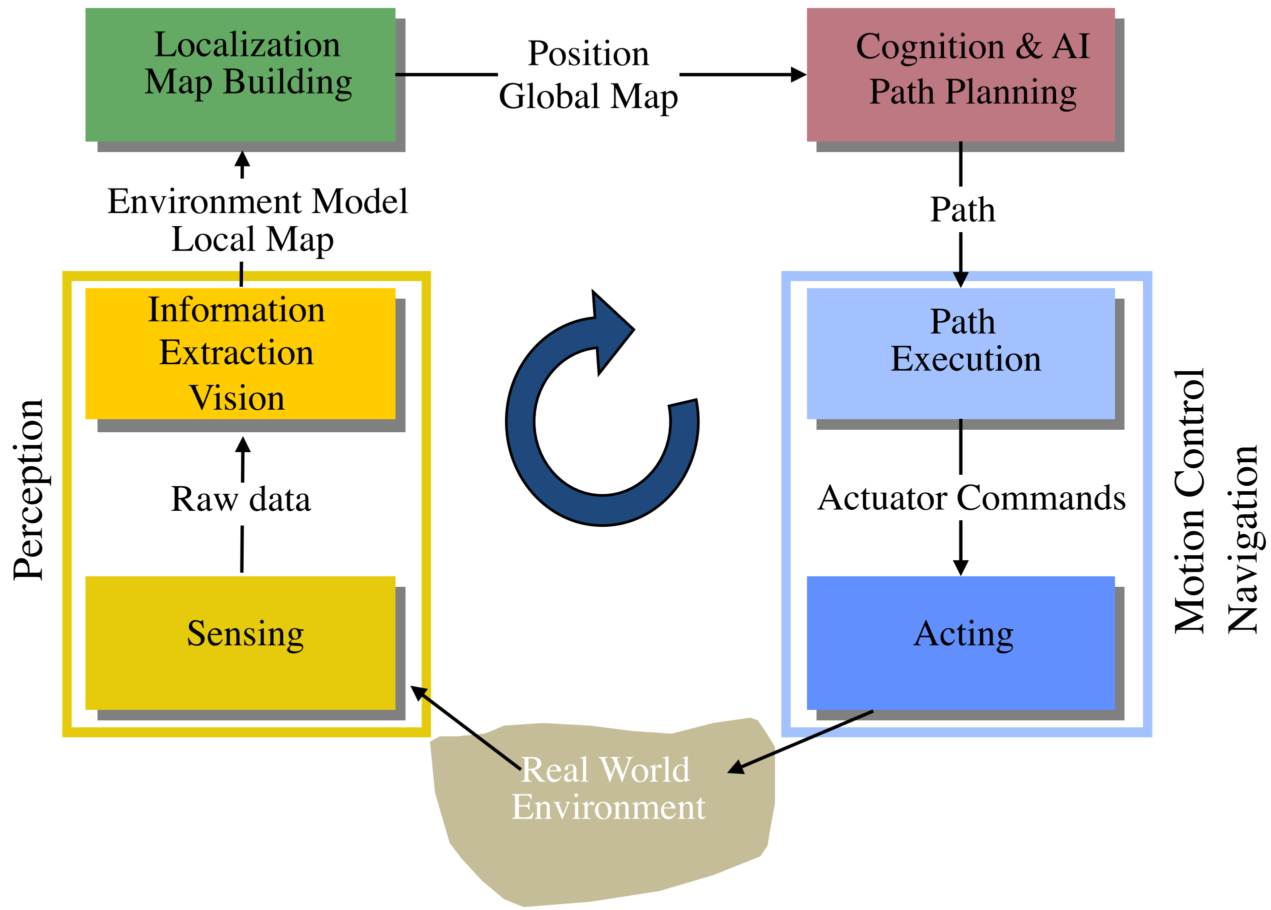
CS283: **Robotics Fall 2020: Kinematics**

Sören Schwertfeger / 师泽仁

ShanghaiTech University

KINEMATICS

General Control Scheme for Mobile Robot Systems



With material from Roland Siegwart and Davide Scaramuzza, ETH Zurich

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

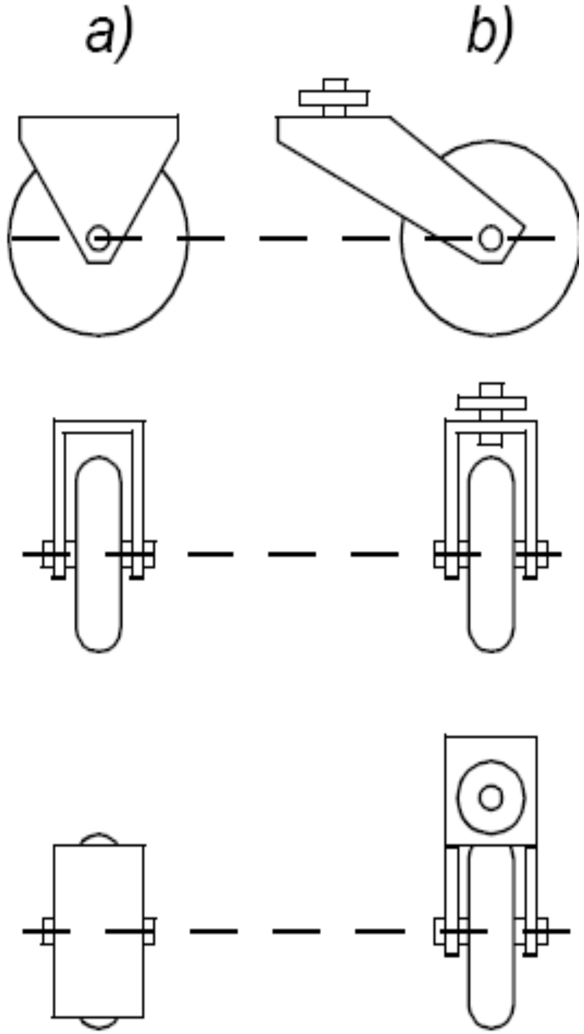
Mobile Robots with Wheels

- 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

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

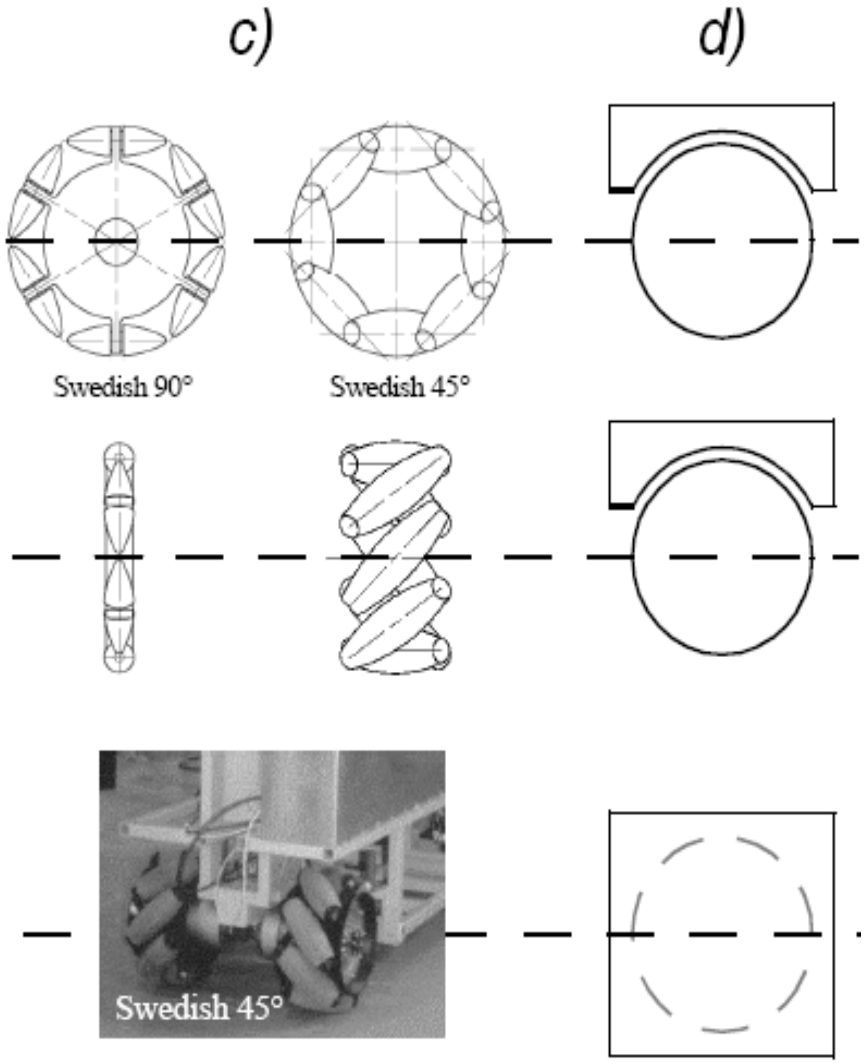
- b) Castor wheel: Three degrees of freedom; rotation around the wheel axle, the contact point and the castor axle



The Four Basic Wheels Types

- 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

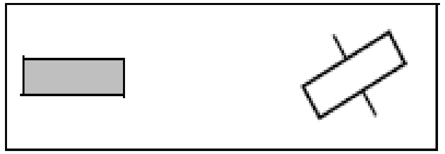


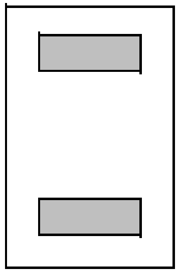
Characteristics of Wheeled Robots and Vehicles

