(Probabilistic) Robotics

Artificial intelligence (EDAP01)
Lecture 10
2020-02-19
Elin A. Topp

Course book (chapters 15 and 25), images & movies from various sources, and original material
(Some images and all movies will be removed for the uploaded PDF)

Outline

AI in Robotics - integrating the "brain" into the "body"

Today (probabilistic methods):

- Probabilistic methods for Mapping & Localisation (brief recap on filtering / smoothing)
- SJPDAFs for person tracking
- Identifying interaction patterns in Human Augmented Mapping with BNs

Next lecture (applications of AI methods in robotic systems):

- Knowledge representation, reasoning, and NLP to support HRI and high-level robot programming
- Deliberation & High level decision making and planning
- (Reinforcement) Learning in robotics

But first:

- Some intro to robot(ic)s

What is a "Robot"?

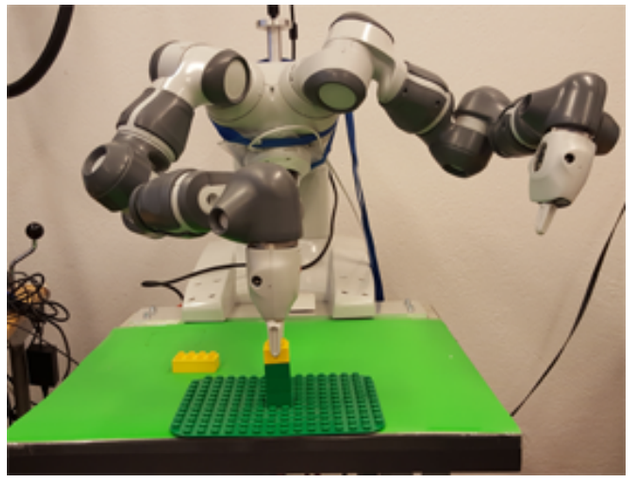
✓

✓

✓

Honda Asimo

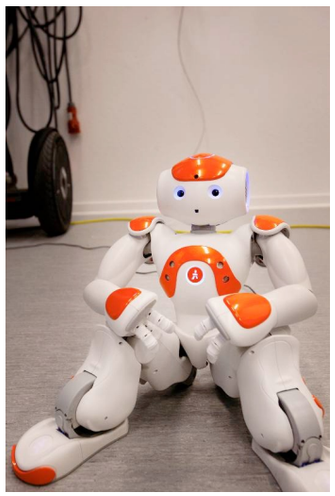
✓



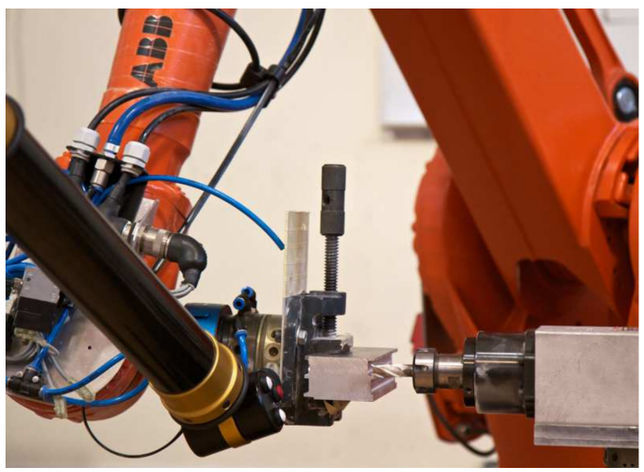
✓

Keepon

?



✓



✓

Leonardo (MIT)

?

iCub (IIT)

✓

• • •

Robots, and what they can do…

How far have we come?

Robots, and what they can do…

How far have we come?

