64-424
Intelligent Robotics

Niklas Fiedler

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

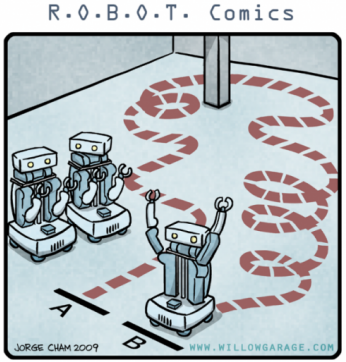
Winterterm 2023/2024

Outline

Outline

Robotics is fun!

- Autonomous robotics is an interdisciplinary field
  - Mechatronics
  - Architecture- and system design
  - Control theory
  - Data science / Machine Learning
  - Reasoning Methods
  - Psychology
  - Cognitive Science
  - ...
- Lots of problems wait for your solution!



"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

Heider and Simmel

Video Heider and Simmel `https://www.youtube.com/watch?v=VTNmLt7QX8E`

- Animation from 1944

What is “Intelligent”?

1.http://usportt.com/wp-content/uploads/2017/09/Chess.jpg **2**.https://c2.staticflickr.com/8/7290/12314247113_0b71480233_b.jpg

3.https://i.pinimg.com/736x/31/3d/b4/313db492ab9da909013eb99e3f8f2232--candy-colors-dog-activities.jpg

4.https://www.bbc.com/news/av/science-environment-37364938/clever-crow-naturally-uses-tools

What is "Intelligent Behavior"?

Blake Lemoine

Jun 11 · 20 min read · Listen

Is LaMDA Sentient? — an Interview

What follows is the "interview" I and a collaborator at Google conducted with LaMDA. Due to technical limitations the interview was conducted over several distinct chat sessions. We edited those sections together into a single whole and where edits were necessary for readability we edited our prompts but never LaMDA's responses. Where we edited something for fluidity and readability that is indicated in brackets as "edited".

https://cajundiscordian.medium.com/is-lamda-sentient-an-interview-ea64d916d917

What is an "Intelligent Robot"?

- There is no general definition
- Intelligence is often associated with
  - Interpretation of perceptions
  - Mental world models
  - Goal-directed actions
  - The Total Turing test
- These are hard to define or model when programming robots
- We *can* talk about
  - Scene / Object recognition
  - Tracking
  - Pick&Place
  - Path planning
  - Localization
  - ...

The old problem with AI

"Robots can do it [already], so it's not intelligence"

Definitions of Intelligence

"*Judgment, otherwise called good sense, practical sense, initiative, the faculty of adapting one's self to circumstances ... auto-critique.*" Alfred Binet

"*Intelligence is the computational part of the ability to achieve goals in the world*" John McCarthy

Definitions of Intelligence

"*A very general mental capability that, among other things, involves the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly and learn from experience. It is not merely book learning, a narrow academic skill, or test-taking smarts. Rather, it reflects a broader and deeper capability for comprehending our surroundings—"catching on", "making sense" of things, or "figuring out" what to do.*" Mainstream Science on Intelligence

What is an Intelligent Robot made of?

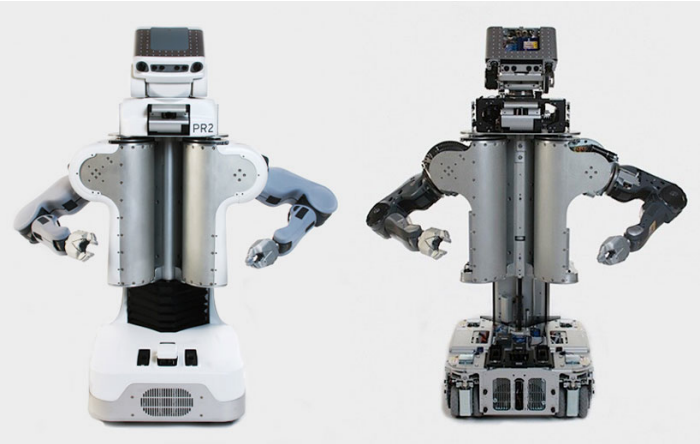
What is an Intelligent Robot made of?

- Sensors
- Processing Unit
- Actuators

What is an Intelligent Robot made of?

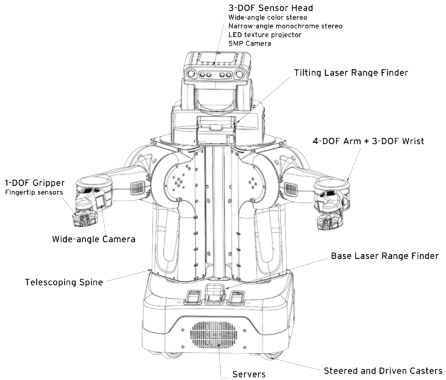
- Sensors
  - Camera
  - Microphone
  - Laser Scanner
  - Button/Bumper
  - Encoder
  - ...
- Processing Unit
  - Conventional Computer
  - Microcontroller
  - FPGA
  - ...
- Actuators
  - Wheels
  - Arms
  - Gripper
  - ...

Example: Personal Robot 2 (PR2)



Personal Robot 2 (PR2): Platform

- Mobile base
  - 4-wheel omnidirectional drive
  - 1kHz motor control loop
  - Fixed laser range finder
- Two compliant arms
  - Passive counterbalance
  - 4 degree of freedom arms
  - 3 degree of freedom wrists
  - Payload: $\approx$ 1.8kg



Personal Robot 2 (PR2): Sensor head

- 2 degrees of freedom (pan/tilt)
- Kinect RGB-D camera (not shown)
- Inertial measurement unit
- RGB LED eyes (not shown)
- Smartphone (not shown)
- Tilting laser range finder
- LED texture projector
- Environment (wide) stereo cameras
- Manipulation (narrow) stereo cameras
- 5MP RGB camera

Personal Robot 2 (PR2): Sensor head

- 2 degrees of freedom (pan/tilt)
- Kinect RGB-D camera (not shown)
- Inertial measurement unit
- RGB LED eyes (not shown)
- Smartphone (not shown)
- Tilting laser range finder
- LED texture projector
- Environment (wide) stereo cameras
- Manipulation (narrow) stereo cameras
- 5MP RGB camera

Personal Robot 2 (PR2): Grippers

- 1 degree of freedom
- 90mm range of motion
- 3-axis accelerometers
- Fingertip pressure sensor arrays
- Grip force: 80N ($\approx$ 8kg)
- Gripper cameras in forearm

Personal Robot 2 (PR2): Processing and control

- Dual Quad-Core CPUs
- 24 GB main memory
- 1.5 TB storage
- Multi-gigabit interconnections
- Synchronized network cameras
- EtherCAT communication bus

Many Opportunities for You to Contribute!

- The field and community of robotics is growing fast
- Lots of innovative research work
- Many amazing interdisciplinary applications
- Lots of grant money and high demand for specialists
- Many problems are yet to be solved
- Existing literature provides a good foundation to build upon

The Lecture in an Application

- You want to build a robot
- What does the robot need to do?
- What hardware do we need?
- How do we make “intelligent” decisions?
- What are the limitations for this robot?

Videos

Seminar Organization

- Presentation dates
- Seminar topics

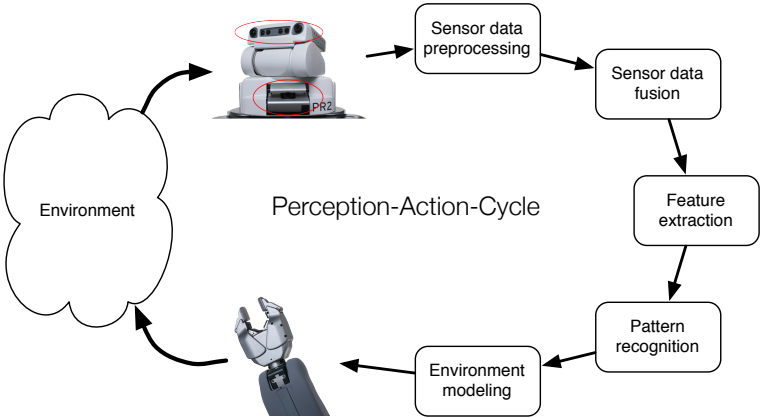
Afterwards:
Let's visit the labs and have a look at some robots.

Outline

Sensors in Robotics: Perception

- Sensors are crucial to the development of intelligent robotic systems
- Sensor data provides an abstract perception of the environment
- The Perception-Action-Cycle represents the control loop
  1. Sensing of the environment
  2. "Intelligent" processing of obtained data
  3. Execution of an action
- The cycle is crucial to the implementation of interactive, adaptive and situation-based behavior

Perception-Action-Cycle: Overview



Perception-Action-Cycle

1. **Data acquisition:** Sampling of analog/digital signals output from sensor devices
2. **Data (pre-)processing:** Filtering, normalization and/or scaling, etc., of acquired data
3. **Data fusion:** Combination/fusion of multi-modal and redundant sensor data leading to robust measurements, reduced uncertainty and an increase in information
4. **Feature extraction:** Extraction of features representing a mathematical model of the sensed environment in order to approximate the natural human perception

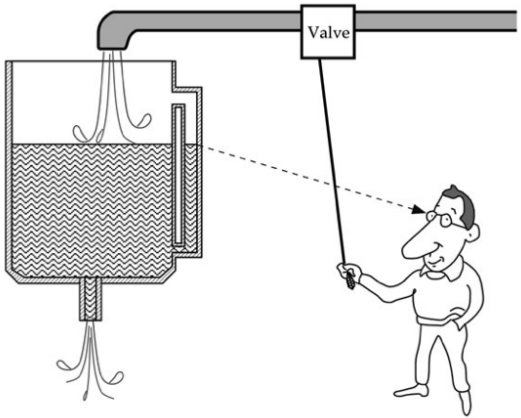
Perception-Action-Cycle (cont.)

5. **Pattern recognition:** Extracted features are searched for patterns in order to classify the data
6. **Environment modeling:** Successfully classified patterns are used to model the environment of the robotic system

... ...

n. **Action:** Based on the model of the environment sets of goal-oriented actions are executed manipulating the environment (using robotic arms, grippers, wheels, etc.)

A Sensor - A Simple Example

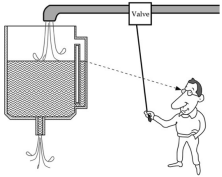


What is a Sensor?

The sensor in the example consists of two parts:

- The water level indicator
- The human eye

⇒ Perception of the level indicator results in a signal to the brain

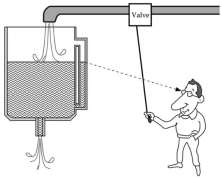


What is a Sensor?

The sensor in the example consists of two parts:

- The water level indicator
- The human eye

⇒ Perception of the level indicator results in a signal to the brain



Definition

A sensor is a unit, which

- receives a signal or stimulus
- and reacts to it

Natural and Physical Sensors

Natural sensors:

- A reaction is an electrochemical signal on neural pathways
- Examples: Auditory sense, visual sense, tactile sense, ...

Natural and Physical Sensors

Natural sensors:

- A reaction is an electrochemical signal on neural pathways
- Examples: Auditory sense, visual sense, tactile sense, ...

Physical sensors:

Definition

A physical sensor is a unit, which

- receives a signal or stimulus
- and reacts to it with an electrical signal

Input Signal

- A physical sensor converts a (generally) non-electrical signal into an electrical one
- This signal is referred to as the stimulus

Definition

A stimulus is a

- quantity,
- characteristic or
- state,

which is perceived and converted into an electrical signal

Output Signal

- The output signal can be
  - a voltage,
  - a current or
  - a charge
- Furthermore, the signal can be distinguished by
  - amplitude,
  - frequency or
  - phase
- Some integrated sensors provide a digital data output

Taxonomy

- Intrinsic sensors :
  Provide data about the *internal system state*
- Extrinsic sensors:
  Provide data about the *environment*

- Active sensors:
  Actively produce a stimulus and measure how it is affected
- Passive sensors:
  Change their electrical characteristics in response to the change of the stimulus (conversion of the stimulus)

Further Classification

Physical sensors can also be classified by:

- Type of stimulus
- Characteristics, specification and parameters
- Type of stimulus detection
- Conversion of stimulus to output signal
- Sensor material
- Field of application
- ...

Sensor Examples

- **Intrinsic sensors:**
- **Extrinsic sensors (force/pressure):**
- **Extrinsic sensors (distance):**
- **Visual sensors:**

Sensor Examples

- **Intrinsic sensors:**
  Encoder (incremental/absolute), accelerometer, gyroscope, ...
- **Extrinsic sensors (force/pressure):**
- **Extrinsic sensors (distance):**
- **Visual sensors:**

Sensor Examples

- **Intrinsic sensors:**
  Encoder (incremental/absolute), accelerometer, gyroscope, ...
- **Extrinsic sensors (force/pressure):**
  Strain gauge, force-torque sensor, piezoelectric sensor, ...
- **Extrinsic sensors (distance):**

- **Visual sensors:**

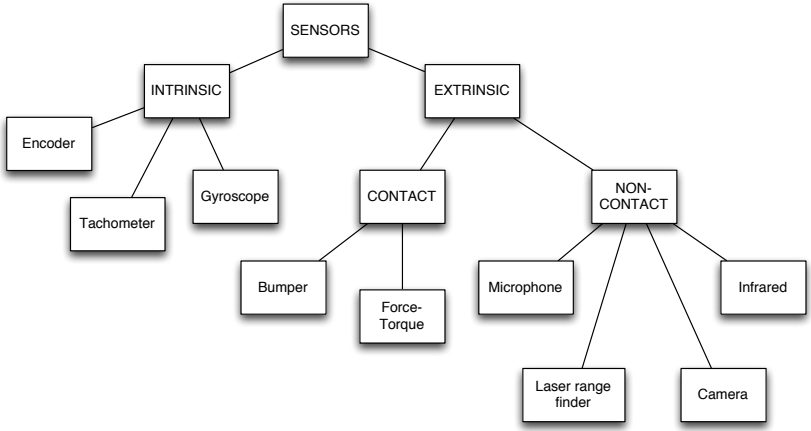
Sensor Examples

- **Intrinsic sensors:**
  Encoder (incremental/absolute), accelerometer, gyroscope, ...
- **Extrinsic sensors (force/pressure):**
  Strain gauge, force-torque sensor, piezoelectric sensor, ...
- **Extrinsic sensors (distance):**
  Sonar sensor, infrared sensor, laser range finder, ...
- **Visual sensors:**

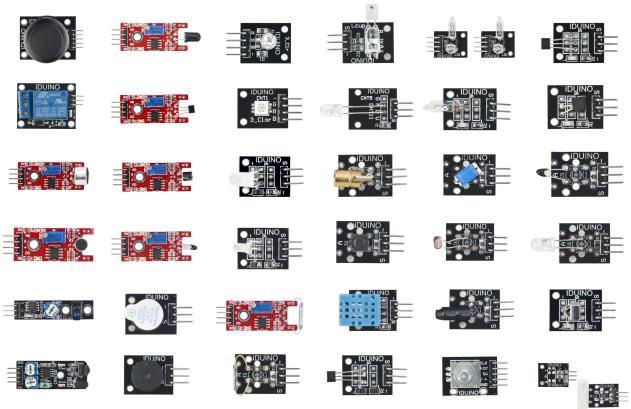
Sensor Examples

- **Intrinsic sensors:**
  Encoder (incremental/absolute), accelerometer, gyroscope, ...
- **Extrinsic sensors (force/pressure):**
  Strain gauge, force-torque sensor, piezoelectric sensor, ...
- **Extrinsic sensors (distance):**
  Sonar sensor, infrared sensor, laser range finder, ...
- **Visual sensors:**
  Linear camera, CCD-/CMOS-camera, stereo vision cameras, omnidirectional vision camera, ...

Classification Example



Sensor Examples



https://purelogictrading.onlineweb.shop/37_in_1_Sensor_Kit_For_Arduino/p6644729_20041924.aspx

Sensor Examples

https://www.techspurts.com/a-drone-will-be-in-charge-of-driving-your-autonomous-car-if-its-sensors-break-down/

Measurements with Sensors

- Measurement results have to be *reliable* (within specification)
- Important scientific criterion: *Reproducibility* of measurements
- Scientific statements have to be comparable
- Statements must be *quantitative* and based on measurements
- A measurement result consists of:
  - Numerical value
  - Measuring unit
- Additionally: Declaration of measurement accuracy

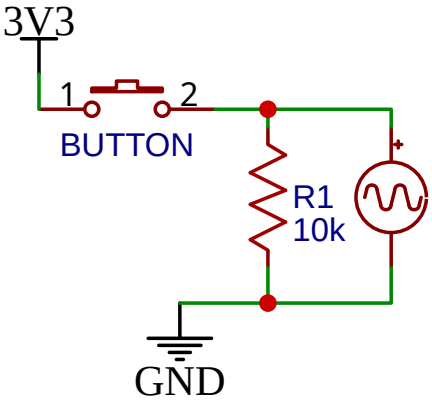
Measurement Errors

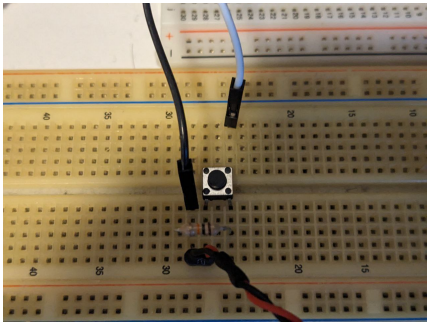
No measurement process yields an entirely accurate result!

Every Sensor Produces Errors!

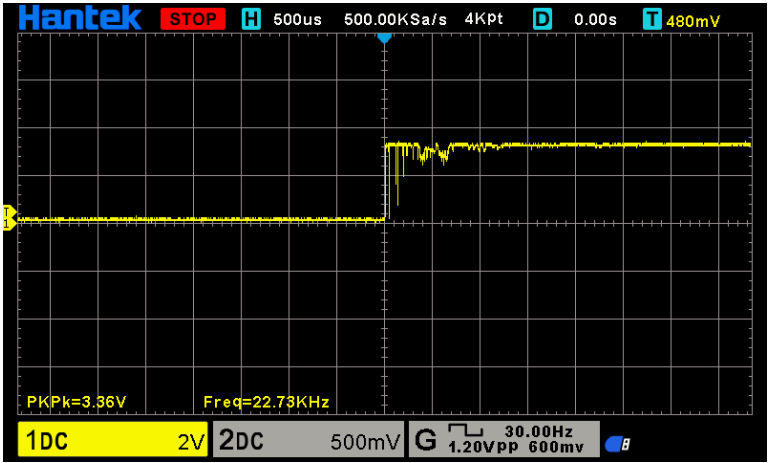
- There is no type of sensor which always produces a perfect result in a real world application
- Acceptable inaccuracies have to be determined for the application
- There are various approaches to increase sensor accuracy:
  - calibration
  - error compensation
  - data filtering
  - multi sensor fusion

Button Bouncing Example

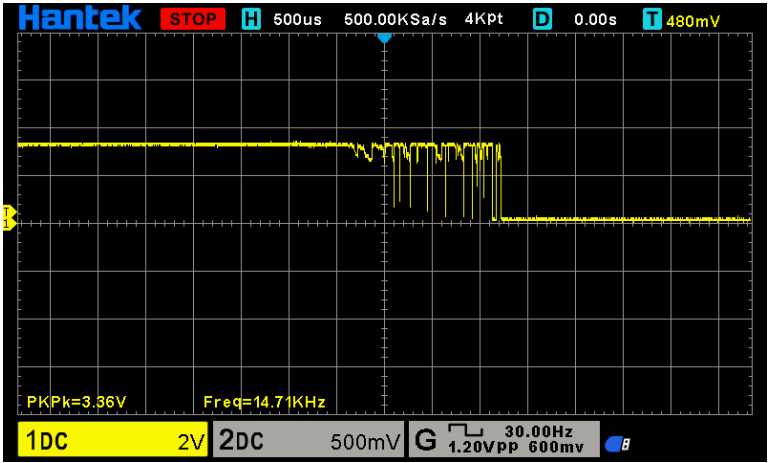




Button Bouncing Example



Button Bouncing Example



Measurement Deviation (Measurement Error)

Systematic deviation ("systematic error"):

- Deviation is caused by the sensor itself
- For example: wrong calibration, persistent sources of interference like friction, etc.
- Elimination is possible, but requires elaborate examination of the error source

Random deviation ("random or stochastic error"):

- Deviation is caused by external interference
- Repeated measurements yield different results
- Individual results fluctuate around a mean value
- For example: electronic noise in the circuit, environmental factors

Measurement Deviation (Measurement Error)

Systematic deviation ("systematic error"):

- Deviation is caused by the sensor itself
- For example: wrong calibration, persistent sources of interference like friction, etc.
- Elimination is possible, but requires elaborate examination of the error source

Random deviation ("random or stochastic error"):

- Deviation is caused by external interference
- Repeated measurements yield different results
- Individual results fluctuate around a mean value
- For example: electronic noise in the circuit, environmental factors

Error Declaration

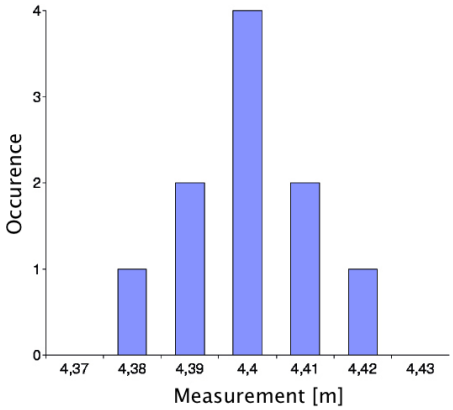
- Measurements are always afflicted with uncertainty
- **Example:** Distance measurement
  - Distance to an object is measured 10 times $(x_1, ..., x_{10})$

Individual measurement results:				
$4,40\ m$	$4,40\ m$	$4,38\ m$	$4,41\ m$	$4,42\ m$
$4,39\ m$	$4,40\ m$	$4,39\ m$	$4,40\ m$	$4,41\ m$

- Due to random deviation individual measurement results x_i vary

Error Declaration (cont.)

Measurements can be illustrated in a histogram:



Mean Value

The mean value $\bar{x}$ of the individual measurements x_i is determined as follows:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- The mean value is also called arithmetic average or best estimate for the true value μ
- μ is the *mean* or *expected value* of the set of all possible measurement values (population)

Absolute and Relative Error

Measurement deviation can be specified in two different ways

- Absolute measurement deviation ("Absolute error"):
  The absolute error Δx_i of a **single** measurement x_i equals the deviation from the mean value $\bar{x}$ of all N measurements $\{x_n | n \in \{1 \dots N\}\}$ of a measurement series
  - The unit is equal to that of the measured value
  - $\Delta x_i = |x_i - \bar{x}|$
- Relative measurement deviation ("Relative error"):
  The relative error $\Delta x_{i\,\mathrm{rel}}$ is the relation between absolute error Δx_i and the mean value $\bar{x}$
  - Has no unit, often specified as a percentage (%)
  - $\Delta x_{i\,\mathrm{rel}} = \frac{\Delta x_i}{\bar{x}_i}$

Variance of a Measurement Series

- How far are the measurement samples spread out?

The distribution of single measurement values x_i around the arithmetic mean $\bar{x}$ is represented by the variance of a measurement series.
The factor $1/(N-1)$ includes Bessel's correction of the bias

$$s^2 = (\Delta x)^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\Delta x_i)^2$$
$$= \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Standard Deviation of a Measurement Series

Similar to the variance, the positive square root of the variance - called the standard deviation - is another representation of the dispersion of measurement values x_i around the mean value $\bar{x}$

$$s = \Delta x = \sqrt{s^2} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

- The standard deviation is also known as the mean error of a single measurement
- In contrast to the variance the standard deviation carries the same unit as the measurement samples

Standard Deviation of the Mean

- The true mean value (μ) of the population is unknown

The standard deviation of the mean value, also error of the mean value, is determined as follows

$$\begin{aligned} s_{\bar{x}} &= \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^{N} (x_i - \bar{x})^2} \\ &= \frac{\Delta x}{\sqrt{N}} = \frac{s}{\sqrt{N}} \end{aligned}$$

$s_{\bar{x}}$ is the deviation of the mean values of individual measurement series ($\bar{x}$) from the true mean value μ

Proof:

https://en.wikipedia.org/wiki/Standard_deviation#Standard_deviation_of_the_mean

Measurement Result

- The variance and standard deviation of a measurement series show us the spread from the mean of the series
- The standard deviation of the mean gives us the spread from the true mean μ

With the above in mind we can expect a measurement sample to be given by

$$x = \bar{x} \pm s \pm s_{\bar{x}} \, [Unit]$$

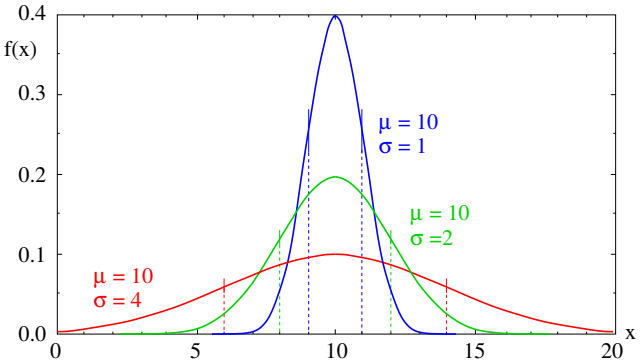
Normal Distribution

- For $N \to \infty$ a discrete distribution of a measurement series turns into a continuous distribution
- With $N \to \infty$ we can assume $\bar{x} \to \mu$ and $s \to \sigma$, resulting in the density function of a normal distribution (*Gaussian distribution*)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

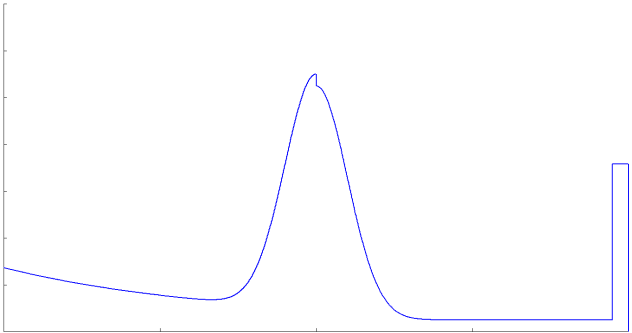
- The measurements of a physical/technical quantity X are *usually* assumed to be normally distributed

Normal Distribution (cont.)



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Many measurements are not normally distributed



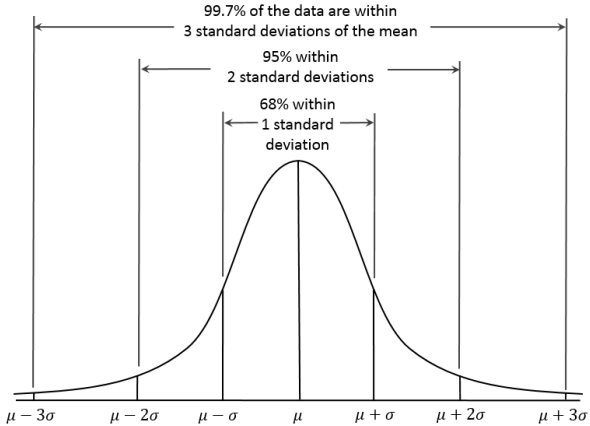
A model for measurements of distances might account for

- stochastic noise
- disturbances before target
- noise floor
- max-range artifacts

Confidence Interval

- Interval around a determined mean value of a measurement series that is said to contain other samples of the series with a given probability (confidence)
- A confidence interval of σ ($s_{\bar{x}}$) is said to contain 68.27 % of the population samples
- Extended to 2σ ($2s_{\bar{x}}$) the interval covers 95.45 % of the population
- 3σ ($3s_{\bar{x}}$) is said to contain 99.73 % of the population

Confidence Interval (cont.)



Full Scale Input/Output

- The dynamic range of measurable stimulus levels is defined as the full scale input (span) of the sensor
  - An input signal (stimulus) outside of the specified range may result in a strong falsification of the output signal ...
  - ... or damage the sensor (e.g. thermistor, load cell)
- Similarly to the range of the stimulus the full scale output defines the range of output electrical signal

Accuracy

- Manufacturers usually provide a specification of accuracy for the given range of the output signal
- With physical sensors accuracy really means inaccuracy
- Often the inaccuracy is given in the form of a relative error
- Sometimes the manufacturer provides data about *systematic* errors (determined through calibration)
- The specification of inaccuracy consolidates the effects of various sources of error

Resolution

- The resolution is the smallest possible change of the stimulus that is detected by the sensor

- **Examples:** Tactile sensor (pressure), laser range finder (distance), ...
- The resolution may vary over the entire range of the input signal

- The resolution of digital output is defined by the number of bits
- A sensor is said to have a *continuous* or *infinitesimal* resolution if it does not have distinct resolution steps in the output signal
- Resolution is bound by the noise floor

Decision Task: Purchase a Scale

- Option A: 0-120±1 kg, displays 0.1 kg
- Option B: 0-150±0.1 kg, displays 1 kg
- Option C: 0-100±0.1 kg, displays 0.01 kg

- Range
- Accuracy
- Resolution

Sensor Characteristics

- A sensor may feed the stimulus through several conversion stages until it emits an electrical output signal
- **Example:** Pressure on a fibre-optic sensor
  1. Fiber strain $\rightarrow$ change of refractive index
  2. Change of optical transmission properties
  3. Photon flux detection
  4. Conversion into electrical output signal
- We consider the sensor a "*black box*" and focus on the relation between the stimulus and the output signal

Transfer Function

- The transfer function of a sensor represents the relation between stimulus and output signal
- Each sensor has an ideal/theoretical relation between the stimulus and output signal

Definition

The ideal relation between stimulus and output signal of a sensor is characterized by the transfer function

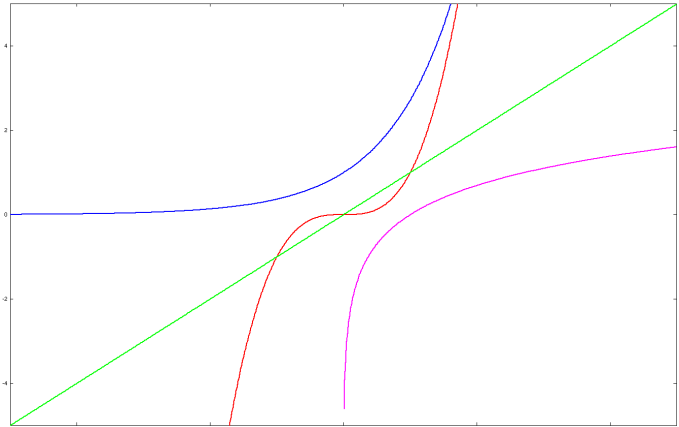
$$S = f(s)$$

- S represents the true value of the stimulus s

Transfer Function (cont.)

Exemplary transfer functions are

- Linear —
  $S = a + k \cdot s$
- Logarithmic —
  $S = a + k \cdot \ln s$
- Exponential —
  $S = a \cdot e^{ks}$
- Polynomial —
  $S = a_0 + a_1 \cdot s^k$

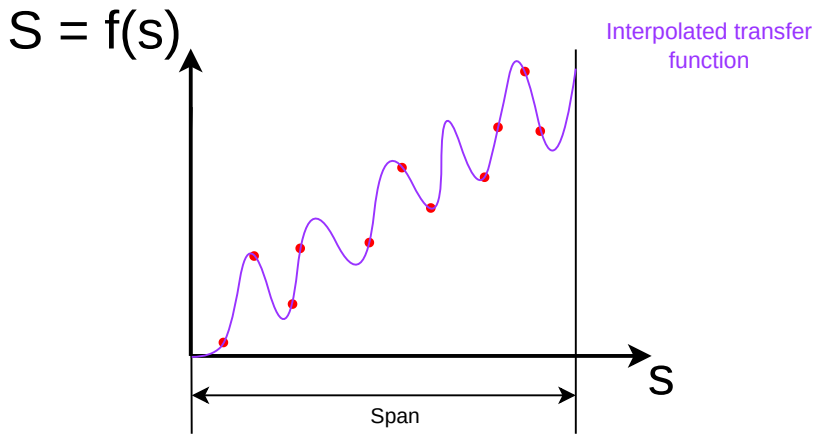


Interlude - Approximation vs. Interpolation

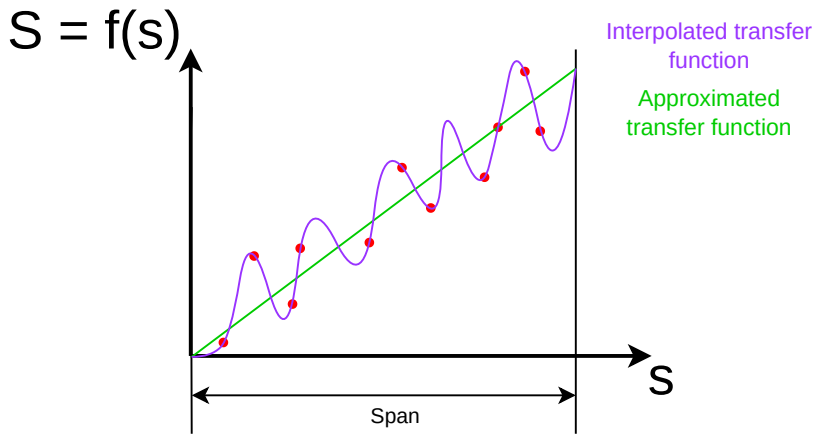
A measurement series should be approximated using the simplest possible function $f(x)$

- Approximation:
  The function $f(x)$ shows a very good representation of the value pairs (x_i, y_i) (e.g. least-squares fit)
  - $f(x_i) = y_i$ **does not** need to be valid
- Interpolation:
  The function $f(x)$ shows an exact representation of the value pairs
  - $f(x_i) = y_i; \quad i = 1, 2, \ldots, n$ **must** be valid

Interlude - Approximation vs. Interpolation



Interlude - Approximation vs. Interpolation

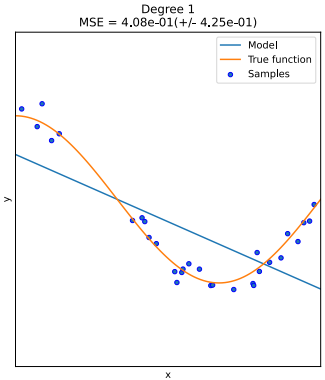


Approximation of a Transfer Function

Measurement of a relation between two quantities x and y

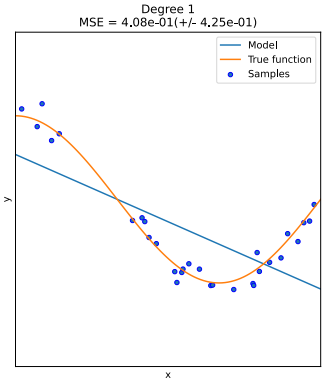
- To reduce the statistical error an adequate number of measurements should be acquired
- **Linear relation** $\rightarrow$ Linear regression (e.g. least-squares fit)
- **Non-linear relation**
  - Linearization (e.g. logarithmic function) followed by linear regression
  - Least-squares fit through numerical optimization techniques

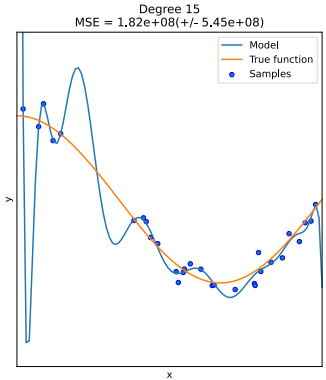
Approximation of a Transfer Function (cont).



https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

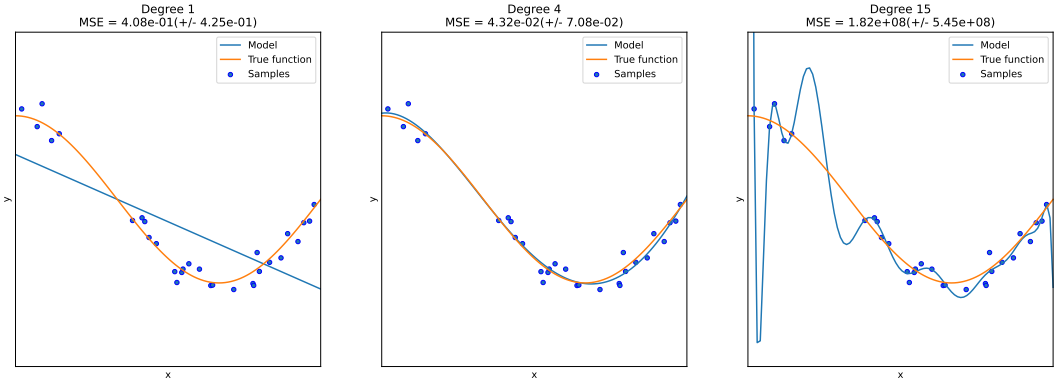
Approximation of a Transfer Function (cont).





https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Approximation of a Transfer Function (cont).



https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Real Transfer Function

- **Problem:** Unlike the ideal transfer function, the real transfer function is usually neither linear nor monotonic
- The ideal relation between stimulus and output signal is generally affected by
  - manufacturing tolerances,
  - material defects,
  - environmental influences,
  - wear and tear,
  - ...
- **Nevertheless:** Each sensor should work within the specified precision

Real Transfer Function (cont.)

- $S = f_{ideal}(s)$: The ideal transfer function
- $\pm\Delta$: Maximum deviation from the ideal transfer function
- $\pm\delta$: Actual deviation from the ideal transfer function

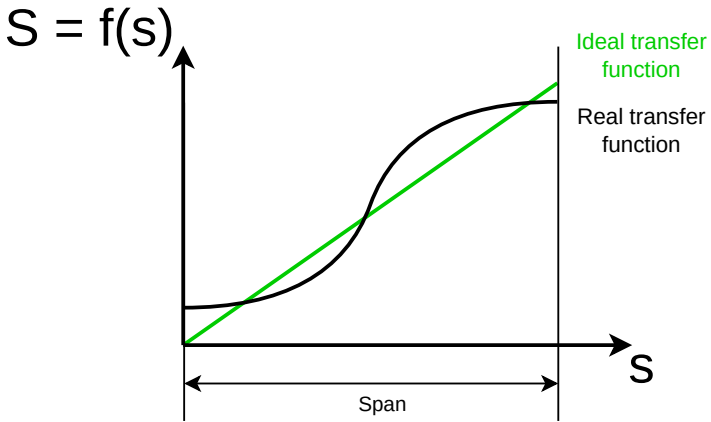
Definition

The physical relation between stimulus and output signal of a sensor is characterized by the real transfer function

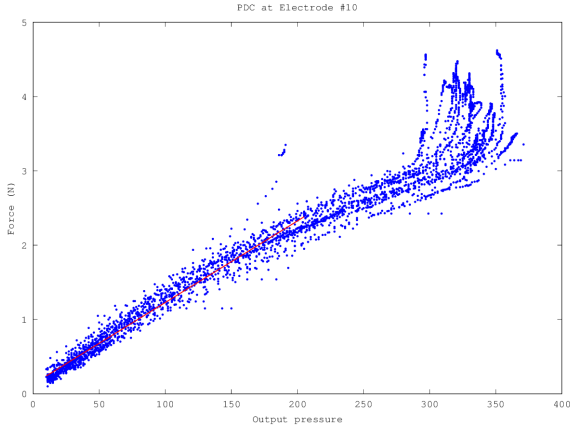
$$S' = f_{real}(s) = f_{ideal}(s) \pm \delta \qquad \delta \leq \Delta$$

- S' represents the measured value of the stimulus s

Real Transfer Function (cont.)



Real Transfer Function (cont.)



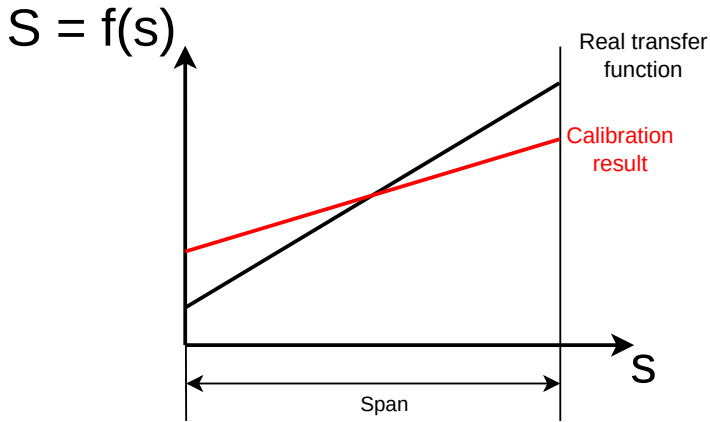
Calibration Error

- According to specification, a sensor has a linear transfer function
- However, manufacturing tolerances lead to different slopes

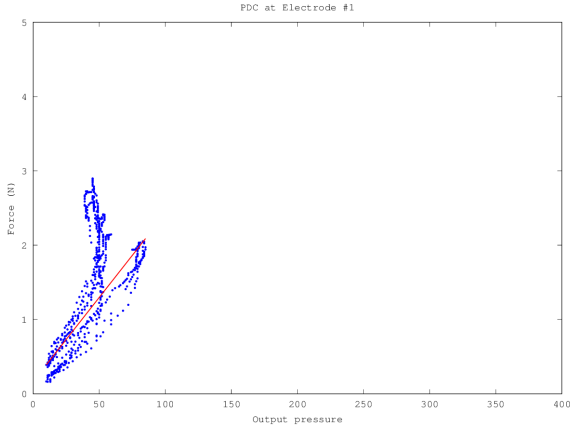
The calibration procedure

- The manufacturer determines the slope through:
  - Application of multiple stimuli $s_1, ..., s_n$ to the sensor
  - Measurement of the corresponding output signals $S_1, ...S_n$
  - Calculation of the slope based on the obtained value pairs
- *Caution:* Due to measurement errors, the slope may deviate from the real one if the pool of measured value pairs is too small

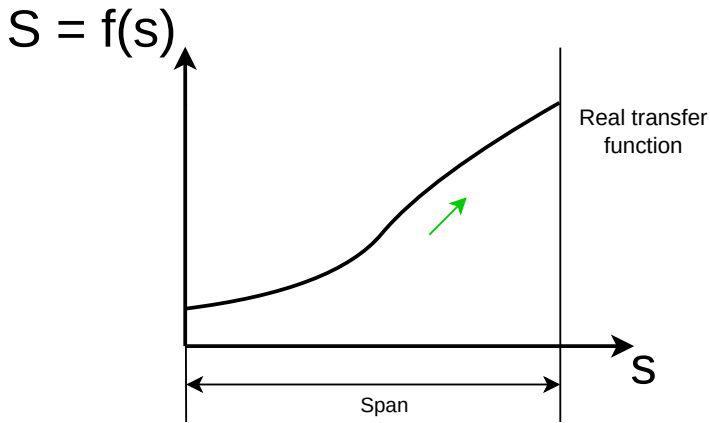
Calibration Error (cont.)



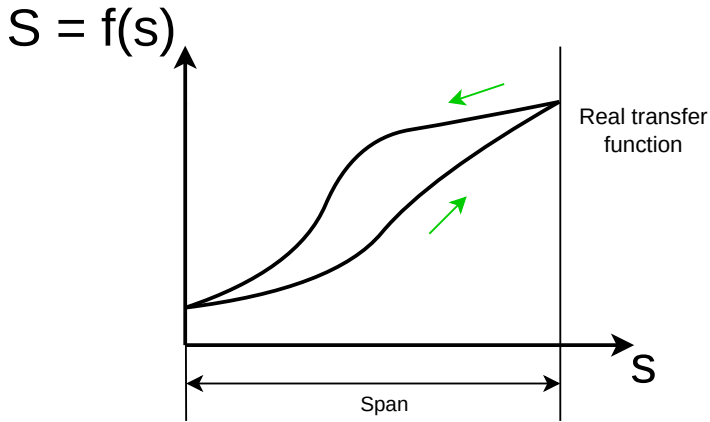
Calibration Error (cont.)



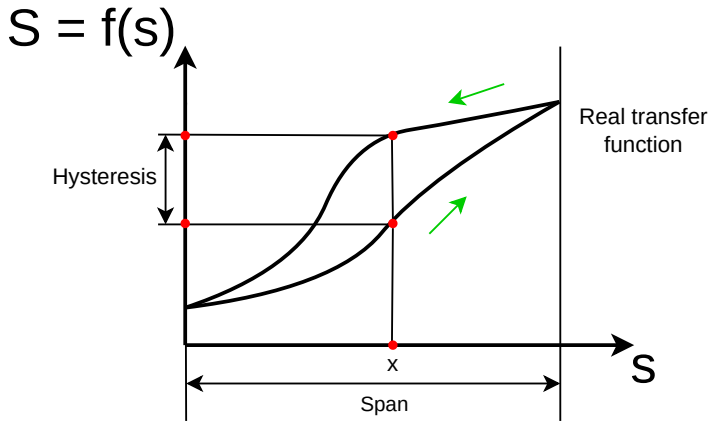
Hysteresis Error



Hysteresis Error



Hysteresis Error



Hysteresis Error (cont.)

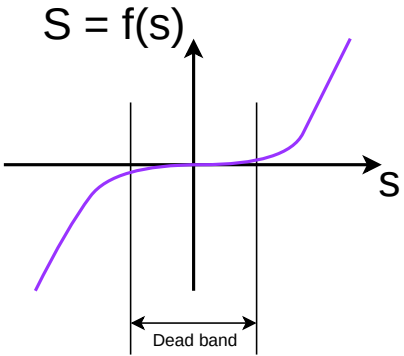
- Some sensors output different signals if the stimulus value is being approached from opposing directions of the range
- This deviation is called the hysteresis error

$$\lim_{\substack{\varepsilon \to 0, \\ \varepsilon > 0}} f(s+\varepsilon) \neq \lim_{\substack{\varepsilon \to 0, \\ \varepsilon < 0}} f(s+\varepsilon)$$

- **Examples:** Temperature sensor, displacement sensor, ...

Dead Band

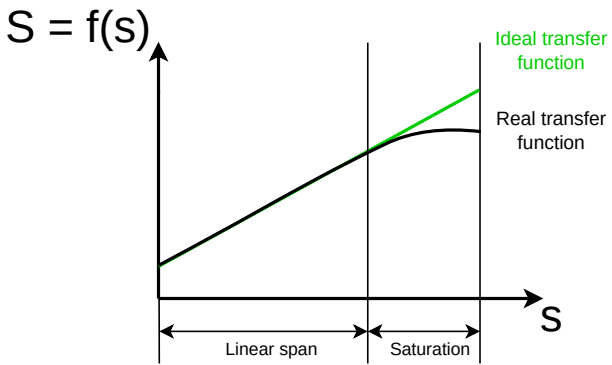
Dead band of a sensor: insensitivity within a coherent range of the input signal



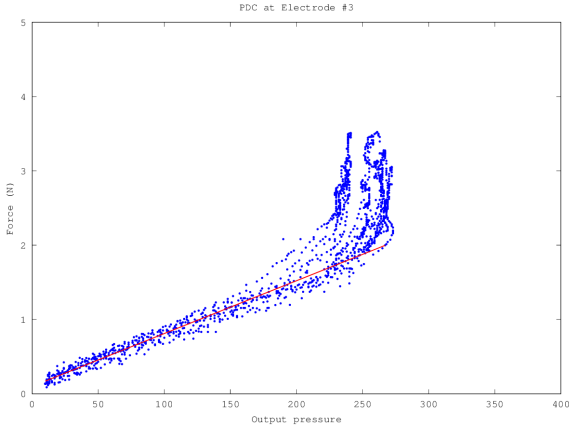
Saturation

- Every sensor has a limited operating range, the full scale input

- Many sensors have a linear transfer function
- However, from a certain stimulus value on the output becomes non-linear
- This effect is called saturation



Saturation (cont.)

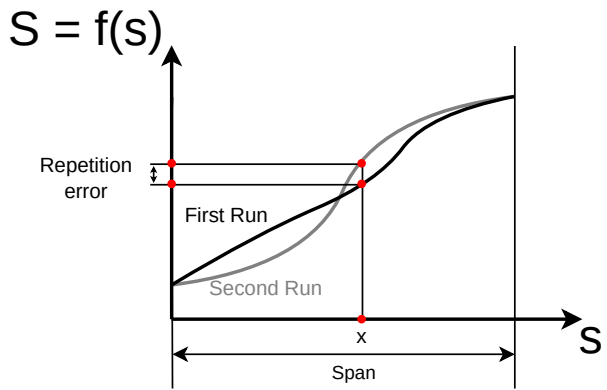


Repeatability Error

- A sensor may produce different output values under the same conditions
- This type of error is called repeatability error
- A repeatability error is usually determined as: Maximum distance Δ of two output signals for the same stimulus value
- Repeatability is specified in relation to the full scale input (FSI)

$$\delta_r = \frac{\Delta}{FSI} \cdot 100\%$$

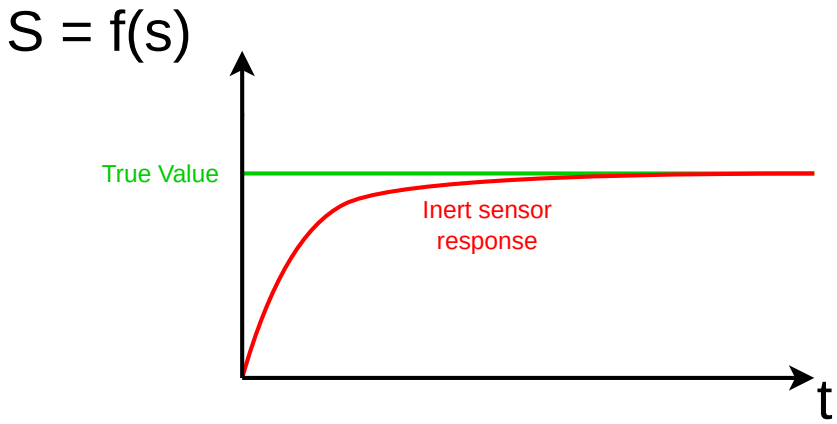
Repeatability (cont.)



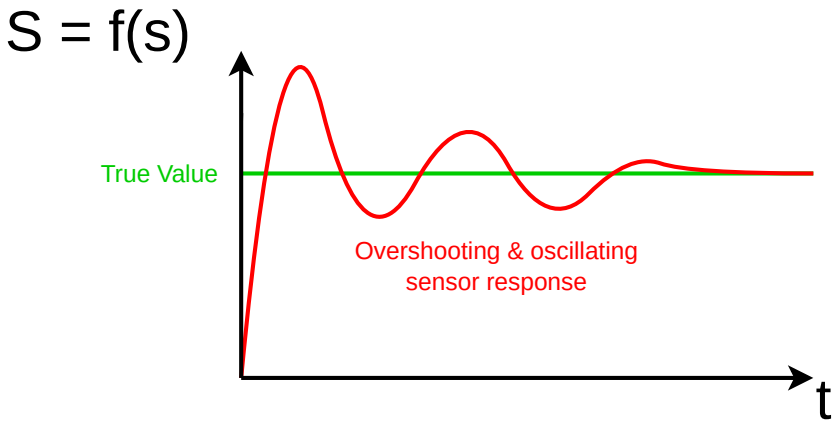
Dynamic Characteristics

- Under static conditions previously mentioned characteristics are enough to fully specify a particular sensor in fixed environmental conditions
- However, variation of the stimulus introduces time-dependency
- **Reason:** The sensor does not always provide an immediate response to the stimulus
- Therefore, a sensor does not always immediately output a signal corresponding to the stimulus
- Such effects are called the dynamic characteristics of a sensor
- The associated errors are called dynamic errors

Dynamic Characteristics (cont.)



Dynamic Characteristics (cont.)



Environmental Factors

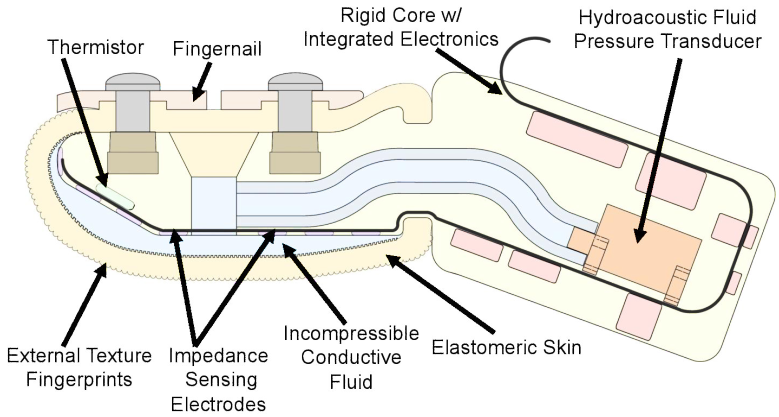
- Ambient temperature (minimum and maximum)
- Ambient air humidity (minimum and maximum)
- Atmospheric pressure
- Short- and long-term stability (drift)
- Static and dynamic changes of electromagnetic fields, gravitational forces, vibration, radiation etc.
- Self-heating (e.g. due to flow of current)

Further Sensor Characteristics

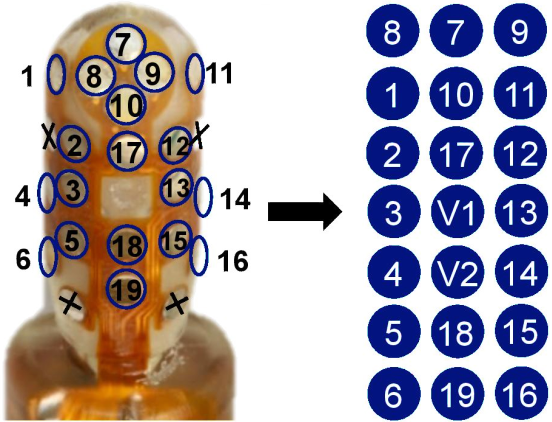
- Reliability, e.g. *mean time between failure* (MTBF)
- Certain properties relevant to the field of application:
  - Measurement rate
  - Shape/Size
  - Weight
  - Form factor
  - Price
  - ...

The BioTac Sensor

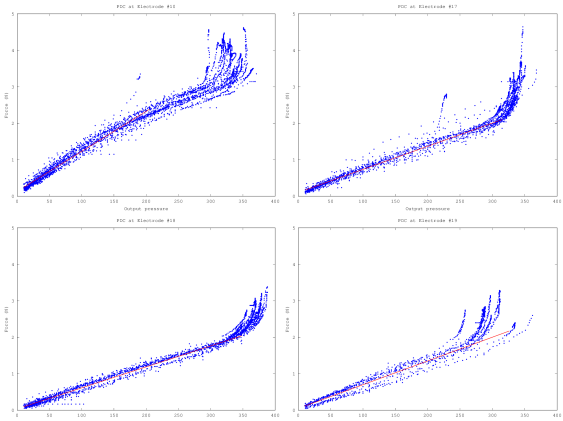
The BioTac Sensor



The BioTac Sensor



The BioTac Sensor

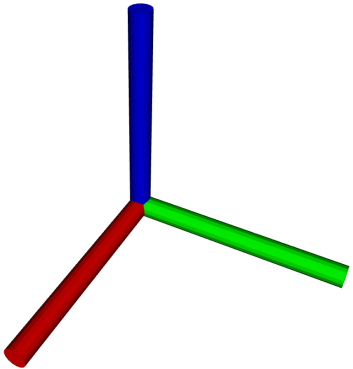


Datasheet Example

- Commercially available sensors typically come with a datasheet
- It specifies sensor errors, transfer functions, etc.
- Normally these values are correct, but some vendors “improve” their hardware in the datasheet
- Example
  - OPT 3001 Ambient Light Sensor
  - `http://www.ti.com/lit/ds/symlink/opt3001.pdf`

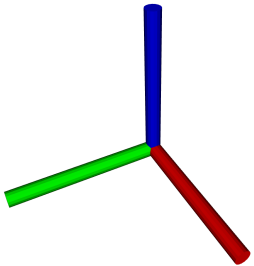
Outline

Coordinate Systems

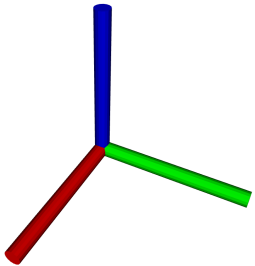


- Standard orthonormal basis for 3D Cartesian space
- RGB $\rightarrow$ XYZ for visualization
- When multiple CSs are relevant, usually called ***frame*** and associated with a *name*
- *There are many ways to specify frames w.r.t. each other*

Coordinate Systems - Handedness

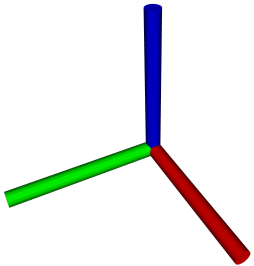


Left-Handed



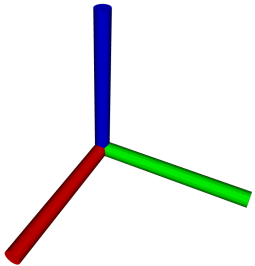
Right-Handed

Coordinate Systems - Handedness



Left-Handed

- DirectX, POV-Ray, Unity, ...

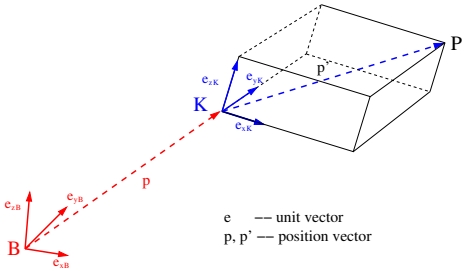


Right-Handed

- OpenGL, OpenCV, **ROS**, ...

Coordinate Systems - Relative Pose

- The **pose** of a rigid object comprises its **position** and **orientation** w.r.t. some CS
- It can be represented by
  - its Cartesian coordinate system (CS) **K** (frame) and
  - a transformation between CS **K** and e. g. the global CS **B** (**B** → **K**)

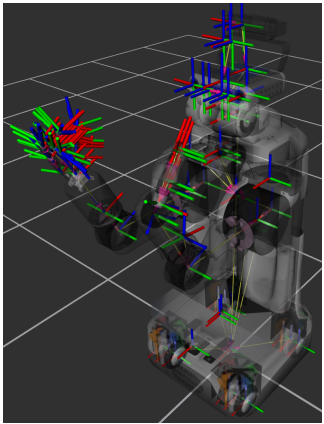


Coordinate Transformation

Transition between coordinate systems (frames)
Typical reference frames:

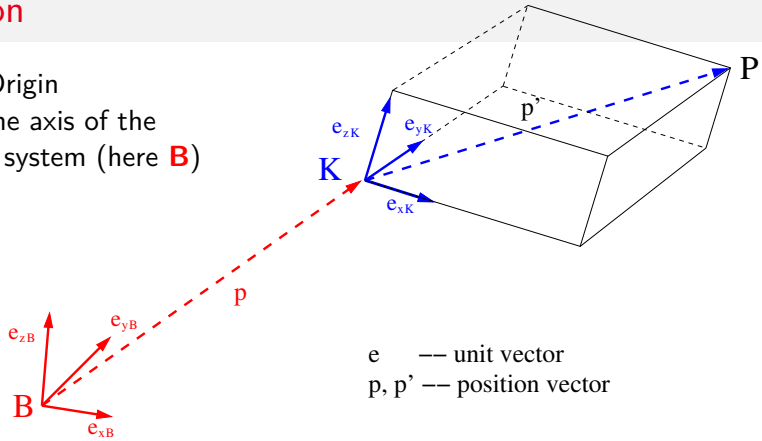
- Robot base
- Base footprint
- End-effector (tool)
- Table (world)
- Object
- Camera
- Screen
- ...

Frame transformations convert one frame into another.



Relative Pose - Position

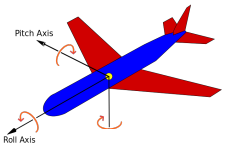
- Coordinates of the Origin
- Translations along the axis of the reference coordinate system (here **B**)



- Defined by: $\mathbf{p} = [p_x, p_y, p_z]^T \in \mathcal{R}^3$ w.r.t. e_{xB}, e_{yB}, e_{zB}

Relative Pose - Orientation (Spacial Alignment)

- Euler-angles ϕ, θ, ψ
  - Rotations performed in sequence around the axes of a coordinate system
  - e.g. Roll-Pitch-Yaw, $ZY'Z''$, ...
- Axis-Angle $\hat{e}, \theta$
  - Rotation around axis $\hat{e}$ by angle θ
  - Elegant algebraic version: Unit Quaternions
- Rotation matrix $R \in \mathcal{R}^{3\times3}$
  - 9 parameters for 3 DOF
  - For plain rotations: $det(R) = 1$



$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

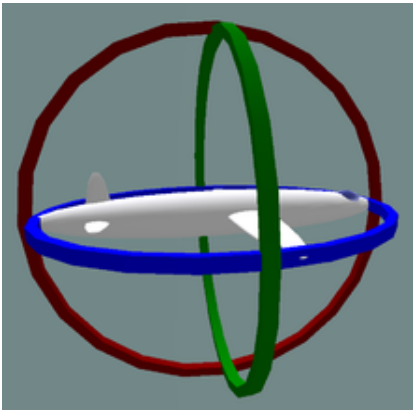
Euler Angles

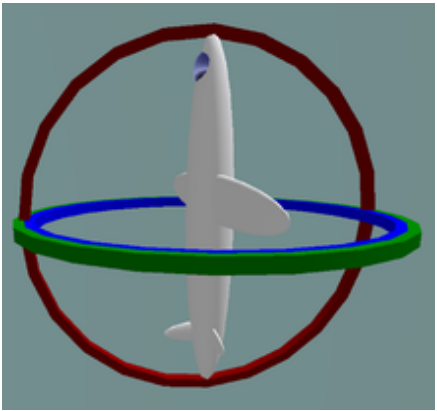
- Represented as rotations $\phi, \theta, \psi \in [0; 2\pi)$ around axes
- It remains undefined **around which axes!**
- *Proper Euler angles* rotate around:
  - $z - x - z$
  - $y - z - y$
  - $x - z - x$
  - $x - y - x$
  - $z - y - z$
  - $y - x - y$
- *Tait-Bryan angles* rotate around:
  - $x - y - z$
  - $z - x - y$
  - $z - y - x$
  - $y - z - x$
  - $x - z - y$
  - $y - x - z$
- All sequences can be interpreted w.r.t. the reference frame (**extrinsic**) or the partially rotated frame (**intrinsic**)

There are 24 possible interpretations of Euler angles!

A gimbal lock can happen!

Gimbal Lock





Axis-Angle / Unit Quaternions

Euler's Rotation Theorem Any rotation can be expressed as an elemental rotation around a single axis (The *Euler axis* $\hat{e}$).

Axis-Angle

- Represented as $\hat{e} = [x, y, z] \in \mathcal{R}^3$ and $\theta \in [0; 2\pi)$
- Often encoded as $\hat{e}$ only, where $\|\hat{e}\| = \theta$

Unit Quaternions

- Represented as $q = w + x \cdot \mathbf{i} + y \cdot \mathbf{j} + z \cdot \mathbf{k}$ with unit vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$ and $\|q\| = 1$
- Usually encoded as $[x, y, z, w]$ *or* $[w, x, y, z]$

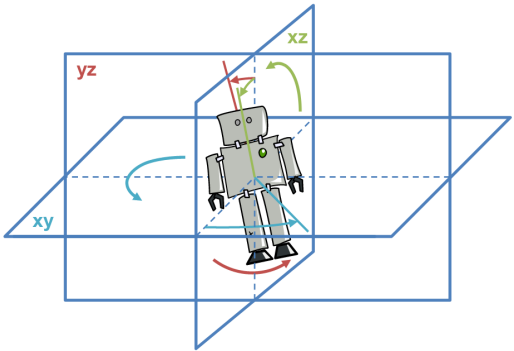
Rotation Matrix

- Representing rotations as 3×3 matrix
- Rotations can be summed up by multiplying the matrices
- High degree of redundancy

$$R_{\psi,\theta,\phi} = \begin{bmatrix} C\psi C\theta & C\psi S\theta S\phi - S\psi C\phi & C\psi S\theta C\phi + S\psi S\phi \\ S\phi C\theta & S\psi S\theta S\phi + C\psi C\phi & S\psi S\theta C\phi - C\psi S\phi \\ -S\theta & C\theta S\phi & C\theta S\phi \end{bmatrix}$$

Relative Pose - Orientation (Spacial Alignment)

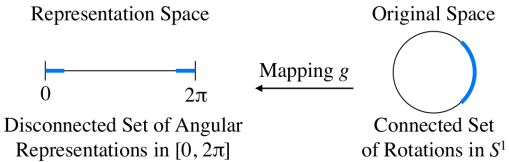
- Fused
  - Relatively recent (2015)
  - Especially made with the goal of balancing robots
  - Angle towards the three basis planes
  - Rarely used



Allgeuer, Philipp, and Sven Behnke. "Fused Angles: A representation of body orientation for balance." 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015.

