Autonomous agents
Lecture 11, 20160225

Decision-making in autonomous robots

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today’s learning goals

- After this lecture you should be able to:
  - Describe and compare two classes of decision-making methods for autonomous robots.
  - Define and describe the utility function method for decision-making.
  - Briefly describe some applications of the utility function method.

Decision-making methods: Taxonomy

- Arbitration methods:
  - (Exactly) one behavior is active and controls the (motor) actions of the robot.
- Cooperative methods:
  - The action taken represents a weighted average of the suggestions given by several behaviors.
- Many different methods have been suggested in each category.
- Here we will consider a (hybrid) method, the utility function method.

Today's learning goals

- After this lecture you should be able to:
  - Describe and compare two classes of decision-making methods for autonomous robots. ✓
  - Define and describe the utility function method for decision-making.
  - Briefly describe some applications of the utility function method.

The utility function (UF) method

- Mainly intended for motor tasks (e.g. navigation, map-building etc.)
- Typical application: An indoor delivery task.
- However, the method can be applied to many different tasks.
- As the name implies, the method is based on utility maximization.

The UF method: Brain processes

- In this method, two kinds of brain processes are defined:
  - (Motor) behaviors
    - Control the motors of the robot
    - Exactly one such behavior is active at any time.
    - Examples: Navigation (along a given path), obstacle avoidance
  - Cognitive processes
    - Do not make use of the robot’s motors.
    - Any number of cognitive processes can be active at any time.
    - Examples: Path planning, image processing, speech recognition

The UF method: Brain processes

- Each brain process B_i is associated with a utility function U_i
- Motor behaviors: At any given time, the behavior with the largest utility value is active.
- Cognitive behaviors: All behaviors with $U_i > 0$ are active.
- The main question, of course, is: *How does one determine the utility functions?*
- Some definitions are needed first (see the following slides):
  - Sensory preprocessing system
  - State variables

UF method: Sensory preprocessing

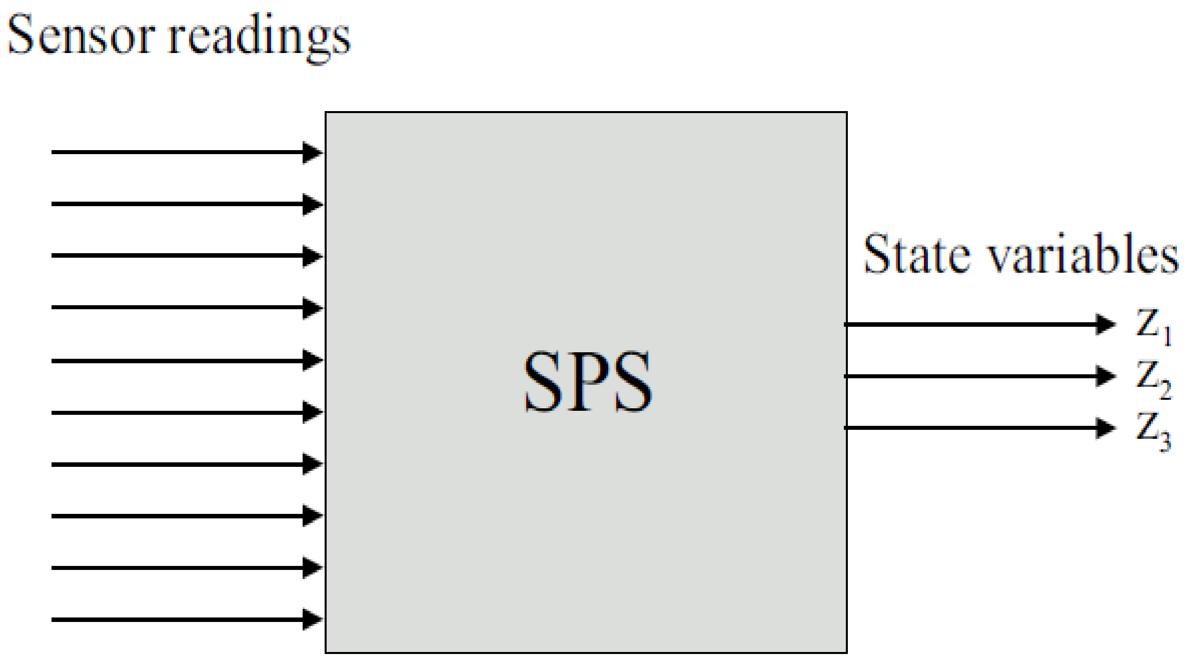


Figure 8.1: *A schematic representation of an SPS. In general, an SPS takes raw sensory readings as input and generates state variables as output.*

