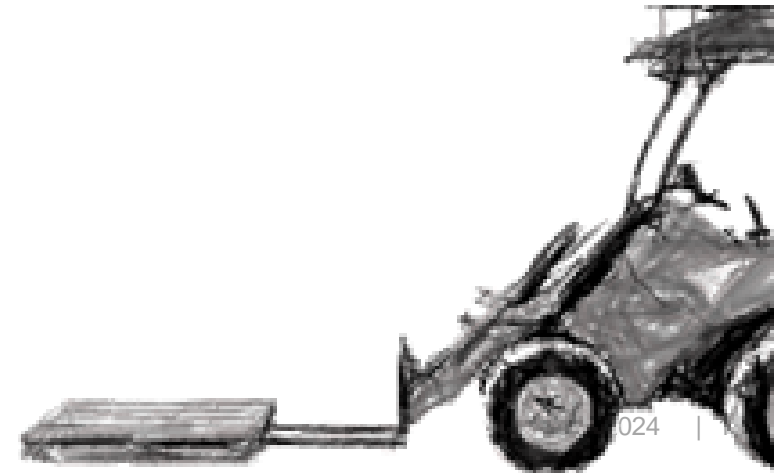


# Introduction to and review of some general concepts

Reza Ghabcheloo

AUT 710



# Concepts

- **Coordinate frames**
- **Kinematics modelling**
- **Dynamics systems and state space**
- Probabilities and distributions
- Map, world model