Relative Pose - Orientation (Spacial Alignment)

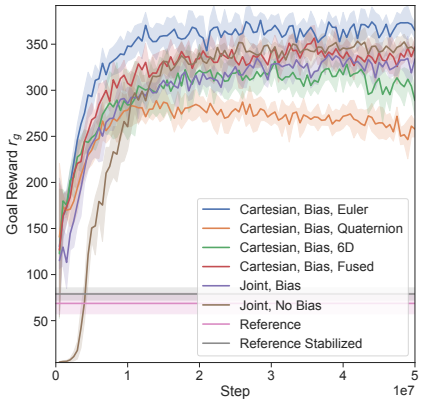
- 6D
  - Recent form of representation (2019)
  - Specifically proposed for usage with NN
  - Ensures continuity of rotation
  - Representation is done with 2 orthogonal unit vectors



Zhou, Yi, et al. "On the continuity of rotation representations in neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

Example from Reinforcement Learning Research: Why Choice of Rotation Matters

- Depending on your use case Euler Angles can be good
- Learning bipedal walk with RL
- Foot orientation is always $< +- 45\,\text{deg}$
- No gimbal issues
- Network learns better with Euler
- Quaternions are generally bad for NN

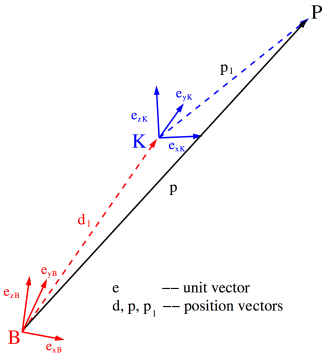


Coordinate Transformation

Points in space referenced in a local frame can be transformed into a global frame through:

- Translations
- Rotations

$$^{B}p = {}^{B}d_1 + {}^{B}R_K \; {}^{K}p_1$$



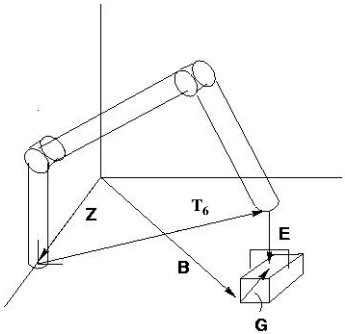
Please note:

- ${}^{B}R_K$: rotation matrix R, that describes rotations to generate frame K from frame B
- ${}^{K}p$: vector p based on frame K.

Application: Relative Transformations

There are the following transformations:

- Z: World $\rightarrow$ Base of the manipulator
- T_6: Base of the manipulator $\rightarrow$ End of the manipulator
- E: End of the manipulator $\rightarrow$ End effector
- B: World $\rightarrow$ Object
- G: Object $\rightarrow$ Grasp Pose



Application: Transformation Chain

There are two descriptions of the end effector's frame, one in relation to the object and the other in relation to the manipulator. Both descriptions are equal:

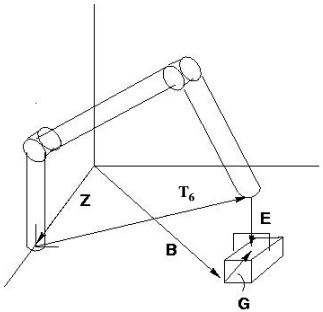
$$ZT_6E = BG$$

To find the manipulator transformation:

$$T_6 = Z^{-1}BGE^{-1}$$

To determine the coordinate frame of the object:

$$B = ZT_6EG^{-1}$$



The transformation chain is also called the *kinematic chain*.

In Reality

If you are programming anything robotic related just use a library which does the transformations for you.
If you use Euler Angles you do have to be aware of their downsides and the order that this library uses.
Generally try to use quaternions, just not for the input/output of NNs.

Examplary libraries:

- tf2 (ROS, C++, Python)
- Robotics Library (C++)
- Robotic System Toolbox (Matlab)
- transforms3d (Python)

Want to Know More?

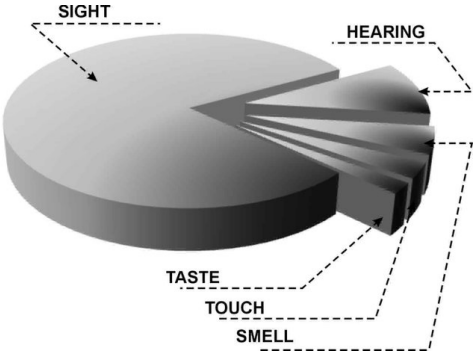
For a more in depth lecture about transformations in the robotic domain, please attend "Introduction to Robotics" in the summer term.

You can find materials here:
https://tams.informatik.uni-hamburg.de/lectures/2019ss/vorlesung/itr/index.php?content=03-documents

Explanations of Quaternions:
https://www.youtube.com/watch?v=zjMuIxRvygQ
https://eater.net/quaternions
https://www.youtube.com/watch?v=jTgdKoQv738

Outline

Introduction - Human Vision



Man, D., & Olchawa, R. (2018). The possibilities of using BCI technology in biomedical engineering. In Biomedical Engineering and Neuroscience: Proceedings of the 3rd International Scientific Conference on Brain-Computer Interfaces, BCI 2018, March 13-14, Opole, Poland (pp. 30-37). Springer International Publishing.

Introduction (cont.)

Various vision systems are used in robotic applications as a fundamental tool for environmental perception

- Much of the information obtained by the human brain includes vision as a dominating/contributing modality
- Roughly 60% of the brain are said to be dealing with visual data - mostly in combination with other modalities

Some of the questions computer/machine vision research is trying to answer are:

- *Where am I?* (Scene modeling, classification, recognition, etc.)
- *What is around me?* (Object detection/recognition, etc.)
- *How can I interact?* (Structure estimation, tracking, etc.)

Introduction (cont.)

Vision sensor technology is evolving steadily, with many different options available

- Linear camera sensor
- Analog CCD camera (black/white or color)
- Standard CMOS camera (e.g. web cam)
- High-Dynamic-Range (HDR) CMOS camera
- Structured light camera (Infrared, RGB)
- Stereo vision system
- Omnidirectional vision system
  - Typically a normal camera with a 360 deg mirror or wide angle lens
- Event cameras
  - Transmit pixel changes rather than images

Event Kamera Video

https://www.youtube.com/watch?v=eomALySSGVU

Introduction (cont.)

Vision systems are widely used in industrial automation applications

- Object grasping tasks
  - Objects with predetermined position (e.g. production line)
  - Randomly positioned objects (e.g. 'bin-picking')
- Object handling tasks
  - Cutting, tying, wrapping, sealing, etc.
  - Inspection during assembly
- Assembly tasks
  - Welding, gluing, attaching, etc.

Introduction (cont.)

Introduction (cont.)

Vision systems in robotics are used for a wide range of applications

- Perception of objects
  - *Static:* Recognition, searching, indexing, ...
  - *Dynamic:* Tracking, manipulation, ...
- Perception of humans
  - Face recognition
  - Gaze tracking
  - Gesture recognition
  - ...

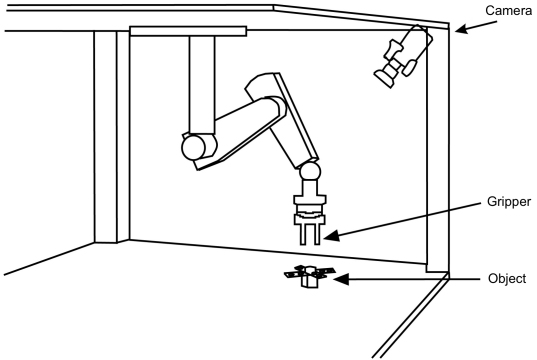
Introduction (cont.)

- Navigation and modeling of the environment
  - Robot localization:
    - Absolute
    - Relative
    - In reference to various coordinate frames
  - Object recognition and localization
  - 3D scene reconstruction
  - Allocation of the environment

Introduction (cont.)

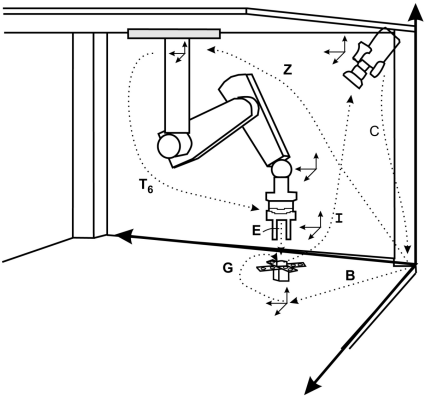
- Vision-based motion control
  - Visual-Servoing
    - Coarse and fine positioning
    - Tracking of movable objects
  - Collision avoidance
    - Depth map based distance measurement
  - Coordination with other robots and/or humans
    - Motion estimation
    - Intention recognition

Application scenario



Very common application scenario: Vision-based manipulator control.

Application scenario (cont.)



Chain of transformations of the common application scenario.

Kinematic chain

Data (points) can be easily transformed between coordinate frames using a suitable transformation sequence of the *kinematic chain*

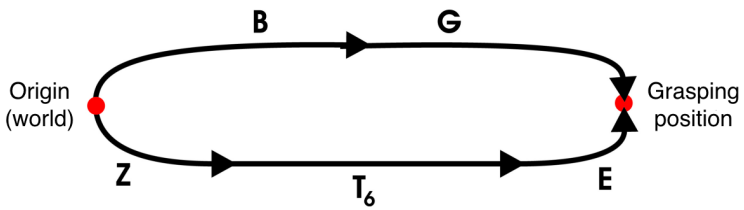
Z: Transformation from world coordinates to manipulator base coordinates
T_6: Compound transformation from the base of the manipulator to the end of the manipulator
E: Transformation from the end of the manipulator to the gripper

B: Transformation from world coordinates to object coordinates
G: Specification of the grasp coordinate frame in reference to the object coordinate frame

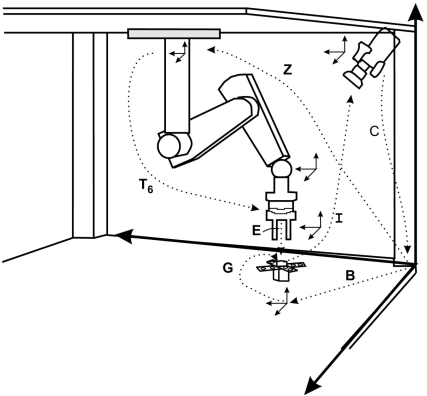
Kinematic chain (cont.)

During grasping and manipulation of the object, the coordinates of the grasping point can be determined in two ways:

$$ZT_6E = BG$$



Kinematic chain (cont.)



Chain of transformations of the common application scenario.

Kinematic chain (cont.)

To determine the transformation of the manipulator, the following equation needs to be solved:

$$T_6 = Z^{-1}BGE^{-1}$$

In order to determine the location of the object after manipulation, one has to solve:

$$B = ZT_6EG^{-1}$$

Kinematic chain (cont.)

A camera included in the scenario introduces two additional transforms:

C: Transformation of camera coordinates into world coordinates
(*Off-line* determination of the transformation through camera calibration)

I: Transformation of grasping point coordinates into the camera coordinate frame
(Grasping point is determined using image processing techniques)

Kinematic chain (cont.)

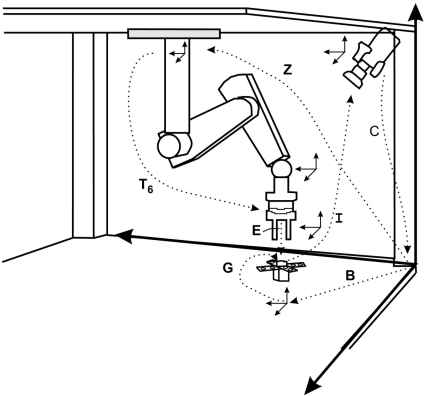
The transformation P from the grasping point to the world coordinate system is given as:

$$P = I\ C$$

The camera to world transformation is determined through the following equation:

$$C = I^{-1}\ P$$

Kinematic chain (cont.)



Chain of transformations of the common application scenario.

Outline

Camera calibration

Camera calibration in the context of image processing is the determination of the intrinsic and/or extrinsic camera parameters

Intrinsic parameters

- Internal geometrical structure and optical features of the camera

Extrinsic parameters

- Three-dimensional position and orientation of the camera's coordinate frame in relation to a reference frame

Camera calibration (cont.)

What information does one get?

In order to reconstruct 3D information of the environment from two or more images, it is necessary to know the relation between the coordinate frames of the 2D image and the objects of the 3D environment

- The relation between 2D and 3D can be described using two transformations
  - Perspective projection (3D $\rightarrow$ 2D)
  - Backprojection (2D $\rightarrow$ 3D)

Camera calibration (cont.)

Perspective projection (3D point → 2D point)

- Knowing the camera projection matrix the 2D projection of a 3D point can be determined (approximated)

Backprojection (2D point → 3D beam)

- If a 2D image point is known, there is a ray in 3D space on which the corresponding 3D point lies
- If there are two or more views of the 3D point, its coordinates can be determined using *triangulation*

Camera calibration (cont.)

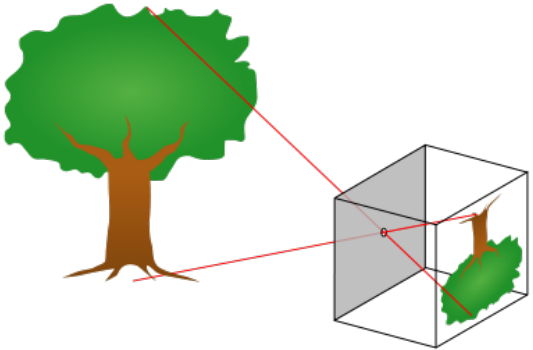
- The perspective projection is useful in order to reduce the search space during scene analysis
- The backprojection is helpful for deriving 3D information based on features in 2D images
- These transformations are used in various applications:
  - Automatic assembly
  - Robot calibration
  - Tracking
  - Trajectory analysis
  - 3D metrology
  - ...

Camera calibration (cont.)

Several calibration techniques have been established

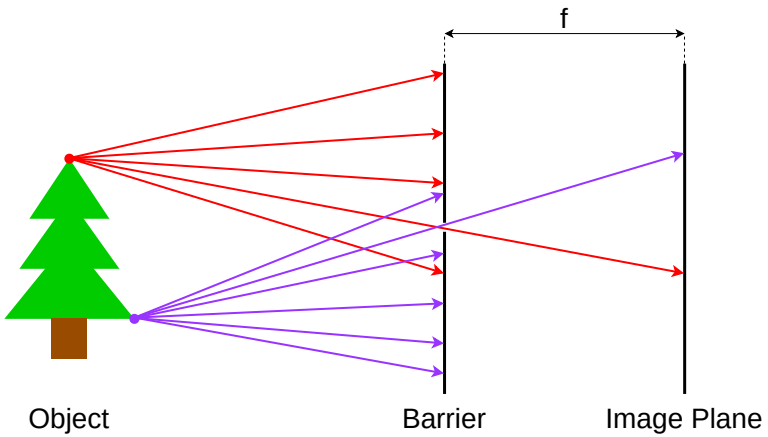
- Camera calibration can be done *on-line* or *off-line*
- Using a calibration object:
  - Identification of the camera parameters
  - Determination of coordinate transformation between camera coordinates and world coordinates
- Using self-calibration approaches
- Using machine learning methods

Pinhole camera model

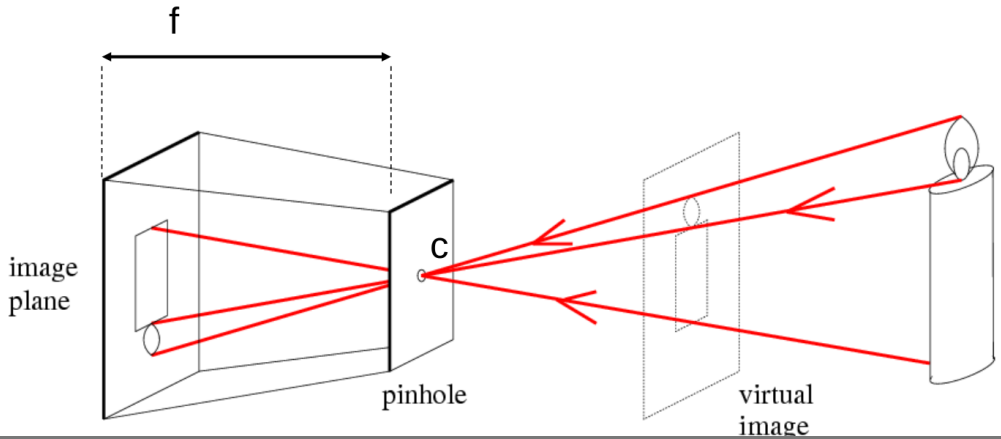


Pinhole camera without lens distortion

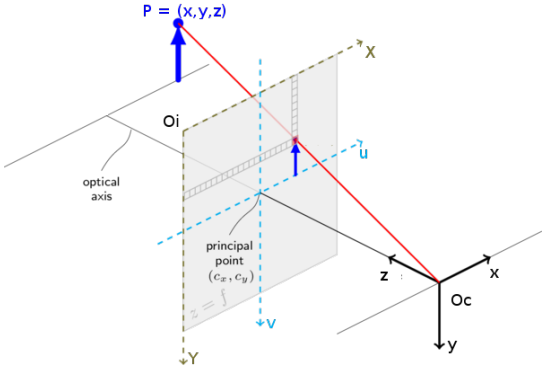
Pinhole camera model



Pinhole camera model



Pinhole camera model 2D



Pinhole camera without lens distortion [OpenCV]

Pinhole camera model (cont.)

- (x_w, y_w, z_w): 3D world coordinate system with the origin O_w [m]
- (x, y, z): 3D coordinate system of the camera with the origin O_c (optical center) [m]
- (X, Y): 2D image coordinate system with the origin O_i [pixels]
- f: Focal length of the camera [m]

Pinhole camera model (cont.)

Projection of camera coordinates onto image coordinates

- Point P is projected onto the corresponding (ideal) image coordinate (u, v)
- *Perspective projection* with focal length f:

$$u = f\frac{x}{z} \qquad v = f\frac{y}{z}$$

- The image coordinates (X, Y) are calculated from (u, v) as follows:

$$X = s_x u + C_x \qquad Y = s_y v + C_y$$

- The scaling factors s_x and s_y are used to convert the image coordinates from meters to pixels [pixels/m]
- C_x, C_y are the coordinates of the principal point in the image

Pinhole camera model - Assumptions

- Intrinsic Assumptions (typical error source in braces)
  - Pixels are squared (sensor manufacturing)
  - Optical center in the center (sensor misplacement)
  - No skew (inaccurate synchronization of the pixel-sampling process)
- Extrinsic Assumptions
  - No rotation
  - Camera at (0,0,0) - no translation

Intrinsic Parameters

Transformation from camera to image coordinates

- Let P(x,y,z) be a point in the camera coordinate system
- Let $f_x = s_x * f$ and $f_y = s_y * f$
- Let θ be the skew angle between x and y axis
- Let C_x, C_y be the principal points coordinates in image plane
- Its projection into the camera coordinate system can be determined as follows:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ with } \quad K = \begin{bmatrix} f_x & -f_x cot(\theta) & c_x \\ 0 & f_y / sin(\theta) & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- K is the camera matrix consisting of the intrinsic parameters

Extrinsic Parameters

Transformation from world to camera coordinates

- Let $P(x_w, y_w, z_w)$ be a point in the world coordinate system
- Its projection into the camera coordinate system can be determined as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + t$$

$$\text{with} \quad R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \quad \text{and} \quad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

- The parameters R and t are the extrinsic parameters

Calibration parameters

The pinhole camera model contains the following calibration parameters:

- The three independent extrinsic parameters of R
- The three independent extrinsic parameters of t
- The intrinsic parameters f_x, f_y, θ, C_x and C_y

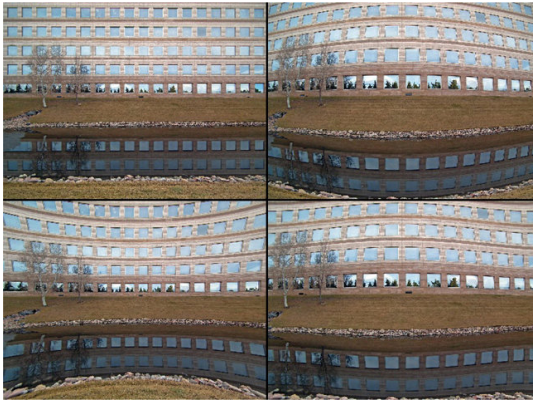
Lens distortion

Most real cameras have lenses and produce a variety of imaging errors and do not satisfy constraints of the pinhole camera model

The main error sources are:

- Low spatial resolution due to low resolution of the camera device being used
- Most (cheap) lenses are asymmetrical and generate distortions
- Imprecision during assembly (pose of the lens or sensor)

Lens distortion (cont.)



Terpstra, T., Miller, S., & Hashemian, A. (2017). An Evaluation of Two Methodologies for Lens Distortion Removal when EXIF Data is Unavailable (No. 2017-01-1422). SAE Technical Paper.

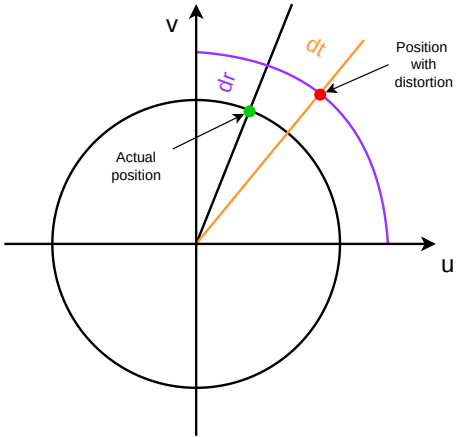
Lens distortion (cont.)

- Distortion by the lens system results in an altered position of the image pixels on the image plane
- The pinhole camera model is no longer sufficient
- It is replaced by the following model:

$$u' = u + D_u(u, v)$$
$$v' = v + D_v(u, v)$$

where u and v are the non-observable, distortion-free image coordinates, and u' and v' the corresponding distorted coordinates

Lens distortion (cont.)

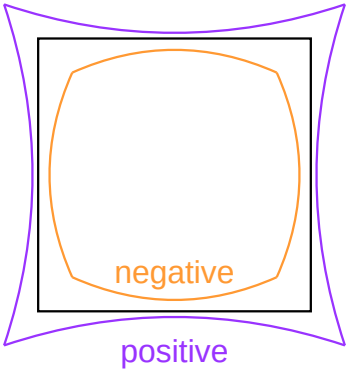


Lens distortion (cont.)

Types of distortion

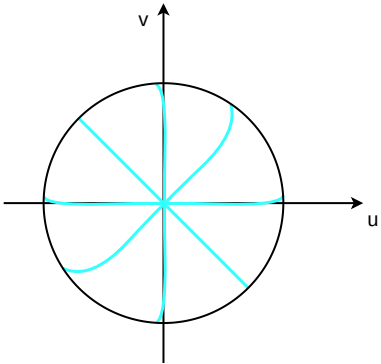
- There are **two** primary types of distortions:
  - *Radial*
  - *Tangential*
- Radial distortion causes an offset of the ideal position inwards (barrel distortion) or outwards (pincushion distortion)
  Possible cause: Flawed radial bend of the lens
- Tangential distortion shifts the ideal position along a tangential curve
  Possible cause: Non-parallel sensor/lens

Lens distortion (cont.)



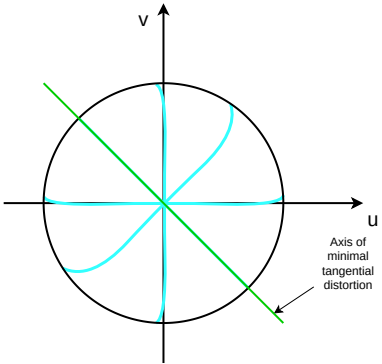
Radial distortion: Straight lines → no distortion

Lens distortion (cont.)



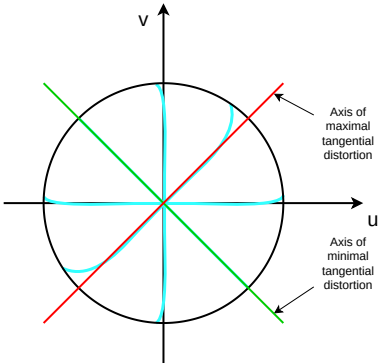
Tangential distortion: Straight lines → no distortion

Lens distortion (cont.)



Tangential distortion: Straight lines → no distortion

Lens distortion (cont.)



Tangential distortion: Straight lines → no distortion

Camera model

Since radial lens distortion is the dominant type, the following equations can be used to establish a simplified, yet more correct camera model:

$$X_d = X(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$Y_d = Y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$\text{with} \quad r^2 = x^2 + y^2$$

Where X_d, Y_d are the distorted image coordinates.

Calibration

The problem camera calibration procedures are trying to solve is the identification of the unknown parameters of the camera model

- The computation of these parameters for the distortion-free camera model yields the position of the camera in world coordinates

Calibration requires a set of m object points, which:

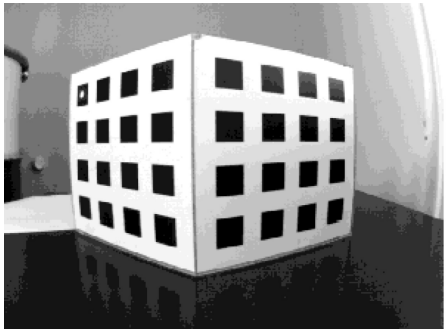
1. Have known world coordinates $\{x_{w,i}, y_{w,i}, z_{w,i}\}$, $i = 1, \ldots, m$ with sufficiently accurate precision
2. Lie within the camera's field of view

These *calibration points* are detected in the camera image with their respective image coordinates $\{X_i, Y_i\}$

Calibration (cont.)

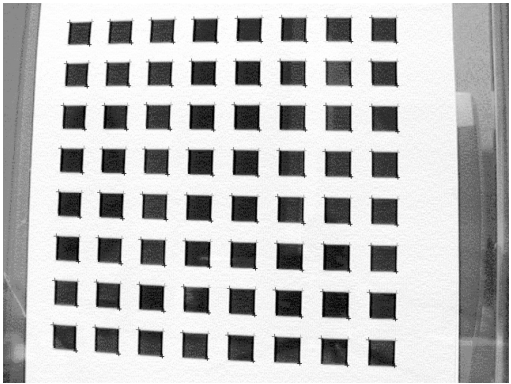
- Having these known world coordinates can be tricky
- Only calibrating intrinsic parameters is easier, since we only need known points in camera frame
- A checkerboard with known square sizes is enough
- Dependent on the use case, intrinsic parameters may be enough

Intrinsic Calibration Objects



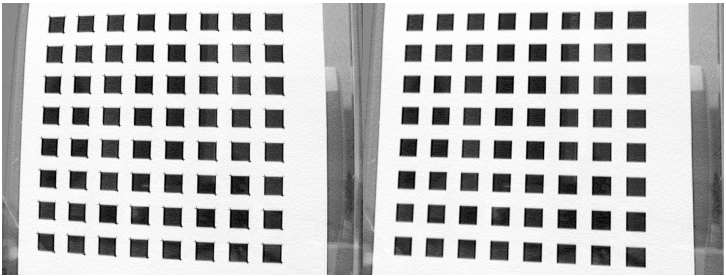
A typical 3D calibration object

Intrinsic Calibration Objects



Typical calibration pattern

Use of Intrinsic Calibration Parameters



Using the determined camera parameters, the image can be undistorted

In the Real World

Different open source implementations exists that do the camera calibration for you.

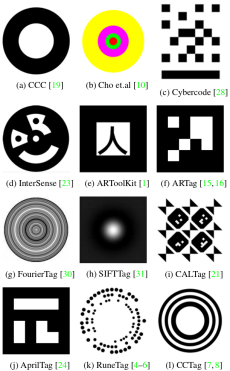
- camera_calibration (ROS package)
- Interactive camera calibration application (OpenCV)
- Camera Calibrator (Matlab)

Outline

Fiducial Markers

- In robotics, we are often interested in getting transforms
  - Robot -> Object
  - Robot -> World
- Getting these directly from camera images is difficult
- Computing poses of known, well-structured elements is easy
- Those are called *fiducial markers* or short *tags*
- Tags can be used for extrinsic calibration
- Also widely used in medicine, pcb assembly, and augmented reality

Fiducial Markers

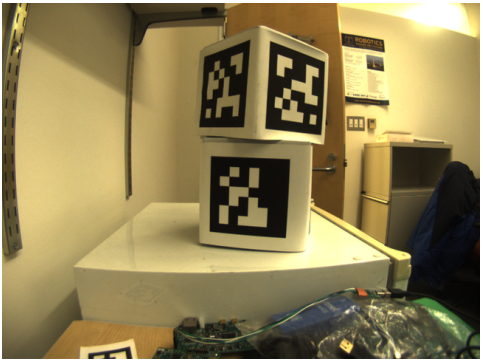


From *DeGol et a., ChromaTag: A Colored Marker and Fast Detection Algorithm*

AprilTag

- One of the most used marker types in robotics is the AprilTag
- There are different families available
- The widely used Tag36h11 family offers 587 tags with minimal Hamming distance of 11 bit
- For each tag you get the ID and the pose
- Very low false-positive detection rates
- False-negative detection highly depends on image quality
- Multiple tags can be grouped as bundle to get better measurements

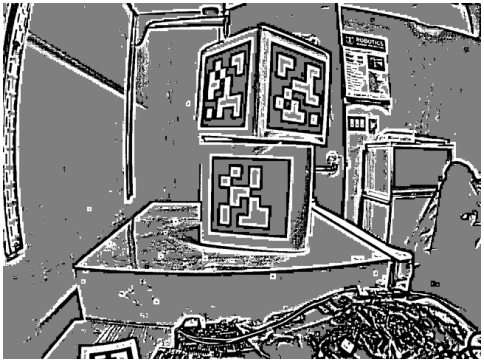
AprilTag Detection



Original image

From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

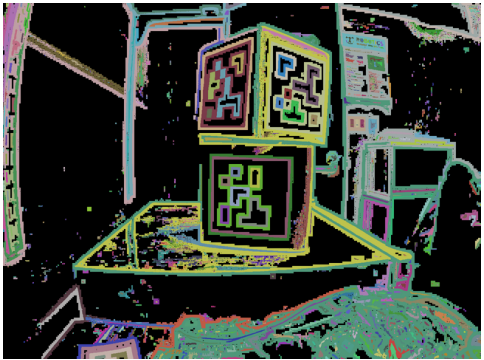
AprilTag Detection (cont.)



Adaptive thresholding

From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

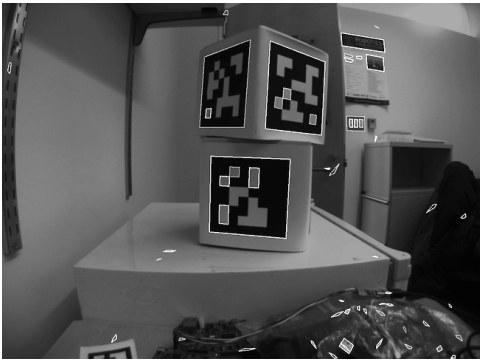
AprilTag Detection (cont.)



Segmentation

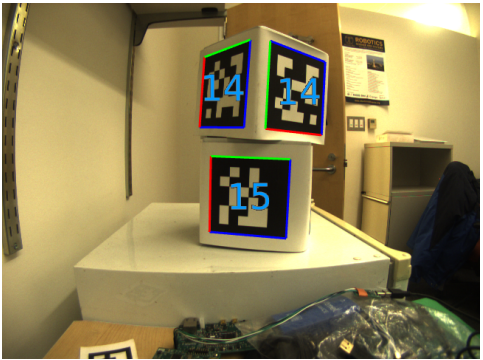
From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

AprilTag Detection (cont.)



From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

AprilTag Detection (cont.)



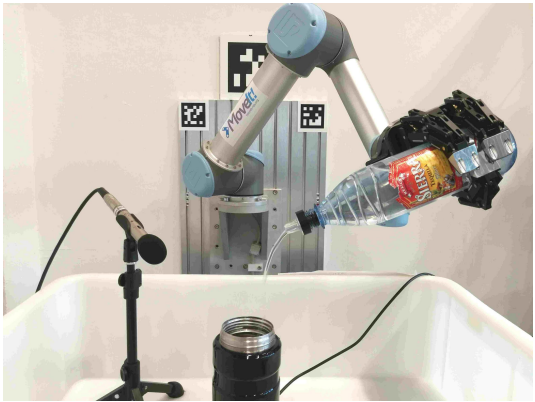
Tag detections

From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

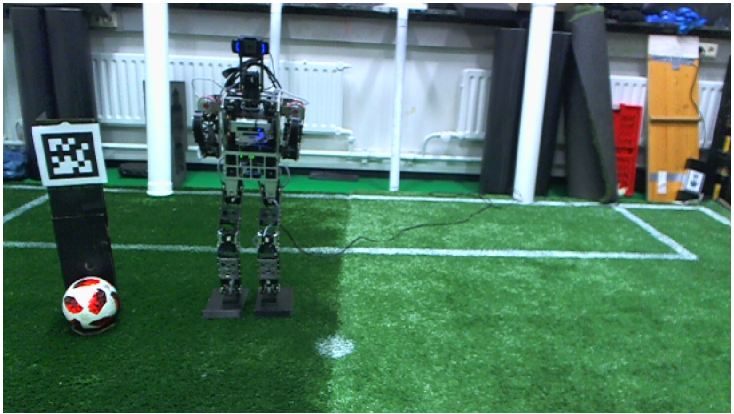
Tag Application Examples

- Extrinsic calibration
- Getting “ground truth” for training data
- Removing the object pose estimation sub task from a larger task, to concentrate on the other aspects

Tag Application Examples



Tag Application Examples



Discussion of Tags

- Pro
  - Simplifies tasks
  - Production costs very low
  - Computational cost comparably low
  - Normally quite precise
  - Ease to employ in controllable environments, e.g. factory
- Contra
  - Difficult to use on large distances
  - Difficult to use on very small scales, e.g. getting finger tip positions
  - Not usable for natural environments
  - Tags in training data lead to learning tag detection
  - Rotation is not so precise (can be improved by bundles)

Summary

- Relies on computing transformation sequences between world, object, reference, camera (...) frames
- Camera calibration requires a set of free parameters to be tuned (can be achieved by a suitable optimization technique)
  - Essential for retrieving accurate sensor measurements
  - Is typically done by using a geometric object with structured visual pattern
  - Lens distortion (radial / tangential) needs to be incorporated
- Usage of tags can largely simplify the task

Outline

Rotation and Motion Measurements

- We measure rotation and motion in various parts of robots

Rotation and Motion Measurements

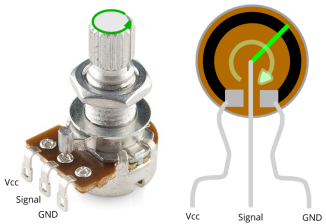
- We measure rotation and motion in various parts of robots
  - Wheels
  - Arms/Legs
  - Pan/Tilt units (sensor and camera position)

Rotation and Motion Measurements

- We measure rotation and motion in various parts of robots
  - Wheels
  - Arms/Legs
  - Pan/Tilt units (sensor and camera position)
- Depending on the application, different features are required
  - Rotatory/linear
  - Incremental/absolute
  - Continuous/discrete
  - Precision
  - Robustness
  - Type of output
  - Etc.

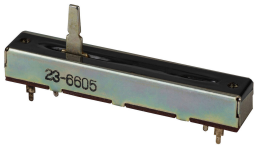
Potentiometer

- A potentiometer gives a resistance value in relation to its absolute position
- Often used in user interfaces but also in (cheap) servo motors
- Has (comparably) high wear due direct contact of the material
- 360° turn not possible



https://core-electronics.com.au/tutorials/potentiometers-and-the-arduino-uno.html

Linear Potentiometer



https://produktefee.de/live-mischpult/

https://www.parts-express.com/B25K-20K-Ohms-Slide-Potentiometer-2-Travel-with-Dust-Cover-029-112

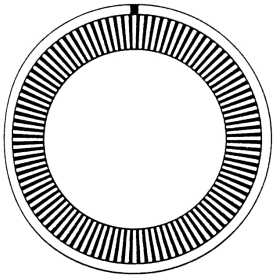
Optical encoder

Use of an optical encoder is a well established approach to measurement of angular or linear motion

- The main component is a mask with transparent and opaque areas
- A ray of light cast onto the mask is registered by a photodiode located on the opposite side
- The mask pattern is usually manufactured as a disk or a strip
- **Disk:** Measurement of angular motion (rotation)
- **Strip:** Measurement of linear motion (translation)
- Measurement with respect to time yields angular/linear velocity

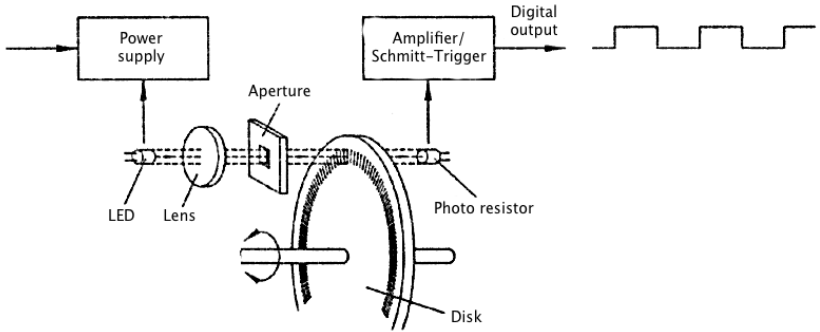
Incremental encoder

- The mask of an incremental encoder consists of equidistant, transparent and opaque areas equal in size



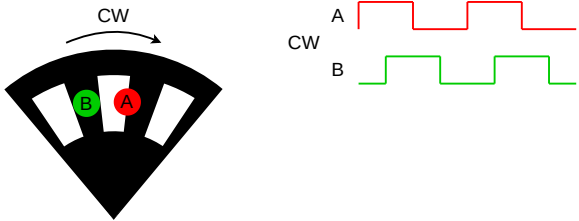
Incremental encoder (cont.)

- A simple (single channel) incremental encoder requires only a single LED [1] and photodiode in order to register motion



Dual channel incremental encoder

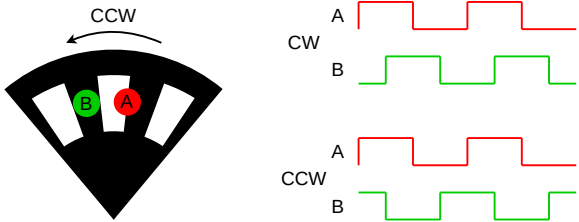
- Using two LEDs and photodiodes (channels A and B) the direction of angular/linear motion can be determined



- Quadrature encoder: Separation of A and B by 90° (in phase)
- Clockwise (CW) rotation → signal A leads

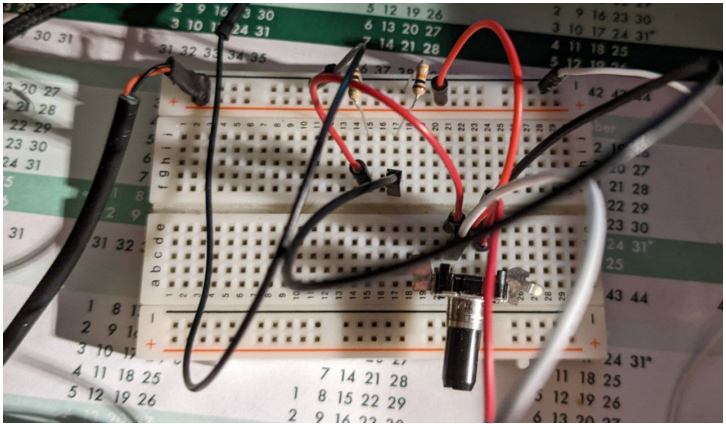
Dual channel incremental encoder

- Using two LEDs and photodiodes (channels A and B) the direction of angular/linear motion can be determined



- Quadrature encoder: Separation of A and B by 90° (in phase)
- Clockwise (CW) rotation → signal A leads
- Counter-clockwise (CCW) rotation → signal B leads

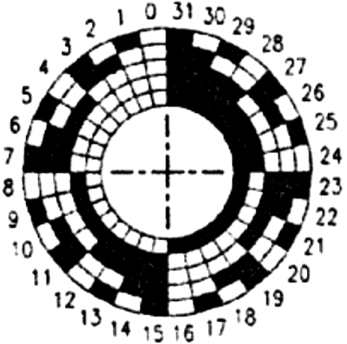
Dual channel incremental encoder (cont.)



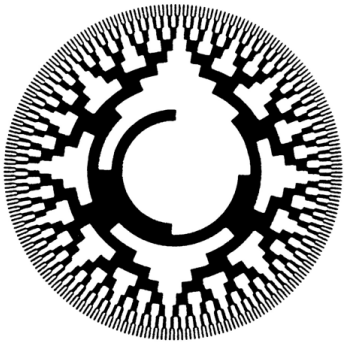
Absolute encoder

- In contrast to an incremental encoder, an absolute encoder provides absolute angles as its output signal
- Advantages:
  - Less errors due to slippage or jumps
  - Initial position not necessary to get current position
- Absolute encoder uses disk/strip with a binary-encoded pattern
- Several LEDs and photodiodes are used to scan the disk/strip
- One unique binary code is allocated to each resolution step
- Resolution directly affects the measurement accuracy

Absolute encoder (cont.)

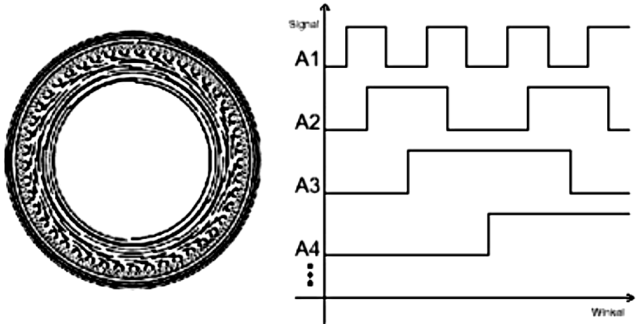


5 bit = 32 steps (11.25°)



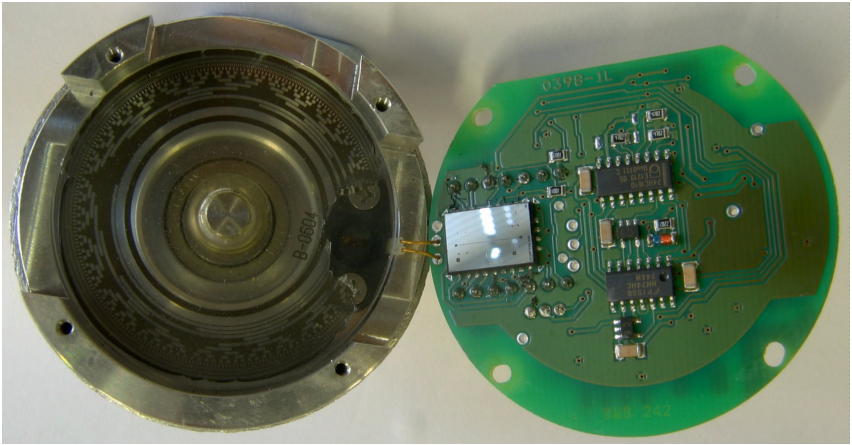
10 bit = 1024 steps (≈ 0.35°)

Absolute encoder (cont.)



- Gray-coded position results in exactly one signal change per tick
- Useful to allow measurement during tick-transition

Absolute encoder (cont.)



Linear absolute encoder

http://www.ladyada.net/wiki/tutorials/learn/calipers/index.html

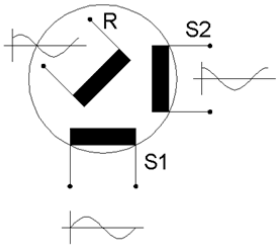
Comparison

Absolute vs. Incremental

- Absolute encoders are used in systems that require precision and cannot afford re-calibration procedures
  - Robotic manipulators
  - Positioning systems
- Incremental encoders have a lower price point
- They are often used in applications that are insensitive to small amounts of inaccuracy, do not require calibration and are mostly used to measure linear motion
  - Drive system of a mobile robot
  - Some input devices

Resolver

- A resolver is another widely used sensor device to measure angular motion
- Based on electromagnetic induction
- The most common type is the brushless transmitter resolver
- The brushless transmitter resolver consists of:
  - A reference winding (rotor) (R)
  - Two secondary windings *SIN* (S1) and *COS* (S2) at 90° to each other



Resolver (cont.)

- The reference winding (R) is powered with an alternating voltage V_R using a rotary transformer
- The field of the reference winding induces voltages into the secondary windings:

$$V_{S1} = V_R \sin(\theta)$$
$$V_{S2} = V_R \cos(\theta)$$

- All signals (input and output) are of the same frequency
- For a static rotor angle θ the output signals are sine waves with constant amplitudes

Resolver (cont.)

- The resolver delivers data about the rotor angle θ through relative amplitudes of the output at the secondary windings:

$$\frac{V_{S1}}{V_{S2}} = \frac{\sin(\theta)}{\cos(\theta)} = \tan(\theta)$$

- At any given time the value of θ corresponds to the ratio of V_{S1}/V_{S2}, regardless of speed or acceleration
- With the above the rotor angle θ is given by:

$$\theta = \arctan2(V_{S1}, V_{S2})$$

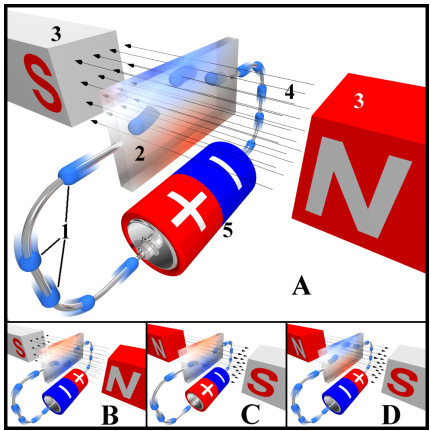
Comparison

Resolvers vs. Optical encoders

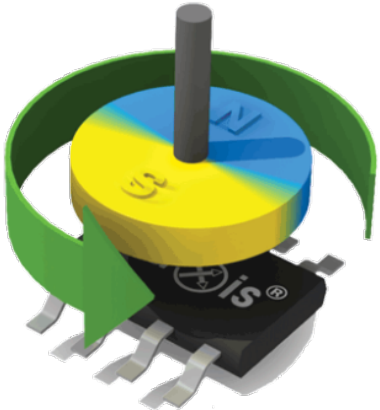
- Resolvers are particularly reliable under demanding conditions
  - The brushless type exhibits virtually no wear
  - The output signal does not drift
  - The effect of extreme temperature conditions is negligible
- Resolvers are more complicated to implement
  - AC power is needed
  - Axis is actively powered
- However, current resolvers and optical encoders are mostly equal on:
  - Resolution
  - Accuracy
  - Dynamic response

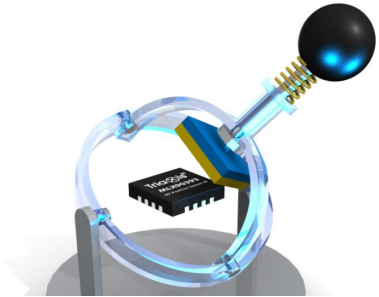
Hall Effect

- Lorentz force is acting on charges in a magnetic field
- This results in an voltage difference orthogonal to the current flow and the magnetic field
- This is called Hall effect / Hall voltage

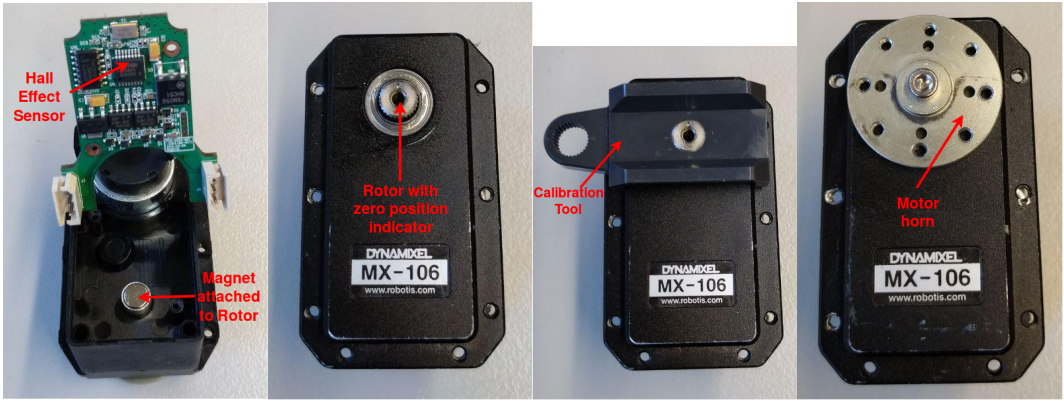


Hall Effect Sensor





Hall Effect Sensor



Jasper Güldenstein, Comparison of Measurement Systems for Kinematic Calibration of a Humanoid Robot

Hall Effect Sensor

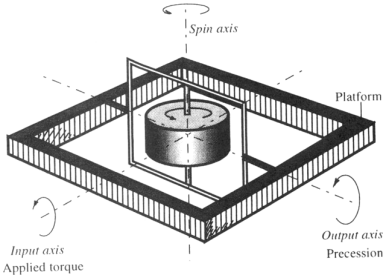
- Smaller than the other solutions
- Comparably cheap
- No AC current needed
- Can be influenced by strong magnetic fields
- Most commonly used in modern robots

Gyroscope

- A gyroscope is a “direction keeper”
- Most commonly used sensor in navigation
- Used in outer space applications
- Categories:
  - Mechanical gyroscope
  - Semiconductor (MEMS) gyroscope
  - ...

Mechanical Gyroscope

- Solid disc rotating around an axis
- Rotation axis (spin axis) is located in a frame
- This frame can rotate around one (or two) axes



Mechanical Gyroscope (cont.)

Two useful properties:

1. Spin axis of a free gyroscope stays fixed in relation to a global coordinate system
2. A gyroscope will deliver an output signal (torque) that is proportional to the angular velocity about an axis perpendicular to the spin axis

The second property is a phenomenon called precession

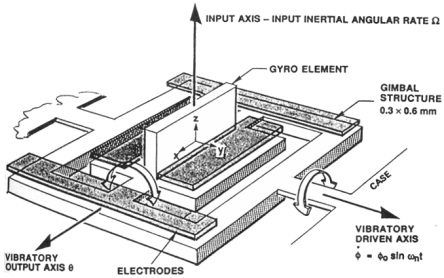
- "Precession is always in such a direction as to align the direction of rotation of the wheel with the direction of rotation of the applied torque"

Mechanical Gyroscope

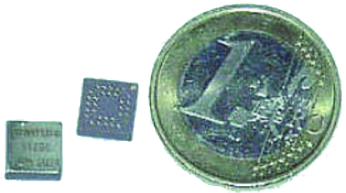
Video
https://www.youtube.com/watch?v=xQb-N486mA4

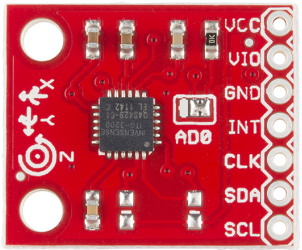
Semiconductor Gyroscope

- Micro-Electro-Mechanical System (MEMS) in silicone
- Manufactures using surface or bulk micromechanic processes
- Various implementations exist



Semiconductor Gyroscope (cont.)





Semiconductor Gyroscope (cont.)

Video
https://www.youtube.com/watch?v=eqZgxR6eRjo (1:30 - 1:47)

Accelerometer

- Relies on displacement of *inertial mass* w.r.t. framing
- Measures *proper* acceleration in one dimension
- This includes gravity as 9.81m/s^2 pointing up

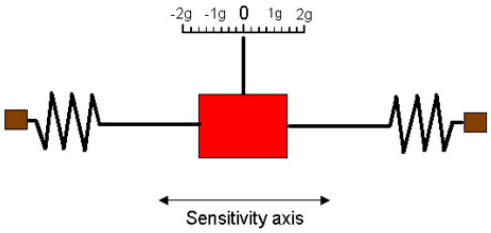
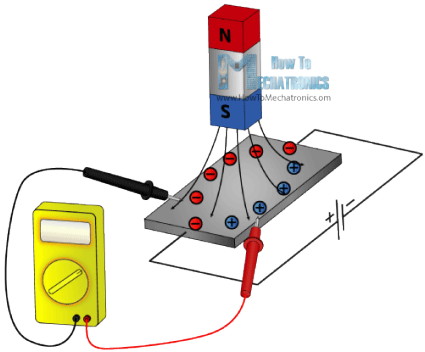


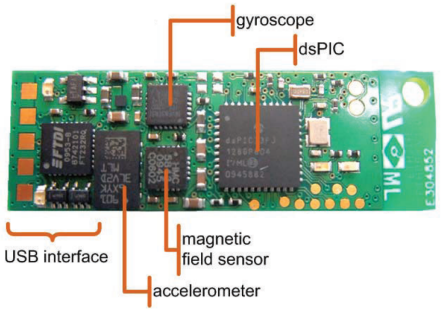
Fig. 1. Schematic structure of an accelerometer

Magnetometer

- Compass
- Measures orientation in magnetic field
- Most sensors are based on measuring the Hall effect
- Application in robotics is difficult due to electromagnetic fields



Inertial Measurement Unit (IMU)



- In practice gyroscopes, accelerometers, and magnetometers are often combined in one device
- This yields a good estimate of the orientation of the device

IMU Application

"Cubli" Video
https://www.youtube.com/watch?v=n_6p-1J551Y

Further Motion Measuring Approaches

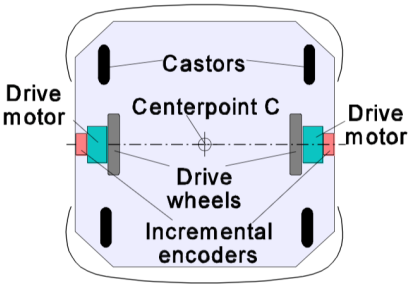
Though the previously presented approaches are the most used ones, there are other possibilities

- Difference between two absolute positions (derivation is velocity)
  - Visual landmarks (e.g. April Tags)
  - Radio landmarks (e.g. GPS) (see lecture "Distance Sensing")
  - Using various distance sensing methods
  - Better to use in combination with Bayes Filter (see lecture "State Estimation")
- Doppler effect (not discussed in this lecture)

Outline

Encoder Applications

- Most common use case: Combination with motors
- Used to measure:
  - Absolute/relative angle
  - Direction of the rotation
  - Angular/linear velocity
- Knowledge about connected transmission and wheels allows to determine the distance traveled



Localization of Mobile Robots

- In most cases, the motors used in mobile robotic systems are equipped with incremental encoders
- Using knowledge about the transmission and the wheel diameter and circumference, the movements of the moving robot can be determined
- A global coordinate frame must be referenced for localization

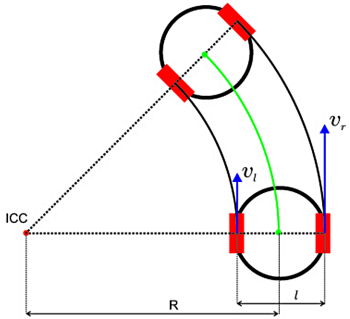
- This basic procedure for the localization of mobile robots is called dead reckoning
- The relative position and orientation of the mobile robot is determined using the history of accumulated measurement values from the incremental encoders
- Open loop localization

Dead Reckoning

- The simplest case of dead reckoning for mobile robots can be set up using a differential drive
- On a differential drive, the two wheels of a robot are located on a shared axis
- Wheel speeds can be controlled and adjusted independently
- The center of the robot is located in the middle of the link between the two wheels
- If wheel speeds are equal, the robot moves forward or backward
- If wheel speeds differ, the robot moves along a circular path
- Cars work in a different (more complicated) way

Dead Reckoning (cont.)

- The center of the circular path which the robot moves along is necessarily a point on the shared axis of the wheels
- This point is called the instantaneous center of curvature (ICC)
- Variation of the wheel speeds changes the location of the ICC

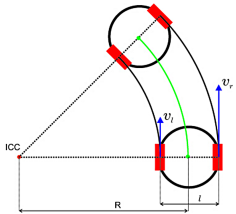


Dead Reckoning (cont.)

- Let ω be the angular velocity of the rotation of the robot around the instantaneous center of curvature
- Let ℓ be the distance (baseline) between the two wheels
- Let R be the distance between the center of the robot and the ICC

The velocities of the wheels (v_l and v_r) are given by:

$$\begin{aligned} v_l &= \omega \cdot (R - \ell/2) \\ v_r &= \omega \cdot (R + \ell/2) \end{aligned}$$



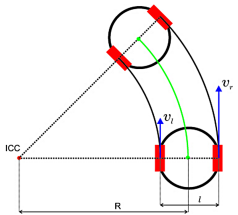
Dead Reckoning (cont.)

- ω, R, v_l and v_r are time-dependent terms

At each point in time, ω and R can be calculated as follows:

$$\omega(t) = \frac{v_r(t) - v_l(t)}{\ell}$$

$$R(t) = \frac{l}{2} \cdot \frac{v_r(t) + v_l(t)}{v_r(t) - v_l(t)}$$



Dead Reckoning (cont.)

If $v_l(t) = v_r(t)$:

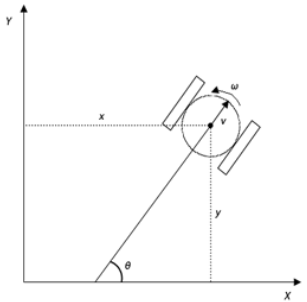
- Equation for the radius is not solvable
- Denominator equals zero
- Radius is effectively infinite
- Robot drives straight ahead

If $v_l(t) = -v_r(t)$:

- Numerator of the equation for the radius becomes zero
- The robot is turning on the spot

Forward Kinematics

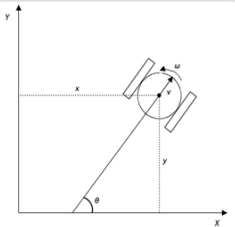
- While driving, the robot changes its position (x, y) and orientation (θ) in reference to a global or world coordinate system
- The triple (x, y, θ) representing position and orientation is called the pose of the robot
- θ is measured in relation to the x-axis of the global coordinate system

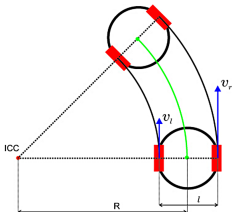


Forward Kinematics (cont.)

- The calculation of the pose which is achieved at given wheel velocities $v_l(t)$ and $v_r(t)$ is called forward kinematics
- In this context the ICC is calculated as follows:

$$ICC = \begin{pmatrix} x - R \cdot \sin(\theta) \\ y + R \cdot \cos(\theta) \end{pmatrix}$$





Forward Kinematics (cont.)

Knowing the ICC, the subsequent pose (x', y', θ') of the robot can be determined at the time of $t = t_0 + \delta t$

- If $v_r(t)$ and $v_l(t)$ remain constant

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega \cdot \delta t) & -\sin(\omega \cdot \delta t) & 0 \\ \sin(\omega \cdot \delta t) & \cos(\omega \cdot \delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega \cdot \delta t \end{bmatrix}$$

- Through integration the *pose* of the robot can be determined for any point in time t starting from (x_0, y_0, θ_0) at $t = 0$
- Wheel velocities $v_l(t)$ and $v_r(t)$ must be known

Forward Kinematics (cont.)

- Use of incremental encoders allows for a simple calculation of wheel velocities v_l and v_r at any given time
- Carried out periodically (δt), integration turns into accumulation
- It is assumed that the speeds remain constant during δt
- **General issue:** Accumulation of measurement errors!

Odometry

- The process of calculating the pose of a robot based on knowledge about its own actions/motions is called odometry
- Errors in orientation exhibit a strong impact of the deviation of the estimated pose from the real one
- Nevertheless, odometry is used in all established mobile robot systems:
  - Odometry is combined with absolute pose measurements
  - Using landmarks for absolute pose determination, a precise odometry may help reducing the number of landmarks needed
  - Sometimes odometry is the only available source of data

Odometry Deviation

Video
https://www.youtube.com/watch?v=ucTT8xNSYWQ

Odometry Deviation

Systematic errors caused by:

- Varying wheel diameters
- Actual baseline differs from expected distance
- Wheels are not on the same axis

- Finite resolution of the encoders
- Finite sampling rate of the encoders

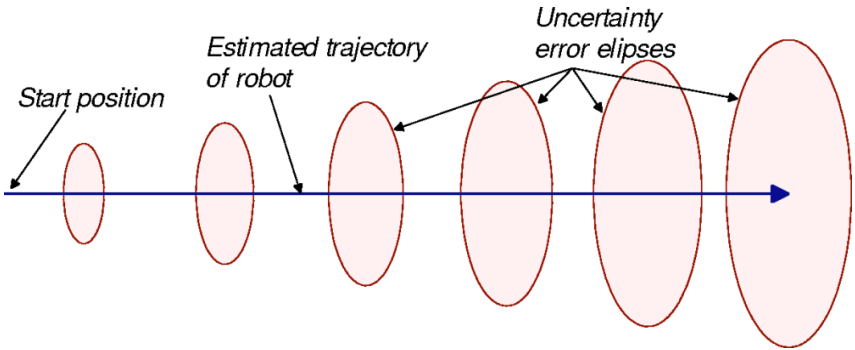
- Varying floor friction

Odometry Deviation (cont.)

Random errors caused by:

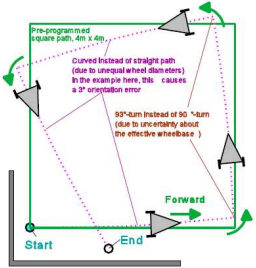
- Uneven ground
- Unexpected objects on the ground
- Spinning wheels
  - Slippery ground
  - Excessive acceleration
  - Skidding (fast turning)
  - Internal/external forces
  - No contact to the ground

Odometry Deviation (cont.)



Odometry Calibration

- Systematic errors can be reduced by calibration
- Random errors cannot be solved by calibration
- Different calibration procedures are possible



David M. Bradley, Odometry: Calibration and Error Modeling

Multi-Sensory Odometry

Odometry can improve through multiple data sources:

- Wheel-based odometry provides superior linear estimates
- IMU (gyroscope) provides superior orientation estimates
- Legged odometry provides equal linear and orientation estimates
  - Quality much lower for running
- Camera-based Visual Flow provides good odometry in structured environments

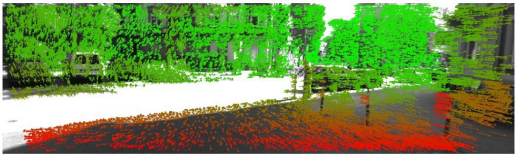
"Gyrodometry"

- Compute linear part by wheel-based odometry,
  use IMU reading for orientation
- Better: Integrate multiple readings through Kalmanfilter!

Visual Odometry

- Use of a (mono/stereo/RGB-D) cameras as motion sensor
- Take difference between two sequential images
- Compute movement that caused this difference
- Different approaches exist
  - Classical feature extraction
  - Learned neural networks
  - ...
- Accuracy bound by used image resolution
- Used resolution often bound by hardware, since visual odometry is expensive to compute

Visual Odometry



(a) Feature matching (2 frames, moving camera)

(b) Feature tracking (5 frames, static camera)

Geiger et al., "Stereoscan: Dense 3d reconstruction in real-time"

Visual Odometry

Video
https://www.youtube.com/watch?v=homos4vd_Zs

Visual Odometry

Different error sources are possible:

- Lighting conditions
- Feature less environment
- Repeating features
- Motion blur
- Rolling shutter effect
- Large parts of the visible environment move in relation to the robot (e.g. when there is a bus in the image)

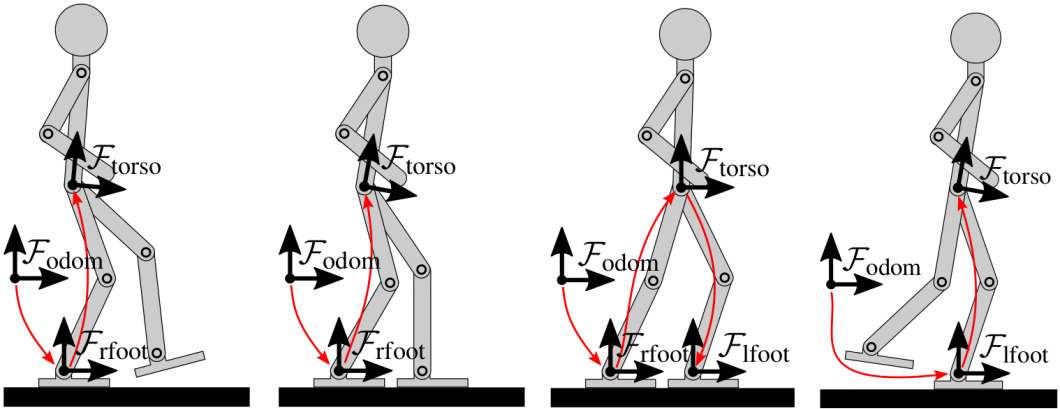
Lidar Odometry

- Similar to visual odometry
- Take “picture” with a laser distance sensor (see lecture “Distance Sensing”)
- Compute difference and get motion
- Features are not visual but structural
- Less prone to light problems

Walking Odometry

- On robots with legs, we cannot directly measure the velocity at the motors
- We need to compute the transformation for each step and sum them up over time
- To compute the transformation of a step, we do forward kinematics through the legs
- We always have one frame on our support foot and a transformation from there to the torso

Walking Odometry



https://www.hrl.uni-bonn.de/teaching/ss19/lecture-humanoid-robotics/slides/hr07_particlefilter.pdf

Walking Odometry

Different error sources are possible:

- Angle of a joint is not correct
- Link length not correctly modeled
- Backlash in joints (depends on the servo)
- Due to multiple joints and links, we have often multiple error sources each step
- Small angular errors (joints), lead quickly to large absolute errors (step position)
- Slippage
- Uneven ground
- ...