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

UF method: Utility functions

$$\tau_i \dot{u}_i + u_i = \sigma_i \left(\sum_{k=1}^{m} w_{ik} z_k + b_i + \Gamma_i \right) \; i = 1, \ldots, n.$$

$$\sigma_i(x) = \tanh(c_i x),$$

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

UF method: Noise tolerance

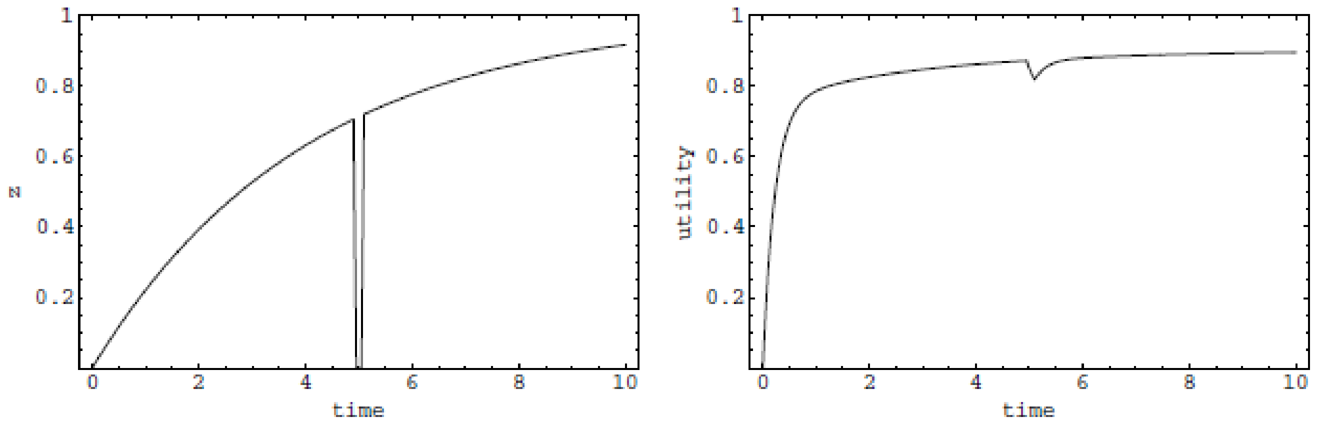
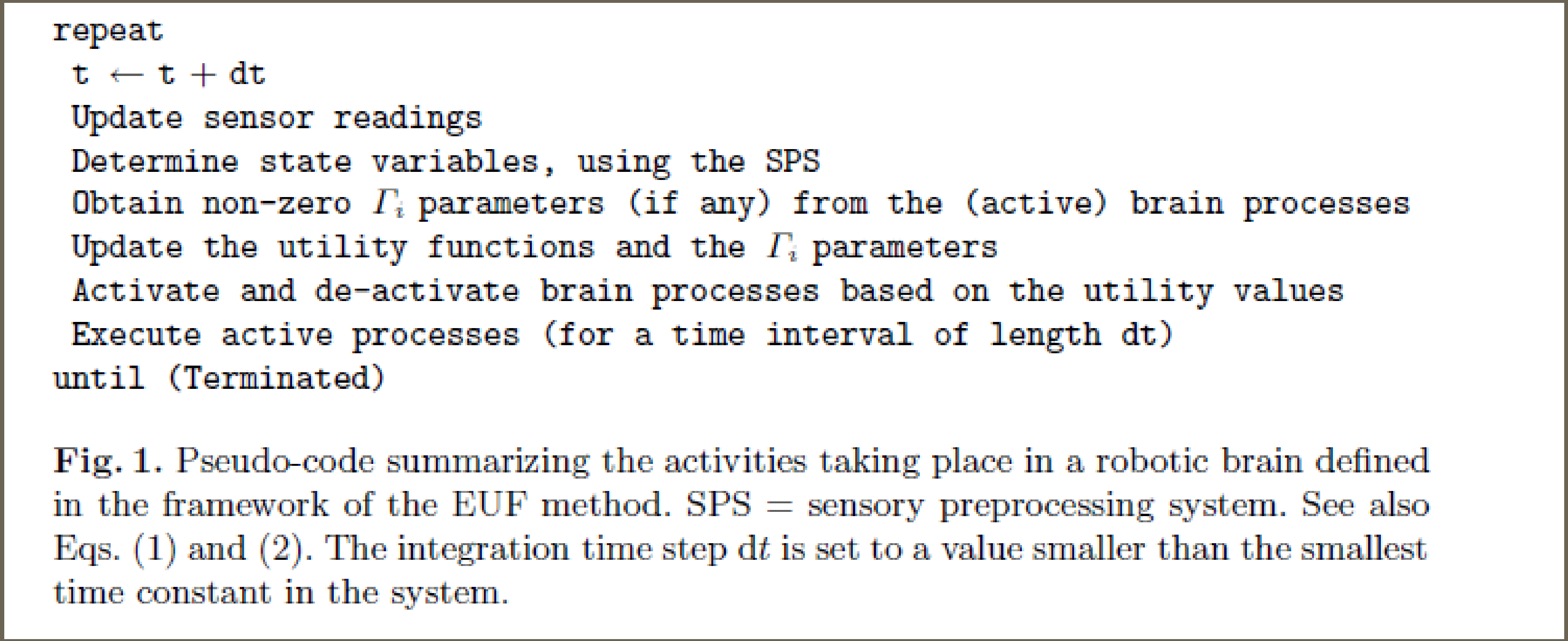