# week 1

# Mobile Robots vs Non mobile! AI Robotics vs Industrial robots!

Conventional robotics: Perfect world assumption

<http://www.kuka-robotics.com/>



Part of industrial robotics is moving away from perfect world assumption

Baxter cobot

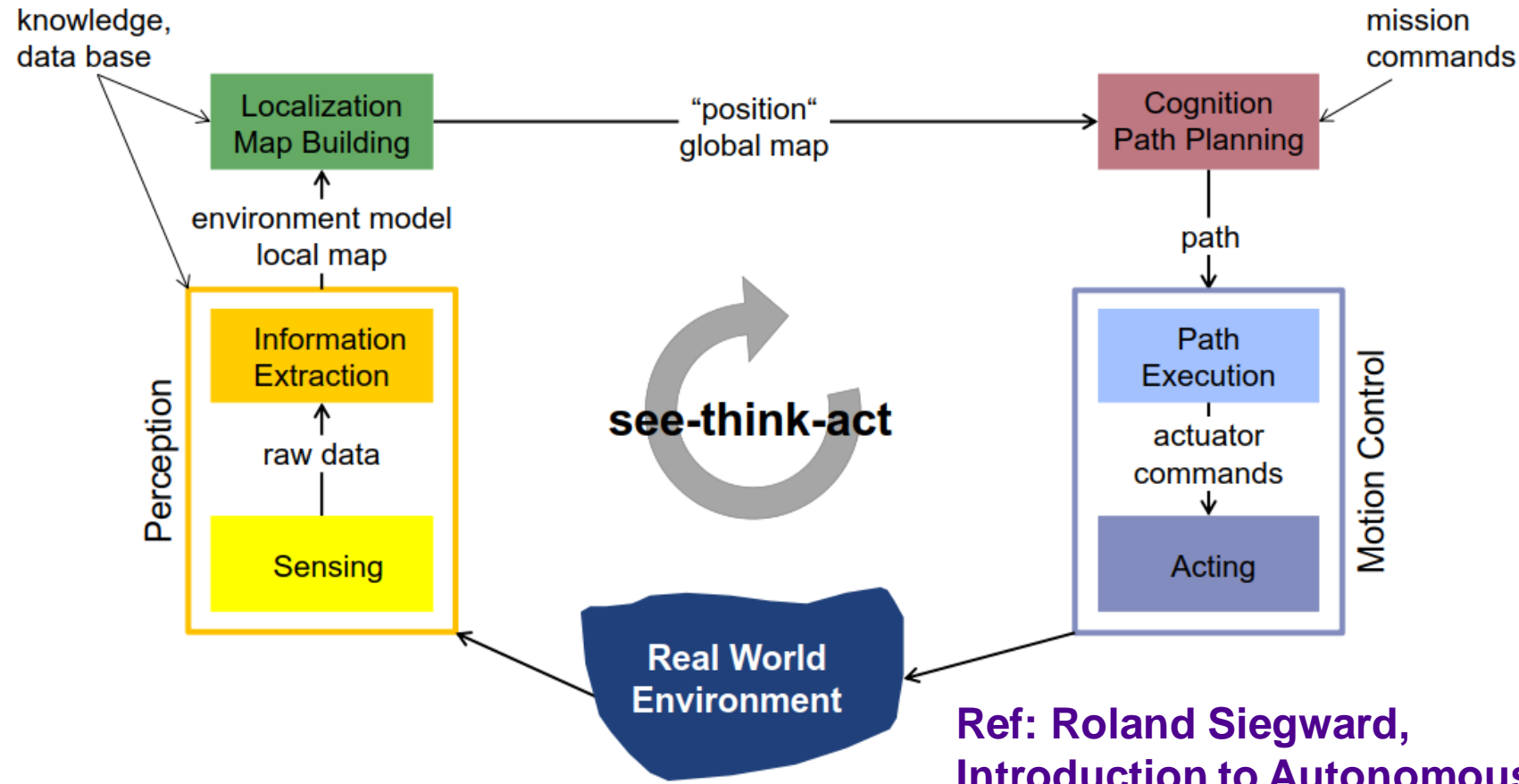


***Many of things we learn in this course also apply to manipulators. Though most of our examples are related to wheeled mobile robots.***

Modern needs: Dynamic and uncertain world



# See Think Act control architecture



Localization  
Path planning  
Control

Sensors  
Maps (knowledge,..)  
Goals (mission  
commands)

Ref: Roland Siegward,  
Introduction to Autonomous  
Mobile Robots

## Onboard:

Color, texture: vision, IR,...

Line, surface: Lidar, Radar, ...

Compliance: IMU, wheel contact,...

Other: radio activity, gas, algae...



Sensors

sensor data

front-end

feature extraction

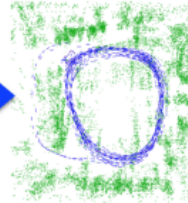
data association:

- short-term (feature tracking)
- long-term (loop closure)

back-end

MAP estimation

SLAM estimate



Pose

World modeling

Decision making

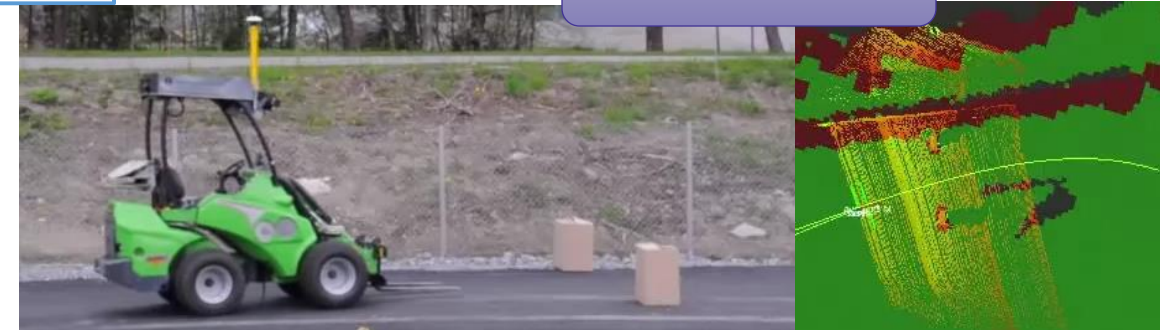
- collision free planning
- Intervention planning
- maintenance planning
- harvesting

Motion control

GNSS, UWB,  
IMU,  
Vision, Lidar,

Elevation maps,  
other vehicles,  
Infra (weather)  
sensors

Third party data



MAP > Maximum a posteriori  
Bayes filter (KF, MCF)

# Background knowledge

- We need to describe robot position with respect to some frame of reference or origin
- We need to define our goals in the same frame
- We need to define the obstacle and world model in the same frame

❑ So, to talk to robot and robots to each other, we need **coordinate frames**. We also need them for other things!

❑ Positions and velocities describe our robot in 2D or 3D. We need **vectors**.

❑ We will need to transform these vectors from one frame to another, back and forth all the time! **Matrices** play a key role, such as rotation matrix. Thus, **Linear algebra**

❑ Robots are physical bodies moving in space. We will describe their motion using **ordinary differential equations (ODE)**. ODEs and motion modeling play a key role in controller and localization design.

# ODEs and Linear algebra

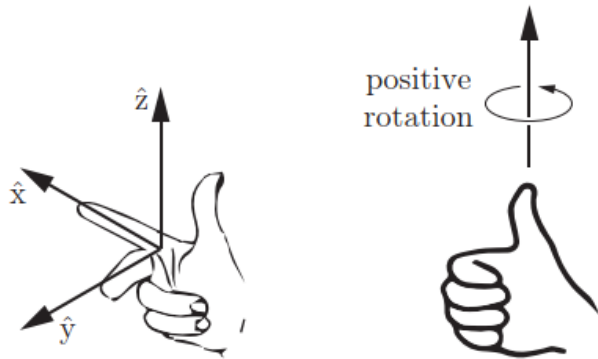
These are nice short videos to refresh your mind:

- <https://www.3blue1brown.com/topics/linear-algebra>
- <https://www.3blue1brown.com/lessons/differential-equations>
- I will review certain things in the first two weeks.



