Autonomous agents
Lecture 8, 20160211

Robotic behaviors II. Exploration and navigation (2)

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

CHALMERS

Today's learning goals

- After this lecture you should be able to
  - Describe and implement Dijkstra's method.
  - Describe and implement the Potential fields method.

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

Navigation: Grid-based methods

- Dijkstra's algorithm
  - Guaranteed to find the shortest path between the start point and the end point.
- A*-algorithm
  - Combines BFS and Dijkstra.
- Here we will consider Dijkstra's algorithm only.

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

Navigation: Dijkstra's algorithm

1. Place the robot at the start node s, which then becomes the current node. Assign the distance value 0 to the start node, and infinity to all other nodes (in practice, use a very large, finite value). Set the status of all nodes to *unvisited*.

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

Navigation: Dijkstra's algorithm

1. Place the robot at the start node s, which then becomes the current node. Assign the distance value 0 to the start node, and infinity to all other nodes (in practice, use a very large, finite value). Set the status of all nodes to *unvisited*.

2. Go through all the unvisited, accessible (i.e. empty) neighbors a_i of the current node c, and compute their distance d from the *start* node s. If d is smaller than the previously stored distance d_i (initially infinite, see Step 1), then (i) update the previously stored distance i.e. set $d_i = d$ and (ii) assign the current node as the predecessor node of a_i.

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

Navigation: Dijkstra's algorithm

1. Place the robot at the start node s, which then becomes the current node. Assign the distance value 0 to the start node, and infinity to all other nodes (in practice, use a very large, finite value). Set the status of all nodes to *unvisited*.

2. Go through all the unvisited, accessible (i.e. empty) neighbors a_i of the current node c, and compute their distance d from the *start* node s. If d is smaller than the previously stored distance d_i (initially infinite, see Step 1), then (i) update the previously stored distance i.e. set $d_i = d$ and (ii) assign the current node as the predecessor node of a_i.

3. After checking all the neighbors of the current node, set its status to visited.

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

Navigation: Dijkstra's algorithm

1. Place the robot at the start node s, which then becomes the current node. Assign the distance value 0 to the start node, and infinity to all other nodes (in practice, use a very large, finite value). Set the status of all nodes to *unvisited*.

2. Go through all the unvisited, accessible (i.e. empty) neighbors a_i of the current node c, and compute their distance d from the *start* node s. If d is smaller than the previously stored distance d_i (initially infinite, see Step 1), then (i) update the previously stored distance i.e. set $d_i = d$ and (ii) assign the current node as the predecessor node of a_i.

3. After checking all the neighbors of the current node, set its status to visited.

4. Select the node (among *all* the unvisited, accessible nodes in the grid) with the smallest distance from the start node, and set it as the new current node.

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

Navigation: Dijkstra's algorithm

1. Place the robot at the start node s, which then becomes the current node. Assign the distance value 0 to the start node, and infinity to all other nodes (in practice, use a very large, finite value). Set the status of all nodes to *unvisited*.

2. Go through all the unvisited, accessible (i.e. empty) neighbors a_i of the current node c, and compute their distance d from the *start* node s. If d is smaller than the previously stored distance d_i (initially infinite, see Step 1), then (i) update the previously stored distance i.e. set $d_i = d$ and (ii) assign the current node as the predecessor node of a_i.

3. After checking all the neighbors of the current node, set its status to visited.

4. Select the node (among *all* the unvisited, accessible nodes in the grid) with the smallest distance from the start node, and set it as the new current node.

5. Return to Step 2, unless the target has been reached.

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

Navigation: Dijkstra's algorithm

1. Place the robot at the start node s, which then becomes the current node. Assign the distance value 0 to the start node, and infinity to all other nodes (in practice, use a very large, finite value). Set the status of all nodes to *unvisited*.

2. Go through all the unvisited, accessible (i.e. empty) neighbors a_i of the current node c, and compute their distance d from the *start* node s. If d is smaller than the previously stored distance d_i (initially infinite, see Step 1), then (i) update the previously stored distance i.e. set $d_i = d$ and (ii) assign the current node as the predecessor node of a_i.