Figure 8.2: *An illustration of the noise tolerance resulting from the differential form of the utility functions. The left panel shows an example of signal loss in a state variable z, in turn caused by signal loss in the sensor(s) whose readings are used when forming the state variable: At around $t = 5$ s, the state variable suddenly dropped to zero for 0.15 s. Right panel: The corresponding change in a utility function, defined by $\tau\dot{u} + u = \sigma(b + wz)$, where $\tau = 0.3$, $b = 1.0$ and $w = 0.5$. The sigmoid parameter c was set to 1.0.*

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

UF method: Flow



```
repeat
  t ← t + dt
  Update sensor readings
  Determine state variables, using the SPS
  Obtain non-zero $\Gamma_i$ parameters (if any) from the (active) brain processes
  Update the utility functions and the $\Gamma_i$ parameters
  Activate and de-activate brain processes based on the utility values
  Execute active processes (for a time interval of length dt)
until (Terminated)
```

Fig. 1. Pseudo-code summarizing the activities taking place in a robotic brain defined in the framework of the EUF method. SPS = sensory preprocessing system. See also Eqs. (1) and (2). The integration time step dt is set to a value smaller than the smallest time constant in the system.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

UF method: Finding utility functions

- In general, the utility functions (i.e the constants defined in Eq. (8.4)) are obtained in simulation, using an evolutionary algorithm.
- Often, however, one can make an initial guess that leads to reasonable (but not perfect) behavior selection, which can then be used as a starting point for the evolutionary algorithm.

Today's learning goals

- After this lecture you should be able to:
  - Describe and compare two classes of decision-making methods for autonomous robots. ✓ ✓
  - Define and describe the utility function method for decision-making.
  - Briefly describe some applications of the utility function method.

UF method: Illustration of UFs.

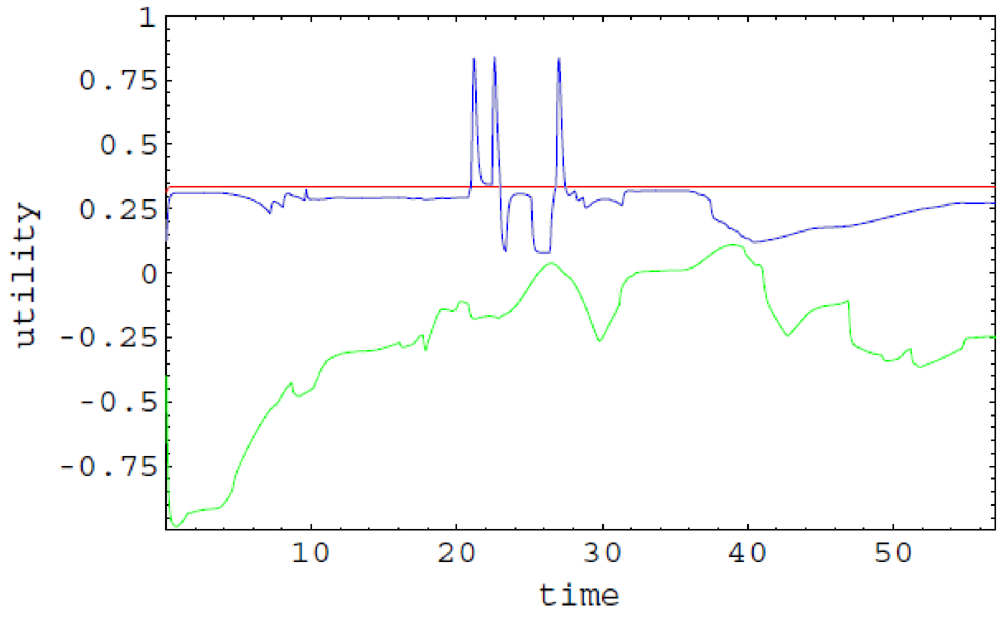


Figure 8.3: *An example of utility function dynamics, showing the variation (with time) of three utility functions u_1 (red), u_2 (green) and u_3 (blue).*

UF method: Example 1

- A specific example can be found in the paper available at

  http://www.me.chalmers.se/~mwahde/AdaptiveSystems/Publications/Wahde_EUF2009.pdf

  (see pp. 7-10)
- In this case, the robot solves a navigation task, using *potential field navigation* and *laser localization* for odometric recalibration.
- Brain processes: Potential field navigation (B1), Side collision avoidance (B2), Frontal collision avoidance (B3), Odometry (B4), Laser localization (B5).
- B4 is a cognitive process, all others are motor behaviors.