Frida "feels" when work's done... 2013 (Magnus Linderoth, LU)

YuMi wraps gifts… 2015 (Maj Stenmark & Andreas Stolt, LU & Cognibotics AB)

Anyone can program robots… right? 2017 (Maj Stenmark, Elin A. Topp, Cognibotics & LU)

YuMi has learned to feel when things click… 2018 (Martin Karlsson, LU)

Types of robots

Industrial robots vs. service robots vs. personal robots / robot toys

Static manipulators vs. mobile platforms (vs. mobile manipulators)

Mechanistic vs. humanoid / bio-inspired / creature-like

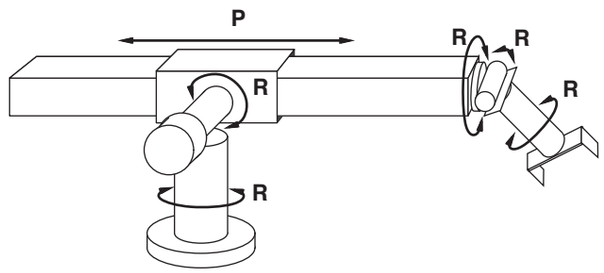
For all in common:
A robot is a (physical) agent in the (physical) world
(with all the consequences that might have... ;-)

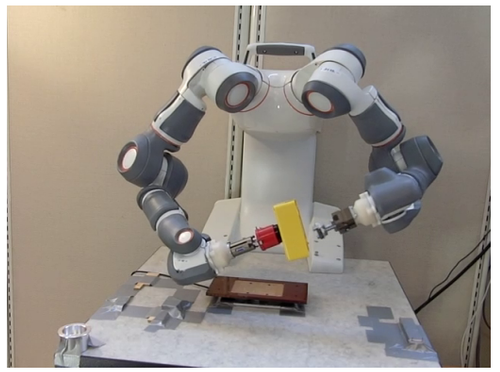
Darpa Urban Challenge 2007
Sting racing crash

Impression from user study
(Maj Stenmark, Elin A. Topp)

Robot actuators - joints and wheels

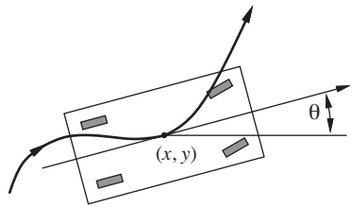
6 DOF (6 "joint") arm:





2x7 DOF ("humanoid" torso "YuMi" / Frida):

2 (3 effective) DOF synchro drive (car):



2 (3 effective) DOF differential drive (Pioneer p3dx):

3 DOF holonomic drive ("shopping cart", DLR's Justin):

Kinematics - controlling the DOFs

Direct (forward) kinematics (relatively simple):

Where do I get with a certain configuration of parts / wheel movement?

Inverse kinematics (less simple, but more interesting):

How do I have to control joints and wheels to reach a certain point?

Dynamics - controlling consequences of movement

Dynamics:

Make the robot move (and move stuff) without falling apart, or crashing into things

- How much payload is possible?
- How fast can I move without tipping over?
- What is my braking distance?
- How do I move smoothly? (ask the automatic control people ;-)

Weight: ca 1300 kg

Payload: ca 150 kg

Impression from user study
(Elin A. Topp)

Dynamics in practice

Dynamics also gets you into two problems: direct and inverse dynamics.

Direct dynamics:

Given masses, external forces, position, velocities and acceleration in the joints / wheels, what forces / moments are put to the depending joints and the tool centre point (TCP)? "Rather" simply solvable, at least more or less straight forward.

Inverse dynamics (again, more interesting than direct dynamics):

While solving the *inverse kinematics* problem is nasty, but still "only" a bunch of linear equations, solving the *inverse dynamics* problem leaves you with a bunch of more or less complex differential equations.

Supporting parts: Sensors

In a predictable world, we do not need perception, but good planning and programming

As the world is somewhat unpredictable, some perception is useful, i.e. robots / robot installations need sensors.

Passive / active sensors.

Range / colour / intensity / force / direction ...

Optical / sound / radar / smell / touch ...

Most common for mobile robots: position (encoders / GPS), range (ultrasound or laser range finder), image (colour/intensity), sound

Most common for manipulators: position (encoders), force / torque, images, (range - infrared, laser RF)

Sensors on a mobile robot

Microphones (sound)

Ultrasound (24 emitters / receivers) (range)

Camera (image - colour / intensity)

Laser range finder (SICK LMS 200) (range)

Infrared (range / interruption)

