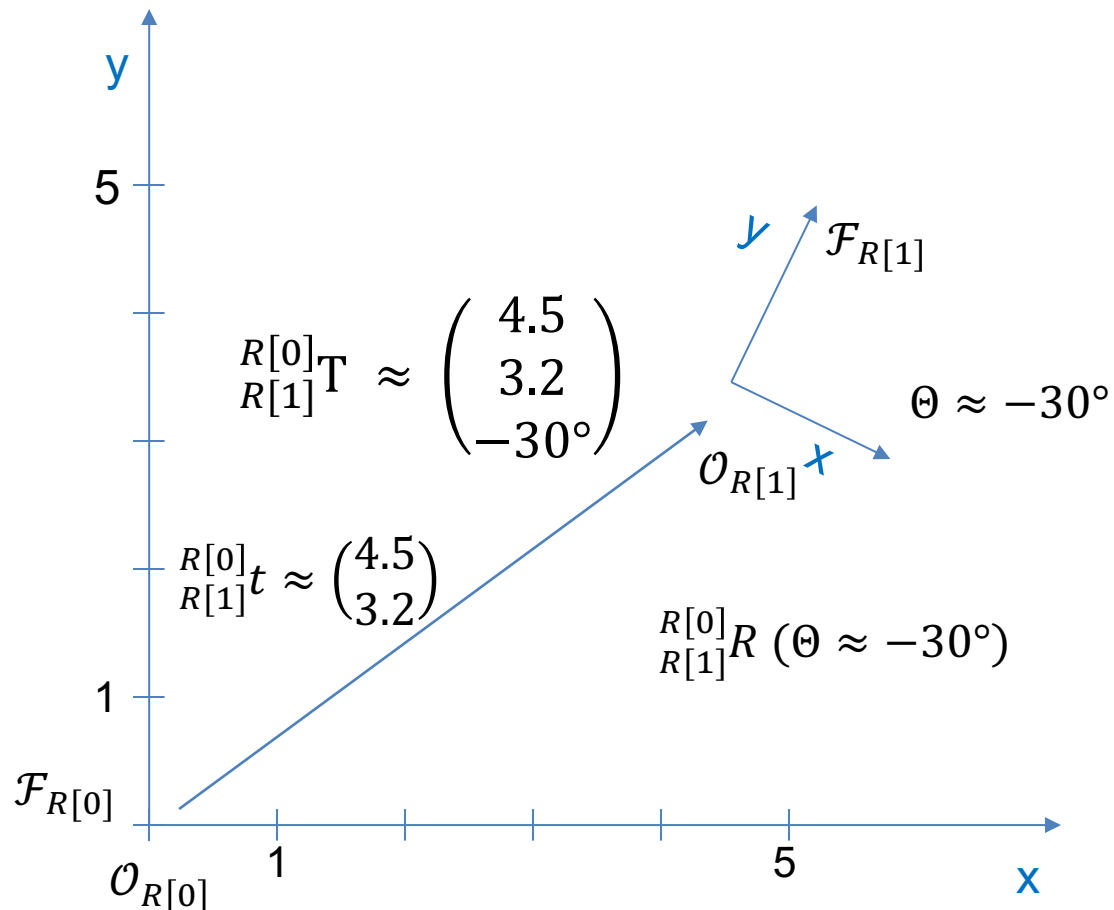
so $O_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

\triangleq ${}^{R[X]}_{R[X+1]}t$: **Translation**

- The angle θ between the x-Axes:
 - **Orientation** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

\triangleq ${}^{R[X]}_{R[X+1]}R$: **Rotation** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

Transform



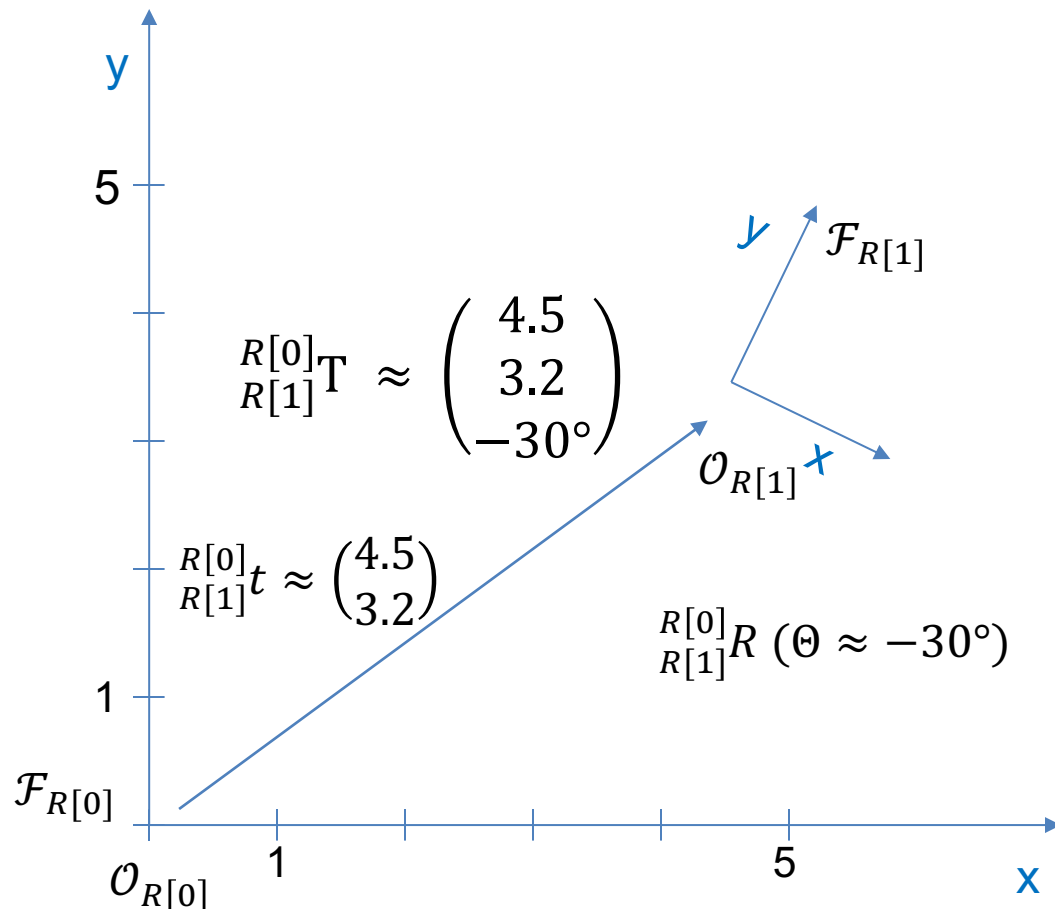
- $\begin{matrix} R[X] \\ R[X+1] \end{matrix} t$: **Translation**
 - Position vector (x, y) of $R[X + 1]$ wrt. $R[X]$
- $\begin{matrix} R[X] \\ R[X+1] \end{matrix} R$: **Rotation**
 - Angle (θ) of $R[X + 1]$ wrt. $R[X]$
- **Transform:** $\begin{matrix} R[X] \\ R[X+1] \end{matrix} T \equiv \left\{ \begin{matrix} R[X] \\ R[X+1] \end{matrix} \begin{matrix} t \\ R \end{matrix} \right\}$

