上海科技大学

ShanghaiTech University

CS283: **Robotics Fall 2016: Sensors**

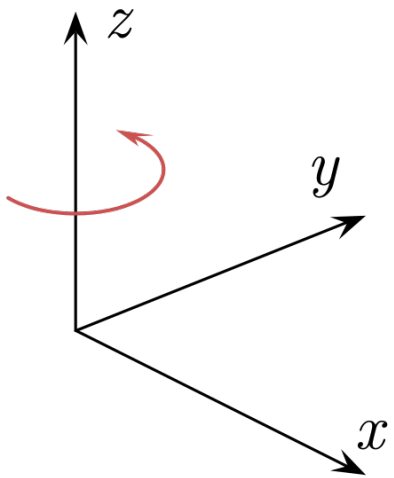
Sören Schwertfeger / 师泽仁

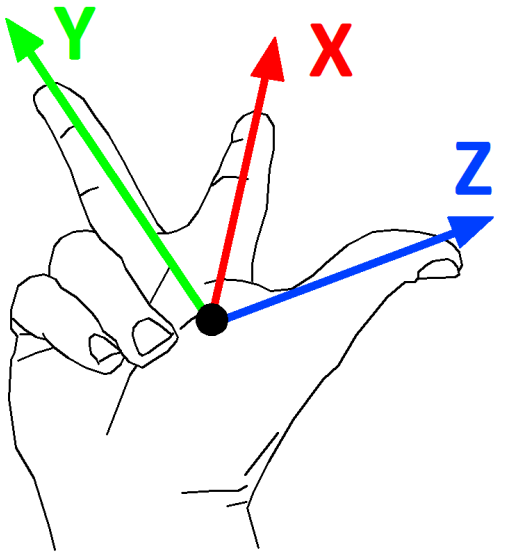
ShanghaiTech University

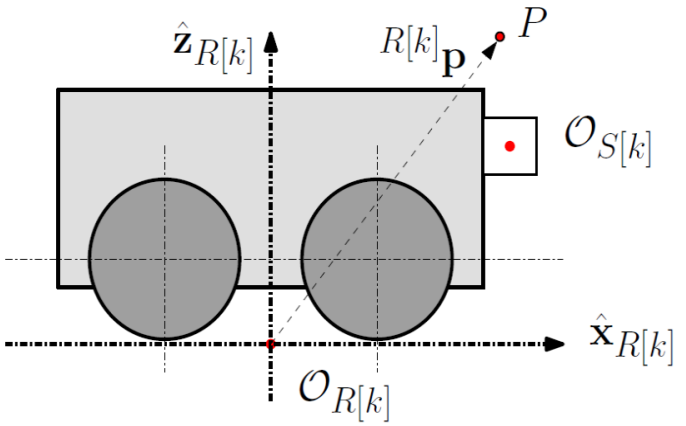
REVIEW TRANSFORMS

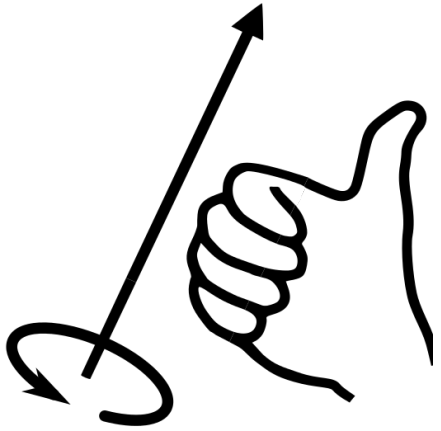
Right Hand Coordinate System

- 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 ???