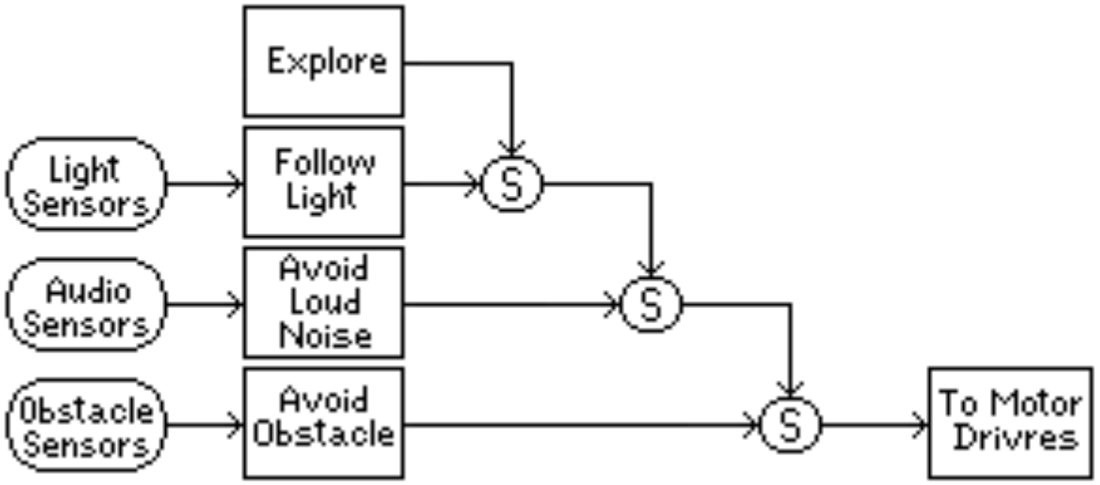
Bumpers (touch)

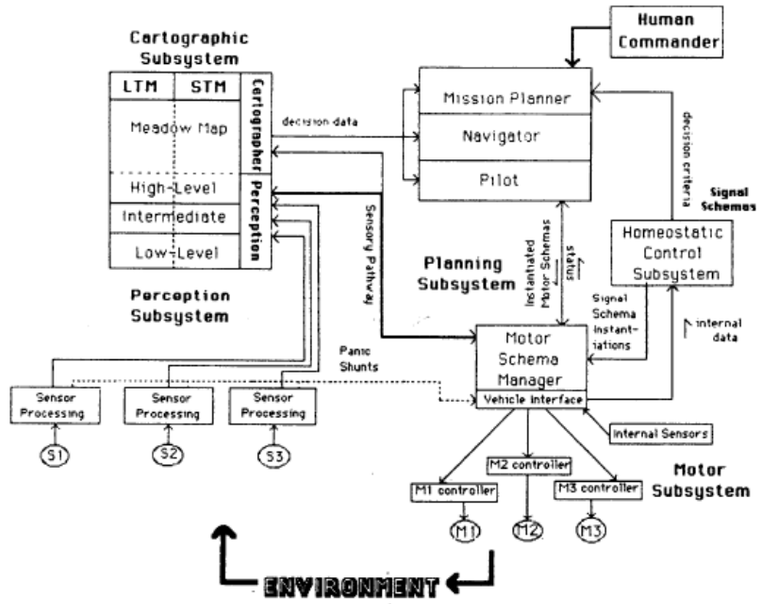
Wheel encoders (position / pose)