- 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

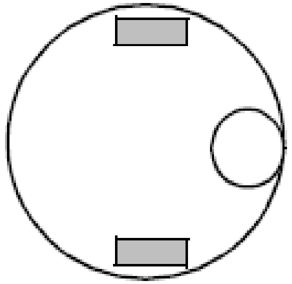
- Two wheels

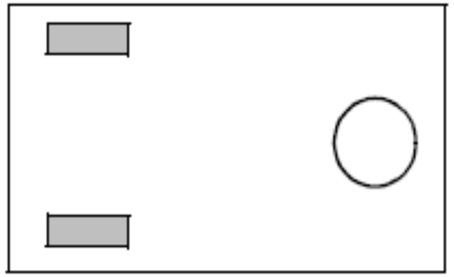


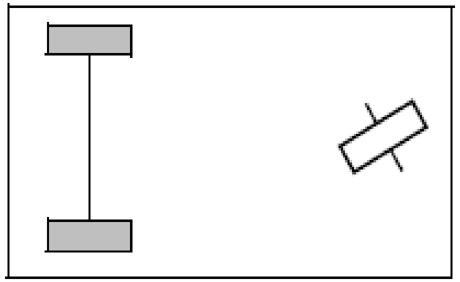


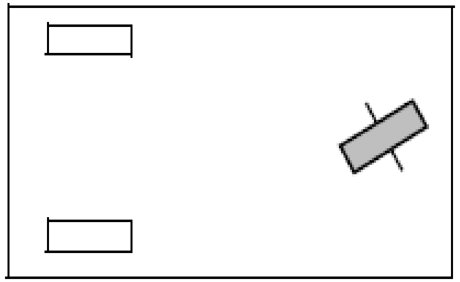
Center of gravity below axle

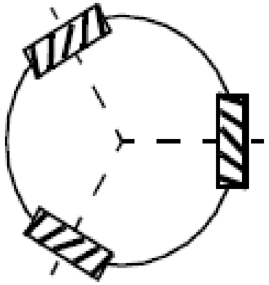
- Three wheels



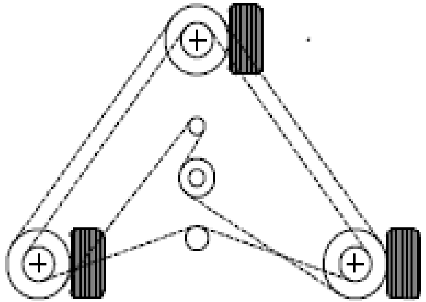








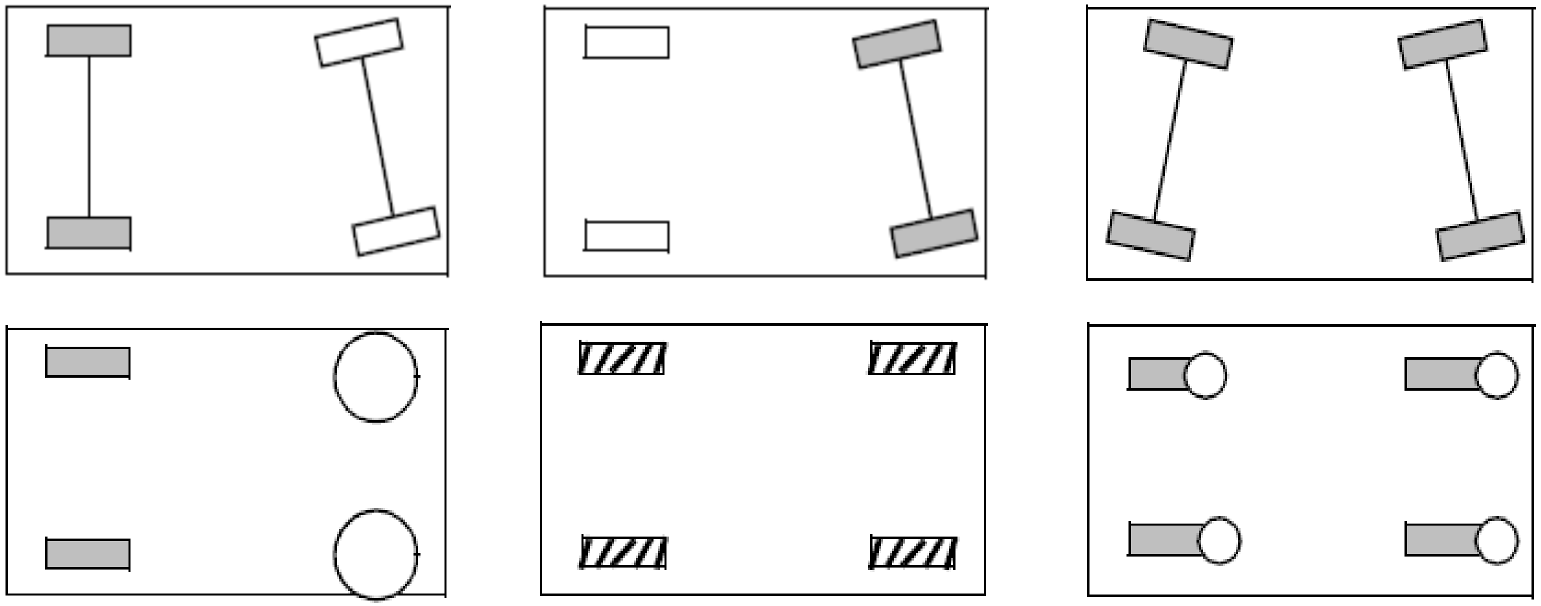
Omnidirectional Drive

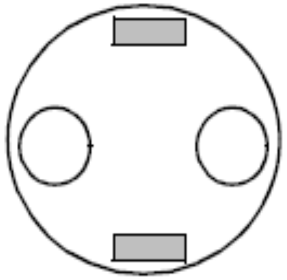


Synchro Drive

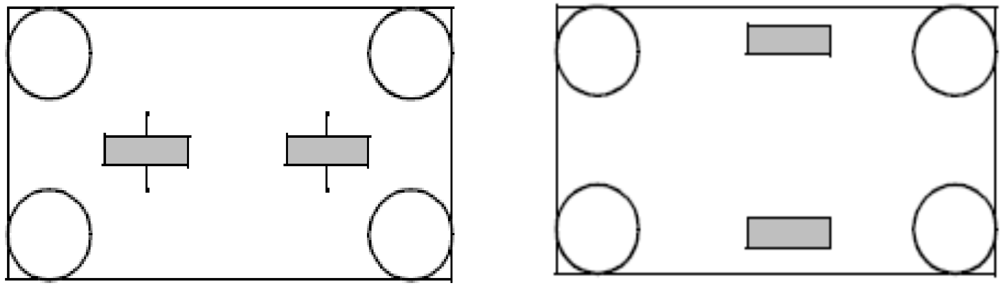
Different Arrangements of Wheels II

- Four wheels



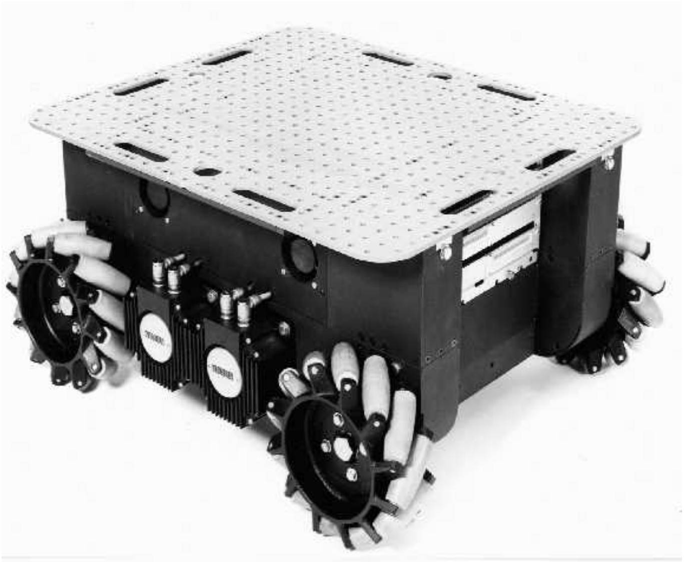


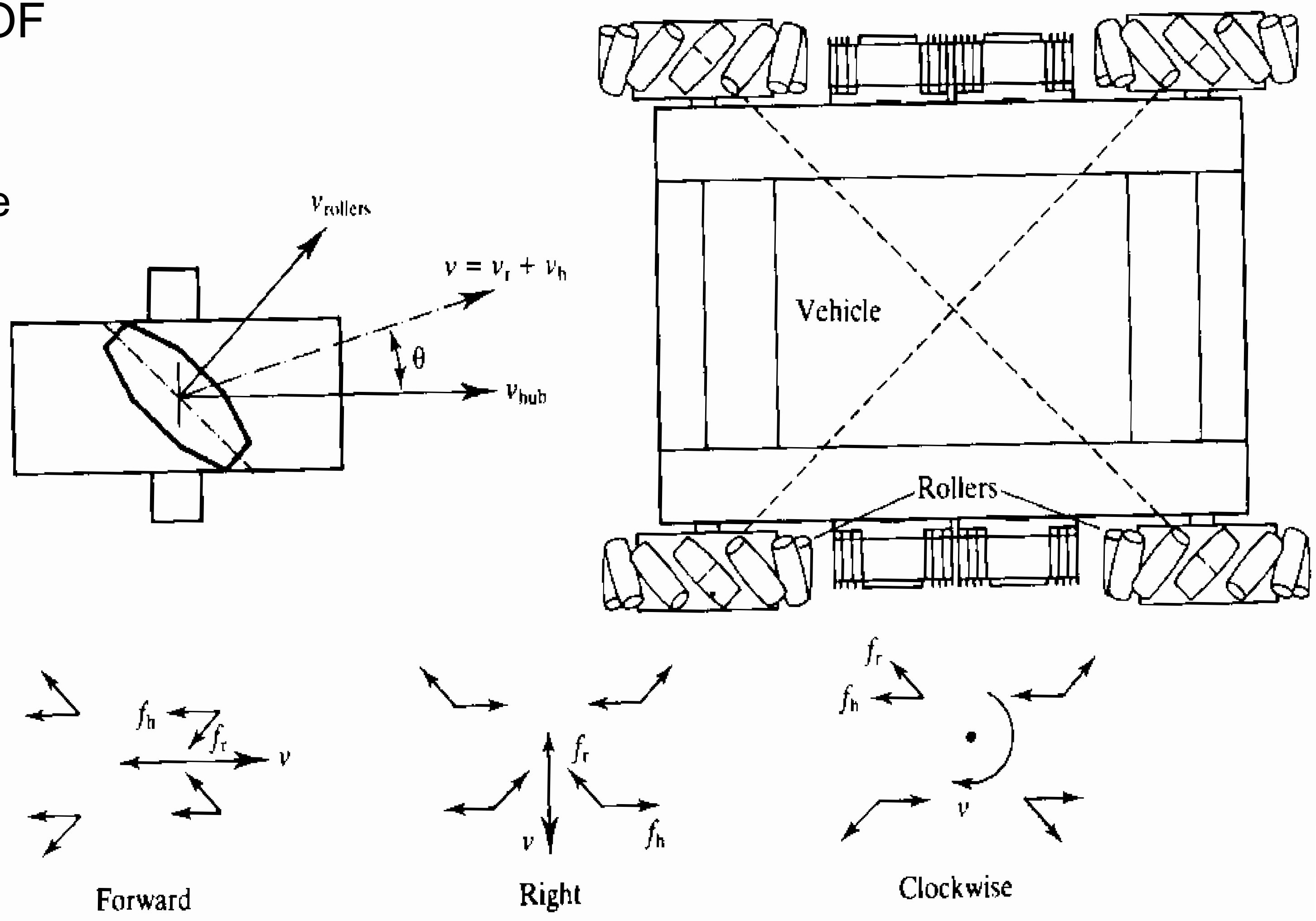
- Six wheels



Uranus, CMU: Omnidirectional Drive with 4 Wheels

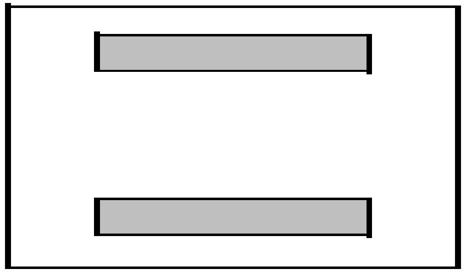
- Movement in the plane has 3 DOF
  - thus only three wheels can be independently controlled