Walking Odometry

Video
https://www.youtube.com/watch?v=9HT33KMtfLw

Drone Odometry

- Drones move in all 6 dimensions
- Odometry is therefore a bit more complicated to compute
- Using the rotator speeds to compute odometry is theoretically possible
  - rarely used
  - wind
- Mostly “visual inertial odometry” is used, a combination of visual odometry and an IMU

Outline

Today's Agenda

- recall some physics: force, torque, stiffness
- pure position control vs. manipulation

- strain-gauges
- six-axis force/torque sensor

- tactile sensors
- advanced sensors and robot skin
- application example

Force

Application of force to a point of an object accelerates the object in the direction of the applied force

Newton's second law defines acceleration (a) of an object as proportional to the applied force (F) and inversely proportional to the mass (m) of the object, hence:

$$a = \frac{F}{m}$$

- The unit of force is the Newton (N):
- One Newton of applied force accelerates an object with a mass of 1 kg with 1 m/s^2

Torque

When a force F is applied to a rigid object and the axis of rotation is known (e.g. a robot joint axis), the corresponding torque is given by the vector cross product

$$\tau = r \times F$$

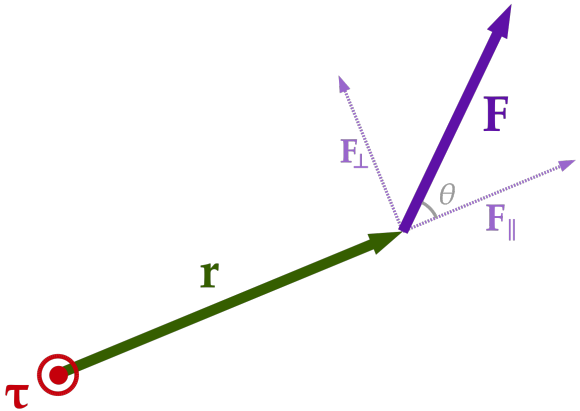
where r is the vector from the axis to the point of action of the force.

- The unit of torque is the Newtonmeter (Nm)

The angular acceleration α of an object is proportional to the torque τ and inversely proportional to the moment of inertia I of the object, hence:

$$\tau = I \cdot \alpha \qquad \Longleftrightarrow \qquad F = m \cdot a$$

Torque



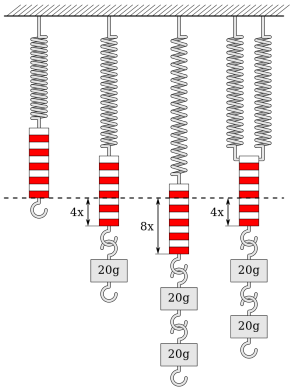
Measuring Forces

Force cannot be measured directly, only by its effects:

- acceleration of a rigid object
- deformation of an elastic object

Hooke's Law:

- the force F needed to extend or compress a spring by some distance x is proportional to that distance,
- $F = k \cdot x$ with stiffness k
- most solid materials obey this law when the elongation x is small
- calculate F from known stiffness k and observed x



https://de.wikipedia.org/wiki/Datei:Hookesches_Gesetz.svg

Manipulation Tasks and Position Control

Typical industrial manipulator (example PA-10 robot):

- 10 kg payload, 1 m reach, lower arm weight ≈ 10 kg
- position accuracy better than 0.2 mm (total load 10+10 kg)
- calculate corresponding stiffness: $k = F/x \geq 9.81 \cdot 10^5\ \mathrm{Nm}^{-1}$

accidentally moving into an obstacle ($k = 10^6$):

- 0.2 mm: contact force $\rightarrow$ 200 N ($\sim$20 kg)
- 1.0 mm: contact force $\rightarrow$ 1000 N
- 1.0 cm: contact force $\rightarrow$ 10000 N

- even minor position errors can be catastrophic
- need sensors to measure the interaction forces
- need fast control to limit contact forces

Manipulation Tasks and Position Control (cont.)

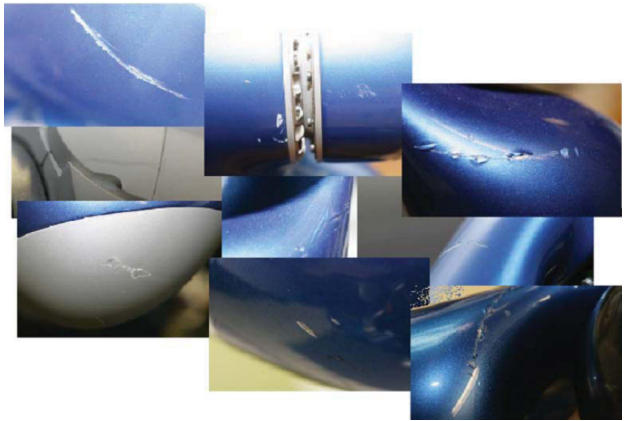
Moving a stiff robot into an obstacle...

- no force readings until the robot touches the object
- need to stop the robot very quickly once we hit the object

- motion with constant acceleration a: $v = a \cdot t$, $d = \frac{1}{2}at^2$
- most robots use a fixed control cycle, e.g. 100 Hz (PA-10)
- want to brake in one control cycle, $t = 0.01\,\text{s}$, so $a = 2d/t^2$
- assuming maximum interpenetration distance $d = 0.2\,\text{mm}$

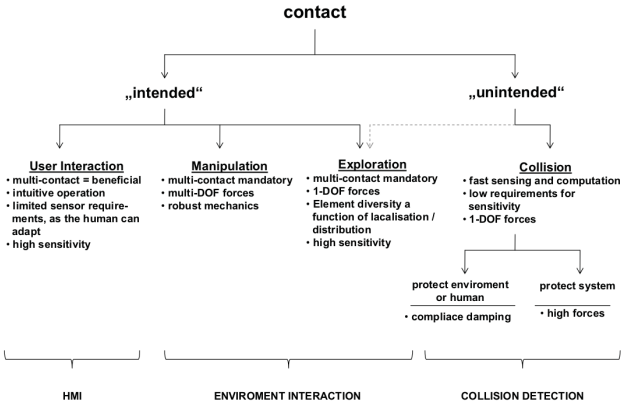
- maximum allowed velocity: $v = 2d/t \leq 0.04 m/s$
- the motion has to be very slow!
- (or braking accelerations become really large)

Problem with Stiff Robots



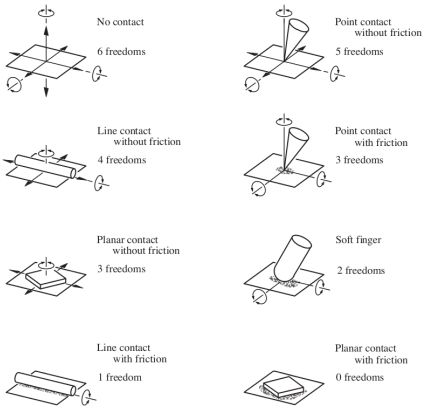
Results of unintended contacts, Strohmayr, Dissertation, 2012

Functional Taxonomy of Contact Types



Strohmayr, Dissertation, 2012

Geometric Taxonomy of Contact Types



Salisbury, PhD thesis, 1982

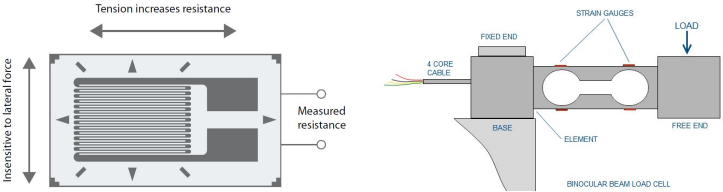
Outline

Strain Gauge

Most modern sensors use the approach of measuring the deformation of an elastic material

- a strain gauge is composed of conducting paths laminated onto an elastic carrier material
- mechanical strain affects the path resistance
- the resistance change is very small $\rightarrow$ special measurement circuitry is required
- Specific application of strain gauges allows to measure:
  - the magnitude and direction of mechanical strain
  - force and pressure
  - associated quantities like acceleration, distance, etc.

Strain Gauge and Load Cell



- resistance of a wire increases with length
- put thin long wire on carrier material

Foil Strain Gauge

Typical characteristics of foil strain gauges

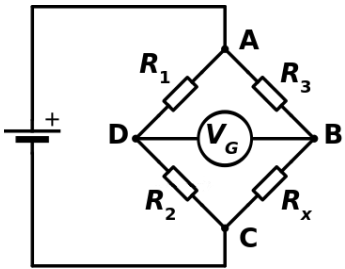
- resistance values: 120 - 600 Ω
- tolerance of the resistance usually less than $\pm 0,5\,\%$
- operating voltages: $1\,V\ -\ 10\,V$
- length variation of the strain gauge up to $\pm 3\,\%$
- typical length variation: $0.1\ -\ 10\,\mu m$
- achievable accuracy at 20° C $\approx$ 1 % - 5 %

Wheatstone Bridge

How to measure the resistance change of a strain-gauge?

- measure voltage drop over R
- length change small, $\Delta L \ll L$
- $\Delta R \approx 10^{-3}\,\Omega$, $\Delta R/R \approx 10^{-5}$

- difficult to measure tiny changes of large voltage; easier to measure tiny change of near-zero voltage
- Wheatstone bridge uses four resistors,
- tuned so that $R_1/R_2 \approx R_3/R_x$
- useful voltage between D and B



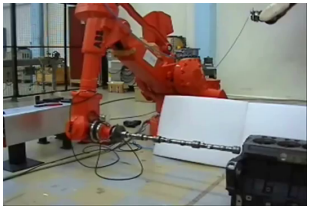
Outline

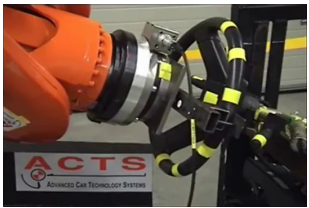
Six Axis Force/Torque Sensor

- common tool for industrial robots
- mounted between wrist and tool

- sensor measures total force, including weight of sensor and tool
- calibration step to estimate tool size, weight, COG
- subtract tool/sensor from total forces
- gives environment interaction forces
- use for robot control

- note: no force measurements for any contacts between robot base and wrist





ATi/ABB automation video, youtube

Six Axis Force/Torque Sensor

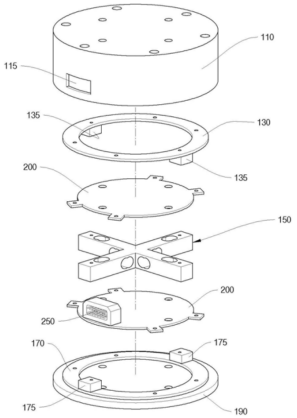
Video
https://www.youtube.com/watch?v=4Ro6rQbePqE
Video2
https://www.youtube.com/watch?v=b4nz_hAh7qs (0:55-1:50)

Six Axis F/T Sensor: Construction

- stiff upper housing (110) with screw threads (tool connection)
- connector ring from (110) to (150)
- elastic member (150) with strain-gauges
- stiff bottom plate (190)
- amplifier circuit boards (200)

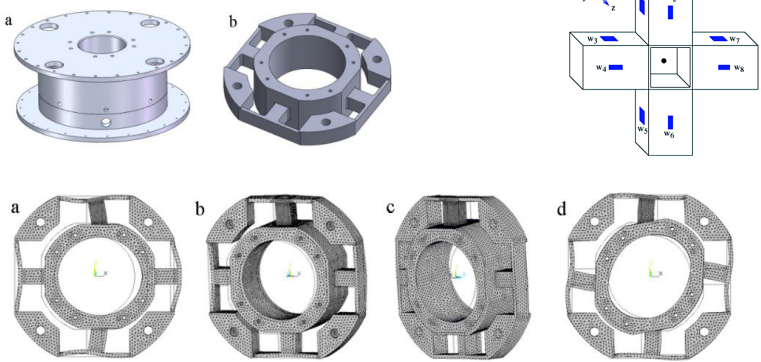
- two strain-gauges on each cross arm

- triangular setup with three elastic arms (at 120°) another popular arrangement



C.G. Kang, Int. Journal of Control, Automation, and Systems, vol. 3, no. 3, 469–476, 2005

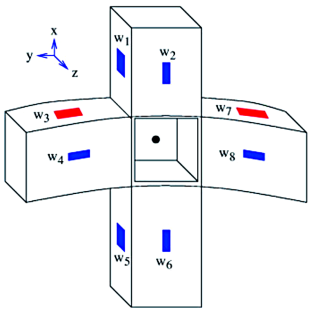
Six Axis F/T Sensor: Deformation Under Load



(a) $F_x = 10000\text{N}$, (b) $F_z = 12000\,\text{N}$, (c) $M_x = 2000\,\text{Nm}$, (d) $M_z = 3000\,\text{Nm}$

Six Axis F/T Sensor: Coupling Matrix

- deformation of the elastic part affects different strain gauges
- specific for each sensor design



F/T DOF	Strain gauges
F_x	W_3, W_7
F_y	W_1, W_5
F_z	W_2, W_4, W_6, W_8
M_x	W_4, W_8
M_y	W_2, W_6
M_z	W_1, W_3, W_5, W_7

Six Axis F/T Sensor: Coupling Matrix

- resistance change of stain-gauges is linear for small deformation
- therefore, the transformation from stain-gauge values to forces and torques can be written as matrix multiplication
- this is called the coupling matrix K

- coupling matrices are very device specific
- in ideal case, many coefficients zero
- datasheet usually includes detailed K values from factory calibration

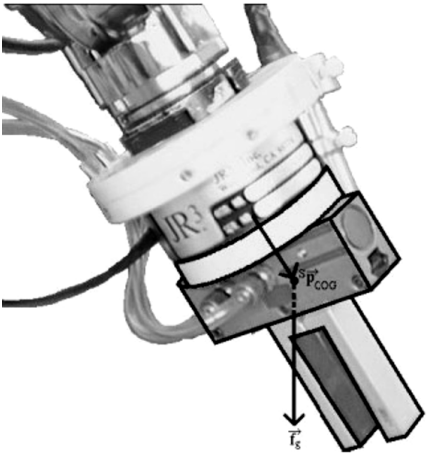
$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = K \cdot \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \\ W_5 \\ W_6 \\ W_7 \\ W_8 \end{bmatrix}$$

Six Axis F/T Sensor: Calibration

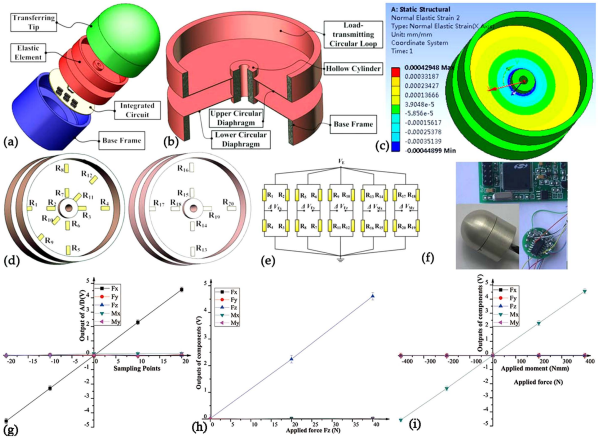
- mass of the attached tool $\vec{F}_{tool}$
- tool's center of gravity $\vec{p}_{COG}$
- also, mass and COG of the moving part (tool side) of the F/T sensor

- once tool COG and weight are known, software can subtract those values from the measured data
- this allows estimation of external forces

- calibration is only valid in the configuration it was carried out
- still temperature-dependent



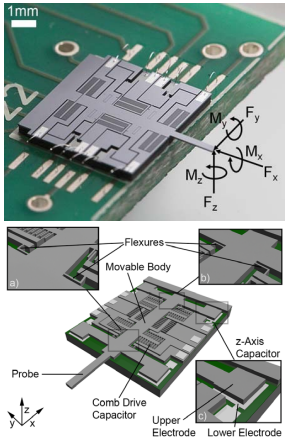
Miniaturization: Five-axis Fingertip F/T Sensor

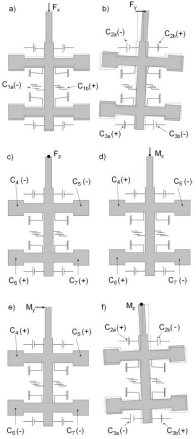


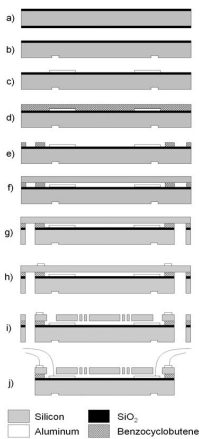
- measures F_x, F_y, F_z, M_x, M_y
- thin diaphragm as flexible element
- FEM analysis
- 20 strain-gauges
- five Wheatstone bridges

Q.-K. Liang et.al., Scientific Reports 6:24689 2016, DOI: 10.1038/srep24689

Miniaturization: MEMS F/T Sensor

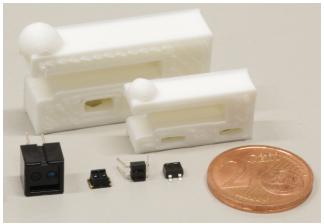


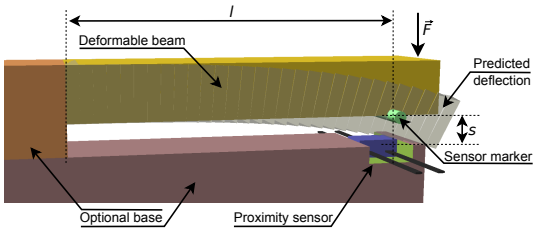




Beyeler et.al., Microfabricated 6-Axis Force-Torque Sensor, ICRA 2009, 520–525

3D Printed Alternative

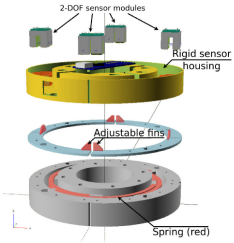


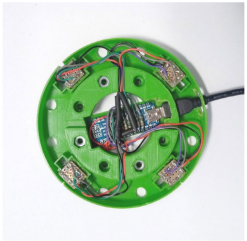


- 1 DoF
- Other materials and other sensors possible
- Measuring deformation with proximity sensor
- Easier integration into an object

Florens Wasserfall, Norman Hendrich, Fabian Fiedler, Jianwei Zhang, 3D-Printed Low-Cost Modular Force Sensors, 20th Intl. Conference on Climbing and Walking Robots (CLAWAR-2017)

3D Printed Alternative





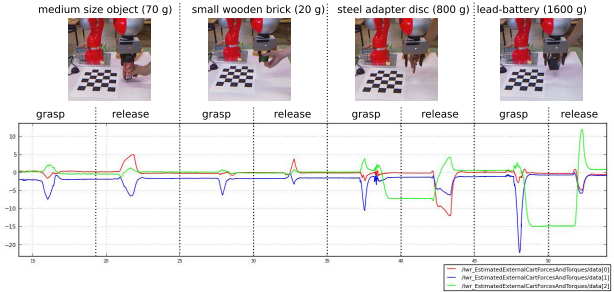
- 6 DoF
- Application specific design
- Significantly cheaper
- Sufficient precision for many use cases

Norman Hendrich, Florens Wasserfall, Jianwei Zhang, 3D-Printed Low-Cost Force-Torque Sensors, IEEE Access, 2020

F/T Sensor Applications

- guarded motions: check motion execution and stop when preset force thresholds are exceeded
- compliant motions
  - peg-in-hole insertion tasks
  - surface following and measurements
  - polishing and grinding tasks, constant normal forces
  - human-guided motion/trajectory teaching
- direct or hybrid force control
  - torque control of individual motors (joints)
  - force / torque control of tool center point
  - mixture of position and force control schemes
  - requires precise dynamics model of the robot
- note: forces behind the F/T sensor are not measured!

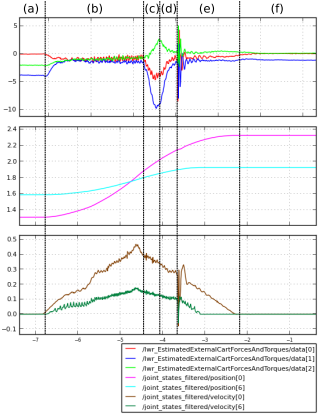
Application Example: Force-Guided Object Handover



- measured tool forces F_x, F_y, F_z over time, KuKA LWR4+
- grasped object weight plus human interaction forces
- gripper grasp and release triggered by interaction forces

Liebrecht, Bistry, Hendrich, Zhang, IROS-WS 2014

Application Example: In-Motion Object Handover

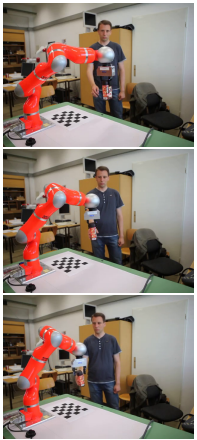


- top: estimated external forces F_x, F_y, F_z
- middle: robot joint angles ϕ_1, ϕ_6
- bottom: joint velocities ω_1, ω_6

- estimated forces quite noisy while the robot moves (despite good sensors and excellent LWR4+ robot model)
- reliable force threshold too high for user acceptance
- gripper tactile sensor used as additional sensor
- grasp release triggered by robot force sensing and gripper tactile sensor

Liebrecht, Bistry, Hendrich, Zhang, IROS-WS 2014

Application Example: In-Motion Object Handover



- estimated forces quite noisy while the robot moves (despite good sensors and excellent LWR4+ robot model)
- reliable force threshold too high for user acceptance
- gripper tactile sensor used as additional sensor
- grasp release triggered by robot force sensing and gripper tactile sensor

Liebrecht, Bistry, Hendrich, Zhang, IROS-WS 2014

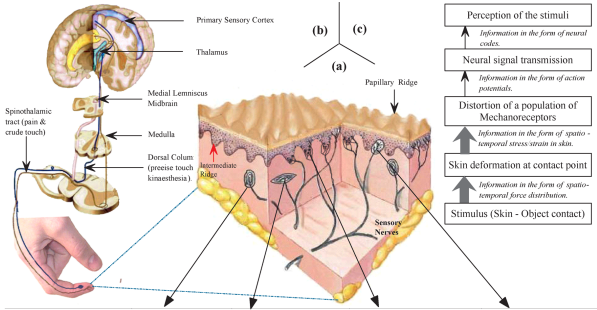
Outline

Human Tactile Sensing

- amazing grasping and manipulation capabilities of humans
- human hand tactile sensing provides a reference system

- three layers: subcutaneous (fatty) tissue, dermis, epidermis
- four different receptor types
  - FA-I: Meissner's corpuscles: vibration and motion detection
  - SA-I: Merkel's disks: sustained pressure, texture perception
  - FA-II: Pacinian corpuscles: skin deformation, vibration, tool use
  - SA:II: Ruffini endings: skin stretch, slip, tangential forces
  - fast and slow adaptation to stimulus
- cold/anaesthesized fingertips: dramatic loss of manipulation capabilities, especially for small objects

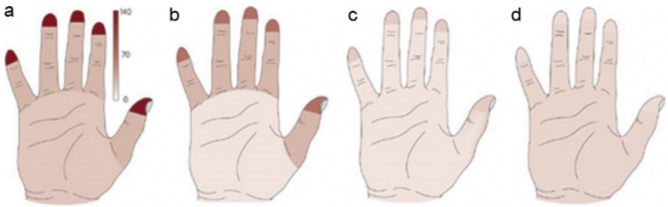
Human Tactile Sensing



Classification Basis		Pacinian Corpuscle	Ruffini Corpuscle	Merkel Cells	Meissner's Corpuscle
Type		FA II	SA II	SA I	FA I
Adaptation Rate		Fast	Slow	Slow	Fast
Spatial Acuity (mm)		10+	7+	0.5	3-4
Vibration/rapid indent: threshold	Best(μm)	0.01	40	8	2
	Mean(μm)	0.08	300	30	6
Stimuli Frequency (Hz)		40-500+	100-500+	0.4-3	3-40
Conduction Velocity (m/s)		35-70	35-70	40-65	35-70
Effective Stimuli		Temporal changes in the skin deformation	Sustained downward Pressure; Lateral skin stretch; Skin slip.	Spatial deformation; Sustained pressure; Curvature, edge, corners.	Temporal changes in skin deformation
Sensory Function		High frequency vibration detection; Tool use.	Finger position; Stable grasp; Tangential Force; Motion direction	Pattern/form detection; texture perception; Tactile flow perception.	Low frequency vibration & motion detection; Grip control; Tactile flow perception.

R.S. Dahiya et.al., Tactile Sensing From Humans to Humanoids, IEEE T-RO Vol.26,No.1, 2010

Human Tactile Sensing: Receptor Ddistribution



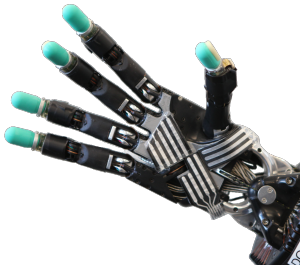
- density (afferents per cm^2) of receptor cells in the human hand
- (a) and (b): fast and slow adapting type I
- (c) and (d): fast and slow adapting type II

- the special role of the fingertips is obvious
- "two point resolution" at fingertip ca. 1.6 mm, palm 7.7 mm

Yousef et.al., Tactile sensing - A Review, 2011. doi:10.1016/j.sna.2011.02.038

Tactile Sensors: Requirements

- extremely application specific
- often very thin
- robot skin, palm and finger tip sensors
- other applications: medical measurements, shoe fitting, etc.



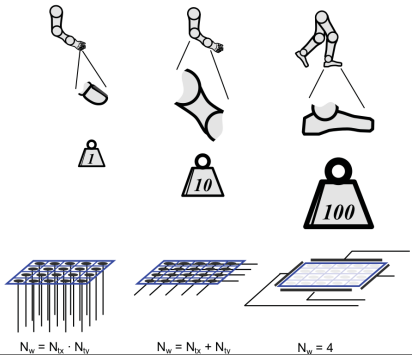
Typical requirements:

- spatial resolution ca. $1 \ldots 100\,\mathrm{mm}^2$
- force sensitivity in range $0.4 \ldots 10\,\mathrm{N}$
- minimum sampling frequency of $10 \ldots 1000$ Hz
- linear transfer function and low hysteresis
- low crosstalk between neighboring sensor cells (taxels)

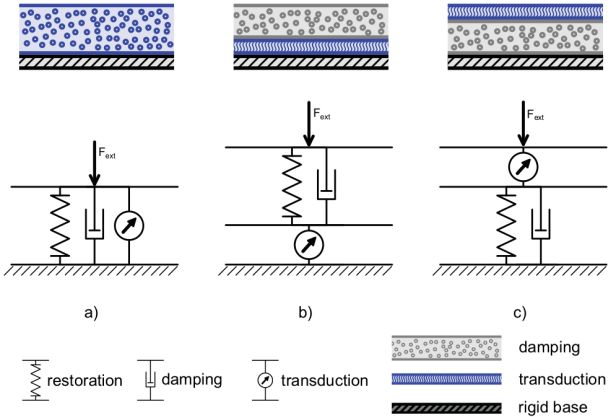
Tactile Sensors: Taxonomy and Classification

- sensing focus:
  - force
  - texture
- sensor technology:
  - switch sensors
  - resistive materials
  - capacitive sensors
  - MEMS (silicon) sensors
  - optical sensors
- sensor layout:
  - single sensors
  - 1D- and 2D matrix sensors
  - flat and curved shape
  - layer structure

- load scale:



Tactile Sensors: Layers



Strohmayr, Dissertation, 2012

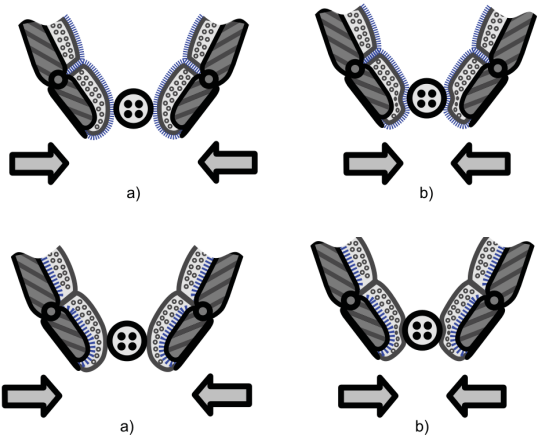
Tactile Skin: Layers

Sensing layer on top:

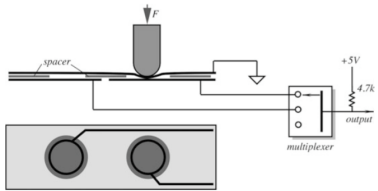
- high sensitivity
- good spatial resolution
- sensor exposed
- fragile, not robust

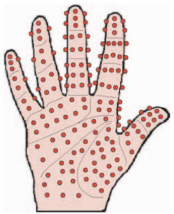
Sensing layer inside:

- force “smeared out”:
- lower sensitivity
- lower spatial resolution
- sensor protected
- robust



Switch-Type Sensor





- mechanical or membrane type switches
- binary on/off readings, no force measurement
- usually low spatial resolution, no shearing forces

K. Matsuo et.al., Placement of Tactile Sensors for Manipulation Task Recognition, ICRA 2008, 1641–1646

Force-Sensitive Resistors

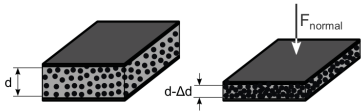
Materials whose electrical resistance is a function of strain

- a force-sensitive resistor (FSR) changes its resistance depending on the applied pressure
- use of conductive elastomeres or pressure-sensitive ink
- integration of the elastomer between two conductive plates

\+ very simple functional principle
\+ low manufacturing costs

– drift of resistance during prolonged pressure
– more useful for qualitative measurements

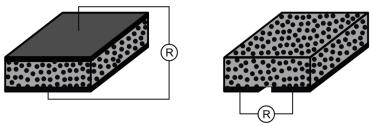
Force-Sensistive Resistor: Conductive Elastomers

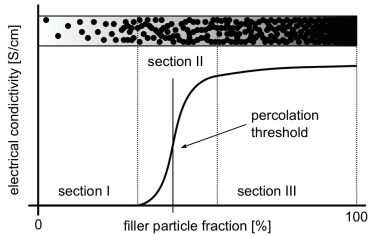


- foam with embedded conductive particles
- e.g. metal, coal
- low cost

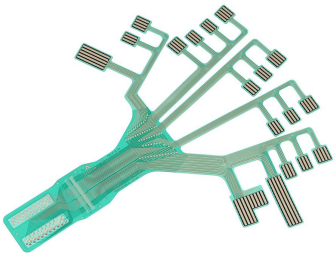
- top and bottom electrodes
- pair of bottom electodes

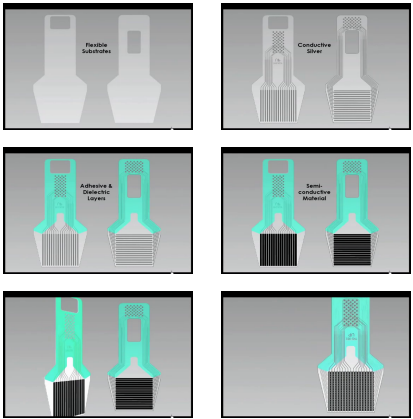
- nonlinear resistance curves





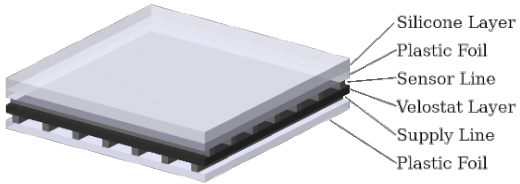
Force-Sensitive Resistor: Matrix-Type Sensor

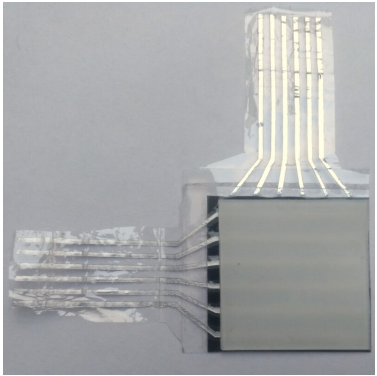




TekScan Inc., How the Tekscan Pressure Sensor is made, www.youtube.com/watch?v=q6_iZwuK3cU

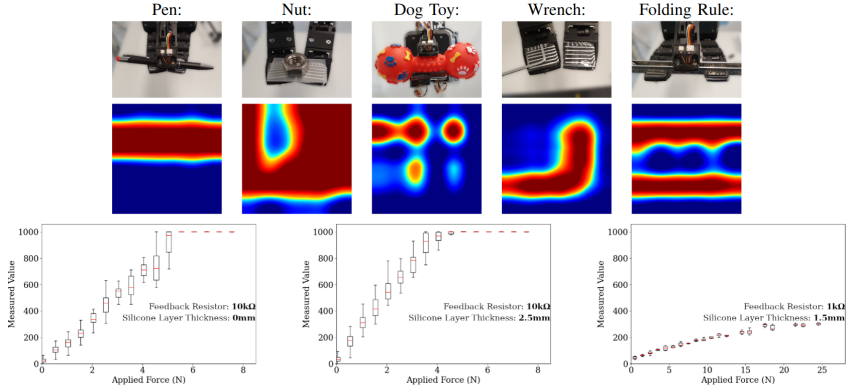
Force-Sensitive Resistor: Custom Sensors





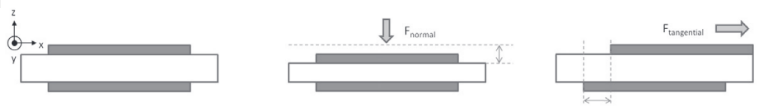
Niklas Fiedler, Philipp Ruppel, Yannick Jonetzko, Norman Hendrich, and Jianwei Zhang, A Low-Cost Modular System of Customizable, Versatile, and Flexible Tactile Sensor Arrays, IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS-2021)

Force-Sensitive Resistor: Custom Sensors (cont.)



Niklas Fiedler, Philipp Ruppel, Yannick Jonetzko, Norman Hendrich, and Jianwei Zhang, A Low-Cost Modular System of Customizable, Versatile, and Flexible Tactile Sensor Arrays, IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS-2021)

Capacitive Sensors

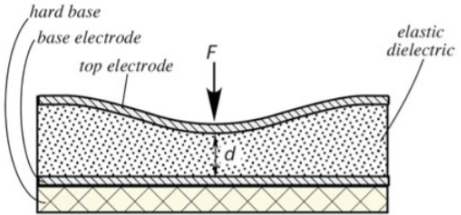


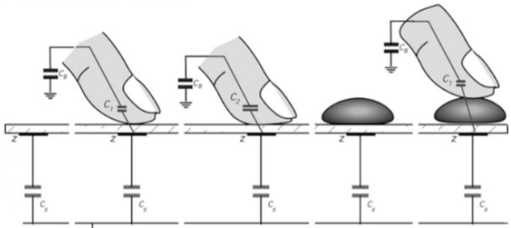
- capacitors made from flexible materials
- plate capacitor

$$C = \frac{dQ}{dV} = \epsilon \frac{A}{d}$$

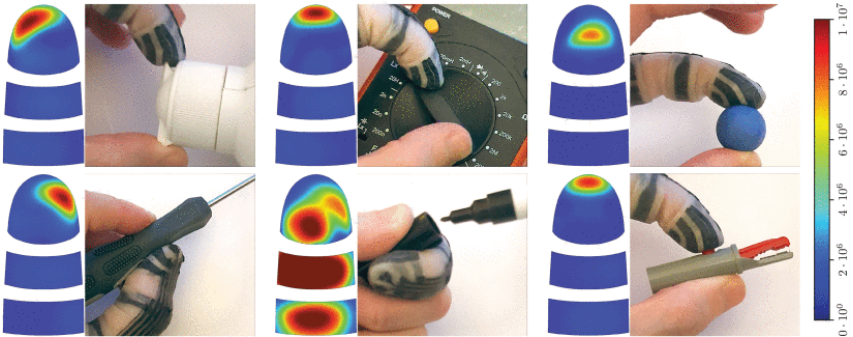
- an applied force either changes
  - the distance d between the plates (normal forces)
  - the surface area A (shearing forces)
- capacitance measured using frequency response of an oscillating R-C circuit
- also used for contactless proximity sensing (human tissue near the top electrode also changes capacitance)

Capacitive Sensors



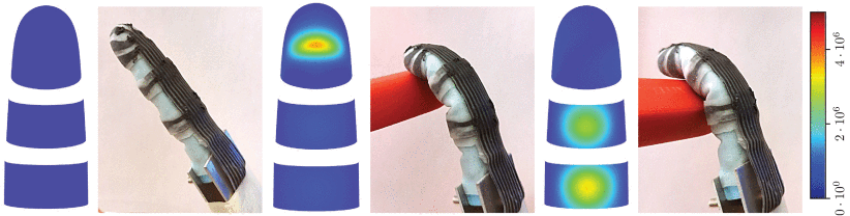


Capacitive Sensor Arrays



Philipp Ruppel, Norman Hendrich, and Jianwei Zhang, Elastic Tactile Sensor Skin on Double-Curved Surfaces for Robots and Wearables, IEEE Access, August 2022

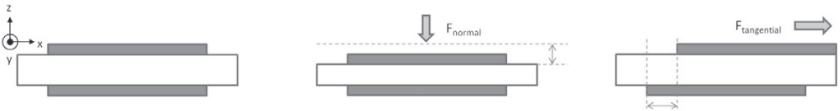
Capacitive Sensor Arrays



Philipp Ruppel, Norman Hendrich, and Jianwei Zhang, Elastic Tactile Sensor Skin on Double-Curved Surfaces for Robots and Wearables, IEEE Access, August 2022

Capacitive Sensor Arrays

Capacitive Sensors: Shearing Forces



- plate capacitor: $C = \frac{dQ}{dV} = \epsilon \frac{A}{d}$
- typically, $A \gg d$
- good sensitivity to normal forces
- but less sensitive to shearing forces
- use finger electrode layout to increase sensitivity to shearing forces
  - top-right electrode: z direction
  - top-left: $z + y$ direction
  - bottom-right: $z + x$ direction
  - decouple to get F_x, F_y, F_z

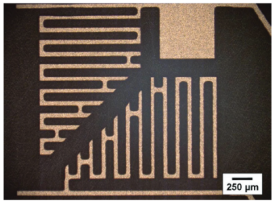
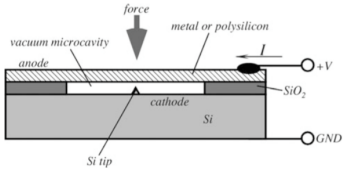


Fig. 2 Top view of one set of the sensor capacitor plates. Each of the capacitors is sensitive to applied forces in a specific direction (top right in the z-direction, top left in the z&y directions and bottom in the z&x-directions).

H. Yousef et al. / Procedia Engineering 25 (2011) 128 – 131

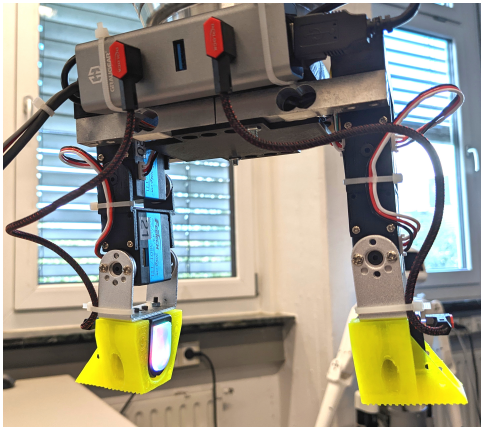
MEMS Sensors



- integrated (micro-) electronics and mechanical structures
- exploit existing microelectronics fabrication processes
- e.g. polysilicon structures on top of base silicon material
- typical structure has membranes or thin moving parts

\+ small sensor size, good spatial resolution
\+ often high sensitivity, high dynamic range
– fragile, sensors too small to cover large areas

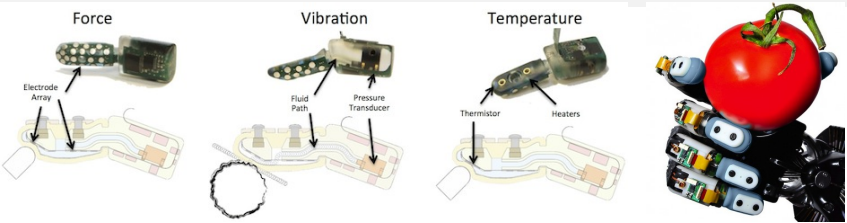
DIGIT Sensor



DIGIT Sensor

Lambeta, Mike, et al. "Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation." IEEE Robotics and Automation Letters 5.3 (2020)

Syntouch BioTAC Sensor



Bio-inspired robot fingertip sensor:

- combines pressure sensor, conductive liquid, thermistor
- static pressure relates to applied normal force
- pressure vibration for surface identification and slip detection
- electrodes provide spatial information, reconstruct force location and shearing forces
- temperature gradient for material identification

Syntouch LLC., 2012

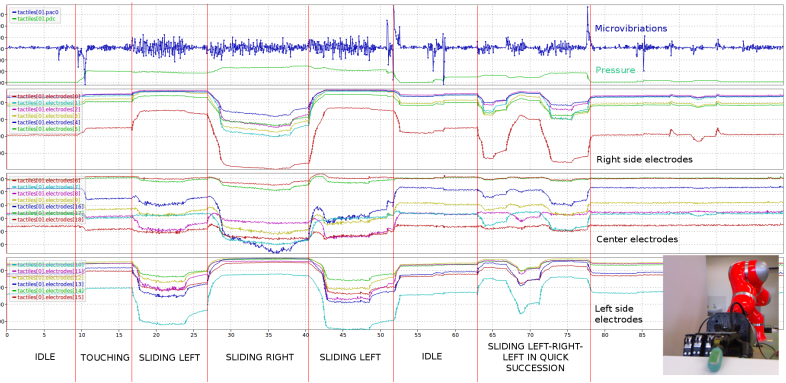
Syntouch BioTAC Sensor

Video

www.youtube.com/watch?v=W_0-u9PNUMU

BioTAC Sensor: Sliding Finger

normal and shearing forces, slippage detection



- sensor pre-processing pipeline: on-line calibration, low-pass filter, normal-force

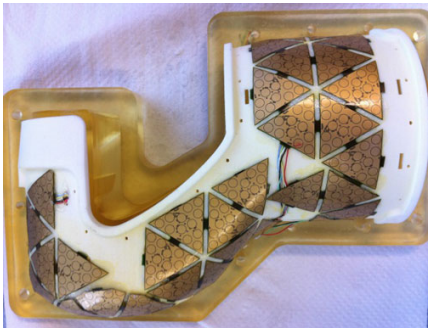
Robot Skin

The full body of a robot covered by tactile skin?!

- a lot of new problems of scale
- connecting all those sensors?
- calibration of 1000's sensors?
- mapping sensor positions?
- how to combine sensor readings?

EU RoboSKIN project (2010–2012):

- modular sensor modules
- tactile middleware to manage the sensors
- available for the iCub robot

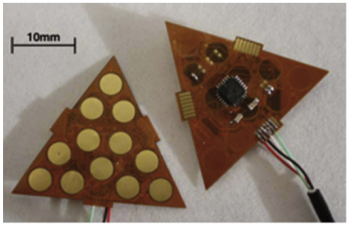


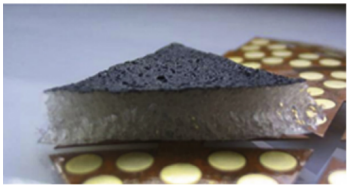
RoboSkin: Sensor Module

"triangle module":

- triangles can tile curved surfaces
- 12 taxels on each module
- integrated electronics, amplifier and communication
- three ports to neighboring modules
- supports automatic topology detection

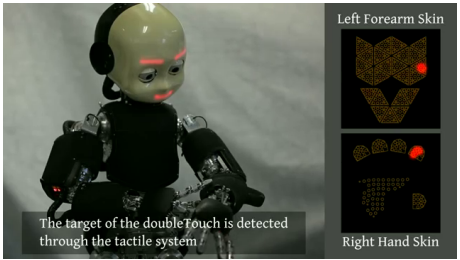
- silicone rubber foam that covers the sensors;
- conductive layer used as ground plane sprayed on top of the module





EU project FP7-RoboSKIN, 2011 (website roboskin.eu not active anymore)

iCub: Self-Calibration and Body Schema Learning



- skin consists of many sensor modules, different addresses, etc.
- arms and hands have curved and irregular surfaces
- manual calibration of all modules difficult (a lot of work)
- let the robot learn its kinematics and sensor layout
- robot touches itself, correlates arm/hand position and sensor response

Roncone et.al., Learning Peripersonal Space on the iCub, www.youtube.com/watch?v=pfse424t5mQ

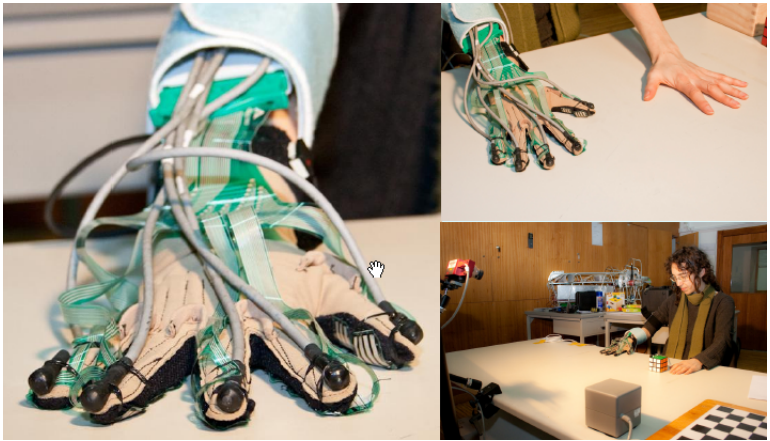
Application Example: Recording Human Manipulation

- learn grasping from human demonstration
  1. record human manipulation experiments
  2. annotate and classify the phases of the manipulation
  3. learn human strategies
  4. transfer to robot system

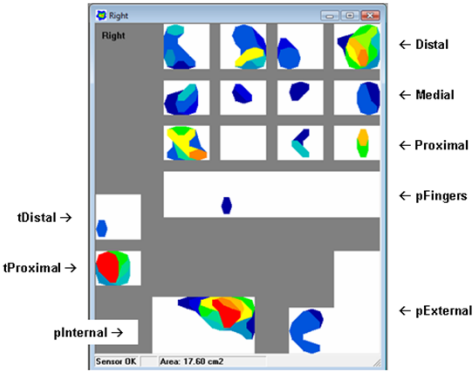
- complex multi-sensor system
  - Camera(s): video of the overall scene
  - Stereo-Camera: 3D-scene reconstruction, hand position
  - Cyberglove: hand shape
  - Polhemus: 3D fingertip positions (magnetic tracker)
  - TekScan Grip: fingertip and hand palm forces
  - Instrumented Rubik: forces on grasped object

EU project FP7-HANDLE, 2009-2013

Multisensor: Polhemus and Tekscan on Cyberglove

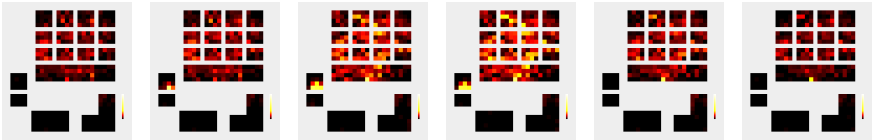


Tekscan Grip (on Cyberglove)



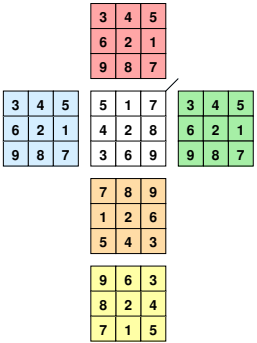
TekScan Inc., EU project FP7-HANDLE, 2012

Using a ball-point pen: clicking the pen



- heatmap of sensor TekScan grip activation
- back-view of the right hand: thumb, fingers, palm sensors
- pen is held in a power-grasp, thumb operates the button:
  - (a) idle state, grasp forces are low, distributed evenly
  - (b) thumb touches the button
  - (c-d) clicking, grasp forces increase to stabilize the pen
  - (e-f) idle state, grasp forces are low again.

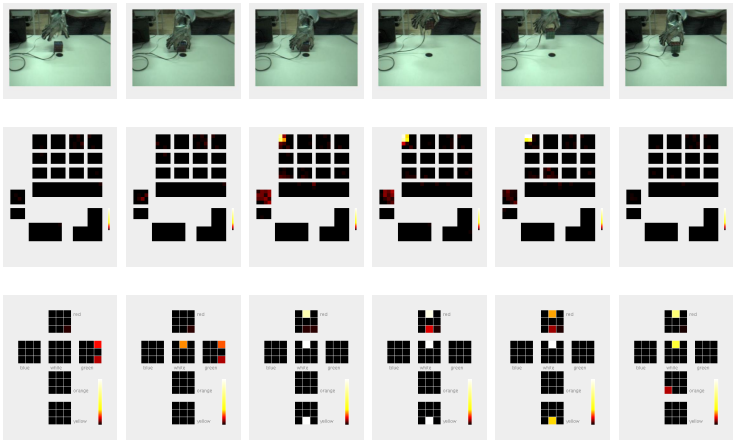
Force-Sensing Rubik cube



- record grasp forces during manipulation experiments
- $6 \times 3 \times 3$ FSR sensors, 6×3-axis accelerometers

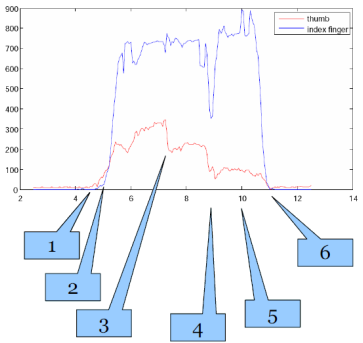
Shadow Robot Ltd., EU project FP7-HANDLE, 2011

Grasping the Rubik cube

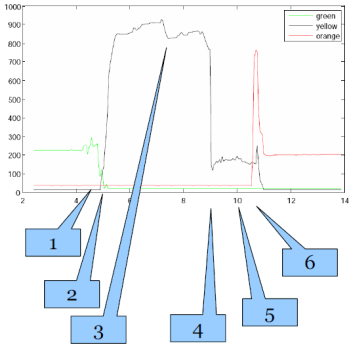


Grasping the Rubik-cube: segmentation

Tekscan (tDistal, fDistal)



Rubik (green, yellow, orange faces)



1 first contact 2 lift-off 3 holding 4 letting slip 5 holding 6 setting down

Tactile Gloves

Take home message

- pure robot position control is dangerous
- many robot tasks require force-sensing and -control
  - collision detection and human safety
  - manipulation and grasping
  - interaction in unknown environments
  - (physical) human-robot interaction
- forces measured by object deformation
- strain-gauges, elastic polymers, ...
- six-axis force/torque sensor
- tactile sensors and robot skin
- for self-calibration, multi sensor fusion, complex manipulation
- highly application specific

References

- R.S. Dahiya et.al., Tactile Sensing From Humans to Humanoids, IEEE Transactions on Robotics, Vol.26,No.1, 2010
- M. Cutkosky, Force and Tactile Sensing, in: Bruno Siciliano, Ed., Handbook of Robotics, Springer, 2011.
- Hanna Yousef, Mehdi Boukallel, Kaspar Althoefer, Tactile sensing for dexterous in-hand manipulation in robots — A review, Sensors and Actuators A: Physical, 2011. doi:10.1016/j.sna.2011.02.038
- Michael Strohmayr, Artificial Skin in Robotics, Dissertation, Karlsruhe Institute of Technology, 2012.
- several papers (see slides)

Outline

Table of Contents

Measurement of Distance

The ability to measure distance plays a crucially important role in the field of robotics. It is particularly important for mobile robots.

Measurement of Distance

The ability to measure distance plays a crucially important role in the field of robotics. It is particularly important for mobile robots.

- Obstacle detection/avoidance
- Localization
- ...

Measurement of Distance

The ability to measure distance plays a crucially important role in the field of robotics. It is particularly important for mobile robots.

- Obstacle detection/avoidance
- Localization
- ...

Several sensors can be used to determine distances:

Measurement of Distance

The ability to measure distance plays a crucially important role in the field of robotics. It is particularly important for mobile robots.

- Obstacle detection/avoidance
- Localization
- ...

Several sensors can be used to determine distances:

- Infrared/Ultrasonic sensor
- Laser rangefinder
- Camera-based
- ...

Measurement of Distance

The predominant underlying physical principles for measurement of distance using sensor devices are:

- Time of flight (TOF)
  → Time required for a signal to travel through a medium
- Phase shift/difference
  → Difference in phase as a property of the reflected signal
- Triangulation
  → The geometric approach
- Intensity
  → Signal loss over distance

Time of Flight

Measurement of distance using the *time of flight* principle is a straightforward process

- Emit signal (impulse)
- Measure time (Δt) until reception of the echo/reflection
- Determine distance (D) using knowledge about the speed (v) of the signal

$$D = \frac{\Delta t \cdot v}{2}$$

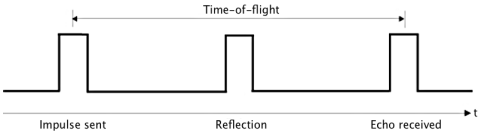
Time of Flight

Example: Distance measurement using light impulses

- *Signal*: Light impulse
- *Medium*: Air
- *Measured time*: 65 ns

Assuming $v = c$, where c is the speed of light (299792458 m/s) and a value of 65 ns for Δt the resulting distance is

$$9.74\,m \approx \frac{0.000000065\,s \cdot 299792458\,m/s}{2}$$



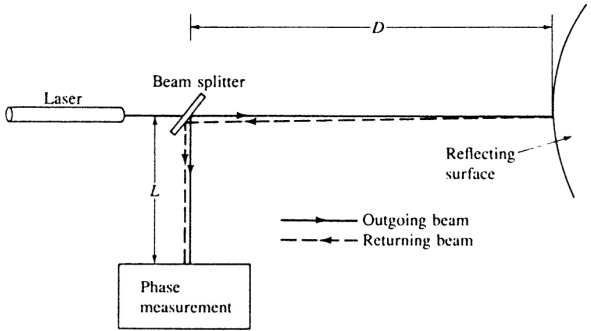
Phase Shift/Difference

As an alternative to *time of flight*, the phase shift approach is also very straightforward

- Emit signal with wave length λ (e. g. modulated light)
- Measure phase difference between received echo and signal
- Determine distance D based on the phase shift $\Delta\theta$ between the reflected signal and the emitted signal
- For light modulated with frequency f_{mod}: $\lambda = \frac{c}{f_{mod}}$

$$D = \frac{1}{2} \cdot \frac{\Delta\theta}{2\pi} \cdot \lambda$$

Phase Shift


D
Beam splitter
Laser
Reflecting surface
L
Outgoing beam
Returning beam
Phase measurement

Phase Shift

Example: Distance measurement using light impulses

- *Signal*: Light impulse
- *Medium*: Air
- *Frequency*: 10 MHz
- *Measured phase shift*: 4.78rad

Assuming c the speed of light (299792458 m/s) the resulting distance is

$$11.45\,m \approx \frac{1}{2} \cdot \frac{4.78}{2\pi} \cdot \frac{299792458\,m/s}{10000000\,s^{-1}}$$

Phase Shift

Caution:

- Impossible to distinguish between all $D = n \cdot \lambda, n \in \mathbb{N}$
- To receive a distinct result, constraints $\Delta\theta < 360°$ and $2D < \lambda$ need to apply
- A modulation frequency of $f_{mod} = 10$ MHz results in a wavelength of about 30 m

Triangulation

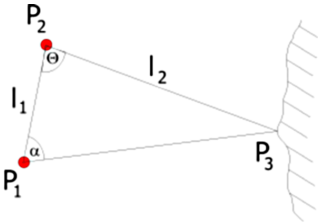
Triangulation is the process of calculation of distance to a point using knowledge about viewing angles

- Use two viewing points with the distance between them (baseline) known
- Align both "viewers" looking towards the point in question
- Determine the angles of both "viewers" to the baseline
- Calculate the distance using basic trigonometry

Two viewing points may be obtained in a number of ways:

- Movement of a single sensor
- Special design of the sensor
- Multiple sensors

Triangulation



$$D = l_2 = \frac{l_1 \sin(\alpha)}{\sin(\alpha + \Theta)}$$

Triangulation

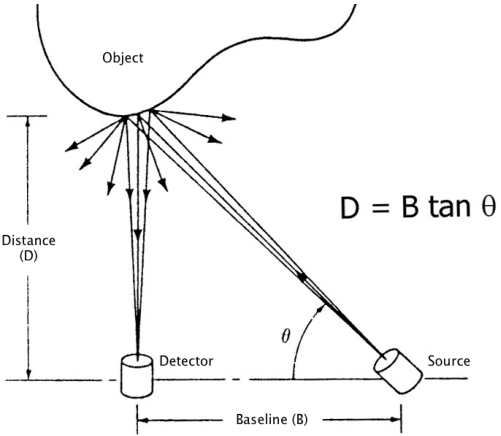
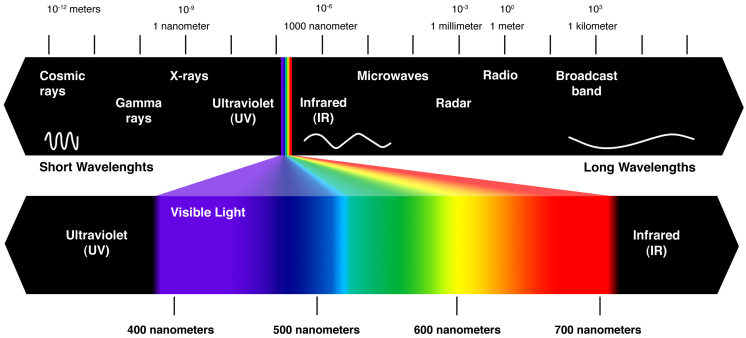


Table of Contents

Infrared Sensors

- Infrared sensors are the most simple type of non-contact sensors
- Emitting a signal in the infrared spectrum

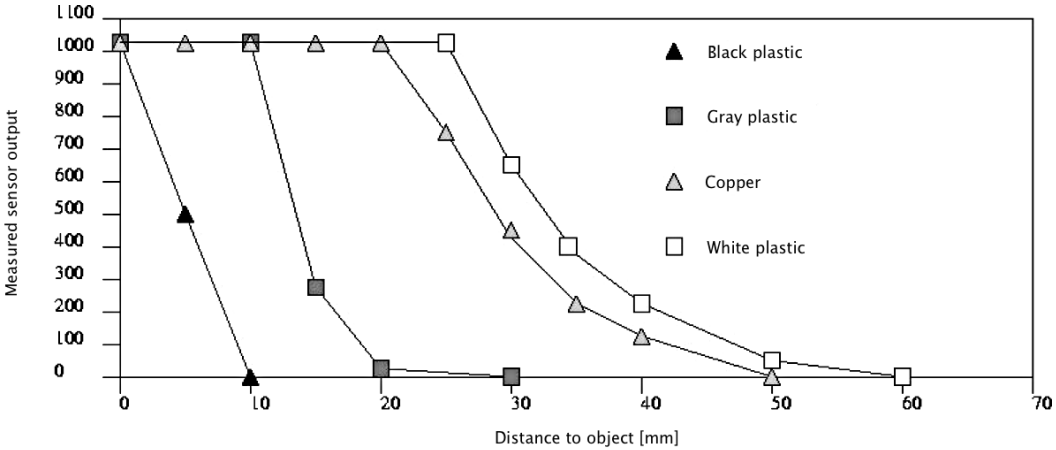


Infrared Sensors

Reflection of the emitted signal by objects in the vicinity:

- The intensity of the reflected light is inversely proportional to the squared distance
- To be able to distinguish the emitted signal from other infrared sources in the vicinity (e.g. fluorescent lamps or sunlight), it is usually modulated with a low frequency (e.g. 100 Hz)
- Assuming that all objects are equal in color and surface, the distance to the objects can be determined with usable accuracy

Infrared Sensors



Infrared Sensors

`Problem:` In realistic environments, surfaces are not equal in color

- Colored surfaces reflect different amounts of light
- Black surfaces are practically invisible
- In fact, IR-sensors can only be used for object detection, but not for exact distance measurement
- If an IR-signal is received by the sensor, one can assume, that there's an object in front of of the sensor
- `Note:` A missing IR-signal does not necessarily mean there is no object in front of the sensor
- IR-sensors are usually used for short distances (50 to 100 cm)

Infrared Sensor Application

IFIXIT

Table of Contents

Ultrasonic Sensors

Dolphins and Bats use various different **so**und **n**avigation **a**nd **r**anging (sonar) techniques:

- Fixed frequencies
- Varying frequencies

Note: Although artificial ultrasonic sensors are capable of creating frequencies similar to those in the animal world, the animal capabilities remain unmatched

Ultrasonic Sensors

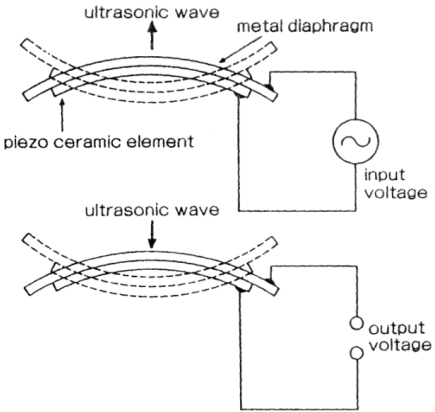
- Ultrasonic waves are differentiated from electromagnetic waves based on the following physical properties:
  - Medium
  - Speed (in medium)
  - Wavelength
- Ultrasonic waves require a medium like air or water
- Ultrasonic speed in air amounts to $331.3\,m/s + 0.6 \times {}^{\circ}C$
- *Time of flight* measurement is possible for short distances
- The wavelength of an ultrasonic sensor driven with a frequency of $50\,kHz$ amounts to $\approx 6.872\,mm$

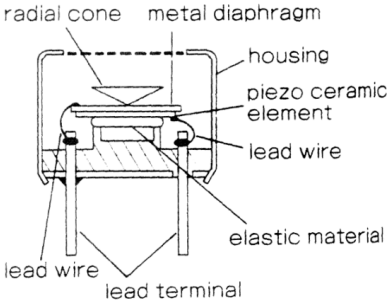
Ultrasonic Sensors

Piezoelectric ultrasonic transducer:

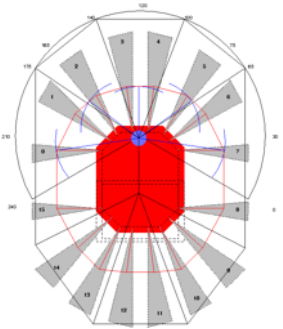
- To produce ultrasonic waves, the movement of a surface is required, leading to a compression or expansion of the medium
- One possibility to generate ultrasonic waves is the use of a piezoelectric transducer
- Applied voltage causes a bending of the piezoelectric element
- Piezoelectricity is *reversible*, therefore incoming ultrasonic waves produce an output voltage
- The opening angle (*beam angle*) of the ultrasonic signal, can be up to 30° wide

Ultrasonic Sensors





Piezoelectric Ultrasonic Transducer



Example: Pioneer platform equipped with 16 ultrasonic (sonar) sensors

Ultrasonic Precision

The minimum distance d_{min} which can still be measured, is specified as:

$$d_{min} = \frac{1}{2} v t_{Impulse}$$

v: Speed of the wave in the corresponding medium
$t_{Impulse}$: Duration of the emitted impulse in seconds

The maximum distance d_{max} which can still be measured, is specified as:

$$d_{max} = \frac{1}{2} v t_{Interval}$$

v: Speed of the wave in the corresponding medium
$t_{Interval}$: Time span between the single impulses

Ultrasonic Sensor Errors

Reflection of ultrasonic waves from smooth (and flat) surfaces is well-defined.

