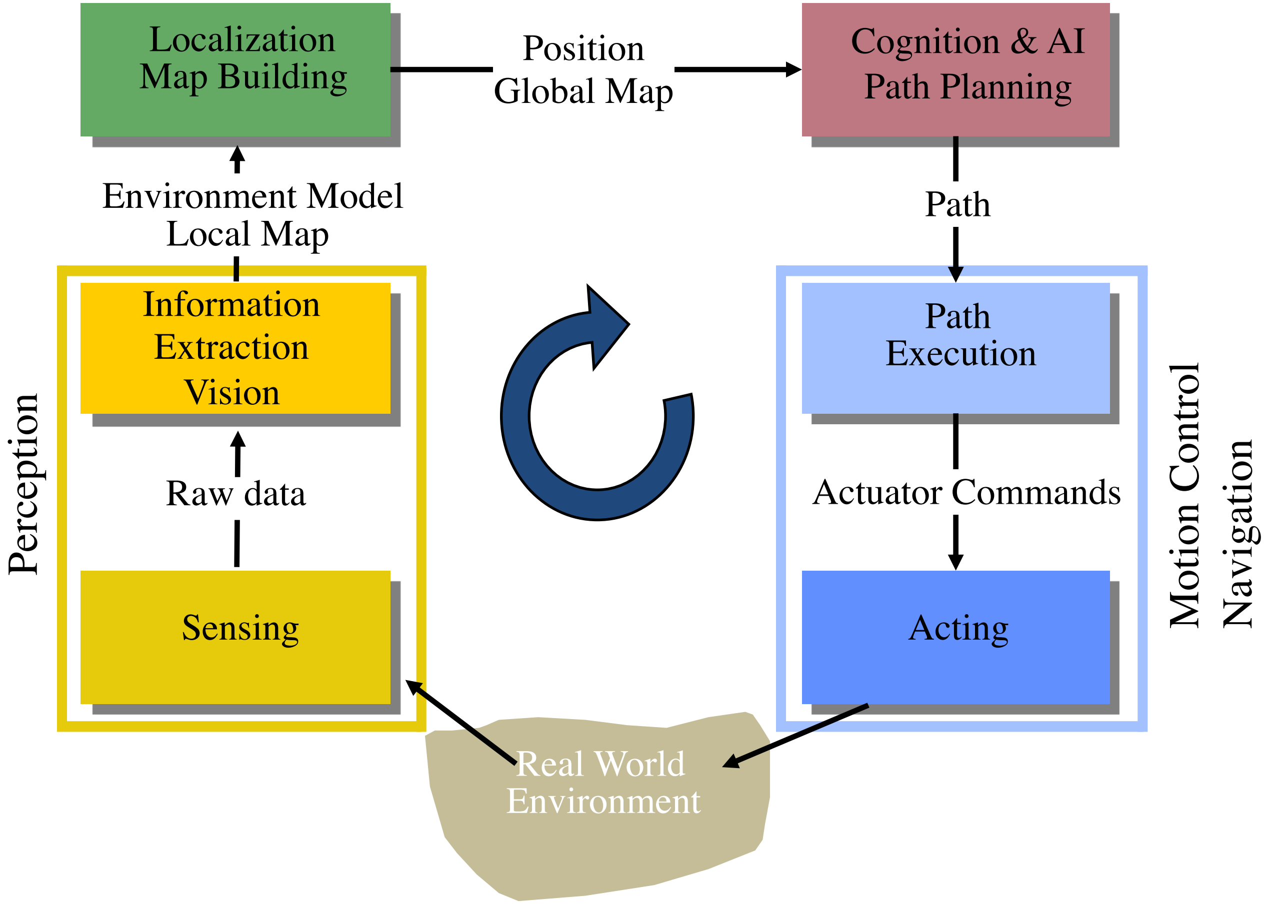
CS283: **Robotics Fall 2019: Kinematics**

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

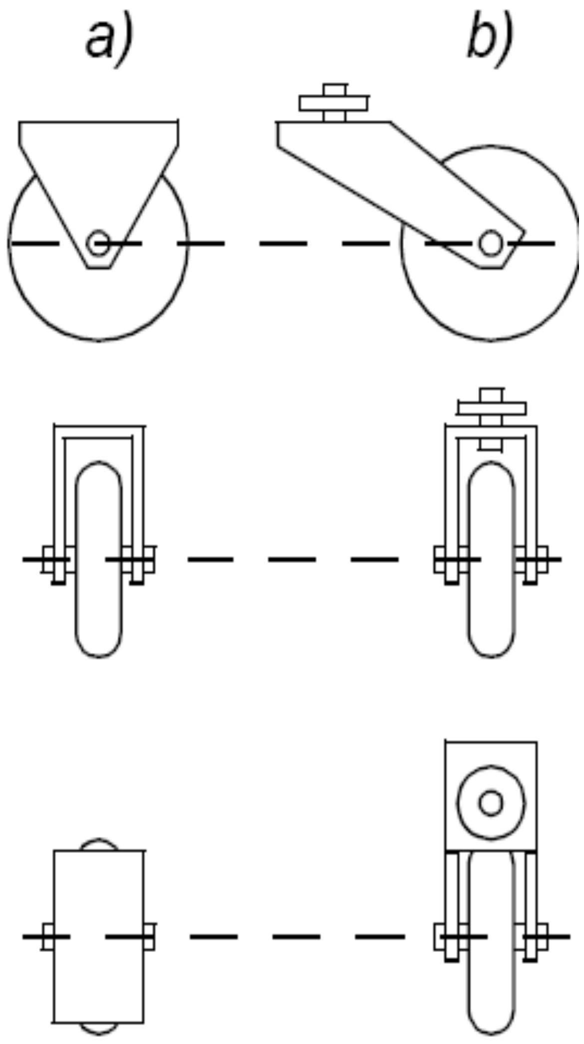
Mobile Robots with Wheels

- 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

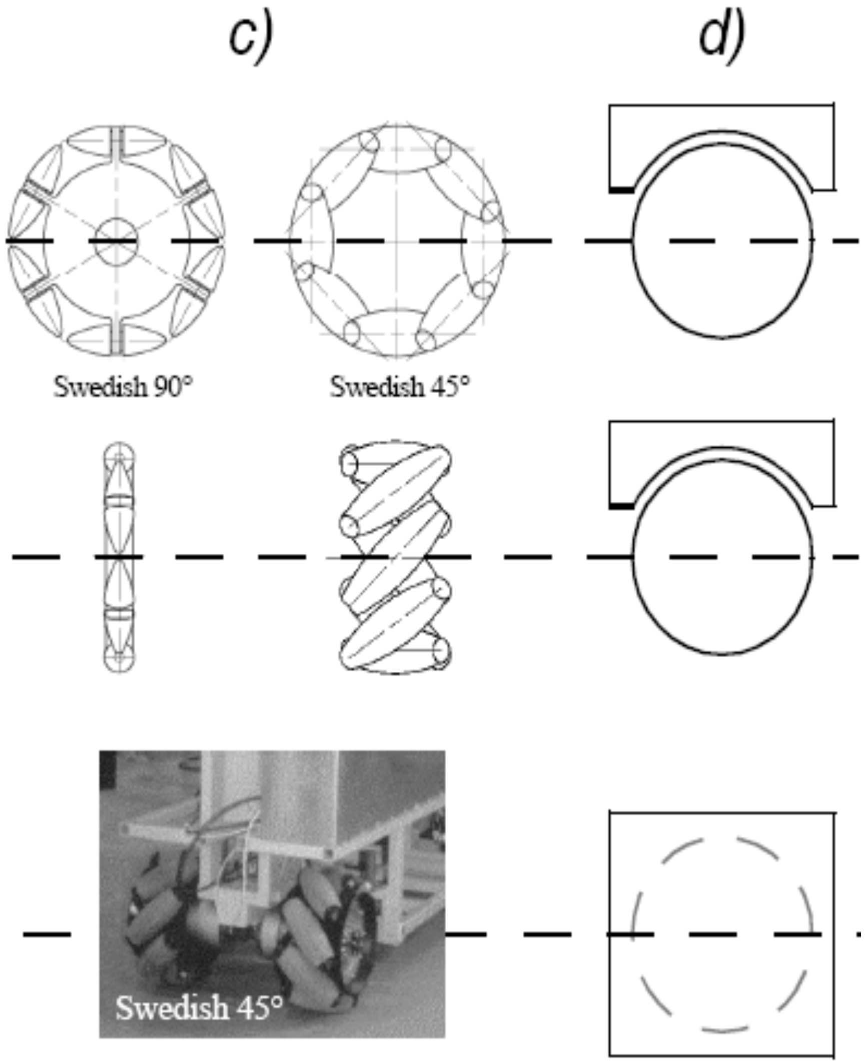
- b) Castor wheel: Three degrees of freedom; rotation around the wheel axle, the contact point and the castor axle



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

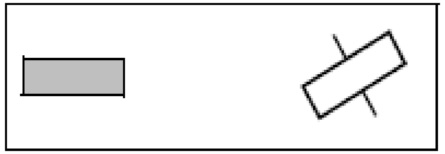


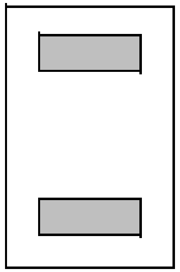
Characteristics of Wheeled Robots and Vehicles