# Coordinate frames

# Right-hand coordinate frame



**Figure 3.2:** (Left) The  $\hat{x}$ ,  $\hat{y}$ , and  $\hat{z}$  axes of a right-handed reference frame are aligned with the index finger, middle finger, and thumb of the right hand, respectively. (Right) A positive rotation about an axis is in the direction in which the fingers of the right hand curl when the thumb is pointed along the axis.

Right-hand coordinate, that is,

$$\hat{x} \times \hat{y} = \hat{z}$$

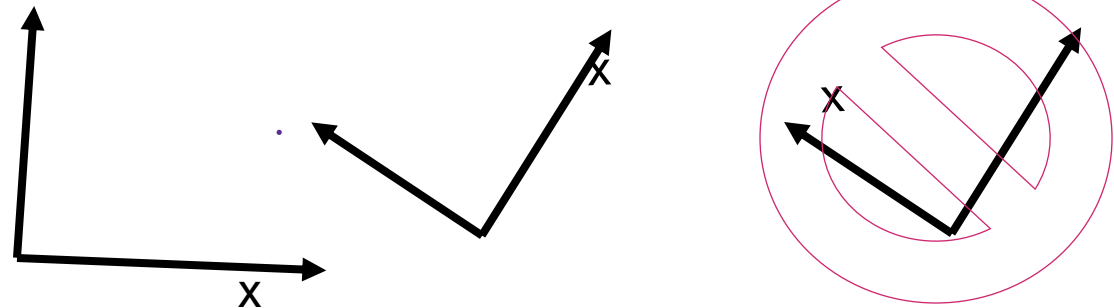
$$\hat{y} \times \hat{z} = \hat{x}$$

$$\hat{z} \times \hat{x} = \hat{y}$$

$\times$  is cross product.

$\hat{\phantom{x}}$  denotes unit vector

- If you want to work on 2D, then we only need two of these three. Let's say x-y. Then you need to make sure all the third coordinates (z-axis) point up or all point down. Although you do not use or draw them, they are still there!
- In this course, we will remain on 2D!



# Models and coordinate frames

modeling	term
a sensor to another sensor	Extrinsic calibration
a robot link to another	Kinematics modeling
a sensor to robot body	Extrinsic calibration
a point of the environment to a sensor	Perception, object detection, object pose estimation
robot body to a reference frame	Localization / robot pose estimation