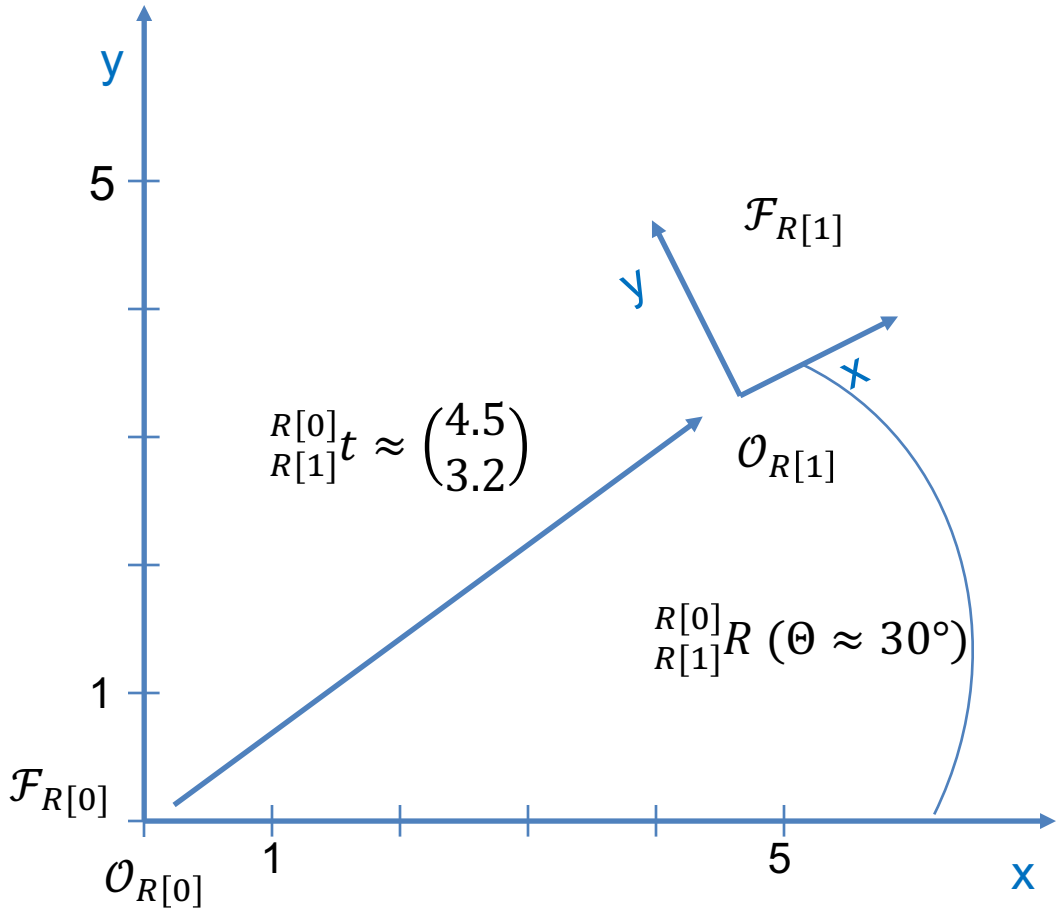






Right Hand Rule http://en.wikipedia.org/wiki/Right-hand_rule

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

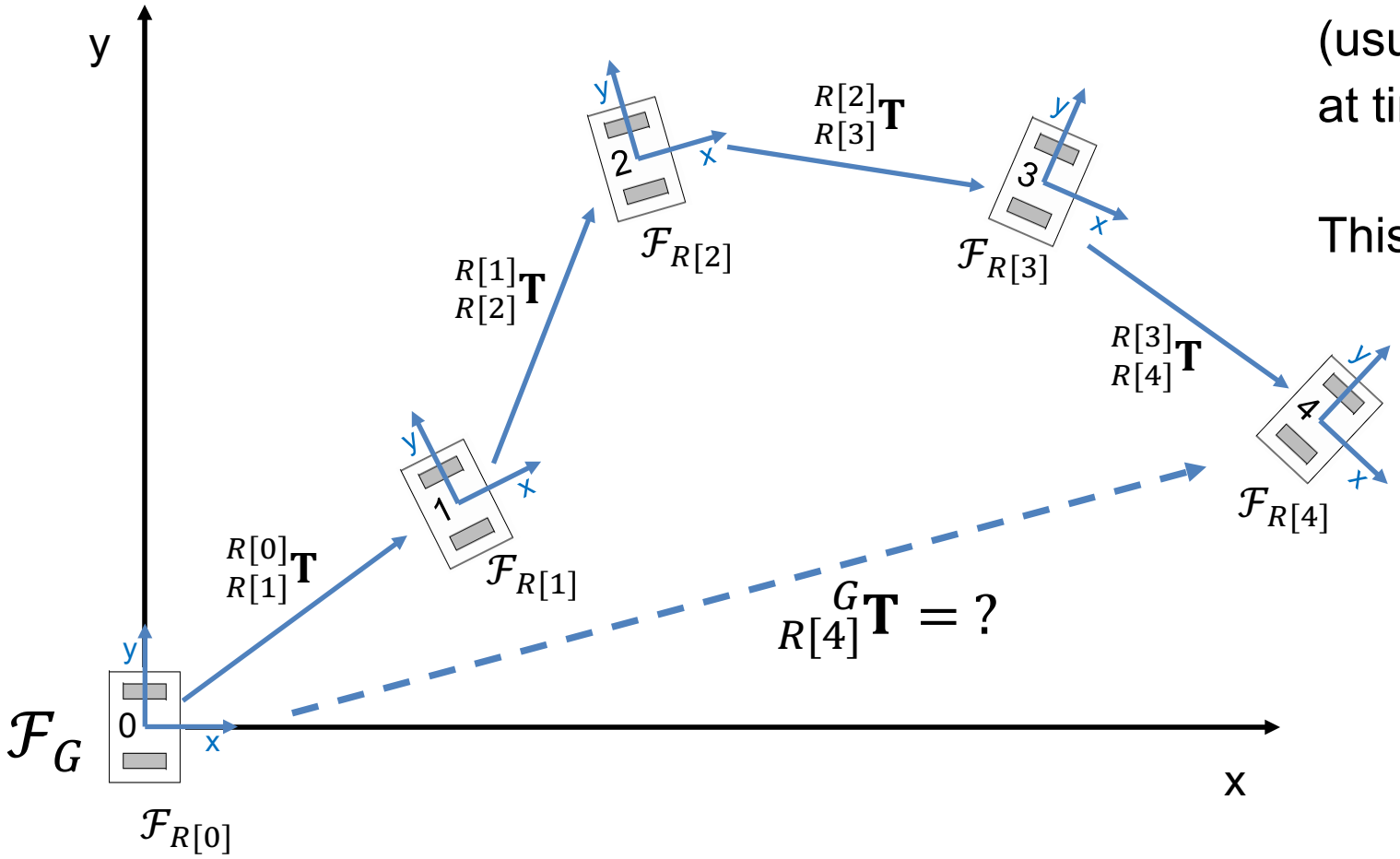
    to $\mathcal{O}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

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

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

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:

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

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

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

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

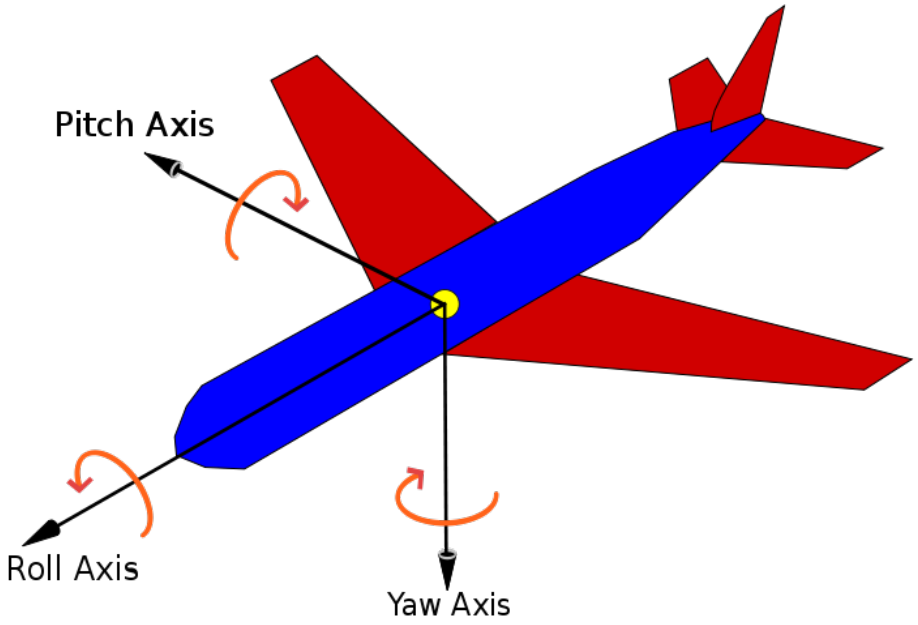
$$ {}^{R[X]}_{R[X+1]}t_y $$

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

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



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

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

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

Transform in 3D

Matrix Euler Quaternion

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

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

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

Rotation Matrix 3x3

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

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

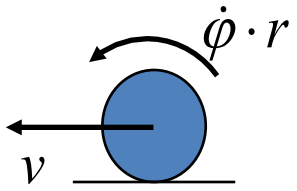
$$ R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

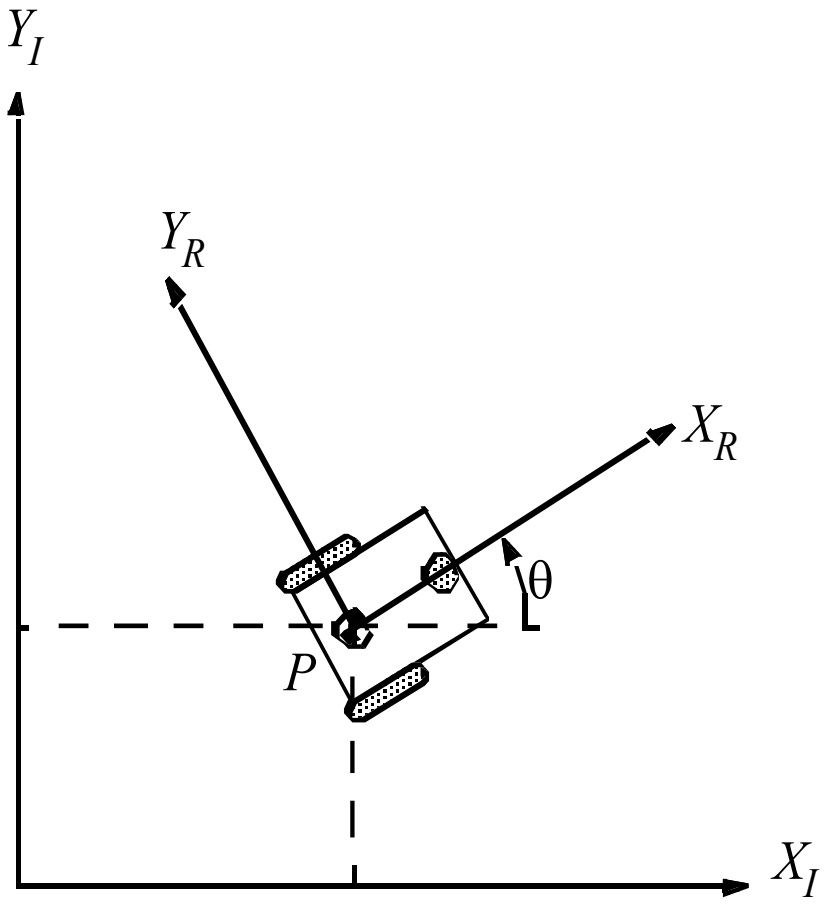
$$ R = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma) $$

yaw = α, pitch = β, roll = γ

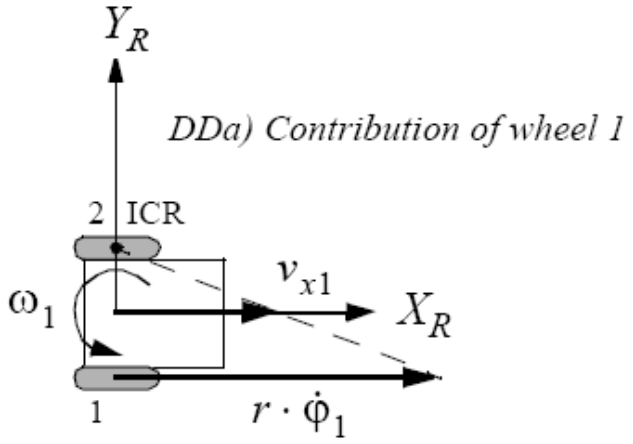
Wheel Kinematic Constraints: Assumptions