- 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

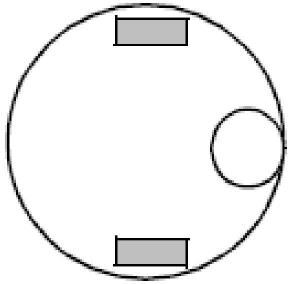
- Two wheels

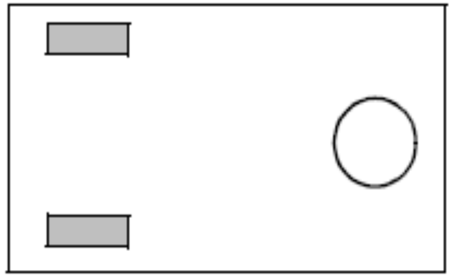


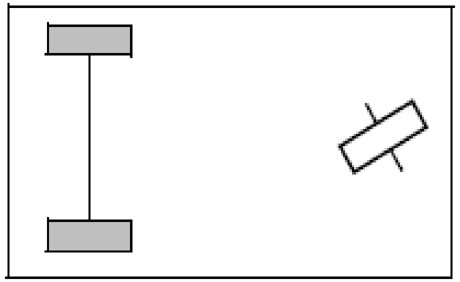


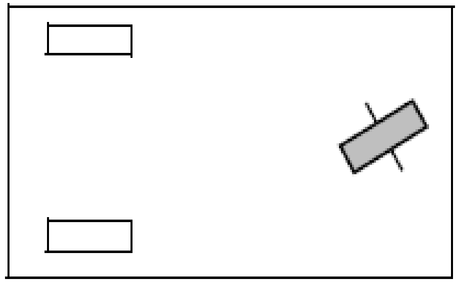
Center of gravity below axle

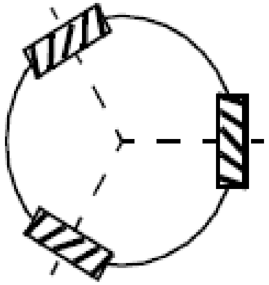
- Three wheels



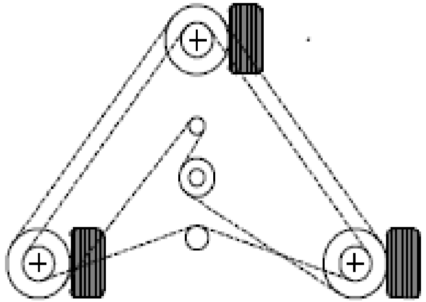








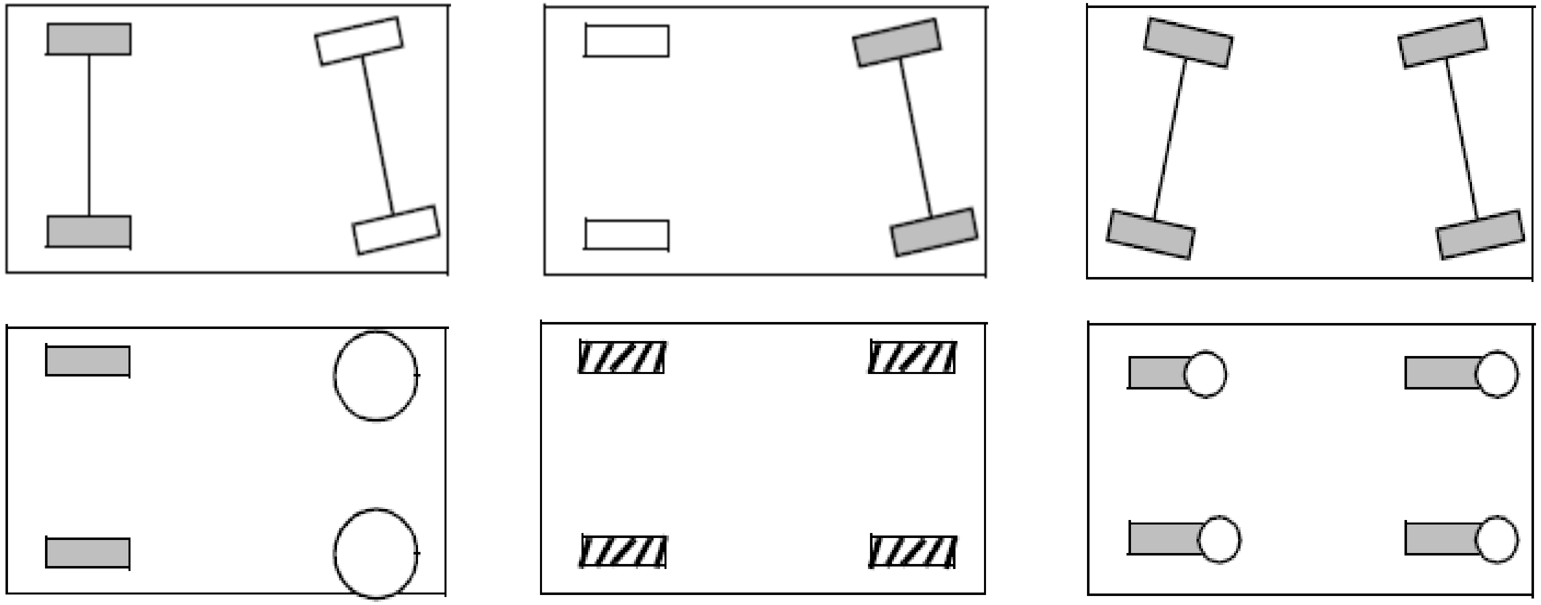
Omnidirectional Drive



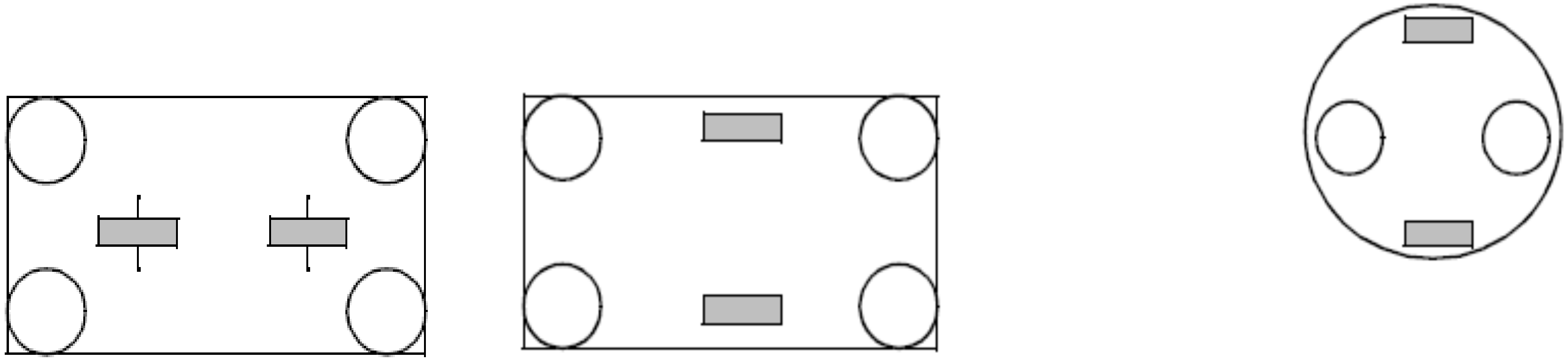
Synchro Drive

Different Arrangements of Wheels II

- Four wheels

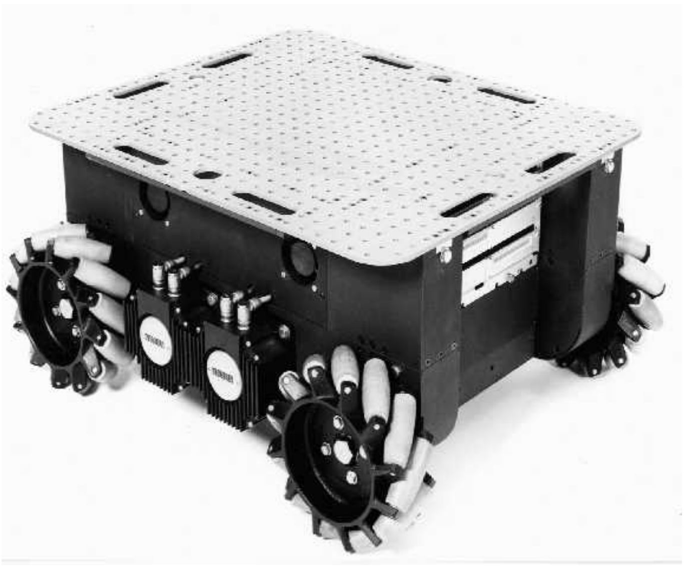


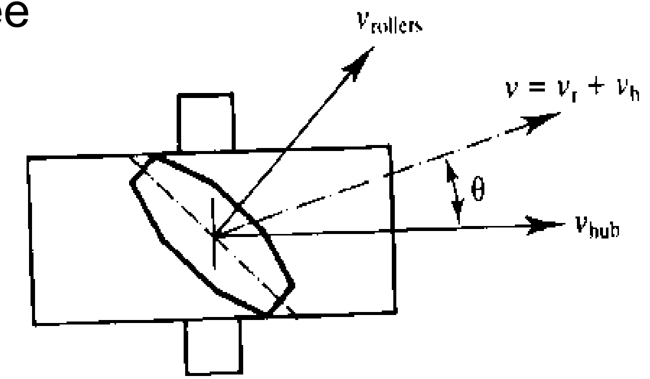
- Six wheels

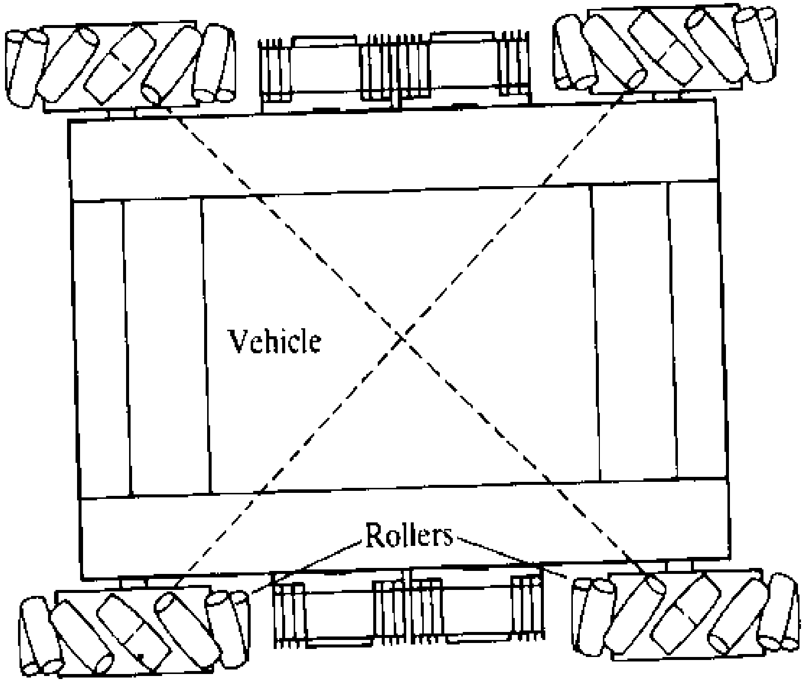


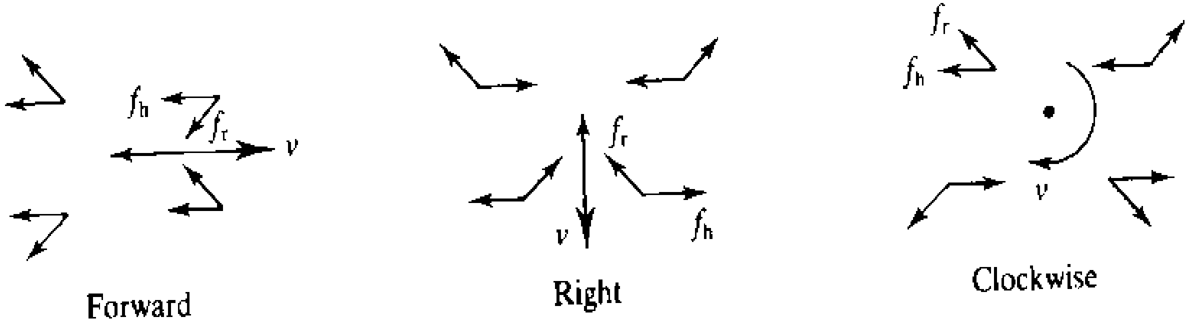
Uranus, CMU: Omnidirectional Drive with 4 Wheels