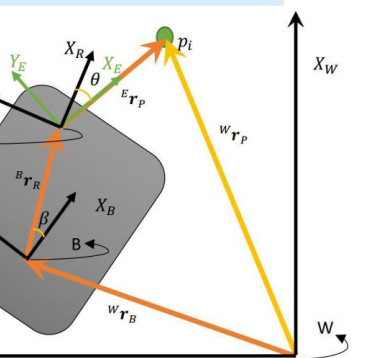
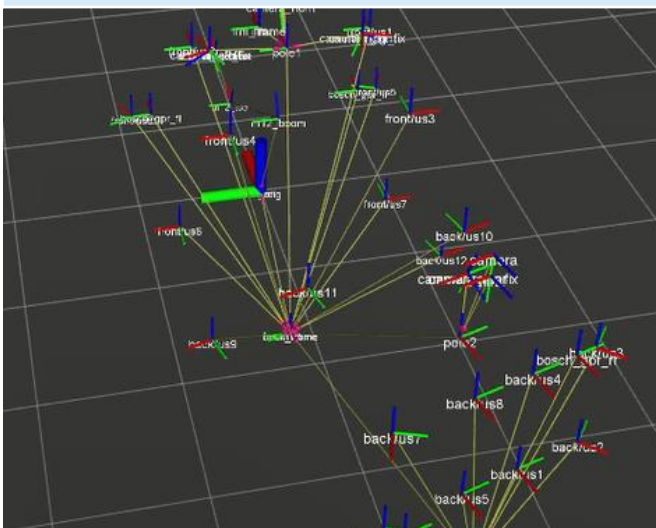
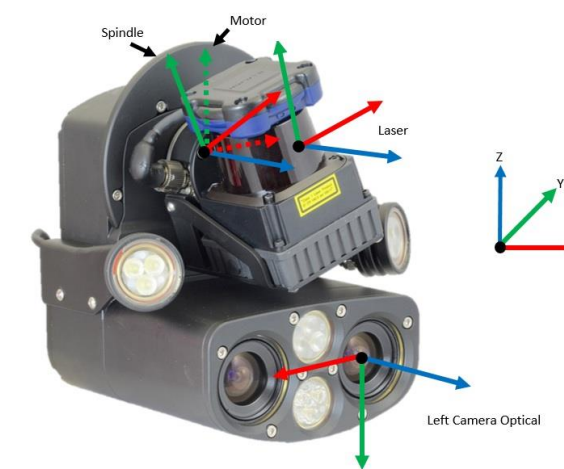
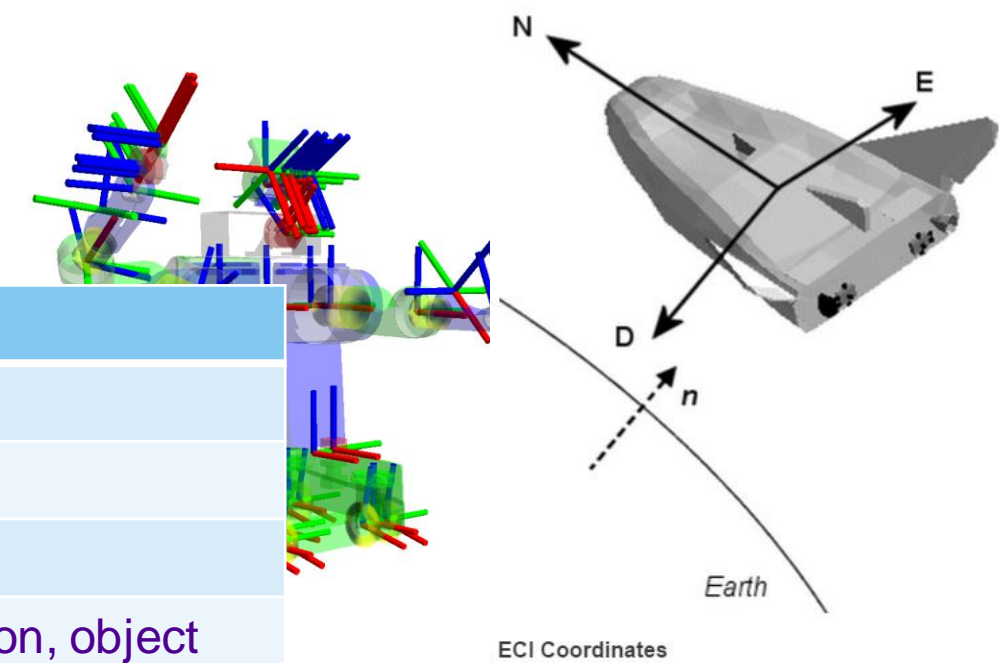
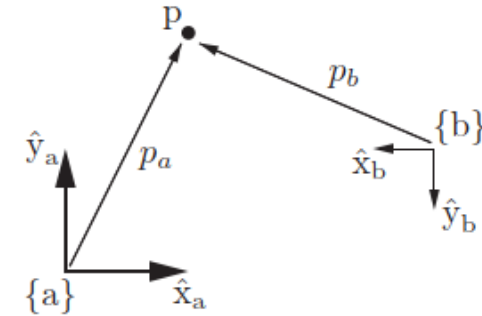


Figure 27: Point  $p_i$  represented in radar frame  $R$ , body frame  $B$  and world frame  $W$



# Points and vectors

- $p$  : a point in a physical space (note:  $p$  is not italic)
- expressed & seen from frame  $\{a\}$ , denoted  $p_a \in R^3$
- $v$  : a free vector in a physical space (only length and direction matters, but not rooted anywhere; for example linear velocity vector)
- Expressed and seen from frame  $\{a\}$ , denoted  $v_a \in R^3$
- $p_a$  is then a vector rooted at the origin of  $\{a\}$



**Figure 3.1:** The point  $p$  exists in physical space, and it does not care how we represent it. If we fix a reference frame  $\{a\}$ , with unit coordinate axes  $\hat{x}_a$  and  $\hat{y}_a$ , we can represent  $p$  as  $p_a = (1, 2)$ . If we fix a reference frame  $\{b\}$  at a different location, a different orientation, and a different length scale, we can represent  $p$  as  $p_b = (4, -2)$ .