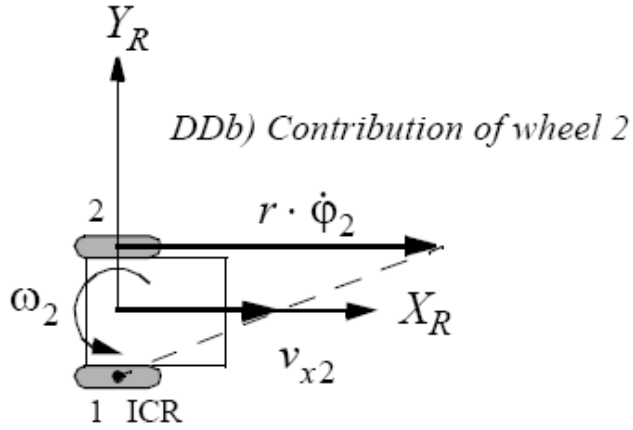
- Movement on a horizontal plane
- Point contact of the wheels
- Wheels not deformable
- Pure rolling
  - $v_c = 0$ at contact point
- No slipping, skidding or sliding
- No friction for rotation around contact point
- Steering axes orthogonal to the surface
- Wheels connected by rigid frame (chassis)





Forward Kinematic Model: Geometric Approach





Differential-Drive:

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

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

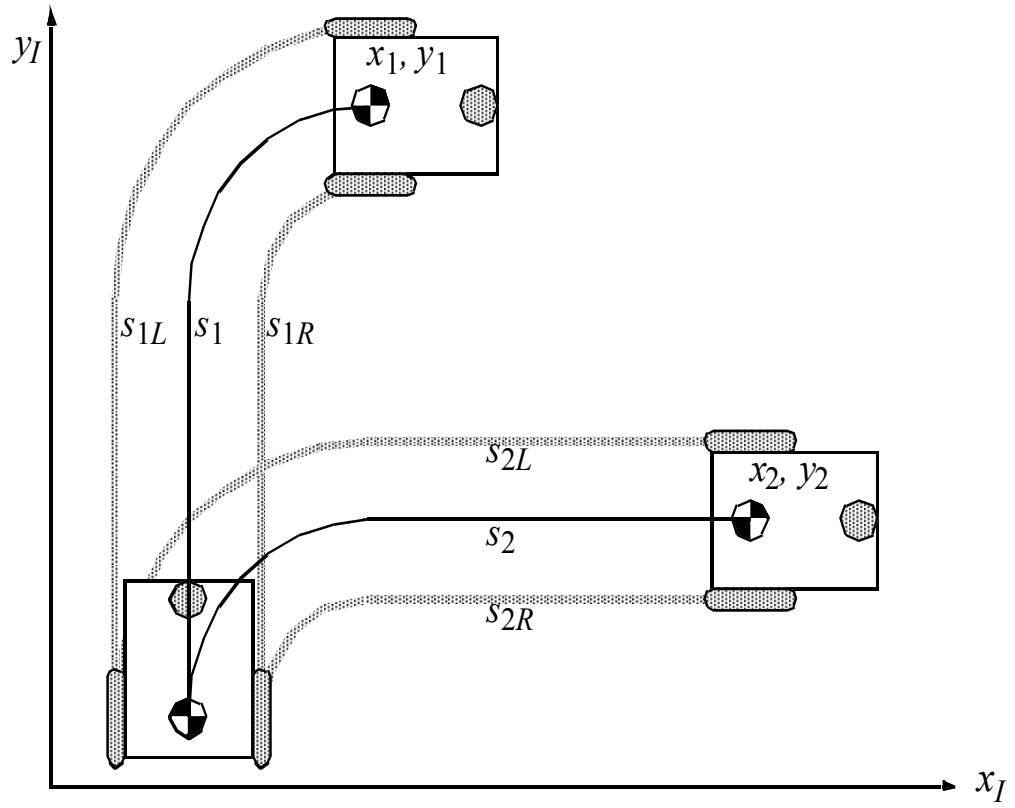
$$\rightarrow \dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}_I = R(\theta)^{-1} \begin{bmatrix} v_{x1} + v_{x2} \\ v_{y1} + v_{y2} \\ \omega_1 + \omega_2 \end{bmatrix} = R(\theta)^{-1} \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix}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

