Intelligent Robotics

# Mapping and Navigation

Luís Paulo Reis (University of Porto)

Nuno Lau (University of Aveiro)

Armando Sousa (University of Porto)

# Background

- **Localization** – Where am I?
- **Mapping** – My (dynamic?) surroundings
- **Navigation** – How do I get where I want to go?
- TREND: **SLAM –**
  - Simultaneous Localization and Mapping

# The Representation Problem

**Representation is the form in which information is stored or encoded in the robot (Mataric)**

- **Representation is more than memory**
- **It has a significant impact on robot control**

# What can the robot represent

- **Self**
  - Stored proprioception, self-limitations, goals, intentions, plans
- **Environment**
  - Navigable spaces, structures
- **Objects, people, other robots**
  - Detectable things in the world
- **Actions**
  - Outcomes of specific actions in the environment
- **Task**
  - What needs to be done, where, in what order, how fast, etc.

# Navigation challenges

- **Path planning problem**
  - Robot has a map, knows own and target positions
- **Localization problem**
  - Robot has a map showing target, doesn’t know own position
- **Coverage problem**
  - Robot has a map, knows where it is, but doesn’t know where the target is
- **Mapping problem**
  - Robot does not have a map, may known own position
- **Simultaneous localization and mapping**
  - Robot does not have a map, and doesn’t know own position

# Navigation questions

- **Where am I going?**
  - Usually defined by human operator or mission planner
- **What is the best way to get there?**
  - Path planning problem
- **Where have I been?**
  - Mapping problem
- **Where am I?**
  - Localization problem

# Different types of representation

- **Maze navigator robot**
  - Exact path it has taken: “Go straight 2m, turn left 90 deg, go straight…”. This is an **odometric path**
  - Sequence of moves at particular landmarks: “Left at 1st junction, right at $2^{nd}$ junction, straight…”. This is a **landmark-based path**
  - What to do at each landmark: “At the green/red junction go left, at the red/blue junction go right, …”. This is a **landmark-based map**
  - The map of the maze. This is a **metric map**