- 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



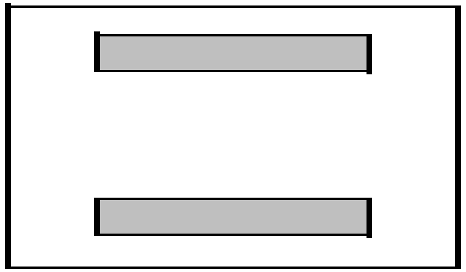


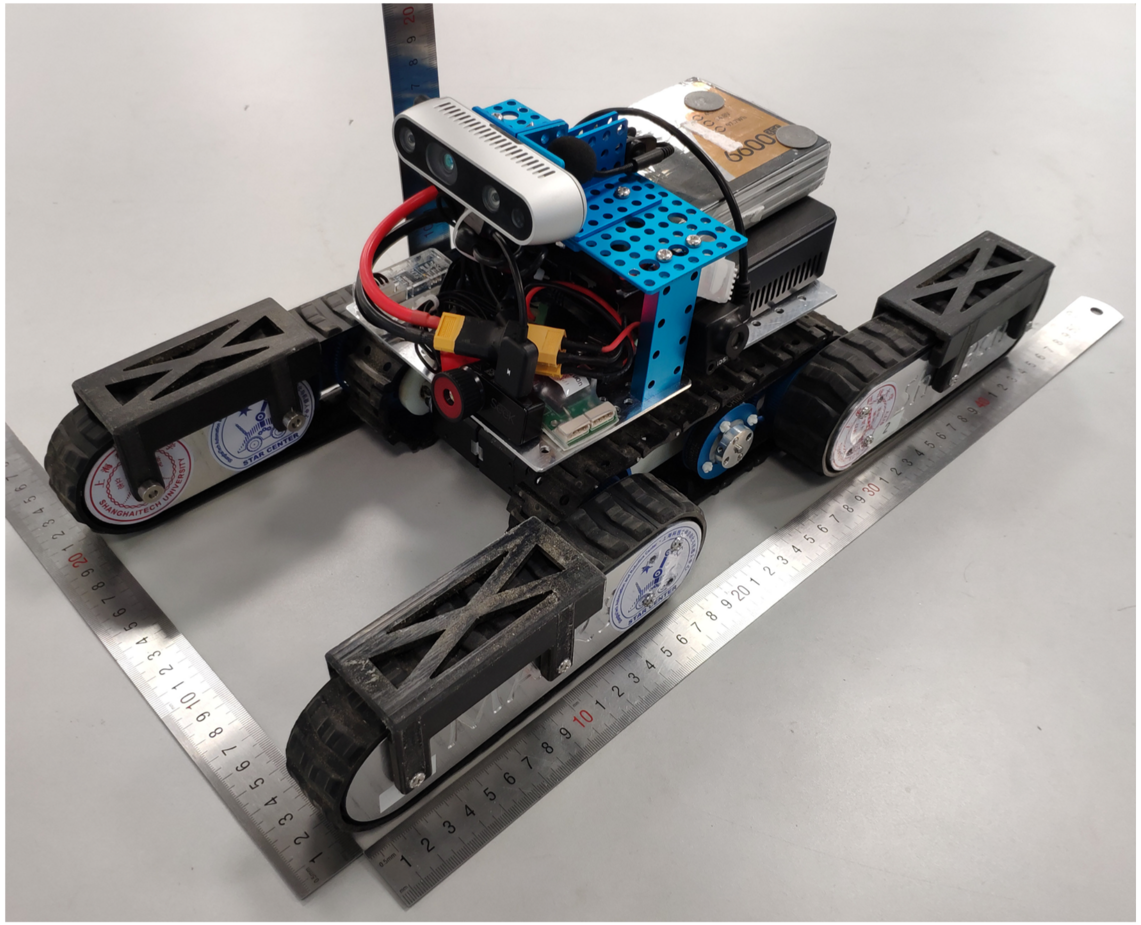




MARS Rescue Robot: Tracked Differential Drive

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





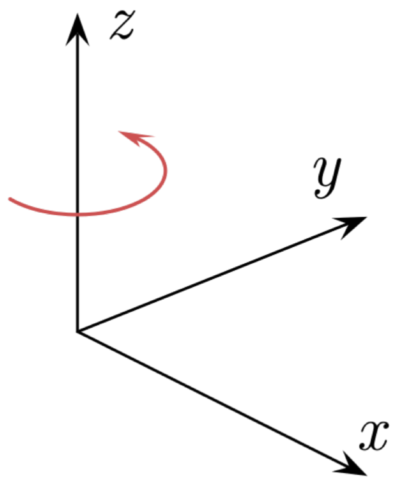
Introduction: Mobile Robot Kinematics

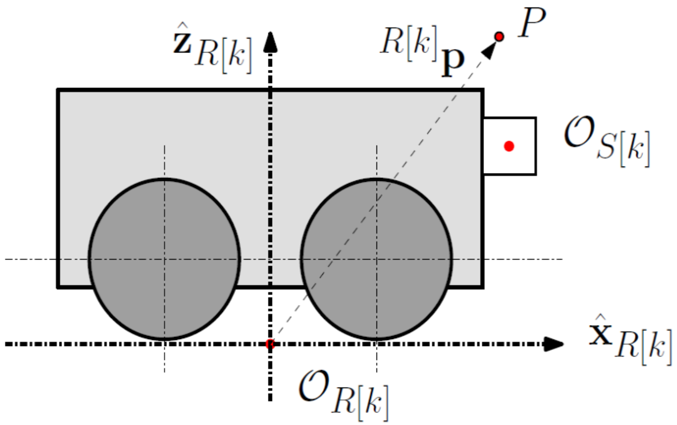
- Aim
  - Description of mechanical behavior of the robot for *design* and *control*
  - Similar to robot manipulator kinematics
  - However, mobile robots can move unbound with respect to its environment
    - there is no direct way to measure the robot's position
    - Position must be integrated over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

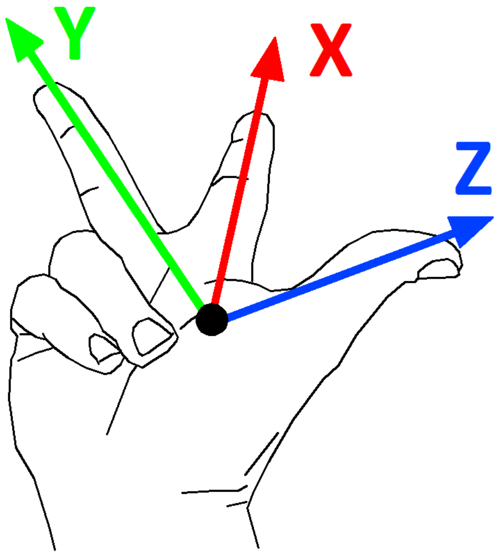
COORDINATE SYSTEM

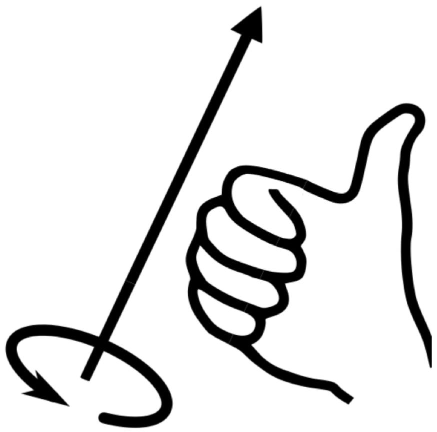
Right Hand Coordinate System

- 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 ???