Geometry approach to Odometry

We want to know:

- Position of the robot (x, y)
- Orientation of the robot (θ)
- => together: Pose $\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$

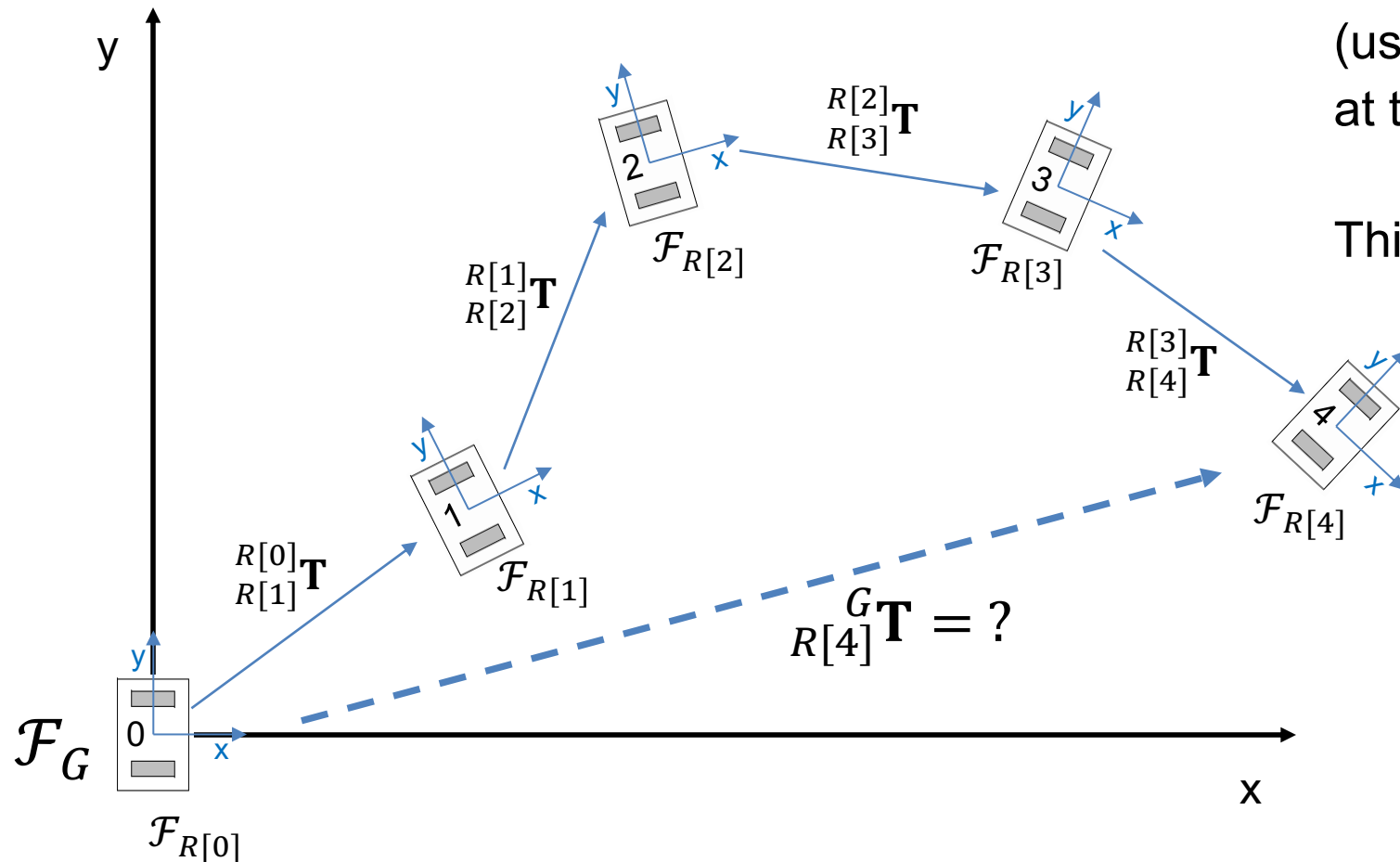
With respect to (wrt.) \mathcal{F}_G : The global frame; global coordinate system



$$\mathcal{F}_{R[0]} = \mathcal{F}_G \Rightarrow {}^G \mathcal{F}_{R[0]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$${}^G \mathcal{F}_{R[1]} = R_{R[1]}^{R[0]} T \approx \begin{pmatrix} 4.5 \\ 3.2 \\ 30^\circ \end{pmatrix}$$

Mathematical approach: Transforms



Where is the Robot now?

The pose of $\mathcal{F}_{R[X]}$ with respect to \mathcal{F}_G (usually = $\mathcal{F}_{R[0]}$) is the pose of the robot at time X.

This is equivalent to ${}^G R[X] \mathbf{T}$

Chaining of Transforms

$${}^G R[X+1] \mathbf{T} = {}^G R[X] \mathbf{T} {}^{R[X]} R[X+1] \mathbf{T}$$

often: $\mathcal{F}_G \equiv \mathcal{F}_{R[0]} \Rightarrow {}^G R[0] \mathbf{T} = id$

ADMIN

Project

- 2 credit points!
- Work in groups, min 2 students, max 3 students!
- Next lecture: Topics will be proposed...
 - You can also do your own topic, but only after approval of Prof. Schwertfeger
 - Prepare a short, written proposal till next Tuesday!
- Topic selection: next Thursday!
 - One member writes an email for the whole group to Ziwen: zhuangzw (at)shanghaitech.edu.cn ; Put the other group members on CC
 - Subject: [Robotics] Group Selection
- One graduate student from my group will co-supervise your project
- Weekly project meetings!
- Oral "exams" to evaluate the contributions of each member
- No work on project => bad grade of fail

Grading

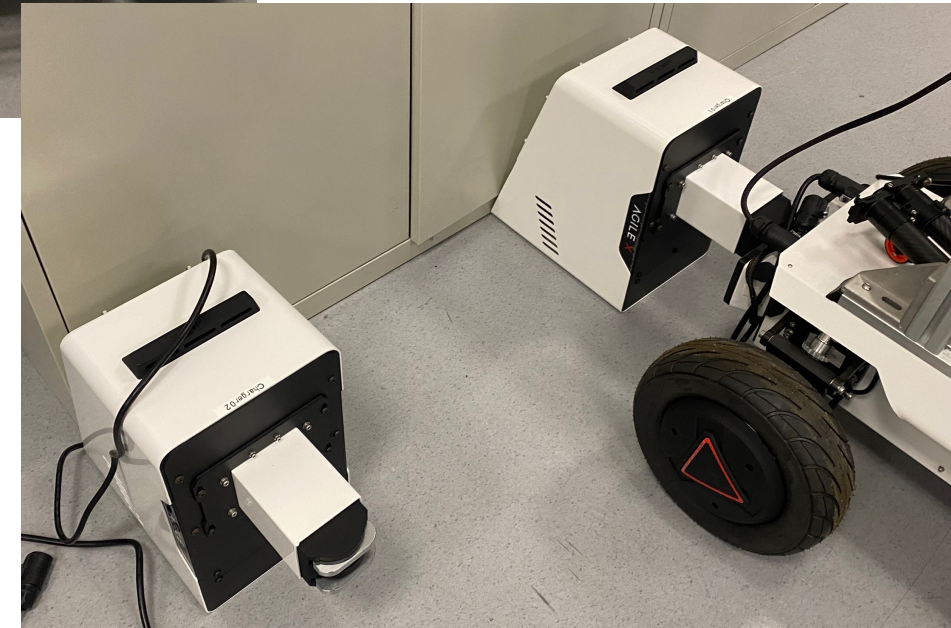
- Grading scheme is not 100% fixed
- Approximately:
 - Lecture: 50%
 - Quizzes during lecture (reading assignments): 4%
 - Homework: 18%
 - Midterm: 8%
 - Final: 20%
 - Project: 50%
 - Paper Presentation: 5%
 - Project Proposal: 5%
 - Intermediate Report: 5%
 - Weekly project meetings: 10%
 - Final Report: 10%
 - Final Demo: 10%
 - Final Webpage: 5%