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

Mobile Robot Kinematics: Non-Holonomic Systems

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

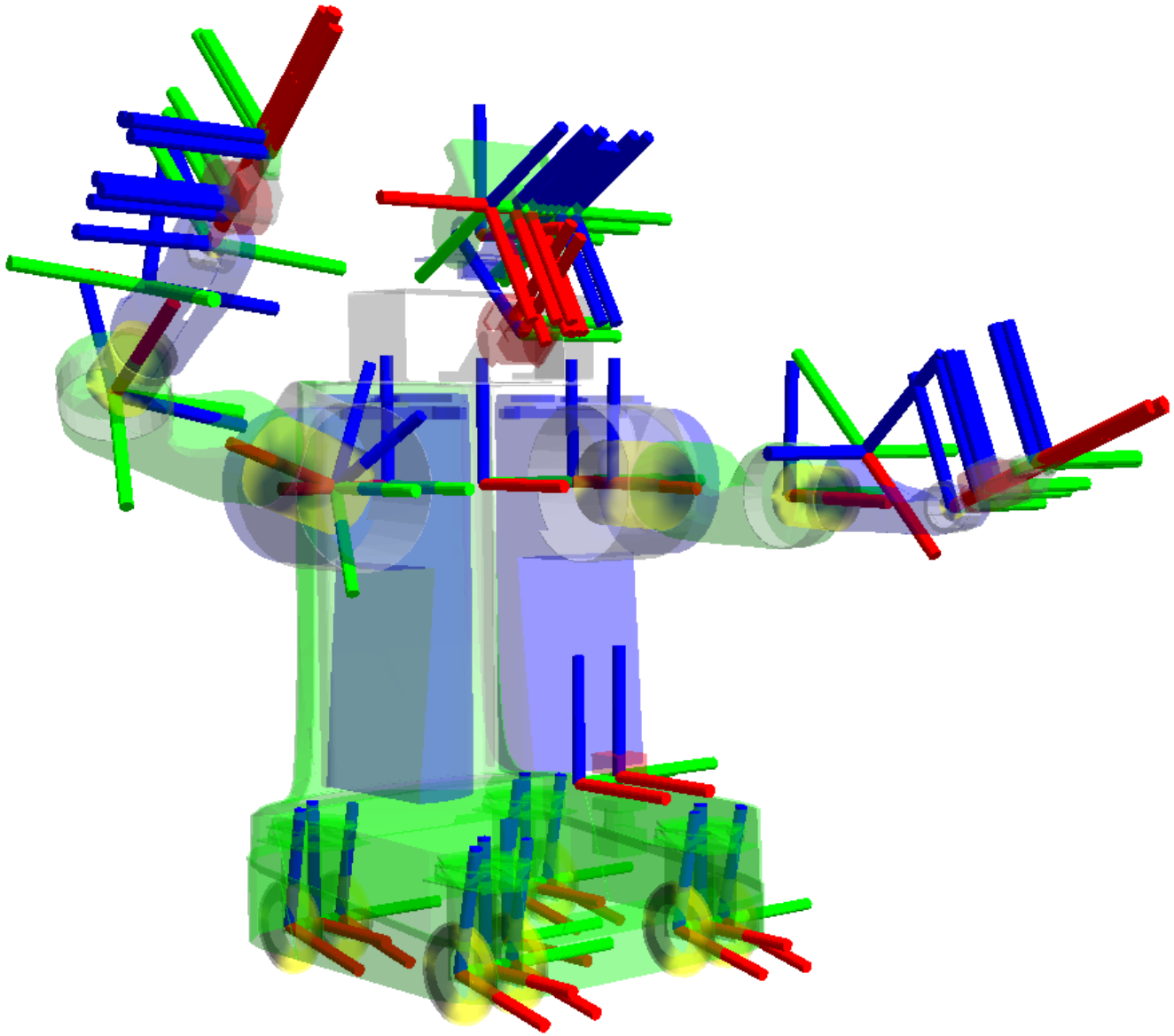
but: $x_1 \neq x_2$; $y_1 \neq y_2$



- 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: Transforms : TF

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



ROS geometry_msgs/TransformStamped

- $\mathbf{T}_{\text{header.frame\_id[header.stamp]}}^{\text{child\_frame\_id[header.stamp]}}$

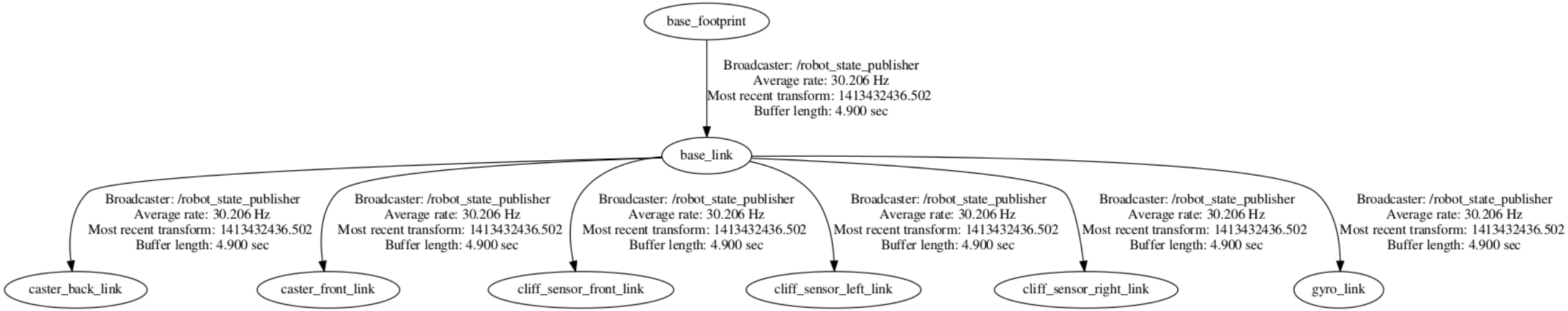
- Transform between header (time and reference frame) and child_frame
- 3D Transform representation:
  - geometry_msgs/Transform:
    - Vector3 for translation (position)
    - Quaternion for rotation (orientation)