UF method: Example 1

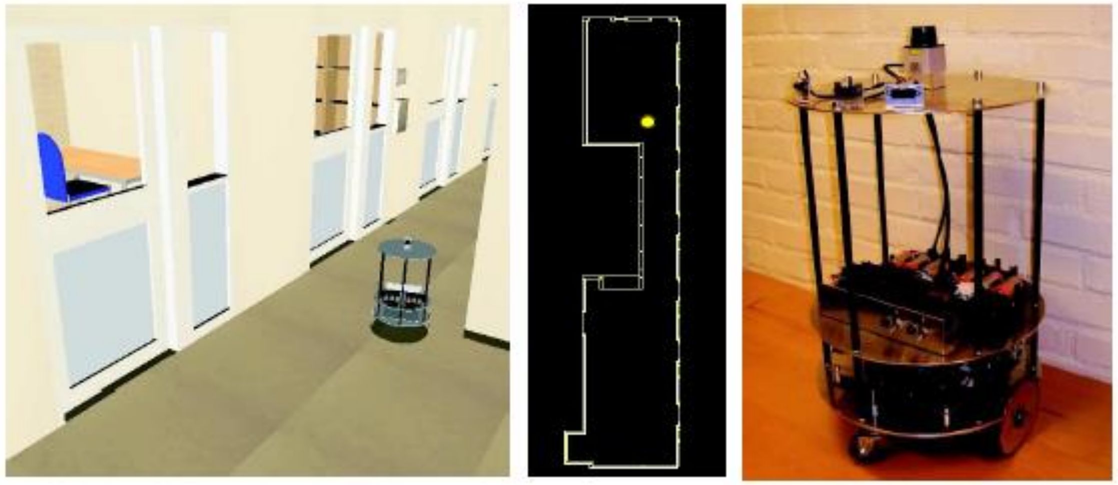


Fig. 2. Left panel: A screenshot from a simulation, showing a wheeled robot in a typical office environment. Middle panel: A schematic view (from above) of the office environment in which the simulations were carried out. Right panel: The physical counterpart (currently under construction) to the robot used in the simulations.

UF method: Example 1

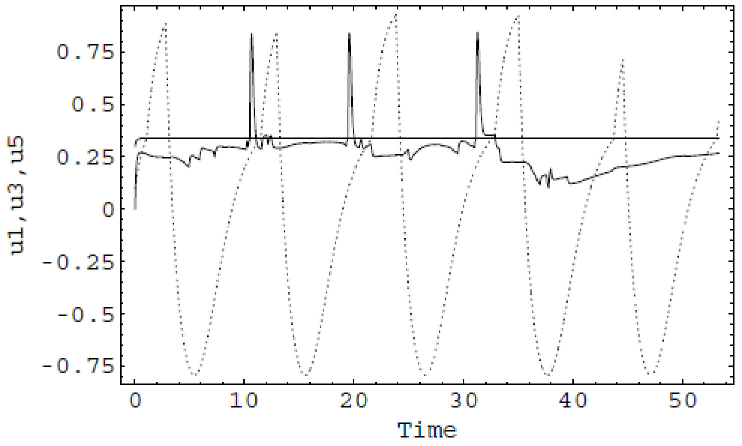


Fig. 3. The variation of utility values for brain processes B1 (solid, almost constant), B3 (solid, with spikes), and B5 (dotted), during a simulation.

UF method: Example 2

- Another example can be found in the paper available at http://www.intechopen.com/articles/show/title/reliable-long-term-navigation-in-indoor-environments
- In this case, the robot uses a grid-based navigation method, combined with localization and moving obstacle avoidance.
- Brain processes: Grid navigation (B1), Odometry (B2), Odometric calibration (B3), Moving obstacle detection (B4), Moving obstacle avoidance (B5), Long-term memory (B6).
- B1 and B5 are motor behaviors, all others are cognitive processes.