Campus Autonomy: Hunter SE self charging

- Ackermann (car) steering robot
- 1. locate QR code with omni-camera
- 2. local 2D map from 3D LRF scanner
- 3. plan path to connect to charging port
- 4. execute path

- Difficulty: medium
- Require reliable performance and good code!

- Graduate Supervisor: Xie Fujing



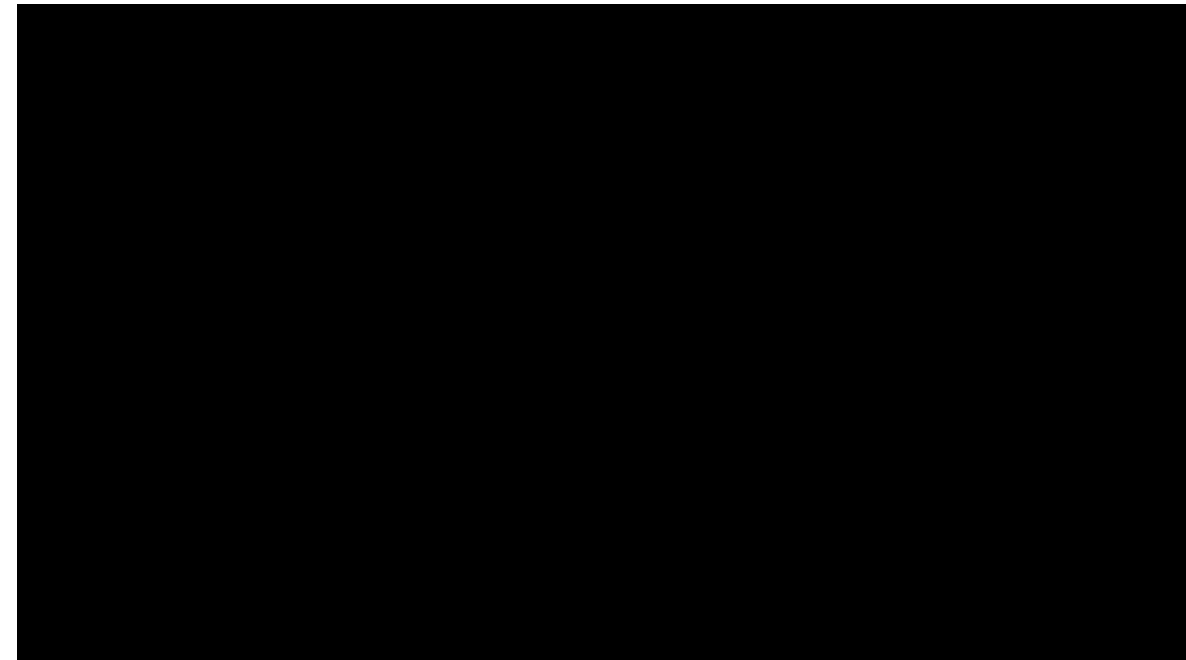
Campus Autonomy: High Speed Navigation

- Use ROS `move_base` and TEB planner for high speed robot control.
 - Include robot dynamics (mass, acceleration, ...)
 - Use 3D LRF to detect and predict motion of obstacles (open source software available)
 - High-speed navigation through light crowds of students.
-
- Difficulty: medium
 - Requite: good demo
 - Graduate Supervisor: Chengqian Li



Campus Autonomy: Human Robot Interaction via Laser-painting

- Humans cannot predict path of robot:
 - No rails; no non-verbal communication
- Fix: project planned path & other info onto the street!
- Tasks: Program Lasercube (or Chinese alternative) to project path onto street
- Do and evaluate actual user studies:
 - Evaluate speed and smoothness & # of near accidents with different versions of displays and no display
 - Quiz participants which interaction they felt most safe with
- Difficulty: Medium
- Graduate Supervisor: Ziwen Zhuang



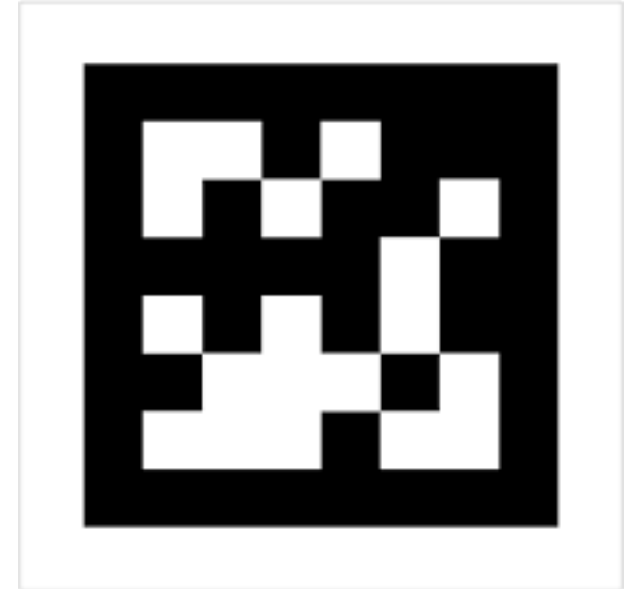
User-friendly Robot Controller: Steamdeck

- Steamdeck:
 - Linux system with touchscreen, joysticks, WiFi, USB-C, etc.
 - Can run ROS and rviz visualization
- Tasks:
 - Install ROS and rviz
 - Program **user-friendly** GUI for Campus Autonomy and Mapping Robot
 - Include 5G teleoperation capabilities
 - Include support for Console control via ssh & keyboard
- Difficulty: Medium +
- Program some GUI with Qt
- Graduate supervisor: Ziwen Zhuang



SLAM Evaluation via AprilTags

- Gather ground truth location for Mapping Robot Project
- Print very big AprilTags – and distribute in scenario (e.g. underground parking)
- Use Faro 3D scanner to (semi-) automatically detect and locate AprilTags
- Write a small program to detect AprilTags in the sensor data
- (If observed with more than one camera, minimize error)
- Generate ground truth trajectories with this
- Difficulty: Medium
- Graduate Supervisor: Ziwen Zhuang



eFMT Ceiling Visual Odometry

- Visual Odometry & Localization of robot indoors with up-looking camera and eFMT algorithm
- 2 Student Project
- Do proper scientific experiment & evaluation & comparison with other approaches
- Difficulty: coding: easy; eval: advanced
- Graduate Supervisor: Chenqian Li