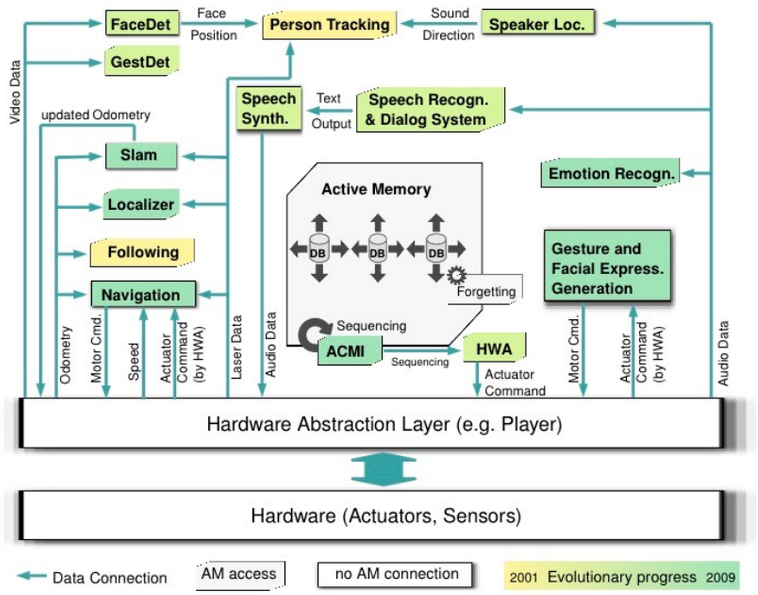
System integration

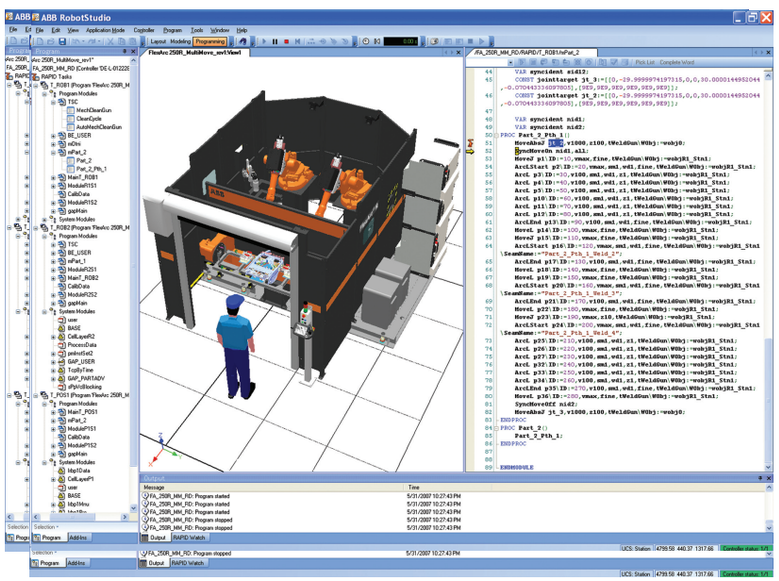
Make sensors, actuators and algorithms work together

Architectures, "operating systems", controllers, programming tools ...









Some fundamental AI challenges in robotics - or should we say robotics challenges in AI?

We need to describe and control our robots' movement to make them do something

We need to either plan and program every little detail of their movement
OR
We need to equip them with capabilities for perception, understanding, insight, deliberation - but when we do that, we need to "plan and program" for that

How can a mobile platform know how to go to place X, when it does not know where X is - and what its own position is? (Describe the world)

How can a two-armed robot know how to glue two work-pieces together, if it does not know what the work-pieces are, where to find them, how to grasp them, how to direct them towards each other, or how to glue stuff together? (Describe tasks)

How can a robot understand that a correctly planned action did not result in the foreseen outcome? What should it do to resolve this situation? (Describe problems)

Outline

AI in Robotics - integrating the "brain" into the "body"

Probabilistic methods:

- Recap: Filtering and smoothing
- Probabilistic methods for Mapping & Localisation (with brief recap on filtering / smoothing)

Filtering:
Prediction & update (FORWARD-step)

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = f(\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t}), \mathbf{e}_{t+1}) = \mathbf{f}_{1:t+1}$$

$= \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, \mathbf{e}_{t+1})$ (decompose)

$= \alpha\ \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ (Bayes' Rule)

$= \alpha\ \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\ \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ (1. update under Markov assumption (sensor model), 2. one-step prediction)

$= \alpha\ \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, \mathbf{e}_{1:t})\, P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$ (sum over atomic events for $\mathbf{X}$)

$= \alpha\ \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)\, P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$ (Markov assumption)

$\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ ("forward message", propagated recursively

$\mathbf{f}_{1:t+1} = \alpha\ FORWARD(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$ through "forward step function")

$\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{X}_0)$

Smoothing:
"explaining" backward

$\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) = fb(\mathbf{X}_k, \mathbf{e}_{1:k}, \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k))$ *with* $0 \leq k < t$ (understand the past from the recent past)

$= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t})$ (decompose)

$= \alpha\ \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\ \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k})$ (Bayes' Rule)

$= \alpha\ \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\ \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)$ (Markov assumption)

$= \alpha\ \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}$ (forward-message x backward-message)

with x indicating componentwise (pointwise, cf course book, page 574) multiplication

Smoothing: calculating backward message

$\boldsymbol{b}_{k+1:t} = \mathbb{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k)$

$= \sum_{\boldsymbol{x}_{k+1}} \mathbb{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k, \boldsymbol{x}_{k+1})\ \mathbb{P}(\boldsymbol{x}_{k+1} \mid \boldsymbol{X}_k)$ (conditioning on $\boldsymbol{X}_{k+1}$, i.e., looking "backward")