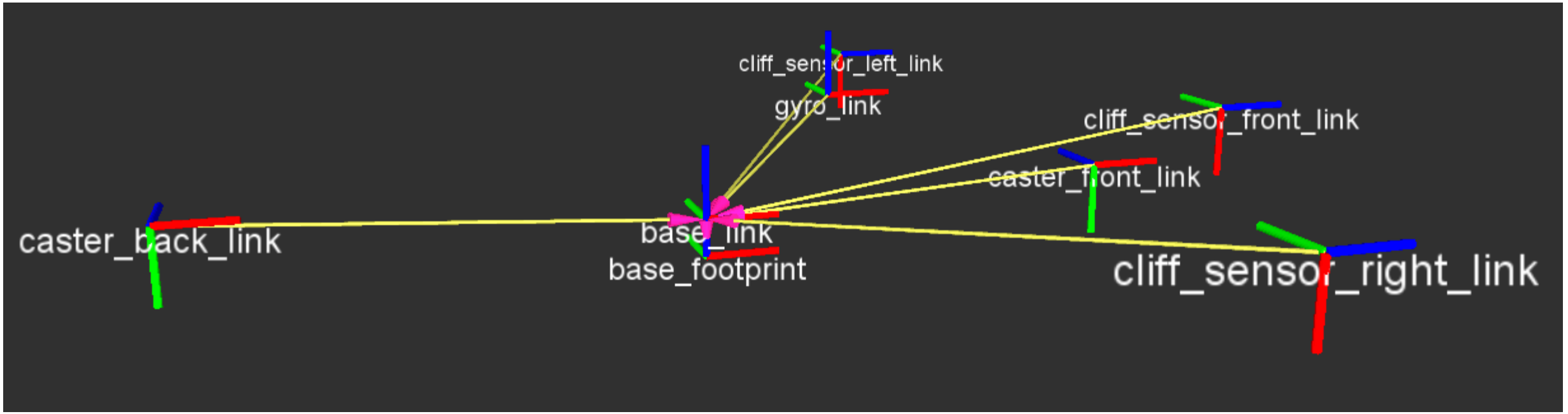
```
rosmsg show geometry_msgs/TransformStamped

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

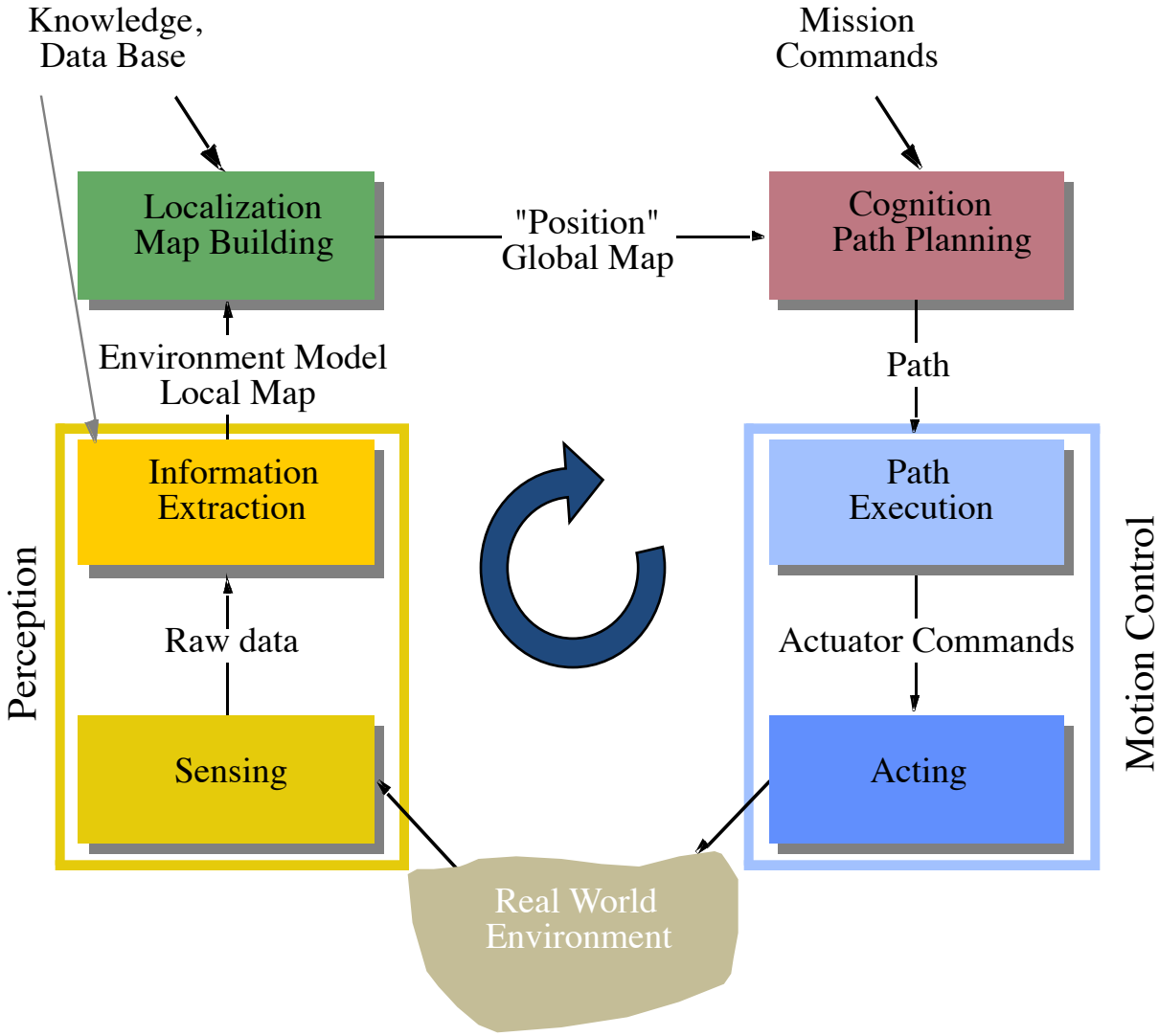

cliff_sensor_left_link
gyro_link
cliff_sensor_front_link
caster_front_link
caster_back_link
base_link
base_footprint
cliff_sensor_right_link

Transforms in ROS

- Imagine: Object recognition too 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

HIGH-LEVEL CONTROL SCHEMES

General Control Scheme for Mobile Robot Systems



ADMIN

Admin

- HW1 Questions?
- HW2 will be published today
- Next lecture (Wed Sep 28) in the Robotics Lab!?
  - Live demos and hands-on ROS
  - Bring your Laptops

SENSORS

Introduction to Autonomous Mobile Robots page 102 ff

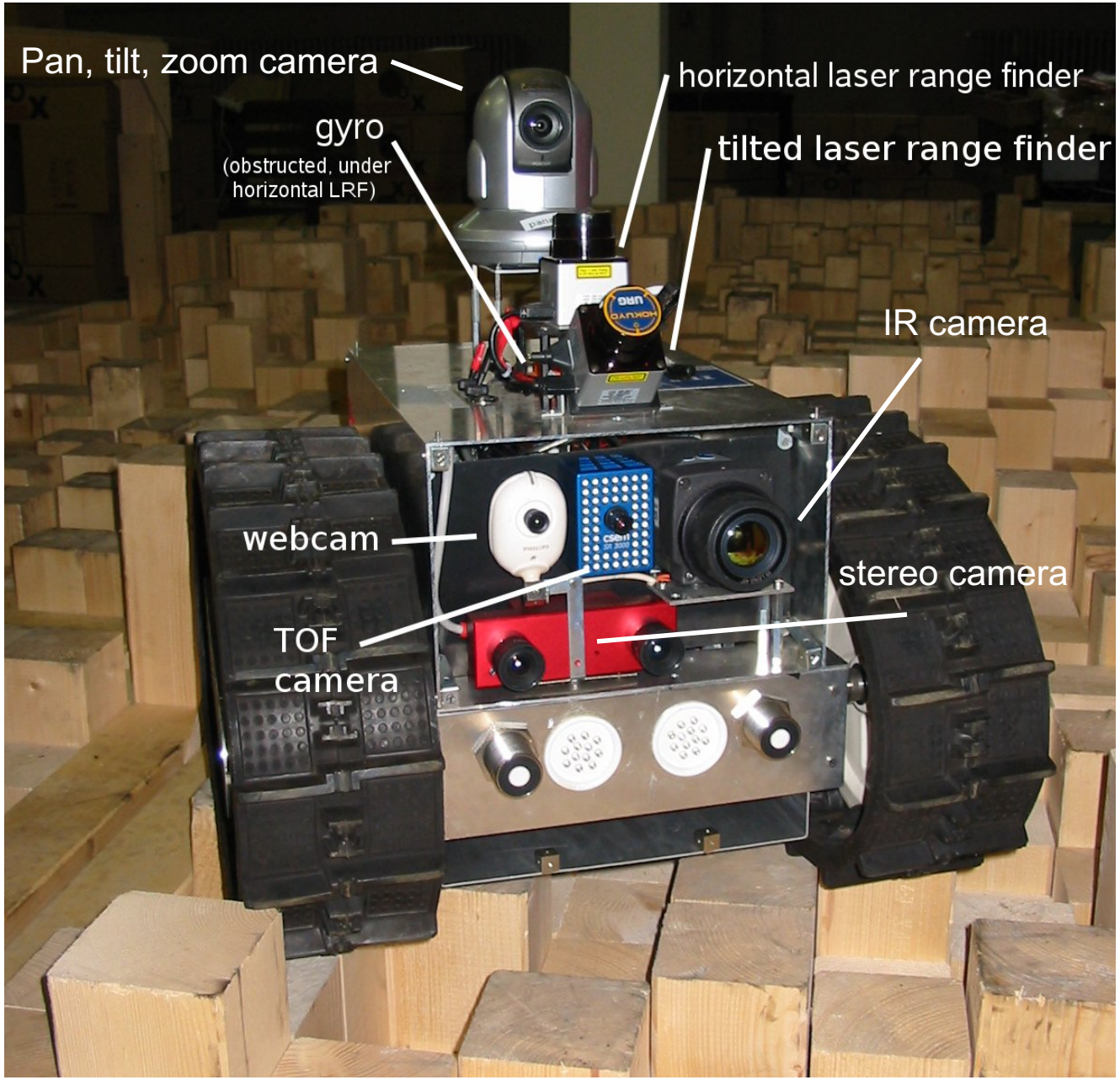
Sensors for Mobile Robots

- Why should a robotics engineer know about sensors?
  - Is the **key technology** for perceiving the environment
  - **Understanding the physical principle** enables appropriate use

- Understanding the physical principle behind sensors enables us:
  - To **properly select** the sensors for a given application
  - To **properly model** the sensor system, e.g. resolution, bandwidth, **uncertainties**

Example

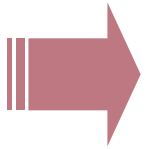
- Jacobs Robotics Rugbot
- Search and Rescue Robot
- RoboCup Rescue



Dealing with Real World Situations

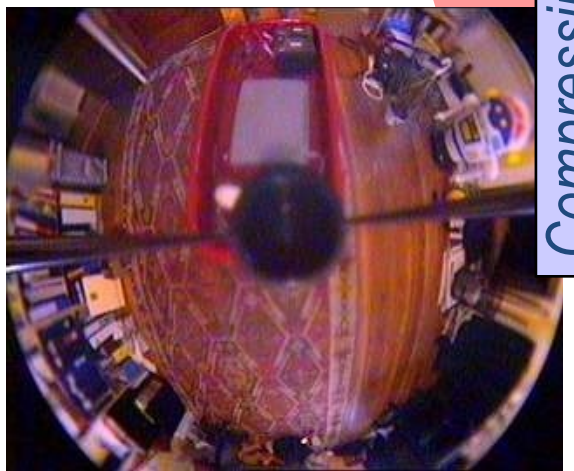
- Reasoning about a situation

- Cognitive systems have to interpret situations based on uncertain and only partially available information
- The need ways to learn functional and contextual information (semantics / understanding)



Probabilistic Reasoning

Perception for Mobile Robots



Places / Situations
A specific room, a meeting situation, …

Servicing / Reasoning

- **Functional / Contextual Relationships of Objects**
  - *imposed*
  - *learned*
  - *spatial / temporal/semantic*

Objects
Doors, Humans, Coke bottle, car , …

Interaction

- **Models / Semantics**
  - *imposed*
  - *learned*

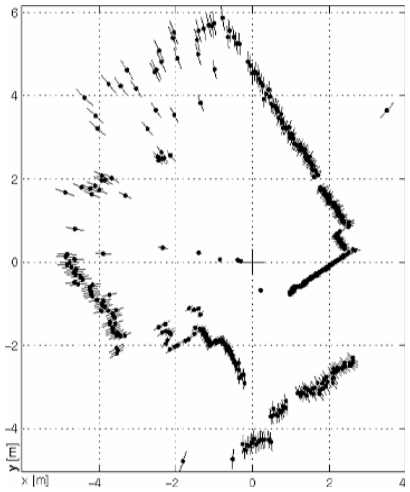
Features
Lines, Contours, Colors, Phonemes, …

Navigation

- **Models**