Rethinking the Fourier-Mellin Transform:
Multiple Depths in the Camera's View

Qingwen Xu, Haofei Kuang, Laurent Kneip, Sören Schwertfeger
School of Information Science and Technology (SIST), ShanghaiTech University

Fetch Robot: Whiteboard Cleaning

- Use the Fetch Robot to clean the whiteboard
 - Actual application: cleaning surfaces in hospitals
- Use 2D LRF in base for navigation along wall-mounted whiteboard
- Use RGB-D camera in head to scan board
- Clean whole board and clean just the dirty places
 - Plan arm motion accordingly
- Difficulty: Advanced
- Supervisor: Yinjie Li



Strawberry Picking Robot

- Build upon the work of the Cotton Project => whole pipeline works already
- Modifications:
 - 3D print tool: scissors to cut stem + big spoon to catch strawberry
 - Change computer vision from cotton detection to strawberry detection
 - Integrate changes
- Test on plastic & real strawberries
- Difficulty: Advanced
- Graduate Supervisor: Xie Fujung

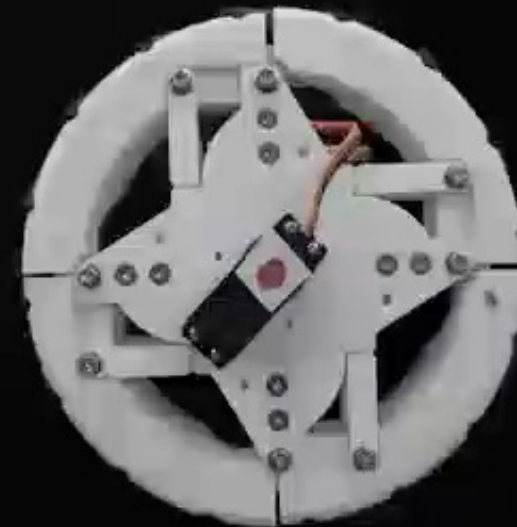


- **Allow 4 students:**
 - 2x Gripper design & construction
 - 1x Computer vision
 - 1x Autonomy & Planning
 - All: Integration

OmniWheg Robot

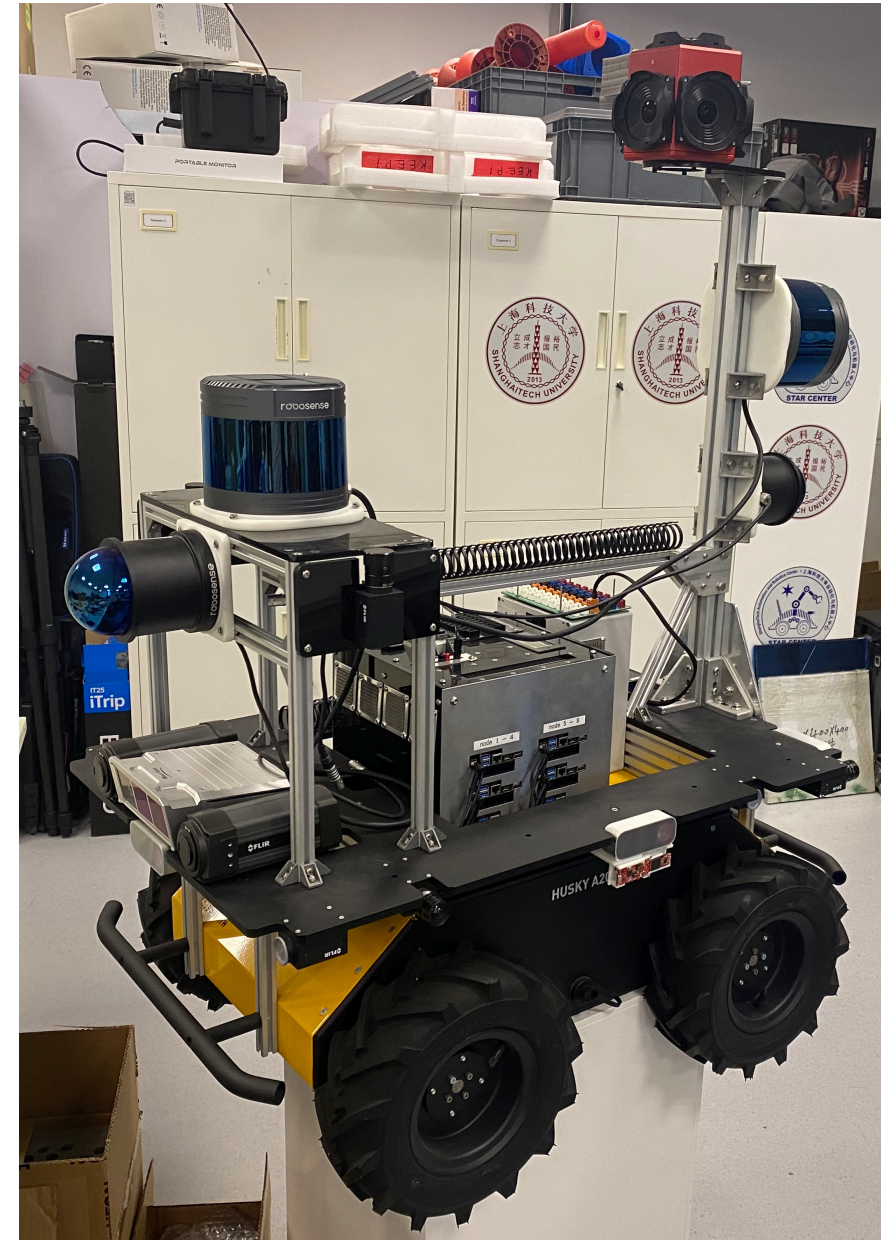
- We have new version of this robot with stronger motors
 - Help with upgrading power supply system
 - Stair detection with depth camera
 - Automatic stair climbing
 - Help with experiments & evaluation
-
- Difficulty: Medium
 - Graduate Supervisor: Gu Jun

Open wheel



Calibration for Mapping Robot

- The new mapping robot has many different sensors that need to be extrinsically calibrated (find pose of each sensor w.r.t. the robot).
- Tasks:
 - Calibrate “exotic” sensors like radar & ultrasound
 - Write a nice script/ GUI to use data from ROS bags for calibration with the available calibration tools
 - Calibrate the robot
- Difficulty: Advanced
- If work is very good: become co-author of our Mapping Robot Journal paper
- Graduate Supervisor: Jiang Wenqing