$= \sum_{\boldsymbol{x}_{k+1}} P(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{x}_{k+1})\ \mathbb{P}(\boldsymbol{x}_{k+1} \mid \boldsymbol{X}_k)$ (cond. indep. - Markov assumption)

$= \sum_{\boldsymbol{x}_{k+1}} P(\boldsymbol{e}_{k+1}, \boldsymbol{e}_{k+2:t} \mid \boldsymbol{x}_{k+1})\ \mathbb{P}(\boldsymbol{x}_{k+1} \mid \boldsymbol{X}_k)$ (decompose)

$= \sum_{\boldsymbol{x}_{k+1}} P(\boldsymbol{e}_{k+1} \mid \boldsymbol{x}_{k+1})\ P(\boldsymbol{e}_{k+2:t} \mid \boldsymbol{x}_{k+1})\ \mathbb{P}(\boldsymbol{x}_{k+1} \mid \boldsymbol{X}_k)$ (1. sensor, 2. backward msg, 3. transition model)

$= BACKWARD(\boldsymbol{b}_{k+2:t}, \boldsymbol{e}_{k+1})$

$\mathbb{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k)$ ("backward message", propagated recursively)

$\boldsymbol{b}_{k+1:t} = BACKWARD(\boldsymbol{b}_{k+2:t}, \boldsymbol{e}_{k+1})$ (through "backward step function")

$\boldsymbol{b}_{t+1:t} = \mathbb{P}(\boldsymbol{e}_{t+1:t} \mid \boldsymbol{X}_t) = \mathbb{P}(\mid \boldsymbol{X}_t) = \boldsymbol{1}$

Forward-backward smoothing as matrix-vector operations

Forward-equation

$$\boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t+1}) = f(\boldsymbol{P}(\boldsymbol{X}_t \mid \boldsymbol{e}_{1:t}), \boldsymbol{e}_{t+1}) = \boldsymbol{f}_{1:t+1} = \alpha \; \boldsymbol{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t) \, P(\boldsymbol{x}_t \mid \boldsymbol{e}_{1:t})$$

becomes $\boldsymbol{f}_{1:t+1} = \alpha \; \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$ (Matrix-matrix and matrix-vector scalar multiplication!)

Backward-equation

$$\boldsymbol{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k) = \boldsymbol{b}_{k+1:t} = \sum_{\boldsymbol{x}_{k+1}} P(\boldsymbol{e}_{k+1} \mid \boldsymbol{x}_{k+1}) \, P(\boldsymbol{e}_{k+2:t} \mid \boldsymbol{x}_{k+1}) \, \boldsymbol{P}(\boldsymbol{x}_{k+1} \mid \boldsymbol{X}_k)$$

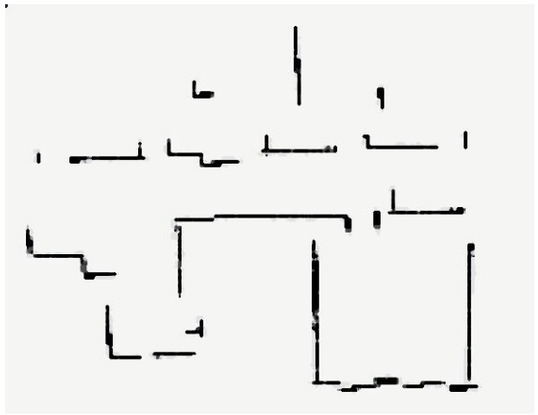
becomes $\boldsymbol{b}_{k+1:t} = \boldsymbol{T}\boldsymbol{O}_{k+1} \boldsymbol{b}_{k+2:t}$

Forward-Backward-equation is then still $\alpha \; \boldsymbol{f}_{1:k} \times \boldsymbol{b}_{k+1:t}$