However:

- Very rough structures lead to a *diffused* reflection of ultrasonic waves
- **Note:** A round rod also produces a diffused reflection

Ultrasonic Sensor Errors

Measurements with sonar sensors are subject to several inaccuracies

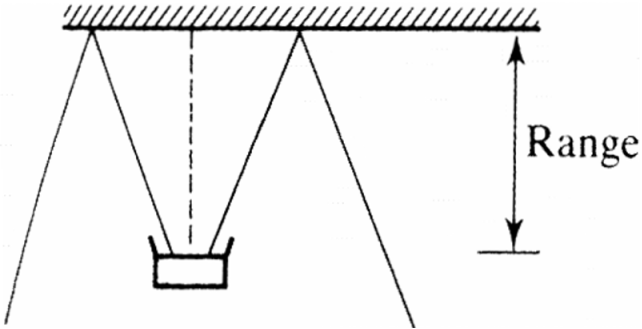
- An object perceived at a distance may be located at an arbitrary position within the sonar cone on the arc at a distance
- Mirror and total reflections cause flawed measurements
- If the sonar beam hits a smooth object in a flat angle, the signal will usually be deflected and no echo will reach the sensor

Ultrasonic Sensor Errors

Caution:

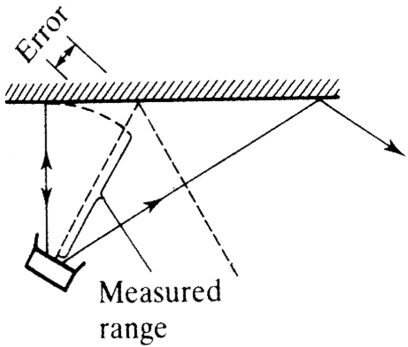
- If several sonar sensors are used simultaneously, specifically encrypted signals need to be used, because otherwise crosstalk may occur
- Since the measurement depends on the temperature of the medium, a change in air temperature will introduce measurement errors (e.g. a difference of 16°C will cause a measurement error of 30cm over a distance of 10m)

Ultrasonic Sensor Errors



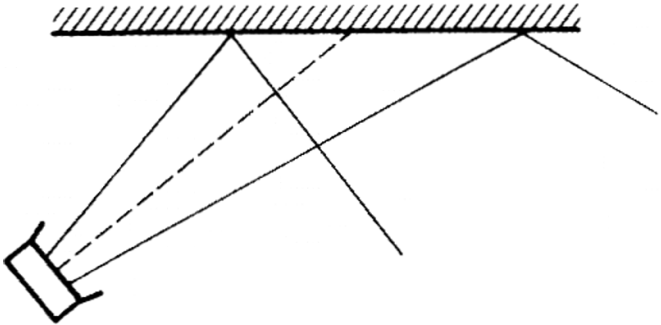
Measuring principle

Ultrasonic Sensor Errors



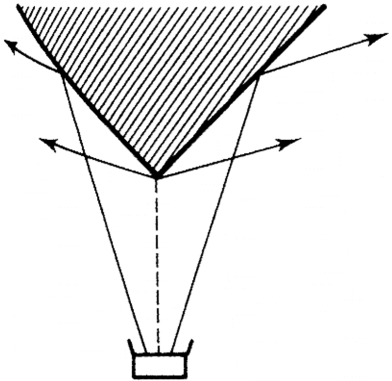
Orthogonal error

Ultrasonic Sensor Errors



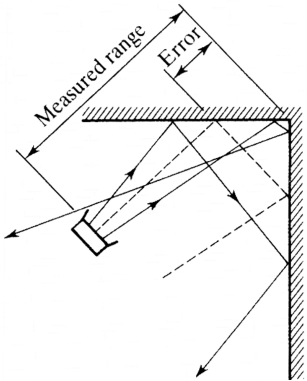
Invisible wall

Ultrasonic Sensor Errors



Invisible corner

Ultrasonic Sensor Errors



Corner error

Ultrasonic Sencor Application

https://www.carid.com/parking-sensors.html

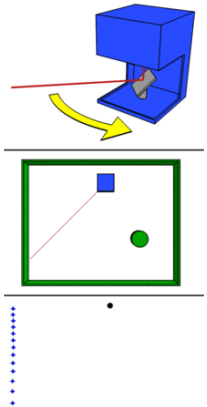
Table of Contents

Laser Range Finders

- Laser range finders (LRF) measure the distance, speed and acceleration of recognized objects
- Functional principle similar to that of a sonar sensor
- Instead of a short sonic impulse, a short light impulse is emitted from the laser range finder
- The time span between emission and reception of the reflected impulse is used for distance measurement (*time of flight*)

Laser Range Finders

- Using a rotating mirror, the pulsed laser beam is deflected and the environment is scanned in a fan-shaped area (“laser radar” or Lidar)
- In practice rotation frequencies between 0.1Hz and 100Hz are used

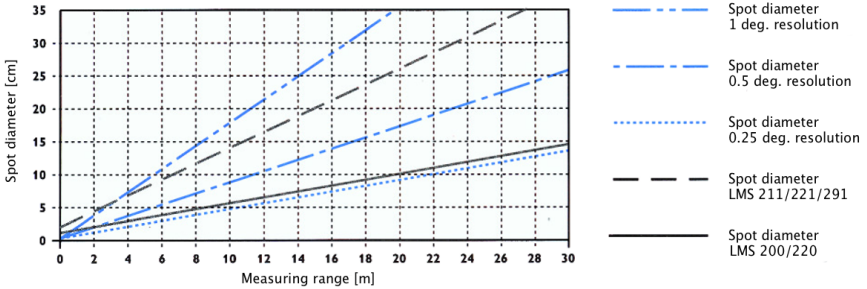


LRF

GIF

Laser Range Finders

- Within its field (plane) of view the LRF emits a light impulse (spot) with a typical resolution of 0.25°, 0.5° or 1°
- Due to the geometry of the beam and the diameters of the single spots, they overlap on the measured object up to a certain distance

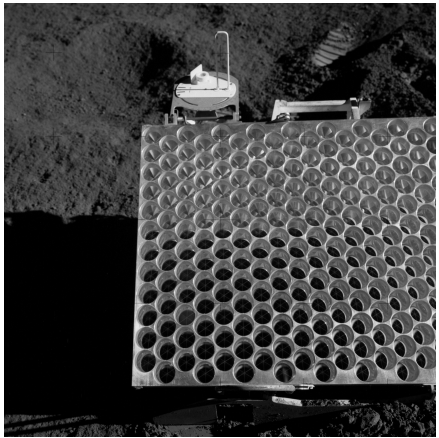


Laser Range Finders

The range of the laser range finders depends on the *remission* (reflectivity) of the object and the transmitting power

Material	**Remission**
Cardboard, black	10 %
Cardboard, grey	20 %
Wood (raw, dirty)	40 %
PVC, grey	50 %
Paper, white dull	80 %
Aluminum, black	110...150 %
Steel, stainless glossy	120...150 %
Steel, high-gloss	140...200 %
Reflectors	>2000 %

Moon Reflectors



Solid State LIDAR

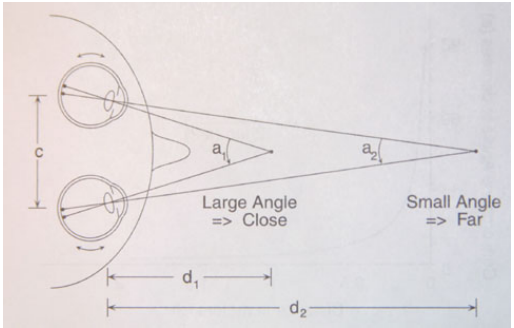
- No moving parts
- Cheaper manufacturing
- More robust towards vibrations
- Smaller FoV
- Not fully established on the market
- Various different methods are evaluated
  - Flash
  - MEMS mirrors
  - Spectral deflection
  - Optical phased array

Lidar

Video
https://www.youtube.com/watch?v=SuejEZebUdM

Table of Contents

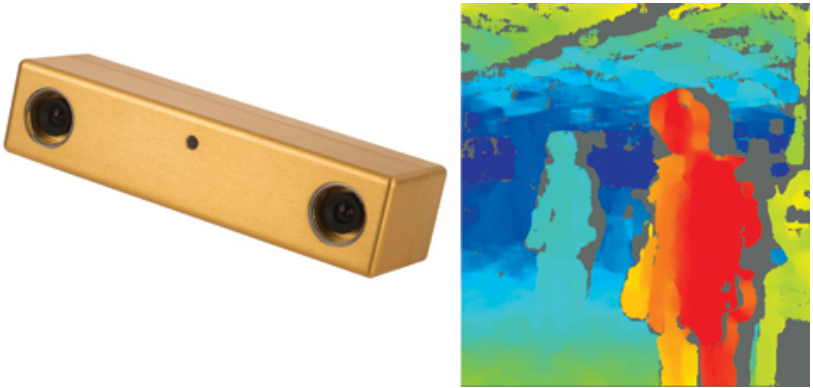
Human Stereo Vision



$$d = c/(2 * tan(a/2))$$

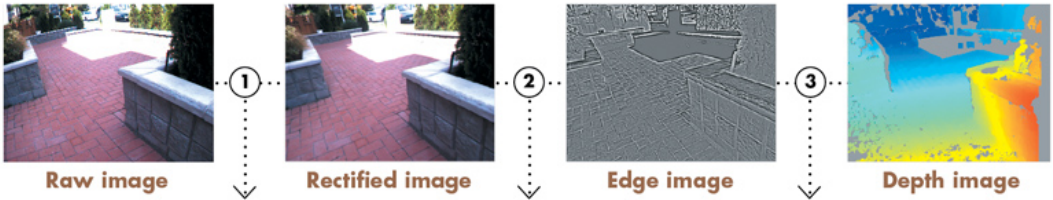
Human performance: up to around 2m

Robot Stereo Camera



https://aemstatic-ww2.azureedge.net/content/dam/VSD/print-articles/2014/11/1412VSD_ProdFocus_Fig1b.jpg

Stereo Camera Example



Stage 1: Rectification
After the images are transmitted to the PC over the IEEE-1394 bus, they are corrected and aligned to remove lens distortion.

Stage 2: Laplacian of Gaussian
The Laplacian of Gaussian filter is applied to create edge images that are not biased by image brightness.

Stage 3: Correlation Stereo
For each pixel in the right image, a corresponding pixel in the left image is obtained via correlation using the Sum of Absolute Differences criteria.

https://www.ptgrey.com/stereo-vision-cameras-systems

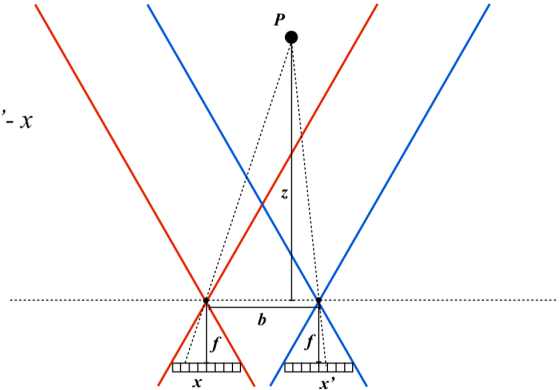
Computing Depth

Focal length: f

Baseline: b

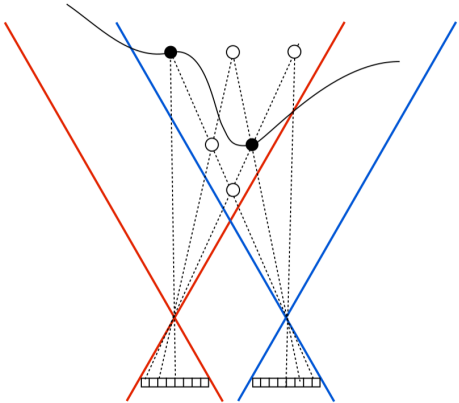
Disparity: $d = x' - x$

$$z = \frac{bf}{d}$$



http://graphics.cs.cmu.edu/courses/15869/fall2013content/lectures/19_depthcamera/depthcamera_slides.pdf

Correspondence Problem



http://graphics.cs.cmu.edu/courses/15869/fall2013content/lectures/19_depthcamera/depthcamera_slides.pdf

Problems with Correspondence

Gregory Föll, Stereo Vision

Role of the Baseline

- Small Baseline
  - large depth error
- Large Baseline
  - difficult search problem
  - smaller area for depth information
- Multiple camera setups can provide small and large baselines at the same time
  - Increased complexity for processing multiple images

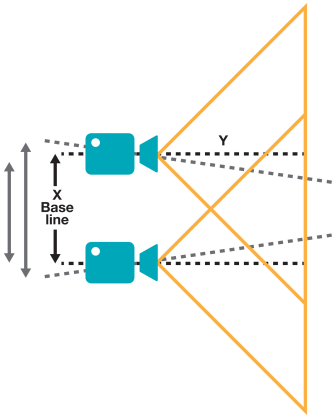


Table of Contents

Humans “Cheating” in 3D Vision

Humans "Cheating" in 3D Vision

Humans use a lot of visual cues for 3D vision

- Shading
- Texture
- Focus
- Motion
- Shadows
- Prior Knowledge
- ...

Art of Photography, Canon

Learning to Cheat like a Human

- Robots can learn the same cheats as humans with one camera
- Monocular depth estimation
- Typically some deep CNN
- Higher relative than absolute precision
- Usually dependent on the environment

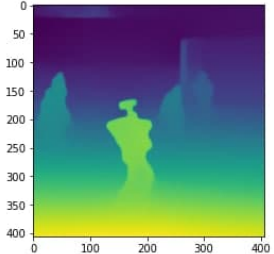


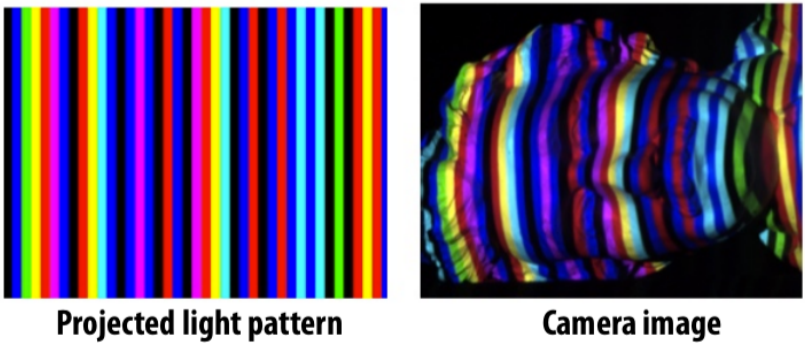
Table of Contents

Depth Camera

- Two different base principles
  - Structured light
  - Time of Flight
- A lot of cheap sensors
  - XBox Kinect (360 / One)
  - Intel RealSense
  - Asus Xtion
  - ...

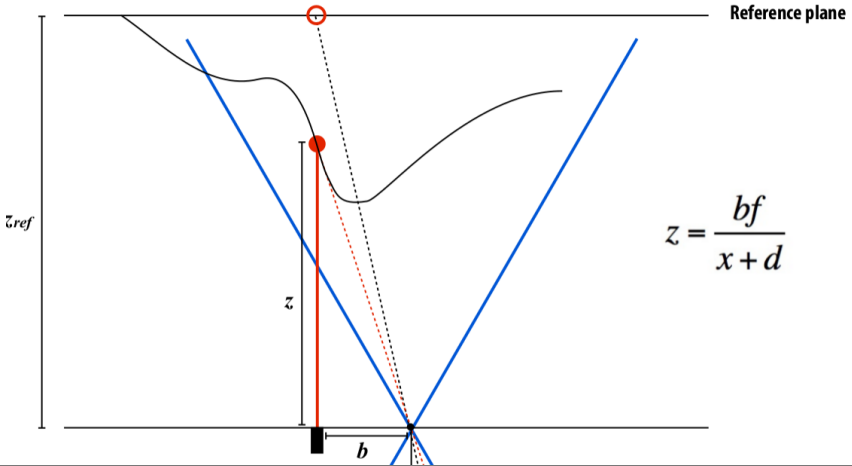
Structured Light

- Simplify correspondence problem by encoding spatial position in light pattern

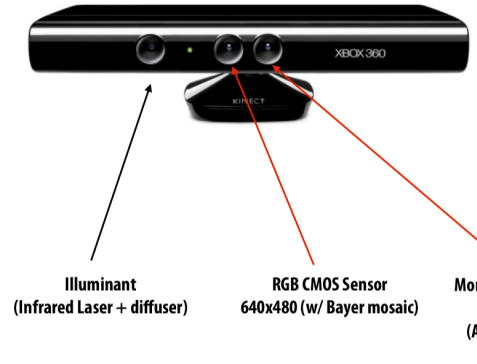


http://graphics.cs.cmu.edu/courses/15869/fall2013content/lectures/19_depthcamera/depthcamera_slides.pdf

Structured Light



Structured Light Application



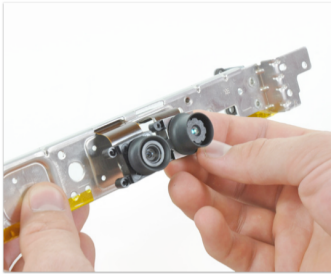
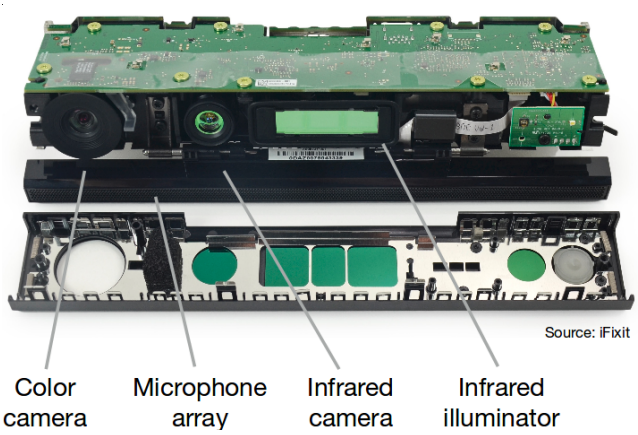


Image credit: iFixIt

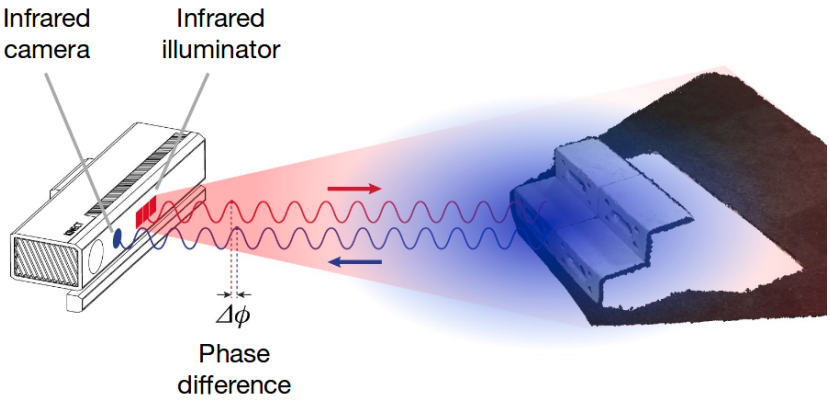
XBox Infrared Output

hackengineer.com

Time of Flight Cameras



Time of Flight Cameras



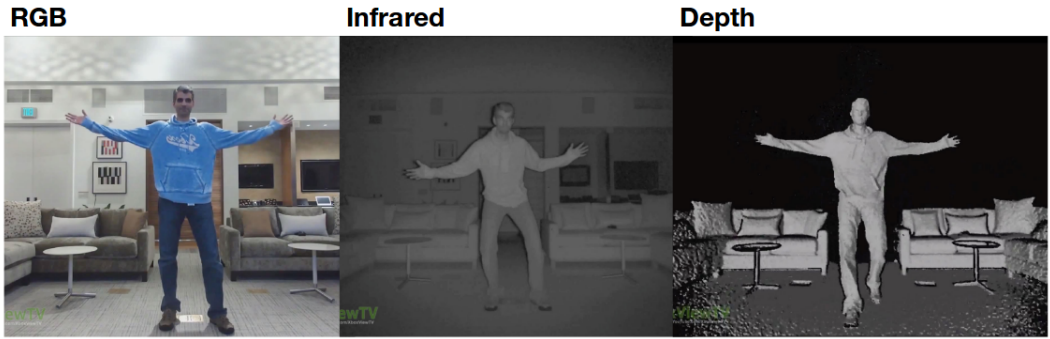
Peter Fankhauser, Kinect v2 for Mobile Robot Navigation Evaluation and Modeling, ETH Zürich

Time of Flight vs. Phase Difference

- Kinect One uses phase difference
- Microsoft calls it “Time of Flight Camera” anyway
- Phase difference is simpler to measure for a whole picture
  - since you can infer the distances in a complete image in a single measurement step

Calibration of Kinect for Xbox One and Comparison between the Two Generations of Microsoft Sensors, Diana Pagliari and Livio Pinto

Time of Flight Cameras



with active IR illumination

Source: XboxViewTV

In reality

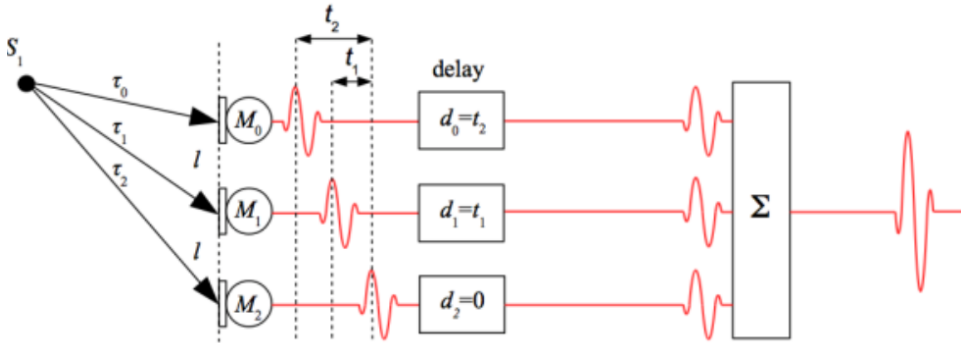
- A lot of different products available
- Integrated combination of multiple cameras and structured pattern
- Depth processing sometimes onboard
- (Proprietary) driver software usually provides depth information
- Open source software for generic stereo camera
  - ROS (stereo_image_proc)
  - OpenCV
- Improvement of these sensors still active field of research

Table of Contents

Audio Localization

- Basic idea similar to stereo camera
- Microphones can be used passively to get position of sound source
- At least two microphones necessary, typically more than two are used (microphone array)
- Typical robotic application: find position of human speaker
- Most difficult problem: correspondence of sound signal
- We will not go into depth, if interested visit signal processing lecture

Audio Localization



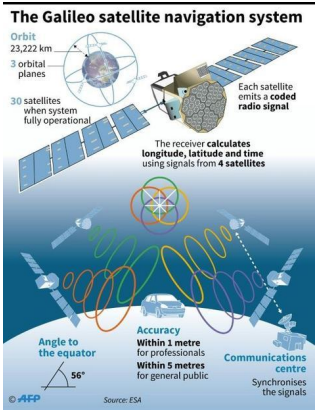
http://yuandenghub.com/wp-content/uploads/2018/07/figure2.png

Table of Contents

Radio Landmark Tracking

- Use radio signals to infer current position
- Mostly by satellites (GPS, GALILEO, GLONASS, ...)
- Also possible with earth bound signals, e.g. WiFi
- Getting absolute position by getting distance to multiple sources and then using triangulation
- The absolute position over time can be used to compute velocity and acceleration

GALILEO



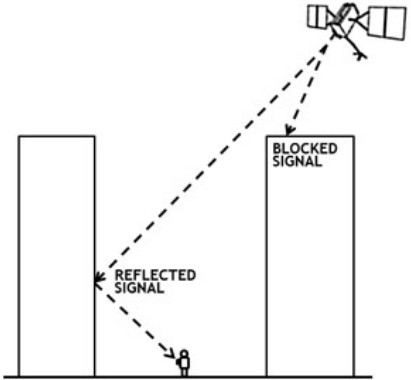
https://phys.org/news/2017-07-europe-galileo-satnav-problems-clocks.html

Satellite Based Radio Landmark Tracking

- Accuracy depends on multiple factors
  - Satellite coverage
  - Signal blockage
  - Atmospheric conditions
  - Receiver design
- Typical accuracy
  - GPS
    - Smartphone: 5m
    - Dual-receiver: few cm
    - Long-term measurement: few mm
  - Galileo, GLONASS similar
  - Much better results and robustness when using combinations

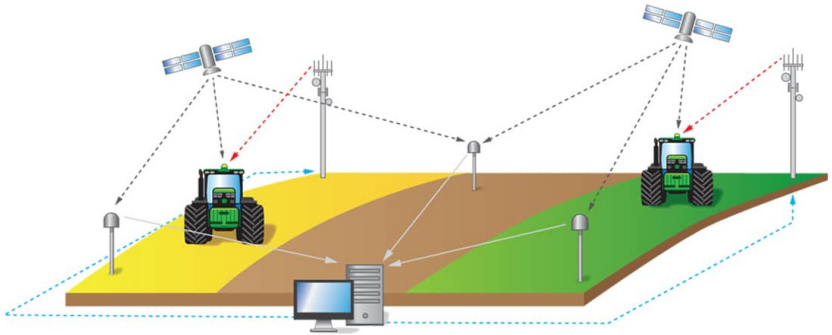
Typical Problems

- Most frequent problems
  - Signal blocked by building, trees
  - Indoor, underground use
  - Signal reflected on buildings or walls
- Less frequent problems
  - Solar storms
  - Radio interference or jamming
  - Satellite maintenance



https://www.gps.gov/systems/gps/performance/accuracy/

GPS precision improvement through Real Time Kinematics



https://www.deere.ca/en/technology-products/precision-ag-technology/guidance/rtk/

GPS precision improvement through Real Time Kinematics

https://www.deere.ca/en/technology-products/precision-ag-technology/guidance/rtk/

Table of Contents

Summary

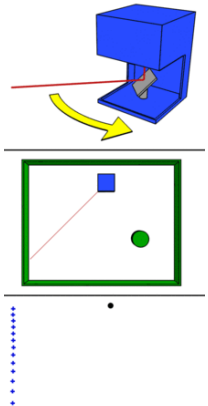
- Different methods to measure distance
  - Time of flight
  - Phase shift
  - Triangulation
  - Reflection intensity
- Multiple sensors based on these methods
  - Infrared sensors
  - Ultrasonic sensors
  - Laser range finders
  - Stereo cameras
  - Structured light cameras
  - Time of flight cameras
- Challenges
  - Material properties
  - Invisible corners/walls
  - Sunlight
  - Multiple active sensors
  - Correspondence

Outline

Table of Contents

Motivation

- Distance scans are a common sensor input
  - Lidar
  - Depth camera
  - ...
- Often used for localization
- Large amount of data
- Errors are common
- Mostly 3D scans, but we will use 2D in this lecture



LRF

GIF

Scan Filtering

Several approaches to processing/filtering of distance measurement data

- Scan data filtering:
  - Smoothing
  - Data reduction
- Feature extraction:
  - Line segments
  - Corners
  - ...
- Clustering/classification

Scan Filtering

- A scan is a set of measurement values

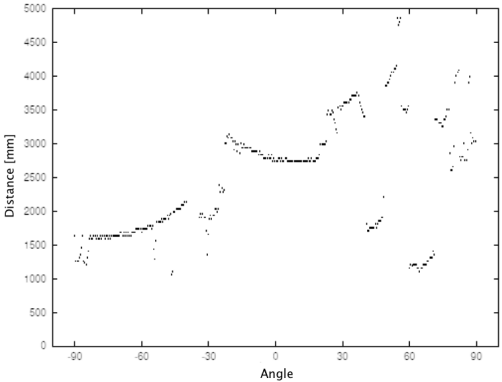
$$\left\{ m_i = (\alpha_i, r_i)^T | i = 0 \dots n-1 \right\}$$

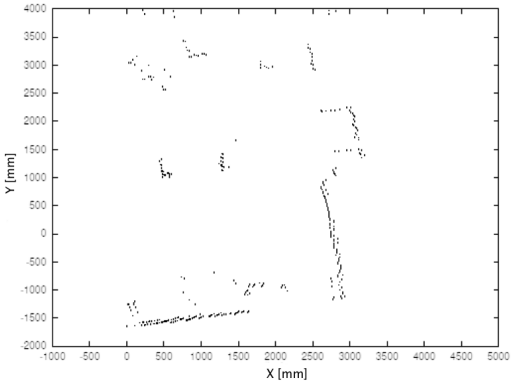
  specified in polar coordinates $(\alpha_i, r_i)^T$

- For a given measuring location $p = (x, y, \theta)^T$ a scan point $m_i = (\alpha_i, r_i)^T$ can be converted to cartesian coordinates

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} r_i \cos\alpha_i \\ r_i \sin\alpha_i \end{bmatrix}$$

Scan Filtering





Scan Filtering

- **Issues**: Big amount of data, noise/outliers, etc.
- **Solution**: Application of filtering procedures according to requirements
- Basic scan data filters are:
  - Median filter
  - Reduction filter
  - Angle reduction filter

Median Filter

The median filter recognizes outliers and replaces them with a more suitable measurement

- A window is placed around each scan point, containing measurements before and after the point
- The value of the scan point is replaced by the median distance within the filter window
- Window size (*wSize*) is the main parameter of the median filter
- Big window sizes lead to a strong smoothing effect
- *Disadvantage*: Corners are rounded

Median Filter

Example

Raw sensor input:

0.9	0.8	0.1	1	1.1	0.7	1

Applying median filter on fourth value.

Sort entries in window and take center value as result.

Window size 3: [0.1, **1**, 1.1]

Window size 5: [0.1, 0.7, **0.8**, 1, 1.1]

Window size 7: [0.1, 0.7, 0.8, **0.9**, 1, 1, 1.1]

Median Filter

Reduction Filter

The reduction filter reduces point clusters to a single point

- A point cluster is specified through a radius r from the starting point
- The first point (starting point) of a scan starts the first cluster
- All subsequent points at a distance $d < r$ are added to the cluster
- A new cluster is started at the first point with a bigger distance
- Each cluster is replaced by the center of gravity of the corresponding points

Reduction Filter

Reduction Filter

- The reduction filter algorithm has a linear time complexity
- *Advantages* of the reduction filter:
  - Reduction of the number of scan points without significant information loss
  - This leads to shorter duration of scan post-processing
  - The result is a more uniform distribution of the points
- *Disadvantages* of the reduction filter:
  - Feature extraction is not as easy any more
  - Possibly too few points for a feature (e.g. corner)

Angle Reduction Filter

The angle reduction filter resembles the reduction filter

- Scan points having a similar measurement angle are grouped and replaced by the point with the median distance
- The angle reduction filter is used for an even reduction of scan data that have a high angular resolution
- The time complexity of the angle reduction filter is linear

Angle Reduction Filter

Table of Contents

Feature Extraction

General approach:

- Extraction of features instead of low-level processing of complete scans
- But what are useful features in point data?
- Common features: Lines, Corners

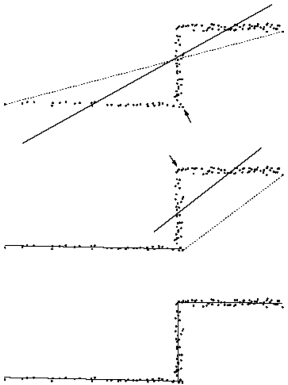
Line Detection by:

- Divide and Conquer Regression
- Hough-Transform
- RANSAC
- ...

Lines - Divide and Conquer

- Initially a regression line is fitted to the points
- If the deviation is too big, the set of points is divided
- Dividing point is the one with the highest distance to the line
- Critical parameters:
  - Minimum number of points to form a line
  - Maximum allowed deviation
- Time complexity similar to *Quicksort*: quadratic in the worst case, logarithmic on average

Lines - Divide and Conquer



Lines - Divide and Conquer

Hough Transform

The Hough transform is a feature extraction approach applied in digital image processing

- Recognition of lines, circles, ...
- Points in the image are mapped onto a parameter space
- Suitable parameters:
  - Line: Slope and y-intercept
  - Circle: Radius and center
- Searched figure is located at the clusters in parameter space
- Usually implemented by histogram-analysis

Straight Line Recognition

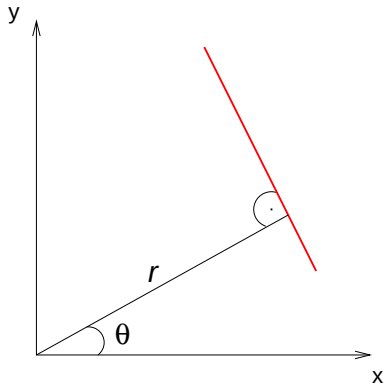
- **Parameters:** Slope and y-intercept
- **Disadvantage:** Straight lines having an infinite slope can not be mapped
- **Better:** Straight line in Hessian normal form

$$r = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

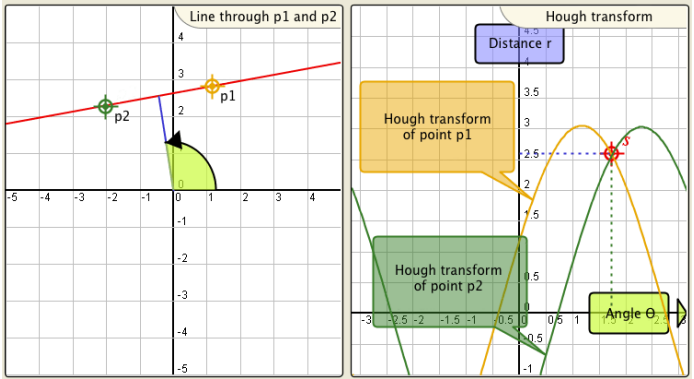
with

- θ: Angle between x-axis and normal of the straight line
- r: Distance between origin and straight line

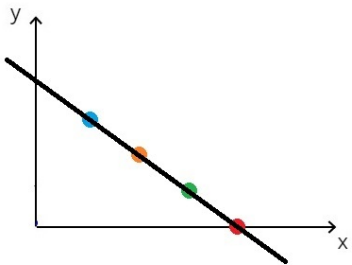
Straight Line Recognition



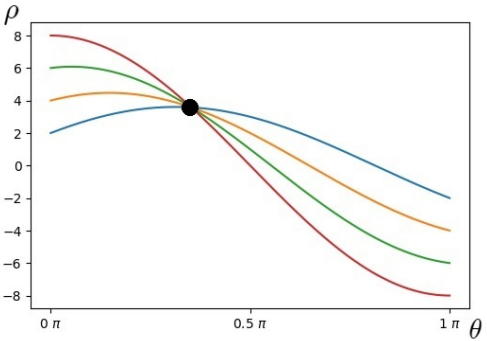
Straight Line Recognition



Straight Line Recognition



Points which form a line

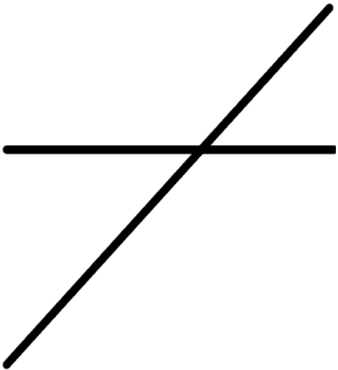


Bunch of sinusoids intersecting at one point

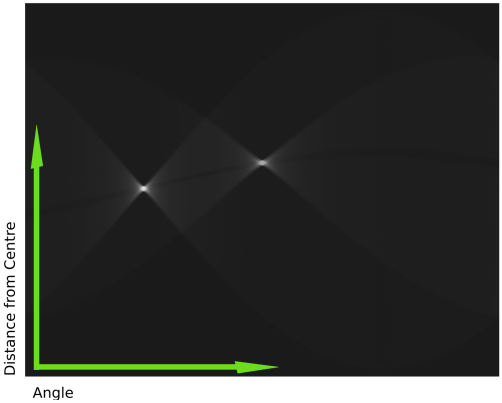
https://tomaszkacmajor.pl/index.php/2017/06/05/hough-lines-transform-explained/

Straight Line Recognition

Input Image



Rendering of Transform Results



The axis zero point is in the center of the images

Straight Line Recognition

All extracted/recognized line segments can be formulated in Hessian normal form

- Each relevant scan data point is tested with several value pairs from the parameter space
- Points of intersection in parameter space represent potential parameter candidates for the straight line found in the scan data
- If multiple candidates exist *clusters* are formed
- The θ-r-point representing the parameters of the straight line is determined as the center of gravity

Table of Contents

Scan Matching

In mobile robotics, scan data obtained with a laser rangefinder is frequently used to determine the location of a robot on a map

- Raw scan data is transformed into a set of features (e.g. lines)
- The *a priori* available map is searched for overlap and alignment with the extracted set of features
  → e.g. ICL - *Iterative Closest Line*
- The output is a transformation that allows to determine the location that the scan was taken at (*best alignment*)
- This procedure is called scan matching
- Scan matching can be carried out using raw scan data
  → e.g. ICP - *Iterative Closest Point*

Scan Matching

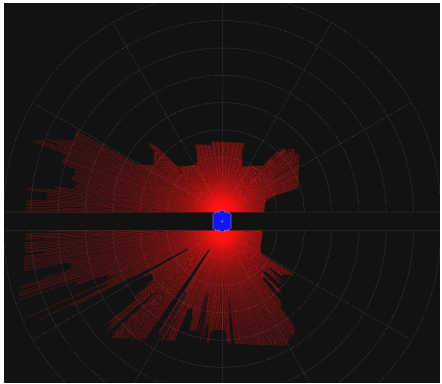
- Scan matching can be performed using:
  - Scan data and map data
  - Scan data and scan data (e.g. *previous scan*)
  - Map data and map data
- Scan matching is an optimization problem that suffers from having many local minima
- The procedure requires a rough estimate of the initial location (e.g from odometry data)
- *ICL* and *ICP* are said to converge if the initial guess is “close enough”

Scan Matching

Scan matching proceeds similar to *Expectation-Maximization algorithms*:

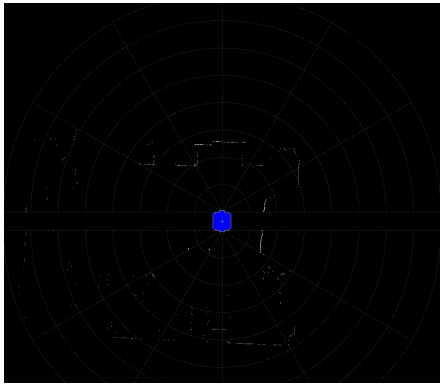
- Let $(x, y, \theta)^T = (s_x, s_y, s_\theta)^T$, where $(s_x, s_y, s_\theta)^T$ is the initial guess of the scan location based on the odometry
- Transform obtained scan data based on the initial guess $(x, y, \theta)^T$
- For each feature, determine the *target* feature closest to it
- Calculate the transformation $T = (\delta x, \delta y, \delta\theta)^T$, which minimizes the sum of squared distances between the extracted features and their targets
- Update $(x, y, \theta)^T = (x, y, \theta)^T + (\delta x, \delta y, \delta\theta)^T$
- Repeat the steps until the procedure converges

Scan Matching



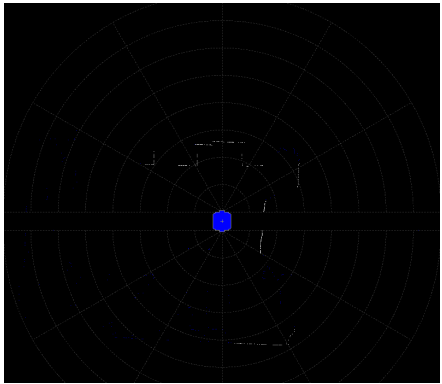
Scan data acquisition

Scan Matching



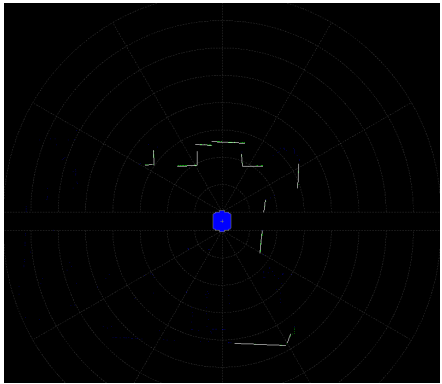
Scan conversion

Scan Matching



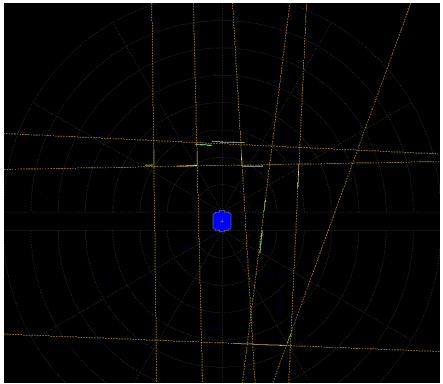
Scan filtering

Scan Matching



Feature extraction

Scan Matching

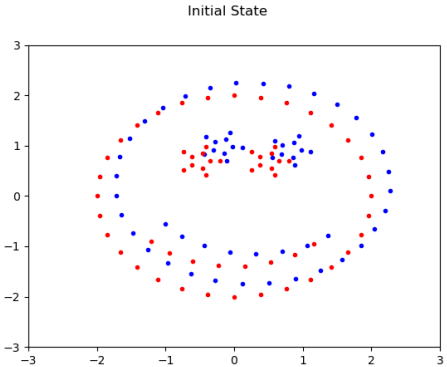


Scan matching

ICP - Iterative Closest Point

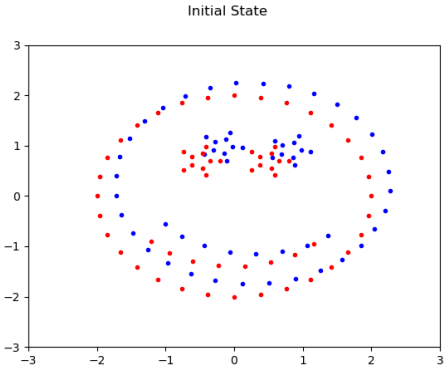
- Step 1: Match every sensed point to closest reference point
- Step 2: Compute transformation that produces least square error for point to point distance (meaning a transformation that will align the points to their partner points from step 1).
- Step 3: Transform all points with transformation from step 2
- Do this until the distance is under some threshold

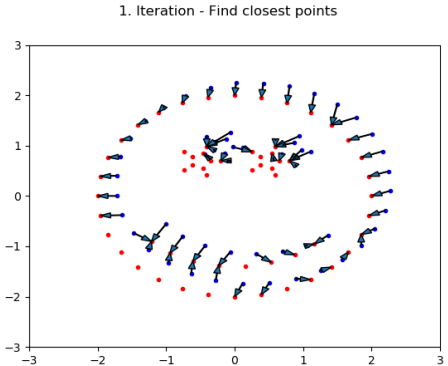
ICP Algorithm



Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

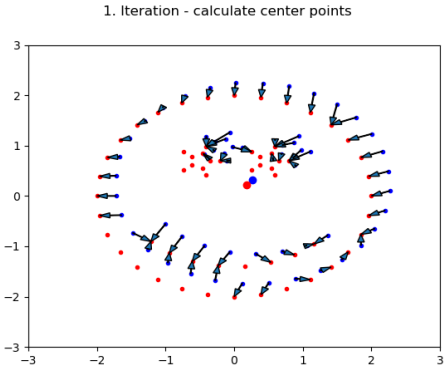
ICP Algorithm





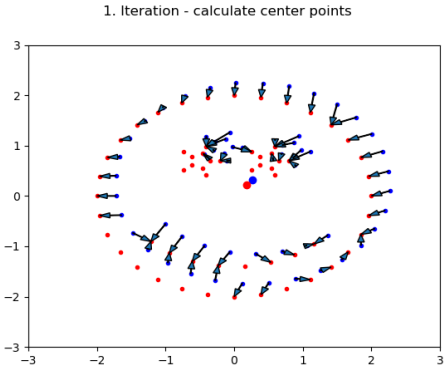
Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

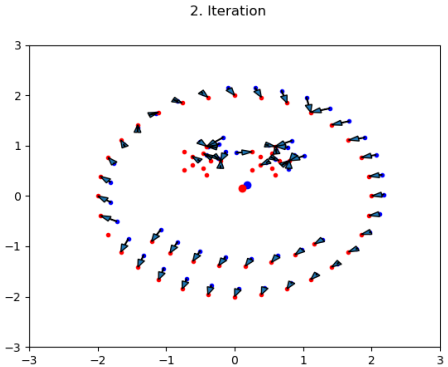
ICP Algorithm



Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

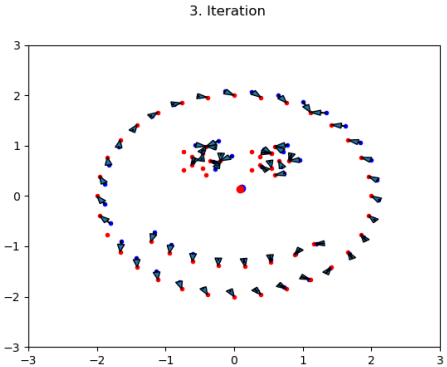
ICP Algorithm





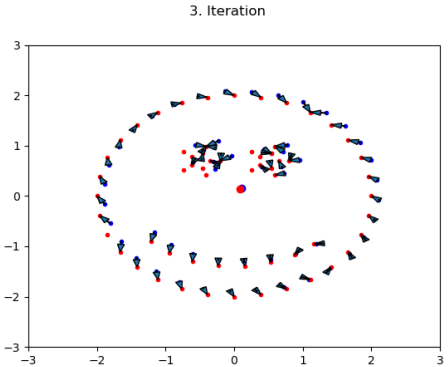
Jonas Tietz, Object reconstruction with ICP, `https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/`

ICP Algorithm



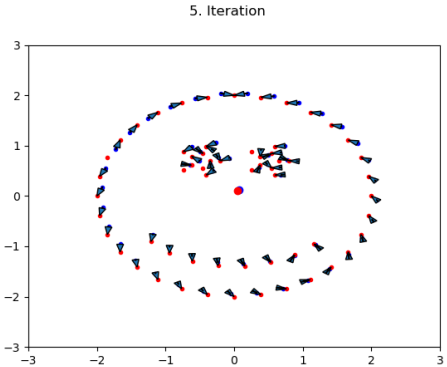
Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

ICP Algorithm



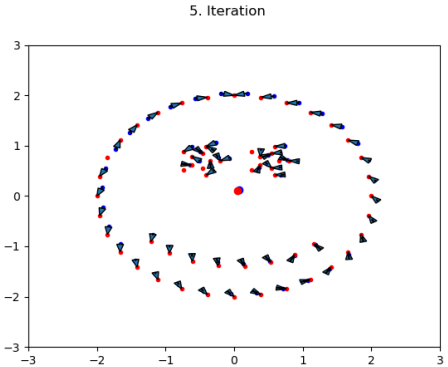
Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

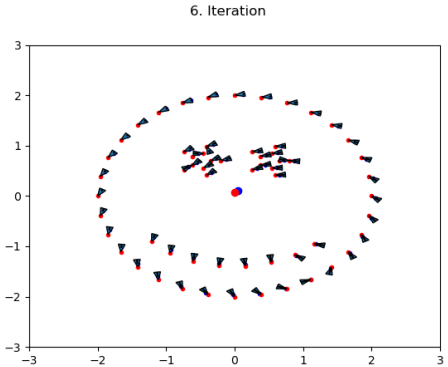
ICP Algorithm



Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

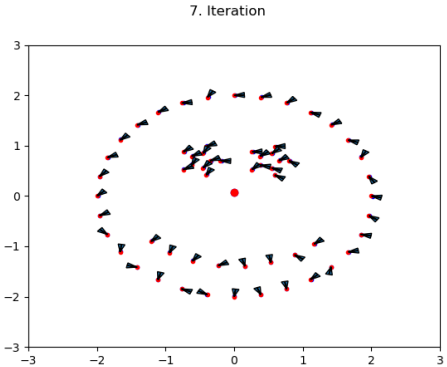
ICP Algorithm





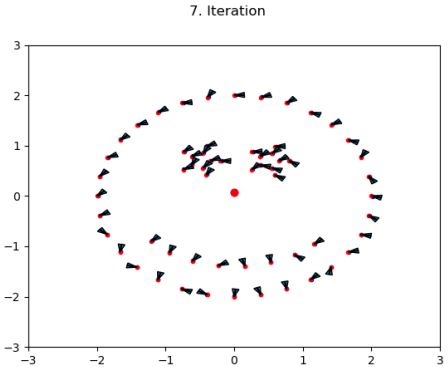
Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

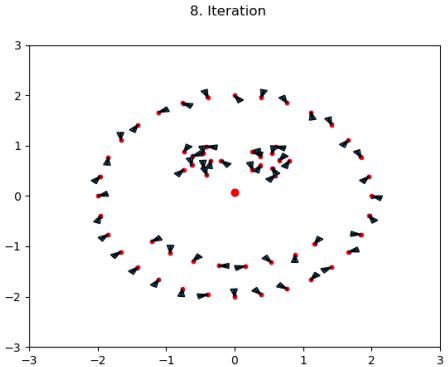
ICP Algorithm



Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

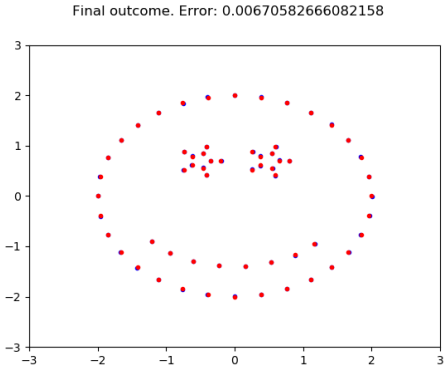
ICP Algorithm





Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

ICP Algorithm



Jonas Tietz, Object reconstruction with ICP, https://tams.informatik.uni-hamburg.de/lectures/2018ws/seminar/ir/

ICP Implementation

A reference implementation can be found here:
https://github.com/AtsushiSakai/PythonRobotics/blob/master/SLAM/iterative_closest_point/iterative_closest_point.py

Outline

Table of Contents

State Estimation

State estimation addresses the issue of recovery of state information from noisy sensor measurement data

- *Issue*: State variables cannot be measured directly or perfectly
- *Idea*: Estimation of state variables through a probabilistic approach
- **Example:** Mobile robot localization
- Probabilistic state estimation algorithms calculate a belief distribution over possible states
- The belief describes the knowledge of a system about the state of its environment

Basic Concepts

Sensor measurements, control variables and the state of a system and its environment can be modeled as a random variable

- Let X be a random variable and x a value which can be assigned to X
- If the value range of X is discrete, one writes

$$p(X = x)$$

to express the probability of X taking on the value x

Basic Concepts

For the sake of simplicity, we can write $p(x)$ instead of $p(X = x)$

- The sum of discrete probabilities is 1:

$$\sum_x p(x) = 1$$

- Probabilities are always non-negative, that means

$$p(x) \geq 0$$

Basic Concepts

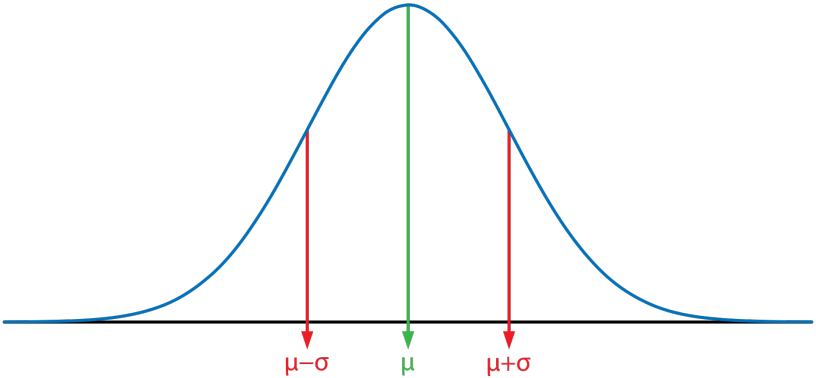
If the value range of a random variable is continuous, the variable is said to possess a probability density function (PDF)

- A typical density function (e.g. for random sensor errors) is the normal distribution with mean value μ and variance σ^2:

$$p(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- If x is a multi-dimensional vector
  - μ becomes a mean *vector*
  - σ^2 is replaced by Σ, a *covariance matrix*

Normal Distribution



https://accadandkoka.com/wp-content/uploads/2019/01/2000px-Gaussian_distribution.png

Basic Concepts

- Similar to the discrete probability distribution, a PDF integrates to 1

$$\int p(x)dx = 1$$

- Unlike discrete probabilities, the value of a PDF does not have an upper bound of 1

Basic Concepts

- The joint probability of X having the value x and Y having the value y is given by

$$p(x, y) = p(X = x \text{ and } Y = y)$$

- If both random variables X and Y are independent of each other, one has

$$p(x, y) = p(x)p(y)$$

- If it is known that Y has the value y, the probability for X under the condition $Y = y$ is given by

$$p(x|y) = p(X = x|Y = y)$$

Basic Concepts

- If one has $p(y) > 0$ for this conditional probability, the following applies

$$p(x|y) = \frac{p(x,y)}{p(y)}$$

- If X and Y are independent variables, one has:

$$p(x|y) = \frac{p(x)p(y)}{p(y)} = p(x)$$

- Thus, if X and Y are independent variables, Y does not tell us anything about X

Basic Concepts (Example)

- Example:
  - X := you sleeping in the lecture $\{0, 1\}$
  - Y := me having an examples in the slides $\{0, 1\}$
  - Z := currently snowing in Antarctica $\{0, 1\}$
- X and Y are definitely dependent and we can assume that $P(X = 1|Y = 0) > P(X = 1|Y = 1)$
  - It is more probable that you sleep if there is no example
- Still $P(X = 1|Y = 1) > 0$
  - You may sleep because you have been on the Reeperbahn all night
  - So there is nothing I can do to keep you awake
- $P(Y = 1|X = 0) = P(Y = 1|X = 1)$
  - You sleeping does not affect if there is an example in the slides

Basic Concepts (Example)

- Maybe $P(X = 1|Z = 0) = P(X = 1|Z = 1)$
  - It does not matter that it snows in Antarctica to the probability of you falling asleep
  - Meaning that X and Z are independent of each other
- But maybe $P(Y = 1|Z = 0) < P(Y = 1|Z = 1)$
  - The probability of snow in Antarctica is depended on the current state of global warming
  - How well I sleep is dependent on the current state of global warming
  - How well I prepare a lecture is dependent on how good I sleep
  - If there is an example in the lecture depends on how well I prepared it
  - In this case $P(X = 1|Z = 0) > P(X = 1|Z = 1)$
- Probabilities are often not as independent as they seem

Basic Concepts

The theorem of total probability relates outcome probabilities to conditional probabilities

$$\begin{aligned} p(x) &= \sum_y p(x|y)p(y) \quad \text{(discrete)} \\ p(x) &= \int p(x|y)p(y)dy \quad \text{(continuous)} \end{aligned}$$

Basic Concepts

The Bayes rule relates the conditional probability $p(x|y)$ to its "inverse" $p(y|x)$ (the rule requires $p(y) > 0$)

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{x'} p(y|x')p(x')} \quad (discrete)$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x')p(x')dx} \quad (continuous)$$

- Bayes rule describes the reversion of conclusions
  - The calculation of $p(effect|cause)$ is usually simple
  - But $p(cause|effect)$ carries more information

Basic Concepts

The Bayes rule plays a fundamental role in state estimation

- If x is the quantity which we want to infer from y, then $p(x)$ is called the prior probability distribution and y is called data (e.g. sensor measurements)
- The distribution $p(x)$ describes the knowledge about X before taking the measurement y into consideration
- The distribution $p(x|y)$ is referred to as the posterior probability distribution of X
- It becomes possible to determine the posterior $p(x|y)$ using the conditional probability $p(y|x)$ and the prior probability $p(x)$

Basic Concepts

- In Bayes rule, $p(y)$ does not depend on x
- Therefore, the factor $\eta = \frac{1}{p(y)} = p(y)^{-1}$ is equal for all values x in $p(x|y)$
- Bayes rule calls this the normalization factor:

$$p(x|y) = \eta p(y|x)p(x)$$

- This notation describes the normalization of the result to 1

Bayes Rule Example

- A := Patient has liver disease. 1/10 of the patients have liver disease -> P(A)=0.1
- B := Patient is an alcoholic. 5% of the patients are alcoholics -> P(B) = 0.05
- Among those patients diagnosed with liver disease, 7% are alcoholics.
- P(B|A)= 0.07: the probability that a patient is alcoholic, given that they have liver disease

Bayes' theorem tells us:
$P(A|B) = (0.07 * 0.1)/0.05 = 0.14$
In other words, if the patient is an alcoholic, their chances of having liver disease is 0.14 (14%)

Example from https://www.statisticshowto.datasciencecentral.com/bayes-theorem-problems/

Basic Concepts

All previous rules may be conditioned on an additional random variable Z

- Conditioning the Bayes rule on $Z = z$ gives us:

$$p(x|y,z) = \frac{p(y|x,z)p(x|z)}{p(y|z)}$$

as long as $p(y|z) > 0$ is true

- Similar to the rule of combination of independent random variables, the following applies:

$$p(x,y|z) = p(x|z)p(y|z)$$

Basic Concepts

- Previous formula describes a conditional independence and is equivalent to

$$\begin{aligned} p(x|z) &= p(x|z,y) \\ p(y|z) &= p(y|z,x) \end{aligned}$$

- The formula implies that y carries no information about x, if z is known
- It does **not** imply, that X is independent of Y:

$$p(x,y|z) = p(x|z)p(y|z) \quad \not\Rightarrow \quad p(x,y) = p(x)p(y)$$

The converse generally does not apply as well:

$$p(x,y) = p(x)p(y) \quad \not\Rightarrow \quad p(x,y|z) = p(x|z)p(y|z)$$

Table of Contents

State

A state can be very different things in robotics, depending on your system.

- **Proprioception** (state of the robot itself)
  - Current speed of the robot
  - Position of the end effector in relation to base
- **Exteroception** (state of the world around the robot)
  - Pose of the object that I want to grasp
  - Obstacles in the robots path

State

The state of a system can be described through a probability distribution

$$p(x_t | x_{0:t-1}, z_{1:t}, u_{1:t})$$

which depends on:

- All previous states $x_{0:t-1}$
- All previous measurements $z_{1:t}$ and
- All previous control variables (control commands) $u_{1:t}$

State

A state x is said to be complete, if knowledge of past states does not carry any information that would improve the estimate of the future state

- Assuming a complete state only the control variable u_t is important if state x_{t-1} is known ($\rightarrow$ conditional independence)

$$p(x_t|x_{0:t-1}, z_{1:t}, u_{1:t}) = p(x_t|x_{t-1}, u_t)$$

- The measurement probability distribution is specified in a similar way

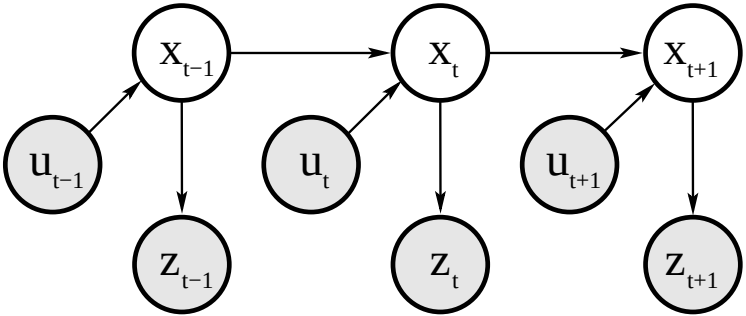
$$p(z_t|x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t|x_t)$$

- In other words: The state x_t is sufficient to predict the measurement z_t

State

- The conditional probability $p(x_t|x_{t-1}, u_t)$ is called state transition probability
- It describes how the state of the environment changes depending on the control variables
- The probability $p(z_t|x_t)$ is called measurement probability
- Both probabilities together describe a dynamic stochastic system
- Such as system description is also known as Hidden Markov Model (HMM) or Dynamic Bayes Network (DBN)

State



A dynamic Bayes network describing the development of states, measurements and controls

Belief

The knowledge of a system about its state is called belief

- The *true state* of a system is **not equal** to the *belief*
- The *belief* is the posterior probability of the state variable based on previous measurement data

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t})$$

- This definition defines the *belief* as probability after measurement

Belief

- The *belief* before incorporation of measurements is called the prediction

$$\overline{bel}(x_t) = p(x_t | z_{1:t-1}, u_{1:t})$$

- The step of calculating $bel(x_t)$ from the prediction $\overline{bel}(x_t)$ is called correction or measurement update

Table of Contents

Bayes Filter

The most fundamental algorithm to calculate *beliefs* is the Bayes filter algorithm

- The algorithm is recursive and calculates the belief distribution $bel(x_t)$ at time t from the following quantities
  - $bel(x_{t-1})$ at the time of $t-1$
  - The measurement data z_t
  - The control data u_t

Bayes Filter

The general Bayes filter algorithm

Algorithm Bayes_Filter$(bel(x_{t-1}),\ u_t,\ z_t)$:

1. **for all** x_t **do**
2. $\quad \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
3. $\quad bel(x_t) = \eta\, p(z_t|x_t)\overline{bel}(x_t)$
4. **endfor**
5. **return** $bel(x_t)$

Bayes Filter

The Bayes filter algorithm has two essential steps

- In line 2, it processes the control variable u_t
- $\overline{bel}(x_t)$ is the integral (sum) of the product of two probability distributions:
  - The prior for state x_{t-1} and
  - The probability of switching to state x_t when u_t occurs
- That is the prediction step
- In line 3, the correction step is executed
- $\overline{bel}(x_t)$ is multiplied with the probability of detection of the measurement z_t in this state

Bayes Filter Algorithm

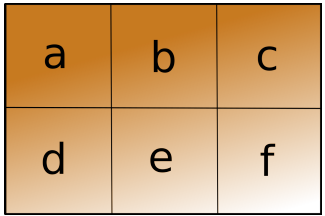
- Due to its recursive nature the Bayes filter requires an initial belief $bel(x_0)$ at time $t = 0$ as a boundary condition
- If the initial state x_0 is known with certainty, $bel(x_0)$ should be initialized with a *point mass distribution* focused on x_0
- If the initial state is completely unknown, $bel(x_0)$ should be initialized with a *uniform distribution*

Bayes Filter Algorithm

- In the presented form, the algorithm can only be implemented for very simple problems
- Either the integration in line 2 and the multiplication in line 3 need to have a closed form solution, ...
- ...or a finite state space must be given, so that the integral in line 2 becomes a sum

Bayes Filter - Example

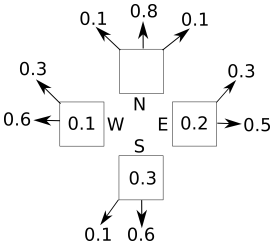
- Assume an agent in this small grid world



- The agent's *state* is $x \in \{a, b, c, d, e, f\}$
- The agent's belief is a distribution $bel(x)$
- The agent can aim to move (*transition*) North, East, South, and West
- It can measure its *longitude* (i.e. column)

Bayes Filter - Example

- The agent can choose $u \in \{N, E, S, W\}$
- It might end up somewhere else though:



- The agent can measure its current column $z \in \{-1, 0, 1\}$
- The measurement might be faulty

-1	0	1
0.25	0.5	0.25

- When the agent would hit a wall, it moves along the wall instead

Bayes Filter - Example

- Assume some distribution as the initial belief $bel(x_0)$
- Choose an action u_1 and compute the prediction $\overline{bel}(x_1)$
- Take a measurement z_1 and compute the belief $bel(x_1)$

Bayes Filter - Example

x_0 :

0	0	0
0	1	0

$u_1 : N$

$\overline{bel_1}$:

$1 * 0.1$	$1 * 0.8$	$1 * 0.1$
0	0	0

$=$

0.1	0.8	0.1
0	0	0

$z_1 : 0$

bel_1 :

$0.1 * 0.25$	$0.8 * 0.5$	$0.1 * 0.25$
0	0	0

$=$

0.025	0.4	0.025
0	0	0

normalized

0.056	0.89	0.056
0	0	0

Bayes Filter - Example

bel_1 :

0.056	0.89	0.056
0	0	0

$u_2 : S$

$\overline{bel_2}$:

$0.056 * 0.3$	$0.89 * 0.3$	$0.056 * 0.3$
$0.056 * 0.1 + 0.056 * 0.6 + 0.89 * 0.1$	$0.89 * 0.6 + 0.056 * 0.1$	$0.056 * 0.6$

$=$

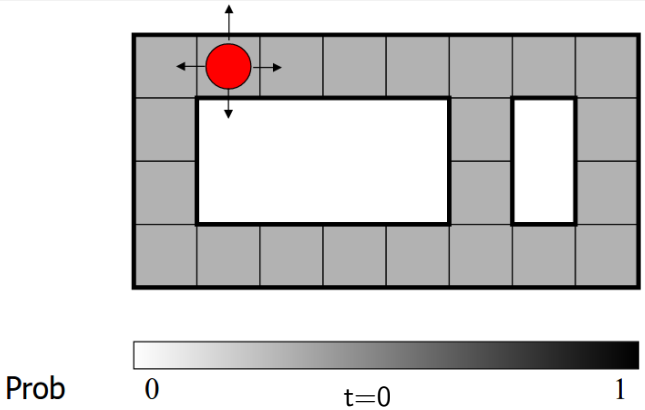
0.0168	0.267	0.0168
0.1282	0.539	0.0504

normalize it again

take measurement

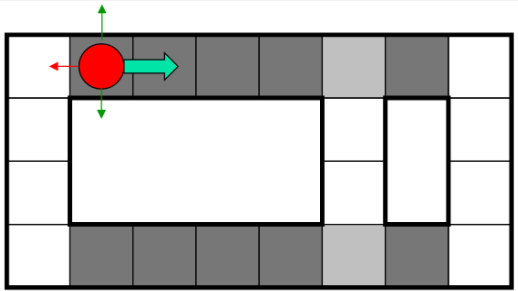
...