UF method: Example 2

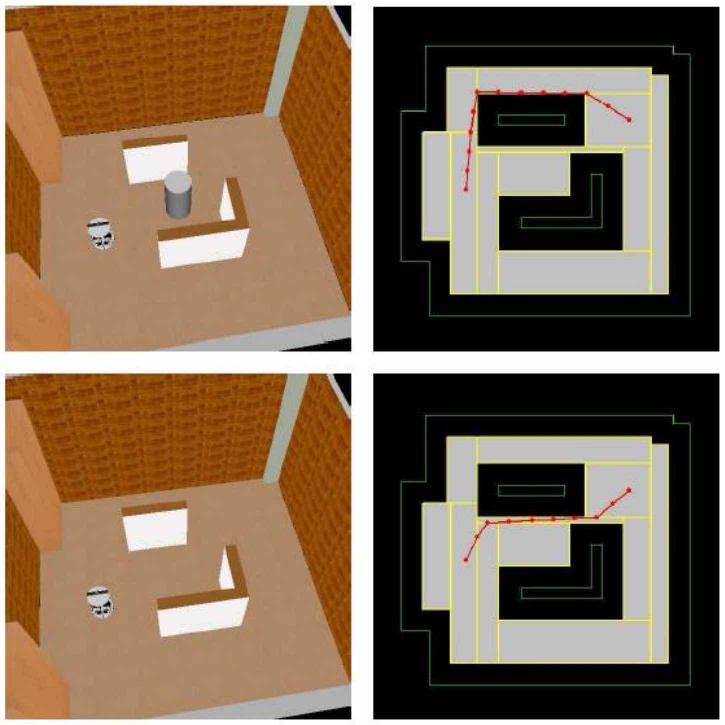


Fig. 7. Path planning with (upper panels) and without (lower panels) moving obstacles present. The moving obstacle is represented as a gray cylinder.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

UF method: Example 2

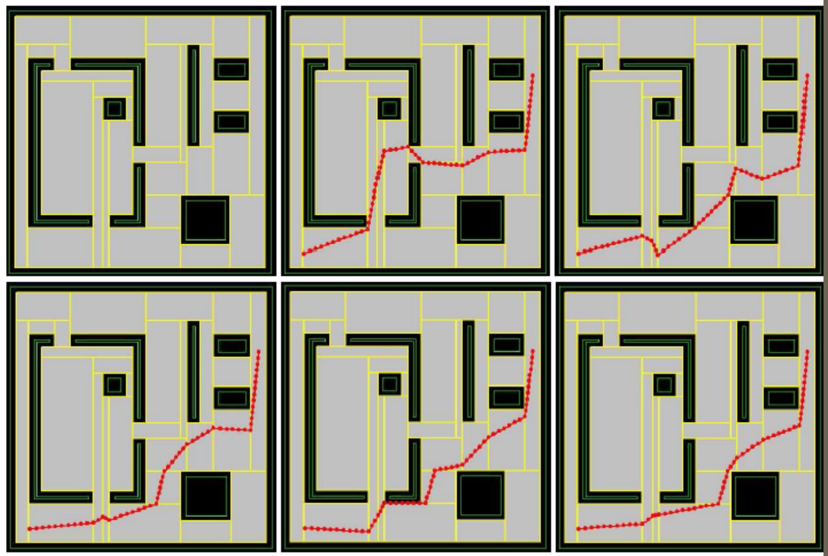


Fig. 6. Path planning: The upper left panel shows the convex navigation grid for the arena described in Sect. 6 and the remaining panels show the paths considered during path planning. In this case, the robot was located in the lower left corner of the arena, and its navigation target was located near the upper right corner. The first path considered is shown in the upper middle panel. After some time, the optimization procedure found a shorter path (upper right panel), along a different route. In the remaining steps (lower panels) the path planning algorithm further optimized the path (length minimization) by adjusting the placement of the waypoints (on the edges of the convex grid cells). Note that, in this figure, the actual navigation path is shown, i.e. not only the waypoints on the cell edges, but also all the intermediate (interpolated) waypoints.