  - *imposed*
  - *learned*

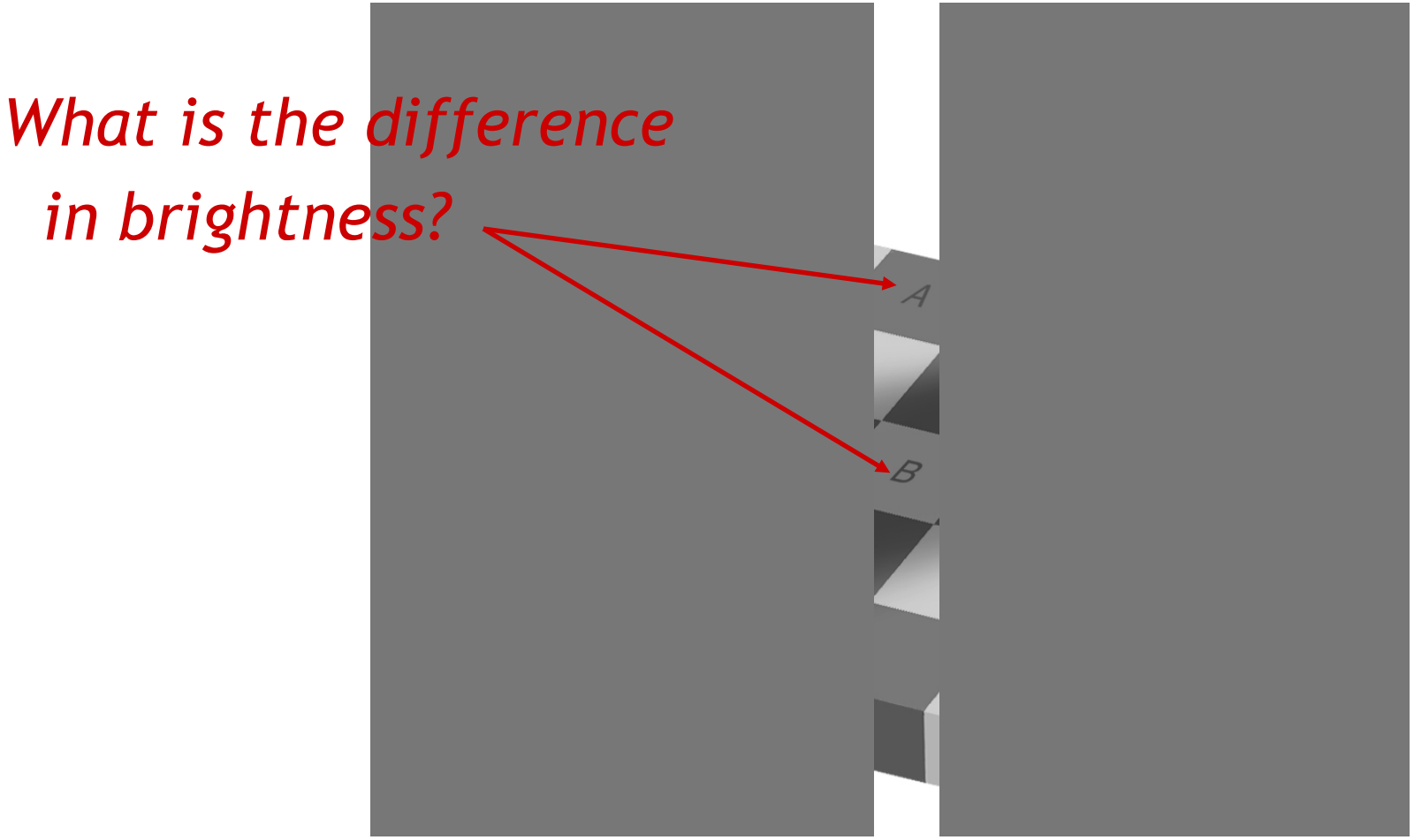
Raw Data
Vision, Laser, Sound, Smell, …

Compressing Information



The Challenge

- Perception and models are strongly linked



- http://web.mit.edu/persci/people/adelson/checkershadow_downloads.html

Classification of Sensors

- What:
  - Proprioceptive sensors
    - measure values internally to the system (robot),
    - e.g. motor speed, wheel load, heading of the robot, battery status
  - Exteroceptive sensors
    - information from the robots environment
    - distances to objects, intensity of the ambient light, unique features.
- How:
  - Passive sensors
    - Measure energy coming from the environment
  - Active sensors
    - emit their proper energy and measure the reaction
    - better performance, but some influence on environment

General Classification (1)

General classification (typical use)	Sensor Sensor System	PC or EC	A or P
Tactile sensors (detection of physical contact or closeness; security switches)	Contact switches, bumpers	EC	P
	Optical barriers	EC	A
	Noncontact proximity sensors	EC	A
Wheel/motor sensors (wheel/motor speed and position)	Brush encoders	PC	P
	Potentiometers	PC	P
	Synchros, resolvers	PC	A
	Optical encoders	PC	A
	Magnetic encoders	PC	A
	Inductive encoders	PC	A
	Capacitive encoders	PC	A
Heading sensors (orientation of the robot in relation to a fixed reference frame)	Compass	EC	P
	Gyroscopes	PC	P
	Inclinometers	EC	A/P

A, active; P, passive; P/A, passive/active; PC, proprioceptive; EC, exteroceptive.

General Classification (2)

General classification (typical use)	Sensor Sensor System	PC or EC	A or P
Ground-based beacons (localization in a fixed reference frame)	GPS Active optical or RF beacons Active ultrasonic beacons Reflective beacons	EC EC EC EC	A A A A
Active ranging (reflectivity, time-of-flight, and geo-metric triangulation)	Reflectivity sensors Ultrasonic sensor Laser rangefinder Optical triangulation (1D) Structured light (2D)	EC EC EC EC EC	A A A A A
Motion/speed sensors (speed relative to fixed or moving objects)	Doppler radar Doppler sound	EC EC	A A