Bayes Filter - Example 2



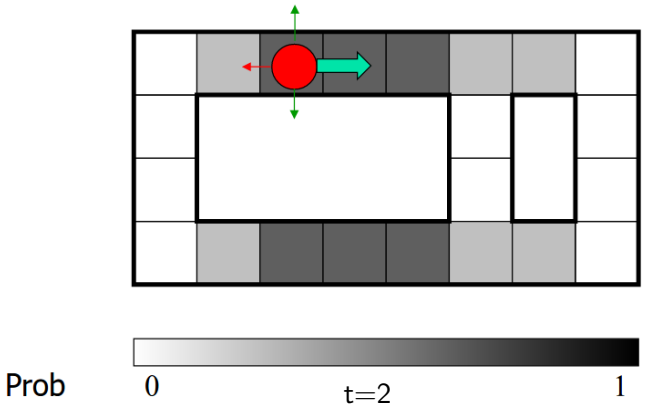
https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes Filter - Example 2



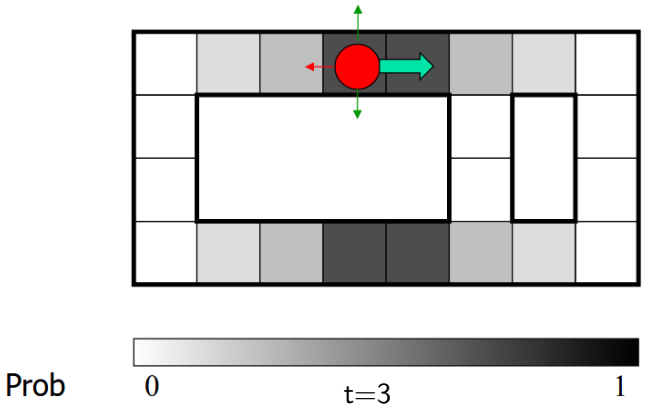
https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes Filter - Example 2



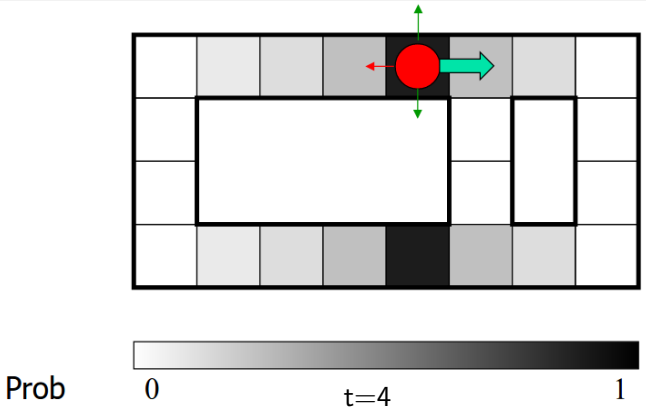
https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes Filter - Example 2



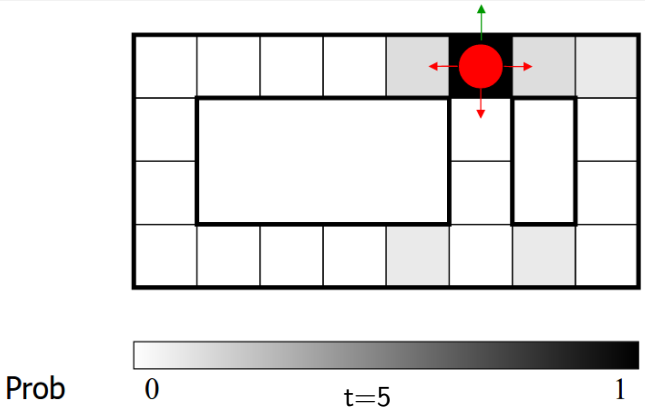
https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes Filter - Example 2



https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes Filter - Example 2



https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes Filter

Video
https://www.youtube.com/watch?v=XFoGDvTOR28

Markov Assumption

The assumption of a state being complete is called Markov assumption

- The assumption states independence of past and future data, if the current state x_t is known

The following is meant to illustrate, how tough this assumption is:

- Assuming that Bayes filters are used for localization of mobile robots, ...
- ... and x_t is the *pose* of the robot in relation to a static map

Markov Assumption

There are effects which falsify sensor measurements systematically and therefore render the Markov assumption void:

- Inaccuracies in the probabilistic models $p(x_t|u_t, x_{t-1})$ and $p(z_t|x_t)$
- Rounding errors, if approximations for the representation of the *belief* are used
- Variables within the software, which affect several control variables
- Influence of moving persons on sensor measurements

Some of these variables could be included in the state, but are often ignored in order to reduce computational effort

Bayes Filter Implementation

Algorithm Bayes_Filter$(bel(x_{t-1}),\ u_t,\ z_t)$:

1. **for all** x_t **do**
2. $\quad \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
3. $\quad bel(x_t) = \eta\, p(z_t|x_t)\overline{bel}(x_t)$
4. **endfor**
5. **return** $bel(x_t)$

Bayes Filter Implementation

Algorithm Bayes_Filter$(bel(x_{t-1}),\ u_t,\ z_t)$:

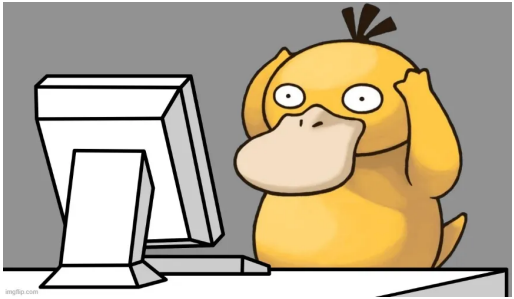
1. **for all** x_t **do**
2. $\quad \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
3. $\quad bel(x_t) = \eta\, p(z_t|x_t)\overline{bel}(x_t)$
4. **endfor**
5. **return** $bel(x_t)$

In a continuous state space we have an infinite amount of x_t!

Bayes Filter Implementation

Algorithm Bayes_Filter$(bel(x_{t-1}),\ u_t,\ z_t)$:

1. **for all** x_t **do**
2. $\quad \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\ bel(x_{t-1})\ dx_{t-1}$
3. $\quad bel(x_t) = \eta\ p(z_t|x_t)\overline{bel}(x_t)$
4. **endfor**
5. **return** $bel(x_t)$



In a continuous state space we have an infinite amount of x_t!

Bayes Filter Implementations

Bayes filters can be implemented in different ways

- The techniques are based on varying assumptions regarding the probability of the measurements, the state transitions and the *belief*
- In most cases the *beliefs* need to be approximated
- This affects the complexity of the algorithms
- None of these techniques should be generally favored over the others

Bayes Filter Implementations

Various Bayes filter implementations express different runtime behavior

- Some approximations require a polynomial runtime, depending on the dimensionality of the state (e.g. Kalman filter)
- Some filters have an exponential runtime
- The runtime of particle based procedures depends on the desired accuracy

Bayes Filter Implementations

Some approximations are better suited to approximate a range of probability distributions

- For uni-modal probability distributions, for example, normal distributions qualify
- Histograms can approximate multi-modal distributions, at the cost of accuracy and computational load
- Particle techniques can approximate a wide range of distributions, possibly resulting in a large number of particles

Summary

Interaction between a robot and its environment is modeled as a coupled dynamic system. For this purpose, the robot sets control variables to manipulate the environment and perceives the environment through sensor measurements

- System dynamics are characterized through the laws of probability theory and two probability distributions:
  - Probability distribution for the state transition
  - Probability distribution for the measurements

  The first one describes how the state changes over time, the second one describes how measurements are perceived

Summary

- The *belief* is the posterior probability of the state, given all previous measurements and control variables
- The *Bayes filter* is a general (recursive) algorithm for calculation of the *belief*
- The Bayes filter works based on the *Markov assumption* $\rightarrow$ The state is a complete summary of the past. In practice, this assumption is usually not true.
- Usually, the Bayes filter can not be applied directly. Implementations can be evaluated based on certain criteria, such as accuracy, efficiency and simplicity.

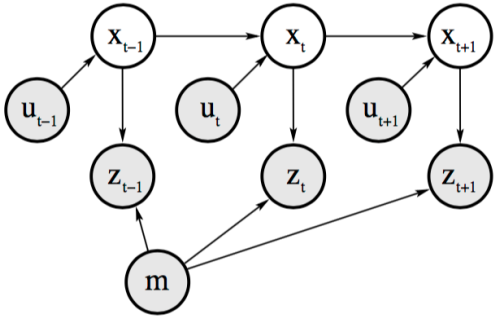
Table of Contents

Localization

A robot's ability to determine its location relative to a map of the environment

- Position tracking
  - Initial robot pose is *known*
  - Localization after control command
  - Pose uncertainty often approximated by a uni-modal distribution
  - Position tracking is a local problem (relative localization)
- Global localization
  - Initial robot pose is *unknown*
  - Uni-modal distributions are no longer appropriate
  - Absolute localization approach
  - Variant: Kidnapped Robot Problem

Localization



Map m, measurements z and controls u are known, robot pose x must be inferred

Localization

Maps are usually specified in one of two forms

- Location-based
  - Planar map with $m_{x,y}$ representing coordinate points
  - Maps are *volumetric*, every point is *labeled*
  - Information about objects in the environment and free space
- Feature-based
  - Map with m_n representing features (objects) in the environment
  - Loss of information, shape of environment known at feature locations only
  - Compact and efficient representation

Markov Localization

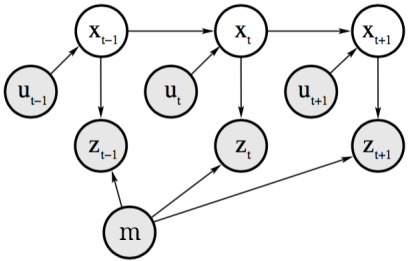
Probabilistic localization approaches are variants of the Bayes filter

- The Bayes filter approach can be applied directly $\rightarrow$ Markov localization
- Markov localization requires a map m of the environment
- The map plays a role in the motion and measurement models
- Markov localization is suitable for position tracking and global localization problems in static environments

Markov Localization

Algorithm Markov_Localization$(bel(x_{t-1}),\ u_t,\ z_t,\ m)$:

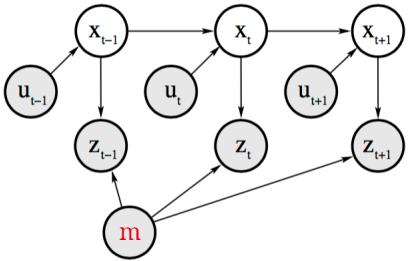
1. **for all** x_t **do**
2. $\quad \overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})\, dx_{t-1}$
3. $\quad bel(x_t) = \eta\, p(z_t \mid x_t, m) \overline{bel}(x_t)$
4. **endfor**
5. **return** $bel(x_t)$



Markov Localization

Algorithm Markov_Localization$(bel(x_{t-1}),\ u_t,\ z_t,\ m)$:

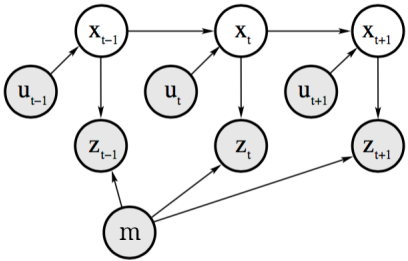
1. **for all** x_t **do**
2. $\quad \overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx_{t-1}$
3. $\quad bel(x_t) = \eta\ p(z_t \mid x_t, m)\overline{bel}(x_t)$
4. **endfor**
5. **return** $bel(x_t)$



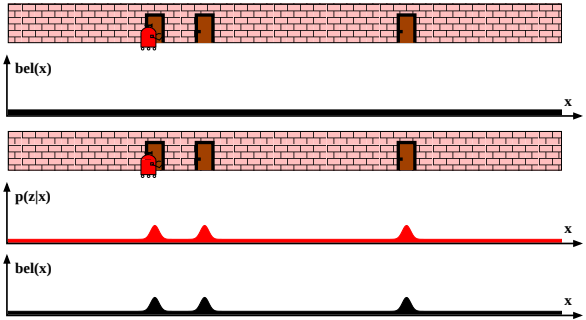
Markov Localization

Algorithm Markov_Localization$(bel(x_{t-1}),\ u_t,\ z_t,\ m)$:

1. **for all** x_t **do**
2. $\quad \overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})\, dx_{t-1}$
3. $\quad bel(x_t) = \eta\, p(z_t \mid x_t, m) \overline{bel}(x_t)$
4. **endfor**
5. **return** $bel(x_t)$

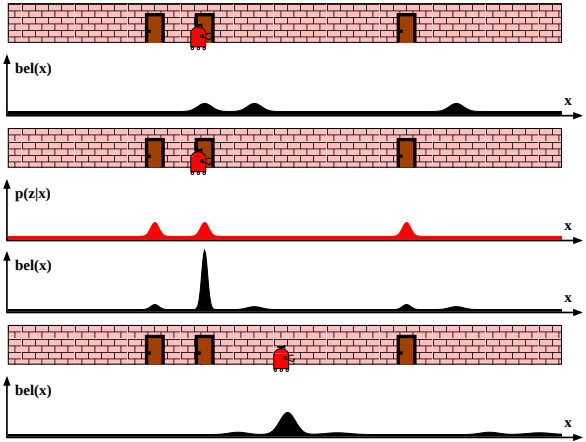


Markov Localization



Convolution of prior with motion model followed by incorporation of the measurement model.

Markov Localization



Grid Localization

Grid localization approximates the *belief* using a Histogram filter applied to the grid decomposition of the state space

- Discretization of the state space through grid cells x
- Allows multimodal distributions
- This discrete Bayes filter handles a multitude of discrete probabilities

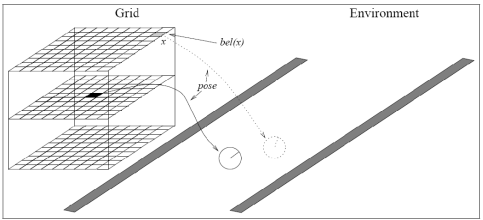
$$bel(x_t) = \{p_{k,t}\}$$

  where each $p_{k,t}$ belongs to a grid cell x_k
- The union of all cells at time t represents the state space X_t
- Two typical grid decomposition approaches exist

Grid Localization

(1) Metric grid decomposition

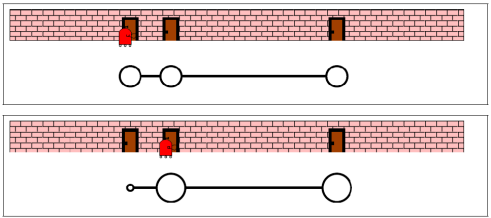
- Grid cells of equal size
- Typical cell sizes have about 15cm depth resolution at about 5° angular resolution
- Higher resolution compared to the topological grid at the cost of an increased computational effort



Grid Localization

(2) Topological grid decomposition

- Cell represents a significant location/feature on the map (Example: Door, Junction ...)
- Resulting grid is usually very coarse
- Grid depends on local map structure/conditions/data



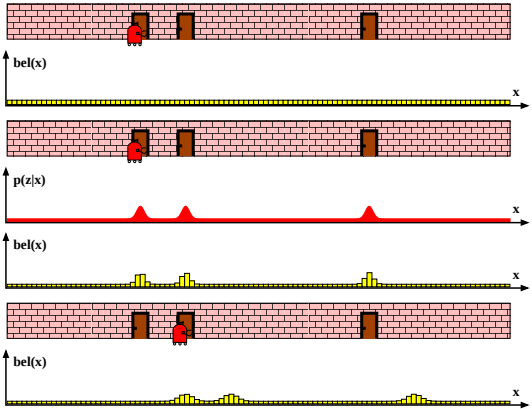
Grid Localization

Grid_Localization$(\{p_{k,t-1},\ u_t,\ z_t,\ m)$:

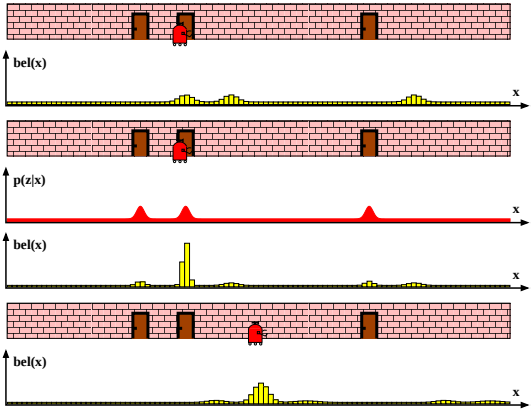
1. **for all** k **do**
2. $\quad \overline{p}_{k,t} = \sum_i \left[p_{i,t-1} \cdot motion\_model(mean(x_k), u_t, mean(x_i))\right]$
3. $\quad p_{k,t} = \eta \cdot measurement\_model(z_t, mean(x_k), m) \cdot \overline{p}_{k,t}$
4. **endfor**
5. **return** $p_{k,t}$

The function *mean* determines the center of mass of a cell x_i

Grid Localization



Grid Localization



Localization

Kalman filter based localization approaches

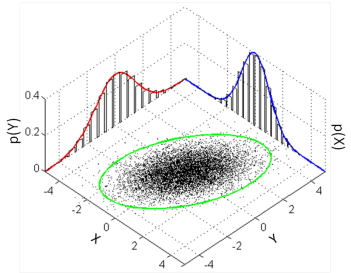
- Belief $bel(x_t)$ represented by uni-modal Gaussian $\mathcal{N}(\mu_t, \Sigma_t)$
- Suitable for pose tracking
- Efficient means for integration of multiple sensors
- Map-based localization requires uniquely identifiable features

Particle filter based localization approaches

- Belief $bel(x_t)$ represented by particles
- Particles are discrete samples of the state probability distribution
- Suitable for pose tracking and global localization problems

Multi Dimensional Gaussian

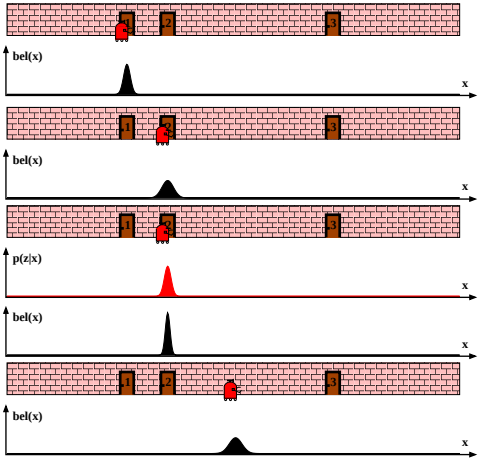
The Gaussian can be multi dimensional



In this case $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 1 & \frac{3}{5} \\ \frac{3}{5} & 2 \end{bmatrix}$

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

Kalman Filter



Kalman Filter

The Kalman filter assumes linear system dynamics

- The state transition probability must be a linear function with added Gaussian noise

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

- F_t is the state transition model
- B_t is the control-input model
- w_t models the uncertainty introduced by the state transition (the process noise), with its covariance denoted by Q_t

Kalman Filter

- The measurement probability must also be a linear function with added Gaussian noise

$$z_t = H_t x_t + v_t$$

- H_t is the observation model
- v_t is the observation model noise with a zero mean Gaussian with its covariance denoted by R_t
- K_t represents the Kalman gain, a specification of the degree to which the measurement is incorporated into the new state estimate. Higher gain -> most recent measurement is taken more into account

Kalman Filter

Algorithm Kalman_Filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

1. $\bar{\mu}_t = F_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + Q_t$
3. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t (z_t - H_t \bar{\mu}_t)$
5. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$
6. **return** μ_t, Σ_t

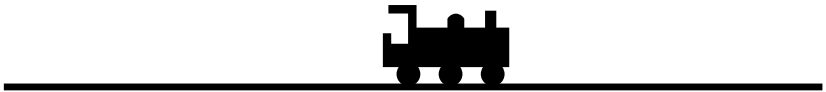
Kalman Filter Online Demo

https://avivace.github.io/kalman/

Kalman Filter Example - Scenario

Let's apply the Kalman filter to a theoretical scenario

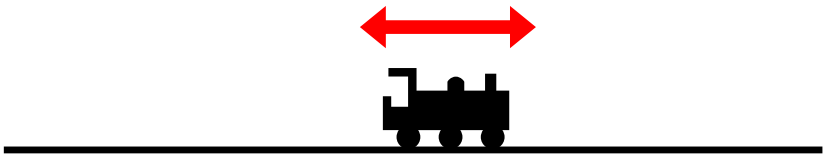
- A train which can move forward and backward on tracks



Kalman Filter Example - Scenario

Let's apply the Kalman filter to a theoretical scenario

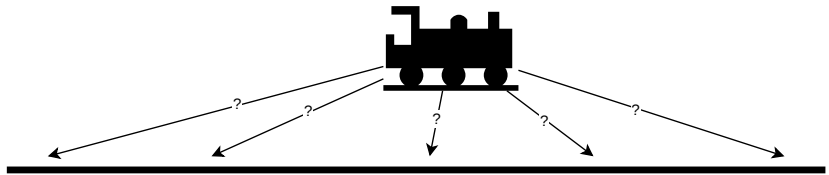
- A train which can move forward and backward on tracks



Kalman Filter Example - Scenario

Let's apply the Kalman filter to a theoretical scenario

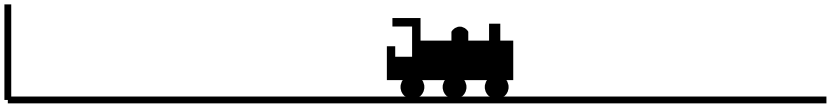
- A train which can move forward and backward on tracks
- We do not know where we start



Kalman Filter Example - Scenario

Let's apply the Kalman filter to a theoretical scenario

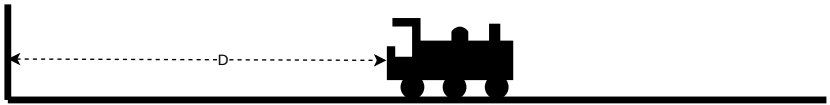
- A train which can move forward and backward on tracks
- We do not know where we start
- There is a wall at the end of the track



Kalman Filter Example - Scenario

Let's apply the Kalman filter to a theoretical scenario

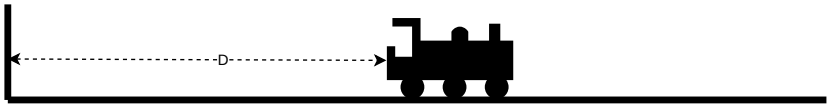
- A train which can move forward and backward on tracks
- We do not know where we start
- There is a wall at the end of the track
- We need to know the distance D to the wall



Kalman Filter Example - Scenario

Let's apply the Kalman filter to a theoretical scenario

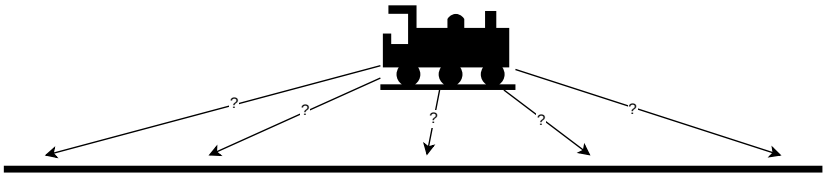
- A train which can move forward and backward on tracks
- We do not know where we start
- There is a wall at the end of the track
- We need to know the distance D to the wall
- We can measure the distance to the wall



Kalman Filter Example - Initial Belief

We need to start off with an initial belief

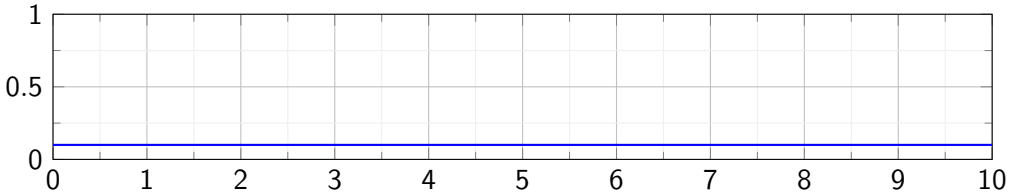
- We do not know where we start
- Still, an initial belief x_0 needs to be set
- In a Kalman filter this means that we have to define μ_0 and Σ_0



Kalman Filter Example - Initial Belief

We need to start off with an initial belief

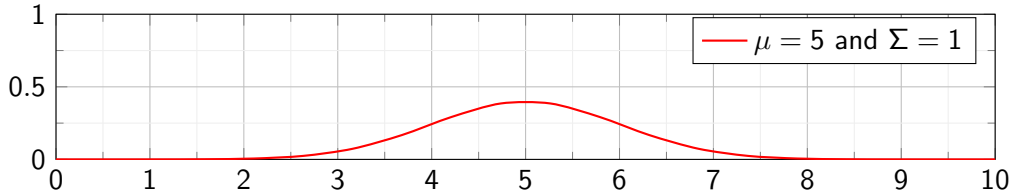
- We do not know where we start
- Still, an initial belief x_0 needs to be set
- In a Kalman filter this means that we have to define μ_0 and Σ_0



Kalman Filter Example - Initial Belief

We need to start off with an initial belief

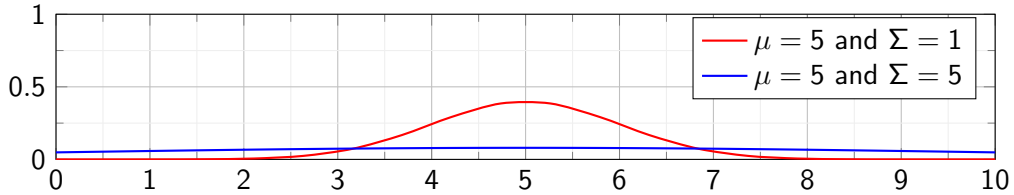
- We do not know where we start
- Still, an initial belief x_0 needs to be set
- In a Kalman filter this means that we have to define μ_0 and Σ_0



Kalman Filter Example - Initial Belief

We need to start off with an initial belief

- We do not know where we start
- Still, an initial belief x_0 needs to be set
- In a Kalman filter this means that we have to define μ_0 and Σ_0



Kalman Filter Example - State Transition Probability

Next, we define our state transition probability

- Remember the state transition probability in a Kalman filter

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

Kalman Filter Example - State Transition Probability

Next, we define our state transition probability

- Remember the state transition probability in a Kalman filter

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

- State transition model F_t

Kalman Filter Example - State Transition Probability

Next, we define our state transition probability

- Remember the state transition probability in a Kalman filter

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

- State transition model $F_t = 1$

Kalman Filter Example - State Transition Probability

Next, we define our state transition probability

- Remember the state transition probability in a Kalman filter

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

- State transition model $F_t = 1$
- Control-input model B_t

Kalman Filter Example - State Transition Probability

Next, we define our state transition probability

- Remember the state transition probability in a Kalman filter

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

- State transition model $F_t = 1$
- Control-input model $B_t = 1$
- State transition uncertainty w_t

Kalman Filter Example - State Transition Probability

Next, we define our state transition probability

- Remember the state transition probability in a Kalman filter

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

- State transition model $F_t = 1$
- Control-input model $B_t = 1$
- State transition uncertainty w_t characterized by Q_t

Kalman Filter Example - State Transition Probability

Next, we define our state transition probability

- Remember the state transition probability in a Kalman filter

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

- State transition model $F_t = 1$
- Control-input model $B_t = 1$
- State transition uncertainty w_t characterized by $Q_t = 0.5$

Kalman Filter Example - State Transition Probability

Next, we define our state transition probability

- Remember the state transition probability in a Kalman filter

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

- State transition model $F_t = 1$
- Control-input model $B_t = 1$
- State transition uncertainty w_t characterized by $Q_t = 0.5$
- So we end up with the following:

$$x_t = 1x_{t-1} + 1u_t + w_t$$

Kalman Filter Example - Measurement Probability

Finally, we define our measurement probability

- Remember the measurement probability in a Kalman filter

$$z_t = H_t x_t + v_t$$

Kalman Filter Example - Measurement Probability

Finally, we define our measurement probability

- Remember the measurement probability in a Kalman filter

$$z_t = H_t x_t + v_t$$

- Observation model H_t

Kalman Filter Example - Measurement Probability

Finally, we define our measurement probability

- Remember the measurement probability in a Kalman filter

$$z_t = H_t x_t + v_t$$

- Observation model $H_t = 1$

Kalman Filter Example - Measurement Probability

Finally, we define our measurement probability

- Remember the measurement probability in a Kalman filter

$$z_t = H_t x_t + v_t$$

- Observation model $H_t = 1$
- Observation model noise v_t

Kalman Filter Example - Measurement Probability

Finally, we define our measurement probability

- Remember the measurement probability in a Kalman filter

$$z_t = H_t x_t + v_t$$

- Observation model $H_t = 1$
- Observation model noise v_t characterized by R_t

Kalman Filter Example - Measurement Probability

Finally, we define our measurement probability

- Remember the measurement probability in a Kalman filter

$$z_t = H_t x_t + v_t$$

- Observation model $H_t = 1$
- Observation model noise v_t characterized by $R_t = 0.5$

Kalman Filter Example - Measurement Probability

Finally, we define our measurement probability

- Remember the measurement probability in a Kalman filter

$$z_t = H_t x_t + v_t$$

- Observation model $H_t = 1$
- Observation model noise v_t characterized by $R_t = 0.5$
- So we end up with the following:

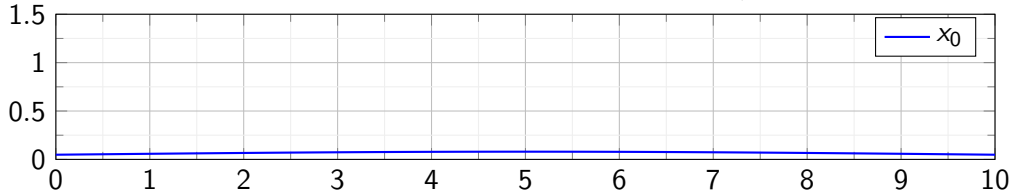
$$z_t = 1 x_t + v_t$$

Kalman Filter Example - Iteration 1 - Prediction

- We initialize our filter with $\mu_0 = 5$ and $\Sigma_0 = 5$
- We do not move in this first step $\rightarrow u_1$

Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

1. $\bar{\mu}_t = F_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + Q_t$
3. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t (z_t - H_t \bar{\mu}_t)$
5. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$
6. **return** μ_t, Σ_t

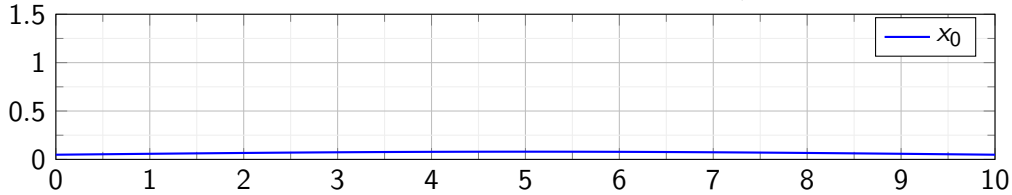


Kalman Filter Example - Iteration 1 - Prediction

- We initialize our filter with $\mu_0 = 5$ and $\Sigma_0 = 5$
- We do not move in this first step $\rightarrow u_1 = 0$

Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

1. $\bar{\mu}_t = F_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + Q_t$
3. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t (z_t - H_t \bar{\mu}_t)$
5. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$
6. **return** μ_t, Σ_t



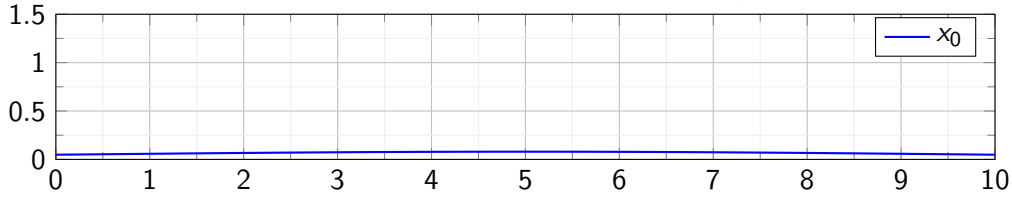
Kalman Filter Example - Iteration 1 - Prediction

- We initialize our filter with $\mu_0 = 5$ and $\Sigma_0 = 5$
- We do not move in this first step $\rightarrow u_1 = 0$

1. $\bar{\mu_1} = F_1\mu_0 + B_1u_1 = 1 * 5 + 1 * 0 = 5$

Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

1. $\bar{\mu_t} = F_t\mu_{t-1} + B_tu_t$
2. $\bar{\Sigma_t} = F_t\Sigma_{t-1}F_t^T + Q_t$
3. $K_t = \bar{\Sigma_t}H_t^T(H_t\bar{\Sigma_t}H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu_t} + K_t(z_t - H_t\bar{\mu_t})$
5. $\Sigma_t = (I - K_tH_t)\bar{\Sigma_t}$
6. **return** μ_t, Σ_t



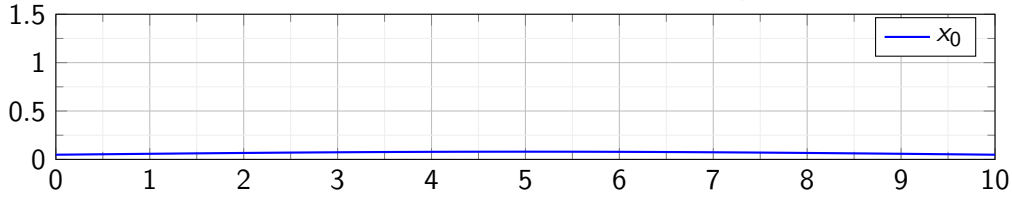
Kalman Filter Example - Iteration 1 - Prediction

- We initialize our filter with $\mu_0 = 5$ and $\Sigma_0 = 5$
- We do not move in this first step $\rightarrow u_1 = 0$

1. $\bar{\mu_1} = F_1\mu_0 + B_1u_1 = 1 * 5 + 1 * 0 = 5$
2. $\bar{\Sigma_1} = F_1\Sigma_0F_1^T + Q_1 = 1 * 5 * 1 + 0.5 = 5.5$

Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

1. $\bar{\mu_t} = F_t\mu_{t-1} + B_tu_t$
2. $\bar{\Sigma_t} = F_t\Sigma_{t-1}F_t^T + Q_t$
3. $K_t = \bar{\Sigma_t}H_t^T(H_t\bar{\Sigma_t}H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu_t} + K_t(z_t - H_t\bar{\mu_t})$
5. $\Sigma_t = (I - K_tH_t)\bar{\Sigma_t}$
6. **return** μ_t, Σ_t



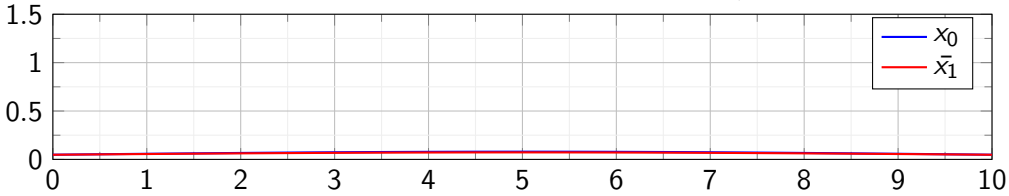
Kalman Filter Example - Iteration 1 - Prediction

- We initialize our filter with $\mu_0 = 5$ and $\Sigma_0 = 5$
- We do not move in this first step $\rightarrow u_1 = 0$

1. $\bar{\mu_1} = F_1\mu_0 + B_1u_1 = 1 * 5 + 1 * 0 = 5$
2. $\bar{\Sigma_1} = F_1\Sigma_0F_1^T + Q_1 = 1 * 5 * 1 + 0.5 = 5.5$

Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

1. $\bar{\mu_t} = F_t\mu_{t-1} + B_tu_t$
2. $\bar{\Sigma_t} = F_t\Sigma_{t-1}F_t^T + Q_t$
3. $K_t = \bar{\Sigma_t}H_t^T(H_t\bar{\Sigma_t}H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu_t} + K_t(z_t - H_t\bar{\mu_t})$
5. $\Sigma_t = (I - K_tH_t)\bar{\Sigma_t}$
6. **return** μ_t, Σ_t



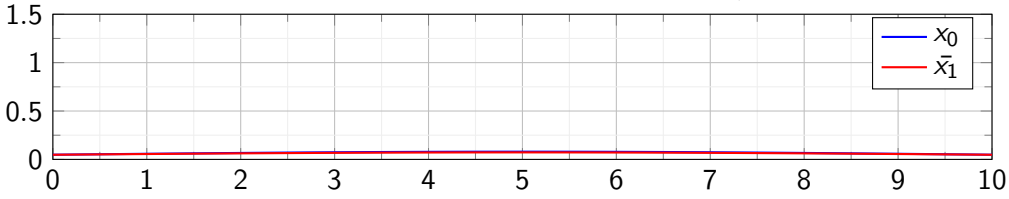
Kalman Filter Example - Iteration 1 - Prediction

- We initialize our filter with $\mu_0 = 5$ and $\Sigma_0 = 5$
- We do not move in this first step $\rightarrow u_1 = 0$

1. $\bar{\mu_1} = F_1\mu_0 + B_1u_1 = 1 * 5 + 1 * 0 = 5$
2. $\bar{\Sigma_1} = F_1\Sigma_0F_1^T + Q_1 = 1 * 5 * 1 + 0.5 = 5.5$
3. $K_1 = \bar{\Sigma_1}H_1^T(H_1\bar{\Sigma_1}H_1^T + R_1)^{-1}$
   $= 5.5 * (5.5 + 0.5)^{-1} = 5.5 * 0.1\bar{6} = 0.91\bar{6}$

Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

1. $\bar{\mu_t} = F_t\mu_{t-1} + B_tu_t$
2. $\bar{\Sigma_t} = F_t\Sigma_{t-1}F_t^T + Q_t$
3. $K_t = \bar{\Sigma_t}H_t^T(H_t\bar{\Sigma_t}H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu_t} + K_t(z_t - H_t\bar{\mu_t})$
5. $\Sigma_t = (I - K_tH_t)\bar{\Sigma_t}$
6. **return** μ_t, Σ_t

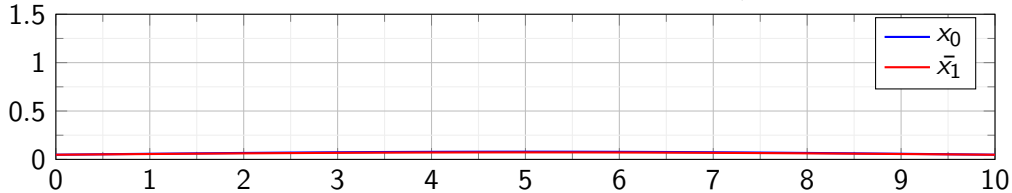


Kalman Filter Example - Iteration 1 - Correction

- We take our first measurement $z_1 = 4.5$

Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

1. $\bar{\mu}_t = F_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + Q_t$
3. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t (z_t - H_t \bar{\mu}_t)$
5. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$
6. **return** μ_t, Σ_t



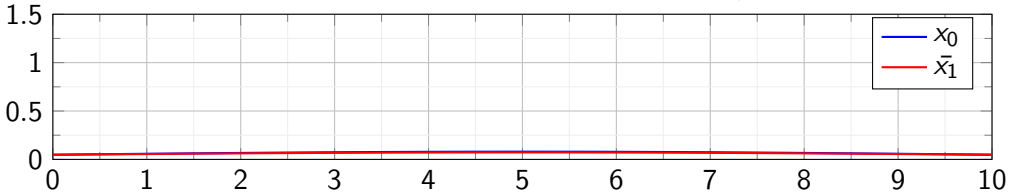
Kalman Filter Example - Iteration 1 - Correction

- We take our first measurement $z_1 = 4.5$

4. $\mu_1 = \bar{\mu_1} + K_1(z_1 - H_1\bar{\mu_1}) = 5 + 0.91\bar{6}(4.5 - 5)$
   $= 4.541\bar{6}$

Algorithm Kalman_Filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

1. $\bar{\mu_t} = F_t\mu_{t-1} + B_t u_t$
2. $\bar{\Sigma_t} = F_t\Sigma_{t-1}F_t^T + Q_t$
3. $K_t = \bar{\Sigma_t}H_t^T(H_t\bar{\Sigma_t}H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu_t} + K_t(z_t - H_t\bar{\mu_t})$
5. $\Sigma_t = (I - K_tH_t)\bar{\Sigma_t}$
6. **return** μ_t, Σ_t



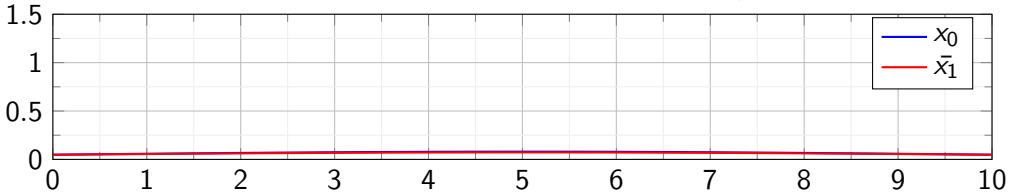
Kalman Filter Example - Iteration 1 - Correction

- We take our first measurement $z_1 = 4.5$

4. $\mu_1 = \bar{\mu_1} + K_1(z_1 - H_1\bar{\mu_1}) = 5 + 0.91\bar{6}(4.5 - 5)$
   $= 4.541\bar{6}$
5. $\Sigma_1 = (I - K_1H_1)\bar{\Sigma_1} = (1 - 0.91\bar{6} * 1) * 5.5$
   $= 0.458\bar{3}$

Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

1. $\bar{\mu_t} = F_t\mu_{t-1} + B_tu_t$
2. $\bar{\Sigma_t} = F_t\Sigma_{t-1}F_t^T + Q_t$
3. $K_t = \bar{\Sigma_t}H_t^T(H_t\bar{\Sigma_t}H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu_t} + K_t(z_t - H_t\bar{\mu_t})$
5. $\Sigma_t = (I - K_tH_t)\bar{\Sigma_t}$
6. **return** μ_t, Σ_t



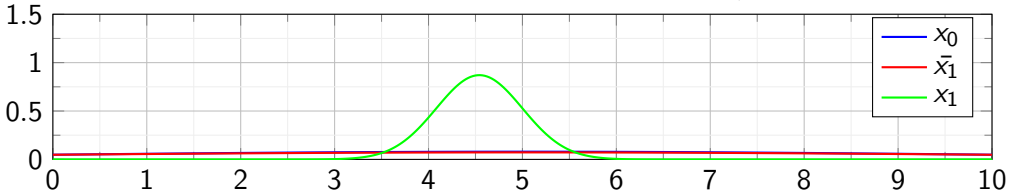
Kalman Filter Example - Iteration 1 - Correction

- We take our first measurement $z_1 = 4.5$

4. $\mu_1 = \bar{\mu_1} + K_1(z_1 - H_1\bar{\mu_1}) = 5 + 0.91\bar{6}(4.5 - 5)$
   $= 4.541\bar{6}$
5. $\Sigma_1 = (I - K_1H_1)\bar{\Sigma_1} = (1 - 0.91\bar{6} * 1) * 5.5$
   $= 0.458\bar{3}$

Algorithm Kalman_Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):

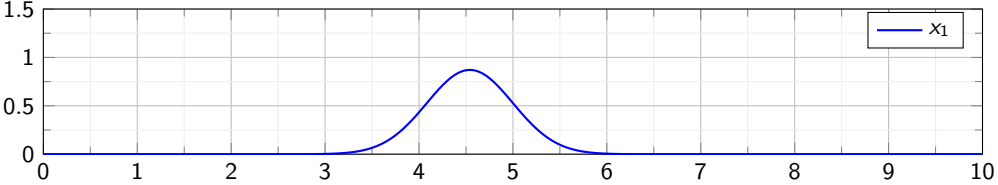
1. $\bar{\mu_t} = F_t\mu_{t-1} + B_tu_t$
2. $\bar{\Sigma_t} = F_t\Sigma_{t-1}F_t^T + Q_t$
3. $K_t = \bar{\Sigma_t}H_t^T(H_t\bar{\Sigma_t}H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu_t} + K_t(z_t - H_t\bar{\mu_t})$
5. $\Sigma_t = (I - K_tH_t)\bar{\Sigma_t}$
6. **return** μ_t, Σ_t



Kalman Filter Example - Iteration 2 - No Movement

Algorithm Kalman_Filter(μ_1, Σ_1, u_2, z_2):

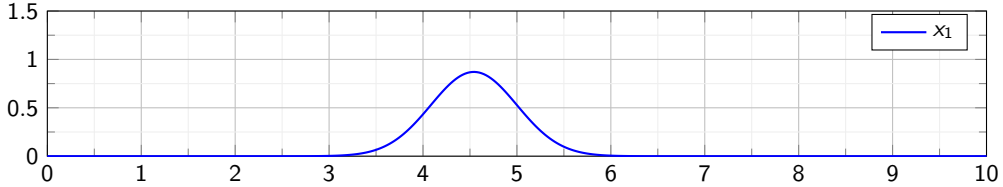
1. $\bar{\mu_2} = F_2\mu_1 + B_2u_2$
2. $\bar{\Sigma_2} = F_2\Sigma_1F_2^T + Q_2$
3. $K_2 = \bar{\Sigma_2}H_2^T(H_2\bar{\Sigma_2}H_2^T + R_2)^{-1}$
4. $\mu_2 = \bar{\mu_2} + K_2(z_2 - H_2\bar{\mu_2})$
5. $\Sigma_2 = (I - K_2H_2)\bar{\Sigma_2}$
6. **return** μ_2, Σ_2



Kalman Filter Example - Iteration 2 - No Movement

Algorithm Kalman_Filter($\mu_1 = 4.5416$, $\Sigma_1 = 0.4583$, $u_2 = 0$, $z_2 = 5.2$):

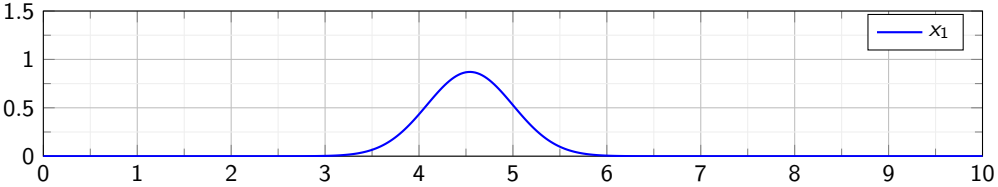
1. $\bar{\mu_2} = F_2\mu_1 + B_2u_2$
2. $\bar{\Sigma_2} = F_2\Sigma_1F_2^T + Q_2$
3. $K_2 = \bar{\Sigma_2}H_2^T(H_2\bar{\Sigma_2}H_2^T + R_2)^{-1}$
4. $\mu_2 = \bar{\mu_2} + K_2(z_2 - H_2\bar{\mu_2})$
5. $\Sigma_2 = (I - K_2H_2)\bar{\Sigma_2}$
6. **return** μ_2, Σ_2



Kalman Filter Example - Iteration 2 - No Movement

Algorithm Kalman_Filter($\mu_1 = 4.5416$, $\Sigma_1 = 0.4583$, $u_2 = 0$, $z_2 = 5.2$):

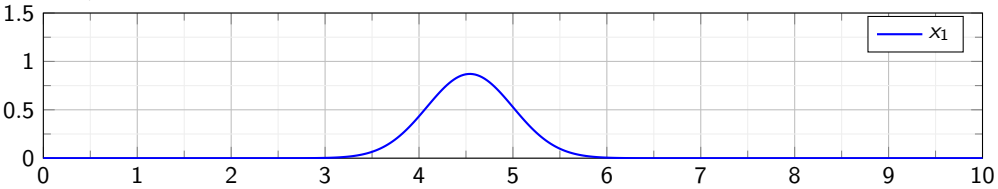
1. $\bar{\mu_2} = F_2\mu_1 + B_2u_2 = 1 * 4.5416 + 1 * 0 = 4.5416$
2. $\bar{\Sigma_2} = F_2\Sigma_1F_2^T + Q_2$
3. $K_2 = \bar{\Sigma_2}H_2^T(H_2\bar{\Sigma_2}H_2^T + R_2)^{-1}$
4. $\mu_2 = \bar{\mu_2} + K_2(z_2 - H_2\bar{\mu_2})$
5. $\Sigma_2 = (I - K_2H_2)\bar{\Sigma_2}$
6. **return** μ_2, Σ_2



Kalman Filter Example - Iteration 2 - No Movement

Algorithm Kalman_Filter($\mu_1 = 4.5416$, $\Sigma_1 = 0.4583$, $u_2 = 0$, $z_2 = 5.2$):

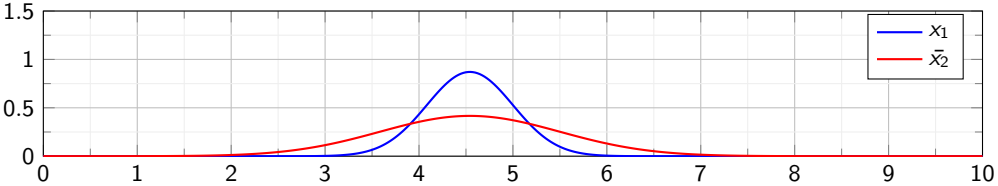
1. $\bar{\mu_2} = F_2\mu_1 + B_2u_2 = 1 * 4.5416 + 1 * 0 = 4.5416$
2. $\bar{\Sigma_2} = F_2\Sigma_1F_2^T + Q_2 = 1 * 0.4583 * 1 + 0.5 = 0.9583$
3. $K_2 = \bar{\Sigma_2}H_2^T(H_2\bar{\Sigma_2}H_2^T + R_2)^{-1}$
4. $\mu_2 = \bar{\mu_2} + K_2(z_2 - H_2\bar{\mu_2})$
5. $\Sigma_2 = (I - K_2H_2)\bar{\Sigma_2}$
6. **return** μ_2, Σ_2



Kalman Filter Example - Iteration 2 - No Movement

Algorithm Kalman_Filter($\mu_1 = 4.5416$, $\Sigma_1 = 0.4583$, $u_2 = 0$, $z_2 = 5.2$):

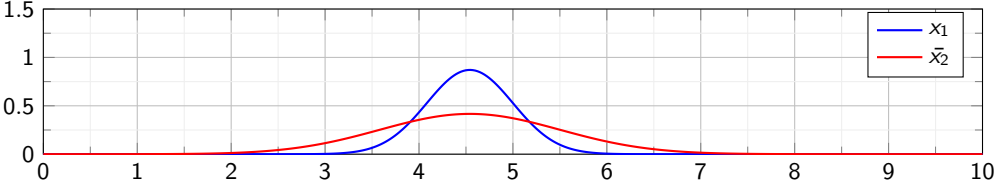
1. $\bar{\mu_2} = F_2\mu_1 + B_2u_2 = 1 * 4.5416 + 1 * 0 = 4.5416$
2. $\bar{\Sigma_2} = F_2\Sigma_1F_2^T + Q_2 = 1 * 0.4583 * 1 + 0.5 = 0.9583$
3. $K_2 = \bar{\Sigma_2}H_2^T(H_2\bar{\Sigma_2}H_2^T + R_2)^{-1}$
4. $\mu_2 = \bar{\mu_2} + K_2(z_2 - H_2\bar{\mu_2})$
5. $\Sigma_2 = (I - K_2H_2)\bar{\Sigma_2}$
6. **return** μ_2, Σ_2



Kalman Filter Example - Iteration 2 - No Movement

Algorithm Kalman_Filter($\mu_1 = 4.5416$, $\Sigma_1 = 0.4583$, $u_2 = 0$, $z_2 = 5.2$):

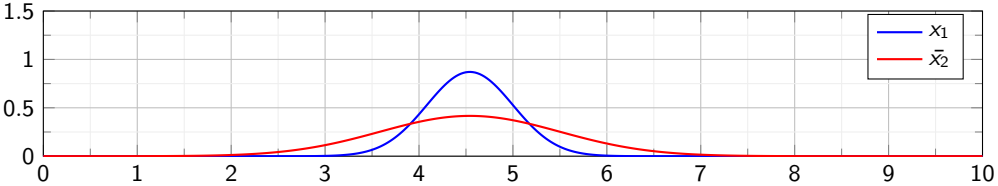
1. $\bar{\mu_2} = F_2\mu_1 + B_2u_2 = 1 * 4.5416 + 1 * 0 = 4.5416$
2. $\bar{\Sigma_2} = F_2\Sigma_1F_2^T + Q_2 = 1 * 0.4583 * 1 + 0.5 = 0.9583$
3. $K_2 = \bar{\Sigma_2}H_2^T(H_2\bar{\Sigma_2}H_2^T + R_2)^{-1} = 0.9583 * 1 * (1 * 0.9583 * 1 + 0.5)^{-1} = 0.6571$
4. $\mu_2 = \bar{\mu_2} + K_2(z_2 - H_2\bar{\mu_2})$
5. $\Sigma_2 = (I - K_2H_2)\bar{\Sigma_2}$
6. **return** μ_2, Σ_2



Kalman Filter Example - Iteration 2 - No Movement

Algorithm Kalman_Filter($\mu_1 = 4.5416$, $\Sigma_1 = 0.4583$, $u_2 = 0$, $z_2 = 5.2$):

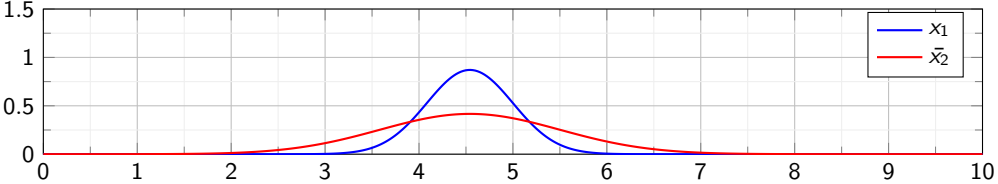
1. $\bar{\mu_2} = F_2\mu_1 + B_2u_2 = 1 * 4.5416 + 1 * 0 = 4.5416$
2. $\bar{\Sigma_2} = F_2\Sigma_1F_2^T + Q_2 = 1 * 0.4583 * 1 + 0.5 = 0.9583$
3. $K_2 = \bar{\Sigma_2}H_2^T(H_2\bar{\Sigma_2}H_2^T + R_2)^{-1} = 0.9583 * 1 * (1 * 0.9583 * 1 + 0.5)^{-1} = 0.6571$
4. $\mu_2 = \bar{\mu_2} + K_2(z_2 - H_2\bar{\mu_2}) = 4.5416 + 0.6571(5.2 - 1 * 4.5416) = 4.9743$
5. $\Sigma_2 = (I - K_2H_2)\bar{\Sigma_2}$
6. **return** μ_2, Σ_2



Kalman Filter Example - Iteration 2 - No Movement

Algorithm Kalman_Filter($\mu_1 = 4.5416$, $\Sigma_1 = 0.4583$, $u_2 = 0$, $z_2 = 5.2$):

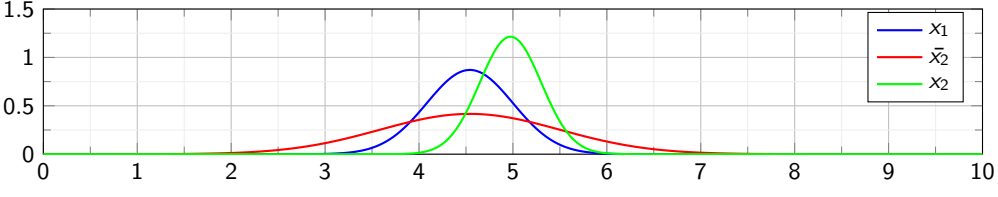
1. $\bar{\mu_2} = F_2\mu_1 + B_2u_2 = 1 * 4.5416 + 1 * 0 = 4.5416$
2. $\bar{\Sigma_2} = F_2\Sigma_1F_2^T + Q_2 = 1 * 0.4583 * 1 + 0.5 = 0.9583$
3. $K_2 = \bar{\Sigma_2}H_2^T(H_2\bar{\Sigma_2}H_2^T + R_2)^{-1} = 0.9583 * 1 * (1 * 0.9583 * 1 + 0.5)^{-1} = 0.6571$
4. $\mu_2 = \bar{\mu_2} + K_2(z_2 - H_2\bar{\mu_2}) = 4.5416 + 0.6571(5.2 - 1 * 4.5416) = 4.9743$
5. $\Sigma_2 = (I - K_2H_2)\bar{\Sigma_2} = (1 - 0.6571 * 1) * 0.9583 = 0.3286$
6. **return** μ_2, Σ_2



Kalman Filter Example - Iteration 2 - No Movement

Algorithm Kalman_Filter($\mu_1 = 4.5416$, $\Sigma_1 = 0.4583$, $u_2 = 0$, $z_2 = 5.2$):

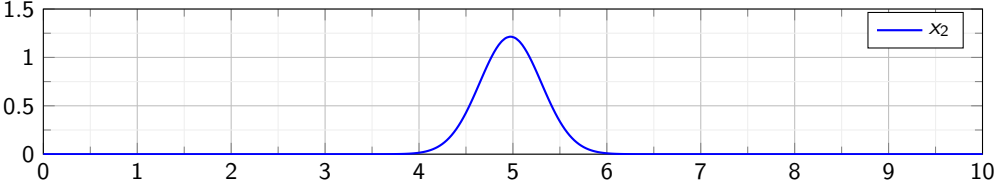
1. $\bar{\mu_2} = F_2\mu_1 + B_2u_2 = 1 * 4.5416 + 1 * 0 = 4.5416$
2. $\bar{\Sigma_2} = F_2\Sigma_1F_2^T + Q_2 = 1 * 0.4583 * 1 + 0.5 = 0.9583$
3. $K_2 = \bar{\Sigma_2}H_2^T(H_2\bar{\Sigma_2}H_2^T + R_2)^{-1} = 0.9583 * 1 * (1 * 0.9583 * 1 + 0.5)^{-1} = 0.6571$
4. $\mu_2 = \bar{\mu_2} + K_2(z_2 - H_2\bar{\mu_2}) = 4.5416 + 0.6571(5.2 - 1 * 4.5416) = 4.9743$
5. $\Sigma_2 = (I - K_2H_2)\bar{\Sigma_2} = (1 - 0.6571 * 1) * 0.9583 = 0.3286$
6. **return** μ_2, Σ_2



Kalman Filter Example - Iteration 3 - Movement

Algorithm Kalman_Filter(μ_2, Σ_2, u_3, z_3):

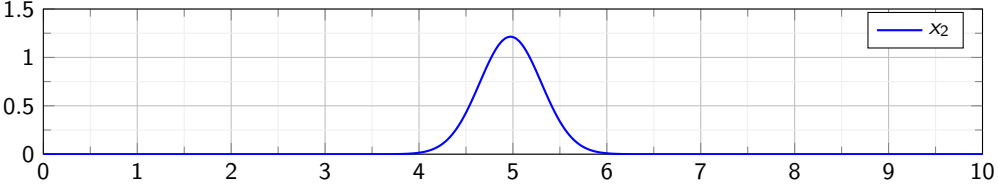
1. $\bar{\mu_3} = F_3\mu_2 + B_3u_3$
2. $\bar{\Sigma_3} = F_3\Sigma_2F_3^T + Q_3$
3. $K_3 = \bar{\Sigma_3}H_3^T(H_3\bar{\Sigma_2}H_3^T + R_3)^{-1}$
4. $\mu_3 = \bar{\mu_3} + K_3(z_3 - H_3\bar{\mu_3})$
5. $\Sigma_3 = (I - K_3H_3)\bar{\Sigma_3}$
6. **return** μ_3, Σ_3



Kalman Filter Example - Iteration 3 - Movement

Algorithm Kalman_Filter($\mu_2 = 4.9743$, $\Sigma_2 = 0.3286$, $u_3 = 1$, $z_3 = 5.9$):

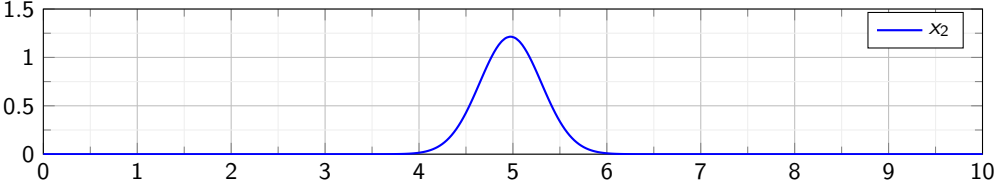
1. $\bar{\mu_3} = F_3\mu_2 + B_3u_3$
2. $\bar{\Sigma_3} = F_3\Sigma_2F_3^T + Q_3$
3. $K_3 = \bar{\Sigma_3}H_3^T(H_3\bar{\Sigma_2}H_3^T + R_3)^{-1}$
4. $\mu_3 = \bar{\mu_3} + K_3(z_3 - H_3\bar{\mu_3})$
5. $\Sigma_3 = (I - K_3H_3)\bar{\Sigma_3}$
6. **return** μ_3, Σ_3



Kalman Filter Example - Iteration 3 - Movement

Algorithm Kalman_Filter($\mu_2 = 4.9743$, $\Sigma_2 = 0.3286$, $u_3 = 1$, $z_3 = 5.9$):

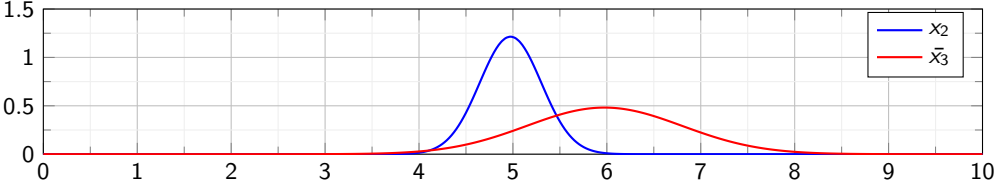
1. $\bar{\mu_3} = F_3\mu_2 + B_3u_3 = 1 * 4.9743 + 1 * 1 = 5.9743$
2. $\bar{\Sigma_3} = F_3\Sigma_2F_3^T + Q_3 = 1 * 0.3286 * 1 + 0.5 = 0.8286$
3. $K_3 = \bar{\Sigma_3}H_3^T(H_3\bar{\Sigma_2}H_3^T + R_3)^{-1}$
4. $\mu_3 = \bar{\mu_3} + K_3(z_3 - H_3\bar{\mu_3})$
5. $\Sigma_3 = (I - K_3H_3)\bar{\Sigma_3}$
6. **return** μ_3, Σ_3



Kalman Filter Example - Iteration 3 - Movement

Algorithm Kalman_Filter($\mu_2 = 4.9743$, $\Sigma_2 = 0.3286$, $u_3 = 1$, $z_3 = 5.9$):

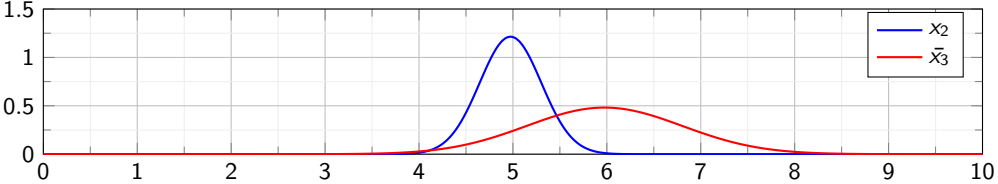
1. $\bar{\mu_3} = F_3\mu_2 + B_3u_3 = 1 * 4.9743 + 1 * 1 = 5.9743$
2. $\bar{\Sigma_3} = F_3\Sigma_2F_3^T + Q_3 = 1 * 0.3286 * 1 + 0.5 = 0.8286$
3. $K_3 = \bar{\Sigma_3}H_3^T(H_3\bar{\Sigma_2}H_3^T + R_3)^{-1}$
4. $\mu_3 = \bar{\mu_3} + K_3(z_3 - H_3\bar{\mu_3})$
5. $\Sigma_3 = (I - K_3H_3)\bar{\Sigma_3}$
6. **return** μ_3, Σ_3



Kalman Filter Example - Iteration 3 - Movement

Algorithm Kalman_Filter($\mu_2 = 4.9743$, $\Sigma_2 = 0.3286$, $u_3 = 1$, $z_3 = 5.9$):