  - It might be better to arrange three swedish wheels in a triangle

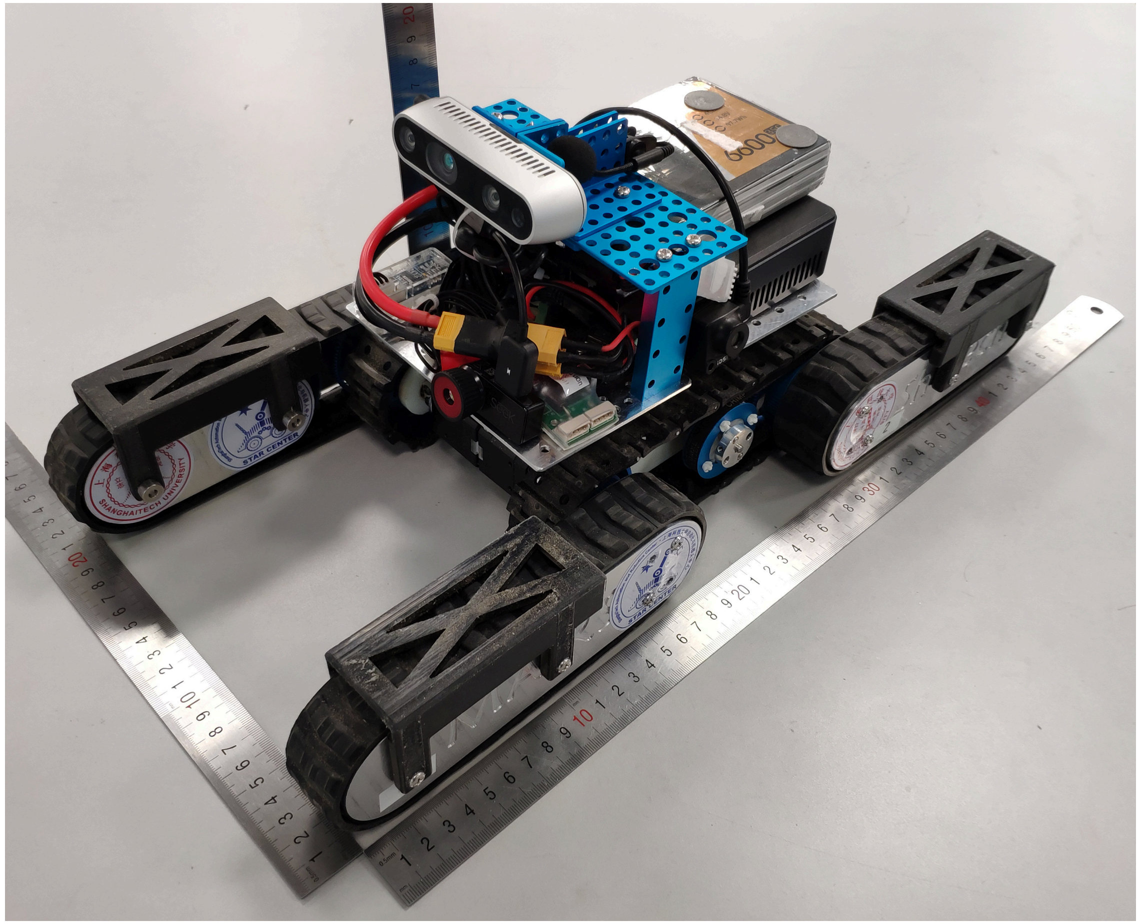




MARS Rescue Robot: Tracked Differential Drive

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

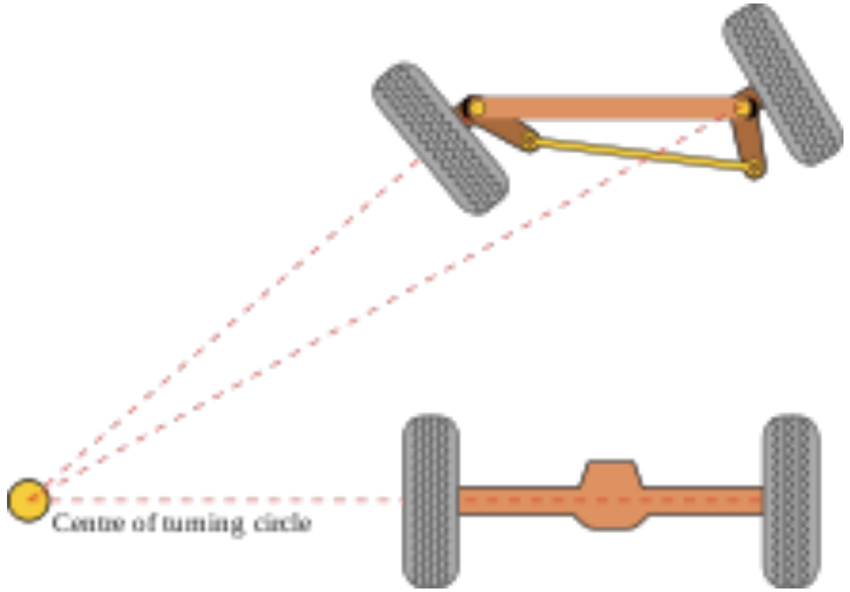




Differential Drive Robots

Ackermann Robot

- No sideways slip than differential drive during turning ☺
- Cannot turn on the spot ☹



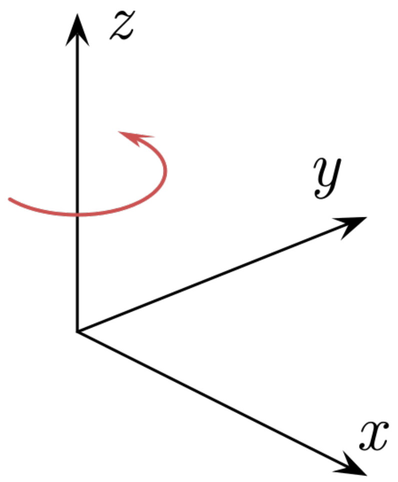
Introduction: Mobile Robot Kinematics

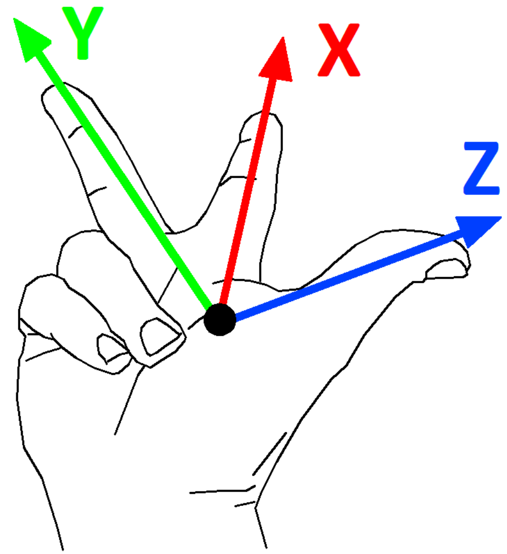
- Aim
  - Description of mechanical behavior of the robot for *design* and *control*
  - Similar to robot manipulator kinematics
  - However, mobile robots can move unbound with respect to its environment
    - there is no direct way to measure the robot’s position
    - Position must be integrated over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