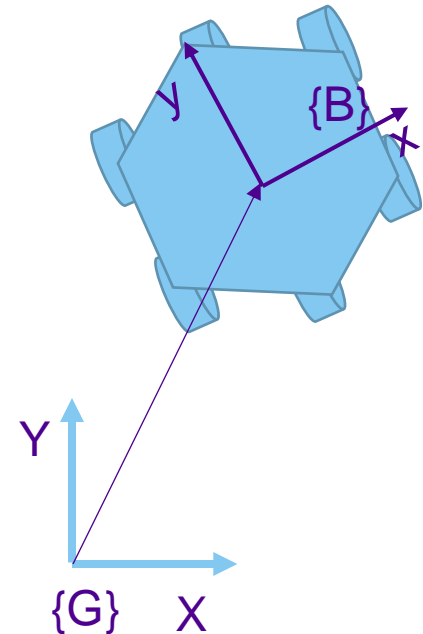
ROS keeps track of the frames and their relations using tf package.

[ROS tf tree](#)

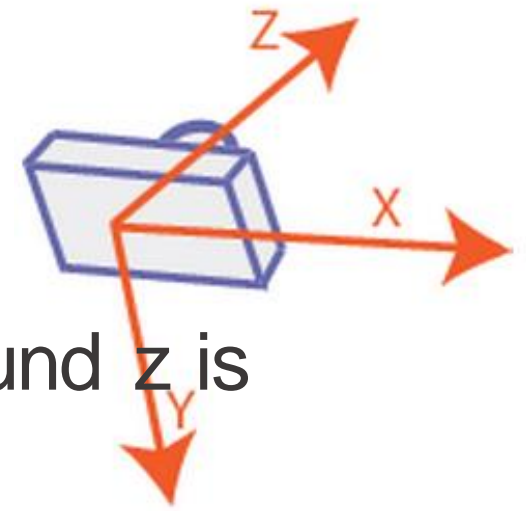
# Robot body frame

## Sensor frame

- We are concerned with both position and orientation of our robot.
- Orientation matters also for many of the sensors like cameras.
- Therefore, in 2D world, we need
  - Position vector  $(x, y)$
  - Heading angle  $\psi$
- $\{B\}$  body frame
- $\{G\}$  global frame, fixed to robot environment



# 2D world



- Right hand coordinate system: positive rotation around  $z$  is from  $x$  to  $y$

Q: my nose pointing at  $X$ , my head pointing at  $Z$ , what is  $Y$ ?

- Position: North-East-Down or  $X$ - $Y$ - $Z$

Q: How would you plot NE frames in Matlab?

- Orientation: we will use 'ZYX' order of Euler angles (roll, pitch, yaw); study of robots in 2D world:  $roll=0$ ,  $pitch=0$ ,  $\psi$ =heading or yaw. Or simply only rotation around  $z$ -axis
  - In this class, we will typically use  $XY$  coordinates, such that  $Z$  points away from screen.
- Q: what is positive rotation, CW or CCW?

# Coordinate systems

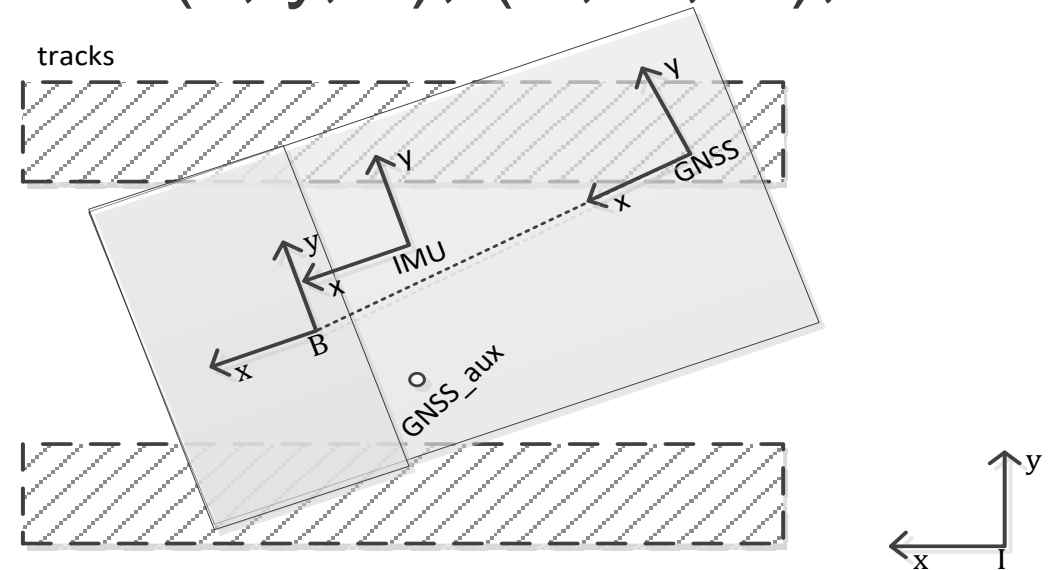
- For robots moving in small space, we usually use a simple system: a flat world with a world frame, with respect to which all other frames are defined
- For robots moving on large field (aircrafts) flat world assumption is not valid.