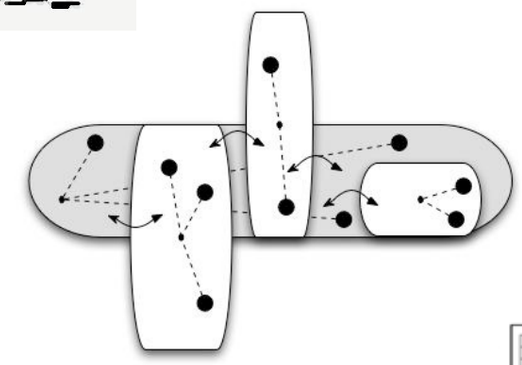
Cf. https://en.wikipedia.org/wiki/Forward–backward_algorithm
for an illustration of the book-example in the "umbrella world"

Mapping

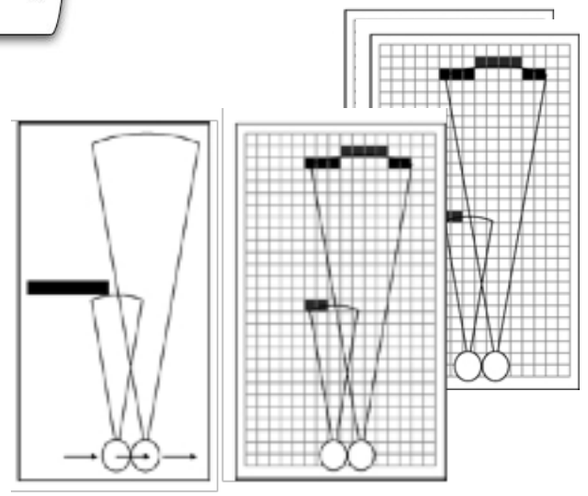
Geometrical approaches



Topological approaches

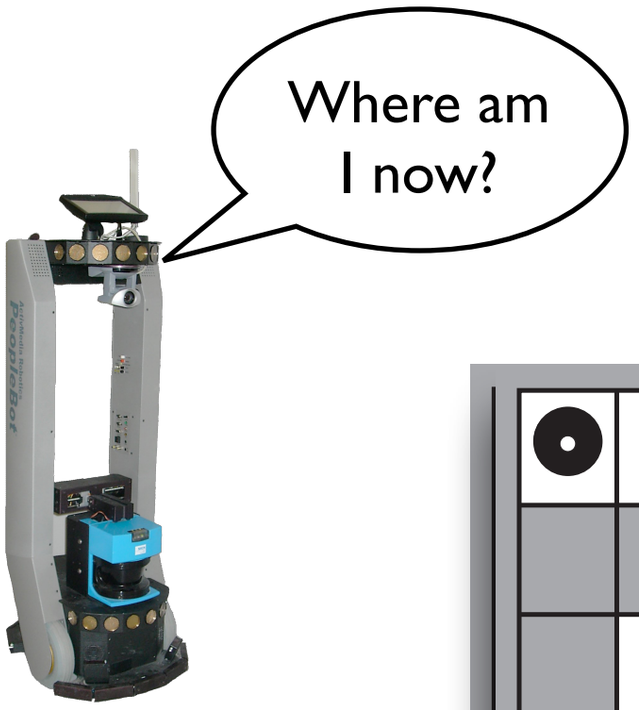


Occupancy grid approaches (e.g., Sebastian Thrun)