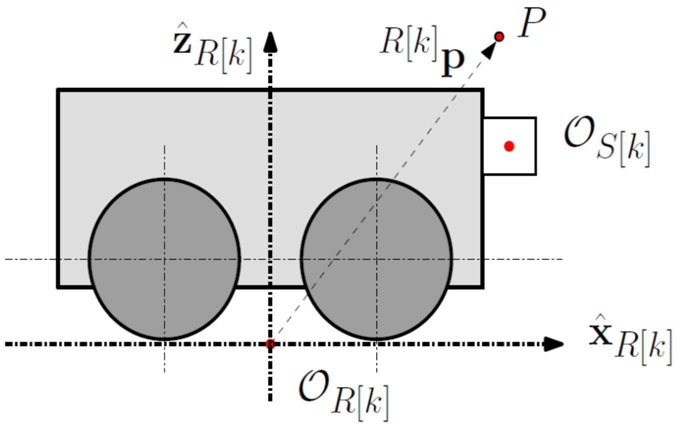
COORDINATE SYSTEM

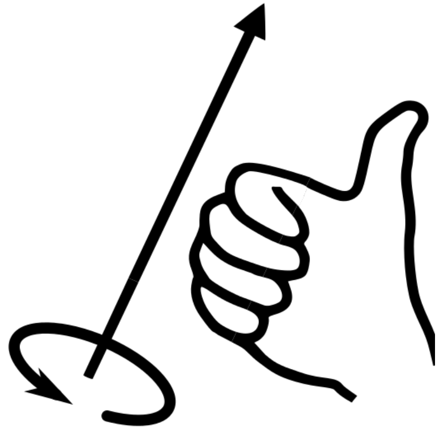
Right Hand Coordinate System

- 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 ???



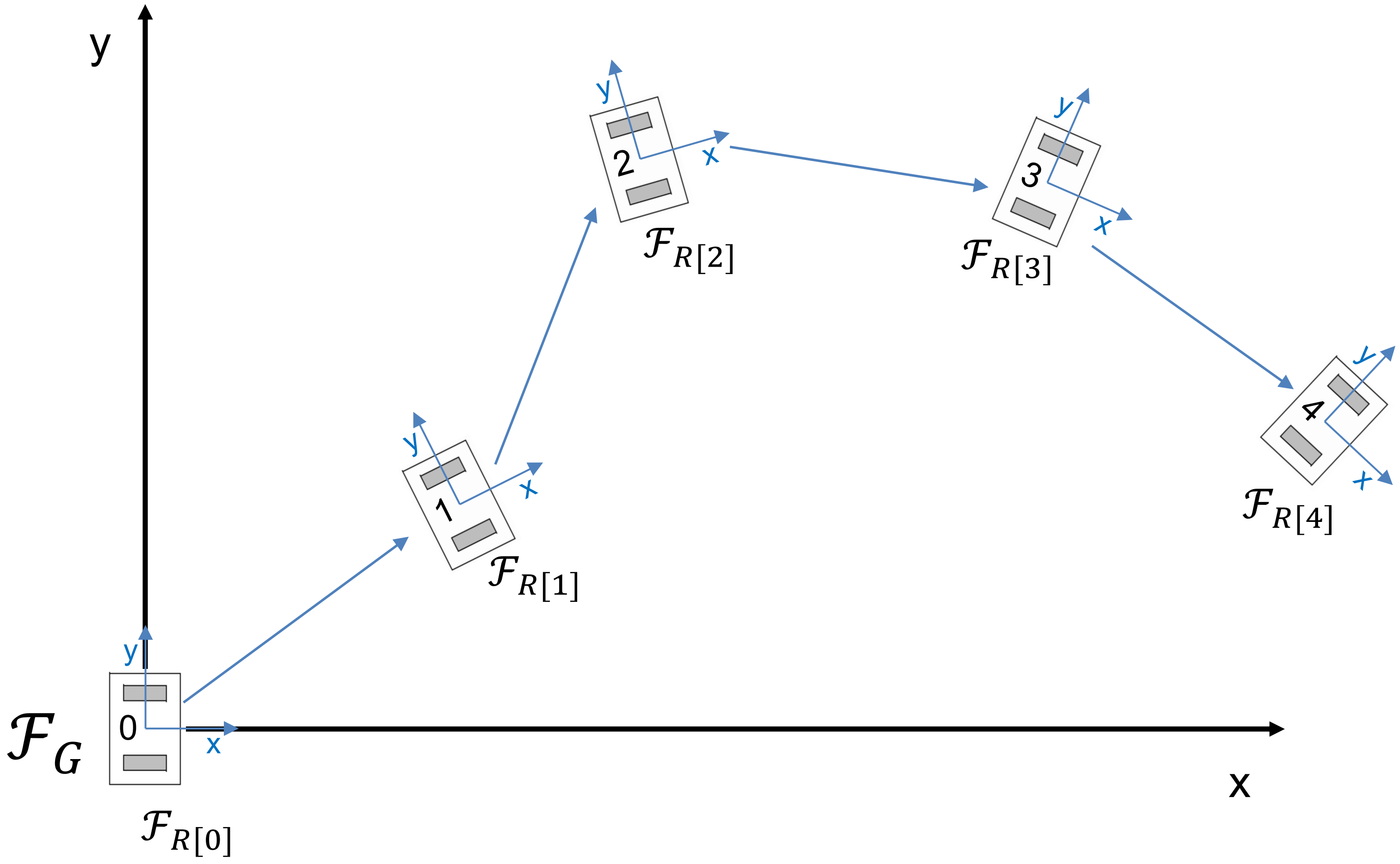






Right Hand Rule http://en.wikipedia.org/wiki/Right-hand_rule

Odometry



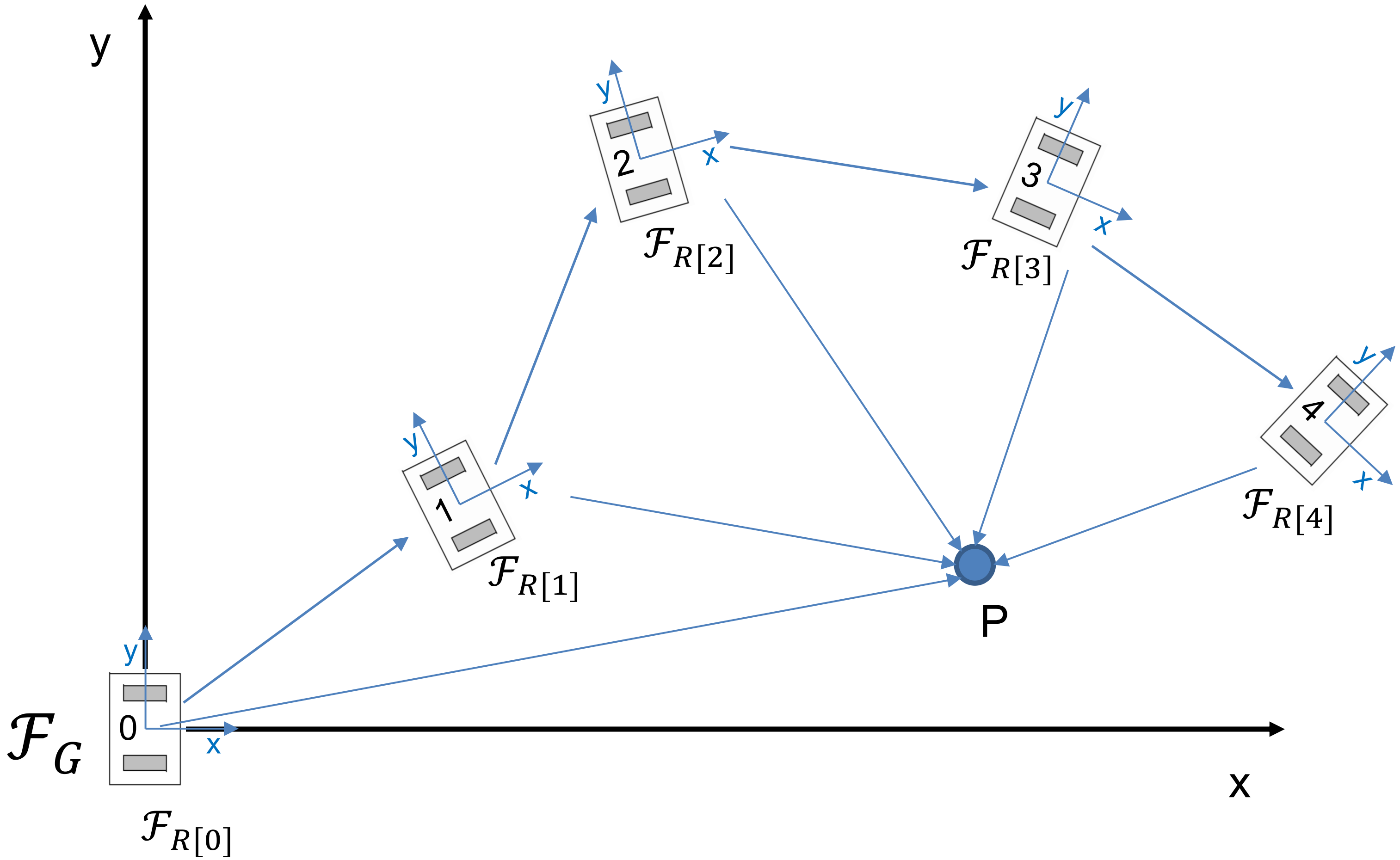
With respect to the robot start pose:
Where is the robot now?