# 3D space, position

- Flat world
  - Position represented by 3 variables ( $x, y, z$ ), ( $N, E, D$ ), etc in distance unit
- Non flat
  - Latitude, longitude, altitude



<https://novatron.fi/xsite-pro/>



Q: what is the third coordinate in the diagram?



# Orientation is more involved

- For frames with limited rotational motions, we may use Euler angles with 3 variables
  - Euler angles (roll, pitch, yaw)
  - For large motions, Euler angles will be ambiguous.
- Otherwise, use
  - quaternions  $q = (q_1, q_2, q_3, q_4)$
  - direction cosine matrix (rotation matrix)  
$$R \in SO(3), RR^T = I, \det(R) = 1$$

# Extra reading

## Earth related coordinate systems

- Matlab/Simulink Aerospace blockset has a very good overview

<https://se.mathworks.com/help/aeroblks/about-aerospace-coordinate-systems.html>

# State, motion models, ODE

# Dynamical systems and motion models

- We use ODEs to describe motion of the robot or sensor (in fact the motion of the frame attached to the body!)
- The models are written in number of coupled first order ODEs
  - $\dot{x}_1 = f_1(x_1, x_2, \dots, u_1, \dots)$
  - $\dot{x}_2 = f_2(x_1, x_2, \dots, u_1, \dots)$
  - $\dot{x}_i$  denote derivative of  $x_i$  with respect to time,  $i=1,2,\dots$
  - $u_i$  denote control variables  $i=1,2,\dots$
  - $x_i$ 's are called system states; they represent the memory of the dynamical system of its past- Because, to predict the trajectory of your robot in the future, you just need the current state and the inputs that are going to be applied. You do not need the past states or inputs.
  - That is why it is called state space model.
- For brevity, we typically write above models in vector forms. That is  $\dot{x} = f(x, u)$ ,

$$\text{where } x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \end{bmatrix}, u = \begin{bmatrix} u_1 \\ \dots \end{bmatrix}, f(x, u) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dots \end{bmatrix} = \begin{bmatrix} f_1(x, u) \\ f_2(x, u) \\ \dots \end{bmatrix}$$

Programmatically, code is typically more readable and might be more efficient in matrix form. But not always possible to code them in matrix form.

# Example, 1D robot



- $m\ddot{x} + b\dot{x} = \tau$
- $m$  (mass),  $b$  (viscos friction),  $\tau$  (control input, force),  $x, \dot{x}, \ddot{x}$  (position, speed, acceleration)
- Now we want to convert this to a state space mode.
- $x_1 = x, x_2 = v, u = \tau$  ← one control (force), two states (position and speed)
- Motion model:
  - $\dot{x}_1 = x_2$
  - $\dot{x}_2 = -\frac{b}{m}x_2 + \frac{1}{m}u$
  - State vector  $x = \begin{bmatrix} x \\ v \end{bmatrix}$
- *For the purpose of this course, we will mainly deal with only kinematics and no dynamics. That is, forces and masses are not considered, and motions are described by speeds as control variables, and positions as states. Then your model is simply  $\dot{x}_1 = v$ ,*

← Coupled first order ODEs

# What can we do this the motion model?

- Motion model: 
$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{b}{m}x_2 + \frac{1}{m}u \end{cases}$$

We can now think of two cases:

- Simulating the robot: given initial state and control input, what is the trajectory robot will take. This is done by integrating the above equations. This is one step in localization.
  - This will use in Part 1
- Control of the robot (the inverse problem): given a desired trajectory and the robot state, what is the control that keeps my robot on the path.
  - You will learn this in Part 2

# Simulating the robot


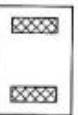

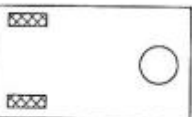
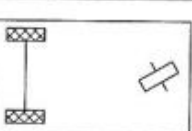
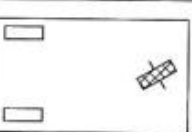

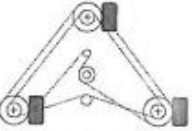
- Recall how we integrate  $x(t) = x(t_0) + \int_{t_0}^{t_f} f(t) dt$ , where  $\dot{x} = f(t)$
- To solve the motion model: 
$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{b}{m} x_2 + \frac{1}{m} u \end{cases}$$

*we just need to know initial state, the control values and use some numerical ODE solver (e.g. runge kutta)*