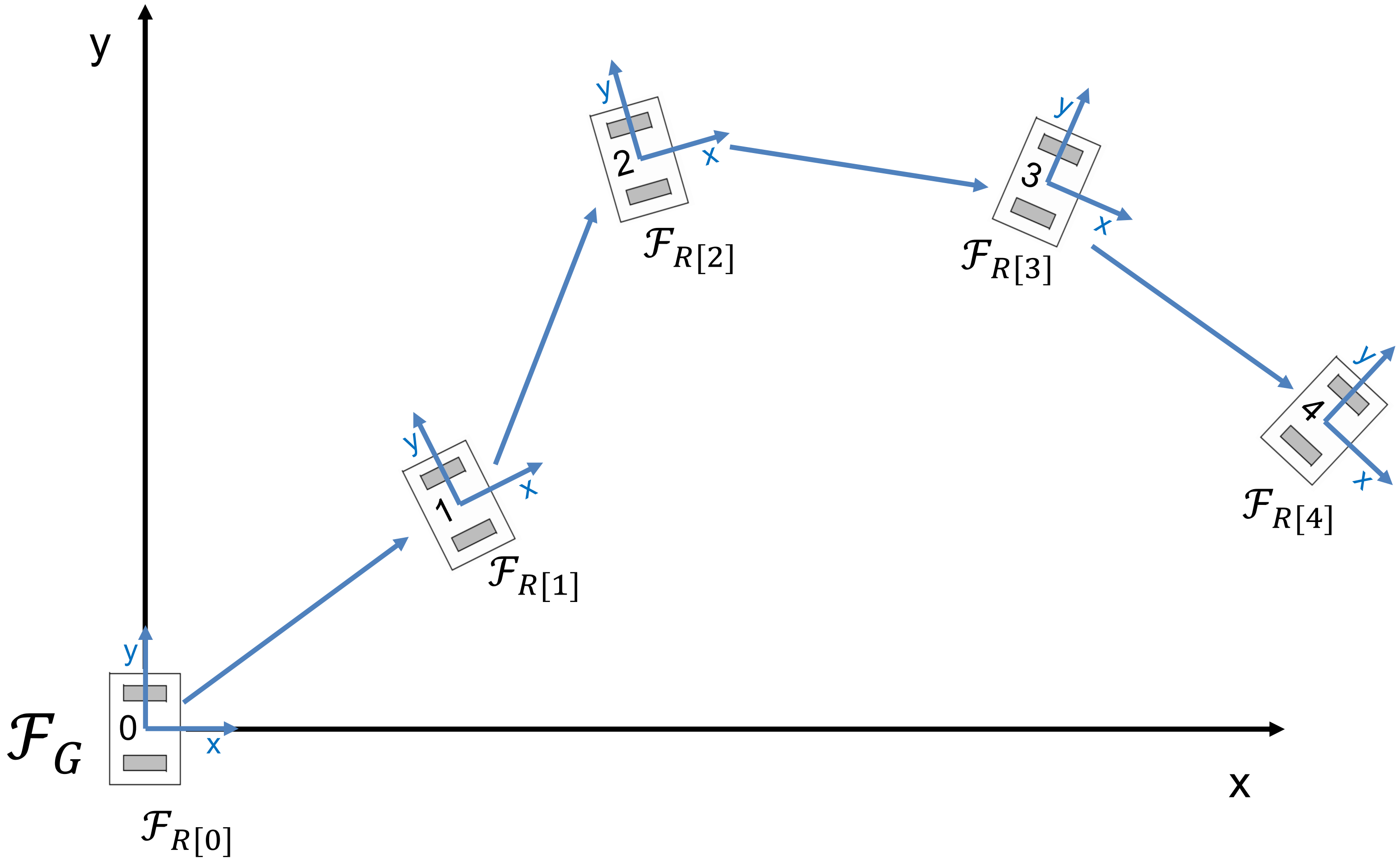






Right Hand Rule http://en.wikipedia.org/wiki/Right-hand_rule

Odometry



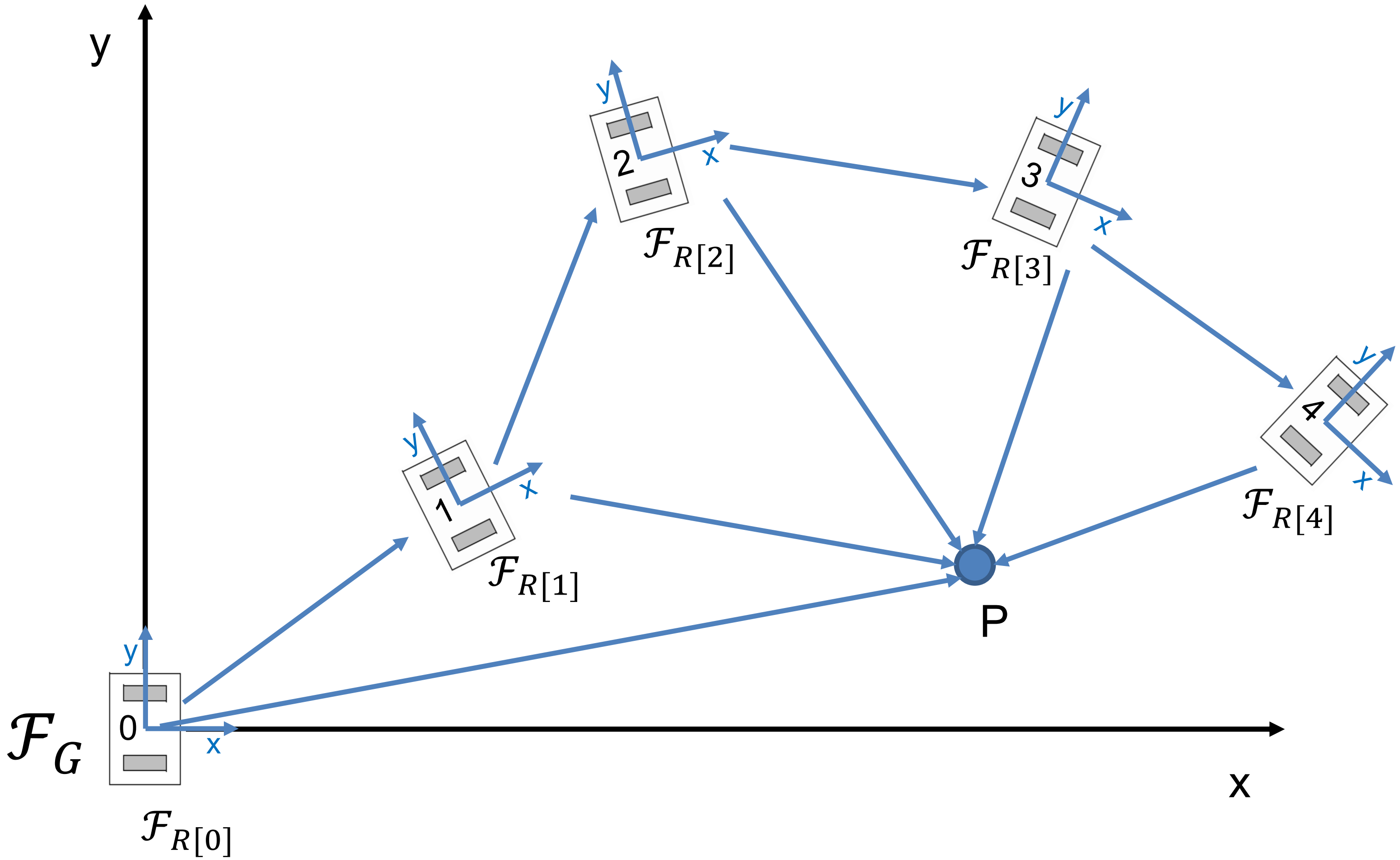
With respect to the robot start pose:
Where is the robot now?

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

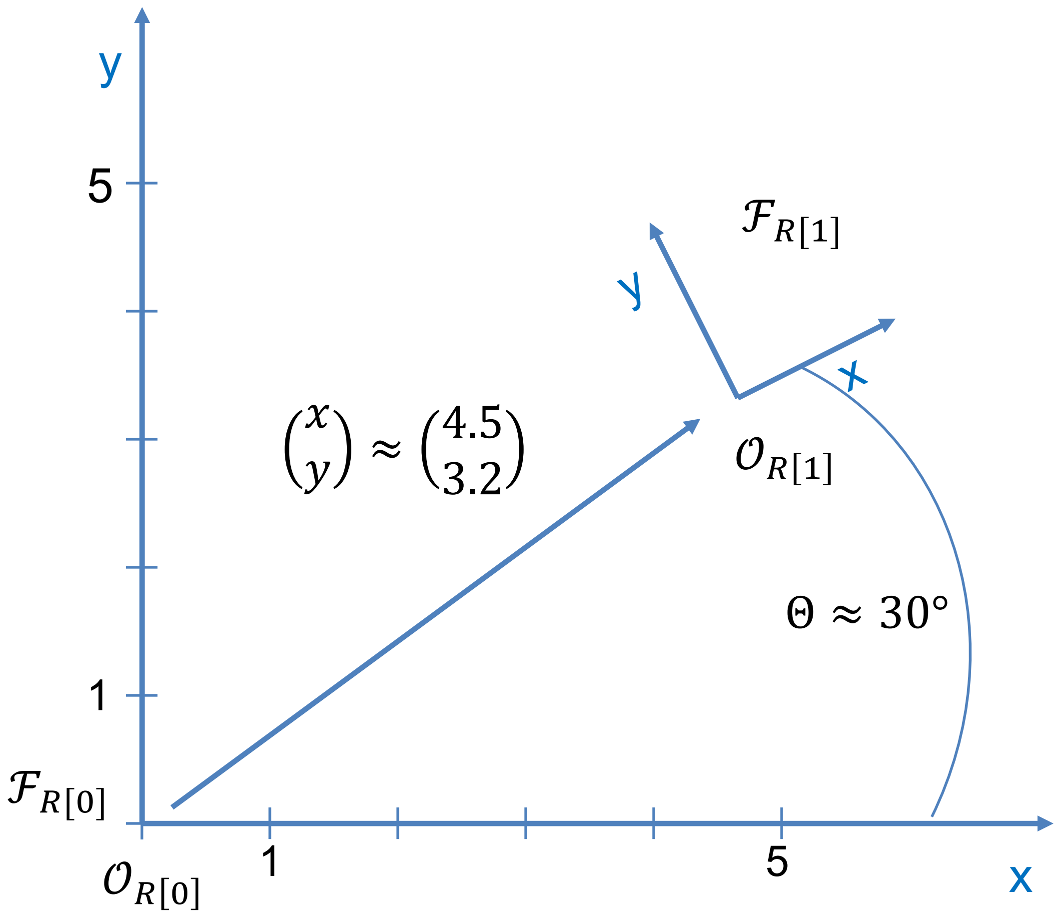


$\mathcal{O}_{R[X]}$: Origin of $\mathcal{F}_{R[X]}$
(coordinates (0, 0)

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

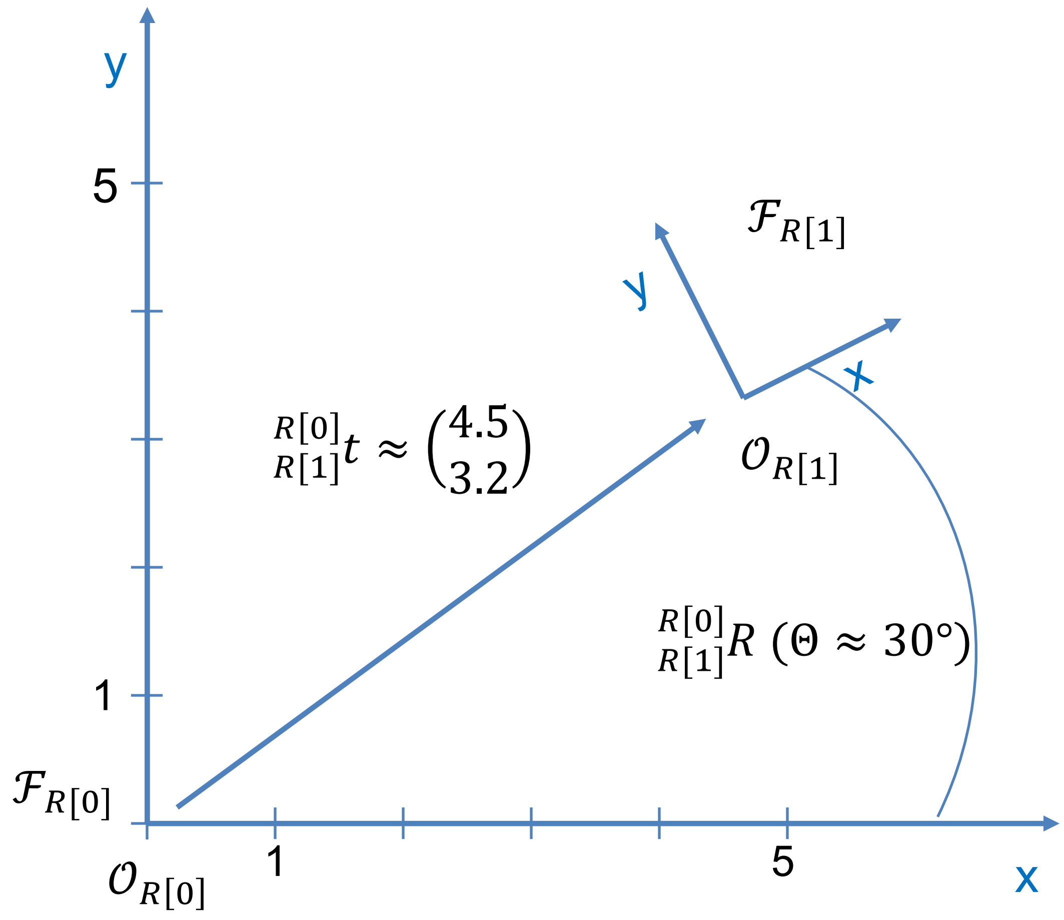
- Object P is observed at times 0 to 4
- Object P is static (does not move)
- The Robot moves
(e.g. $\mathcal{F}_{R[0]} \neq \mathcal{F}_{R[1]}$)
- => (x, y) coordinates of P are different in all frames, for example:
  - $\overrightarrow{\mathcal{O}_{R[0]}\mathrm{P}} \neq \overrightarrow{\mathcal{O}_{R[1]}\mathrm{P}}$