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 $\mathcal{F}_{R[X]}$

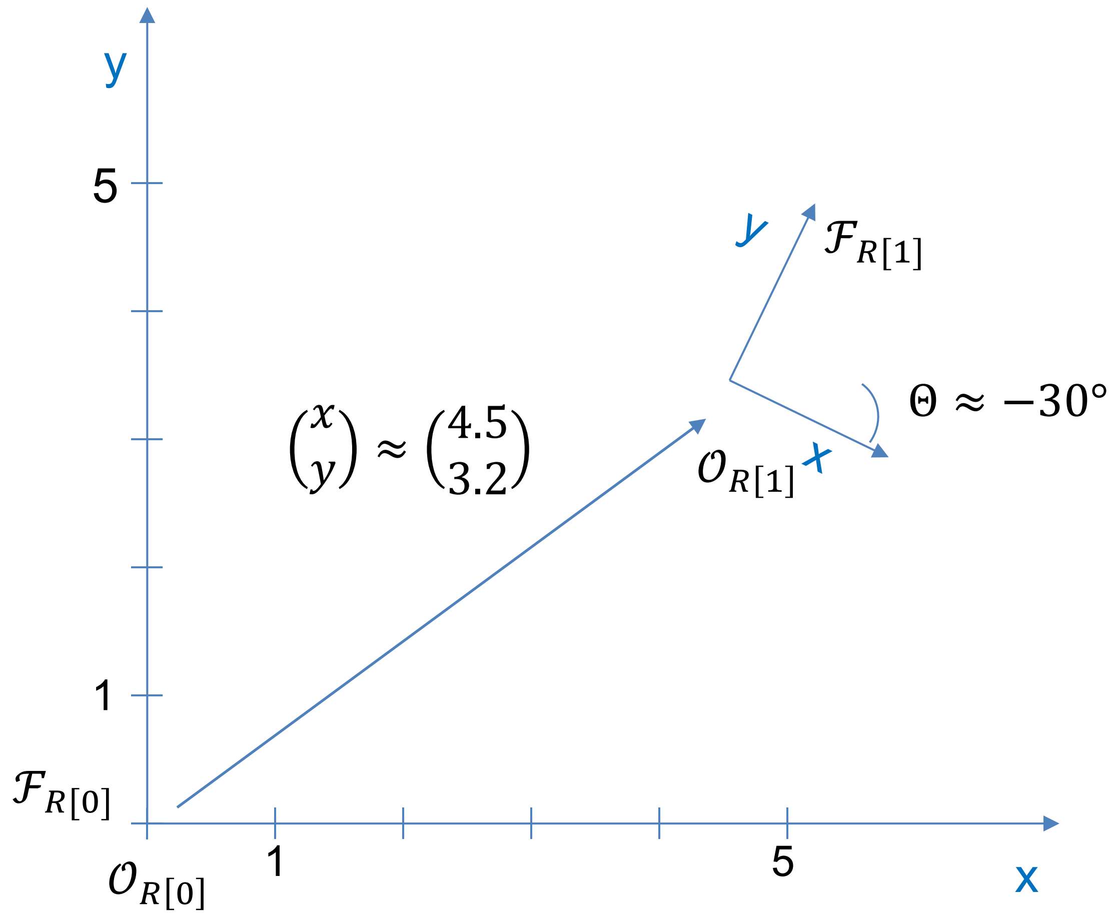


$\mathcal{O}_{R[X]}$: Origin of $\mathcal{F}_{R[X]}$ (coordinates (0, 0))

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

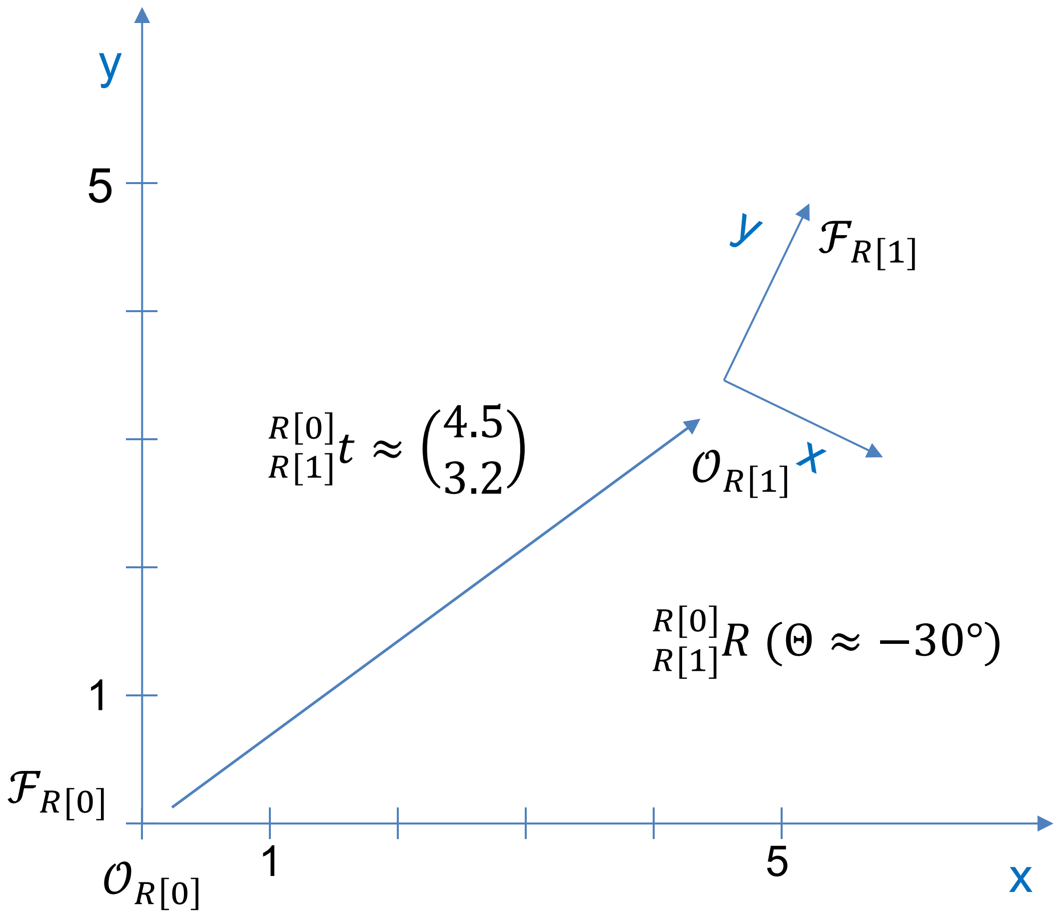
- Object P is observed at times 0 to 4
- Object P is static (does not move)
- The Robot moves (e.g. $\mathcal{F}_{R[0]} \neq \mathcal{F}_{R[1]}$)
- => (x, y) coordinates of P are different in all frames, for example:
  - $\overrightarrow{\mathcal{O}_{R[0]}\mathrm{P}} \neq \overrightarrow{\mathcal{O}_{R[1]}\mathrm{P}}$

Position, Orientation & Pose



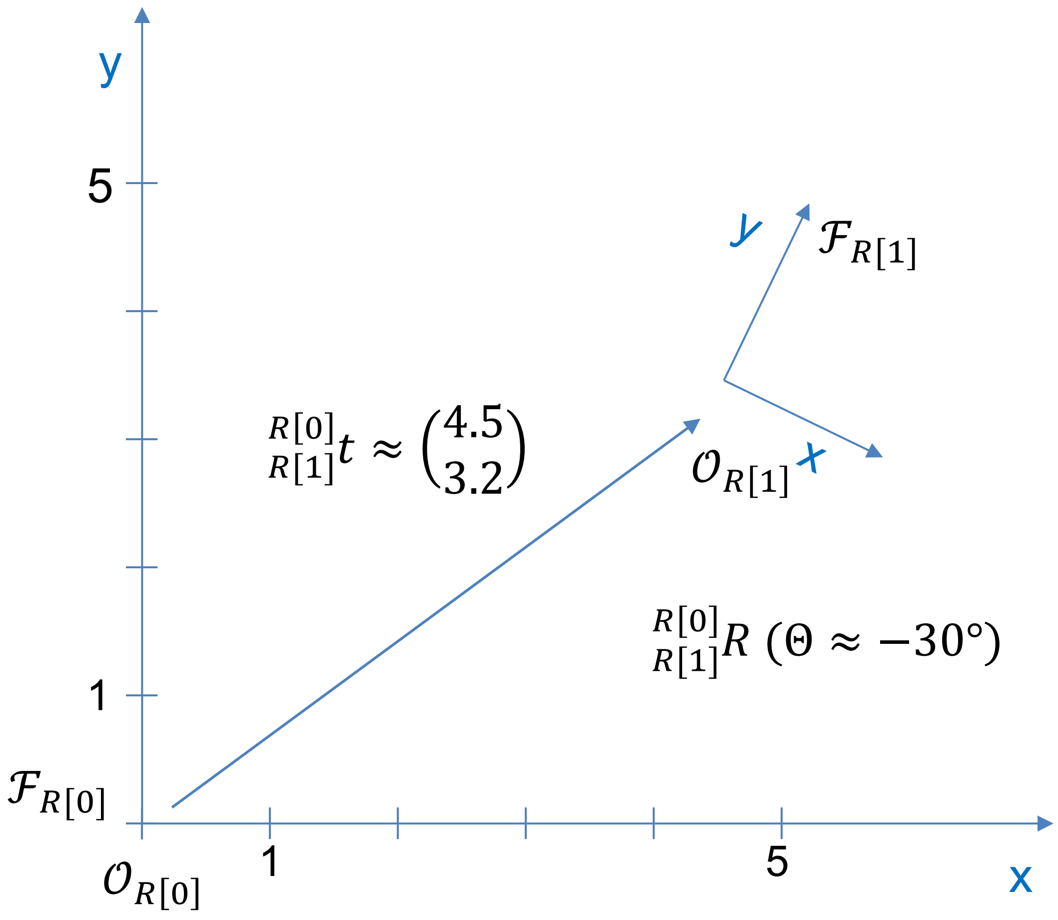
- **Position**:
  - $\begin{pmatrix} x \\ y \end{pmatrix}$ coordinates of any object or point (or another frame)
  - with respect to (wrt.) a specified frame
- **Orientation**:
  - (Θ) angle of any oriented object (or another frame)
  - with respect to (wrt.) a specified frame
- **Pose**:
  - $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$ 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. $\mathcal{F}_G$