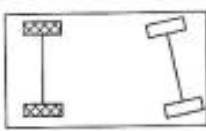
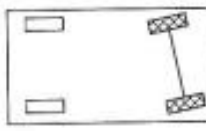


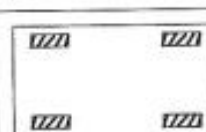

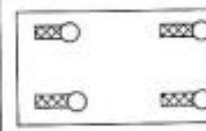
- input  $u(t)$ :  $t \in [t_0, t_f]$  and initial states  $x(t_0) = \begin{bmatrix} x(t_0) \\ v(t_0) \end{bmatrix}$
- For our purpose, we will typically use simple Euler discretization, then it is easy to calculate [https://en.wikipedia.org/wiki/Euler\\_method](https://en.wikipedia.org/wiki/Euler_method)

# Mobile robot kinematic models



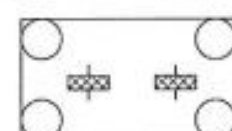

# of wheels	Arrangement	Description	Typical examples
2		One steering wheel in the front, one traction wheel in the rear	Bic
		Two-wheel differential drive with the center of mass (COM) below the axle	Cye
3		Two-wheel centered differential drive with a third point of contact	Non EPF
		Two independently driven wheels in the rear/front, one unpowered omnidirectional wheel in the front/rear	Man incl Pyg
		Two connected traction wheels (differential) in rear, one steered free wheel in front	Piag
		Two free wheels in rear, one steered traction wheel in front	Nept Univ
		Three motorized Swedish or spherical wheels arranged in a triangle; omnidirectional movement is possible	Stan Trib Palm (CM)
		Three synchronously motorized and steered wheels; the orientation is not controllable	"Syn Denn Instit Robo

Wheel configurations for rolling vehicles








# of wheels	Arrangement	Description
4		Two motorized wheels in the rear, two steered wheels in the front; steering has to be different for the two wheels to avoid slipping/skidding.
		Two motorized and steered wheels in the front, two free wheels in the rear; steering has to be different for the two wheels to avoid slipping/skidding.
		Four steered and motorized wheels
		Two traction wheels (differential) in rear/front, two omnidirectional wheels in the front/rear
		Four omnidirectional wheels
		Two-wheel differential drive with two additional points of contact
		Four motorized and steered castor wheels

# Steering and drive mechanisms

Wheel configurations for rolling vehicles

# of wheels	Arrangement	Description	Typical examples
6		Two motorized and steered wheels aligned in center, one omnidirectional wheel at each corner	First
		Two traction wheels (differential) in center, one omnidirectional wheel at each corner	Terregator (Carnegie Mellon University)

Icons for the each wheel type are as follows:

	unpowered omnidirectional wheel (spherical, castor, Swedish)
	motorized Swedish wheel (Stanford wheel)
	unpowered standard wheel
	motorized standard wheel
	motorized and steered castor wheel
	steered standard wheel
	connected wheels

Ref: Roland Siegward,  
Introduction to Autonomous  
Mobile Robots

# Robot states and inputs

- Recall that our simple 1D robot with force as control input, has two states: position and speed.
- For the purpose of this course, we will mainly deal with only kinematics and no dynamics. That is, forces and masses are not considered, and motions are described by speeds as control variables, and positions as states
- In this case, the robot state is described by robot pose = position and orientation. That is, a vector of three variables  $(x, y, \psi)$  in 2D world.
- In 3D, we will need a frame which includes three variable position and orientation will depend on our representation. (3 Euler angles, 4 quaternion, ...)

# Omni-directional robot kinematics

- Omni-directional robot equation of motion

$$\dot{x} = v_{xG}$$

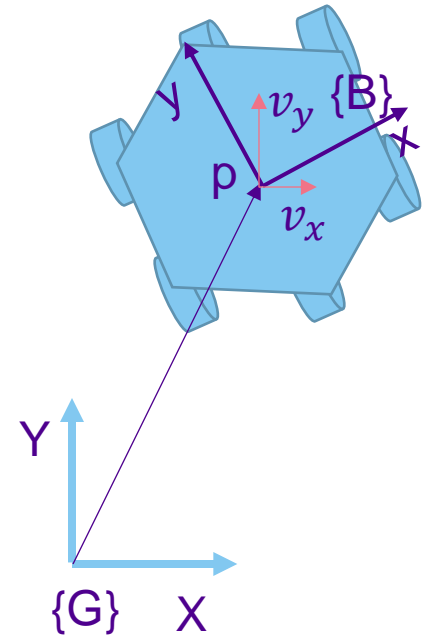
$$\dot{y} = v_{yG}$$

$$\dot{\psi} = \omega_z$$

Robot state / pose  $(x, y, \psi)$

Low level control servos will provide us three independent control variables  $(v_{xG}, v_{yG}, \omega_z)$ , defined in global reference frame  $\{G\}$ .