Vision-based sensors (visual ranging, whole-image analy-sis, segmentation, object recognition)	CCD/CMOS camera(s) Visual ranging packages Object tracking packages	EC	P

Characterizing Sensor Performance

- Basic sensor response ratings
  - Range
    - upper limit; lower limit
  - Resolution
    - minimum difference between two values
    - usually: lower limit of dynamic range = resolution
    - for digital sensors it is usually the A/D resolution.
      - e.g. 5V / 255 (8 bit)
  - Linearity
    - variation of output signal as function of the input signal
    - linearity is less important when signal is treated with a computer

$$x \rightarrow f(x)$$

$$y \rightarrow f(y)$$

$$\alpha \cdot x + \beta \cdot y \rightarrow f(\alpha \cdot x + \beta \cdot y) = \alpha \cdot f(x) + \beta \cdot f(y)$$

Characterizing Sensor Performance

- Bandwidth or Frequency
  - the speed with which a sensor can provide a stream of readings
  - usually there is an upper limit depending on the sensor and the sampling rate
  - lower limit is also possible, e.g. acceleration sensor
  - one has also to consider phase (delay) of the signal

- Sensitivity
  - ratio of output change to input change $\frac{dy}{dx}$

- Cross-sensitivity (and cross-talk)
  - sensitivity to other environmental parameters (e.g. temperature, magnetic field)
  - influence of other active sensors

In Situ Sensor Performance

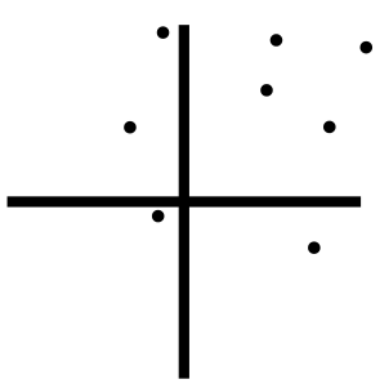
- In Situ: Latin for “in place”

- Error / Accuracy
  - How close to true value
- Precision
  - Reproducibility

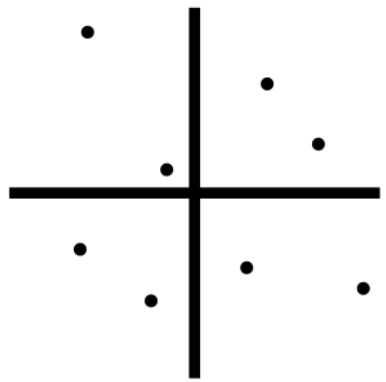
$$\left(accuracy = 1 - \frac{|m - v|}{v}\right)$$

error

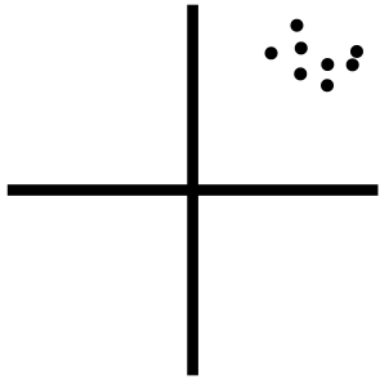
m = measured value
v = true value



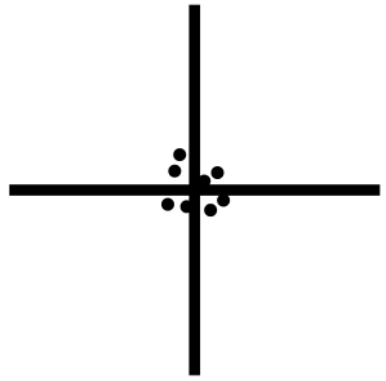
(a) Low precision and low accuracy



(b) Low precision and high accuracy



(c) High precision and low accuracy



(d) High precision and high accuracy

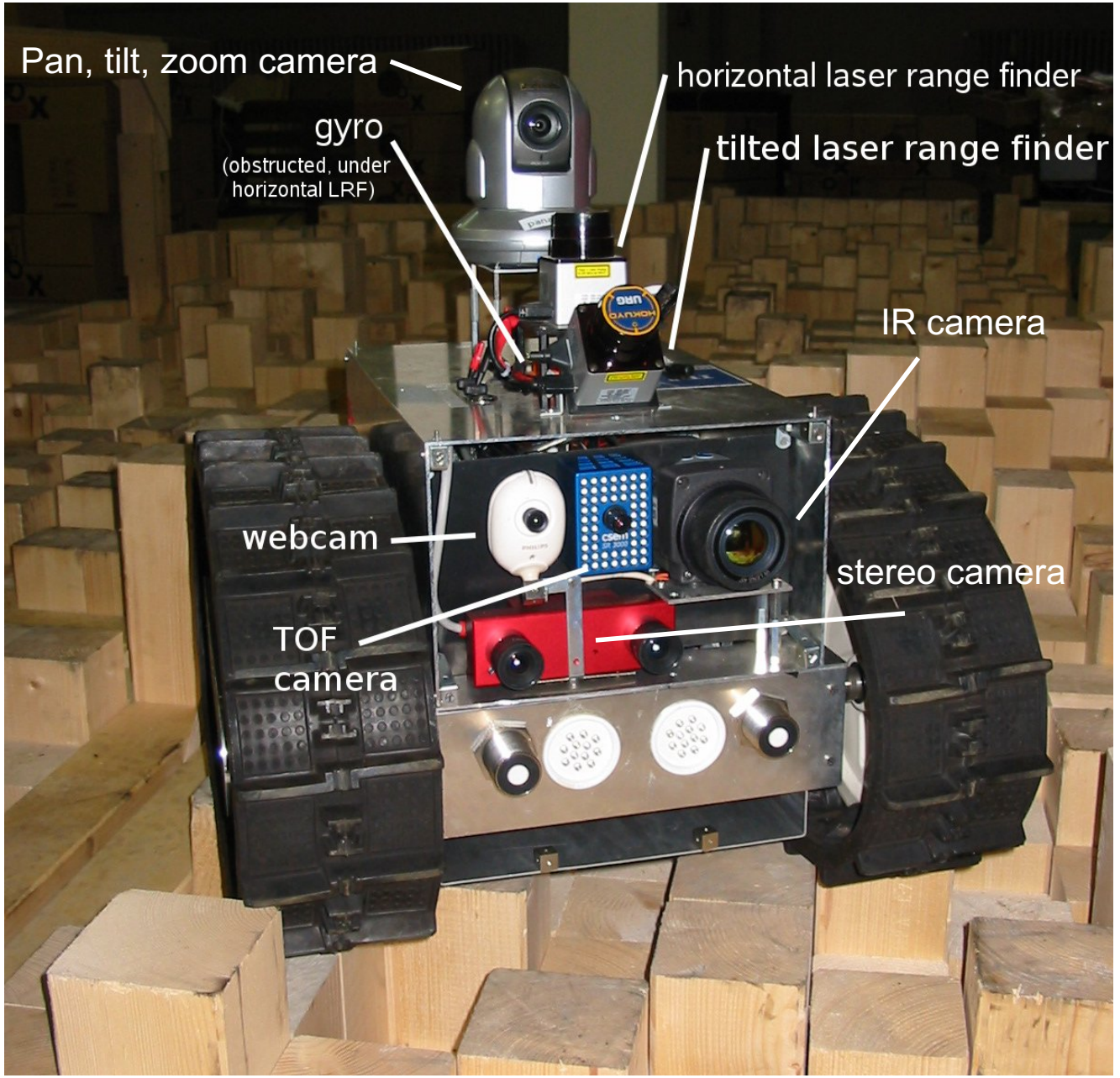
Types of error

- Systematic error -> deterministic errors
  - caused by factors that can (in theory) be modeled -> prediction
  - e.g. calibration of a laser sensor or of the distortion caused by the optic of a camera