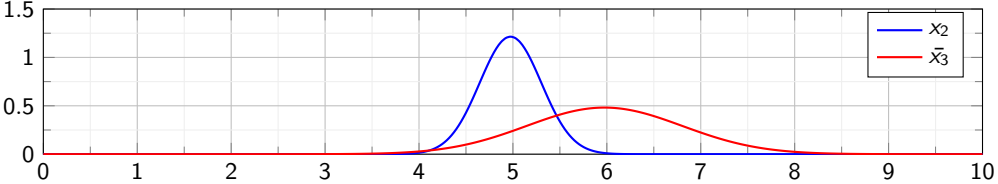
1. $\bar{\mu_3} = F_3\mu_2 + B_3u_3 = 1 * 4.9743 + 1 * 1 = 5.9743$
2. $\bar{\Sigma_3} = F_3\Sigma_2F_3^T + Q_3 = 1 * 0.3286 * 1 + 0.5 = 0.8286$
3. $K_3 = \bar{\Sigma_3}H_3^T(H_3\bar{\Sigma_2}H_3^T + R_3)^{-1} = 0.8286 * 1 * (1 * 0.8286 * 1 + 0.5)^{-1} = 0.6237$
4. $\mu_3 = \bar{\mu_3} + K_3(z_3 - H_3\bar{\mu_3})$
5. $\Sigma_3 = (I - K_3H_3)\bar{\Sigma_3}$
6. **return** μ_3, Σ_3



Kalman Filter Example - Iteration 3 - Movement

Algorithm Kalman_Filter($\mu_2 = 4.9743$, $\Sigma_2 = 0.3286$, $u_3 = 1$, $z_3 = 5.9$):

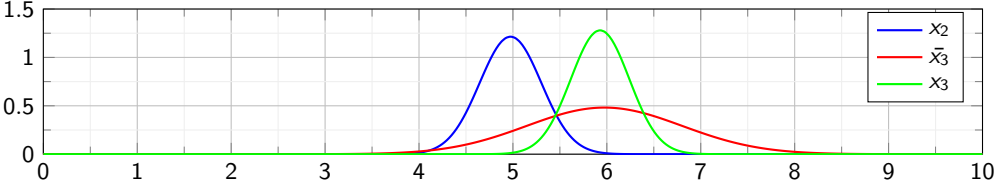
1. $\bar{\mu_3} = F_3\mu_2 + B_3u_3 = 1 * 4.9743 + 1 * 1 = 5.9743$
2. $\bar{\Sigma_3} = F_3\Sigma_2F_3^T + Q_3 = 1 * 0.3286 * 1 + 0.5 = 0.8286$
3. $K_3 = \bar{\Sigma_3}H_3^T(H_3\bar{\Sigma_2}H_3^T + R_3)^{-1} = 0.8286 * 1 * (1 * 0.8286 * 1 + 0.5)^{-1} = 0.6237$
4. $\mu_3 = \bar{\mu_3} + K_3(z_3 - H_3\bar{\mu_3}) = 5.9743 + 0.6237(5.9 - 1 * 5.9743) = 5.9280$
5. $\Sigma_3 = (I - K_3H_3)\bar{\Sigma_3} = (1 - 0.6237 * 1) * 0.8286 = 0.31180$
6. **return** μ_3, Σ_3



Kalman Filter Example - Iteration 3 - Movement

Algorithm Kalman_Filter($\mu_2 = 4.9743$, $\Sigma_2 = 0.3286$, $u_3 = 1$, $z_3 = 5.9$):

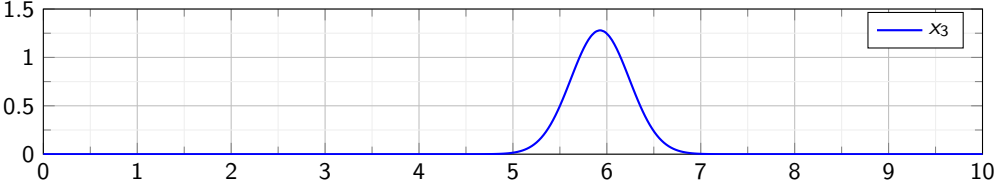
1. $\bar{\mu_3} = F_3\mu_2 + B_3u_3 = 1 * 4.9743 + 1 * 1 = 5.9743$
2. $\bar{\Sigma_3} = F_3\Sigma_2F_3^T + Q_3 = 1 * 0.3286 * 1 + 0.5 = 0.8286$
3. $K_3 = \bar{\Sigma_3}H_3^T(H_3\bar{\Sigma_2}H_3^T + R_3)^{-1} = 0.8286 * 1 * (1 * 0.8286 * 1 + 0.5)^{-1} = 0.6237$
4. $\mu_3 = \bar{\mu_3} + K_3(z_3 - H_3\bar{\mu_3}) = 5.9743 + 0.6237(5.9 - 1 * 5.9743) = 5.9280$
5. $\Sigma_3 = (I - K_3H_3)\bar{\Sigma_3} = (1 - 0.6237 * 1) * 0.8286 = 0.31180$
6. **return** μ_3, Σ_3



Kalman Filter Example - Iteration 4 - No Measurement

Algorithm Kalman_Filter(μ_3, Σ_3, u_4, z_4):

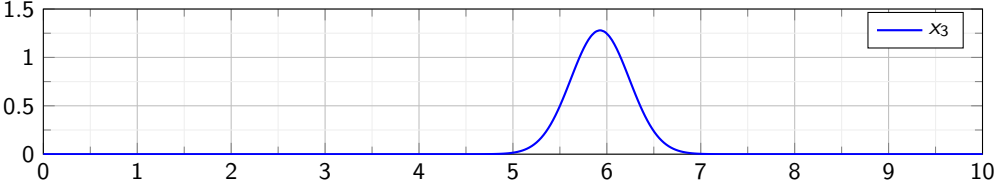
1. $\bar{\mu_4} = F_4\mu_3 + B_4u_4$
2. $\bar{\Sigma_4} = F_4\Sigma_3F_4^T + Q_4$
3. $K_4 = \bar{\Sigma_4}H_4^T(H_4\bar{\Sigma_4}H_4^T + R_4)^{-1}$
4. $\mu_4 = \bar{\mu_4} + K_4(z_4 - H_4\bar{\mu_2})$
5. $\Sigma_4 = (I - K_4H_4)\bar{\Sigma_4}$
6. **return** μ_4, Σ_4



Kalman Filter Example - Iteration 4 - No Measurement

Algorithm Kalman_Filter($\mu_3 = 5.9280$, $\Sigma_3 = 0.31180$, $u_4 = 1$, $z_4 = XX$):

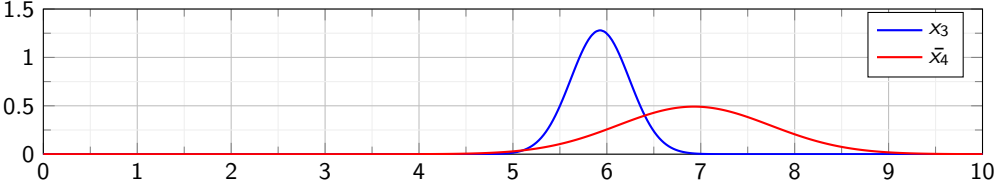
1. $\bar{\mu_4} = F_4\mu_3 + B_4u_4$
2. $\bar{\Sigma_4} = F_4\Sigma_3F_4^T + Q_4$
3. $K_4 = \bar{\Sigma_4}H_4^T(H_4\bar{\Sigma_4}H_4^T + R_4)^{-1}$
4. $\mu_4 = \bar{\mu_4} + K_4(z_4 - H_4\bar{\mu_2})$
5. $\Sigma_4 = (I - K_4H_4)\bar{\Sigma_4}$
6. **return** μ_4, Σ_4



Kalman Filter Example - Iteration 4 - No Measurement

Algorithm Kalman_Filter($\mu_3 = 5.9280$, $\Sigma_3 = 0.31180$, $u_4 = 1$, $z_4 = XX$):

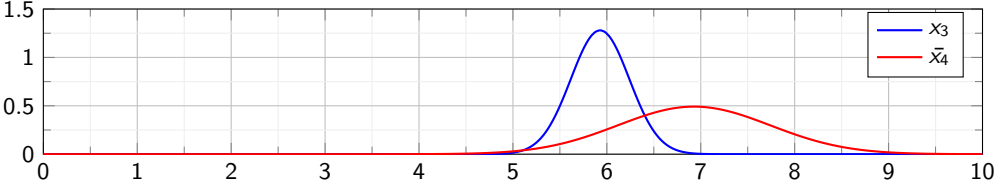
1. $\bar{\mu_4} = F_4\mu_3 + B_4u_4 = 1 * 5.9280 + 1 * 1 = 6.9280$
2. $\bar{\Sigma_4} = F_4\Sigma_3F_4^T + Q_4 = 1 * 0.31180 * 1 + 0.5 = 0.81180$
3. $K_4 = \bar{\Sigma_4}H_4^T(H_4\bar{\Sigma_4}H_4^T + R_4)^{-1}$
4. $\mu_4 = \bar{\mu_4} + K_4(z_4 - H_4\bar{\mu_2})$
5. $\Sigma_4 = (I - K_4H_4)\bar{\Sigma_4}$
6. **return** μ_4, Σ_4



Kalman Filter Example - Iteration 4 - No Measurement

Algorithm Kalman_Filter($\mu_3 = 5.9280$, $\Sigma_3 = 0.31180$, $u_4 = 1$, $z_4 = XX$):

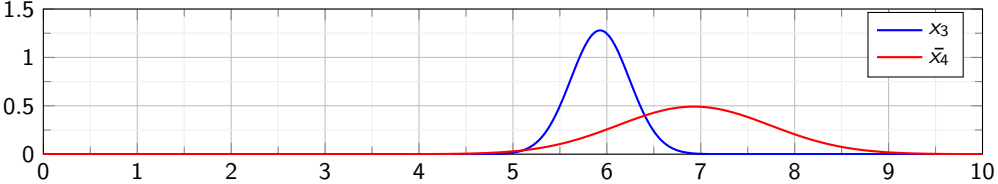
1. $\bar{\mu_4} = F_4\mu_3 + B_4u_4 = 1 * 5.9280 + 1 * 1 = 6.9280$
2. $\bar{\Sigma_4} = F_4\Sigma_3F_4^T + Q_4 = 1 * 0.31180 * 1 + 0.5 = 0.81180$
3. $K_4 = \bar{\Sigma_4}H_4^T(H_4\bar{\Sigma_4}H_4^T + R_4)^{-1}$
4. $\mu_4 = \sout{\bar{\mu_4} + K_4(z_4 - H_4\bar{\mu_2})}$
5. $\Sigma_4 = \sout{(I - K_4H_4)\bar{\Sigma_4}}$
6. **return** μ_4, Σ_4



Kalman Filter Example - Iteration 4 - No Measurement

Algorithm Kalman_Filter($\mu_3 = 5.9280$, $\Sigma_3 = 0.31180$, $u_4 = 1$, $z_4 = XX$):

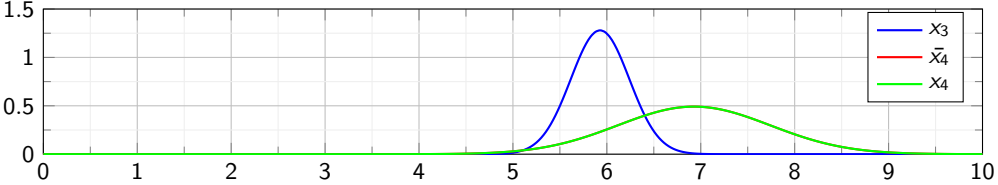
1. $\bar{\mu_4} = F_4\mu_3 + B_4u_4 = 1 * 5.9280 + 1 * 1 = 6.9280$
2. $\bar{\Sigma_4} = F_4\Sigma_3F_4^T + Q_4 = 1 * 0.31180 * 1 + 0.5 = 0.81180$
3. $K_4 = \bar{\Sigma_4}H_4^T(H_4\bar{\Sigma_4}H_4^T + R_4)^{-1}$
4. $\mu_4 = \bar{\mu_4}$
5. $\Sigma_4 = \bar{\Sigma_4}$
6. **return** μ_4, Σ_4



Kalman Filter Example - Iteration 4 - No Measurement

Algorithm Kalman_Filter($\mu_3 = 5.9280$, $\Sigma_3 = 0.31180$, $u_4 = 1$, $z_4 = XX$):

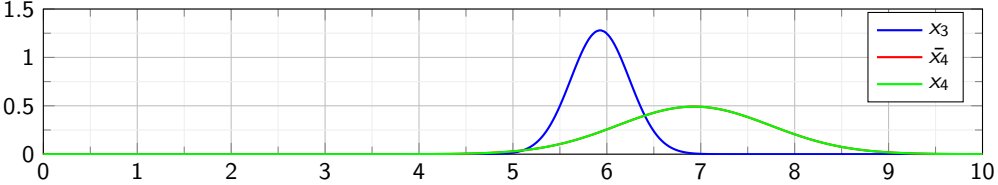
1. $\bar{\mu_4} = F_4\mu_3 + B_4u_4 = 1 * 5.9280 + 1 * 1 = 6.9280$
2. $\bar{\Sigma_4} = F_4\Sigma_3F_4^T + Q_4 = 1 * 0.31180 * 1 + 0.5 = 0.81180$
3. $K_4 = \bar{\Sigma_4}H_4^T(H_4\bar{\Sigma_4}H_4^T + R_4)^{-1}$
4. $\mu_4 = \bar{\mu_4}$
5. $\Sigma_4 = \bar{\Sigma_4}$
6. **return** μ_4, Σ_4



Kalman Filter Example - Iteration 4 - No Measurement

Algorithm Kalman_Filter($\mu_3 = 5.9280$, $\Sigma_3 = 0.31180$, $u_4 = 1$, $z_4 = XX$):

1. $\bar{\mu}_4 = F_4\mu_3 + B_4u_4 = 1 * 5.9280 + 1 * 1 = 6.9280$
2. $\bar{\Sigma}_4 = F_4\Sigma_3F_4^T + Q_4 = 1 * 0.31180 * 1 + 0.5 = 0.81180$
3. ~~$K_4 = \bar{\Sigma}_4H_4^T(H_4\bar{\Sigma}_4H_4^T + R_4)^{-1}$~~
4. $\mu_4 = \bar{\mu}_4$
5. $\Sigma_4 = \bar{\Sigma}_4$
6. **return** μ_4, Σ_4



Kalman Filter

Advantages:

- Highly efficient (prediction and correction steps in closed form)
- **Optimal for linear Gaussian systems**

The correctness of the Kalman filter crucially depends on the assumptions that the measurements are a linear function of the state and that the next state is a linear function of the current state

- Most problems in robotics are non-linear
  - State transitions and measurements are usually non-linear
  - **So the Kalman filter is not directly applicable!**

Extended Kalman Filter

The Extended Kalman filter (EKF) relaxes the *linearity* assumption

- State transition probability and measurement probability

$$\begin{aligned} x_t &= f(u_t, x_{t-1}) + \epsilon_t \\ z_t &= h(x_t) + \delta_t \end{aligned}$$

- However, the belief is no longer a Gaussian
- EKF calculates a *Gaussian approximation* to the true belief
- The approximation is determined through linearization
  - Non-linear functions f and h are approximated by linear functions that are tangent to f or h at the mean of the Gaussian
  - This makes use of their Jacobian matrices F_t and H_t

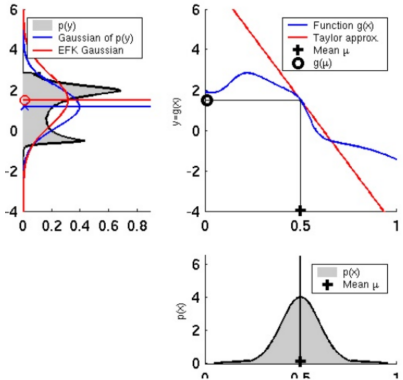
Jacobian Matrix

- The Jacobian Matrix J_f of a function $f : \mathcal{R}^n \rightarrow \mathcal{R}^m$ is the matrix of all first-order partial derivatives of a vector-valued function.

$$(J_f)_{ij} = \frac{\partial f_i}{\partial x_j}$$

$$J_f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Extended Kalman Filter



https://www.slideshare.net/correll/lecture-10-slam

Extended Kalman Filter

Algorithm Extended_Kalman_Filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

1. $\bar{\mu}_t = f(u_t, \mu_{t-1})$
2. $\bar{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + Q_t$
3. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + R_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
5. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
6. **return** μ_t, Σ_t

Extended Kalman Filter

Kalman filter vs. Extended Kalman filter

- The algorithms are quite similar and share several properties
- Most important difference concerns *state prediction* (line 1) and *measurement prediction* (line 4)
  - Linear predictions $\rightarrow$ Non-linear generalizations
- Additionally, EKF uses Jacobians F_t and H_t instead of the corresponding linear system matrices F_t, B_t and H_t

Extended Kalman Filter - Example

- Let a robot's state be characterized by $X = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$
- The robot can move forward by d meters and turn by φrad, but only turns after moving. This can be represented by $u = \begin{pmatrix} d \\ \varphi \end{pmatrix}$
- It can measure its absolute orientation θ (by IMU)
- Define the transition and the measurement model g and h and the covariance matrices of their noise terms, and compute their Jacobian Matrices F and H
- Assume some initial belief $bel(x_0)$, an action u_1, and a measurement z_1 and compute $bel(x_1)$

Extended Kalman Filter - Example

$$f(\begin{pmatrix} d \\ \varphi \end{pmatrix}, \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}) = \begin{pmatrix} x + d * cos(\theta) \\ y + d * sin(\theta) \\ \theta + \varphi \end{pmatrix}; w_t \sim \mathcal{N}(0, Q_u)$$

$$Q_u = \begin{pmatrix} 0.01 * d & 0 & 0 \\ 0 & 0.01 * d & 0 \\ 0 & 0 & 0.01 * \varphi \end{pmatrix}$$

$$h(\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}) = \theta; \delta_t \sim \mathcal{N}(0, R); R = 0.01$$

$$F_t = \begin{bmatrix} \frac{df}{dx} & \frac{df}{dy} & \frac{df}{d\theta} \end{bmatrix} = \begin{pmatrix} 1 & 0 & -d * sin(\theta) \\ 0 & 1 & d * cos(\theta) \\ 0 & 0 & 1 \end{pmatrix}$$

$$H_t = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Extended Kalman Filter - Example

$$bel(X_0) = \mathcal{N}(\mu_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \Sigma_0 = 0)$$

$$u_1 = \begin{pmatrix} 1.0 \\ 1.6 \end{pmatrix}$$

$$\overline{bel}(X_1) :$$

$$\overline{\mu_1} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1.0 * cos(0) \\ 1 * sin(0) \\ 0 + 1.6 \end{pmatrix} = \begin{pmatrix} 1.0 \\ 0 \\ 1.6 \end{pmatrix}$$

$$\overline{\Sigma_1} = F_1 * \Sigma_0 * F_1^T + Q_1 = \begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.16 \end{pmatrix}$$

Extended Kalman Filter - Example

$$K_1 = \begin{pmatrix} 0 \\ 0 \\ \frac{16}{17} \end{pmatrix}, z_1 = 2.0$$

$bel(X_1)$:

$$\mu_1 = \overline{\mu_1} + K_1(z_1 - h(\overline{\mu_1})) = \begin{pmatrix} 1.0 \\ 0 \\ 1.6 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{16}{17} \end{pmatrix} * 0.4 = \begin{pmatrix} 1.0 \\ 0 \\ 1.98 \end{pmatrix}$$

$$\Sigma_1 = (1 - K_1 * H_t) * \overline{\Sigma_1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{17} \end{pmatrix} * \overline{\Sigma_1} \approx \begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{pmatrix}$$

Extended Kalman Filter - Example

$$bel(X_0) = \mathcal{N}(\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{pmatrix})$$

$$u_1 = \begin{pmatrix}1.0\\1.6\end{pmatrix}$$

$$\overline{bel(X_1)} = \mathcal{N}(\begin{pmatrix}1.0\\0\\1.6\end{pmatrix}, \begin{pmatrix}0.01 & 0 & 0\\0 & 0.01 & 0\\0 & 0 & 0.16\end{pmatrix})$$

$$z_1 = 2.0$$

$$bel(X_1) = \mathcal{N}(\begin{pmatrix}1.0\\0.0\\1.98\end{pmatrix}, \begin{pmatrix}0.01 & 0 & 0\\0 & 0.01 & 0\\0 & 0 & 0.01\end{pmatrix})$$

Extended Kalman Filter

Advantages:

- Highly efficient
- Useful for multi-sensor fusion
- Once non-linear functions g and h are linearized, the prediction and update procedures are equivalent to those of the Kalman filter

Disadvantages:

- Not optimal $\rightarrow$ Belief is approximated
- Can diverge if non-linearities are large

EKF Localization

The *Extended Kalman filter* localization is a special case of Markov localization

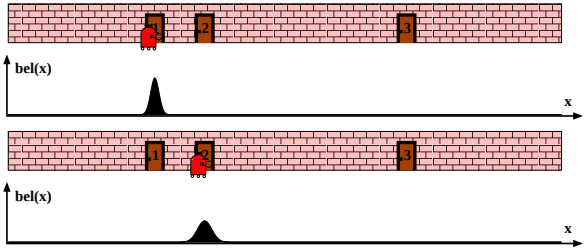
- **Assumption:** The map of the environment is represented as a collection of features

At any point in time the robot observes a vector of ranges to nearby features

- Features can be assumed to be *uniquely identifiable*

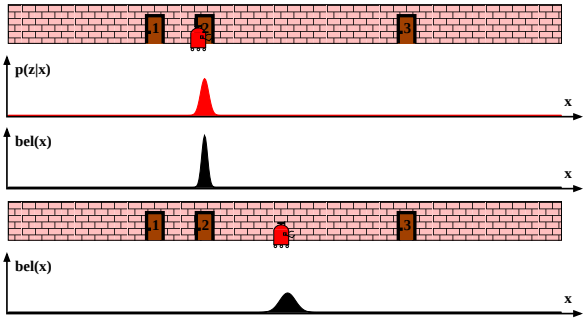
$$z_t = (z_t^1, z_t^2, \ldots, z_t^m)$$

EKF Localization



Uniquely identifiable features. Good knowledge about initial pose followed by convolution with motion model.

EKF Localization



- Belief remains Gaussian at any point in time
- If unique feature identification is not given, maximum likelihood estimation can provide correspondances

EKF

Video
https://www.youtube.com/watch?v=r0sS6ZYaCP4
Video2
https://www.youtube.com/watch?v=h548R3N49No

Unscented Kalman Filter

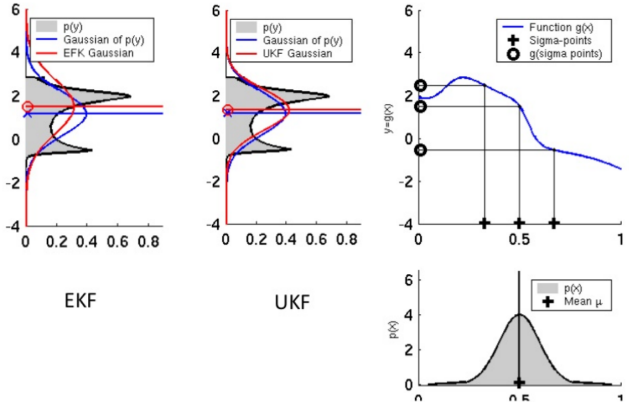
The Unscented Kalman filter (UKF) is a variant of the Kalman filter that improves the belief estimate through a stochastic linearization method: the unscented transform

- It uses a weighted statistical linear regression process

Prediction and correction steps are preceeded with a sigma-point extraction step

1. Deterministic extraction of *sigma-points* (located at the mean and along the axes of the covariance)
2. Assignment of weights to extracted points
3. Transform of points through non-linear functions f and h
4. Computation of Gaussian from weighted points

Unscented Kalman Filter



https://www.slideshare.net/correll/lecture-10-slam

Unscented Kalman Filter

- Highly efficient: Same complexity as EKF (constant factor slower in typical practical applications)
- Better linearization than EKF
- For purely linear problems belief estimate is *equal* to that generated by a Kalman filter
- For non-linear problems the estimate is *equal or better* than that generated by EKF
- UKF is a *derivative-free filter*: No Jacobians needed

- Still not optimal
- UKF Tutorial: `https://towardsdatascience.com/the-unscented-kalman-filter-anything-ekf-can-do-i-can-do-it-better-ce7c773cf88d`

KF based Localization

- EKF and UKF localization are only applicable to *pose tracking* problems
- Linearized Gaussian approaches work well only if the pose uncertainty is small
- Linearization is usually only good in close proximity to the linearization point
- EKF and UKF localization process only a subset of all information in the sensor measurement data
- On the other hand it allows the efficient integration of measurements from multiple sources

Particle Filter based Localization

- Representation of belief by random samples (particles)
- Instead of representing *parameterized distributions* one can also reason with *samples from the distribution*
- Estimation of multi-modal, non-Gaussian, non-linear processes

- Monte Carlo filter is the most popular particle based technique
- Applicable to position tracking and global localization problems
- Naive versions of the algorithm are simple to implement

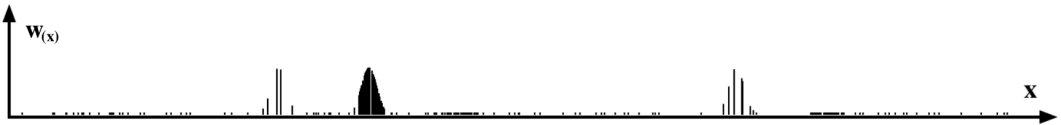
Monte Carlo Localization

- *Monte Carlo localization* (MCL) approximates the belief $bel(x_t)$ through a set of N particles χ_t

$$\chi_t = \left\{ \langle x_t^i, w_t^i \rangle \middle| x_t^i \in X_t, w_t^i \in \mathcal{R}^+ \right\}$$

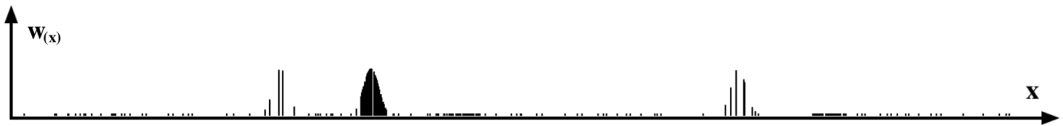
  with $i = 1 \ldots N$ and state space X_t at time t

- Each sample is assigned an *importance weight* w_t^i



Monte Carlo Localization

- Discrete approximation of a probability distribution
- More particles can represent more complex distributions
- Approximation of any distribution is possible *in theory*
- Algorithm is structurally similar to Markov localization, intertwining motion model and sensor model updates



Monte Carlo Localization

- To focus particles on *important regions* of the state space, Monte Carlo methods apply a *resampling* step
  - **Resampling**: Selection of a new set of samples χ_t ...
    - ... from elements of the old sample set χ_{t-1} ...
    - ... generating new samples if necessary
- This ensures that samples with low weights get replaced by more important samples
- It might add alternative hypotheses that were not represented
- Resampling was a major breakthrough for particle filters and made them feasible in practice

Monte Carlo Localization

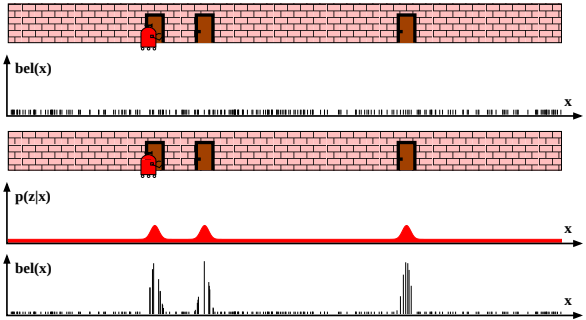
Algorithm Monte_Carlo_Localization$(\chi_{t-1},\ u_t,\ z_t,\ m)$:

1. $\bar{\chi}_t = \chi_t = \emptyset$
2. $\#update\ step$
3. **for** $n = 1$ **to** N **do**
4. $\quad x_t^{[n]} = sample\_motion\_model(u_t, x_{t-1}^{[n]})$
5. $\quad w_t^{[n]} = measurement\_model(z_t, x_t^{[n]}, m)$
6. $\quad \bar{\chi}_t = \bar{\chi}_t \cup \{\langle x_t^{[n]}, w_t^{[n]} \rangle\}$
7. **endfor**

Monte Carlo Localization

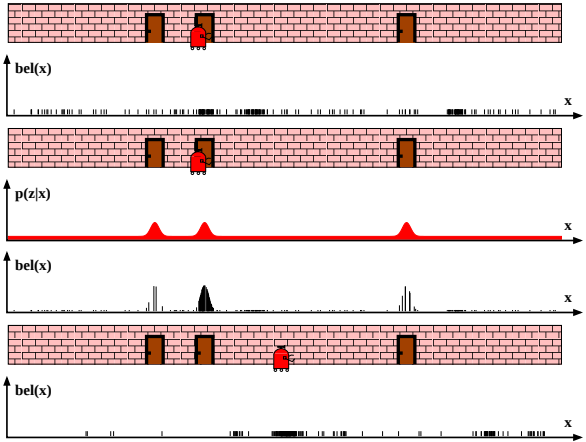
8. *#resampling step*
9. **for** $i = 1$ **to** N **do**
10. *draw* $x_t^{[i]}$ *favoring larger* $w_t^{[i]}$
11. *add* $x_t^{[i]}$ *to* χ_t
12. **endfor**
13. **return** χ_t

Monte Carlo Localization



Random initialization. Incorporation of the motion model with weighting of the samples.

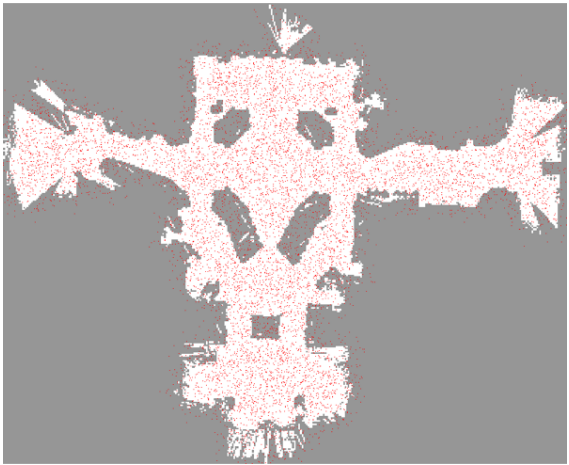
Monte Carlo Localization



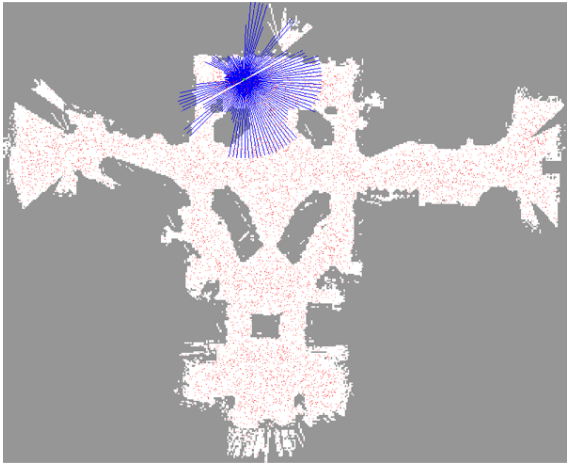
Adaptive Sample Size

- The number of considered particles can be altered online
- If the distribution of the current belief changes its complexity at runtime, the number of particles can be adjusted accordingly
- This is not easy to detect! Common attempts:
  - *Likelihood-based adaptation*:
    If measurements agree with most particles, fewer particles are needed
  - *KLD-sampling*:
    If the expected area of important regions changes, sample size can be adjusted to bound the error in terms of its KL-distance

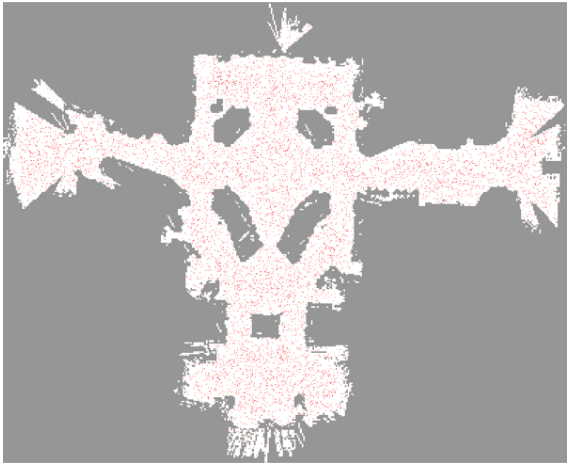
Monte Carlo Localization



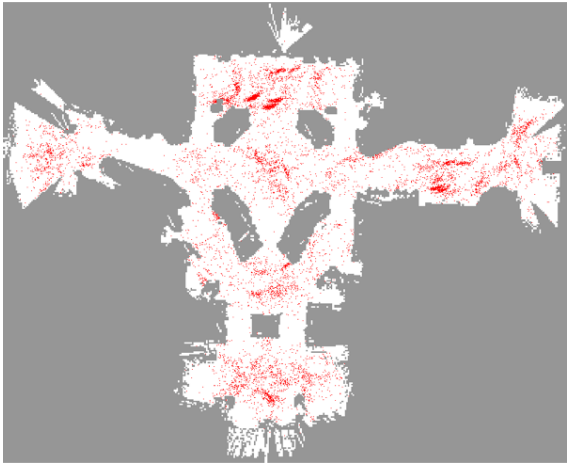
Monte Carlo Localization



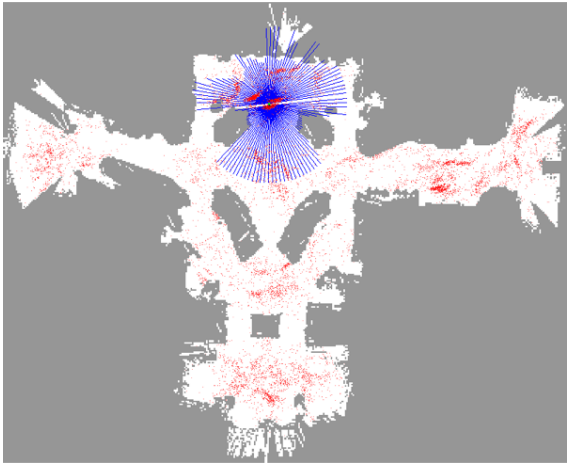
Monte Carlo Localization



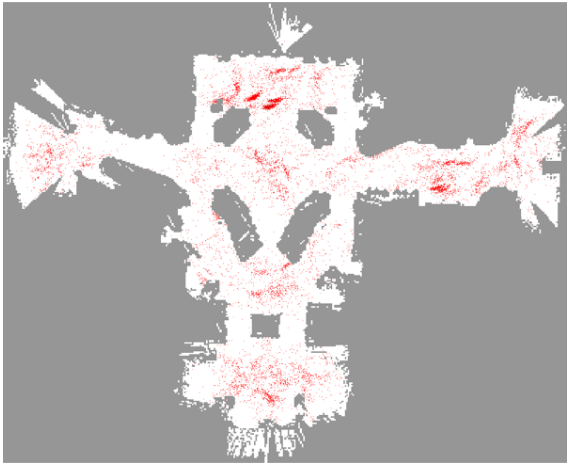
Monte Carlo Localization



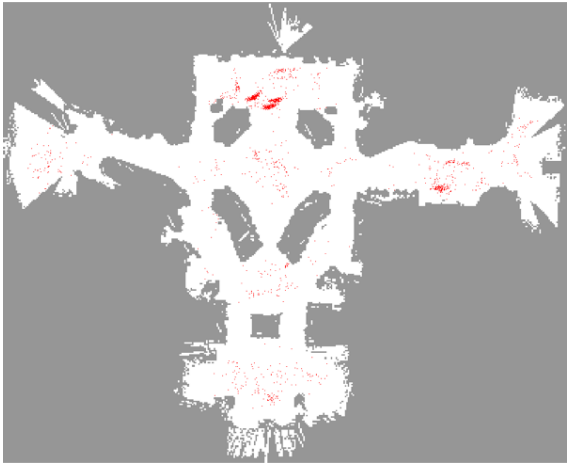
Monte Carlo Localization



Monte Carlo Localization

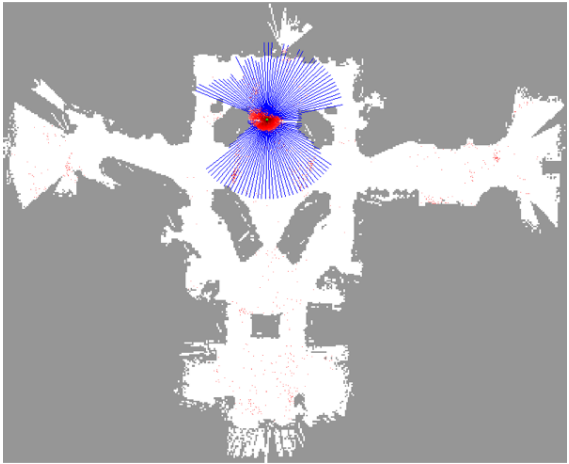


Monte Carlo Localization



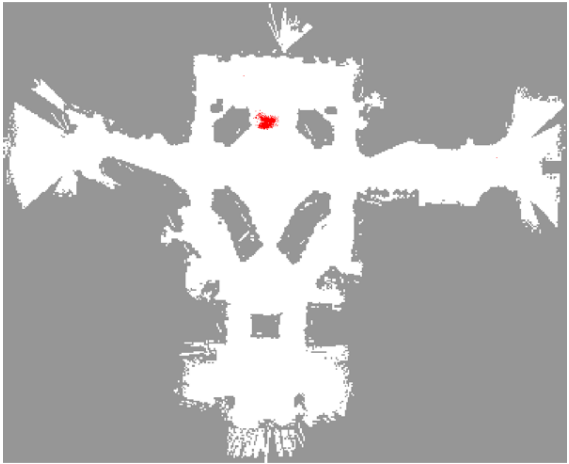
Monte Carlo Localization

Monte Carlo Localization

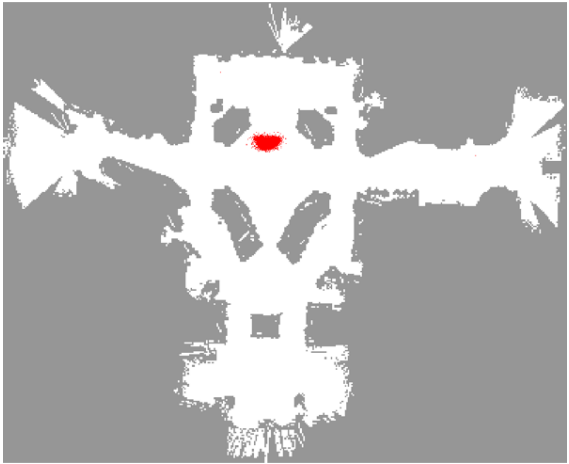


Monte Carlo Localization

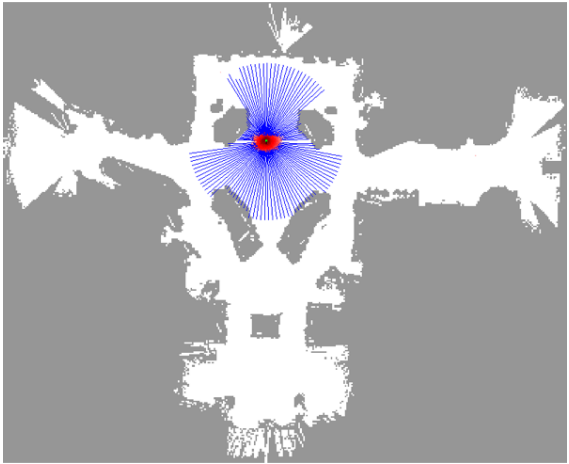
Monte Carlo Localization



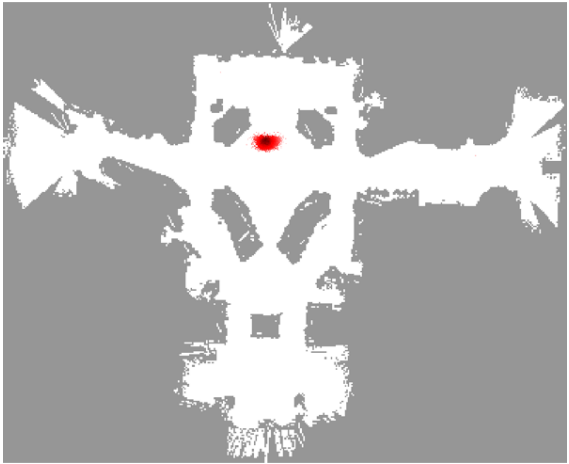
Monte Carlo Localization



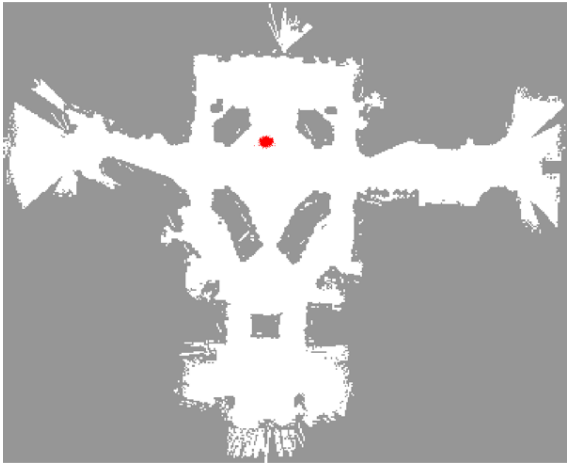
Monte Carlo Localization



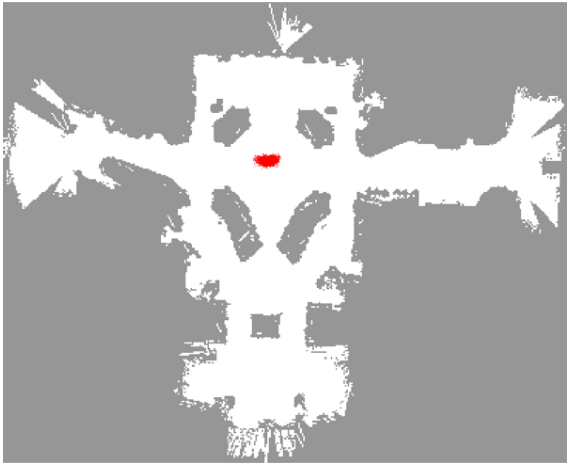
Monte Carlo Localization



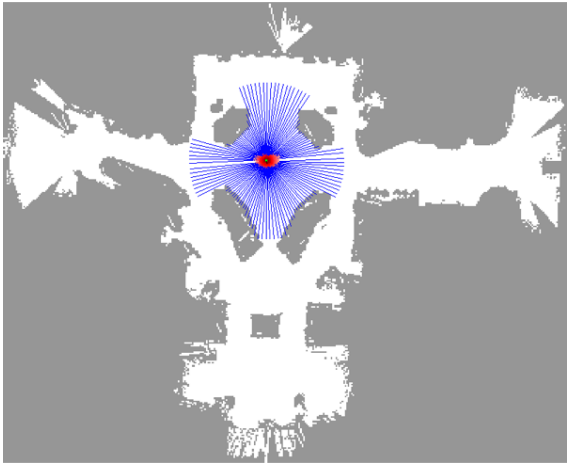
Monte Carlo Localization



Monte Carlo Localization



Monte Carlo Localization

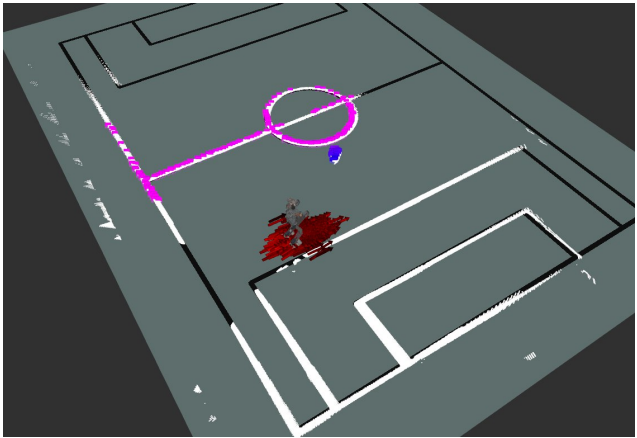


Particle Filter

Video

https://www.youtube.com/watch?v=F6T3dtXviNY

Particle Filter in RoboCup Soccer



Particle Filter

Live demo
https://amrl.cs.utexas.edu/interactive-particle-filters/

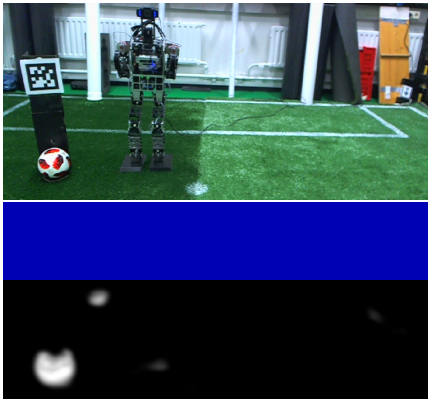
Particle Filter Assumption Extraction

- Belief is represented as particle cloud
- Most methods for further processing need a concrete assumption of the state
- Multiple ways of extracting assumptions from particle clouds:
  - Mean
  - Clustering algorithms (k-means)
  - Fitting Gaussians into particles
    - Expectation-Maximization algorithm
    - Gaussian Mixture Models (GMM)
    - Elbow-method

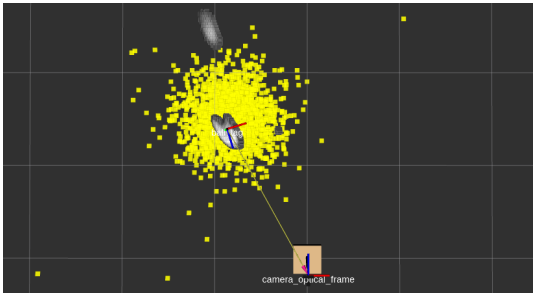
Particle Systems: Other Applications

- Particle-based inference is not restricted to a Pose state space
- Example Particle-based SLAM (*gmapping*)
  - Particles model the robot's pose *and an occupancy grid*, i.e. a probabilistic 2D map
  - Measurements weight *and update* particles
- Example FastSLAM
  - Each particle encapsulates the robot's pose
    and extended Kalman filters for each landmark
- Particle-based Inverse Kinematics
  - Particles represent joint angles of robotic manipulators
  - Optimization w.r.t. target pose and secondary objectives
- Signal processing
- ...

Application Example



- Non gaussian, multi-modal
- Filtering of ball position
- Direct use of FCNN output



Niklas Fiedler, Marc Bestmann, and Jianwei Zhang. "Position Estimation on Image-Based Heat Map Input using Particle Filters in Cartesian Space." 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS). IEEE, 2019.

Application Example

Video

Implementation

Simple python implementation for EKF and particle filters:
`https://atsushisakai.github.io/PythonRobotics/`

A C++ particle filter library for the robotics context:
`https://github.com/bit-bots/particle_filter`

Literature

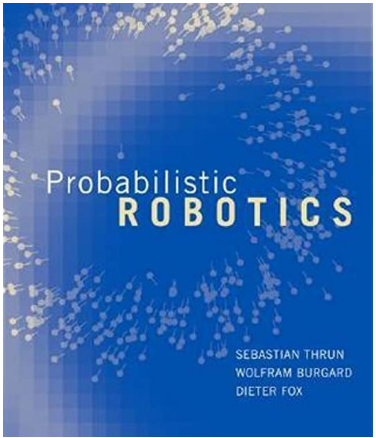

Probabilistic
ROBOTICS
SEBASTIAN THRUN
WOLFRAM BURGARD
DIETER FOX

Table of Contents

Table of Contents

Decision Making

- In the last lectures we learned how to use sensors to estimate the state of the robot and its environment
- Based on this estimation we need do decide which actions to take to achieve our goals
- This is often called *Decision Making*, *Behavior* or *High-Level Control*
- There are a lot of different approaches to do this, all with their advantages and disadvantages
- As the complexity for the robot's tasks grows, the complexity of its decision making grows too

No Silver Bullet

- Phrase coined by Fred Brooks in 1986 when talking about software engineering
- "*There is no single development, in either technology or management technique, which by itself promises even one order of magnitude [tenfold] improvement within a decade in productivity, in reliability, in simplicity*"
- In contrast to advances where we double our computing power every few years (Moore's law)
- The same problem that we already have with general software engineering applies also to robotics

No Silver Bullet

- We can not solve the complexity with a *magic* approach
- There is some *essential complexity* to complex tasks
  - If a robot has to do 3 different tasks, then you can't get around programming or teaching these 3 tasks somehow
- But there is also *accidental complexity*
  - Complexity coming from how the system is implemented
- We can solve accidental complexity by using better methods
  - using a high-level programming language like Python instead of machine code
  - using programming paradigms
  - or in robotics by using a fitting control architecture

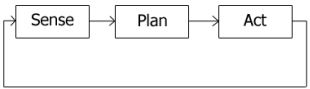
Table of Contents

Robotic Paradigm

- We can hardly use numbers to say what is the best approach (quantitative approach)
- But we can use reasoning to get some insights in an inductive way (qualitative approach)
- Before talking about concrete approaches for decision making, we should talk about categories
- There are four classes of robot control methods:
  - Deliberative
  - Reactive
  - Hybrid
  - Behavior-Based

Deliberative

- "Think, Then Act"
- Top-down approach
- Sensing, filtering, modeling, planning, execution
- Highly sequential
- Complex behaviors easy to model
- Can plan into the future by predicting results of its actions



https://en.wikipedia.org/wiki/Robotic_paradigm
Springer Handbook of Robotics, Chapter 13

Deliberative

- In most real life scenarios almost impossible to use, due to noise and unforeseen changes
- Applications tend to have a global world model
- Planning tends to take time, robot is not very reactive
- Examples:
  - Planning the complete path of a wheeled robot using a model of the world
  - Plan multiple arm movements to cook something

Reactive

- "Don't Think, (Re)Act"
- Bottom-up approach
- Multiple sense-act couplings
- Higher level behaviors emerge implicitly
- Very short reaction time, due to no planning
- Insects are largely working reactive

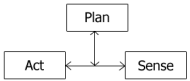
https://en.wikipedia.org/wiki/Robotic_paradigm
Springer Handbook of Robotics, Chapter 13

Reactive

- Capable of optimal performance for some problem types
- Not usable if internal model, memory or learning is required
- Examples:
  - Navigating a robot purely based on its current sensor inputs and a general goal direction

Hybrid

- "Think and Act Concurrently"
- Tries to combine the best of both worlds
- Deliberative part
  - Plans long term goals (low update rate)
  - Guides reactive part towards more optimal trajectories and goals
- Reactive part
  - Deals with current changes (high update rate)
  - Override deliberative part if unforeseen changes happen



https://en.wikipedia.org/wiki/Robotic_paradigm ; Springer Handbook of Robotics, Chapter 13

Hybrid

- Also called *layered robot control*
- Examples:
  - Path planning with additional reactive system for avoidance of obstacles
  - Humanoid robot with reactive system for reflexes (falling, standing up) and deliberative system for its high level goal

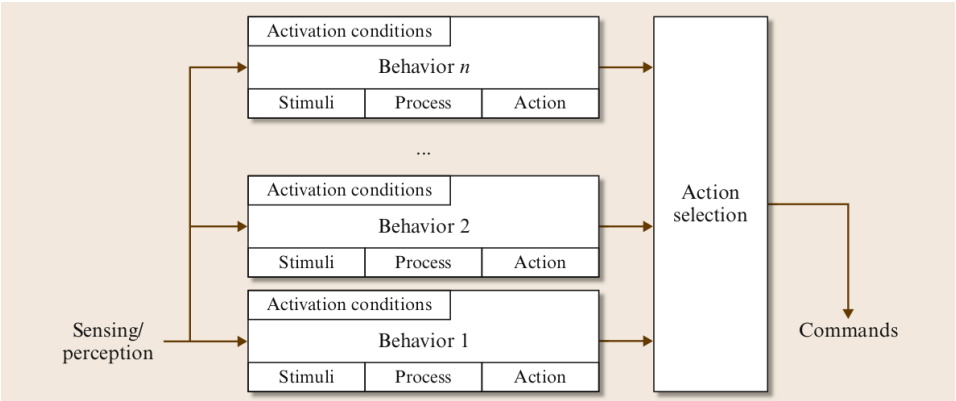
Behavior-Based

- "Think the Way You Act"
- Bottom-Up (similar to reactive approach)
- Multiple distributed, interacting modules, called behaviors
- Each behavior has its own goal
- Over all system behavior emerges from the behaviors
- No centralized world model, but every behavior has its own model

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

Springer Handbook of Robotics, Chapter 13

Behavior-Based



Springer Handbook of Robotics, Chapter 13

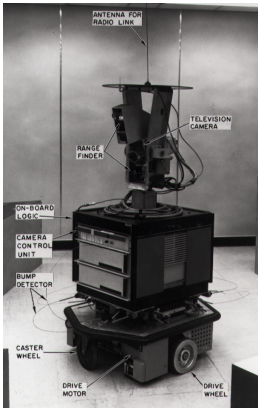
What we will discuss in the next lectures

- Conventional approaches
  - Symbolic reasoning (1950s)
  - Fuzzy logic (1965)
  - Subsumption (1986)
  - Decision trees (<=1963)
  - Finite state machines (<=1962)
  - Hierarchical finite state machines (state charts) (1980s)
- Recent approaches
  - Behavior trees (~2001)
  - Dynamic Stack Decider (2018)
- Design principles
- Advantages and disadvantages
- What to use when

Table of Contents

Motivation

- Symbolic reasoning is one of the oldest and best understood subfields of Artificial Intelligence
- Also called GOFAI ("Good Old-Fashioned Artificial Intelligence")
- Autonomous robots should act intelligently on low *and high* levels of abstraction
- Real-world tasks are usually given on an abstract level in natural language



SRI's Shakey

Examples of Symbolic Tasks for Autonomous Robots

Often, robots should exhibit intelligent adaptive "higher-level" behavior, with respect to their environment.

- Unblock the doorway (Shakey, 1966)
  - It is not specified how to get there
- Fetch a sandwich (JSK Tokyo, 2011)
  - Fetch it from the fridge or buy it in the store next-door
- Prepare a pancake (TUM/IAI Bremen, 2010)
- Set the table (Project RACE, TAMS, Hamburg)
- ...

These tasks require background theories and reasoning capabilities.

Symbolic vs. Statistic Reasoning

- symbolic (e.g. STRIPS):
  - Using symbols to represent the world and reason on it
  - Explainable
  - Needs not much data
  - Exemplary tasks:
    - planning
    - reasoning
    - language generation (sentences)
- statistical (e.g. something using a neural network):
  - Use statistics to represent the world and reason on it
  - Can handle uncertainty
  - Gets better with more data
  - Exemplary tasks:
    - speech generation (sound)
    - motion skills (e.g. walking)

Symbols

A symbol is a single token (or "atom") that can be uniquely identified among others.

- Symbols by themselves are syntactic entities
- Algorithms can operate on them
- They do not carry an inherent meaning/semantics
- Symbols in everyday life are associated with intuitive semantics, e.g. "tree", "red", "to shake"
- Science aims to give precise/consistent definitions of all symbols in use, e.g. "$\mathbb{N}$", "$+$", "$\varnothing$"

Assignable Semantics

In symbolic modeling the semantics of each symbol are influenced by

- Human interpretation
  - What does the designer/the user assume the symbol means?
- The relationship to other symbols
  - e.g. encoded as a theory in first-order predicate logic
- The anchoring/grounding of the symbol in the whole system
  - How does the occurrence of the symbol influence the system?

Human Concepts and Formal Symbols

There is a large discrepancy between formal symbols and everyday human concepts:

- Backed up by commonsense knowledge, formal symbols rely on the specified theory
- Intrinsically vague, this is hard to model for symbols *in a compatible way*
- Often polysemous or homonymous when mapped onto engineered symbols

- These differences can break the autonomous behavior and user interaction in unforeseen situations

Polysemous Examples

- Polysemous (different meaning but a semantic relation)
  - Mouse
    - A small rodent
    - A digital input device
  - Door
    - the object which swings open to allow entrance, as in “Open the door.”
    - the opening created thereby, as in “Walk through the door.”
- Homonymous (no semantic relation)
  - Bass
    - Type of fish
    - A tone of low frequency
    - A musical instrument

Symbol Grounding

The connection between symbols used inside an agent
and their counterparts in the real world is called *grounding*.

This has to rely on the robot's sensing and performance capabilities

- Perception modules can signal internal symbols by interpretation of sensor data
- Action modules can control the robot's behavior whenever special symbols are inferred

Symbol Grounding (2)

Perception modules

- Subfields of computer vision focus on individual modules
- Limited by the quality of the sensor data and applied perception algorithms
- Examples: `in_front_of(table1)`, `in(red_box, room2)`, `filled(bottle, 0.5)`, `activity(harry, reading)`

Action modules

- Focus of traditional robotics
- Control theory, reinforcement learning, ...
- Limited by motor accuracy, proprioception, applied control algorithms, *perception*
- Examples: `move(position5)`, `place(object1, table2)`, `open(shelf5)`, `plug_in_to_outlet(o1)`

On the Attribution of Semantics

- To get to a running behavior *all* modules have to be implemented in a consistent way.
- Many are active research fields

$\Rightarrow$ Most modules rely on simplifying assumptions / stubs

- These drastically reduce the scope and still capture human expectations in the prepared demo setup
- *But*, they often lead to unintuitive behavior and strangely prepared demo setups

- Next time you see a robotic demonstration, you might ask:
  - Which assumptions did the researchers have to make?
  - How many modules would have to be drastically changed to get this to work in a different environment?

Symbol Grounding (3)

Many symbols represent aspects of the environment that are difficult to ground

- Filling level of an opaque bottle, intention of a human, ...
- Try to infer them via explicit perception actions, dedicated prediction modules

Many action symbols can be defined in terms of other symbols and don't have to be implemented directly:

- `do_the_laundry`, `prepare_a_pancake`, `set_table(table2)`
- They can be grounded in terms of sequences of more basic actions

Primitive Actions

- Actions can either be implemented by modules or defined through other actions
- This leaves the choice of primitive/atomic actions

- Primitives have to have well-defined outcomes
- The set of primitive actions should support other demonstrations in the same environment
- Obvious candidates are
  - `set joint to position` X
  - `move base to position` X
  - `look at` X
  - `pickup` X
- Nevertheless, `do_the_laundry` can be a justified primitive action, assuming it "does the job"

Blocks World Planning

- Ignoring grounding, the set of relevant symbols can be used for symbolic inference
- Problems without grounding are sometimes referred to as "blocks world" problems, alluding to the blocks world domain
- Here, everything behaves as specified *by definition*

Start State Goal State

A Word On Notation

Classical predicate logic represents

- atomic statements as "`predicate(param1, param2)`"
- variables as capital letters "`X`, `Y`, `Z`"
- e.g. "`on(glass, X)`"

A large part of the planning community uses a Lisp-like syntax instead:

- statements: "`(predicate parameter1 parameter2)`"
- variables: "`?var1`"
- e.g. "`(on glass ?place)`"

STRIPS Planning

- **St**anford **R**esearch **I**nstitute **P**roblem **S**olver
- plans purposeful sequences of actions in blocks world domains
- defined a de-facto standard form for the formulation of planning problems

STRIPS planning domains contain

- a set of (boolean) variables, i.e. *fluents*, with mutable values
- discrete states characterized by their valid fluents
- a set of action specifications
- actions mark state transitions

STRIPS Planning (2)

STRIPS problems consist of

- The set of relevant fluents $\mathcal{F}$
- The set of effective action schemas $\mathcal{O}$
- The initial set of true fluents $\mathcal{I}$
- The set of literals (fluents or their negation) to be satisfied $\mathcal{G}$

Action schemas consist of

- a *signature* including all free variables, e.g. `action(P1,P2)`
- a set of literals as *precondition* for the action
- a set of literals as *effects* of the action

The STRIPS Blocks World

$\mathcal{F}$ (fluents): $X \in \{a, b, c\}, Y \in \{a, b, c, table\}$

Discrete states:

X is on top of $Y\ldots$	$On(X, Y)$
There is nothing on top of $X\ldots$	$Clear(X)$
X is a movable block…	$Block(X)$

$\mathcal{O}$ (action specifications):

- `move(B,X,Y)`
  - preconditions: $\{On(B, X), Block(B), Block(Y), Clear(B), Clear(Y), B \neq X, B \neq Y, X \neq Y\}$
  - effects: $\{On(B, Y), \neg On(B, X), Clear(X), \neg Clear(Y)\}$
- `move_to_table(B,X)`
  - preconditions:$\{On(B, X), Clear(B), Block(B), Block(X), B \neq X\}$
  - effects: $\{On(B, table), \neg On(B, X), Clear(X)\}$

The STRIPS Blocks World

Assume this initial state and goal:

$$\begin{aligned}\mathcal{I} =& \{Block(a), Block(b), Block(c), Clear(b), Clear(c)\} \\ & \cup \{On(c, a), On(b, table), On(a, table)\} \\ \mathcal{G} =& \{On(a, b), On(b, c)\}\end{aligned}$$

Given $\langle \mathcal{F}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, a planner can find a sequence of variable-free and applicable actions, i.e. a **plan**, to achieve $\mathcal{G}$ from state $\mathcal{I}$

The STRIPS Blocks World

Assume this initial state and goal:

$$\begin{aligned}\mathcal{I} =& \{Block(a), Block(b), Block(c), Clear(b), Clear(c)\} \\ & \cup \{On(c, a), On(b, table), On(a, table)\} \\ \mathcal{G} =& \{On(a, b), On(b, c)\}\end{aligned}$$

Given $\langle \mathcal{F}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, a planner can find a sequence of variable-free and applicable actions, i.e. a **plan**, to achieve $\mathcal{G}$ from state $\mathcal{I}$

One plan for this problem is

- `move_to_table(c,a)`, `move(b,table,c)`, `move(a,table,b)`

Open and Closed Worlds

- STRIPS assumes full information on the fluents of each state
- If a fluent is not in the state description, it is *false*
- This is called the **Closed World Assumption**
- In practice, inference schemes often rely on this assumption

. . . but just because you don't know your car was stolen,
that does not mean it is still where you parked it.

- Many planners do not make the CWA, but the resulting inferences are much weaker

. . . just because you parked your car somewhere, that does not mean it is there anymore.

PDDL

- Planning competitions (e.g. IPC) emerged
- . . . together with the need for a standardized language
- AIPS-98 introduced the **P**lanning **D**omain **D**efinition **L**anguage
- Provides means to specify STRIPS problems
- By now, supports many more elaborate constructions:
  - numeric fluents
  - continuous actions
  - timed events
  - preferences
  - function symbols (object-fluents)

PDDL - an Example Domain

```
(define (domain gripper-strips)
(:predicates (room ?r) (ball ?b) (gripper ?g) (at-robby ?r)
             (at ?b ?r) (free ?g) (carry ?o ?g))
(:action move
:parameters (?from ?to)
:precondition (and (room ?from) (room ?to) (at-robby ?from))
:effect (and (at-robby ?to) (not (at-robby ?from))))
(:action pick
:parameters (?obj ?room ?gripper)
:precondition (and (ball ?obj) (room ?room)
                   (gripper ?gripper) (at ?obj ?room)
                   (at-robby ?room) (free ?gripper))
:effect (and (carry ?obj ?gripper) (not (at ?obj ?room))
             (not (free ?gripper)))))
```

PDDL - an Example Problem

```
(define (problem strips-gripper2)
(:domain gripper-strips)
(:objects rooma roomb ball1 ball2 left right)
(:init (room rooma)
     (room roomb)
     (ball ball1)
     (ball ball2)
     (gripper left)
     (gripper right)
     (at-robby rooma)
     (free left)
     (free right)
     (at ball1 rooma)
     (at ball2 rooma))
(:goal (at ball1 roomb)))
```

Modeling Knowledge

- Autonomous robots often interact with similar environments
- For a start, all perform in the mundane world, where
  - Temporal ordering is transitive
  - People cannot remember things that will happen in the future
  - If you take a rigid object away, it will not be where it was anymore
  - If you cut up cheese, you will get smaller blocks of cheese
  - But if you cut up a cup, you will end up with shards
- This kind of commonsense knowledge has to be hand-crafted into each individual planning problem!