- [Link](#)



# Differential drive kinematics (unicycle model)

Equations of motion is given by

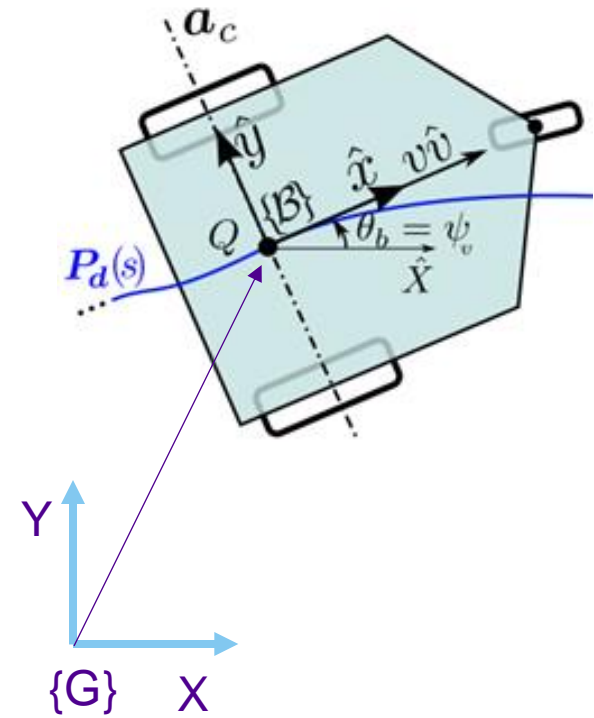
$$\dot{x} = v_{xB} \cos \psi$$

$$\dot{y} = v_{xB} \sin \psi$$

$$\dot{\psi} = \omega_z$$

Robot state / pose  $(x, y, \psi)$

where low level wheel speed servos will provide independent inputs  $(v_{xB}, \omega_z)$  defined in body frame  $\{B\}$ .  $v_x$  linear speed along body x-axis,  $\omega_z$  angular speed around z-axis. We typically call  $(v_x, \omega_z)$  twist command.



# Relating body speeds to wheel speed

$$\dot{x}_B = v_{xB} \cos \psi$$

$$\dot{y}_B = v_{xB} \sin \psi$$

$$\dot{\psi} = \omega_{zB}$$

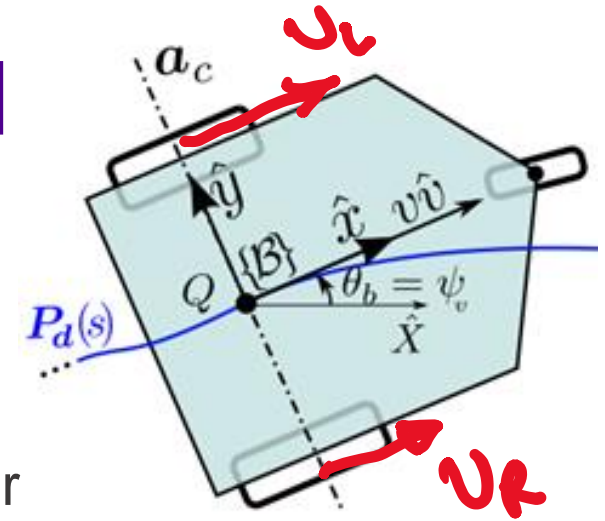
relation between body frame speeds ( $v_x, \omega_z$ ) and wheel angular speeds ( $w_R, w_L$ )

$$v_x = \frac{1}{2}(v_R + v_L)$$

$$\omega_z = \frac{1}{d}(v_R - v_L)$$

$$v_R = r w_R, v_L = r w_L$$

where  $r$  is the radius of the wheels, and  $d$  distance between wheels, and  $w_R$  and  $w_L$  are rotational speed of the wheels.



# Localization and Control

We will use these equations of motions for both **control** and **localization**.

- Control: what should be my next  $v_x$  and  $\omega_z$  to get me closer to the goal and avoid obstacles.
- Odometry based localization: knowing  $v_x$  and  $\omega_z$  and initial conditions  $x_B(0), y_B(0), \psi(0)$ , integrate the kinematic equations to get  $x_B(t), y_B(t), \psi(t)$ 
  - In this case,  $v_x$  and  $\omega_z$  could be the commands we send to the servo controllers or measured from wheel speed. Notice these two are not exactly the same, since due to actuator dynamics and ground friction they will not be executed instantaneously nor perfectly