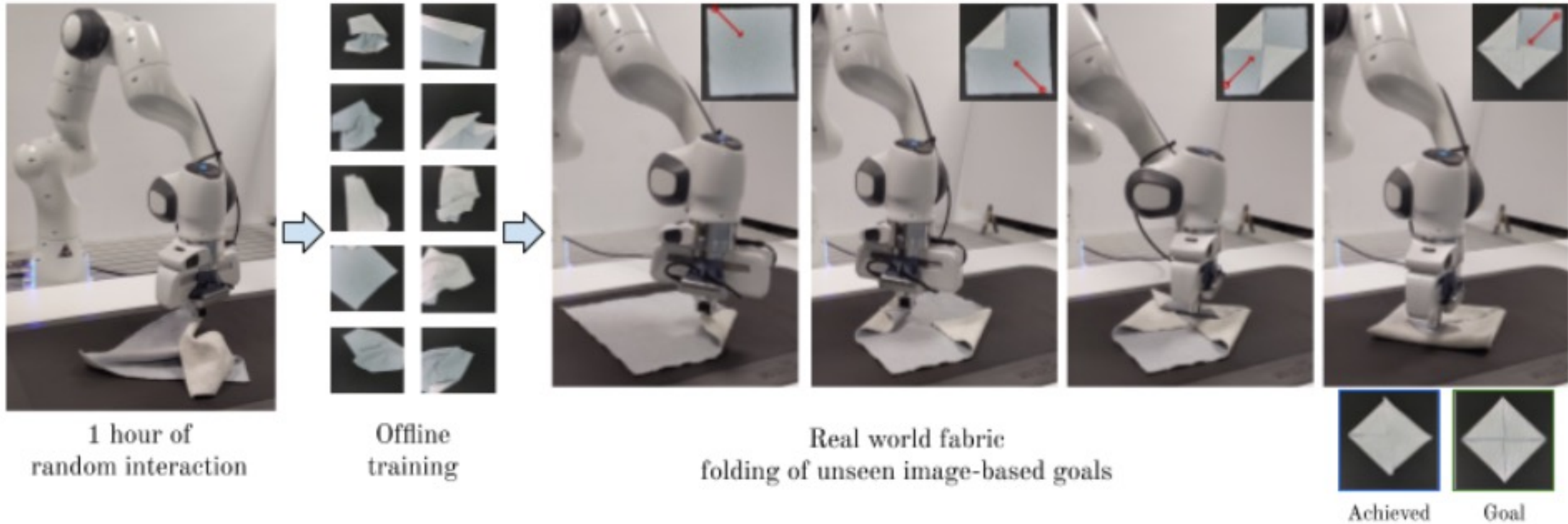


Car Project

- Main purpose:
Create nice demo for visitors:
 - Install screens in car
 - Re-Install sensor frame on Car
 - Nice visualization of sensor data
-
- Work together with the graduate students developing the algorithms (Groups of Prof. Laurent Kneip and Prof. Sören Schwertfeger) for demos
-
- Difficulty: Medium
 - Graduate Supervisor: Yinjie Li



Learn to fold by offline reinforcement learning



- Difficulty: Advanced
- Graduate Supervisor: Ziwen Zhuang

- Max one group per topic!
- In case of double selection we will discuss alternatives with both groups
- If no one changes it, it will be “First come - First Serve”

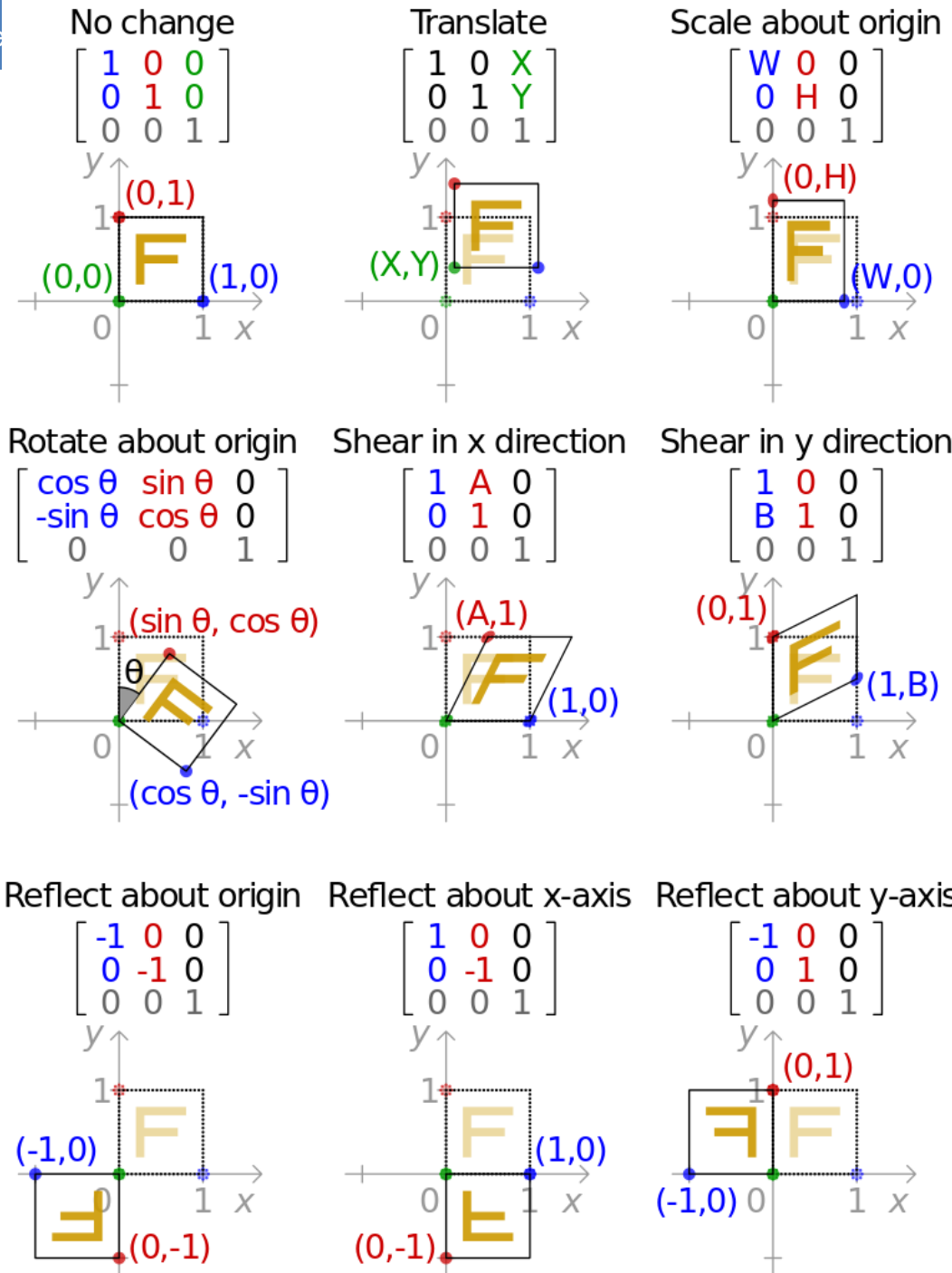
					Difficulty:
Name	Advisor	Hardware	Software	Algorithm	
Campus Autonomy: Hunter SE self charging	Xie Fujing	low	medium	medium	
Campus Autonomy: High Speed Navigation	Chengqian Li	low	low	medium +	
Campus Autonomy: HRI via Laser-painting	Ziwen Zhuang	medium	medium	low	
User-friendly Robot Controller: Steamdeck	Ziwen Zhuang	low	medium +	low	
SLAM Evaluation via AprilTags	Ziwen Zhuang	low	medium	medium	
eFMT Ceiling Visual Odometry	Chengqian Li	low	medium	advanced	
Fetch Robot: Whiteboard Cleaning	Yinjie Li	medium	advanced	medium	
Strawberry Picking Robot	Xie Fujing	advanced	advanced	medium	
OmniWheg Robot	Gu Jun	advanced	low	medium	
Calibration for Mapping Robot	Jiang Wenqing	low	advanced	advanced	
Car Project	Yinjie Li	medium	medium	low	
Learn to Fold	Ziwen Zhuang	low	medium	advanced	