3. After checking all the neighbors of the current node, set its status to visited.

4. Select the node (among *all* the unvisited, accessible nodes in the grid) with the smallest distance from the start node, and set it as the new current node.

5. Return to Step 2, unless the target has been reached.

6. When the target is reached, use the predecessor nodes to trace a path from the target node to the start node. Finally, reverse the path to find the path from the start node to the target node.

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

CHALMERS

Navigation: Dijkstra's algorithm

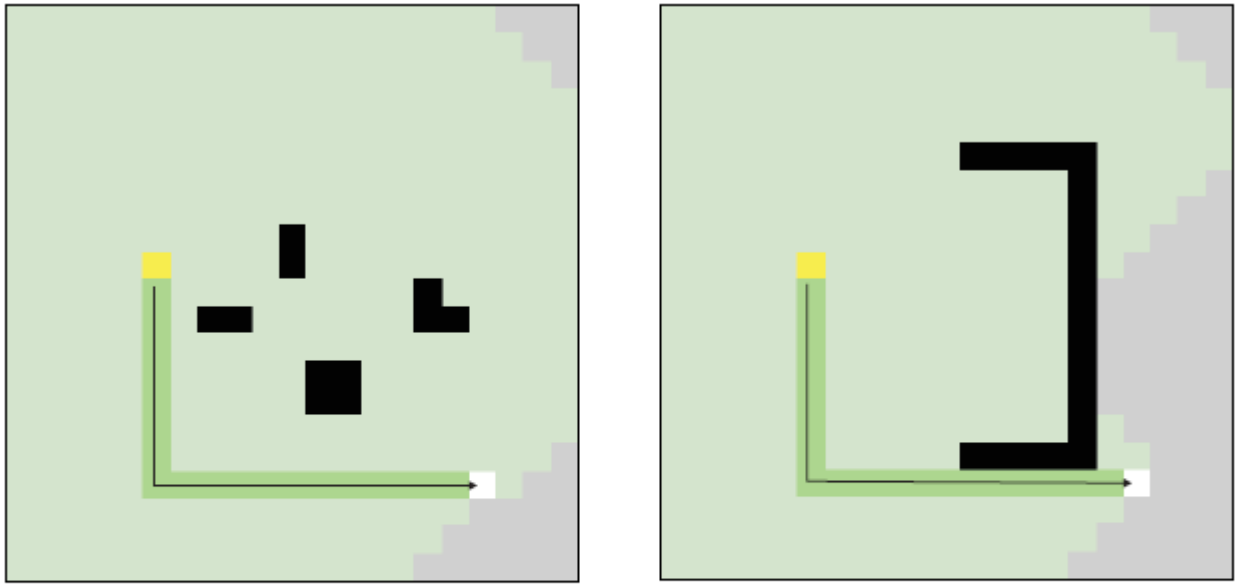


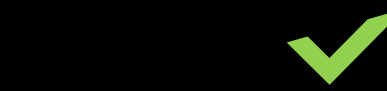
Figure 6.9: *Two examples of paths generated using Dijkstra's algorithm. The cells (nodes) that were checked during path generation are shown in light green, whereas the actual path is shown in dark green and with a solid line. The yellow cell is the start node and the white cell is the target node.*

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

CHALMERS

Today's learning goals

- After this lecture you should be able to
  - Describe and implement Dijkstra's method.
  - Describe and implement the Potential fields method.



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

Navigation: Potential fields

- In the potential fields (PF) method, one introduces an artificial potential $\Phi = \Phi(x, y)$.
- From the potential, one can compute a force at any given point as $\mathbf{F} = -\nabla\Phi$ (cf. gravity)
- However, rather than using the force thus obtained, one uses the gradient of Φ to obtain the desired *direction* of motion:

$$\hat{\mathbf{r}} = -\frac{\nabla\Phi}{|\nabla\Phi|} \equiv -\frac{\left(\frac{\partial\Phi}{\partial x}, \frac{\partial\Phi}{\partial y}\right)}{\sqrt{\left(\frac{\partial\Phi}{\partial x}\right)^2 + \left(\frac{\partial\Phi}{\partial y}\right)^2}}$$