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

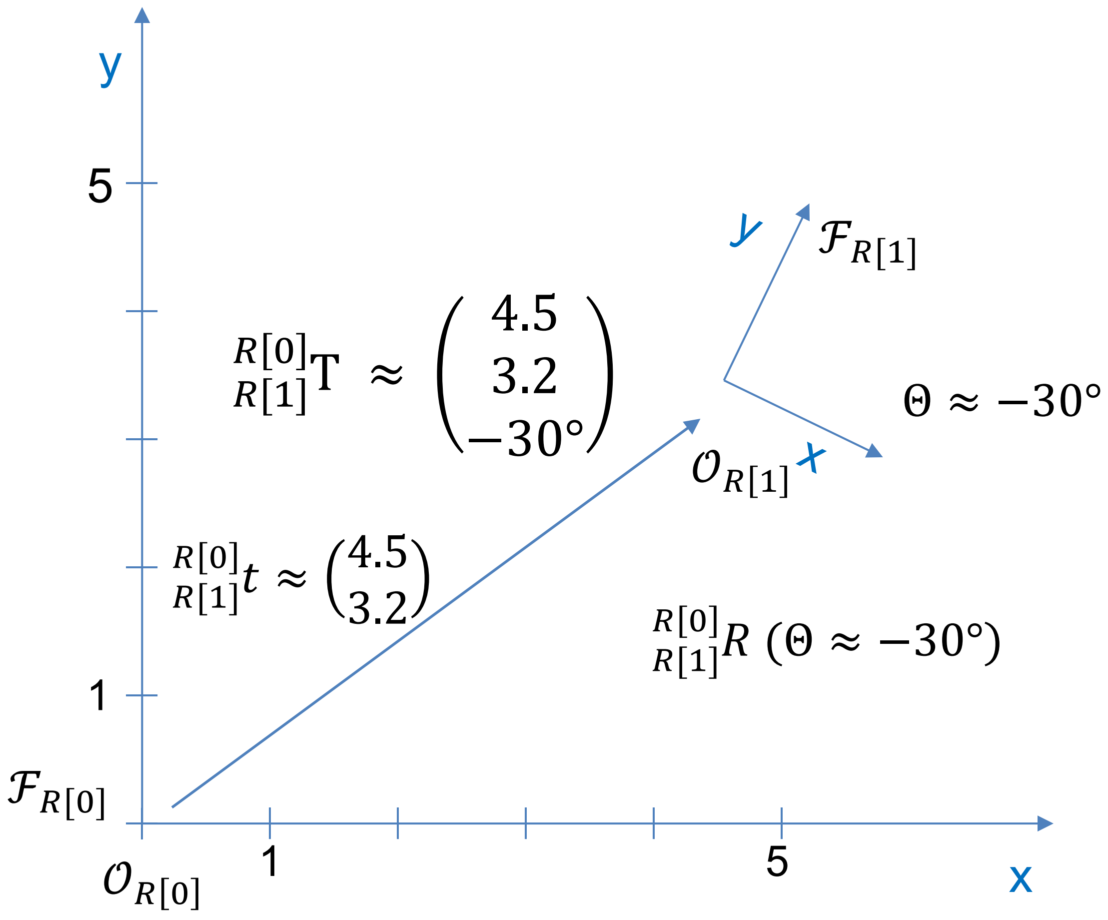
    so $\mathcal{O}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

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

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

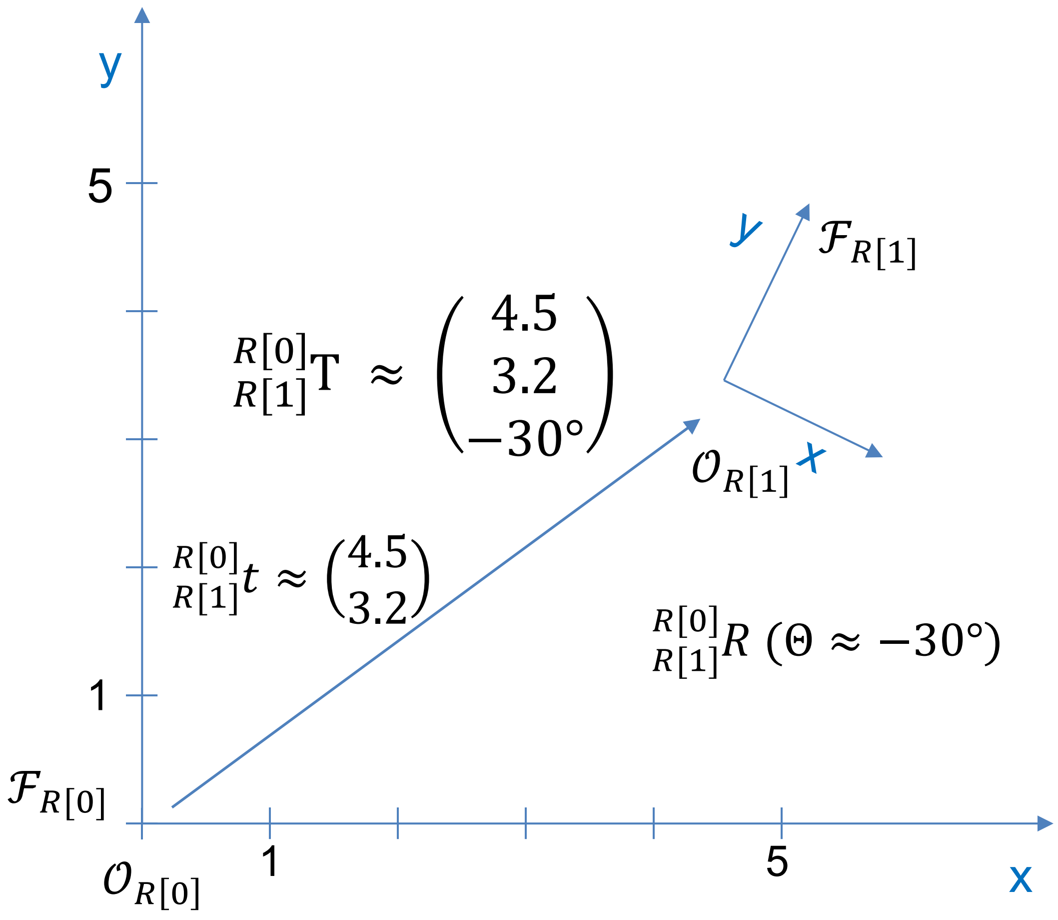
    $\triangleq \quad {}^{R[X]}_{R[X+1]}R$: **Rotation** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

Transform



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

Geometry approach to Odometry



We want to know:

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

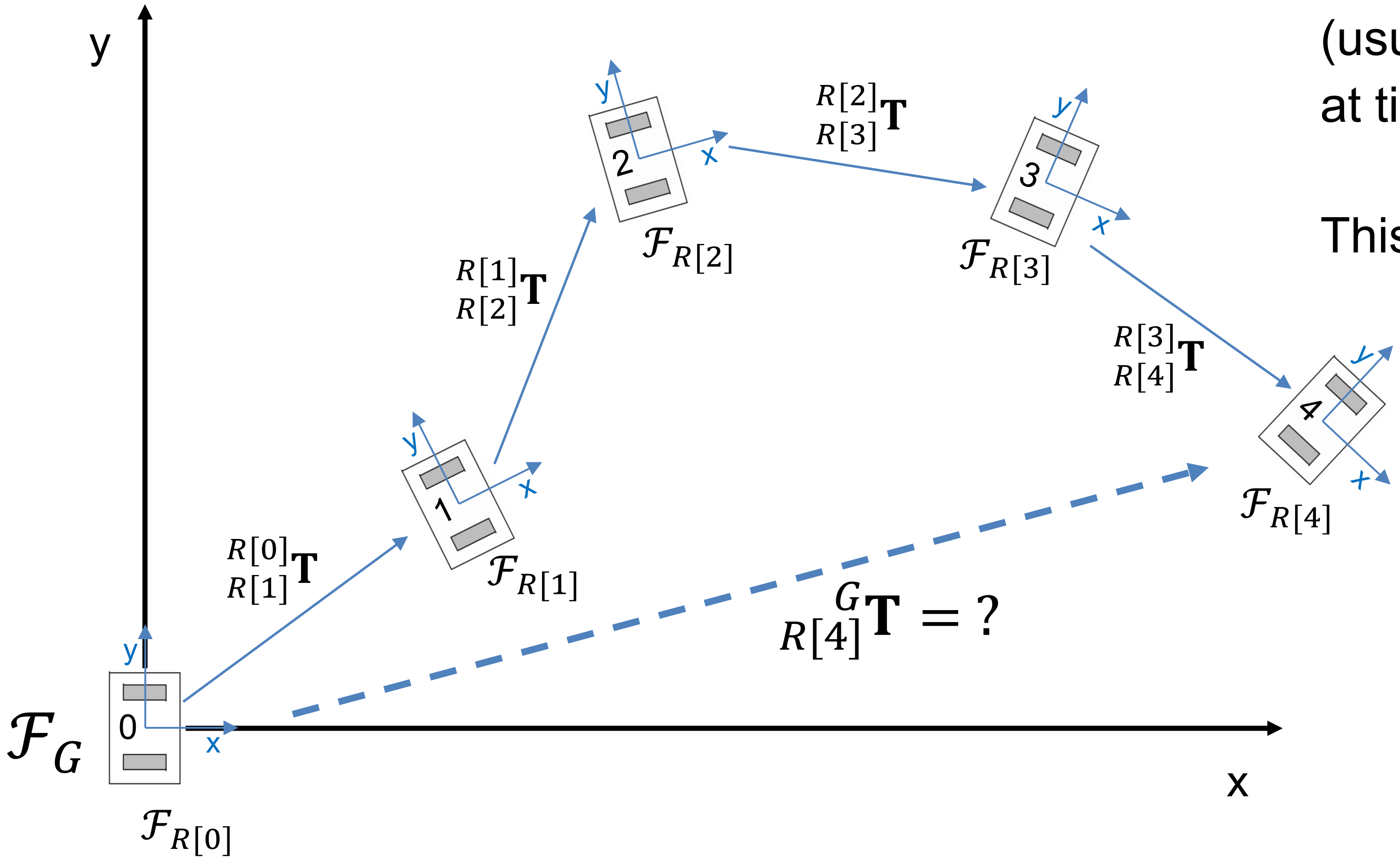
With respect to (wrt.) $\mathcal{F}_G$: The global frame; global coordinate system

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

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

Blackboard: $^{R[1]}_{R[2]}\mathrm{T} \approx \begin{pmatrix} 2 \\ 3 \\ 60° \end{pmatrix}$

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 ${}^{G}_{R[X]}\mathbf{T}$

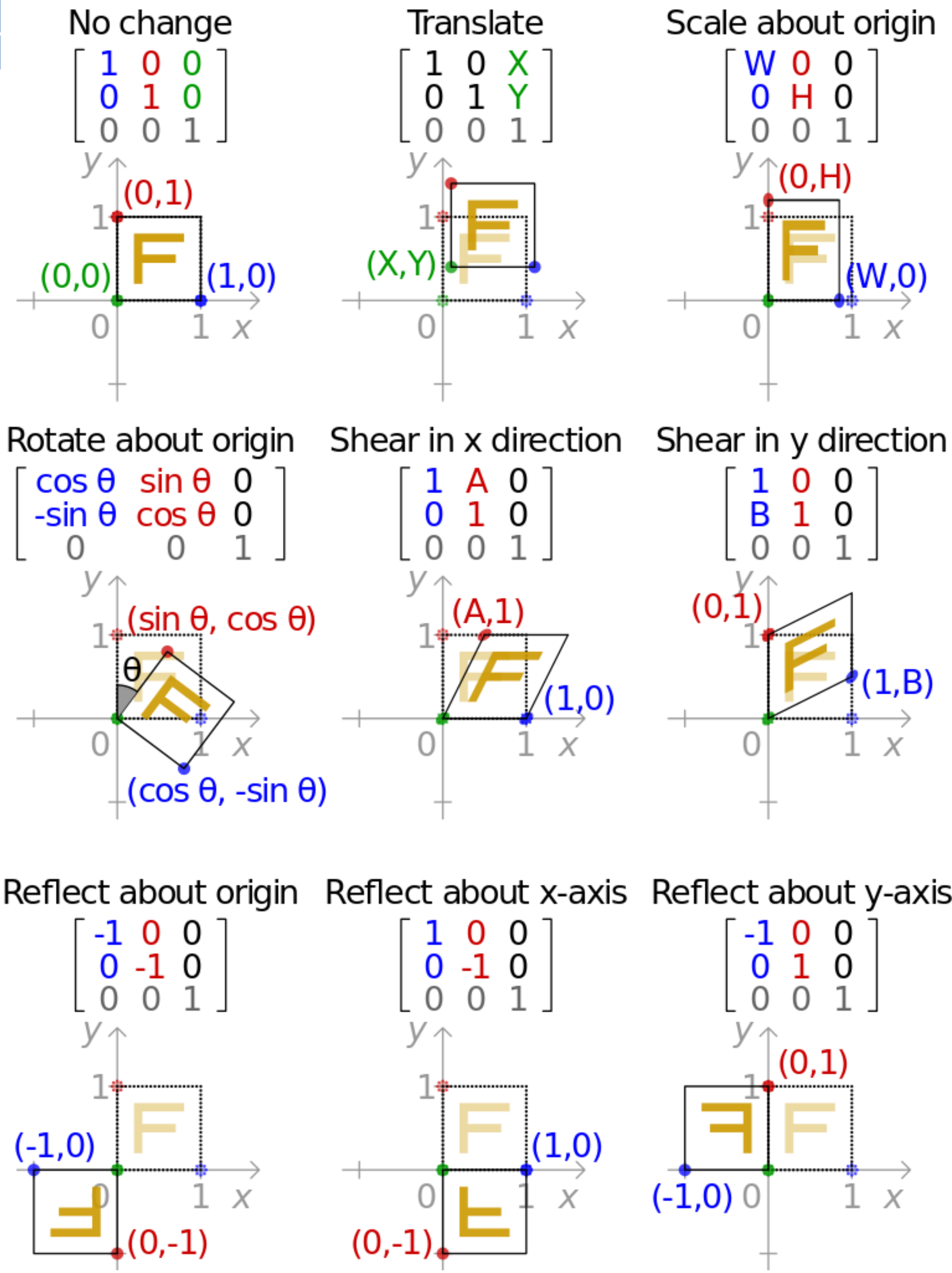
Chaining of Transforms

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

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

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 coorinate system and back)
- Just rotation & translation: rigid motions or proper rigid transformations:
  - Decomposed to rotation and translation
  - => subset of Affine Transofrmations
- In Robotics: Just use term **Transform** or **Transformation** for rigid motions

Lie groups for transformations

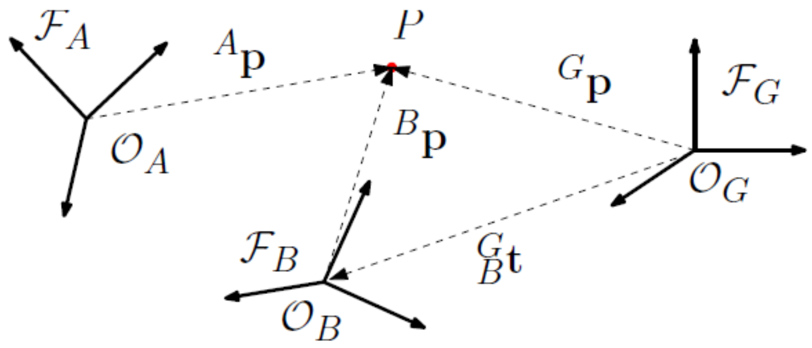
- Smoothly differentiable Group
- No singularities
- Good interpolation

- SO: Special Orthorgonal group
- SE: Special Euclidian group
- Similarity 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

http://ethaneade.com/lie.pdf

Transform



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

Transform between two coordinate frames

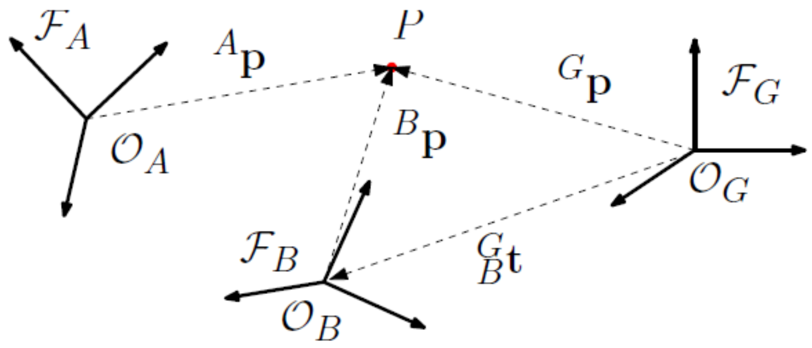
$${}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \triangleq \overrightarrow{\mathcal{O}_{\mathrm{G}}\mathcal{O}_{\mathrm{A}}} \text{ resolved in } \mathcal{F}_{\mathrm{G}}$$

$$\begin{pmatrix} {}^{\mathrm{G}}\mathbf{p} \\ 1 \end{pmatrix} \equiv \begin{pmatrix} {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{R} & {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \\ \mathbf{0}_{1\times[2,3]} & 1 \end{pmatrix} \begin{pmatrix} {}^{\mathrm{A}}\mathbf{p} \\ 1 \end{pmatrix} \qquad {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{T} \equiv \left\{ \begin{matrix} {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \\ {}^{\mathrm{G}}_{\mathrm{A}}\mathbf{R} \end{matrix} \right\}$$