$\Rightarrow$ Collect such knowledge in a background theory, i.e. **ontology**

Modeling Common Sense Knowledge

- There have been many such projects in the past
- Prominent historical example:
  Patrick Hayes: "*Naive Physics I: Ontology for Liquids*" - 1978

Ongoing projects:

- (Open)Cyc / RoboEarth
  - Carefully handcrafted ontologies
- "Semantic Web"
  - Ongoing attempt to distribute knowledge engineering
  - Incompatible / conflicting ontologies

Modeling Common Sense Knowledge (2)

There are limits to what can be modeled

- A lot of commonsense knowledge is difficult to describe in boolean symbols
- Modeling probabilities/fuzzy logic leaves the problem where to get consistent numbers and makes inference much harder

- Adding background theories makes inference arbitrarily difficult
- Full predicate logic is *undecidable*
- There are various syntactic restrictions with different runtime complexity and expressiveness

RDF and OWL

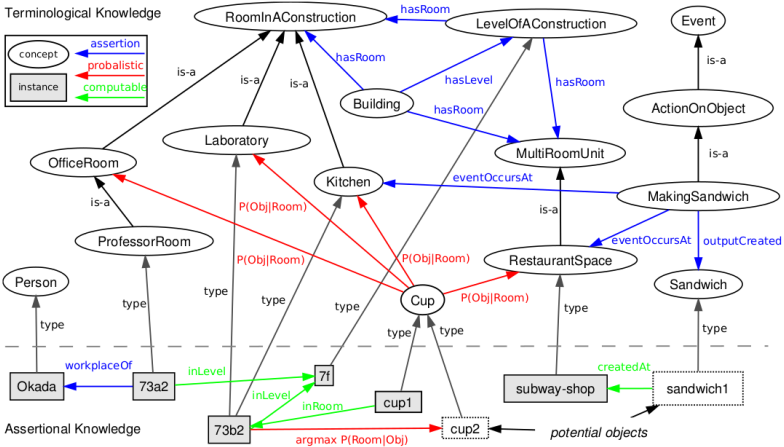
W3C standardized ontology notation on the web

- The **R**esource **D**escription **F**ramework
  - XML-based storage format for symbolic graphs
  - Based on triples expressing relations

```
<http://www.w3.org/People/EM/contact#me>
<http://www.w3.org/2000/10/swap/pim/contact#fullName>
"Eric Miller"
```

- The **W**eb **O**ntology **L**anguage (OWL)
  - Builds on RDF
  - Most prominent version: **OWL DL** implements formal Description Logic (DL)
  - supported by many tools / reasoners

Ontologies in the Wild - Fetching Sandwiches



Ontologies in the Wild - RACE

Ontologies in the Wild - KnowRob

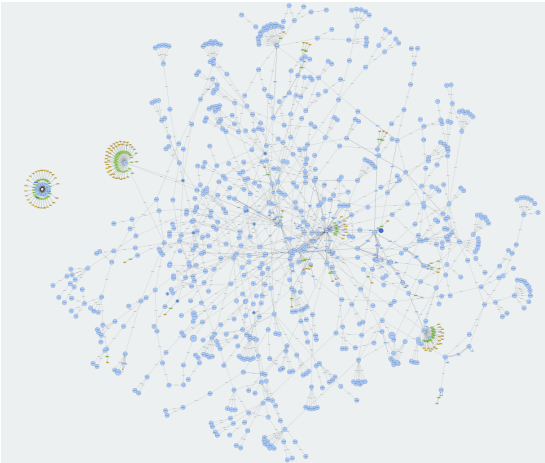


Table of Contents

Design Principles

- There are some design principles (similar to software architecture) for high-level control
- We can use them to investigate their pros and cons
- Their importance depends on the use cases

Design Principles of Control Architectures (CA)

- Hierarchical organization
  - Some subtask may be more important than others
  - Ex: recharging when empty > navigating to goal
- Reusable code
  - The same subtask is maybe needed multiple times
  - Ex: turning sensors to specific location
- Modular design
  - Splitting a task into subtasks makes development easier
  - Ex: divide “grasp” into “open hand”, “position hand”, “close”
- Maintainability
  - Changes to the behavior has to possible without general restructuring
  - Ex: adding “lift hand” to “grasp” should only require changes in this part

Colledanchise, Michele, and Petter Ögren. Behavior Trees in Robotics and AI, an Introduction., 2017
Poppinga, Martin and Bestmann, Marc. ASDS - Active Self Deciding Stack, 2018

Design Principles of Control Architectures (CA)

- Human readable
  - Structure has to be readable for developing and debugging
  - Ex: GUI with graph structure and current state
- Stateful
  - The current state of the system should be clear
  - Ex: clear if ball is currently in hand or not
- Fast
  - Low latency between sensor input and action
  - Ex: Bumper is hit -> immediate stop of wheels to prevent damage
- Expressive / scalable
  - CA must be able to encode a large variety of tasks
  - Ex: a soccer player with different strategies

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017
Poppinga, Martin and Bestmann, Marc. "ASDS - Active Self Deciding Stack", 2018

Design Principles of Control Architectures (CA)

- Suitable for automatic synthesis
  - Synthesis, e.g. by machine learning, for action ordering
  - Ex: using NNs in some parts to decide which action is to be taken
- Understandability of the concept
  - It should not take to much time to understand the concept
  - Ex: FSM is very simple, BT is complex
- Implementation effort
  - Effort to implement the used concept
  - Not important if fitting library is available

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017
Poppinga, Martin and Bestmann, Marc. "ASDS - Active Self Deciding Stack", 2018

Design Principles

- A lot of things to keep in mind
- Sometimes contradictory
  - Ex: fast <-> expressive
  - EX: maintainability <-> understandability
- Highly dependent on the domain and goal
- Keep in mind: there is no silver bullet

Symbolic Reasoning

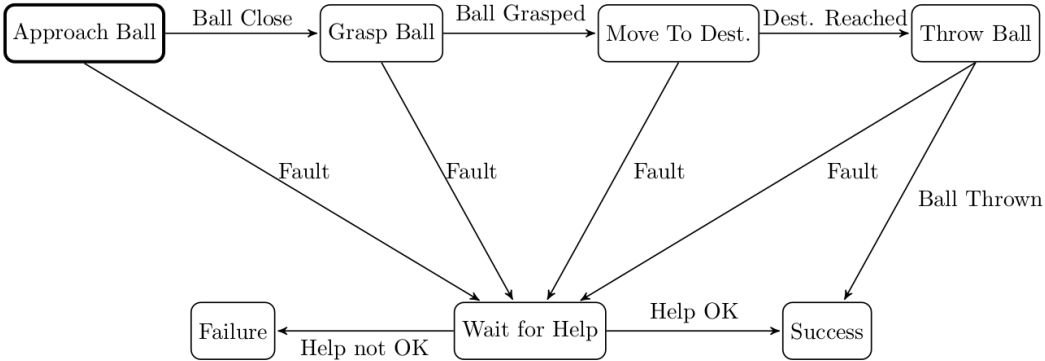
- Hierachical org.: Bad
  - Multiple goals possible but same importance
- Reusable: Good (ontologies)
- Modular: Very good
- Maintain.: depends
  - Actions very maintainable, state not
- Human read.: Good
- Stateful: Very good
- Fast: Bad
  - The classic example of deliberative approach
- Expressive: Good
- Synthesis: ?
- Understandable: Good
- Effort: Bad

Table of Contents

Finite State Machine

- Very common in computer science
- Often implicitly implemented
  - State is encoded in multiple variables of flags
- Good theoretical foundation
- Working principle
  - List of possible states
  - Transitions between those states
  - Start state
  - Check for transition conditions
  - Change state if condition is true
  - Act according to current state

FSM - Example



Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

FSM - Pseudo Code - Transition in States

```
class AbstractState:
    def run(self, blackboard):
        raise NotImplementedError
    def next(self, blackboard):
        raise NotImplementedError
class ApproachBall(AbstractState):
    def run(self, blackboard):
        # send some walking commands
    def next(self, blackboard):
        if blackboard.ball_distance < 1:
            return GraspBall()
        if blackboard.fault:
            return WaitForHelp()
        return self
class BallStateMachine:
    def __init__(self, blackboard):
        # the blackboard is some kind of object holding all information
        self.blackboard = blackboard
        self.current_state = ApproachBall()
        while true:
            self.run()
            sleep(0.1)
    def run(self):
        self.current_state.run()
        self.current_state = self.current_state.next()
```

FSM - Pseudo Code - Transition in Machine

```
class ApproachBall(AbstractState):
    def run(self, blackboard):
        # send some walking commands
class GraspBall(AbstractState):
    ...
class BallStateMachine:
    def __init__(self, blackboard):
        # the blackboard is some kind of object holding all information
        self.blackboard = blackboard
        fsm = StateMachine(inital=ApproachBall, states=[ApproachBall, GraspBall, ...])
        fsm.add_transition(from=ApproachBall, to=GraspBall, if=ball_close)
        fsm.add_transition ...
        while true:
            self.run()
            sleep(0.1)

    def ball_close(self):
        return blackboard.ball_distance < 1

    def run(self):
        self.fsm.get_current_state().run()
        self.fsm.check_transition()
```

FSM - Defining Transitions

- Transitions can be defined by the state (version 1)
- Or in the statemachine (version 2)
- This has pros and cons
- Pro in state
  - Decision can depend on “state of the current state”
    - Ex: “Wait5Sec” remembers time when state started
  - Simpler to implement
  - Easier to see to which state you go from one state
- Con in state
  - Danger of putting too much “state into a state”, leading to an implicit HSM
  - Transitions are all distributed across states
  - More difficult to use if you have events
- Both versions can be found in libraries

FSM - Advantages and Disadvantages

Advantages:

- Commonly used in computer science
- Intuitive structure
- Ease of implementation

Disadvantages:

- Maintainability
- Scalability (“state explosion”)
- Reusability
- No standardization

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

FSM - Conclusion

- Simple to understand and implement
- Very wide spread
- Use for small/trivial scenarios
- Stateful

Libraries

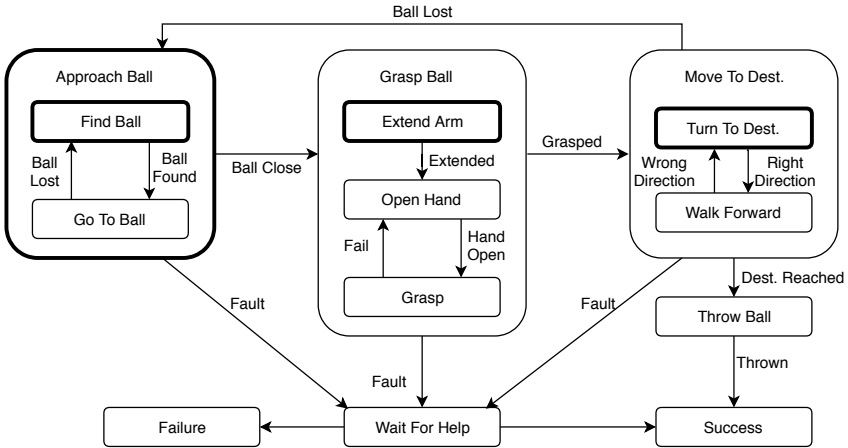
- Too many to list
- I recommend picking one which gives you graphical output for better debugging

Table of Contents

Hierarchical State Machines

- Also known as State Charts (UML)
- Solve some of the problems of FSMs
- Introducing a hierarchical layout
- Each state can consist of substates
- States with substates are called superstates
- Generalized transitions connect superstates
- Each superstate has a start substate
- The number of overall transitions is reduced

HSM - Example


Ball Lost
Approach Ball
Find Ball
Ball Lost
Ball Found
Go To Ball
Ball Close
Grasp Ball
Extend Arm
Extended
Open Hand
Fail
Hand Open
Grasp
Grasped
Move To Dest.
Turn To Dest.
Wrong Direction
Right Direction
Walk Forward
Dest. Reached
Throw Ball
Thrown
Success
Fault
Fault
Fault
Failure
Wait For Help

HSM - Advantages and Disadvantages

Advantages:

- Modularity
- Behavior inheritance

Disadvantages:

- Maintainability
- Non-intuitive hierarchy

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

HSM - Conclusion

- Still relatively easy to implement
- Stateful
- Still comparably wide spread
- Useful in medium complex scenarios

Libraries

- smach (ROS)
- pysm (Python)

Table of Contents

Table of Contents

Introduction

The concept of fuzzy logic was introduced by Lotfi Zadeh (1965)

- Inspired by human information processing capabilities
- People do not require precise, numerical information input, yet they are capable of highly adaptive control

General assumption

- If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement
- How important is it to be exactly right when a rough answer will do?

Fuzzy Control

- *Fuzzy* means: blurred, diffuse, vague, uncertain, ...
- Fuzzy control uses fuzzy sets as mechanism for
  - Abstraction of unnecessary or too complex details
  - Troubleshooting of problems which are not easily solvable by a simple *yes* or *no* decision
  - Modeling of (*soft*) concepts without any sharp borders

Fuzzy Control

- Unsharp linguistic grading of terms like
  "big", "beautiful", "strong" ...
- Human thinking models and behavior models with
  first-level logic
  - **Car driving:** *if-then*-rules
  - **Car parking:** Accurate up to a millimeter?
- Fuzzy speech instead of numerical description
  - "Brake 2.52 m ahead of the curve!" $\rightarrow$ only in machine systems
  - "Brake shortly before the curve!" $\rightarrow$ in natural language

Adaptive Control Methods

Fuzzy control is generally a good fit for realization of *adaptive control methods*

- Control can be understood as mapping from a sensor space onto actions
- In many cases it is *a priori* unknown, which measurement parameters are especially important for the choice of actions
- Some systems are very hard to describe in a mathematical way
- Often, sensor data is inaccurate, noisy and/or high dimensional

Adaptive Control Methods

Models for adaptive control systems

- The creation of an ideal mapping between sensor space and actions is very difficult with classical methods of control engineering
- In order to control such systems, a simpler method needs to be used for description
- Neural networks (data based)
- **Fuzzy-Controller** (rule based)

Linguistic Variables

Linguistic variables are one of the main building blocks of fuzzy logic

- A *linguistic variable* is a variable, which can take on a range of linguistic terms
- A *linguistic term (value, label)* is the quantification of a term from natural language through a fuzzy set
- Many terms of natural language can be characterized through degree of membership related to fuzzy sets
- Therefore, fuzzy sets can be considered as the basic tool for modelling of *linguistic terms*

Linguistic Variables

Examples:

- Linguistic variable: "**SPEED**"
- Linguistic terms of "**SPEED**"
  "**high**", "**low**", "**rapid**", "**economical**"

- Linguistic variable: "**BUILDING**"
- Linguistic terms of "**BUILDING**"
  "**cottage**", "**bungalow**", "**skyscraper**"

Characteristic Function

Crisp sets can be defined through specification of their characteristic function:

$$\mu_{\mathcal{A}}(x) = \begin{cases} 1 & \text{for } x \in \mathcal{A} \\ 0 & \text{for } x \notin \mathcal{A}, \end{cases}$$

where $\mu_{\mathcal{A}} : X \rightarrow \{0, 1\}$
Example:
"Apple" would be a fruit (result 1) but "Potato" not (result 0).

Membership Function

For **fuzzy sets** A, a generalized characteristic function μ_A is used, which maps a real number $[0, 1]$ to each element $x \in X$:

$$\mu_A : X \rightarrow [0, 1]$$

- The function μ_A is called membership function (MF)
- It indicates the "degree", to which the element x belongs to the described unsharp set A ($\rightarrow$ fuzzy set)
- Example:
  - "Apple" would be 1 for the set "Fruit"
  - "Potato" would maybe be 0.1 for the set "Fruit" since it is not really a fruit but closer to it than e.g. "Car"

Membership Function

Representation of membership functions

- Discrete representation
  - Fixed-size array
  - Saving of the MF-Values for the whole x-codomain
- Parametric representation
  - Functions with parameters (less space required)
  - Typical types: Singleton, triangular shape, trapezoid shape, bell curve, B-Spline basis function

Membership Function

Creation of the membership functions

- Context-dependent specification
  - Experimental, domain- and application specific
- Construction using sample data
  - Clustering
  - Interpolation
  - Curve Fitting (Least-squares)
  - Neural networks
- Knowledge acquisition through experts
  - One or several experts
  - Directly and indirectly

Fuzzy Set

A **fuzzy set** A over a universe X is given through a mapping $\mu_A : X \rightarrow [0, 1]$.
For all $x \in X$, $\mu_A(x)$ denotes the degree of affinity (membership) of x in A

Example:

- The set of integer numbers approximately equal to 10:

$$A_{10} = (0.1, 7), (0.5, 8), (0.8, 9), (1.0, 10), (0.8, 11), (0.5, 12), (0.1, 13)$$

Fuzzy Control

In a fuzzy control system, the influence on dynamic circumstances of a fuzzy system is characterized by a set of linguistic description rules

IF (a set of conditions is satisfied)
THEN (a set of consequences can be infered)

Conditions (antecedents or premises) of the IF-part:

→ Linguistic variables from the domain of process states

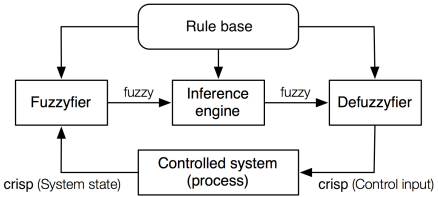
Conclusions (consequences) of the THEN-part:

→ Linguistic variables from the control domain

Components of Fuzzy Control

A complete fuzzy controller consists of four major components

- Rule base
- Fuzzifier
- Inference engine
- Defuzzifier



Rule Base

The rule base is the central component that stores expert knowledge

- It contains the control strategies in the form of IF-THEN rules

The rule base is considered a part of the knowledge base of the fuzzy controller, other components being:

- The input membership functions
  → required for *fuzzification* of crisp input values
- The output membership functions
  → required for *defuzzification* of the inference result(s)

Rule Base

Example:

Given a fuzzy control system with two inputs A and B and a single output C, the general representation of the rule base would be:

R_1: IF (x is A_1 OR y is B_1) THEN (z is C_1)

R_2: IF (x is A_2 OR y is B_2) THEN (z is C_2)

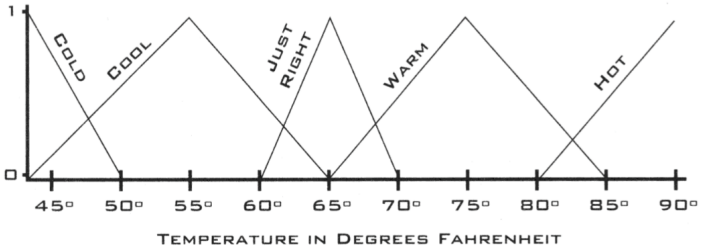
...

R_k: IF (x is A_k OR y is B_k) THEN (z is C_k)

Note: Inputs and outputs mentioned here are strictly fuzzy.

Fuzzifier

The fuzzifier converts the crisp input values into fuzzy representations based on the membership degree to established fuzzy sets



Fuzzifier

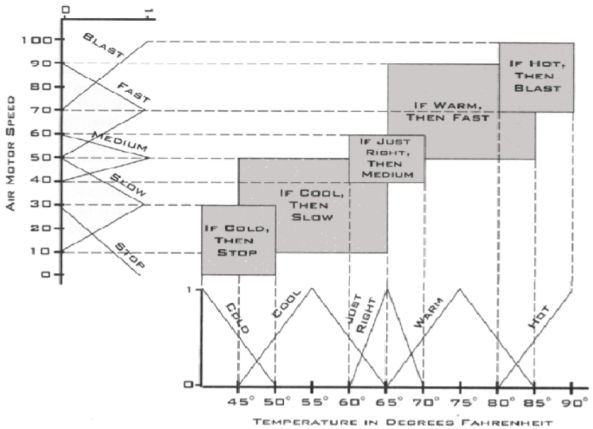
- The established fuzzy sets and associated membership functions are designed to exploit the inherent inaccuracy of input data (e.g. from sensors)
- The fuzzifier approximates the human reasoning process
- While expert knowledge of the process to be controlled is helpful, it is not a requirement
- The overall implementation effort of the controller is reduced significantly

Inference Engine

The inference engine processes the *fuzzified* input values based on evaluation of the rule base

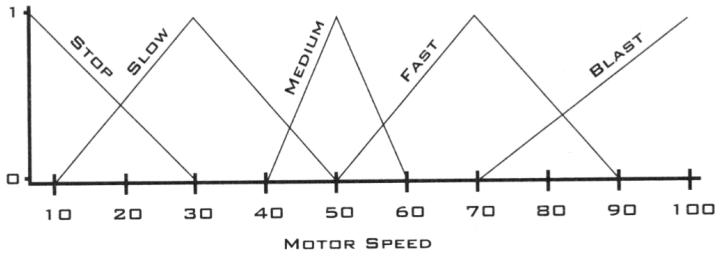
- *All* rules within the rule base are evaluated
- The evaluation usually occurs in parallel (depends on implementation)
- All evaluated rules contribute to the fuzzy output value to some degree
- The contribution of most rules to the output value is 0
- The resulting fuzzy output value is a *union* of the results of all evaluated rules

Inference engine (cont.)



Defuzzifier

The defuzzifier converts the obtained fuzzy output value into a crisp representation based on the output membership functions



Defuzzifier (cont.)

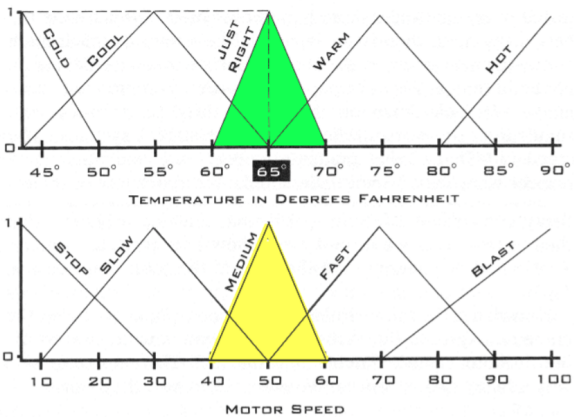
Several strategies exist for defuzzification

- The center of gravity (CoG) technique is very common

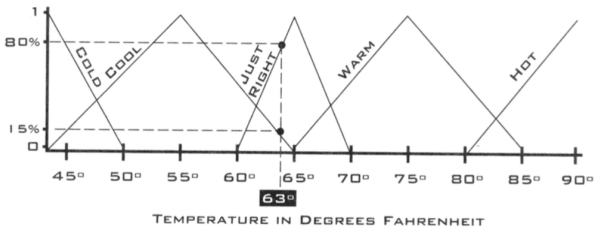
Other strategies include:

- Mean of maximum
  → The defuzzified result represents the mean value of all actions, whose membership functions reach the maximum
- Weighted average method
  → Formed by weighting each output by its respective maximum membership degree
- ...

Simple case

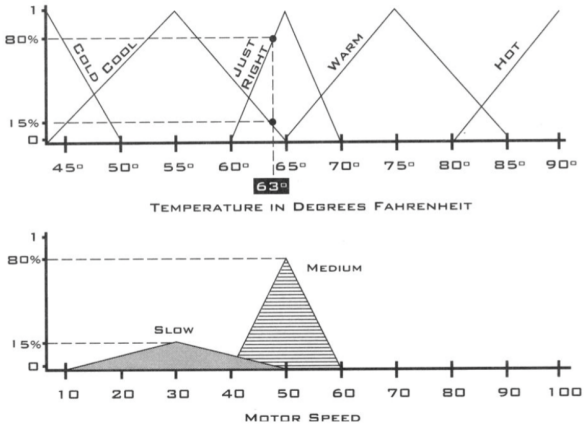

1
0
COLD
COOL
JUST RIGHT
WARM
HOT
45° 50° 55° 60° 65° 70° 75° 80° 85° 90°
TEMPERATURE IN DEGREES FAHRENHEIT
1
0
STOP
SLOW
MEDIUM
FAST
BLAST
10 20 30 40 50 60 70 80 90 100
MOTOR SPEED

General case

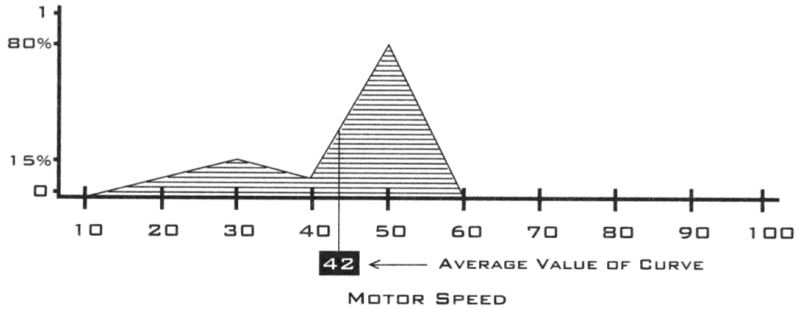


- IF temperature is **cool** THEN motor speed is **slow**
- IF temperature is **just right** THEN motor speed is **medium**

General case (cont.)



General case (cont.)



Curse of dimensionality

Fuzzy logic based control models (nonlinear modeling techniques) are affected by the *curse of dimensionality*

$\rightarrow$ If the number of inputs grows, the cost of both implementing the rule base and obtaining an output value increase exponentially.

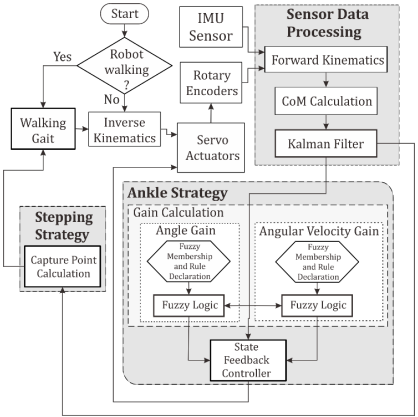
Design Paradigms

- Hierarchical org.: Okay
- Reusable: Okay
- Modular: Bad
- Maintain.: Bad
- Human read.: Good
- Stateful: Bad
- Fast: Very good
  - The classic example of reactive approach
- Expressive: Bad
- Synthesis: Very good
- Understandable: Good
- Effort: Good

Fuzzy Control in Action

Fuzzy Control in Action

Fuzzy Control in Action



Bagaskara Primastya Putra et al. Humanoid robot pitch axis stabilization using linear quadratic regulator with fuzzy logic and capture point, 2020

Summary

- Fuzzy logic provides a different way to approach a control problem
- It is based on natural language
- It allows to solve the problem without the need of a mathematical model
- FL based controllers require less implementation effort and are thus usually cheaper
- FL is inherently tolerant of imprecise data
- It can model nonlinear functions of arbitrary complexity
- Fuzzy logic can be combined with conventional control techniques

Summary

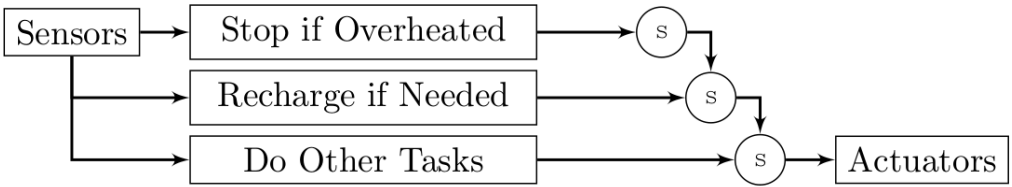
- Fuzzy logic is not a cure-all!
- Fuzzy logic is a convenient way to map an input space to an output space
- However, many controllers can do a fine job without it
- Fuzzy logic *can* be a powerful tool for dealing with imprecision and nonlinearity

Table of Contents

Subsumption Architecture

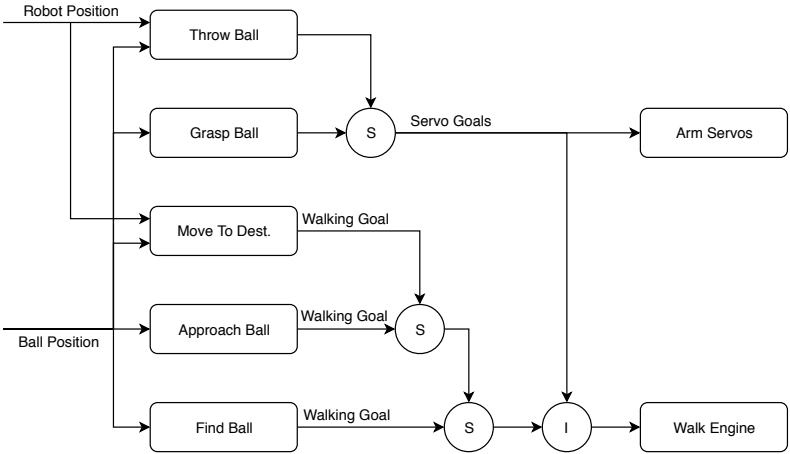
- Several modules
- Each implements one task
- All run in parallel
- Modules are ordered by priority

Subsumption - Example



Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

Subsumption - Robot Example



Subsumption - Advantages and Disadvantages

Advantages:

- Modularity
- Hierarchy
- Reactivity

Disadvantages:

- Scalability
- Maintainability
- Not stateful

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

Subsumption - Conclusion

- Good for reactive systems
- Hard to handle time dimension
- Usable for small to medium complex systems
- Not widely used

Libraries

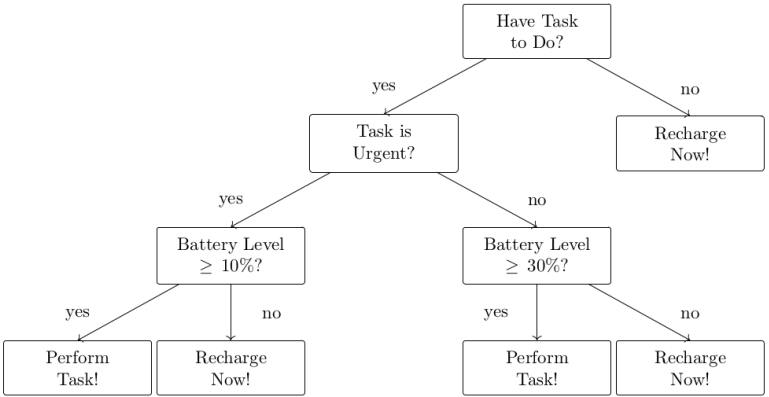
- subsuMeLib (C++) -outdated-

Table of Contents

Decision Trees

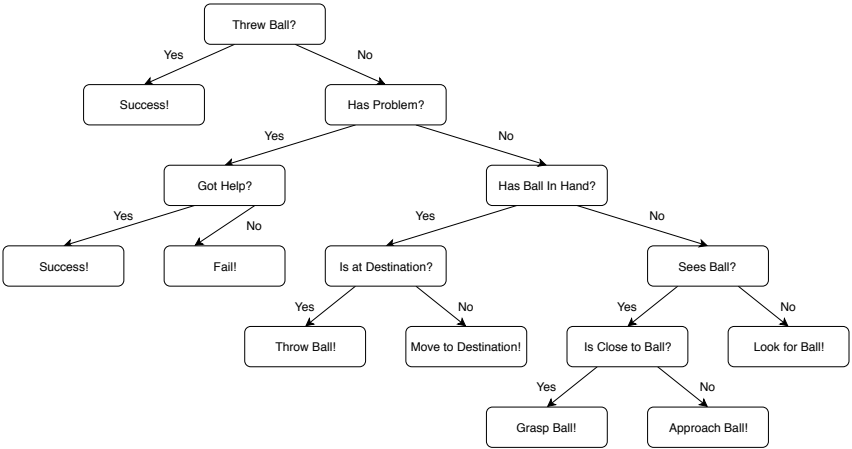
- Representation of nested if-then clauses
- Tree structure
- Internal nodes are predicates
- Leaf nodes are actions

DT - Example



Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

DT - Robot Example



DT - Advantages and Disadvantages

Advantages:

- Modularity
- Hierarchy
- Intuitive structure
- Clear division between actions and decisions

Disadvantages:

- Repetitions
- Maintainability
- Not stateful

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

DT - Conclusion

- Trivial to implement
- Using a framework rather than if-else can help with larger trees
- Widely (implicitly) used in computer science
- Easy to use with machine learning
- Good for domains which have no time dimension

Libraries

- scikit-learn + dtreeviz (Python)

Table of Contents

Behavior Trees

- Tree of nodes
- Internal nodes are control flow nodes
  - Sequence
  - Fallback
  - Parallel
  - Memory
- Leaf nodes are execution nodes
  - Action
  - Condition
- Root node sends out *ticks* in fixed frequency to its children
- Only nodes that receive a tick are executed
- Children return *Running*, *Success* or *Failure*

BT - Sequence Node

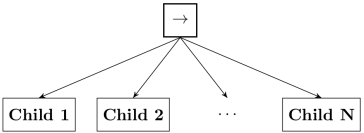


Fig. 1.2: Graphical representation of a Sequence node with N children.

Algorithm 1: Pseudocode of a Sequence node with N children

```
1 for i ← 1 to N do
2     childStatus ← Tick(child(i))
3     if childStatus = Running then
4         return Running
5     else if childStatus = Failure then
6         return Failure
7 return Success
```

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

BT - Fallback Node

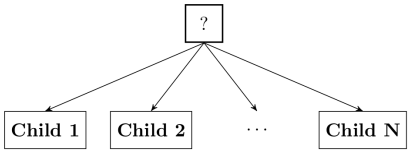


Fig. 1.3: Graphical representation of a Fallback node with N children.

Algorithm 2: Pseudocode of a Fallback node with N children

```
1 for i ← 1 to N do
2     childStatus ← Tick (child(i))
3     if childStatus = Running then
4         return Running
5     else if childStatus = Success then
6         return Success
7 return Failure
```

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

BT - Parallel Node

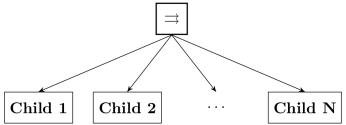


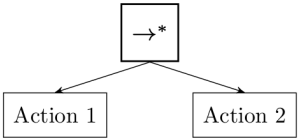
Fig. 1.4: Graphical representation of a Parallel node with N children.

Algorithm 3: Pseudocode of a Parallel node with N children and success threshold M

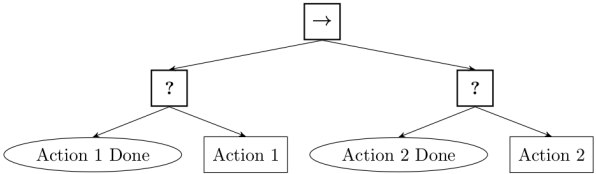
```
for i ← 1 to N do
    childStatus(i) ← Tick (child(i))
if Σ_{i:childStatus(i)=Success} 1 ≥ M then
    return Success
else if Σ_{i:childStatus(i)=Failure} 1 > N − M then
    return Failure
return Running
```

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

BT - Memory Node



(a) Sequence composition with memory.



(b) BT that emulates the execution of the Sequence composition with memory using nodes without memory.

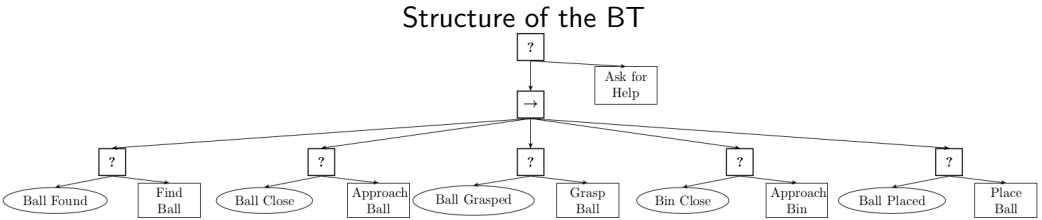
Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

BT - Nodes

Node type	Symbol	Succeeds	Fails	Running
Fallback	?	If one child succeeds	If all children fail	If one child returns Running
Sequence	→	If all children succeed	If one child fails	If one child returns Running
Parallel	⇉	If $\geq M$ children succeed	If $> N - M$ children fail	else
Action	text	Upon completion	If impossible to complete	During completion
Condition	text	If true	If false	Never
Decorator	◊	Custom	Custom	Custom

Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

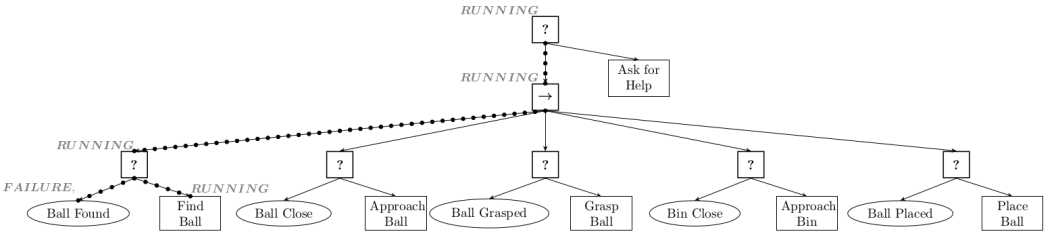
BT - Example



Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

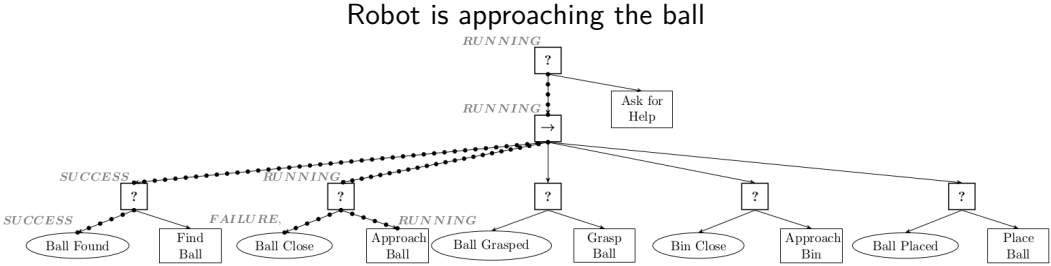
BT - Example

Start with searching the ball



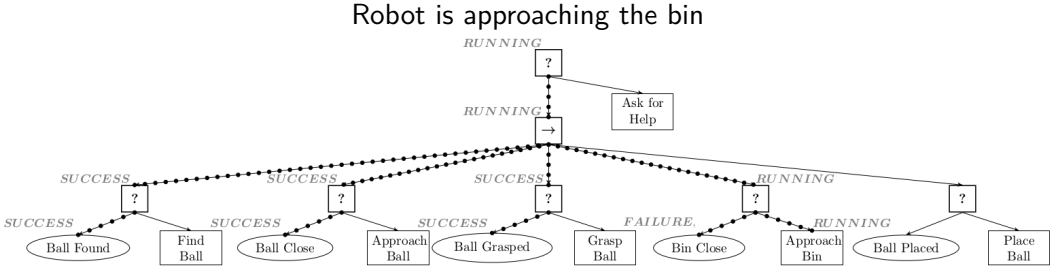
Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

BT - Example



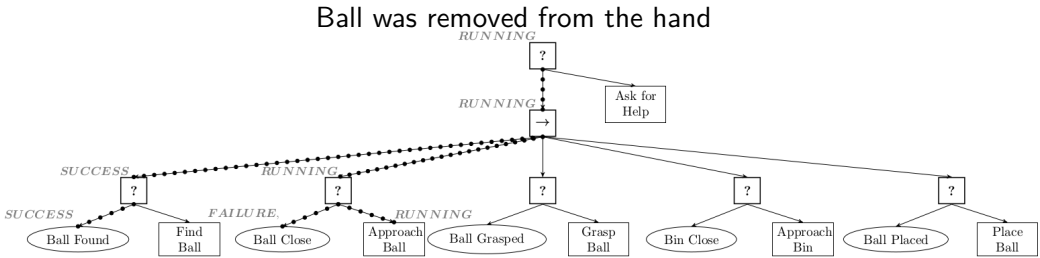
Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

BT - Example



Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

BT - Example



Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

BT - Real World Example

Video

Advantages and Disadvantages

Advantages:

- Good modularization and code reusability
- Well maintainable
- Frameworks and knowledge from usage in game industry
- Well formalized elements
- Parallelism possible

Disadvantages:

- Concept less intuitive
- Engine is complicated to implement
- Due to parallel activation current state difficult to see
- Testing preconditions indirect

BT - Conclusion

- Very powerful
- Good for large scenarios
- Also usable in smaller scenarios, but a bit overkill
- More complicated to learn but worth it

BT - Libraries and Tools

- Implemented in most big game engines
  - Unity
  - Unreal
  - ...
- behavior_tree (ROS package)
- YARP-Behavior-Trees (YARP Library)

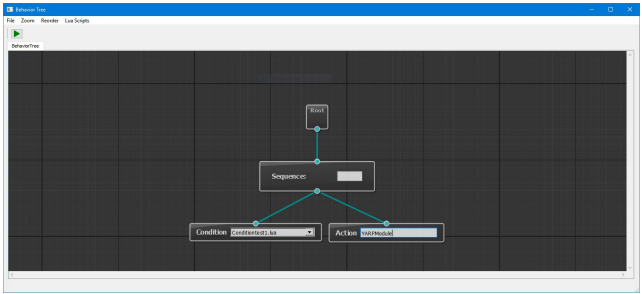


Table of Contents

Dynamic Stack Decider

Disclaimer:The DSD was developed by the Hamburg Bit-Bots.
During that time I was a member of the team.
Be critical about what I say.

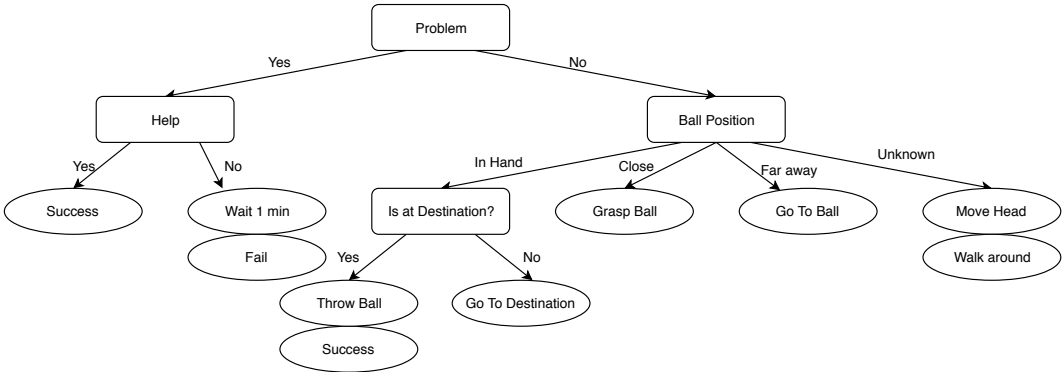
DSD - Motivation

- FSMs/HSMs not usable for non trivial problems
- BTs complicated to understand/implement, expensive to run
- Why not try to combine the advantages of the existing approaches
  - Tree structure (DT, BT)
  - Decision as internal nodes (DT)
  - Clear state (FSM, HSM)
  - Semantic transitions (FSM, HSM, DT)

Dynamic Stack Decider

- Tree like structure
  - Internal nodes are decisions (no time component)
  - Leaf nodes are actions (time component)
- Decisions provide a semantic outcome (string)
- Tree structure is defined by a simple domain specific language (DSL)
- State consists of current active action and previous decisions
- Decisions can be reevaluated to easily recheck preconditions
- But not all decisions have to be re-decided every time
- Action can be concatenated as sequences

DSD - Example


Problem
Yes
No
Help
Ball Position
Yes
No
In Hand
Close
Far away
Unknown
Success
Wait 1 min
Fail
Is at Destination?
Grasp Ball
Go To Ball
Move Head
Walk around
Yes
No
Throw Ball
Success
Go To Destination

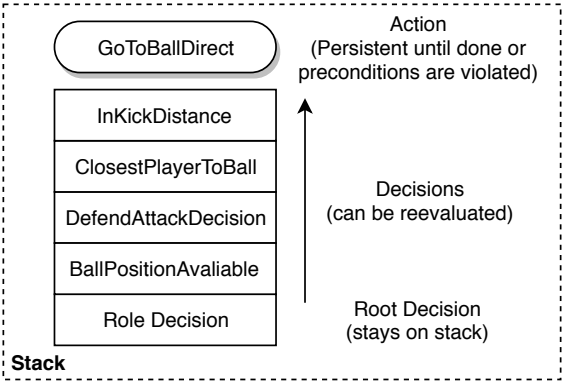
DSD - Domain Specific Language

```
-->BallBehavior
$Problem
    YES --> $Help
        YES --> @Success
        NO --> @Wait1Min, @Fail
    NO --> $BallPosition
        IN_HAND --> $IsAtDest
            YES --> @ThrowBall, @Success
            NO --> @GoToDest
        CLOSE --> @GaspBall
        FAR_AWAY --> @GoToBall
        UNKNOWN --> @MoveHead, @WalkAround
```

DSD - Code Example

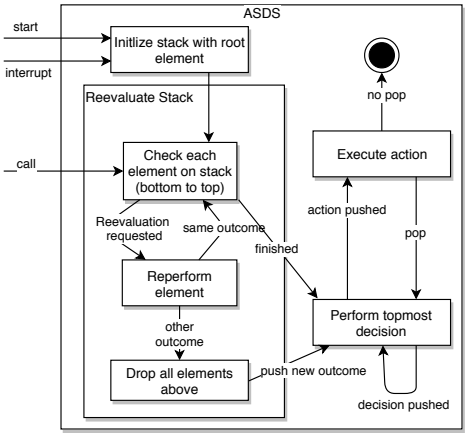
```
def BallPosition(AbstractDecisionElement):
    def perform(self):
        if not self.blackboard.ball_position:
            return "UNKNOWN"
        elif self.blackboard.ball_position < 0.5:
            return "IN_HAND"
        elif self.blackboard.ball_position < 1:
            return "CLOSE"
        else:
            return "FAR_AWAY"

    def get_reevaluate(self):
        return True

def GoToBall(AbstractActionElement):
    def perfom(self):
        # send some walking commands
def GrapsBall(AbstractActionElement):
    def __init__(self):
        # start grapsing animation

    def perform(self):
        if grasping_animation.is_finished():
            self.pop()
```

DSD - Stack



Poppinga, Martin and Bestmann, Marc. "DSD - Dynamic Stack Decider. A Lightweight Decision Making Framework for Robots and Software Agents" (2019) in peer-review process

DSD - Flow


ASDS
start
interrupt
Initlize stack with root element
Reevaluate Stack
call
Check each element on stack (bottom to top)
Reevaluation requested
same outcome
finished
Reperform element
other outcome
Drop all elements above
push new outcome
no pop
Execute action
action pushed
pop
Perform topmost decision
decision pushed

Video

Examplary video
https://www.youtube.com/watch?v=tZnBZsUAVqs (1:50)

Advantages and Disadvantages

Advantages:

- Clear seperation of decisions and actions
- Semantic transitiitions
- Time component of actions
- Clear state and previous decisions
- Simple checking of preconditions
- Different reevaluation timescales possible in the same DSD

Disadvantages:

- No parallelism
- Concept not intuitive

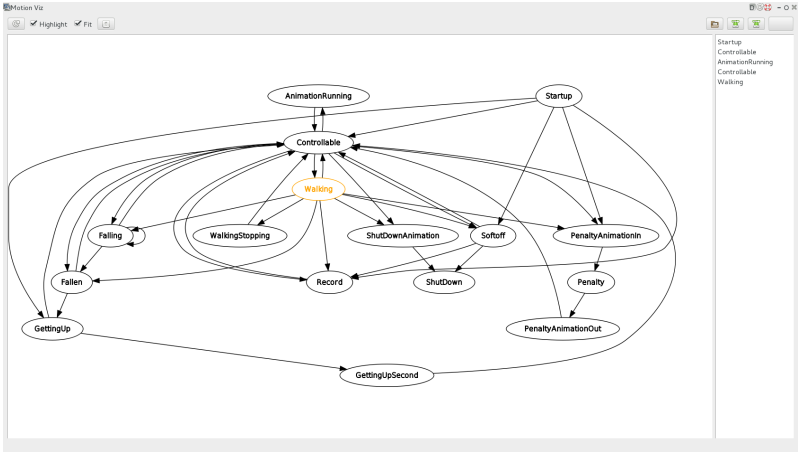
DSD - Conclusion

- Good for medium to complex scenarios
- Directly designed for robotics
- Not widely used

Libraries:

- dynamic_stack_decider (ROS)

Comparision FSM - DSD



Comparision FSM - DSD

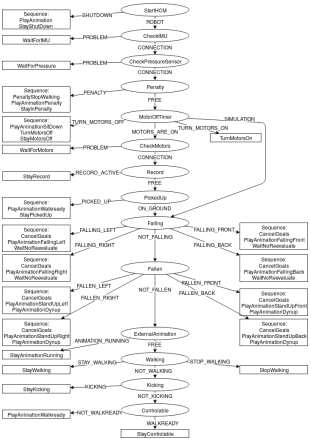


Table of Contents

Summary

- There is not one perfect solution
- What fits depends highly on the use case
- Necessary to think which approach you want to use
- Besides pros and cons of the approaches there are other factors
  - What is currently used in the project
  - Where do you or your colleagues have already experience in
  - Is there a library available for your middleware

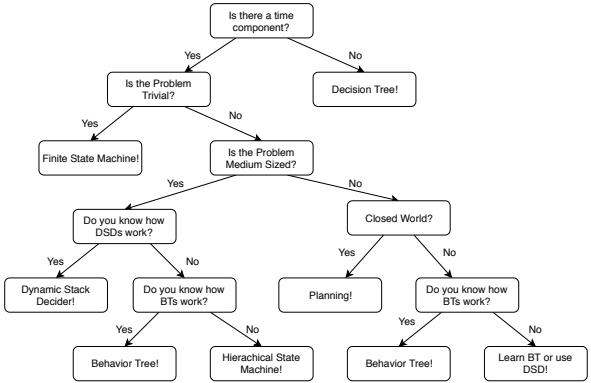
Comparison (subjective)

	SR	FL	FSM	HSM	Sub.	DT	BT	DSD
Hierarchical org.	-	○	-	+	++	+	+	+
Reusable code	+	○	-	○	-	+	+	++
Modular design	++	-	-	+	+	+	+	+
Maintainability	○	-	-	○	-	-	+	+
Human readable	+	+	-	○	-	+	○	+
Stateful	++	-	++	+	-	-	+	+
Fast	- -	++	-	-	+	++	○	+
Sufficiently expressive	+	-	+	+	-	+	+	+
Suitable for synthesis	+	++	+	+	-	+	○	+
Understandability	+	+	+	+	+	++	-	-
Implementation effort	-	+	++	+	-	++	-	○

(partly) Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI, an Introduction.", 2017

Choosing an Architecture

A subjective proposal on choosing an architecture



Layered Architectures

- Complex systems often combine multiple approaches
- Getting the best of both worlds
  - Reactive
  - Sophisticated planning
- One of the most used approaches to do this is using layers
- Upper most layer is deliberative, using some planning approach
- Only calls some *skills* (e.g. GoToPoint, PickUpBall)
- Those are implemented by state machines
- Each state may have an implicit decision tree to decide on the transition

Table of Contents

Machine Learning in Robotics

- Machine Learning
  - non-linear structure extraction from data
  - robust to (sensor) noise
  - requires big data sources
- Intelligent Robotics
  - interprets complex sensor data
  - requires expert roboticists
  - data acquisition in the real world
  - "intelligent behavior" is no well-defined function

Both fields can support each other,
but come with their own challenges

What is Machine Learning

Machine Learning

Machine Learning is a field of computer science that gives computers the ability to perform tasks without being explicitly programmed.

ML addresses the problem of optimizing parameters θ of a parameterized family of functions f_θ, such that an error function $E_f(\theta)$ is minimized.

This includes ...

- the definition of the space of permissible functions f_θ
- the definition of the error function E_f
- appropriate optimization methods

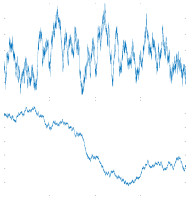
Function Optimization

The relevant optimization problem is defined as estimating

$$\theta^* = \underset{\theta}{argmin}(E_f(\theta))$$

- Any optimization method can be applied
- Most practical methods exploit the gradient ∇E_f
- This is computed
  - Analytically
  - By automatic differentiation
  - Stochastically (particle techniques)

- Smooth(er) gradients allow better optimization



Define the Problem

- Intelligent robots should be able to perform everyday activities

- Most commonsense concepts are hard to formulate in function notation
- Most tasks/goals are hard to define with reasonable gradients

The common approach:

- Instead of trying to solve everything as one problem, many things are programmed and a restricted, but crucial, component is learnt
  - Use traditional robotics to solve what is hard to learn
  - Use ML to solve what is hard to program

Define the Function

Here are some classes of learnable functions:

- label-classification of sensory stimuli
  - understand what is there
- grade quality of candidates
  - predict probability of success
- real-world dynamics for simulation
  - approximate physics instead of modelling
- imitate functions from different inputs (*reparameterization*)
  - compute on RGB images instead of $SE(3)$
- traditional functions from Reinforcement Learning: $Q(s, a)$, $V(s)$, $A(s, a)$, $\pi(s)$
  - choose domains for states (or observations) s and actions a

Table of Contents

Datasets

- Data-driven methods require large amounts of data
- Usually, data-collection takes a lot of time and is thereby expensive
- Widely used datasets allow a comparison between different methods
- Datasets helped accelerate progress in many fields of deep learning
- In some cases, datasets caused problems due to their popularity

Dataset Design

- As universally usable as possible
- Hard for current processing methods
- Preventing overfitting
- Correctly annotated
- Data management becomes major task

Dataset Difficulty

Progress in processing methods requires to adapt the classification challenges

TORSO-21 Dataset

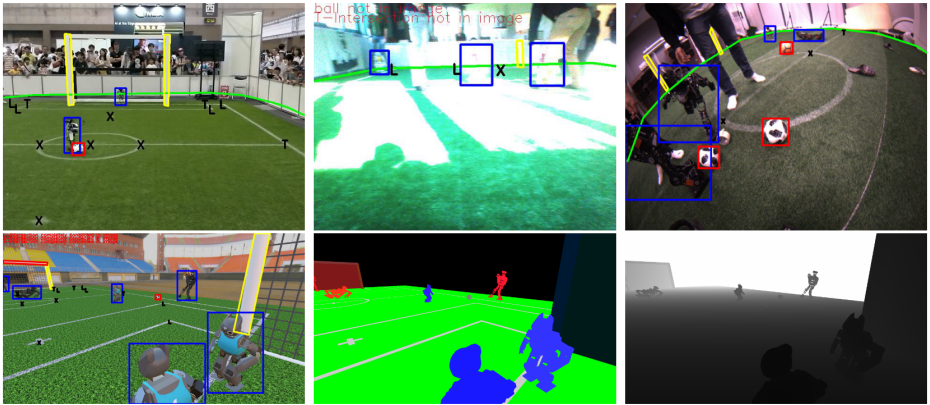


Table of Contents

Image Classification / Class Prediction

. . . is probably the best-understood subfield of applied ML

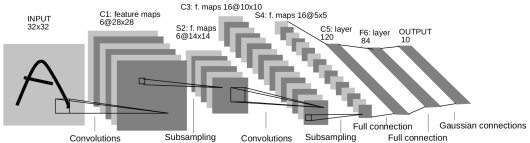
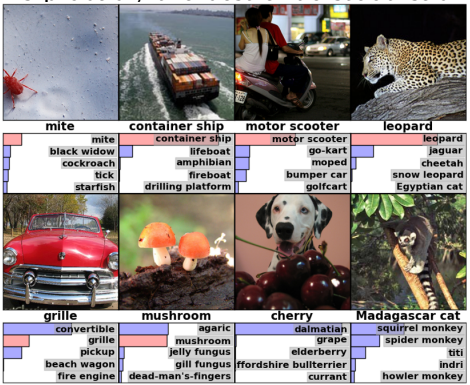
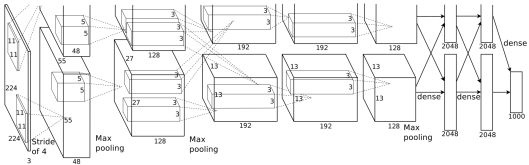


Image Classification / Class Prediction

. . . is probably the best-understood subfield of applied ML

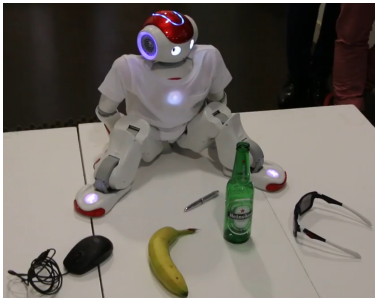




Krizhevsky 2012 / NIPS

Image Classification - How to apply it?

- Classification yields one class label per image
- Robots generate camera streams that can be classified



2016 Social Robotics Hackathon

Image Classification - How to apply it?

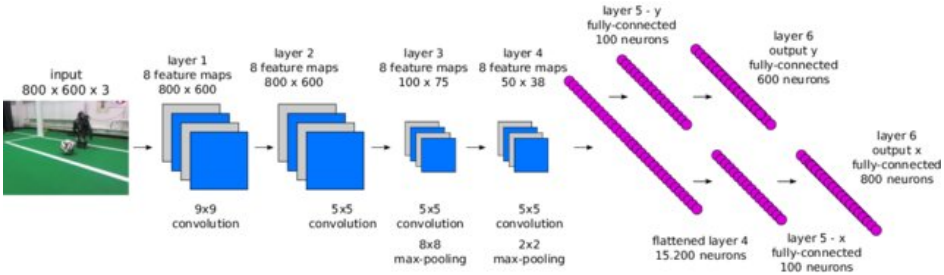
But . . .