TRANSFORMS & STUFF 😊

Affine Transformation

- Function between affine spaces. Preserves:
 - points,
 - straight lines
 - planes
 - sets of parallel lines remain parallel
- Allows:
 - Interesting for Robotics: translation, rotation, (scaling), and chaining of those
 - Not so interesting for Robotics: reflection, shearing, homothetic transforms

- Rotation and Translation:
$$\begin{bmatrix} \cos \theta & \sin \theta & X \\ -\sin \theta & \cos \theta & Y \\ 0 & 0 & 1 \end{bmatrix}$$



Math: Rigid Transformation

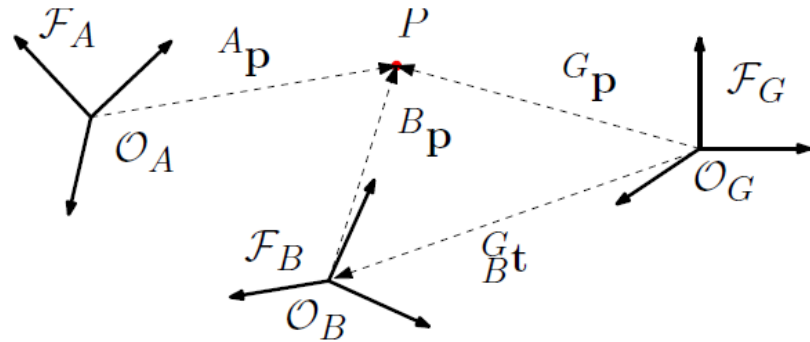
- Geometric transformation that preserves Euclidean distance between pairs of points.
- Includes reflections (i.e. change from right-hand to left-hand coordinate system and back)
- Just rotation & translation: rigid motions or proper rigid transformations:
 - Decomposed to rotation and translation
 - => subset of Affine Transformations
- In Robotics: Just use term **Transform** or **Transformation** for rigid motions (without reflections)

Lie groups for transformations

- Smoothly differentiable Group
- No singularities
- Good interpolation
- SO: Special Orthogonal group
- SE: Special Euclidian group
- Sim_ilarity transform group

Group	Description	Dim.	Matrix Representation
SO(3)	3D Rotations	3	3D rotation matrix
SE(3)	3D Rigid transformations	6	Linear transformation on homogeneous 4-vectors
SO(2)	2D Rotations	1	2D rotation matrix
SE(2)	2D Rigid transformations	3	Linear transformation on homogeneous 3-vectors
Sim(3)	3D Similarity transformations (rigid motion + scale)	7	Linear transformation on homogeneous 4-vectors

Transform



Notation	Meaning
$\mathcal{F}_{R[k]}$	Coordinate frame attached to object 'R' (usually the robot) at sample time-instant k .
$O_{R[k]}$	Origin of $\mathcal{F}_{R[k]}$.
${}^R_{R[k]} \mathbf{p}$	For any general point P , the position vector $\overrightarrow{O_{R[k]}P}$ resolved in $\mathcal{F}_{R[k]}$.
${}^H \hat{\mathbf{x}}_R$	The x-axis direction of \mathcal{F}_R resolved in \mathcal{F}_H . Similarly, ${}^H \hat{\mathbf{y}}_R$, ${}^H \hat{\mathbf{z}}_R$ can be defined. Obviously, ${}^R \hat{\mathbf{x}}_R = \hat{\mathbf{e}}_1$. Time indices can be added to the frames, if necessary.
${}^{R[k]}_{S[k']} \mathbf{R}$	The rotation-matrix of $\mathcal{F}_{S[k']}$ with respect to $\mathcal{F}_{R[k]}$.
${}^R_S \mathbf{t}$	The translation vector $\overrightarrow{O_R O_S}$ resolved in \mathcal{F}_R .

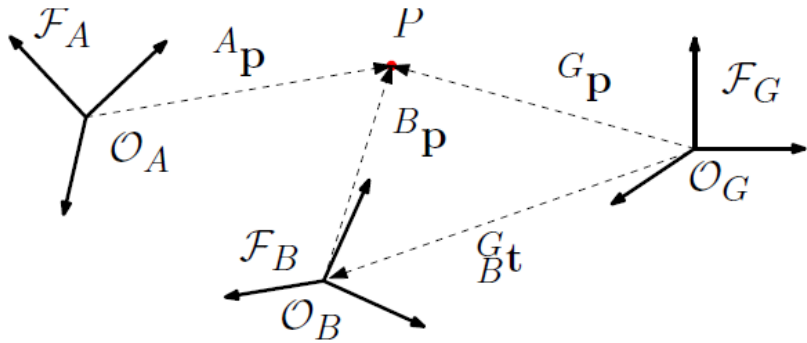
Transform
between two
coordinate frames

$${}^G_A \mathbf{t} \triangleq \overrightarrow{O_G O_A} \text{ resolved in } \mathcal{F}_G \quad \begin{pmatrix} {}^G \mathbf{p} \\ 1 \end{pmatrix} \equiv \begin{pmatrix} {}^G_A \mathbf{R} & {}^G_A \mathbf{t} \\ \mathbf{0}_{1 \times [2,3]} & 1 \end{pmatrix} \begin{pmatrix} {}^A \mathbf{p} \\ 1 \end{pmatrix} \quad {}^G_A \mathbf{T} \equiv \left\{ \begin{matrix} {}^G_A \mathbf{t} \\ {}^G_A \mathbf{R} \end{matrix} \right\}$$

$$\begin{aligned} {}^G \mathbf{p} &= {}^G_A \mathbf{R} \, {}^A \mathbf{p} + {}^G_A \mathbf{t} \\ &\triangleq {}^G_A \mathbf{T} ({}^A \mathbf{p}). \end{aligned}$$

$$\begin{bmatrix} \cos \theta & -\sin \theta & {}^G_A t_x \\ \sin \theta & \cos \theta & {}^G_A t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Transform: Operations



Transform between two coordinate frames (chaining, compounding):

$${}^G\mathbf{T} = {}^G\mathbf{T} {}^A\mathbf{T} \equiv \begin{Bmatrix} {}^G\mathbf{R}^A \mathbf{t} + {}^G\mathbf{t} \\ {}^G\mathbf{R}^A \end{Bmatrix}$$

Inverse of a Transform :

$${}^B\mathbf{T} = {}^A\mathbf{T}^{-1} \equiv \begin{Bmatrix} -{}^A\mathbf{R}^B \mathbf{t} \\ {}^A\mathbf{R}^B \end{Bmatrix}$$

Relative (Difference) Transform : ${}^B\mathbf{T} = {}^G\mathbf{T}^{-1} {}^G\mathbf{T}$

See: **Quick Reference to Geometric Transforms in Robotics** by Kaustubh Pathak on the webpage!