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

Navigation: Potential fields

- Example of potential (functional form):

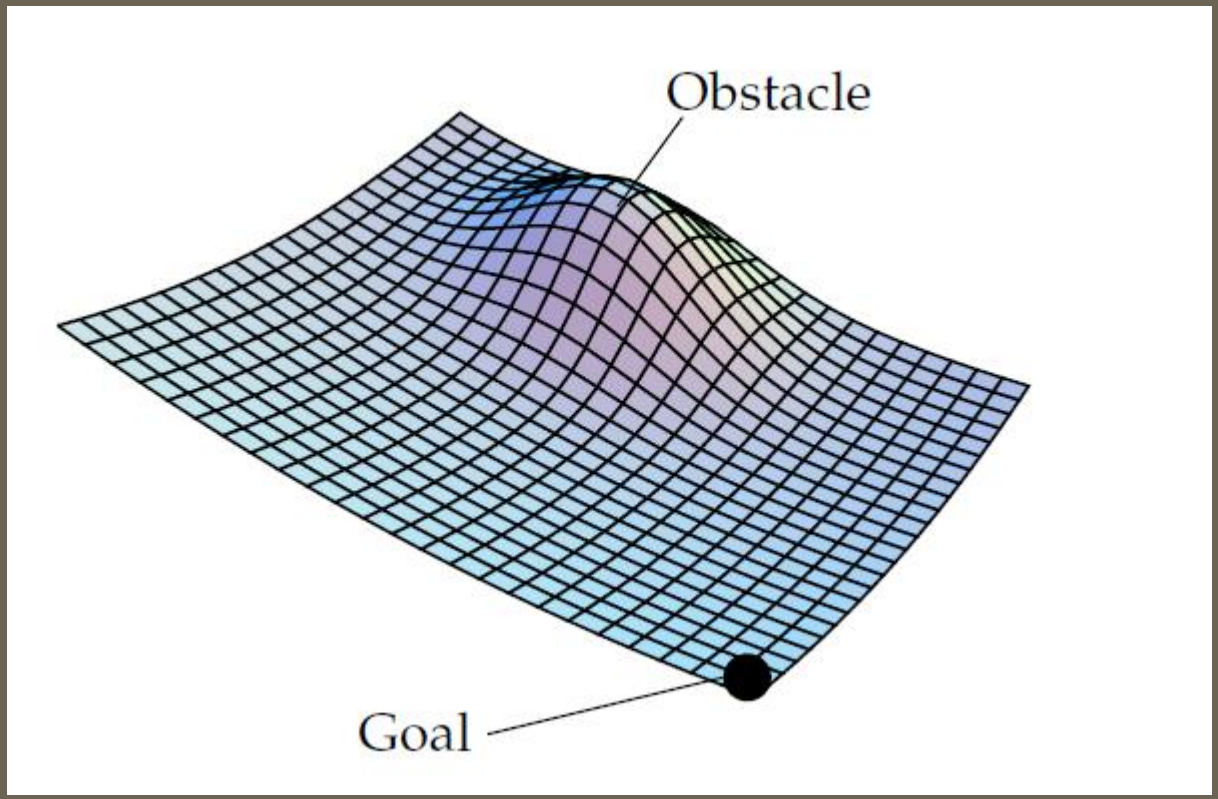
$$\phi(x, y; x_p, y_p, \alpha, \beta, \gamma) = \alpha e^{-\left(\frac{x-x_p}{\beta}\right)^2 - \left(\frac{y-y_p}{\gamma}\right)^2},$$

- The complete potential is obtained by summing the potentials from the goal and all obstacles:

$$\Phi(x, y) = \sum_{i=1}^{k} \phi_i(x, y; x_{p_i}, y_{p_i}, \alpha_i, \beta_i, \gamma_i),$$