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

$$\begin{bmatrix} \cos\theta & -\sin\theta & {}^{\mathrm{G}}_{\mathrm{A}}t_x \\ \sin\theta & \cos\theta & {}^{\mathrm{G}}_{\mathrm{A}}t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Transform: Operations



Transform between two coordinate frames (chaining, compounding):

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

Inverse of a Transform :

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

Relative (Difference) Transform :

$$ {}^{\mathrm{B}}_{\mathrm{A}}\mathbf{T} = {}^{\mathrm{G}}_{\mathrm{B}}\mathbf{T}^{-1}\,{}^{\mathrm{G}}_{\mathrm{A}}\mathbf{T} $$

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

Chaining :

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

In 2D Translation:

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

In 2D Rotation:

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

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

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

$$ {}^{R[X]}_{R[X+1]}t_x $$

$$ {}^{R[X]}_{R[X+1]}t_y $$

$$ {}^{R[X]}_{R[X+1]}\Theta $$

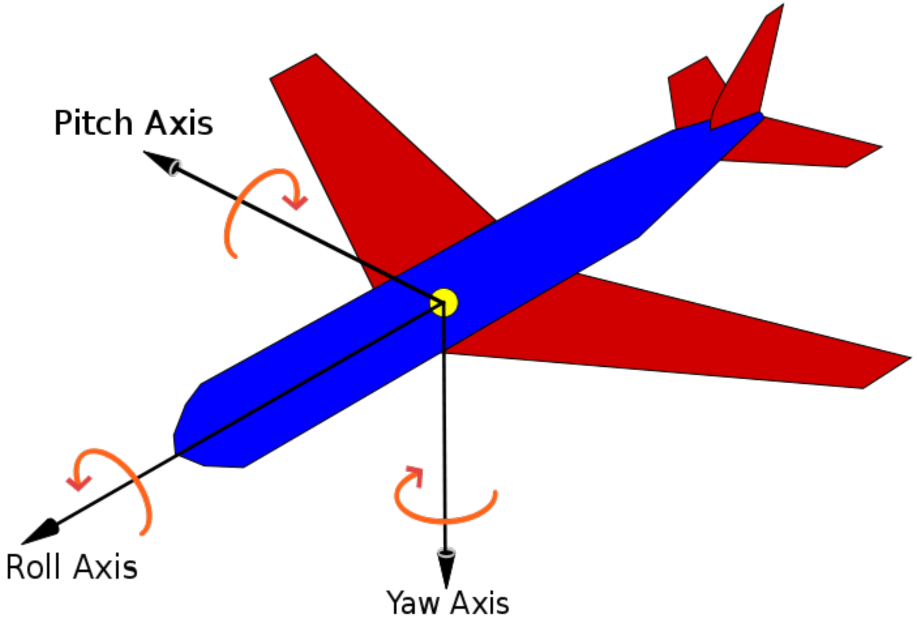
```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose2D pose2D
  float64 x
  float64 y
  float64 theta
```

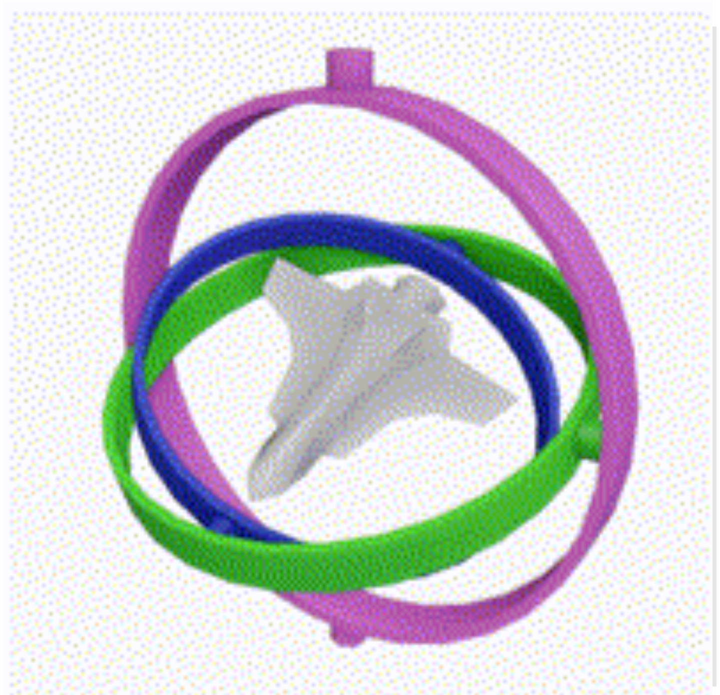
3D Rotation

- 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"





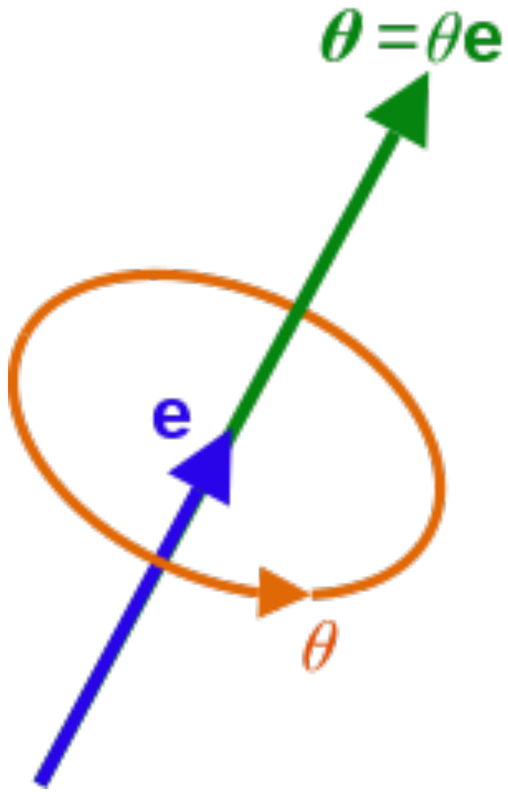
3D Rotation

- Axis Angle
  - Angle θ and
  - Axis unit vector **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



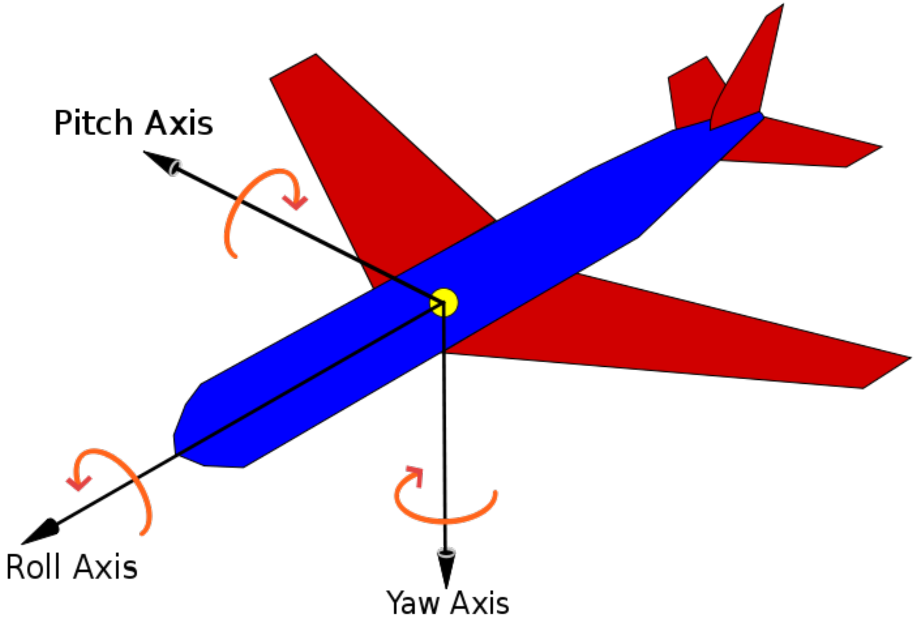
3D Rotation

- 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: q
  - Over determined: 4 variables for 3 DoF (but: unit!)

- Check out: https://eater.net/quaternions !



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

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

$$\check{\mathbf{q}} = \begin{pmatrix} q_0 & q_x & q_y & q_z \end{pmatrix}^{\mathsf{T}} \equiv \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix}$$

Transform in 3D

Matrix Euler Quaternion

$$ {}^{G}_{A}\mathbf{T} = \begin{bmatrix} {}^{G}_{A}\mathrm{R} & {}^{G}_{A}\mathrm{t} \\ 0_{1x3} & 1 \end{bmatrix} = \begin{pmatrix} {}^{G}_{A}\mathrm{t} \\ {}^{G}_{A}\Theta \end{pmatrix} = \begin{pmatrix} {}^{G}_{A}\mathrm{t} \\ {}^{G}_{A}\check{\mathrm{q}} \end{pmatrix} $$

$$ {}^{G}_{A}\Theta \triangleq \left(\theta_r, \theta_p, \theta_y\right)^T $$

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

Rotation Matrix 3x3

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

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

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

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

yaw = α, pitch = β, roll = γ

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/namespacetf.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

ROS Standards:

- Standard Units of Measure and Coordinate Conventions
  - http://www.ros.org/reps/rep-0103.html

- Coordinate Frames for Mobile Platforms:
  - http://www.ros.org/reps/rep-0105.html