UF method: Example 2

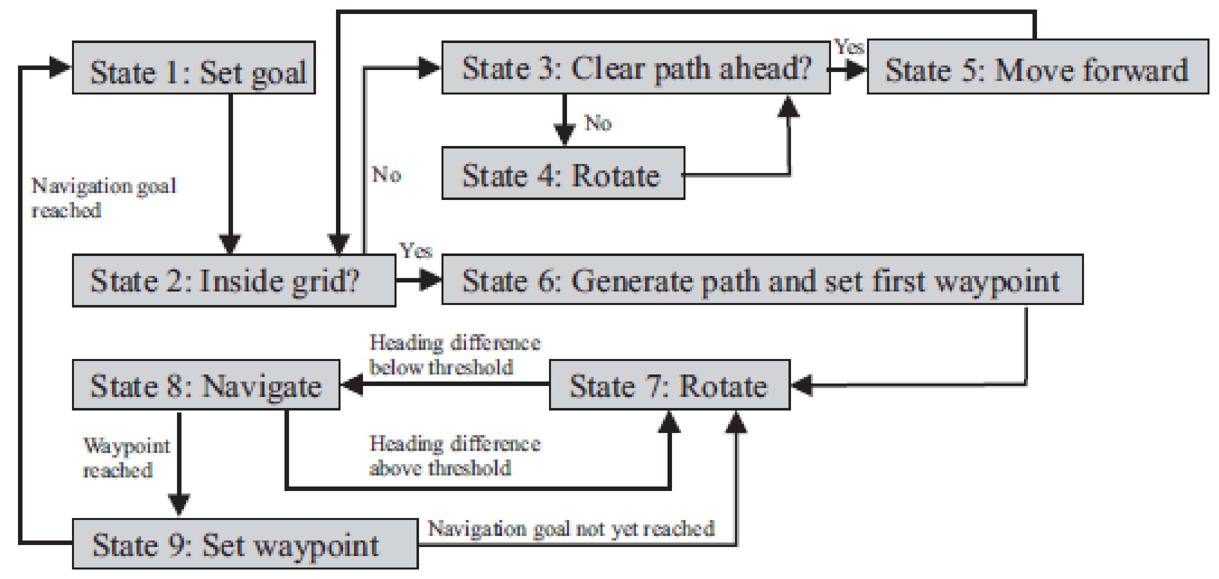


Fig. 8. The FSM implementing grid navigation.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

UF method: Example 2

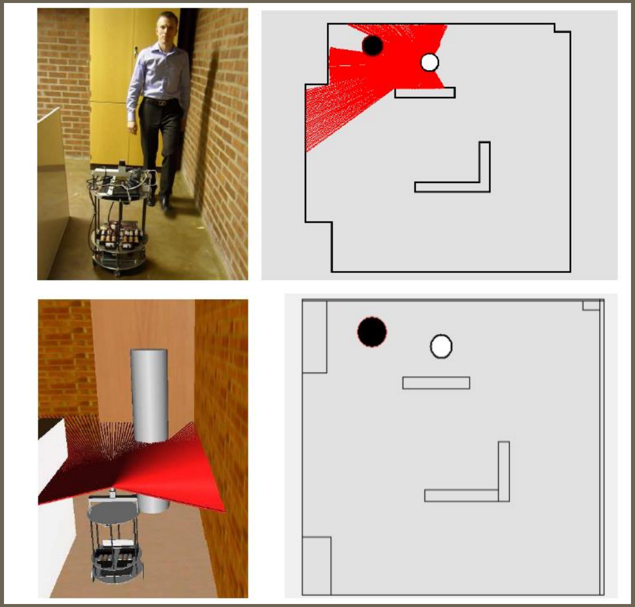


Fig. 10. Upper panels: Detection of a moving obstacle by the real robot. Left: The person approaching the robot. Right: a screenshot showing the LRF rays (for the real robot). The robot is shown as a white disc at its estimated pose, and the detected moving obstacle as a black disc. Lower panels: The same situation, shown for the simulator. Here, the laser rays (which obviously are not visible in the real robot) have been plotted in the screenshot rather than in the map view shown in the right panel.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