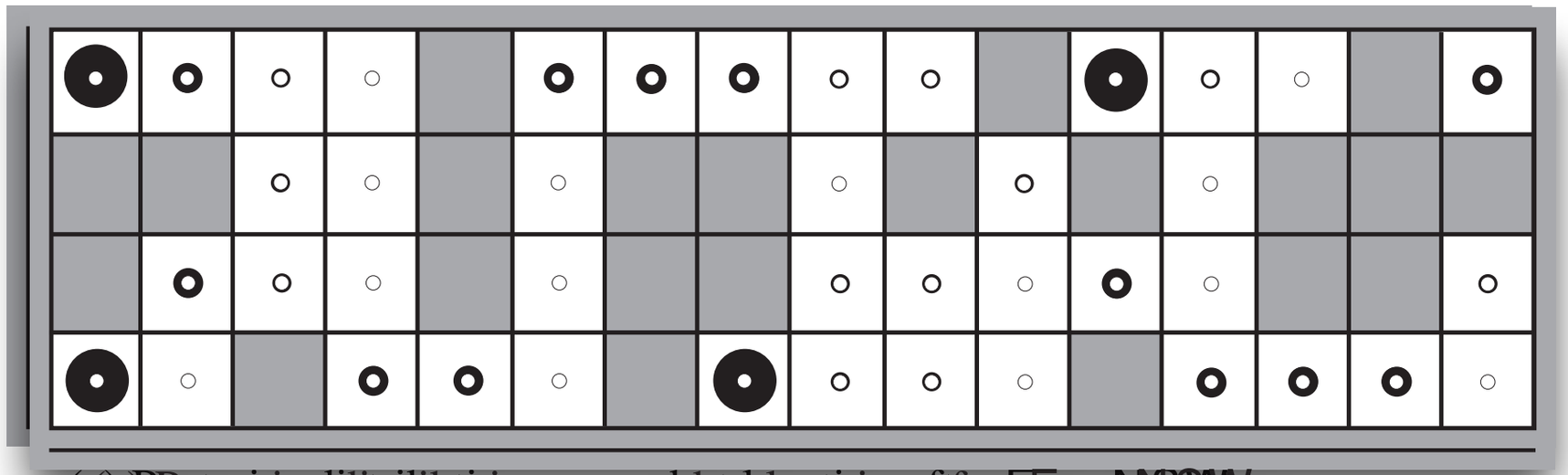


(Hybrid approaches)

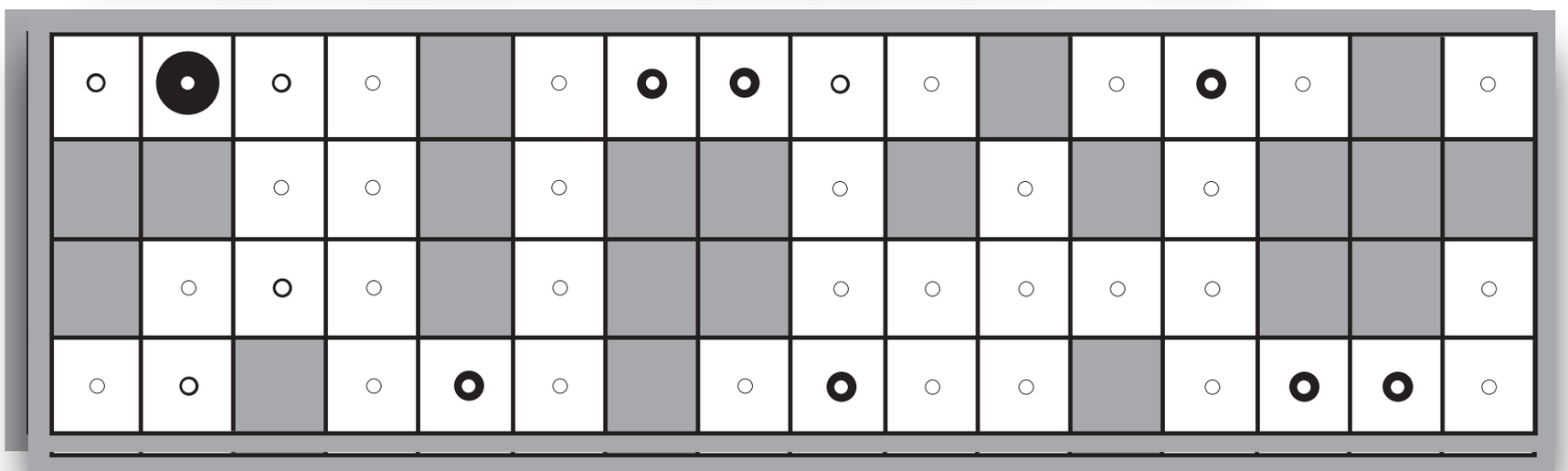
Localisation



HMM in a grid world



(a) Posterior distribution over robot location after E_1 = NSW



(b) Posterior distribution over robot location after E_1 = NSW, E_2 = NS

Data filters for state estimation

0. Represent state, identify system function

1. Estimate / predict state from model applying the function

2. Take a measurement

3. Update state according to model and observation (measurement)

Used for position tracking, detection of significant changes in a data stream, localisation ...