Position, Orientation & Pose



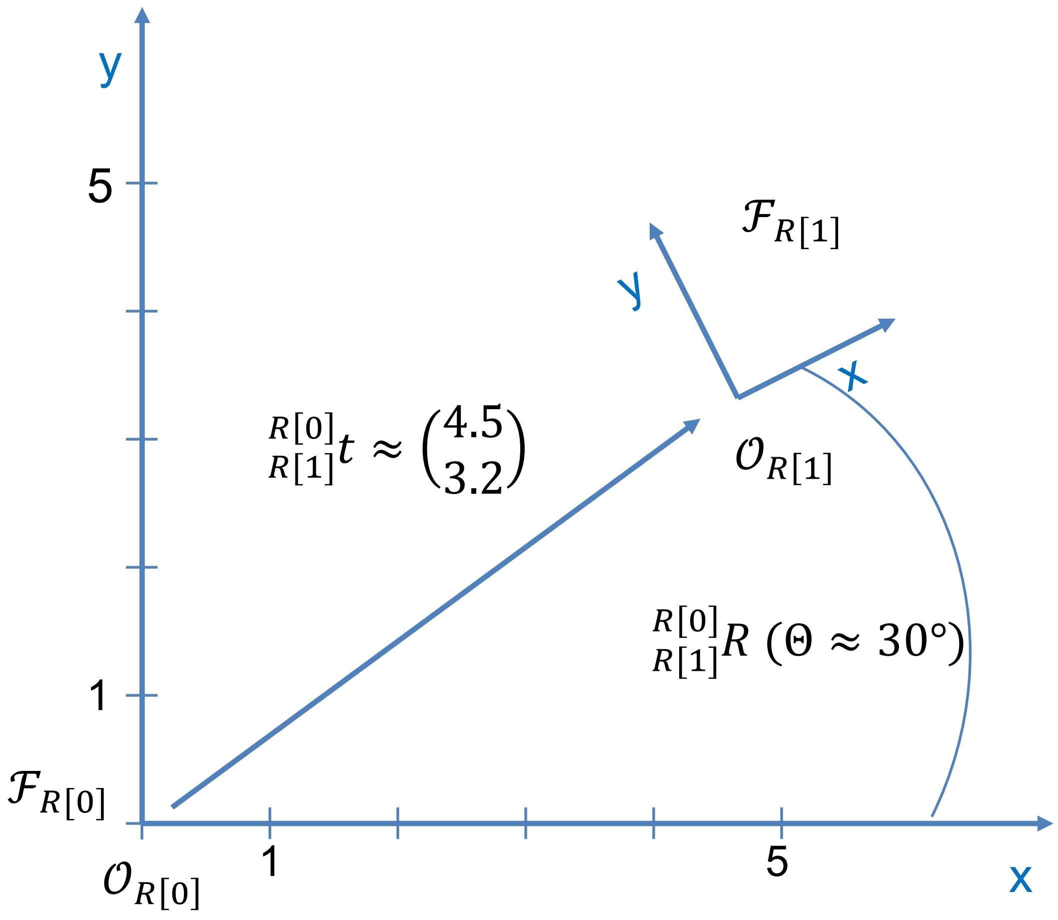
- **Position**:
  - $\begin{pmatrix} x \\ y \end{pmatrix}$ coordinates of any object or point (or another frame)
  - with respect to (wrt.) a specified frame
- **Orientation**:
  - (Θ) angle of any oriented object (or another frame)
  - with respect to (wrt.) a specified frame
- **Pose**:
  - $\begin{pmatrix}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

Translation, Rotation & Transform



- **Translation**:
  - $\binom{x}{y}$ 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



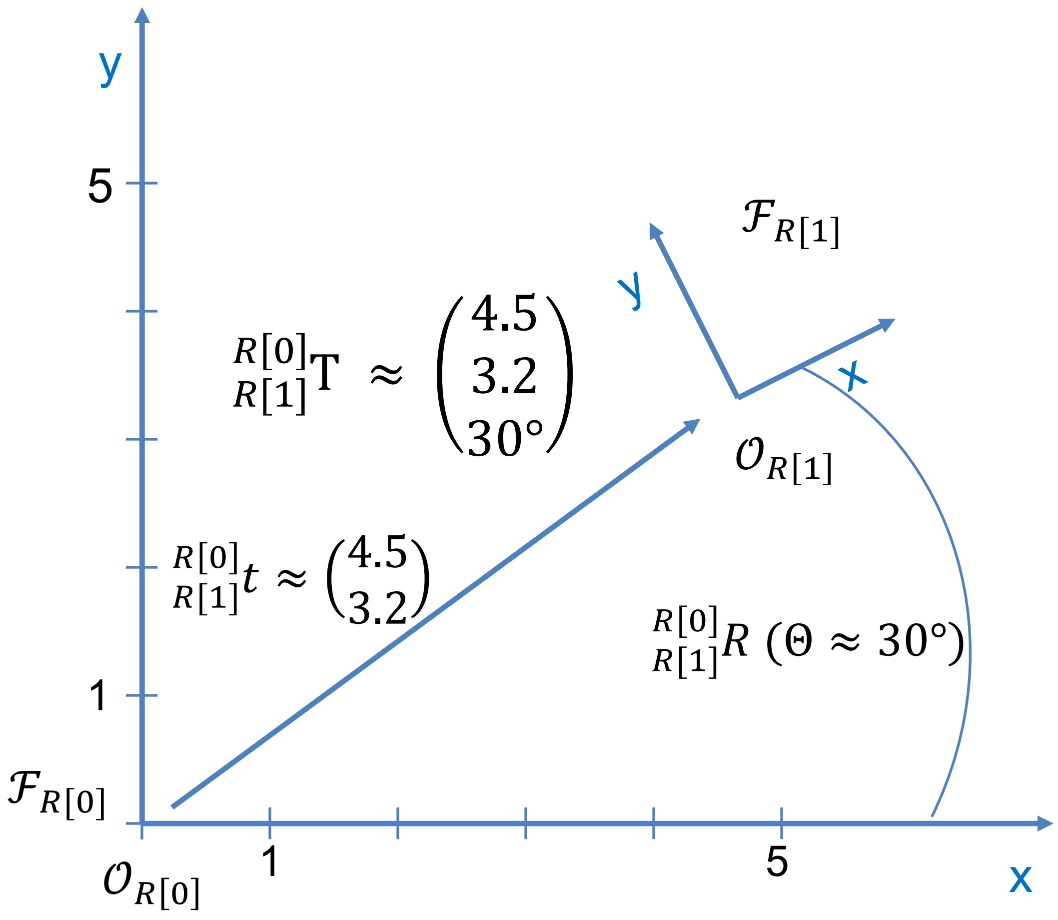
- $\mathcal{F}_{R[X]}$: Frame of reference of the robot at time X
- Where is that frame $\mathcal{F}_{R[X]}$?
  - Can only be expressed with respect to (wrt.) another frame (e.g. global Frame $\mathcal{F}_G$) =>
  - Pose of $\mathcal{F}_{R[X]}$ wrt. $\mathcal{F}_G$
- $\mathcal{O}_{R[X]}$: Origin of $\mathcal{F}_{R[X]}$
  - $\overrightarrow{\mathcal{O}_{R[X]}\mathcal{O}_{R[X+1]}}$: **Position** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

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

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

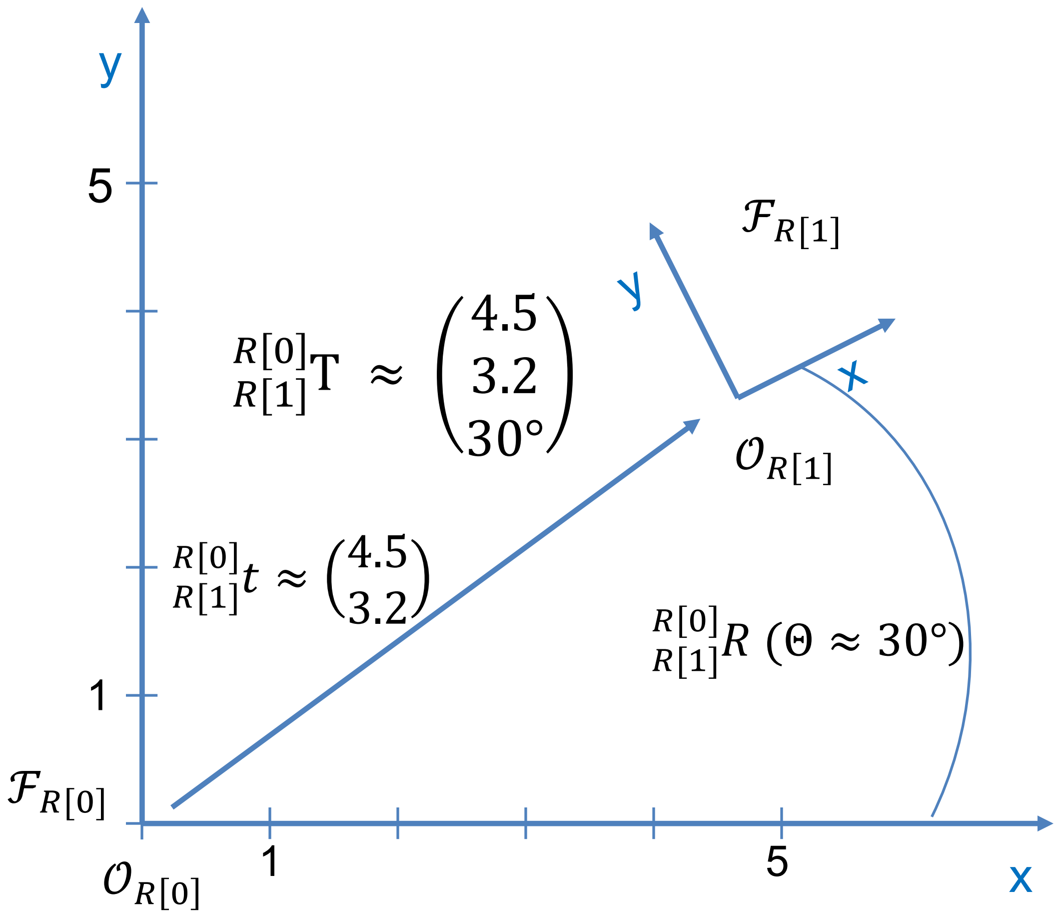
    $\triangleq \ {}^{R[X]}_{R[X+1]}R$: **Rotation** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

Transform



- ${}^{R[X]}_{R[X+1]}t$: **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:** $${}^{R[X]}_{R[X+1]}\mathrm{T} \equiv \begin{Bmatrix} {}^{R[X]}_{R[X+1]}t \\ {}^{R[X]}_{R[X+1]}R \end{Bmatrix}$$

Geometry approach to Odometry



We want to know:

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

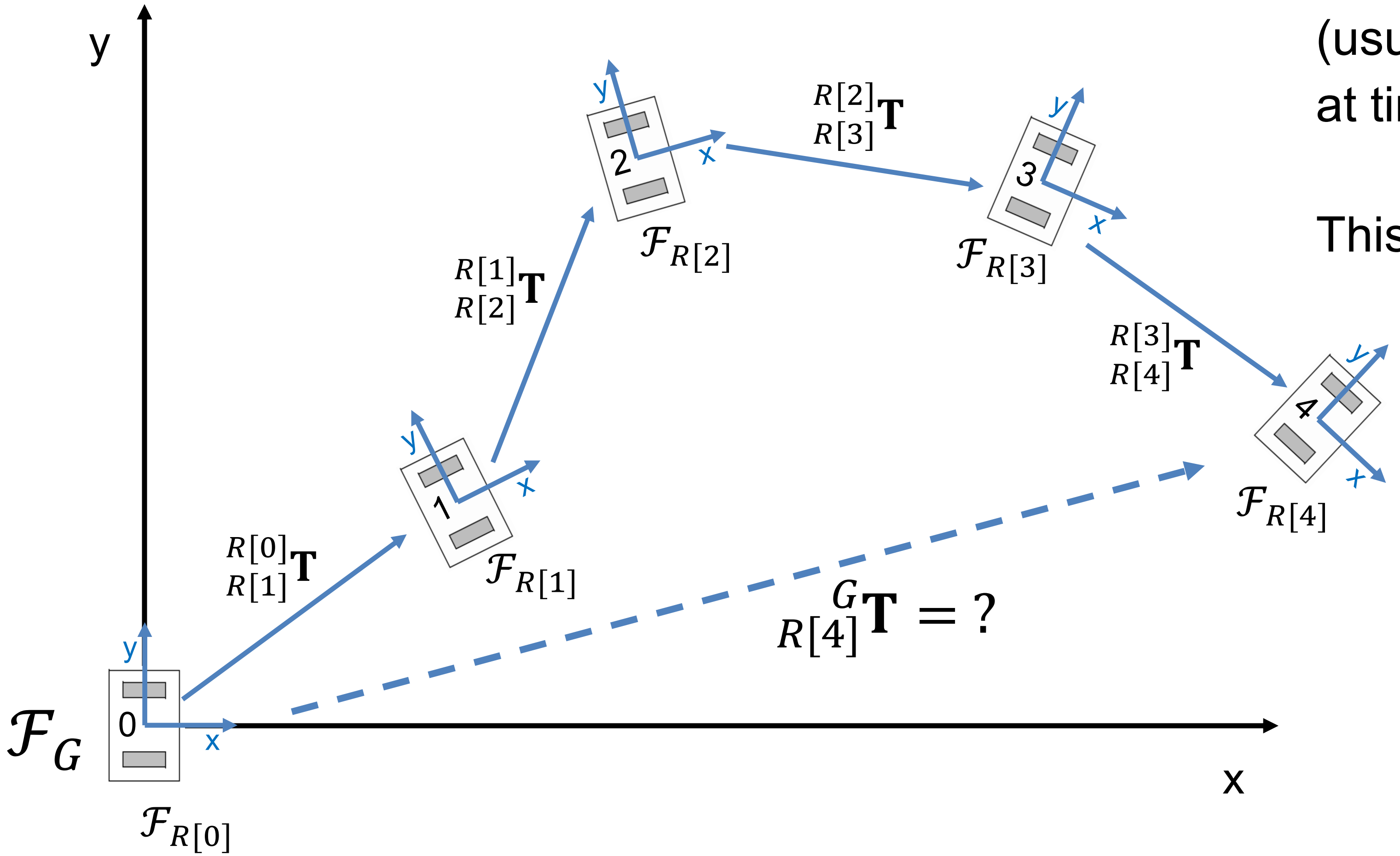
With respect to (wrt.) $\mathcal{F}_G$: The global frame; global coordinate system

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

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

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

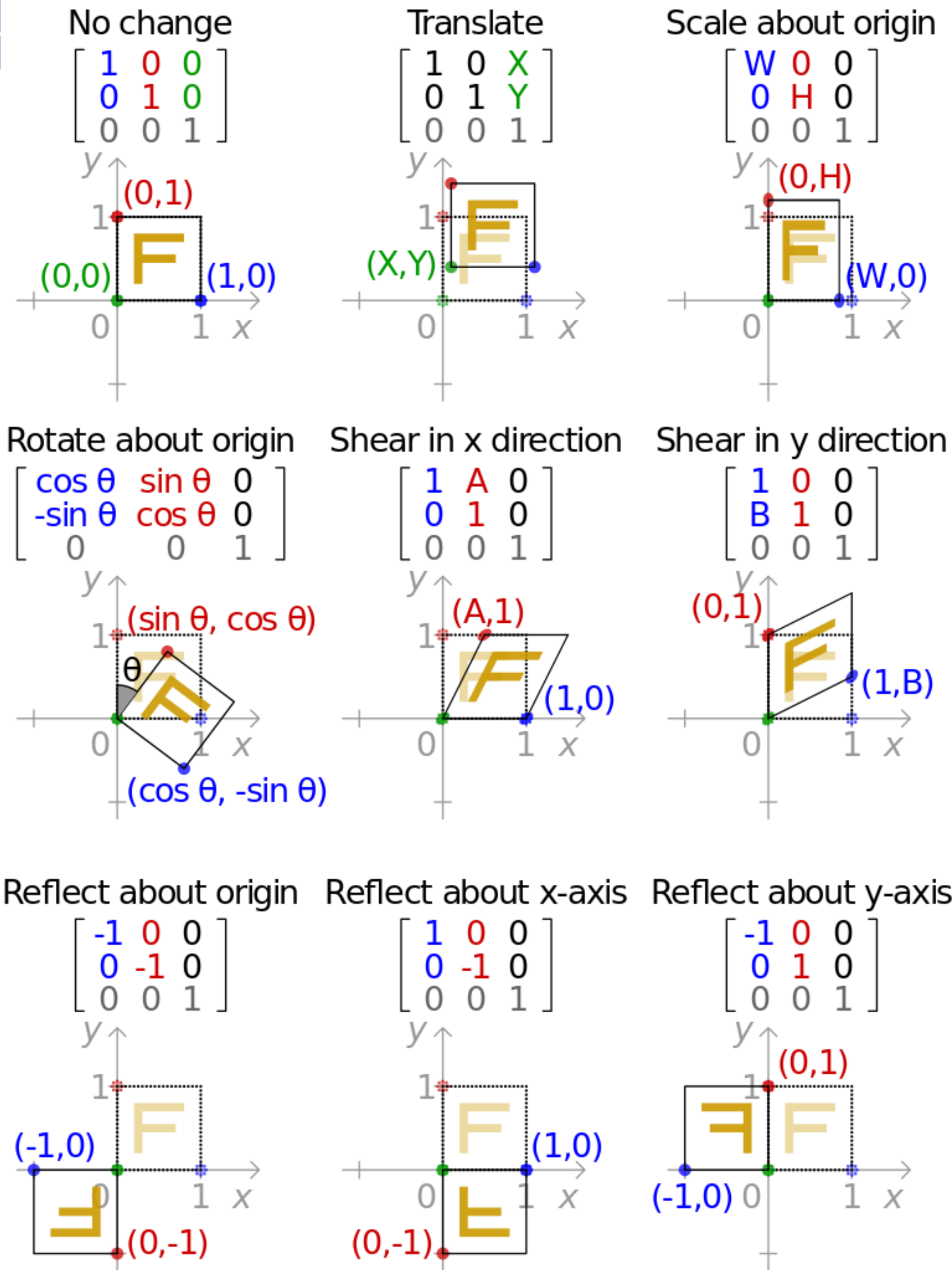
Chaining of Transforms

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

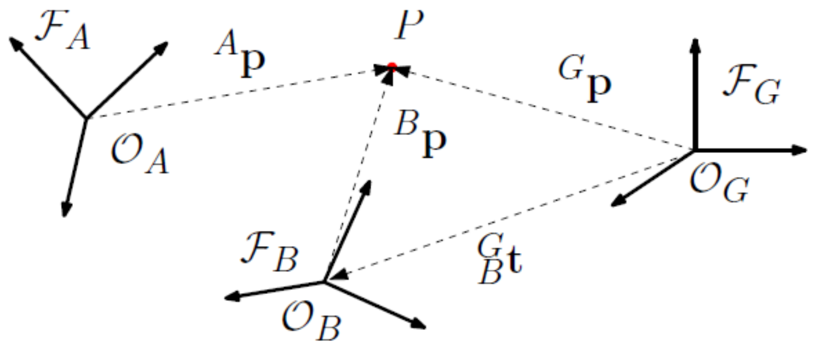
often: $\mathcal{F}_G \equiv \mathcal{F}_{R[0]} \Rightarrow {}^{G}_{R[0]}\mathbf{T} = id$

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



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

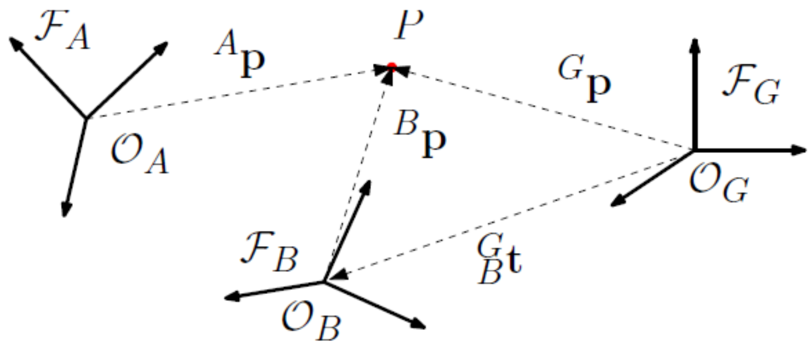
$${}^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \triangleq \overrightarrow{\mathcal{O}_{\mathrm{G}}\mathcal{O}_{\mathrm{A}}} \text{ resolved in } \mathcal{F}_{\mathrm{G}}$$

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

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

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

Transform: Operations



Transform between two coordinate frames (chaining, compounding):

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

Inverse of a Transform :

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

Relative (Difference) Transform :

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

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

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

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

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

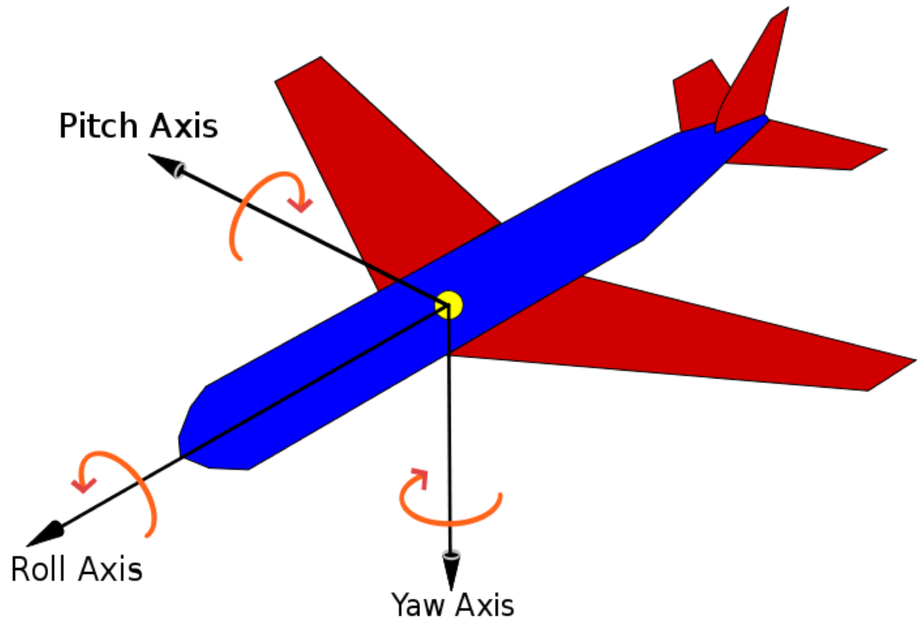
$$
{}^{R[X]}_{R[X+1]}t_y
$$

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

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

3D Rotation

- Euler angles: Roll, Pitch, Yaw
  - ☹ Singularities
- 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
  - Scalar (real) part: q_0 , sometimes q_w
  - Vector (imaginary) part: $\mathbf{q}$
  - Over determined: 4 variables for 3 DoF (but: unit!)