UF method: Example 2

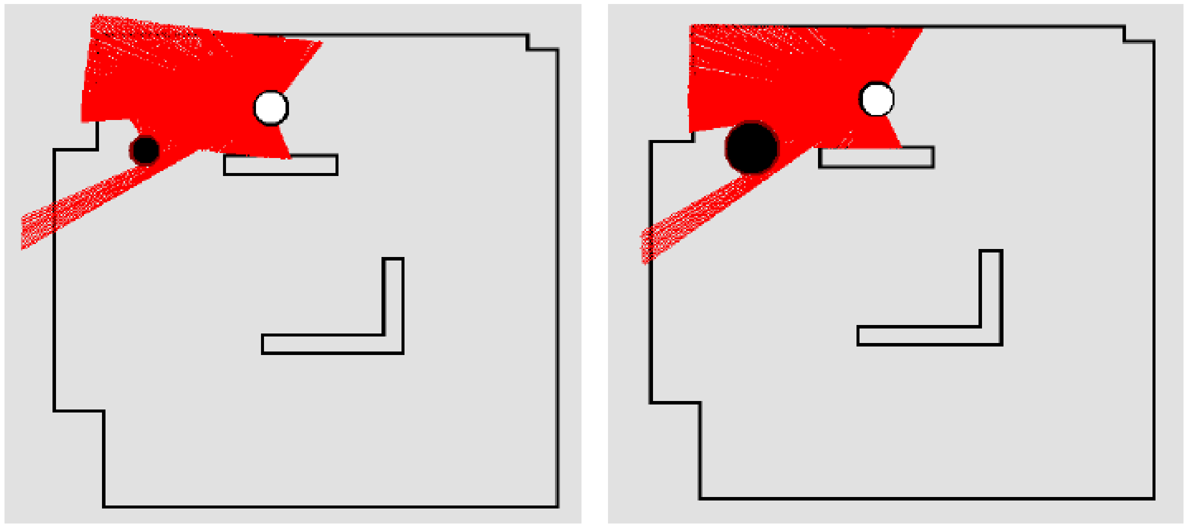
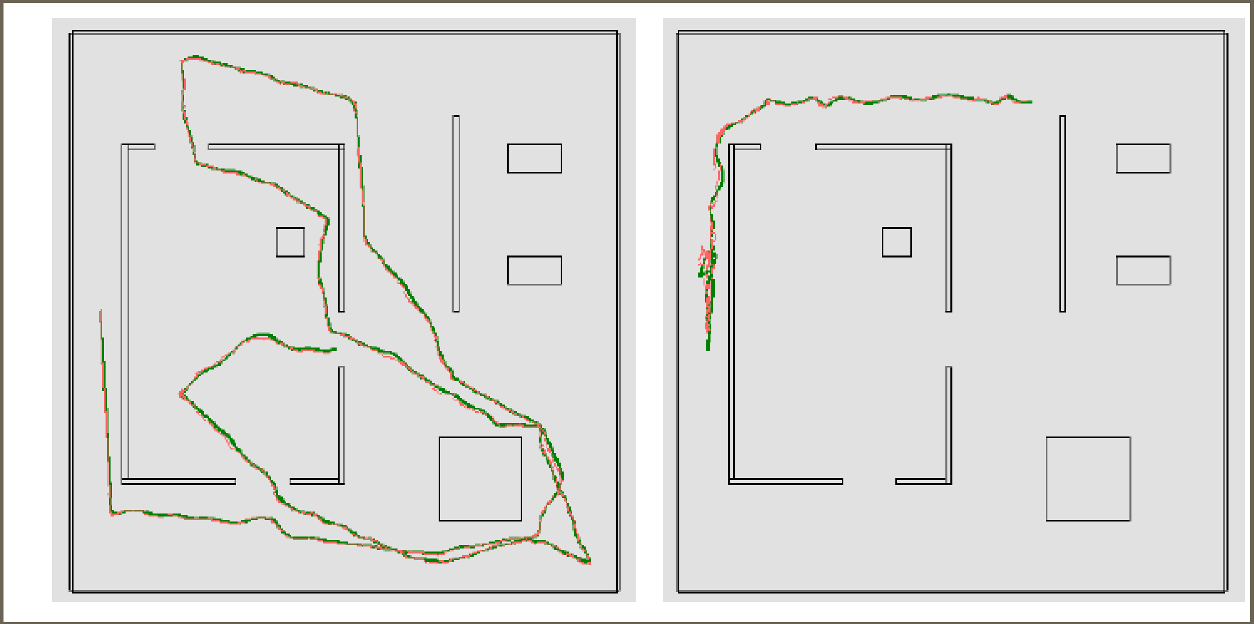


Fig. 11. An example of odometric calibration in the presence of a moving obstacle (shown as a black disc). The real robot, shown as a white disc, was used. The same laser readings (from a single laser scan) are shown in both panels. In the left panel, the origin and direction of the laser readings correspond to the robot's estimated (and incorrect) pose before odometric calibration. In the right panel, the estimated pose has been corrected.

UF method: Example 2



Lower left panel: The trajectory for the first 100 m of navigation. The actual trajectory is shown as a thick green line, and the odometric trajectory as a thin red line. Lower right panel: A later part of the trajectory showing the robot approaching 200 m of distance covered. As can be seen, in the long corridor, the robot suffered a rather large pose error at one point. The error was swiftly corrected, however, so that the robot could continue its task.