E.g., particle filters (Monte Carlo), Kalman filters

Particle filter

1. Represent possible positions by N samples (start with a uniform probability distribution within the possible (or reasonable) state space for t = 0):
$\overrightarrow{x_i}^t = < x_i, y_i, \theta_i >^t , \quad i = 0,...,N$

2. Predict: Estimate movement / new sample pose according to assumed robot movement (+ noise):
$\overrightarrow{x_i}^{t+1} = Tr(< x_i, y_i, \theta_i >^t)$, with *Tr* being the transition model including *some* noise ϵ in each component, e.g.,
$Tr(< x, y, \theta >) = < x + cos(\theta)\delta_t + \epsilon_x, y + sin(\theta)\delta_t + \epsilon_y, \theta + \epsilon_\theta >$ is a simple continued move in same direction for distance δ_t per time step t (plus noise).

3. Update: Take a measurement $\vec{z}^{t+1}$

4. Update: Assign weights to samples according to *posterior probabilities* $P(\overrightarrow{x_i} \mid \vec{z}^{t+1})$

5. Resample (pick "good" samples with high weights, use those as new "seeds", redistribute in position space and add (again) some noise), continue at 2.

Localisation

E.g., Monte Carlo Localisation (D. Fox, S. Thrun, *et al.*)

Particle filter based localisation indoor

Kalman filter

Represent posteriors for transition and sensor models with a Gaussian.

$$\mathcal{N}(\vec{\mu}, \Sigma)(\vec{x}) = \alpha e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1}(\vec{x}-\vec{\mu})}$$

Assume *locally linear* dynamical system with
F, H system matrices, i.e., transition and sensor model;
$\vec{x}_t$ the state in time t, $\vec{z}_t$ measurement in time t (vectors);
Σ_x and Σ_z the covariances for the transitions and measurements, i.e., gaussian noise

$$P(\vec{x}_{t+1} \mid \vec{x}_t) = \mathcal{N}(F\vec{x}_t, \Sigma_x)(\vec{x}_{t+1})$$

$$P(\vec{z}_t \mid \vec{x}_t) = \mathcal{N}(H\vec{x}_t, \Sigma_z)(\vec{z}_t)$$

Then do prediction and update, just as with the HMM (well, almost ;-)

Kalman filter, updates

Update the overall mean *state estimate* $\vec{\mu}_{t+1}$ (essentially $\vec{x}_{t+1}$) based on last estimate $\vec{\mu}_t$ as

$$\vec{\mu}_{t+1} = F\mu_t + K_{t+1}(\vec{z}_{t+1} - HF\vec{\mu}_t)$$

with $F\vec{\mu}_t$ the state prediction and $HF\vec{\mu}_t$ the prediction of the observation; and

update the overall covariance as

$$\Sigma_{t+1} = (I - K_{t+1}H)F\Sigma_t F^T + \Sigma_x$$

with

$$K_{t+1} = (F\Sigma_t F^T + \Sigma_x)H^T(H(F\Sigma_t F^T + \Sigma_x)H^T + \Sigma_z)^{-1}$$

the *Kalman gain matrix*
(essentially what we win in terms of certainty when doing the update)

GPS-free positioning at sea

Depth and magnetic field measurements in a PF + KF combination

[Mårten Lager, Smart Technologies for Unmanned Surface Vessels, Licentiate Thesis, CS LTH, Feb 2019]

GPS-free positioning at sea (2)

Example for discarding samples: When particles are "on the wrong side" of the bottom depth line, they are discarded

[Mårten Lager, Smart Technologies for Unmanned Surface Vessels, Licentiate Thesis, CS LTH, Feb 2019]

GPS-free positioning at sea (3)

Combining the different distributions (for depth and magnetic field measurements) into a joint PDF

[Mårten Lager, Smart Technologies for Unmanned Surface Vessels, Licentiate Thesis, CS LTH, Feb 2019]

Mapping & Localisation: Chicken & Egg?

Simultaneous localisation and mapping (SLAM)

While building the map, stay localised!

Use filters to “sort” landmarks:

Known? Update your pose estimation!

Unknown? Extend the map!

SLAM example

FastSLAM (D. Haehnel)

Summary

AI in Robotics - integrating the "brain" into the "body"

General intro to robotics and the challenges wrt AI methods (and vv)

Probabilistic methods in Robotics and HRI

- Probabilistic methods for Mapping & Localisation

- Reading advice (even if you do not work on assignment 3):
  Dieter Fox *et al.*, *"Monte Carlo Localization: Efficient Position Estimation for Mobile Robots"*, Proceedings of the 16th AAAI Conference on Artificial Intelligence (AAAI-99), AAAI Classic AI Paper Award 2017.