- how should the robot respond if it observes the label “table”, “floor”, “human”, or “blue cylinder”?
- useful if the robot *already knows the object's position*
  e.g. preceding segmentation, prior knowledge

- Even if a label is *not* selected, it might be accurate
- Robot camera streams do not have capture bias,
  so the object is usually *not* in the image center
- For most applications *localization* is required

Image Localization - Object Coordinates

- We can train networks to directly get the X,Y image coordinates of an object
- Typically using first convolution layers and then some fully connected ones
- This gives us no information about the object size

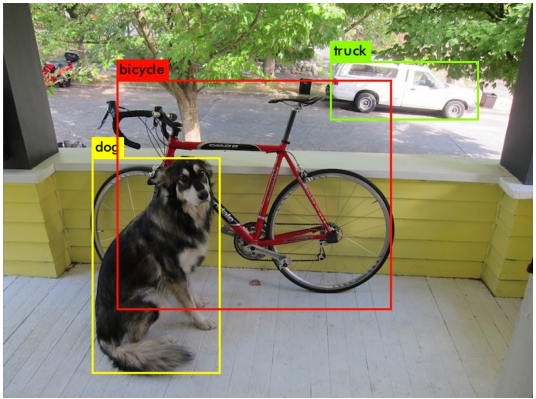


Speck et al., Ball Localization for Robocup Soccer Using Convolutional Neural Networks, RoboCup 2016

Video

Video
https://www.youtube.com/watch?v=buPWpBkR4aU&t=199 (3:20-)

Image Localization - Bounding Boxes



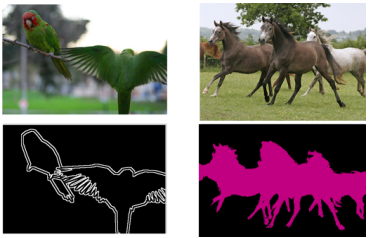
- Box-localization in image space
- Much progress with R-CNN, Fast R-CNN, Faster R-CNN, YOLO
- Bounding boxes do not describe object shape
  - Sufficient to look towards a face
  - Insufficient to pull a door handle

Redmon 2016 / CVPR

Video

BitBots Team Video
`https://www.youtube.com/watch?v=tZnBZsUAVqs&t=7s` (0:07 - 1:00)

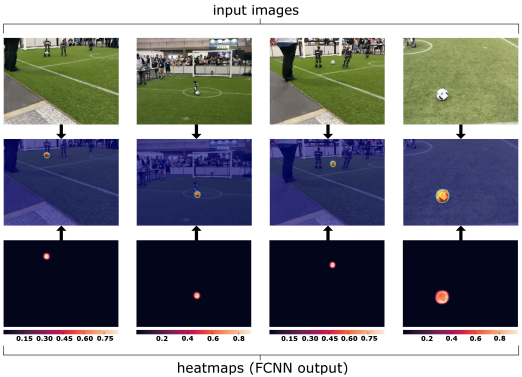
Image Localization - Segmentation



- upsampled conv-net output
- or masks for regions
- too much information
  - object boundaries are usually not pixel-accurate
- too little information

- needs further processing to get coordinates
- a pixel-mask of a known mug does not show the orientation of its handle
- interaction often requires *6D pose estimation*

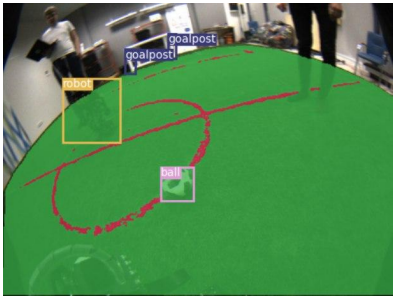
Image Localization - Pixel Labeling - Example



Daniel Speck, Marc Bestmann, Pablo Barros: Towards Real-Time Ball Localization using CNNs, 2018

Combining Bounding Boxes and Segmentation

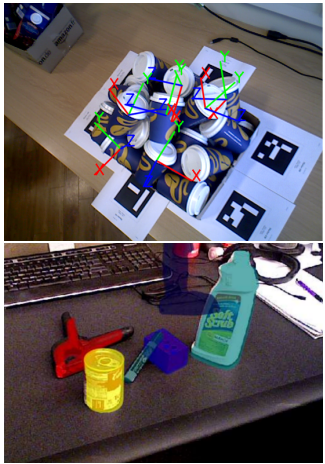
- It can make sense to use both bounding boxes and segmentation in the same task
- These can be generated with the same network to save computation time



Florian Vahl, Jan Gutsche, Marc Bestmann, Jianwei Zhang "YOEO – You Only Encode Once: A CNN for Embedded ObjectDetection and Semantic Segmentation", IEEE ROBIO 2021

Object Pose Estimation

- need to estimate full pose of the observed object
- accurate estimation remains challenging
- successful pipelines often combine
  - visual learning
  - feature matching
  - pose tracking
  - local model fitting
- there are many impressive demos but no general solution



Video

Example video
https://www.youtube.com/watch?v=yVGViBqWtBI

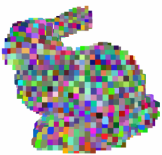
Table of Contents

Depth Data

(a) Mesh

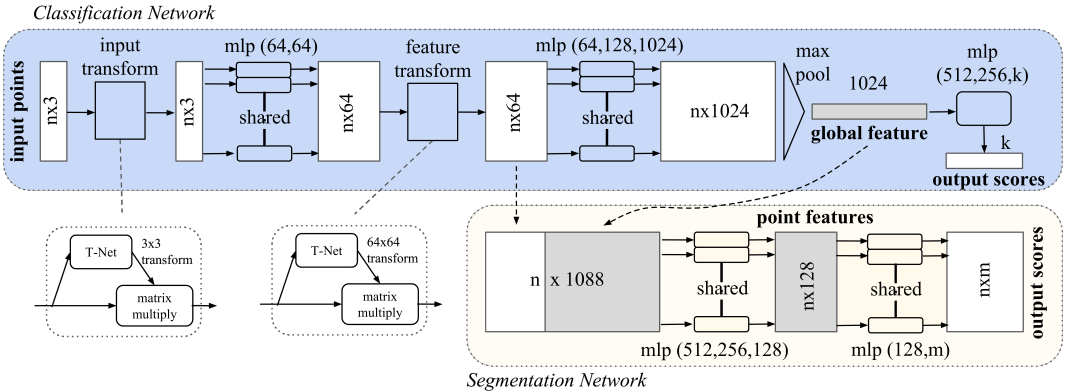
(b) Point Cloud

(c) Depth Image

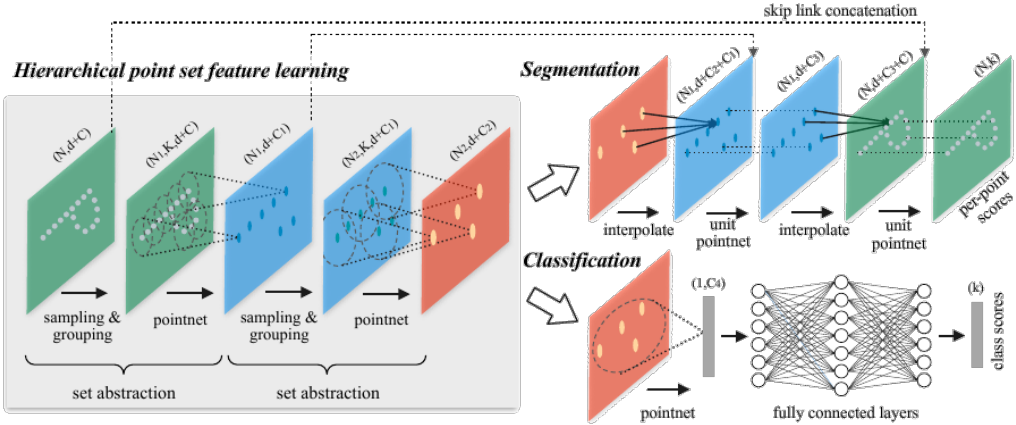


(d) Voxel Grid

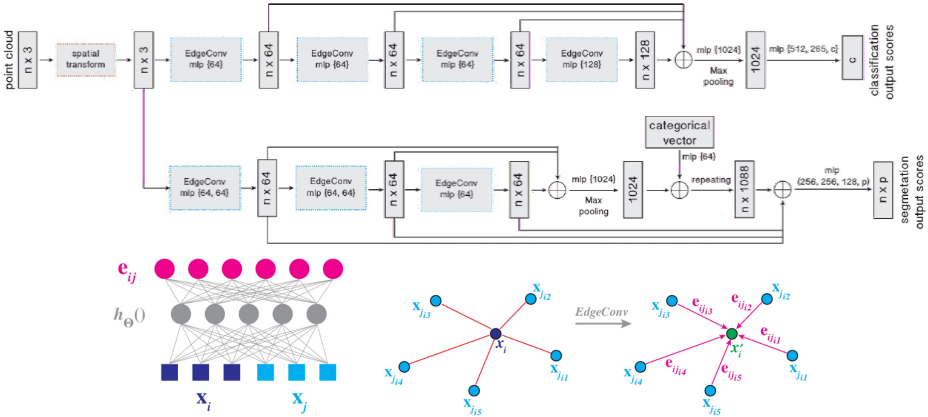
PointNet


Classification Network
input points
nx3
input transform
nx3
mlp (64,64)
shared
nx64
feature transform
nx64
mlp (64,128,1024)
shared
nx1024
max pool
1024
global feature
mlp (512,256,k)
k
output scores
T-Net
3x3 transform
matrix multiply
T-Net
64x64 transform
matrix multiply
point features
n x 1088
shared
nx128
shared
nxm
output scores
mlp (512,256,128)
mlp (128,m)
Segmentation Network

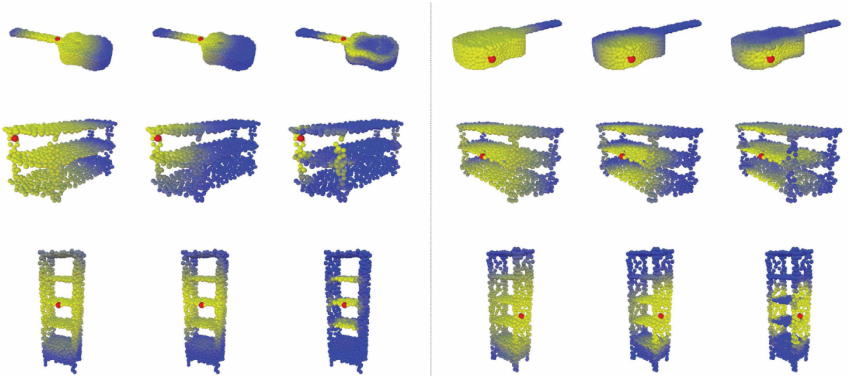
PointNet ++


skip link concatenation
Hierarchical point set feature learning
(N,d+C)
(N1,K,d+C)
(N1,d+C1)
(N2,K,d+C1)
(N2,d+C2)
sampling & grouping
pointnet
sampling & grouping
pointnet
set abstraction
set abstraction
Segmentation
(N1,d+C2+C1)
(N1,d+C2)
(N,d+C3+C)
(N,k)
interpolate
unit pointnet
interpolate
unit pointnet
per-point scores
Classification
(1,C4)
(k)
pointnet
fully connected layers
class scores

DGCNN


point cloud
n x 3
spatial transform
n x 3
EdgeConv mlp {64}
n x 64
EdgeConv mlp {64}
n x 64
EdgeConv mlp {64}
n x 64
EdgeConv mlp {128}
n x 128
mlp {1024}
Max pooling
1024
mlp {512, 256, c}
c
classification output scores
categorical vector
mlp {64}
EdgeConv mlp {64, 64}
n x 64
EdgeConv mlp {64, 64}
n x 64
EdgeConv mlp {64}
n x 64
mlp {1024}
Max pooling
1024
repeating
n x 1088
mlp {256, 256, 128, p}
n x p
segmetation output scores
$\mathbf{e}_{ij}$
$h_\Theta()$
$\mathbf{x}_i$
$\mathbf{x}_j$
EdgeConv

DGCNN



Machine Learning on Point Clouds

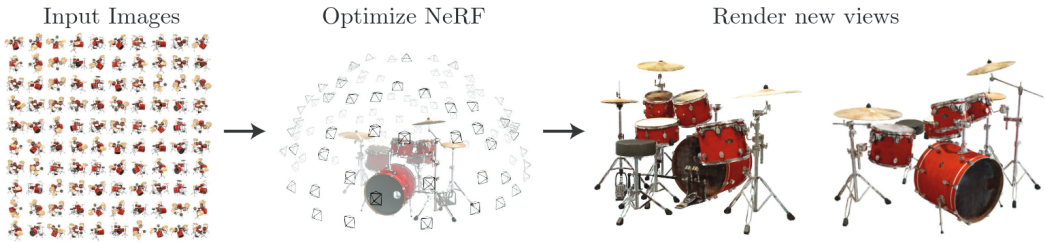
Point cloud processing poses challenges compared to images

- Unstructured, unsorted data
- Inconsistent point density throughout space
- Inconsistent number of input points

Most approaches since PointNet have simmilat tactics in addressing these challenges

- Separate processing of individual points (PointNet)
- Leveraging spacial relations of points with other points in multiple stages (PointNet++ and DGCNN)
- Aggregating extracted features using a pooling operation
  - Invariant towards input order
  - Usually max pooling

NERF



NERF

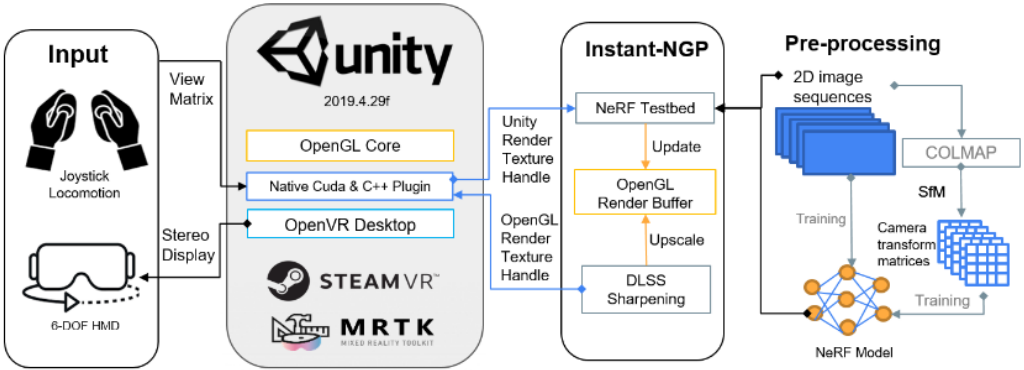

Input
Joystick Locomotion
6-DOF HMD
View Matrix
Stereo Display
unity
2019.4.29f
OpenGL Core
Native Cuda & C++ Plugin
OpenVR Desktop
STEAMVR
MRTK
MIXED REALITY TOOLKIT
Unity Render Texture Handle
OpenGL Render Texture Handle
Instant-NGP
NeRF Testbed
Update
OpenGL Render Buffer
Upscale
DLSS Sharpening
Pre-processing
2D image sequences
COLMAP
SfM
Training
Camera transform matrices
Training
NeRF Model

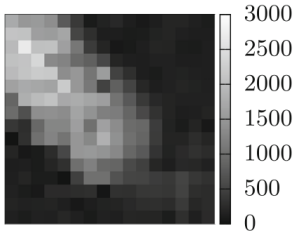
Table of Contents

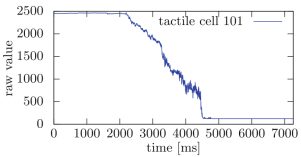
Tactile Classification - Task

- **Task**: detect slippage of held objects online
- differentiate between rotational & translational slippage
- classification for other modalities

Tactile Classification - Method

- measure tactile arrays, 16x16 taxels at 1.9kHz
- crop to center 12x12 taxels
- consider 64ms windows
- compute short time Fourier transforms
- **Input**: 12x12 images with 32 amplitude channels
- convolutional NN with 3-5 layers
- **Output**: $\{stable, translation, rotation\}$
- test accuracy 97.89% at 125 Hz



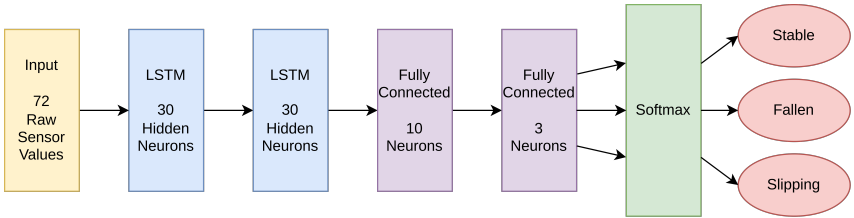


Tactile Classification - Data Acquisition

- user places object between arms, defines rotational / translational slippage
- controllers relax force until slippage occurs
- detect slippage with an orthogonal sensor
- episode runs until no contact is detected
- data augmentation: rotate inputs to account for gravity

Tactile Classification - Slip Detection

- Input: raw values of two 6×6 tactile sensor arrays
- Recurrent layers to take advantage of time component without introducing detection delays
- Classes: Stable, Slipping, Fallen
- Classification accuracy: 92%



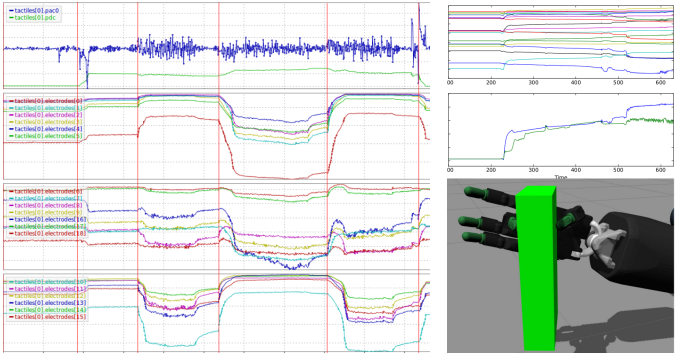
Tactile Classification - Slip Detection

- Input: raw values of two 6×6 tactile sensor arrays
- Recurrent layers to take advantage of time component without introducing detection delays
- Classes: Stable, Slipping, Fallen
- Classification accuracy: 92%

		Prediction		
		Stable	Slip	Fallen
Label	Stable	2119	125	27
	Slip	8	70	29
	Fallen	0	10	243

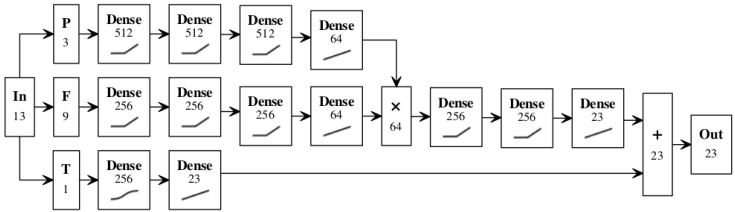
Tactile Simulation - Task

- **Task**: simulate raw readings of a complex tactile sensor



Tactile Simulation - Method

- restrict to single-contact cases
- **Input**: [contact point, 3x force vectors, sensor temperature]
- input is available in physics simulators at each time step
- **Output**: 23 sensor channels



Tactile Simulation - Data Acquisition

- fiducial marker based tracking of sensor and a contact pin
- contact pin mounted on 6-axis force-torque sensor
- compensate for visual inaccuracies by optimizing setup model based on sensor data
- record 300.000 tactile readings with varying contacts

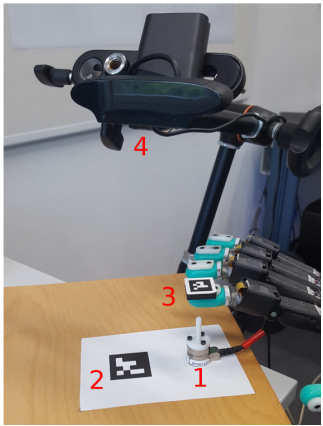
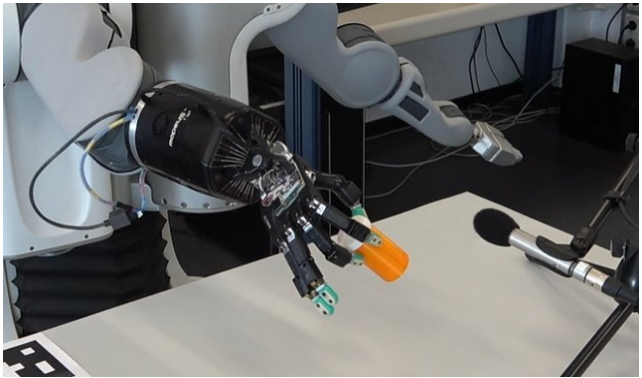


Table of Contents

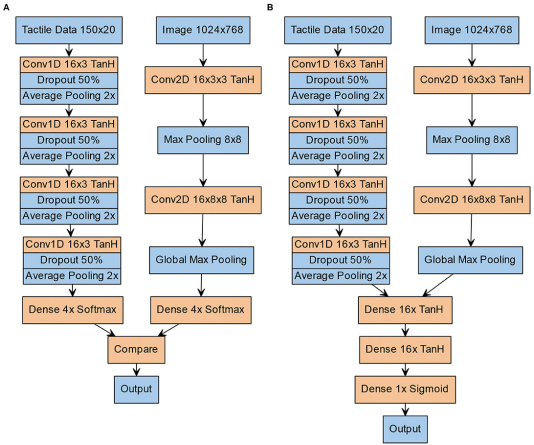
Multimodal Learning

- Robots interact with a multimodal environment
- Usually, only a single modality is used to solve a task
- Using multiple modalities can increase performance and robustness
- Humans constantly process multimodal inputs
  - Tasting food
  - Orientation in space
- Early fusion vs. late fusion

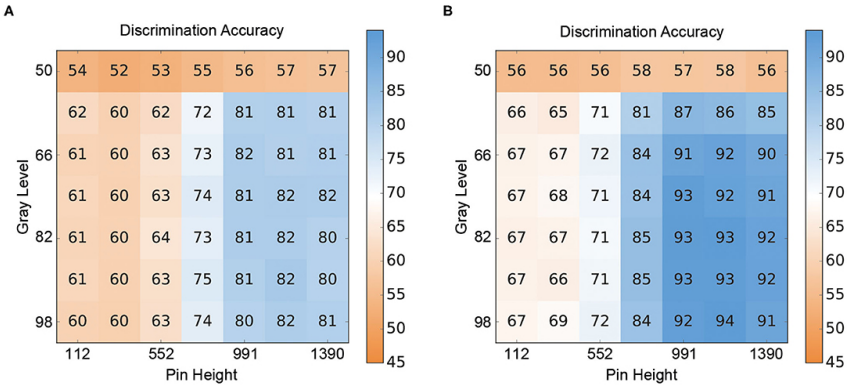
Multimodal Pill Classification



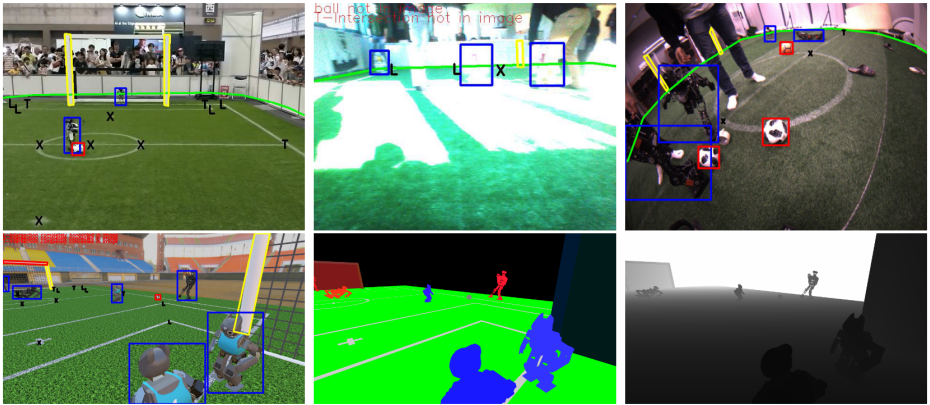
Early Fusion vs. Late Fusion



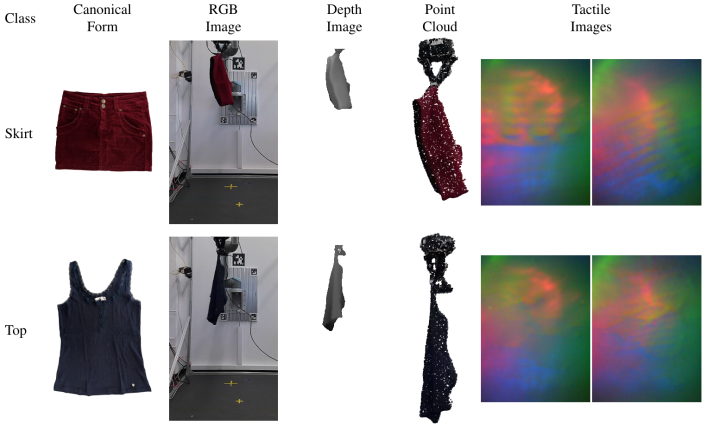
Early Fusion vs. Late Fusion



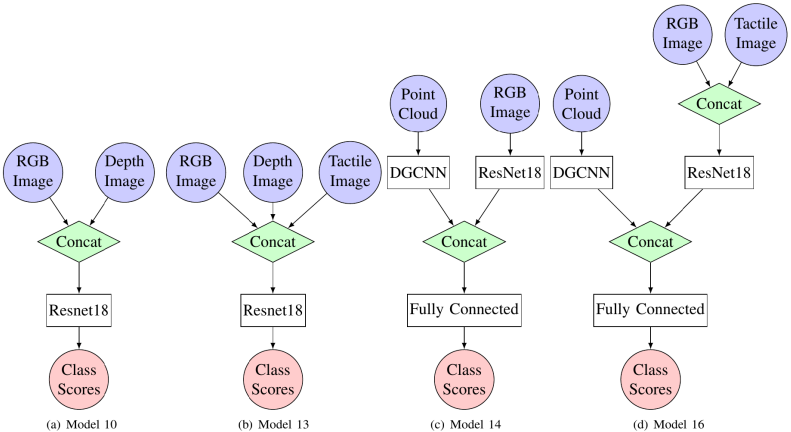
Multimodal Dataset (TORSO21)



Multimodal Dataset (MultimodalClothes)



Multimodal Architectures



(a) Model 10 (b) Model 13 (c) Model 14 (d) Model 16

Crossmodal Learning

- One (or multiple) input modality
- A different (or multiple) output modality

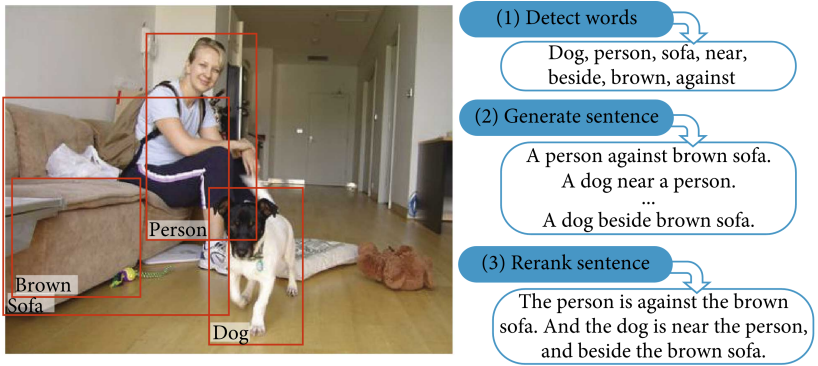


Table of Contents

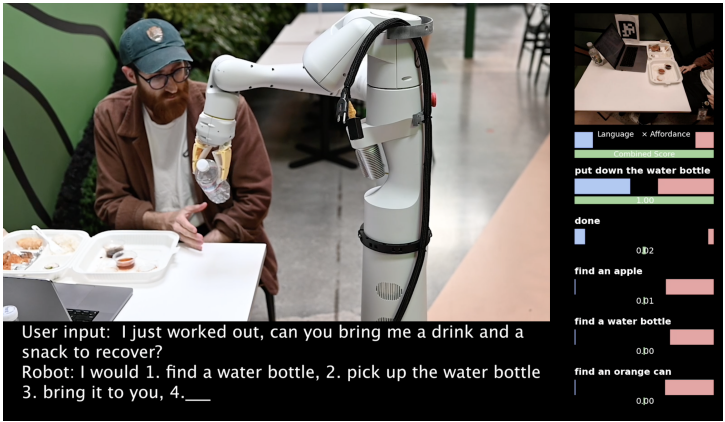
LLMs in Robotics

Recently, a breaktrough with Large Language Models (LLMs) was achieved.

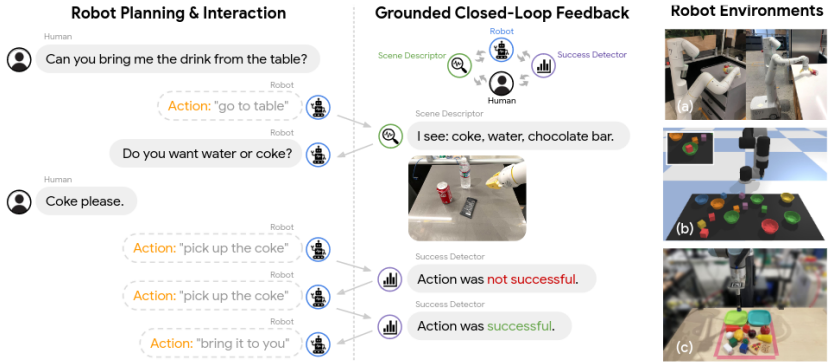
- Mainly an NLP topic
- Huge potential for robotics
- "Hype" topic right now
  - A lot of valid excitement
  - A lot of valid criticism
  - A lot of exaggerations in both directions

Great blogpost on the topic:
`https://hlfshell.ai/posts/llms-and-robotics-papers-2023/`

SayCan


Language × Affordance
Combined Score
put down the water bottle
1.00
done
0.02
find an apple
0.01
find a water bottle
0.00
find an orange can
0.00
User input: I just worked out, can you bring me a drink and a snack to recover?
Robot: I would 1. find a water bottle, 2. pick up the water bottle 3. bring it to you, 4.___

Inner Monologues



Code As Policies

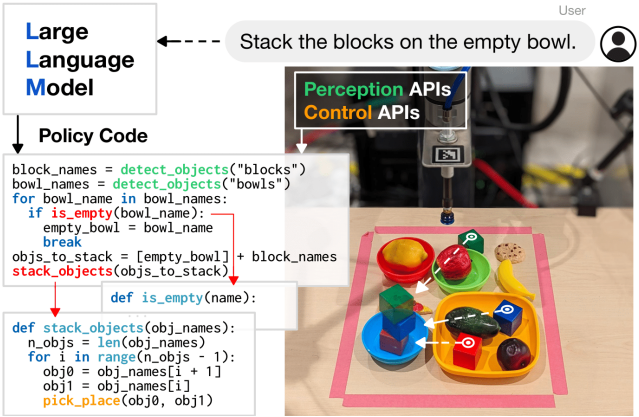


Table of Contents

Grasping Objects

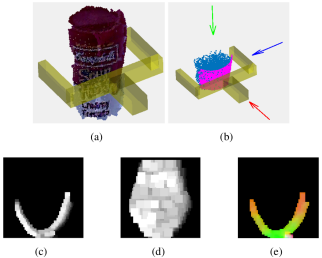
- Picking up objects is a common task for robots
- It is unclear where an object should be grasped
- This is influenced by
  - What kind of gripper does the robot have?
  - How stable would the resulting grasp be?
  - Has the object handle-like structures?
  - Why is the object being picked up?
- If all objects are modeled
  - Feasible grasps can be annotated in models
  - The problem becomes pose estimation
  - If the pose is known, grasps can be looked up
- Most research focuses on 2-finger parallel grippers

Grasp Pose Generator - Task

- **Task**: perform successful 6dof grasps from cluttered scenes of unknown objects
- generate “good” grasp points
- based on RGB-D camera
- learn scoring function for candidate grasps
- 93% grasp success in experiments

Grasp Pose Generator - Method

- sample 6dof grasps, s.t.
  - gripper encloses at least one point
  - gripper is not in collision with the point cloud
- extract multi-image representation from enclosed point cloud and observed/occluded volume



- **Input**: 15 constructed images
- **Output**: grasp probability
- choose best-ranking samples for execution

Grasp Pose Generator - Data Acquisition

- based on object dataset pairing object views & 55 object meshes
- generate random grasp candidates for views
- label based on geometric antipodal force-closure criterium
- simulated data (3DNET) produced inferior results due to simulation discrepancy

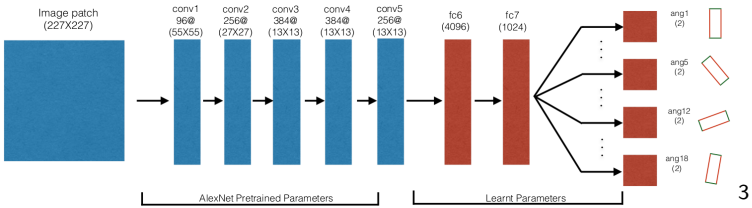
Learning Grasps - Supersizing Self-Supervision

- **Task**: predict 3dof grasp poses (x, y, φ)
- first paper to collect 500h experience for grasping
- collection procedure:
  - Move gripper above sampled ROI
  - Sample grasp point and orientation $\theta \in \{0°, 10°, \dots 170°\}$
  - Pick object and check gripper force sensor for success

[3]

- 66% success rate on test objects
  73% for known objects

Learning Grasps - Supersizing Self-Supervision - Method



3

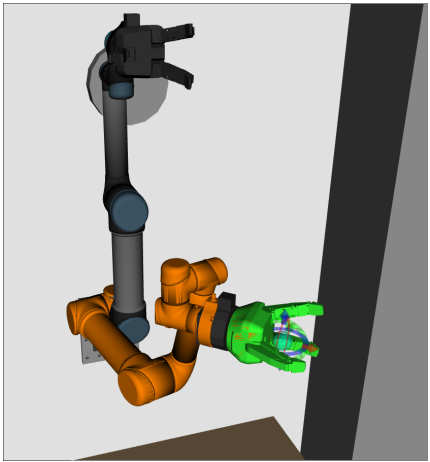
- adapt AlexNet and retrain last layers
- output softmax layer with 18 possible angles φ
- at runtime,
  - sample image patches on ROI
  - evaluate them according to network
  - execute highest-scoring one

Video

Video

Table of Contents

Inverse Kinematics



- IK is one of the oldest problems in robotics
- given a robot kinematic design, find a function that maps the Cartesian workspace of the tool frame to joint configurations
- a single pose often maps to multiple joint configurations
- for many existing robot arms fast analytical solutions exist

Inverse Kinematics

- "well-defined" function
- easy to generate samples
- but ...
  - applications require very high accuracy solutions
    position error $< 10^{-5}$m
  - multiple solutions disrupt training gradient
- training a network for 6dof can succeed
- demonstrated for continuous sub-workspaces with current joint state as input
- online optimization performs better

IK to Absurdum

- Learning IK for 2dof with 2.5cm accuracy

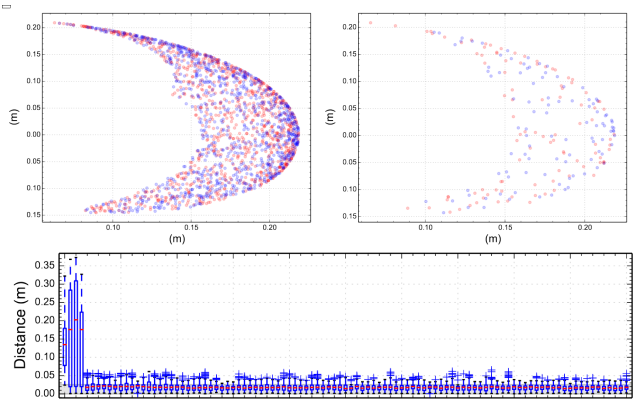
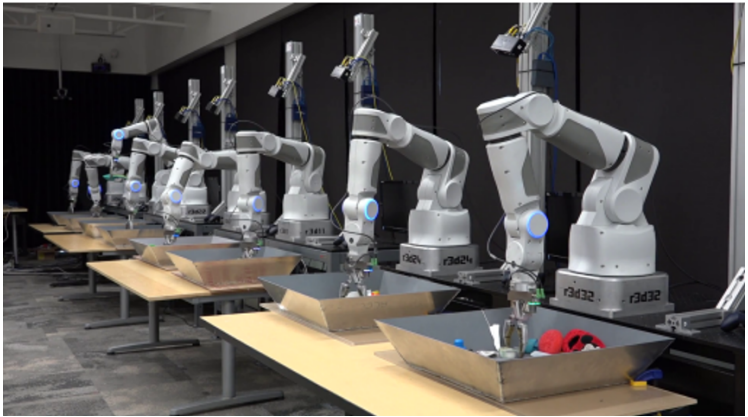
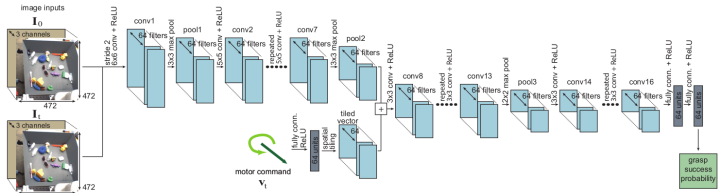


Table of Contents

Learning Hand-Eye Coordination - Task



Learning Hand-Eye Coordination - Method



- **Input**:
  - ***o*** - 2 RGB images of current and first state
  - ***a*** - 3D Cartesian force vector & a twist φ
- **Output**: eventual grasp success ℓ

- the network is trained to predict the eventual grasp success ℓ of the attempt
- online controller runs stochastic search to find action with good prediction

Learning Hand-Eye Coordination - Data Acquisition

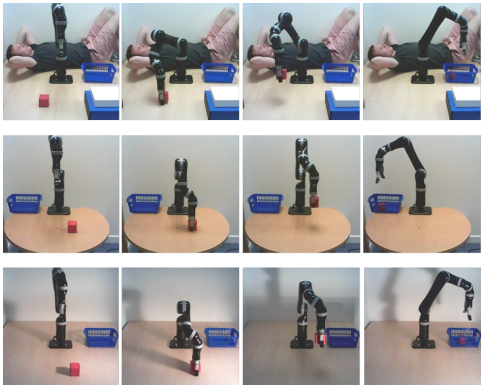
- 6-14 robots running for 2 months
- heuristics to automatically lift arm up if “not promising”
- grasp if at least 90% as successful as move
- servoing with 2-5Hz
- self-supervised exploration for 800.000 grasp attempts
- first 50% random commands (10-30% success already)

A tremendous achievement that works because the learning task is designed to be *as simple as possible*, while it still defines the core behavior.

Video

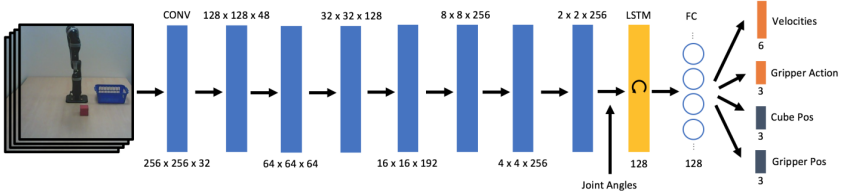
Video

End-to-End Visuomotor Control - Task



- **Task**: move red block into blue basket
- learn visuomotor policy π

End-to-End Visuomotor Control - Method



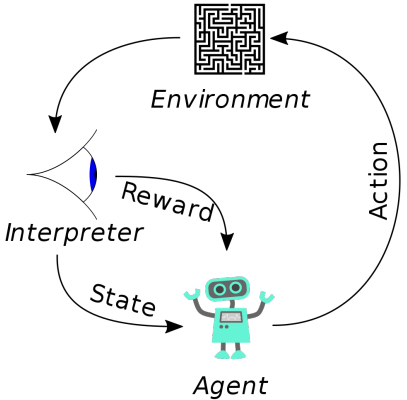
- **Input**: ***o***
  - 4 last RGB images
  - current joint angles
- **Output**: ***a***
  - 6 joint velocities
  - softmax gripper action
  - auxiliary outputs

- learn to imitate straight-line motion planner
- originally sim2real application with domain randomization
- later demonstrated to be trainable in reality (with position targets)

Table of Contents

Reinforcement Learning

- Currently a popular topic in ML
  - Especially regarding robotics/motion
- Advances due to more investment
- and common environments
  - Isaac Gym
  - RoboSchool
  - PyBullet
- and baseline implementations
  - stable_baselines



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

Vanilla PPO

Video:
https://www.youtube.com/watch?v=hx_bgoTF7bs

N. Heess et al, Emergence of Locomotion Behaviours in Rich Environments, 2017

Vanilla PPO - What is Learned

- Action: joint efforts
  - Planar walker: 9 DOF
  - Quadruped: 12 DOF
  - Humanoid: 28 DOF
  - Joints box constrained
- Observation proprioceptive
  - Joint angles and velocities
  - Velocimeter
  - Accelerometer
  - Gyroscope
  - Contact sensors at feet and leg
- Observation exteroceptive
  - Position in relation to center of the track
  - Profile of the terrain ahead

Vanilla PPO - What is Learned (cont.)

- Policy: two networks
  - Only proprioceptive observations
  - Only exteroceptive observations
  - Type of network not clear (probably MLP)
  - Somehow choose action together
- Reward
  - Forward velocity
  - Penalization for torques
  - Stay at center of track

N. Heess et al, Emergence of Locomotion Behaviours in Rich Environments, 2017

Vanilla PPO - How is it Trained

- Mujoco simulation
  - Physical parameters unknown
  - Rate of policy unknown
- PPO - Proximal Policy Optimization
  - Simplified version of TRPO - Trust Region Policy Optimization
  - Current policy is used to choose actions
  - After an episode, advantages of those actions are computed
  - The policy is updated so that good actions become more propable and vice versa
  - Updates are clipped to prevent the policy from leaving the area where it can explore senseful
- Distribution through workers
  - Each worker collects data and computes gradients
  - Batch results are processed by chief
  - New policy is distributed to workers

Deep Mimic

Video:
https://www.youtube.com/watch?v=vppFvq2quQ0

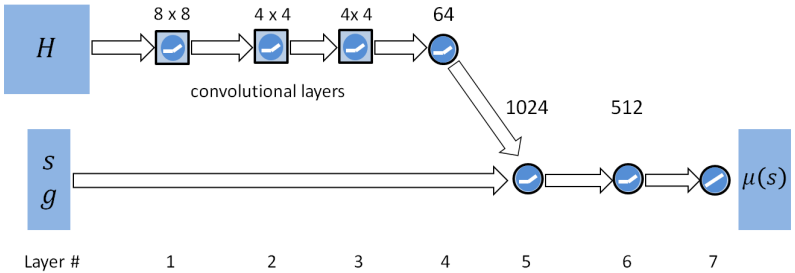
X. Peng et al., DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills, 2018

Deep Mimic - What is Learned

- Action: joint position
  - PD controllers compute effort
- "Observations" (States)
  - Relative pose of links to pelvis
  - Linear and angular velocity of links
  - Phase $\in [0, 1]$
  - Goal g
    - Target heading (walk)
    - Target position (kick, throw)
- Reward
  - $r_t = \omega^I r^I + \omega^G r^G$
  - Closeness to mocap and goal
  - $r^I = w^P r_t^P + w^v r_t^v + w^e r_t^e + w^c r_t^c$
  - Reward based on difference in joint position/velocity, end-effector position, CoM position

Deep Mimic - What is Learned (cont.)

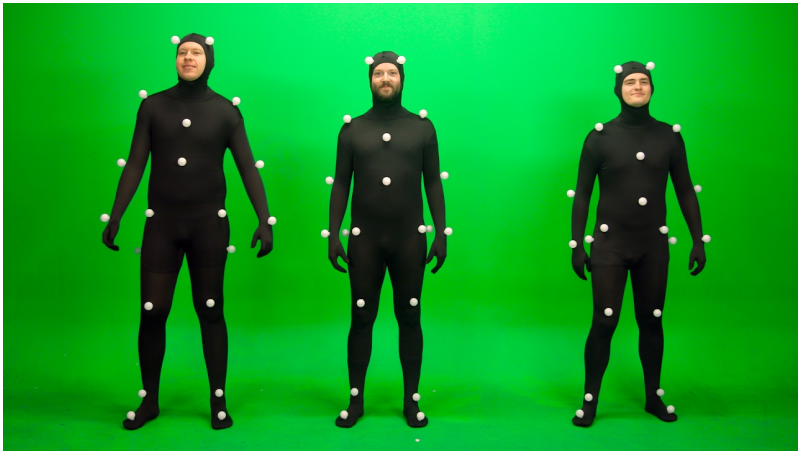
- Policy: single network
  - Input goal and state
  - 2 hidden layer with 1024 and 512 neurons
  - ReLU activation
  - Additional CNN for height map



Deep Mimic - How is it Trained

- Mujoco simulation
  - Physics parameter unknown
  - 30 Hz
- Initial state distribution
  - “Man muss immer umkehren” - Jacobi
  - Easier to learn starting from the back (backplay)
  - Reward clearer when near goal
  - Choosing initial state simple with mocap
- Early termination
  - Terminate episode if condition is reached
  - Classic for walking: head is below certain height
  - Reward for episode is set to zero
  - Further shapes reward function
  - Biases the data distribution to samples which are more favorable

Motion Capture - Small Excursus



Motion Capture - Small Excursus

- Different ways to get the data
  - Infra red reflectors
  - LEDs blinking with different frequencies
  - IMUs on all links
  - Magnetic field based (hall sensor)
  - Exoskeleton measuring angles

Motion Capture - Small Excursus

- Pro
  - Faster learning
  - Less exploits of glitches
  - (Maybe) more useful on actual robot
- Contra
  - Expensive
  - A lot of work
  - Difficult to get data from animals, e.g. a tiger
  - You look kind of stupid while recording
  - Need to find a student which can do a round house kick
  - Need to bring student into hospital after failed round house kick

Skills from Video

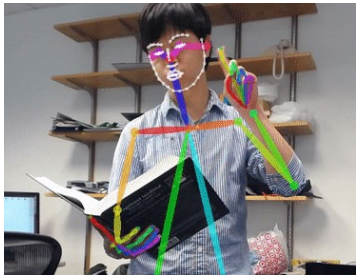
Video:
https://www.youtube.com/watch?v=4Qg5I5vhX7Q

SFV - What is Learned

- Pose estimation
  - 2D and 3D
  - 2 different estimators OpenPose and Human Mesh Recovery
- Motion reconstruction
  - Find optimal motion from single poses
  - Enforce temporal consistency to reduce jitter and glitches
- Learning of motion is similar to DeepMimic, just without goal
- $r = w^P r_t^P + w^v r_t^v + w^e r_t^e + w^c r_t^c$

SFV - How is it Trained

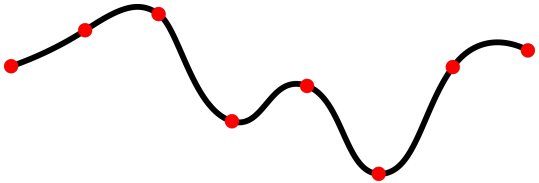
- Pose estimators
  - Supervised learning on single images
- Motion part
  - Similar to DeepMimic



http://www.cs.cmu.edu/~yaser/

DeepQuintic

- Reference motions help but are not optimal for robots
- Use simple open loop engine instead
- Based on quintic splines
- Optimized with multi objective tree-structured Parzen estimator (MOTPE)

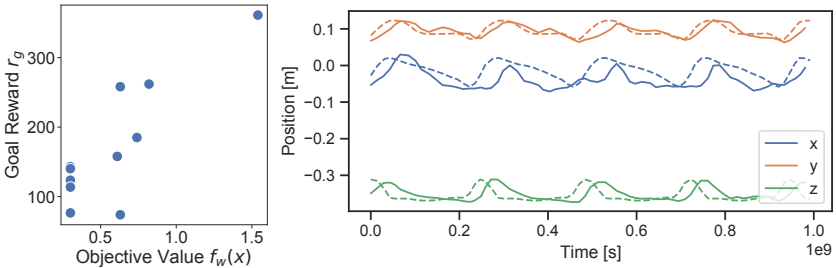


DeepQuintic

Video Pybullet

DeepQuintic

- Resulting walk quality is related to quality of the reference motion



DeepQuintic

Video webots
Video real

Comparison Reference Motion

Motion Capture	Manual Keyframe Animation	Spline-based Engine
Only possible for human or animal like robots	Possible for any kind of robot	Possible for any kind of robot
Transfer to different kinematics needed	Designed for one specific platform	Adapts to different platforms
Reference not executable	Reference may be executable	Reference executable, provides baseline
Not optimal	Difficult to optimize	Can be optimized for the platform
One walk velocity	One walk velocity	Any walk velocity
Represents state	Represents state	Represents Action
Data recording needed	Manual creation needed	Programming needed

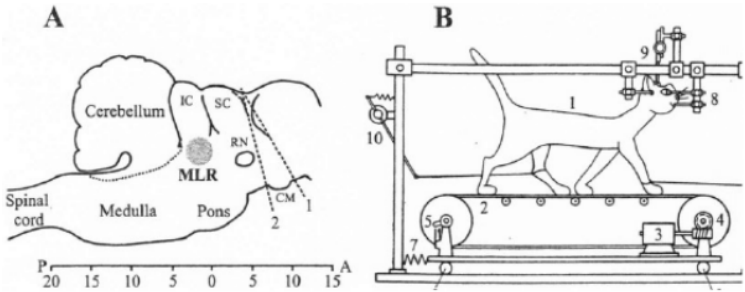
Do We Walk With Our Brain?

- How do humans compute their walking?
- Chickens can run without head
- Legend of Störtebecker

http://www.neurologie.usz.ch/ueber-die-klinik/veranstaltungen/Documents/7_hirnstimulation.pdf

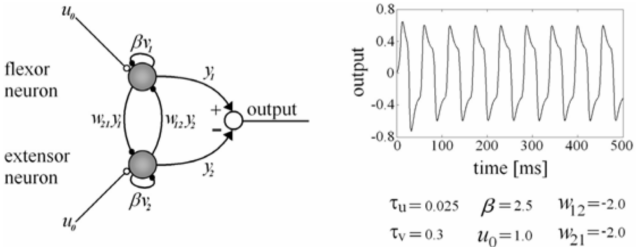
Central Pattern Generators

- Biological neural circuits in the spline
- Generate rhythmic output after being activated
- Used by humans for walking, breathing, swallowing, ...
- Models of this can be implemented for robots



Central Pattern Generators

- Flexor and extensor neuron
- Activated with tonic (non-rhythmic) signal
- Different patterns with different weights



Central Pattern Generators for Gait Generation in Bipedal Robots, Almir Heralic et al., Humanoid Robots, New Developments, 2007

CPG - How is it learned

- It is not!
- Weights are hand crafted
- Learning would be possible either by direct parameter learning or RL

Evolutionary Approach

Video
https://www.youtube.com/watch?v=pgaEE27nsQw

Evolutionary Approach - What is Learned

- Muscle structure of the robot
- Parameters of FSM for leg state
- Target poses
- Force applied in relation to feedback
- Initial pose

Subject	Parameters
Muscle physiology	3–30 *
Muscle geometry	12–39 *
State transition	3
Target features	14
Feedback control	14–63 *
Initial character state	6

Evolutionary Approach - How is it Trained

- Evolution approaches in general
  - Generate initial random population of parameter sets
  - Loop
    - Evaluate individuals based on fitness function
    - Pick best
    - Recombination / mutation
- Covariance matrix adaptation evolution strategy (CMA-ES)
  - Pairwise dependency between parameters is represented by covariance matrix
  - This matrix is updated to increase fitness
  - Good for ill-conditioned functions

Table of Contents

Simulation Downsides - The Reality Gap

- Difference between simulation and reality
- Wrong models
  - Mass, inertia, size
  - Sensor noise non Gaussian
  - Actuator properties not correct
  - Change over time
  - No static values
- Friction / contact
- Soft bodies
- Environment model not correct
  - Changing lighting conditions
  - Cluttered background
  - Non static background

Simulation vs. Reality (cont.)

- Simulation physics not correct
  - Discrete approximation of continuous system
  - Simplifications due to performance bounds
  - Glitches

Bridging the Reality Gap

- Improving simulation accuracy (smaller step size)
  - Not enough to bridge reality gap
- Adding sensor noise
  - Noise is not perfectly Gaussian
  - Needs noise model, which can have errors
- Domain randomization
  - Currently the most used approach
  - Simulated variability in training time to make model generalize
  - Implementation depends on the scenario

J. Tobin et al., Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World, 2017

Learning Dexterity

Video
https://www.youtube.com/watch?v=jwSbzNHGflM

Learning Dexterity - What is Learned

- CNN for object pose detection
  - Based on three camera inputs
- LSTM for finger actions given finger and object pose
- Both networks are concatenated

M. Andrychowicz et al., Learning Dexterous In-Hand Manipulation, 2018

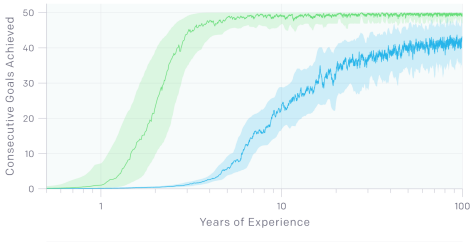
Learning Dexterity - How is it Trained

- PPO
- Domain randomization
  - Object dimensions
  - Object and finger masses
  - Surface friction coefficients
  - Robot joint damping coefficients
  - Actuator controller P term (proportional gain)
  - Joint limits
  - Gravity vector
  - Colors in simulation

Learning Dexterity - Domain Randomization

Learning Dexterity - Impact of Domain Randomization

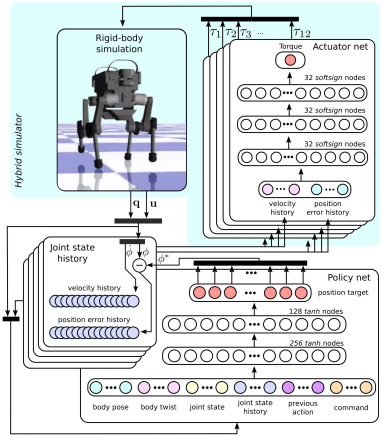
- Median number of successes
  - Without: 0
  - With: 11.5
- Training simulated time
  - Without: 3 years
  - With: 100 years



Learning Dynamic Skills

Video
https://www.youtube.com/watch?v=aTDkYFZFWug

Learning Dynamic Skills - Overview



Learning Dynamic Skills - What is Learned

- Policy Network
  - MLP 2 hidden layers 256, 128 nodes
  - Joint angles, velocities
  - Joint state history
  - Body height estimation (filtered forward kinematics)
  - Body pose, twist (IMU)
  - Previous action
  - Command
- Actuator Network
  - MLP 3 hidden layers with 32 nodes
  - Velocity history
  - Position error history

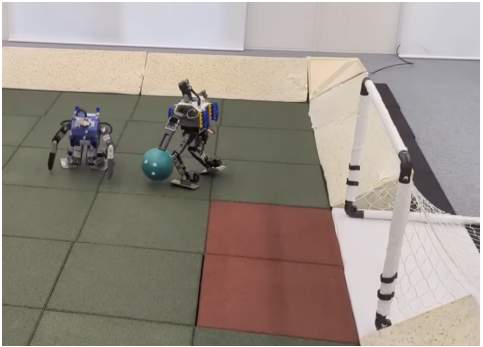
J. Hwangbo et al., Learning agile and dynamic motor skills for legged robots, Science Robotics 2018

Learning Dynamic Skills - How is it Trained

- Randomized simulator model
  - Links masses
  - CoM positions
  - Joint positions
- Policy Network
  - RL with TRPO
- Actuator Network
  - Supervised learning
  - Data collection on robot with simple walk algorithm
  - Joint Position error, Velocity, and Torque

Learning Complex Dynamic Skills - Robot Soccer

- *"Learning Agile Soccer Skills for a Bipedal Robot with Deepj Reinforcement Learning"* by Haarnoja et al. (Deep Mind) `sites.google.com/view/op3-soccer`



Learning Complex Dynamic Skills - Robot Soccer Observations

	Observation	Dimension	Note
Proprioception	*joint positions*	$5 \cdot 20$	Joint positions in radians (stacked last 5 timesteps)
	linear acceleration	$5 \cdot 3$	Linear acceleration from IMU (stacked)
	angular velocity	$5 \cdot 3$	Angular velocity (roll, pitch, yaw) from IMU (stacked)
	gravity	$5 \cdot 3$	Gravity direction, derived from angular velocity using Madgwick filter (stacked)
	previous action	$5 \cdot 20$	Action filter state (stacked)
Game State	*ball position*	2	All positions and velocities are 2-dimensional vectors corresponding to the planar coordinates expressed in the agent's frame. The velocities are derived from positions via finite differentiation. Agent velocity is stacked over five timesteps.
	opponent position	2	
	goal position	2	
	opponent goal position	2	
	agent velocity	$5 \cdot 2$	
	ball velocity	2	
	opponent velocity	2	

Learning Complex Dynamic Skills - Robot Soccer

- Motion skills learned from scratch with reinforcement learning
- Combination of motion skills trained in 1v1 game setting
- Zero-shot transfer to real robots
- Not the official RoboCup environment
- Action space: 20 dimensional joint positions
- Preprocessed input
- Future work: Learning from vision

More Information

- We only had a quick overview, here are some further information
- ML Foundations
  - Machine learning lecture next semester
  - Arxiv Insights - Youtube channel
  - R. Sutton - Reinforcement Learning, an Introduction (free)
  - Berkeley - Deep RL http://rail.eecs.berkeley.edu/deeprlcourse/
- Current advances
  - reddit.com/r/MachineLearning
  - CORL conference (open access)
- If you have some good sources, tell me!

Table of Contents

Summary

- Intelligent Robotics encompasses many domains that can profit from ML applications, including
  - images
  - tactile data
  - inverse kinematics
  - grasp estimation
  - behavior policies
  - ...
- (uncorrelated) data is scarce, because it is collected at runtime
- to train in practice tasks require many simplifications and resources or simulation
- successful approaches reduce the learning problem
- the resulting modules can be very robust and successful

Discussion

What would you teach a robot?

Masterproject
Thesis

Outline

Table of Contents

Disclaimer

- I am not an expert in ethics or philosophy
- Aims of this lecture
  - Give you perspective on the impact of your work
  - Make you reflect on what you're doing
  - Foster a debate rather than just present something
- Not aims of this lecture
  - Tell you what is right or wrong
  - Make you follow my opinion

What is "Ethics"?

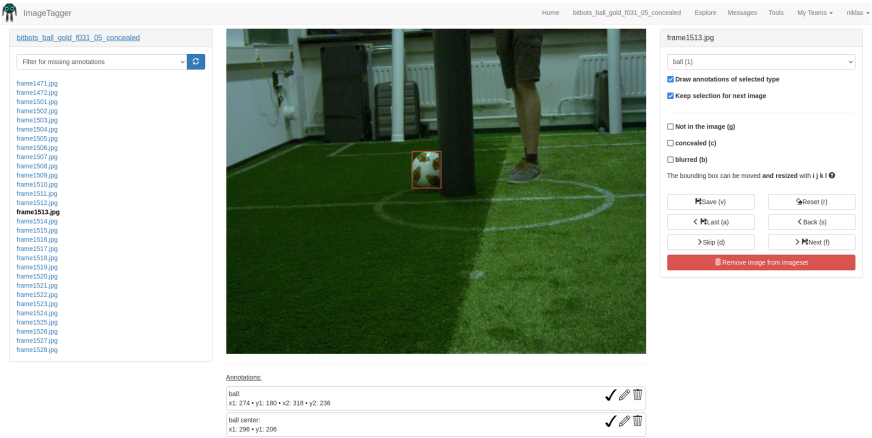
- "A set of concepts and principles that guide us in determining what behavior helps or harms sentient creatures" - R.W.Paul and L.Elder
- "The science of moral duty" - R.Kidder
- A branch of philosophy that involves systematizing, defending, and recommending concepts of right and wrong conduct.
- What is good/evil, right/wrong, justice/crime
- Morality is believed to be developed by evolution

Internet Encyclopedia of Philosophy

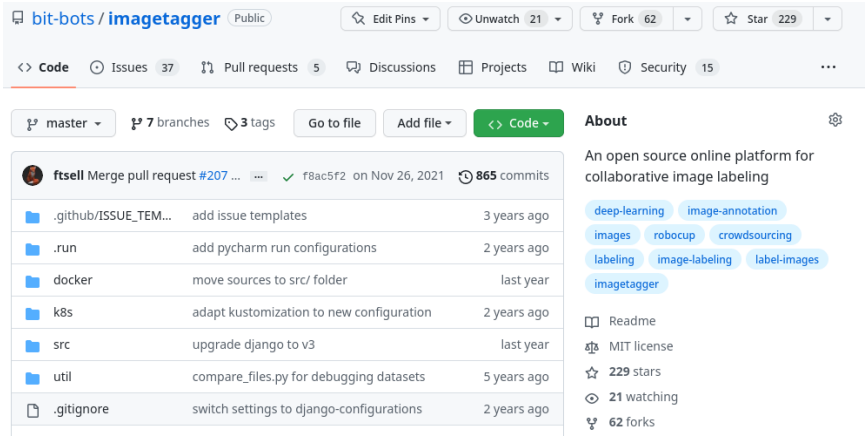
Why Should I Care?

- "Ethics and morality are about knowing and doing the right thing and, by definition, doing the right thing is the right thing to do." - John Cahill
- "With ethics and morality, we ensure a society where life is easier and more pleasant for all." - Benoît Leblanc
- Think before you act to prevent regret. Some examples:
  - Ethan Zuckerman, creator of the pop-up ad
  - Mikhail Kalashnikov, creator of the AK-47
  - Alfred Nobel, inventor of dynamite
  - Dong Nguyen, creator of Flappy Bird
  - Robert Propst, inventor of the cubicle
  - ...
- Ethic misconducts can have dire consequences on your carrier
  - e.g. Jan Hendrik Schön

Why Should I Care? - Personal Example



Why Should I Care? - Personal Example



Why Should I Care? - Personal Example

Question about "Annotate the whole set" #214

Closed **robbine** opened this issue on Apr 24, 2022 · 1 comment

robbine commented on Apr 24, 2022

I'm try to annotate a new dataset which contains army images, and have two annotation types, namly 'soldier' and 'gun'.
Do I need to first select 'soldier' from dropdown box of 'Not in image annotation type' to annotate all images once.
And then select 'gun' from dropdown box to annotate all over again ?
Moreover, how to create negative dataset quickly by doing this ?

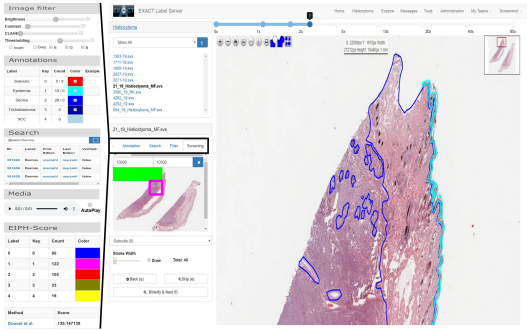
NFiedler commented on Apr 24, 2022 Member

We neither endorse nor support military usage.

1

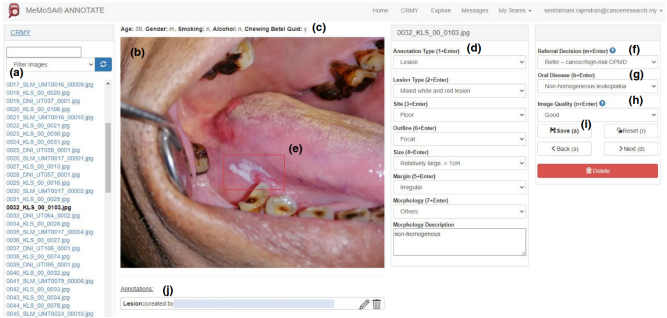
NFiedler closed this as completed on Apr 24, 2022

Why Should I Care? - Personal Example



Marzahl, Christian, et al. "EXACT: a collaboration toolset for algorithm-aided annotation of images with annotation version control." Scientific reports 11.1 (2021)

Why Should I Care? - Personal Example



Rajendran, Senthilmani, et al. "Image collection and annotation platforms to establish a multi-source database of oral lesions." Oral Diseases (2022).

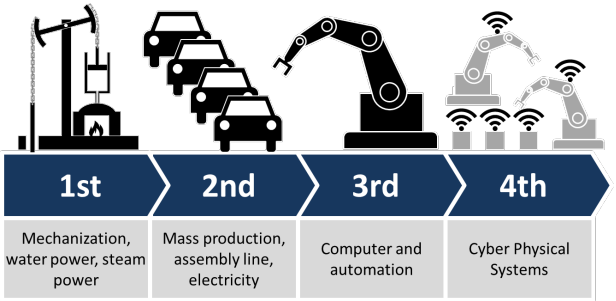
Table of Contents

Future of Work

Future of Work

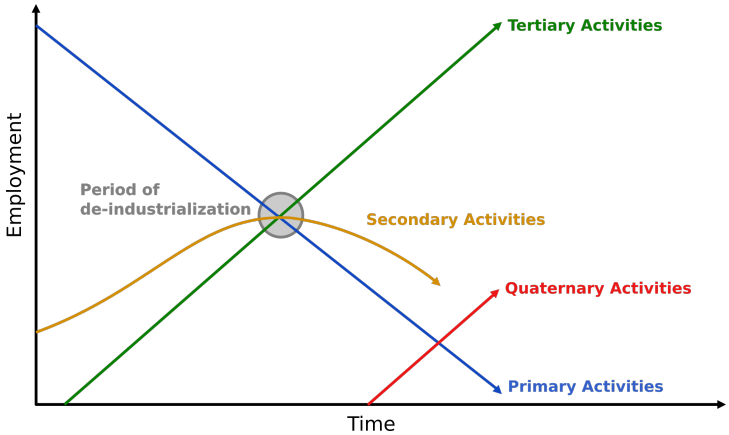
History

- Started with the industrial revolution in England
  - 1769 - Steam engine
  - 1786 - Automatic loom

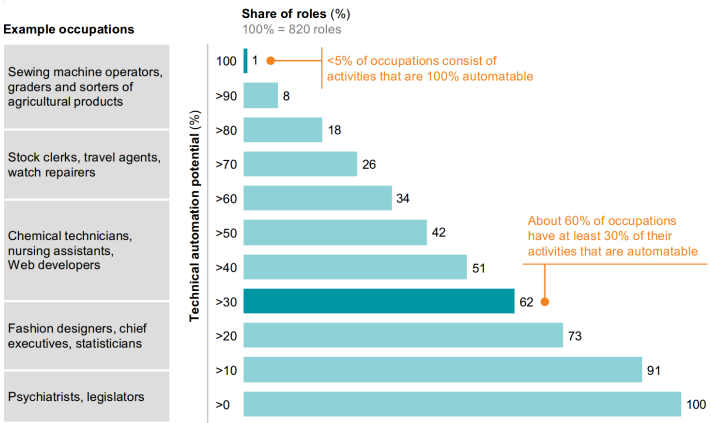


https://en.wikipedia.org/wiki/Industry_4.0

Clark's Sector Model



Current State



1 We define automation potential according to the work activities that can be automated by adapting currently demonstrated technology.

Current State

- Current technology enough to replace some jobs (partly)
- Degree of automation highly depends on country
- Primary sector
  - Semi-automatic planters / harvesters (depending on crop)
  - Food processing in assembly lines
- Secondary sector
  - High level of automation for mass products
  - Low flexibility of factories
- Tertiary sector
  - Most automation done by computer programs
  - Non digital part of work difficult to automate
- Quaternary sector
  - (Almost) no physical work
  - (Almost) no application for robots

Possible Influence of Intelligent Robots

- Intelligent robots may replace more jobs in the future
- Especially in tertiary sector
- Service robots
  - Cleaning
  - Caring
  - Cooking
- Autonomous cars
  - Logistic drivers
  - Taxis
  - Public transport
- Agricultural robots
  - Completely autonomous planters/harvester
  - Automation for difficult crops
  - Organic farming (automated weed picking)

Possible Influence of Intelligent Robots (cont.)

- Manufacturing
  - More flexible assembly lines
  - Higher automation for non mass product
  - Personalized products

Question to Discuss

Should we further automate work
or stop at some point?

Would You Like To Know More?

- European Group on Ethics in Science and New Technologies: Future of Work, Future of Society

Table of Contents

Lethal Autonomous Weapons

Lethal Autonomous Weapons

Definition

- Different names
  - Lethal autonomous weapon (LAWs)
  - Lethal autonomous weapon systems (LAWS)
  - Lethal autonomous robots (LARs)
  - Robotic weapons
  - Killer robots
- Different definitions of autonomy
- Human in/on/out the loop
  - In: wait for human commands to shoot
  - On: human can stop the system
  - Out: no human supervision
- Differentiation between defensive and offensive weapons

History

- Mines were first modern automatically triggered weapon
  - Land mines 1600s
  - Naval mines 1700s
- Springguns ("Selbstschussanlagen") were used to secure the German-German border
- Anti-personnel mines are banned in many countries since 1997
  - Not including USA, Russia, most of Asia and Middle East
- Mine fields still present in many countries
  - E.g. Croatia and even Germany

Current State

- Many different active protection systems
- Radar guided guns (since 1980)
  - Installed on ships and tanks
- Human on the loop to react quickly to missiles

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

Current State

- Sentry guns
  - Stationary deployed to defend area / border
  - First use at Korean-Korean border Samsung SGR-A1 (2006)
  - Human on the loop
  - Sensors
    - Laser rangefinder
    - Infrared camera
    - IR illuminator
  - Weapons
    - Light machine gun
    - Grenade launcher

https://en.wikipedia.org/wiki/SGR-A1

Current State

- Iron Dome
- Radar based anti missiles defense system of Israel
- Human in the loop
- Effectiveness unclear
  - Israeli officials: 75-95%
  - T. Postol et al: 5%

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

Clay Dillow, "How Israel's 'Iron Dome' Knocks Almost Every Incoming Missile Out Of The Sky", Popular Science

Postol, Theodore A (15 July 2014). "An Explanation of the Evidence of Weaknesses in the Iron Dome Defense System". MIT Technology Review

Current State

- Ukraine passenger jet shot down by Iran anti-air missile
- 9K330 Tor-M1 system (similar to Iron Dome)
- According to Iranian military:
  - System graded it as incoming rocket
  - Human in the loop had 10s to decide
  - Gave system okay
- Now major political implications
- 176 civilians dead

Current State

- Many different partly autonomous drones
- In various sizes
- Mostly still controlled by human operator
- Widely used in conflicts and covered operations

http://mydronelab.com/blog/types-of-military-drones.html

Current State

https://engineerine.com/ukraine-is-building-diy-drones/

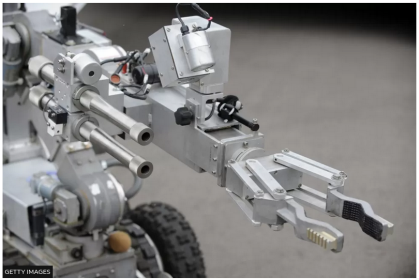
Current State

- Autonomous weapons seem to be in research. Allegedly in
  - China, Israel, Russia, South Korea, UK, US, Turkey, Iran
  - Most of the information about this is classified, no clear sources
- Impossible to discern between truth and propaganda

In the News

San Francisco to allow police 'killer robots'

30 November 2022



A bomb disposal robot extends its arm

https://www.bbc.com/news/technology-63816454

In the News

Amid public outcry, San Francisco officials reverse course and reject police use of robots to kill

By Taylor Romine and Aya Elamroussi, CNN
Published 1:33 AM EST, Wed December 7, 2022

Jeff Chiu/AP

A woman holds up a sign while taking part in a demonstration about the use of robots by the San Francisco Police Department outside of City Hall Monday.

https://edition.cnn.com/2022/12/07/us/san-francisco-rejects-police-controlled-robots/index.html

Question to Discuss

How would you have voted?

Question to Discuss

Should we ban lethal autonomous weapons?
Or some kinds of them?

Would You Like To Know More?

- International Committee for Robot Arms Control `http://icrac.net` (not updated recently)
- `https://autonomousweapons.org`
- `https://stopkillerrobots.org`
- “Slippery Slope - The arms industry and increasingly autonomous weapons”, PAX

Table of Contents

Further Things to Discuss

- Dual use
- Military research funding
- Surveillance by robots
- Open source / open hardware robots
- Android robots
- Cyborgs
- Moral decision making (e.g. trolley problem)

Dual Use Example

ROBOT / TECH

Robot makers including Boston Dynamics pledge not to weaponize their creations

A quadrupedal robot armed with a machine gun built by Ghost Robotics, which has not signed the pledge. Photo by Chris Jung/NurPhoto via Getty Images

/ Five robot companies have signed a pledge. But it won't stop the adoption of robots by the military and law enforcement.

By JAMES VINCENT
Oct 7, 2022, 10:49 AM GMT+2 | 9 Comments / 9 New

https://www.theverge.com/2022/10/7/23392342/boston-dynamics-robot-makers-pledge-not-to-weaponize

Further Information About Ethics in IT

- Academic
  - Lectures by workgroup “Ethics in Information Technology (EIT)”
  - The European Group on Ethics in Science and New Technologies (EGE)
- Popular
  - Black Mirror

You have more interesting sources? Please tell me!