Chaining :
$${}_{R[X+1]}^G \mathbf{T} = {}_{R[X]}^G \mathbf{T} {}_{R[X+1]}^{R[X]} \mathbf{T} \equiv \begin{Bmatrix} {}_{R[X]}^G \mathbf{R} & {}_{R[X+1]}^{R[X]} \mathbf{t} + {}_{R[X]}^G \mathbf{t} \\ {}_{R[X]}^G \mathbf{R} & {}_{R[X+1]}^{R[X]} \mathbf{R} \end{Bmatrix} = \begin{Bmatrix} {}_{R[X+1]}^G \mathbf{t} \\ {}_{R[X+1]}^G \mathbf{R} \end{Bmatrix}$$

In 2D Translation:
$$\begin{bmatrix} {}_{R[X+1]}^G t_x \\ {}_{R[X+1]}^G t_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos {}_{R[X]}^G \theta & -\sin {}_{R[X]}^G \theta & {}_{R[X]}^G t_x \\ \sin {}_{R[X]}^G \theta & \cos {}_{R[X]}^G \theta & {}_{R[X]}^G t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}_{R[X+1]}^{R[X]} t_x \\ {}_{R[X+1]}^{R[X]} t_y \\ 1 \end{bmatrix}$$

In 2D Rotation:

$${}_{R[X+1]}^G \mathbf{R} = \begin{bmatrix} \cos {}_{R[X+1]}^G \theta & -\sin {}_{R[X+1]}^G \theta \\ \sin {}_{R[X+1]}^G \theta & \cos {}_{R[X+1]}^G \theta \end{bmatrix} = \begin{bmatrix} \cos {}_{R[X]}^G \theta & -\sin {}_{R[X]}^G \theta \\ \sin {}_{R[X]}^G \theta & \cos {}_{R[X]}^G \theta \end{bmatrix} \begin{bmatrix} \cos {}_{R[X+1]}^{R[X]} \theta & -\sin {}_{R[X+1]}^{R[X]} \theta \\ \sin {}_{R[X+1]}^{R[X]} \theta & \cos {}_{R[X+1]}^{R[X]} \theta \end{bmatrix}$$

In 2D Rotation (simple):
$${}_{R[X+1]}^G \theta = {}_{R[X]}^G \theta + {}_{R[X+1]}^{R[X]} \theta$$

In ROS: nav_2d_msgs/Pose2DStamped

- First Message at time 97 : G
- Message at time 103 : X
- Next Message at time 107 : X+1

$$R_{[X+1]}^G \mathbf{T} = R_{[X]}^G \mathbf{T} R_{[X+1]}^{R[X]} \mathbf{T}$$

$$R_{[X]}^G \mathbf{T}$$

$$R_{[X+1]}^{R[X]} \mathbf{T}$$

$$R_{[X+1]}^G \mathbf{T}$$

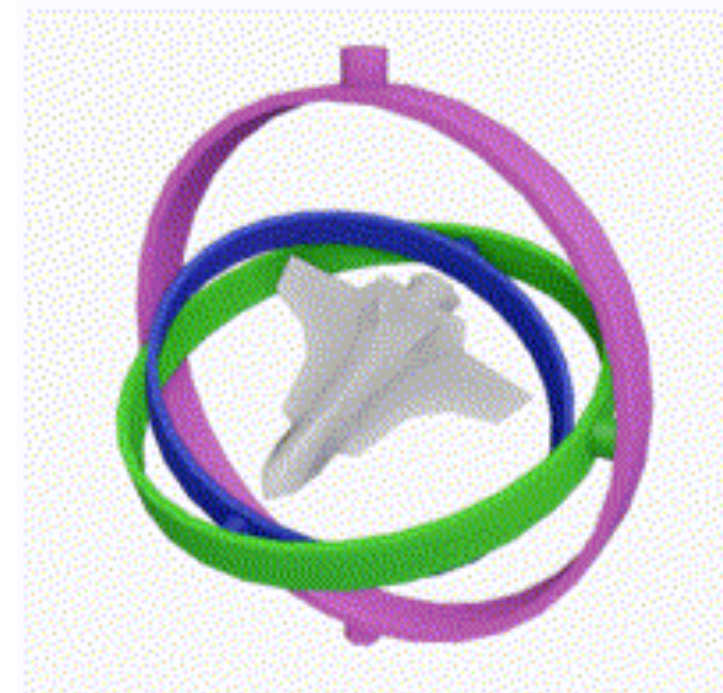
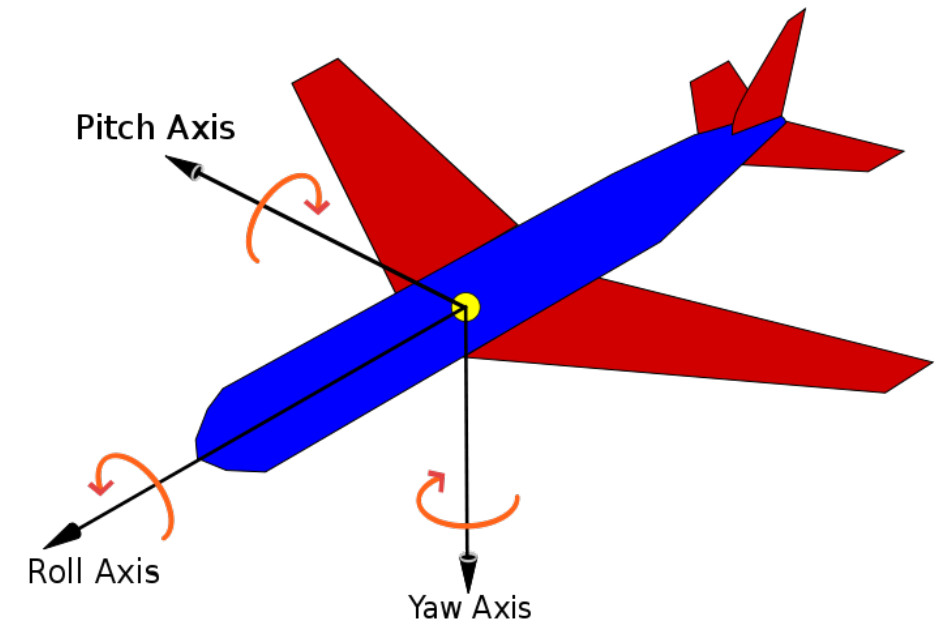
```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose2D pose2D
  float64 x
  float64 y
  float64 theta
```

3D Rotation

- Many 3D rotation representations:

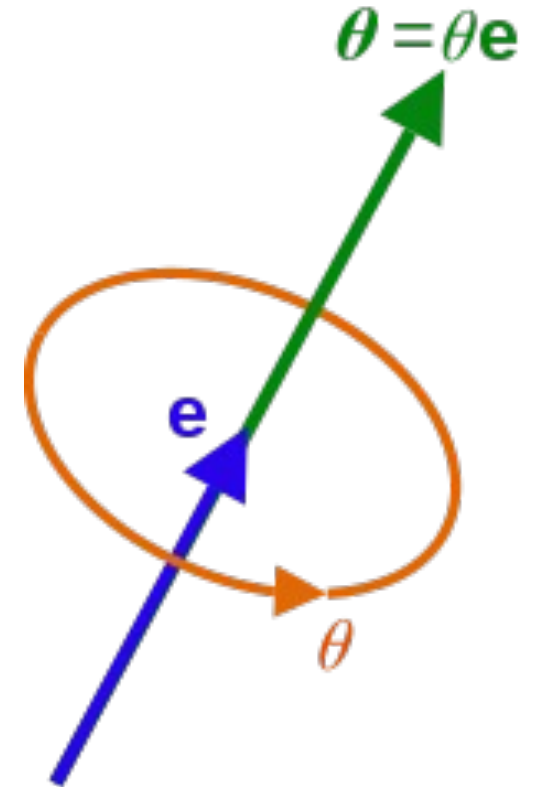
https://en.wikipedia.org/wiki/Rotation_formalisms_in_three_dimensions