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

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

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

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

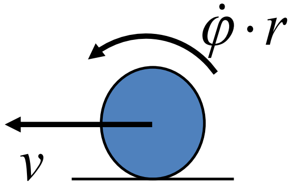
ROS Standards:

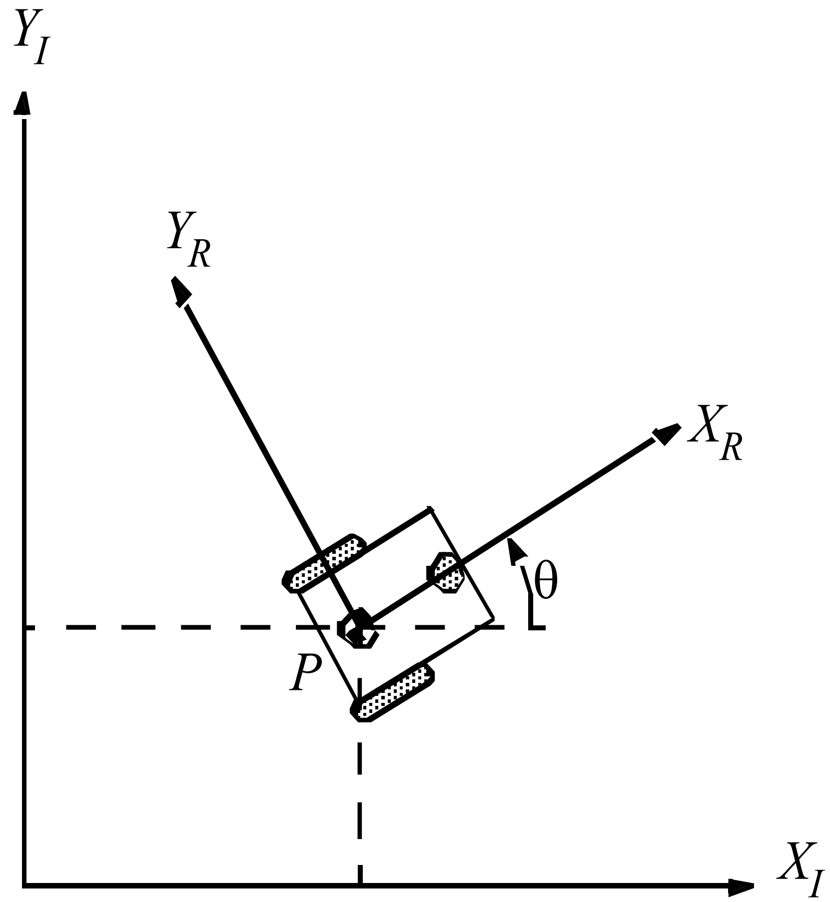
- Standard Units of Measure and Coordinate Conventions
  - http://www.ros.org/reps/rep-0103.html

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

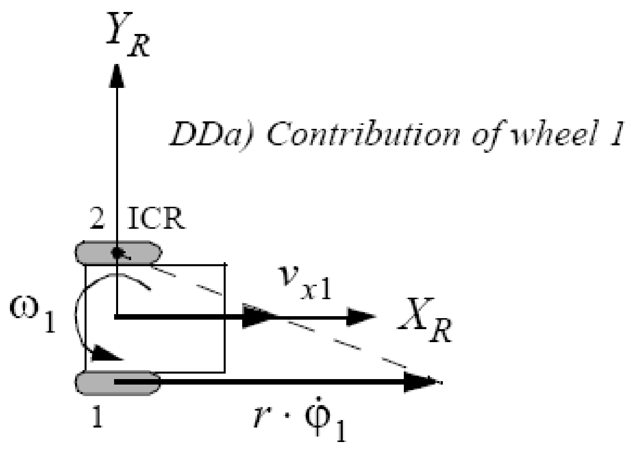
Wheel Kinematic Constraints: Assumptions

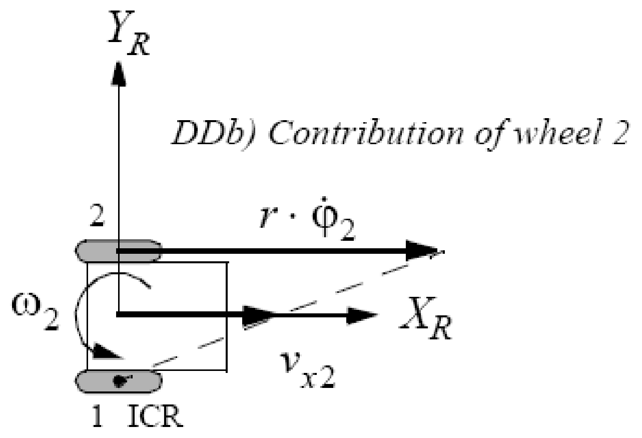
- Movement on a horizontal plane
- Point contact of the wheels
- Wheels not deformable
- Pure rolling
  - $v_c = 0$ at contact point
- No slipping, skidding or sliding
- No friction for rotation around contact point
- Steering axes orthogonal to the surface
- Wheels connected by rigid frame (chassis)





Forward Kinematic Model: Geometric Approach





Differential-Drive:

DDa) $v_{x1} = \frac{1}{2} r\dot{\varphi}_1 \quad ; \quad v_{y1} = 0 \quad ; \quad \omega_1 = \frac{1}{2l} r\dot{\varphi}_1$

DDb) $v_{x2} = \frac{1}{2} r\dot{\varphi}_2 \quad ; \quad v_{y2} = 0 \quad ; \quad \omega_2 = -\frac{1}{2l} r\dot{\varphi}_2$

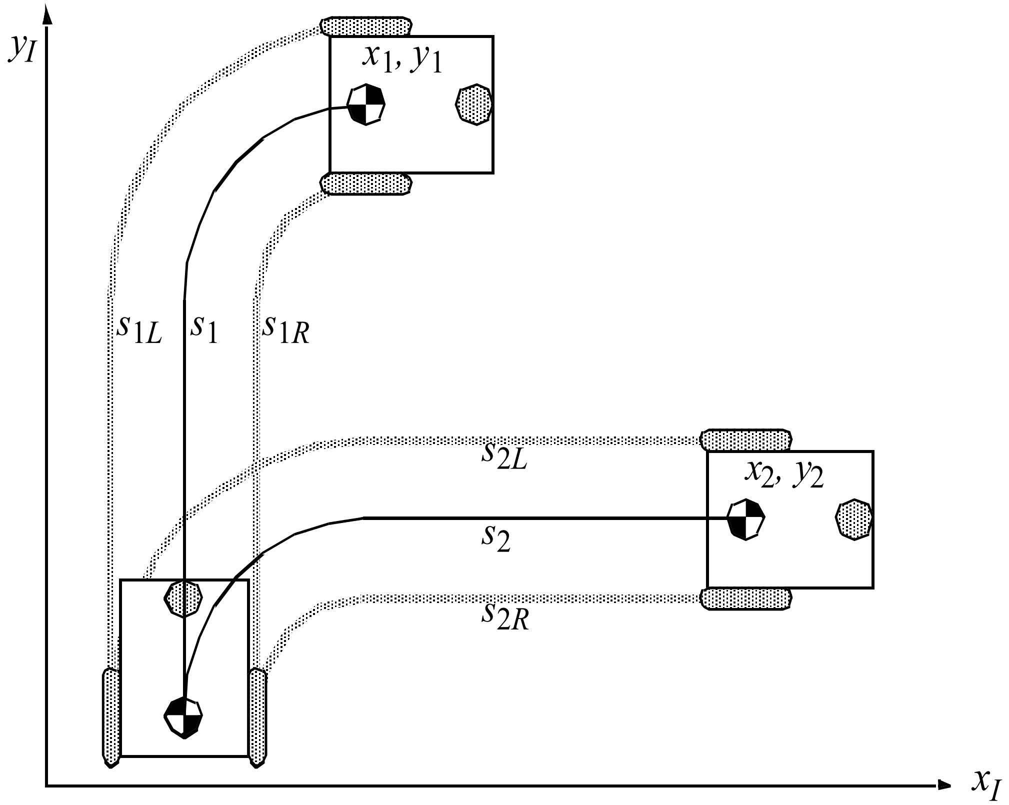
$$-> \dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}_I = R(\theta)^{-1} \begin{bmatrix} v_{x1} + v_{x2} \\ v_{y1} + v_{y2} \\ \omega_1 + \omega_2 \end{bmatrix} = R(\theta)^{-1} \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{bmatrix}$$

Inverse of R => Active and Passive Transform:
http://en.wikipedia.org/wiki/Active_and_passive_transformation