UF method: Example 2

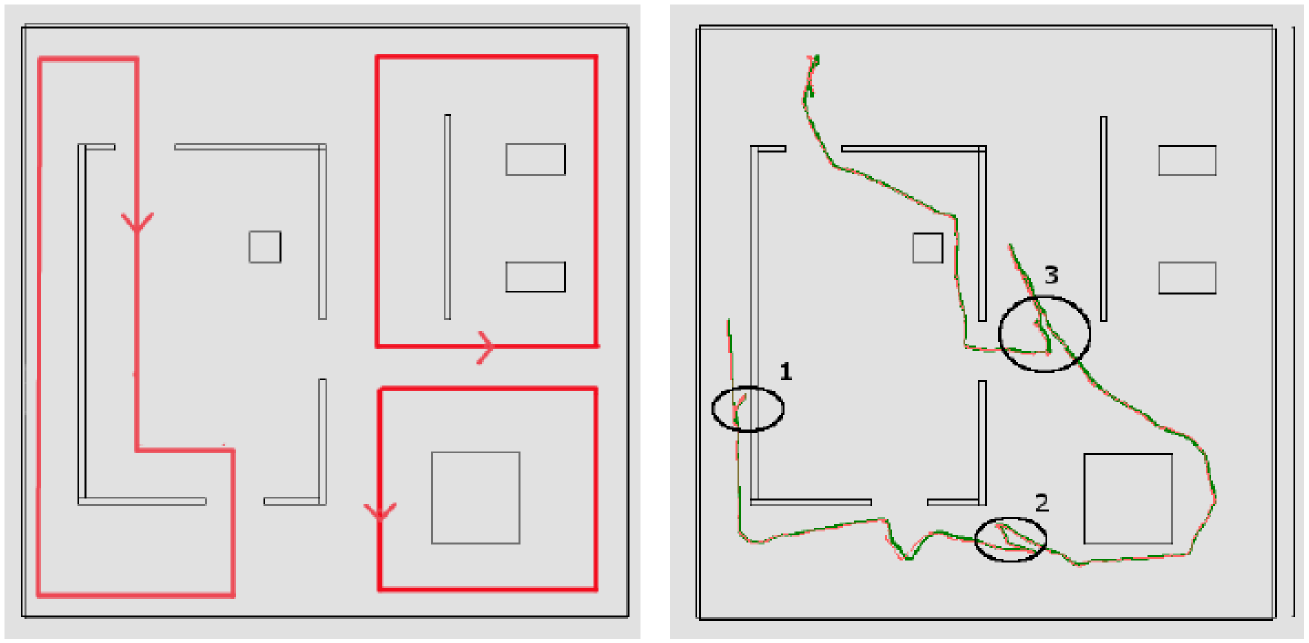


Fig. 13. Left panel: The trajectories of the moving obstacles in Run 2. Right panel: The robot's trajectory during the first 50 m of movement. The ellipses highlight the robot's three evasive maneuvers.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today's learning goals

- After this lecture you should be able to:
  - Describe and compare two classes of decision-making methods for autonomous robots. ✓ ✓
  - Define and describe the utility function method for decision-making.
  - Briefly describe some applications of the utility function method. ✓

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

HP1 results

- Generally acceptable results
- Some (many) students have not provided civic registration number. *Please do so in connection with HP2*.
- Some common errors and mistakes, see the following pages.
- Most errors result from not reading the instructions!

HP1.1 (answers)

- 1.1 (a) $\varphi(t) = \frac{V_0}{4R}\left(\frac{1}{t_2} - \frac{1}{t_1}\right)t^2$, $x(t) = R\left(\frac{t_1+t_2}{t_1-t_2}\right)\sin\varphi(t)$, $y(t) = R\left(\frac{t_1+t_2}{t_1-t_2}\right)(1-\cos\varphi(t))$.
- Note that the radius of the circle equals $\left|R\left(\frac{t_1+t_2}{t_1-t_2}\right)\right|$.
- 1.1 (b) x = -0.0984, y = -0.8851, φ = 5.817 (or 12.1)

HP1 Comments (common errors)

- 1.1
  - The integrals (in (a)) were not solved analytically,
  - The parameter values were inserted when solving the integrals,
  - When solving for y(t), some students forgot that cos(0) = 1,
  - Forgot to multiply the derivatives by dt when integrating,
  - Behavior logic should be contained *within* a state,
  - Plots without title or caption, axes without labels or units,
  - Many don’t even look at their plots. For example y(0) should be 0 in this case!

HP1 Comments (common errors)

- 1.2
  - Robots bouncing off the wall it was following,
  - Robots crashing into the wall it was following,
  - Parameters not defined CreateBrain,
  - FSMs not well structured
    - MANY deviations from the coding standard! IMPROVE UNTIL HP2!
    - States that do nothing
    - No FSM at all!
  - No clear description in the report, no figure etc.

HP1 Comments (common errors)

- 1.3
  - Using odometry for absolute position and heading,
  - Didn’t implement a solution to stop as close as possible to the initial pose,
  - Forgot to specify the maximum number of steps to be used,
  - Use time elapsed (for one lap) to estimate the time for the second the lap: A very brittle and error-prone approach (also seen in HP2, by the way).

About the exam...

- Some typical exam problems (and solutions) will be put on the web page on Monday.
- The exam will contain 4-5 questions.
- Types of problems
  - Definition and description of various concepts
  - Derivations
  - Computational problems