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

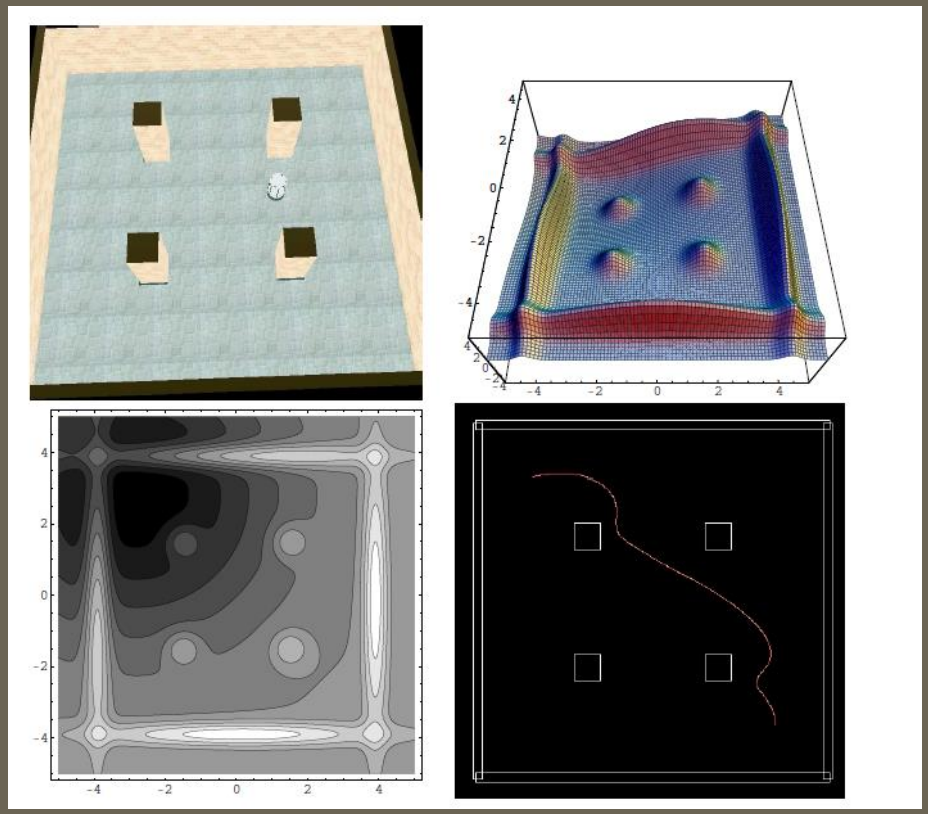
Navigation: Potential fields



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

CHALMERS

Navigation: Potential fields

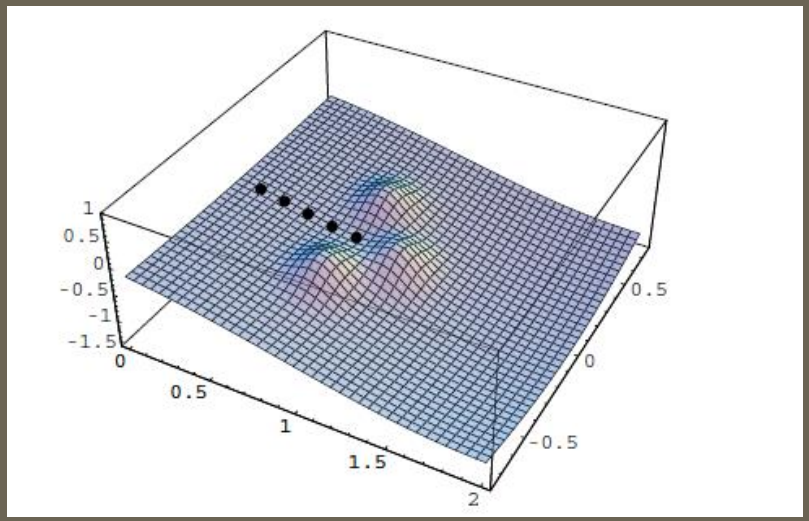


Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

CHALMERS

Navigation: Potential fields

- Problem: Locking phenomenon



- Can be avoided with the use of waypoints etc.

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

CHALMERS

Today's learning goals

- After this lecture you should be able to
  - Describe and implement Dijkstra's method.
  - Describe and implement the Potential fields method.

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