Mobile Robot Kinematics: Non-Holonomic Systems

$s_1 = s_2$; $s_{1R} = s_{2R}$; $s_{1L} = s_{2L}$

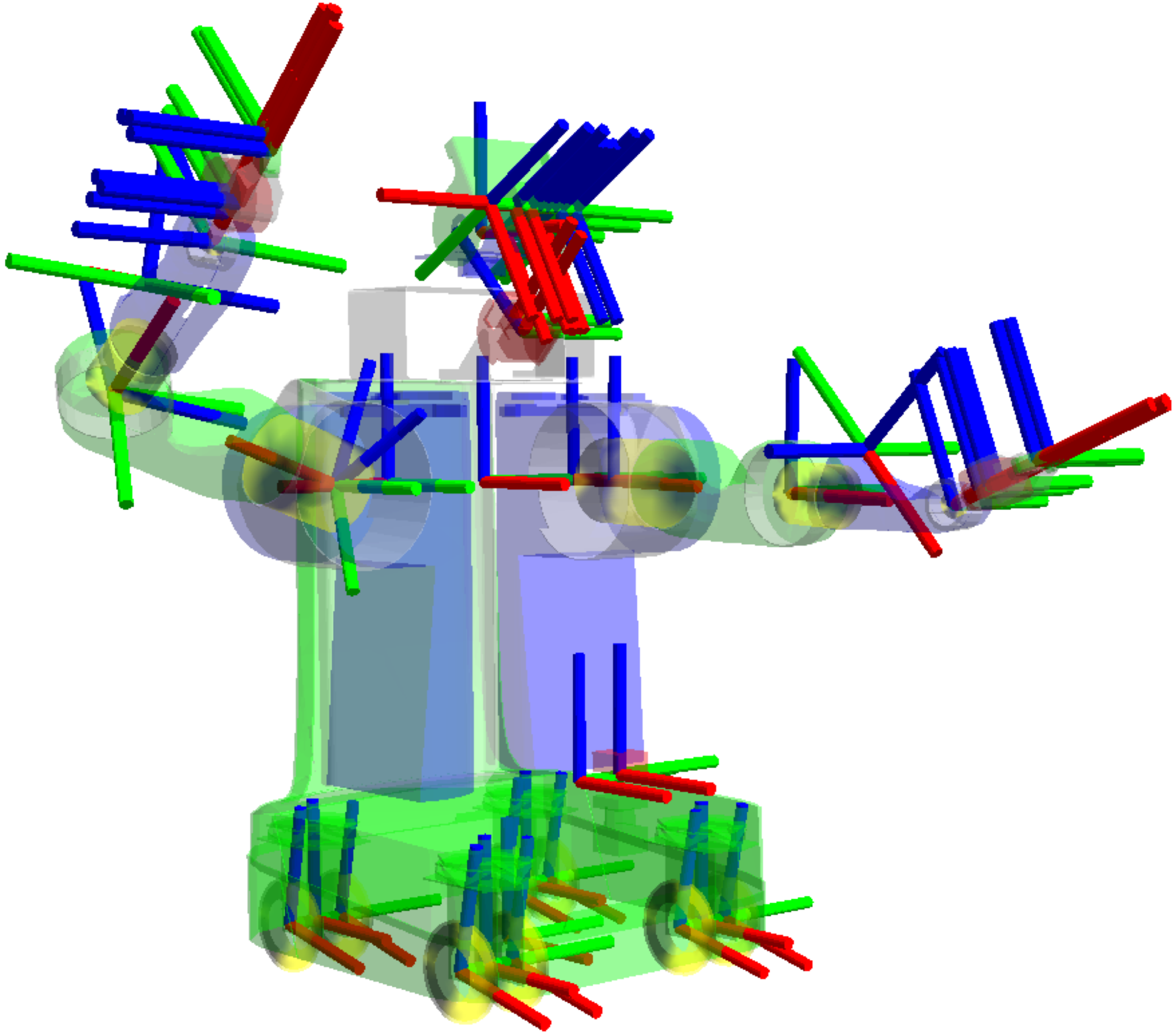
but: $x_1 \neq x_2$; $y_1 \neq y_2$



- Non-holonomic systems
  - differential equations are not integrable to the final position.
  - the measure of the traveled distance of each wheel is not sufficient to calculate the final position of the robot. One has also to know how this movement was executed as a function of time.

ROS: 3D Transforms : TF

- http://wiki.ros.org/tf
- http://wiki.ros.org/tf/Tutorials



ROS geometry_msgs/TransformStamped

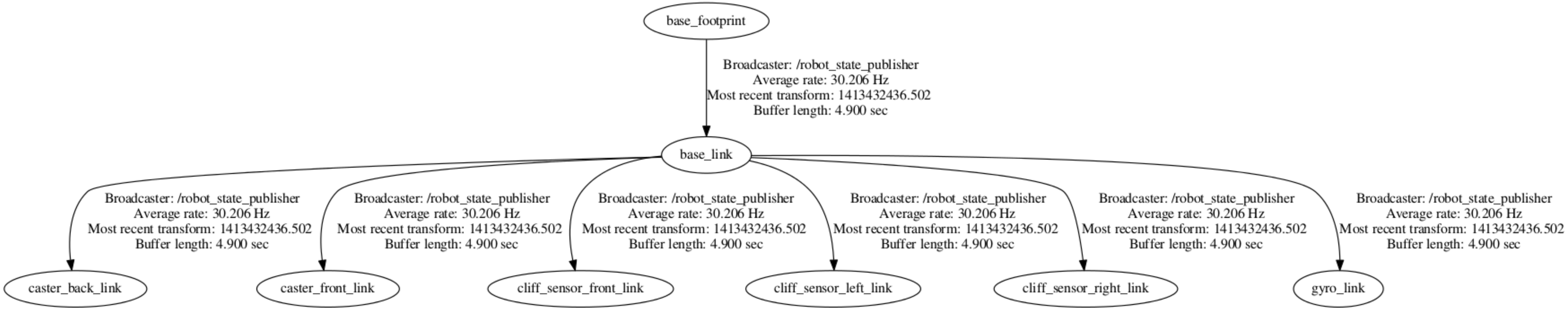
- $\begin{matrix} \text{header.frame\_id[header.stamp]} \\ \text{child\_frame\_id[header.stamp]} \end{matrix} \mathbf{T}$
- Transform between header (time and reference frame) and child_frame
- 3D Transform representation:
  - geometry_msgs/Transform:
    - Vector3 for translation (position)
    - Quaternion for rotation (orientation)

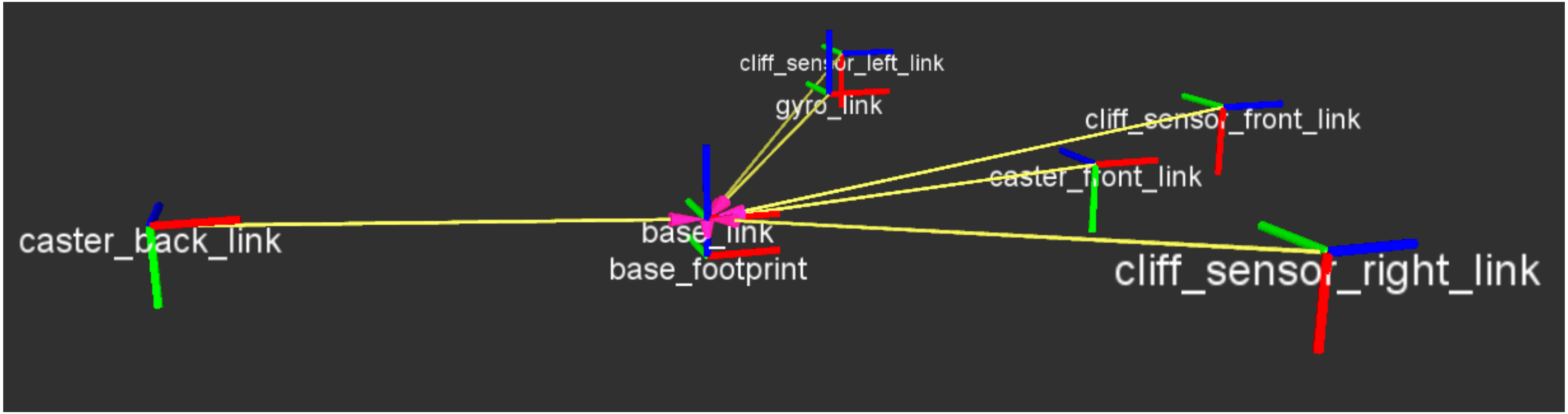
```
rosmsg show geometry_msgs/TransformStamped

std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/Transform transform
  geometry_msgs/Vector3 translation
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion rotation
    float64 x
    float64 y
    float64 z
    float64 w
```

ROS tf2_msgs/TFMessage

- An array of TransformStamped
- Transforms form a tree
- Transform listener: traverse the tree
    - tf::TransformListener listener;
- Get transform:
    - tf::StampedTransform transform;
    - listener.lookupTransform("/base_link", "/camera1", ros::Time(0), transform);
    - ros::Time(0): get the latest transform
    - Will calculate transform by chaining intermediate transforms, if needed

```
rosmsg show tf2_msgs/TFMessage

geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
  string child_frame_id
  geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion rotation
      float64 x
      float64 y
      float64 z
      float64 w
```


base_footprint
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
base_link
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
Broadcaster: /robot_state_publisher
Average rate: 30.206 Hz
Most recent transform: 1413432436.502
Buffer length: 4.900 sec
caster_back_link
caster_front_link
cliff_sensor_front_link
cliff_sensor_left_link
cliff_sensor_right_link
gyro_link


cliff_sensor_left_link
gyro_link
cliff_sensor_front_link
caster_front_link
caster_back_link
base_link
base_footprint
cliff_sensor_right_link

Transforms in ROS

- Imagine: Object recognition took 3 seconds – it found an object with:
  - tf::Transform object_transform_camera; // ${}^{\mathrm{Cam}[X]}_{Obj}\mathbf{T}$ (has tf::Vector3 and tf::Quaternion)
  - and header with: ros::Time stamp; // Timestamp of the camera image (== X)
    - and std::string frame_id; // Name of the frame (“Cam”)
- Where is the object in the global frame (= odom frame) “odom” ${}^{\mathrm{G}}_{Obj}\mathbf{T}$?
  - tf::StampedTransform object_transform_global; // the resulting frame
  - listener.lookupTransform(child_frame_id, “/odom", header.stamp, object_transform_global);
- tf::TransformListener keeps a history of transforms – by default 10 seconds

DEMO

PROJECT SELECTION….

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 deadline: Next Thursday (Sep 19)!
  - One member writes an email for the whole group to Long Xiaoling: longxl(at)shanghaitech.edu.cn ; Put the other group members on CC
  - Subject: [Robotics] Group Selection

- Project Descriptions:
  - https://star-center.shanghaitech.edu.cn/seafile/d/4adbcd21020e4c54a3df/

- 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

					Difficulty:			
# of Group	Name	Info File	Short Description	Advisor	Hardwa	Softwa	Algorith	aperCha
1	Rat detection	RoboRodent.pdf	Implement rat detection and state estimation from overhead RGB camera	Jiawei	low	med	med	med
1	Rat training GUI and simulator	RoboRodent.pdf	Implement a rat training software with GUI and Simulator	Jiawei	none	high	med	med
1 or 2	Life Science Fetch Robot	life_robot.pdf	Mobile manipulation; detect (transparent) test tubes; get tubes or beakers from shelf	Xiaoling	low	high	high	high
1	Event Camera	eventCamera.txt	Use the event cameras...	HongYu	none	med	high	med
1	Wifi Localization	wifi_localization.pdf	Work on localization the robot using WiFi	Haofei	low	med	med	med
1	Differential GPS	dgps.pdf	Work with dGPS. Use 4G to transmit the data	Xiaoling	low	high	low	none
1	dynamic obstacle filtering	dynamic.pdf	Filter dynamic obstacles from 3D LRF scans/ point clouds	Jiawei	none	med	med	low
1	underwater stereo SLAM	underwater.txt	3-camera stereo visual omnidirectional SLAM	Haofei	none	med	high	high
1	Factorization	factorization.txt	3D visual reconsruction from statellite	Qingwen	none	high	med	high
1	Rover SLAM	rover.txt	Use the CAS planetary rover: SLAM and autonomy	Hongyu	none	high	med	low
1	Rover Manipulation	rover.txt	Use the CAS planetary rover: sample collection	Xiaoling	low	high	med	low
1	New Mapping robot	mapper_II.txt	not available anymore.	Hongyu	high	high	low	high
1 or 2	Car project	car_project.txt	finish the smart "mapping" car	Hongyu	high	med	low	none
1 or 2	Elevator project	elevator.txt	autonomous elevator riding	Xiaoling	low	high	med	med
1	RoboCup Odometry	roboCupRescue.txt	Improve odometry	Qingwen	low	med	med	none
1	RoboCup Omni Camera	roboCupRescue.txt	omni camera visualization and VIO	Qingwen	med	high	med	low
1	RoboCup Negative Obstacles	roboCupRescue.txt	autonomy: don't fall down	Qingwen	none	med	high	low
?	Own		Your own project. Needs to be approved!	?	?	?	?	?