- Euler angles:
 - Roll: rotation around x-axis
 - Pitch: rotation around y-axis
 - Yaw: rotation around z-axis
 - Apply rotations one after the other...
 - => Order important! E.g.:
 - x-z-x; x-y-z; z-y-x; ...
 - ☹ Singularities
 - Gimbal lock in Engineering
 - "a condition caused by the collinear alignment of two or more robot axes resulting in unpredictable robot motion and velocities"



3D Rotation

- Axis Angle
 - Angle θ and
 - Axis unit vector \mathbf{e} (3D vector with length 1)
 - Can be represented with 2 numbers (e.g. elevation and azimuth angles)
- Euler Angles: sequence of 3 rotations around coordinate axes
equivalent to:
- Axis Angle: pure rotation around a single fixed axis



3D Rotation

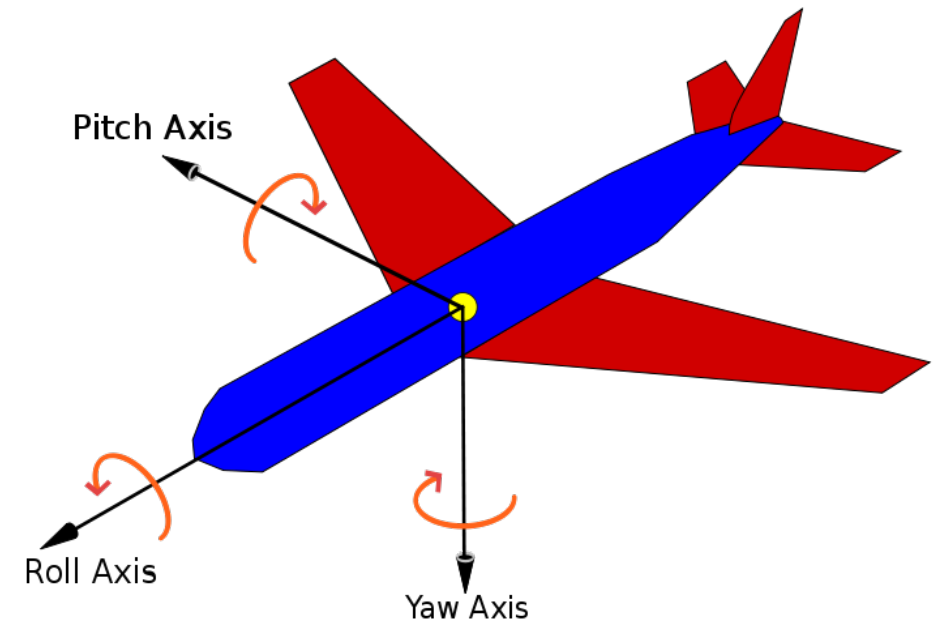
- Quaternions:
 - Concatenating rotations is computationally faster and numerically more stable
 - Extracting the angle and axis of rotation is simpler
 - Interpolation is more straightforward
 - Unit Quaternion: norm = 1

- Versor: <https://en.wikipedia.org/wiki/Versor>

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

- Scalar (real) part: q_0 , sometimes q_w
- Vector (imaginary) part: \mathbf{q}
- Over determined: 4 variables for 3 DoF (but: unit!)

- Check out: <https://eater.net/quaternions> !
Excellent interactive video...



$$\check{\mathbf{p}} \equiv p_0 + p_x \mathbf{i} + p_y \mathbf{j} + p_z \mathbf{k}$$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -\mathbf{1}$$

$$\check{\mathbf{q}} = (q_0 \quad q_x \quad q_y \quad q_z)^T \equiv \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix}$$

Transform in 3D

$${}^G\mathbf{T}_A = \begin{matrix} & \text{Matrix} & \text{Euler} & \text{Quaternion} \\ = & \begin{bmatrix} {}^G\mathbf{R}_A & {}^G\mathbf{t}_A \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} & \begin{pmatrix} {}^G\mathbf{t}_A \\ {}^G\Theta_A \end{pmatrix} & \begin{pmatrix} {}^G\mathbf{t}_A \\ {}^G\check{\mathbf{q}}_A \end{pmatrix} \end{matrix}$$

$${}^G\Theta_A \triangleq (\theta_r, \theta_p, \theta_y)^T$$

In ROS: Quaternions! (w, x, y, z)
Uses Eigen library for Transforms

Rotation Matrix 3x3

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_z(\alpha) R_y(\beta) R_x(\gamma)$$

$$\text{yaw} = \alpha, \text{pitch} = \beta, \text{roll} = \gamma$$

Eigen

- Don't have to deal with the details of transforms too much 😊
- Conversions between ROS and Eigen:
http://docs.ros.org/noetic/api/eigen_conversions/html/namespace_tf.html

```
Matrix3f m;
m = AngleAxisf(angle1, Vector3f::UnitZ())
    * AngleAxisf(angle2, Vector3f::UnitY())
    * AngleAxisf(angle3, Vector3f::UnitZ());
```

https://eigen.tuxfamily.org/dox/group_Geometry_Module.html

class	Eigen::AlignedBox	An axis aligned box. More...
class	Eigen::AngleAxis	Represents a 3D rotation as a rotation angle around an arbitrary 3D axis. More...
class	Eigen::Homogeneous	Expression of one (or a set of) homogeneous vector(s) More...
class	Eigen::Hyperplane	A hyperplane. More...
class	Eigen::Map< const Quaternion< _Scalar >, _Options > Quaternion	expression mapping a constant memory buffer. More...
class	Eigen::Map< Quaternion< _Scalar >, _Options >	Expression of a quaternion from a memory buffer. More...
class	Eigen::ParametrizedLine	A parametrized line. More...
class	Eigen::Quaternion	The quaternion class used to represent 3D orientations and rotations. More...
class	Eigen::QuaternionBase	Base class for quaternion expressions. More...
class	Eigen::Rotation2D	Represents a rotation/orientation in a 2 dimensional space. More...
class	Scaling	Represents a generic uniform scaling transformation. More...
class	Eigen::Transform	Represents an homogeneous transformation in a N dimensional space. More...
class	Eigen::Translation	Represents a translation transformation. More...

Examples of Transforms

- Transform between global coordinate frame and robot frame at time X
- Transform between robot frame at time X and robot frame at time $X+1$
- Transform between robot camera frame and robot base frame (mounted fixed – not dependend on time! => static transform)
- Transform between map origin and door pose in map (not time dependend)
- Transform between robot camera frame and fingers (end-effector) of a robot arm at time X
- Transform between robot camera frame and map frame at time X
- Transform between robot 1 camera at time X and robot 2 camera at time $X+n$