- Random error -> non-deterministic
  - no prediction possible
  - however, they can be described probabilistically
  - e.g. Hue instability of camera, black level noise of camera ..

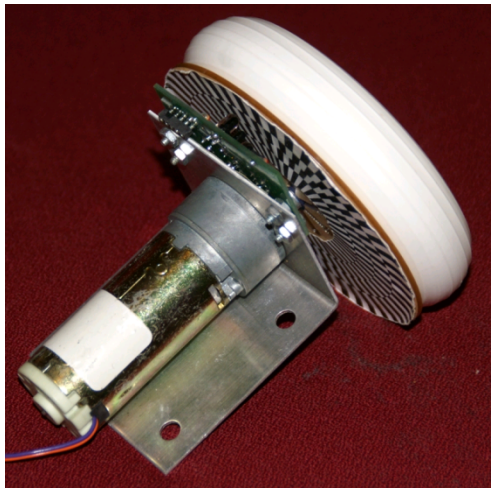
Sensors: outline

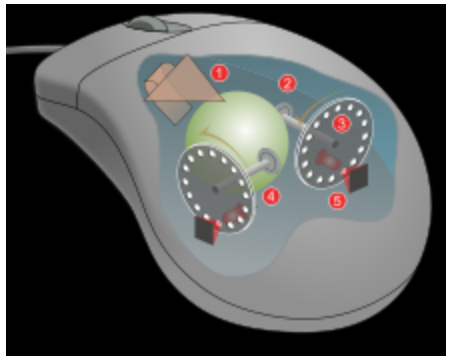
- Optical encoders
- Heading sensors
  - Compass
  - Gyroscopes
- Accelerometer
- IMU
- GPS
- Range sensors
  - Sonar
  - Laser
  - Structured light
- Vision



Encoders

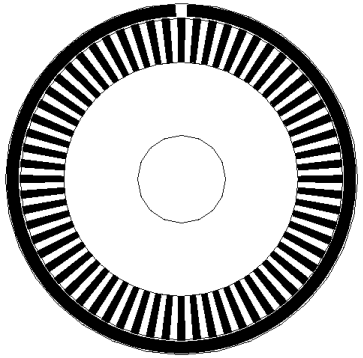
An encoder is an electro-mechanical device that converts the angular position of a shaft to an analog or digital signal, making it an angle transducer

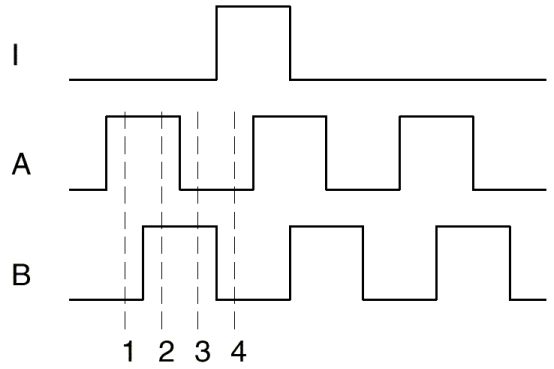




Wheel / Motor Encoders

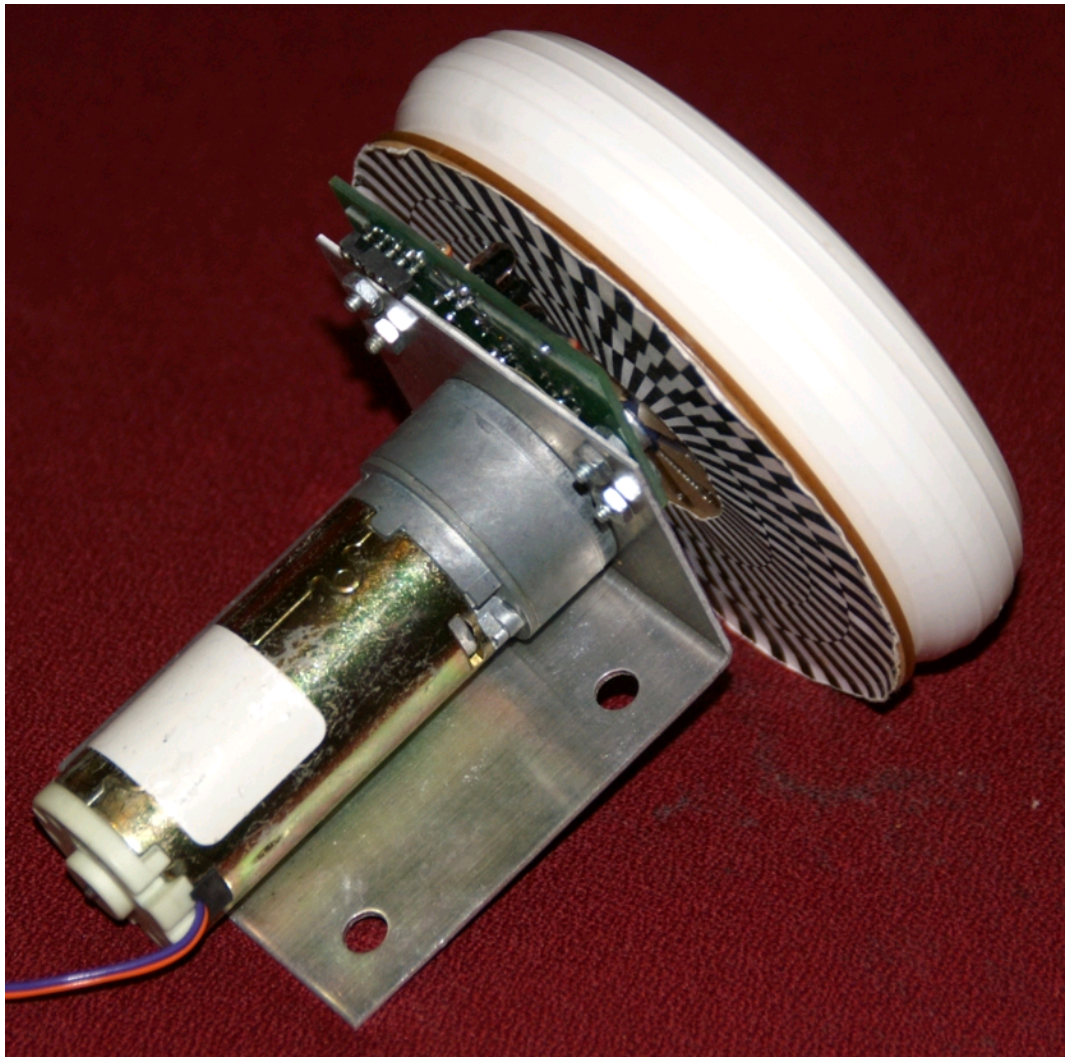
- measure position or speed of the wheels or steering
- integrate wheel movements to get an estimate of the position -> odometry
- optical encoders are proprioceptive sensors
- typical resolutions: 64 - 2048 increments per revolution.
  - for high resolution: interpolation
- optical encoders
  - regular: counts the number of transitions but cannot tell the direction of motion
  - quadrature: uses two sensors in quadrature-phase shift. The ordering of which wave produces a rising edge first tells the direction of motion. Additionally, resolution is 4 times bigger
  - a single slot in the outer track generates a reference pulse per revolution





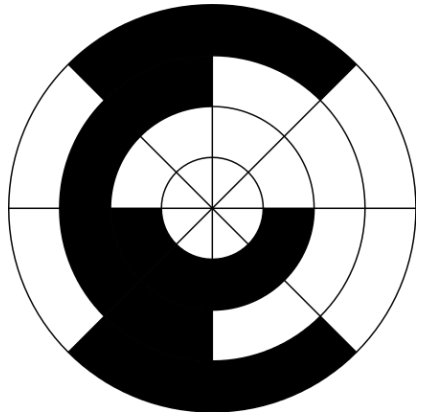
State	Ch A	Ch B
S_1	High	Low
S_2	High	High
S_3	Low	High
S_4	Low	Low

A custom made optical encoder



Gray Encoder

http://en.wikipedia.org/wiki/Gray_code



- Aka: reflected binary code, Gray Code
  - Binary numeral system where two successive values differ in only one bit
  - Also used for error correction in digital communications

- Absolute position encoder
  - Normal binary => change from 011 to 100
  - 2 bits change – NEVER simultaneously =>
  - 011 -> 111 -> 101 -> 100 or
  - 011 -> 010 -> 110 -> 100 ….
  - => wrong encoder positions might be read
  - Gray encoding: only one bit change!

Dec	Gray	Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111