# Metric maps and Topological maps

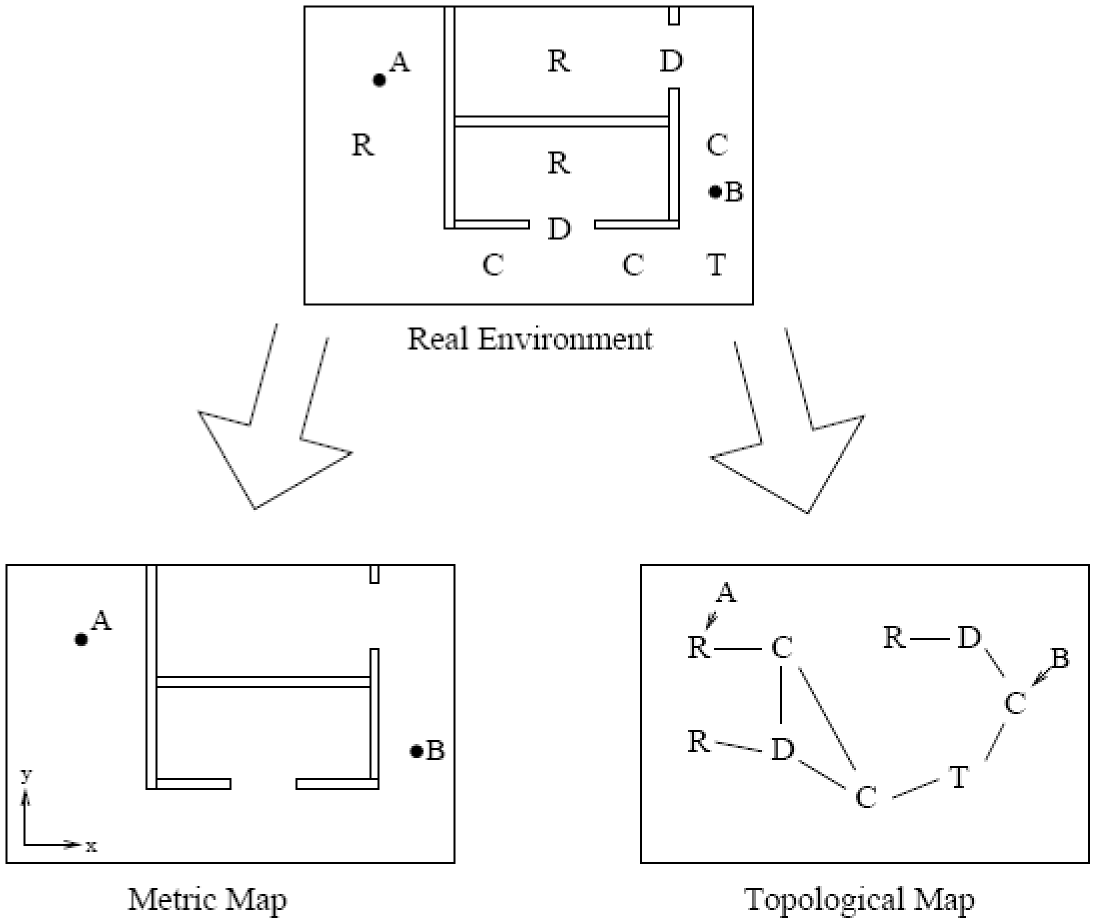


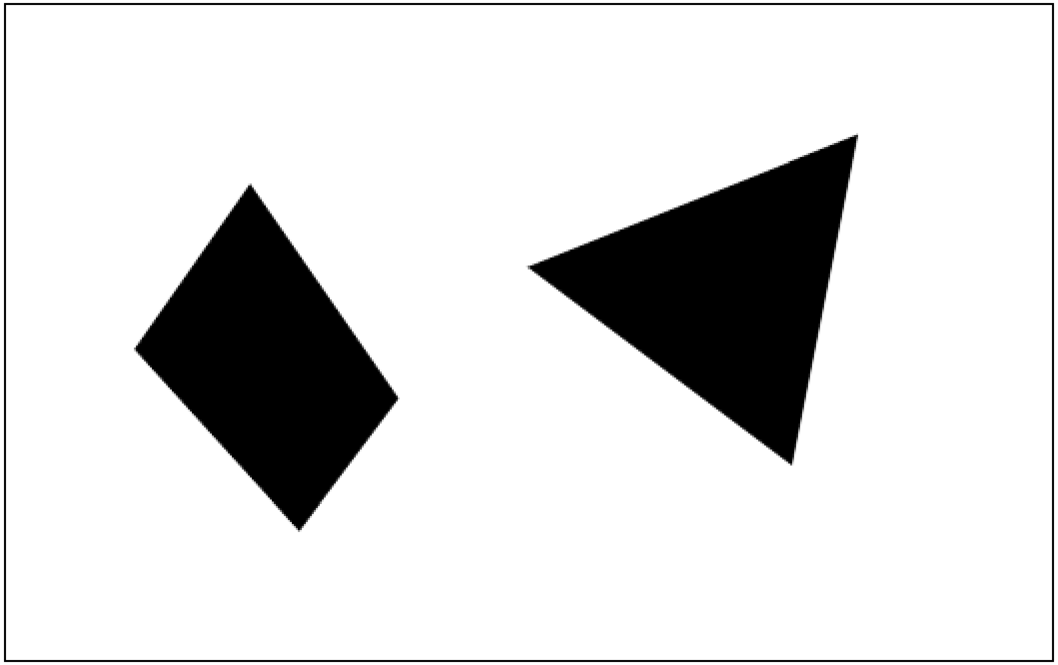

Figure from Meyer, “Map-based navigation in mobile robotics”, 2003, some other figures follow

# Path Planning

- **Methodologies**
  - Roadmap
  - Cell decomposition
- **Roadmap**
  - Derive a graph from free space
  - Graph building
    - Visibility graph
    - Voronoi Diagram
- **Cell decomposition**
  - Free space is decomposed into simple regions (cells)
  - Path between two cells can be easily generated

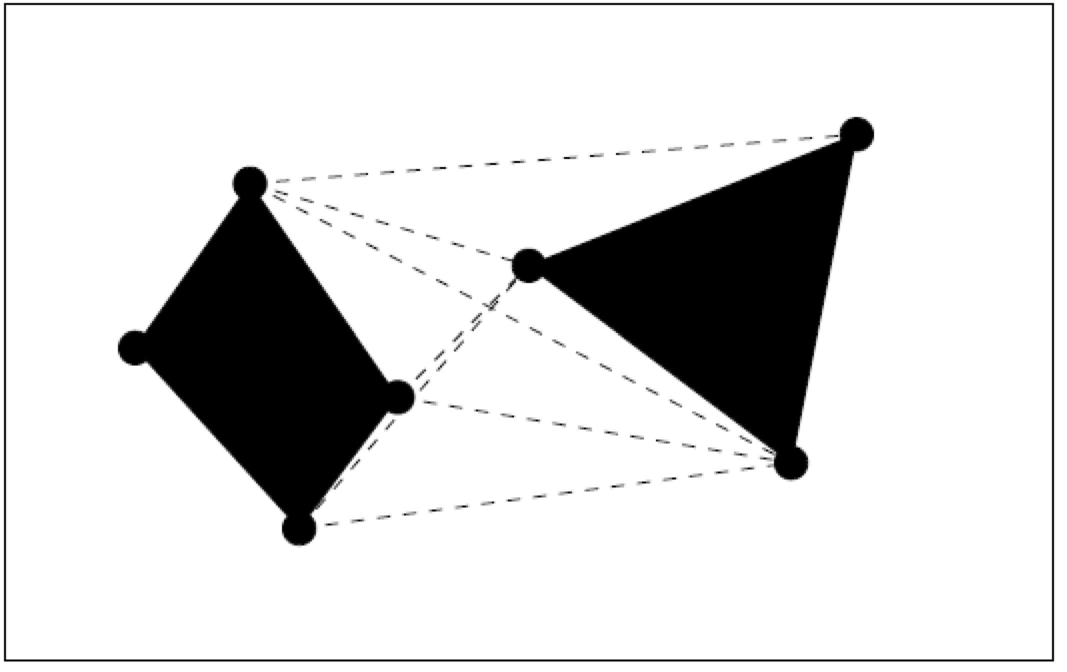
# Visibility graph

- **Graph based representation**
  - Nodes are obstacles angles
  - Edges connect nodes that are visible from each other



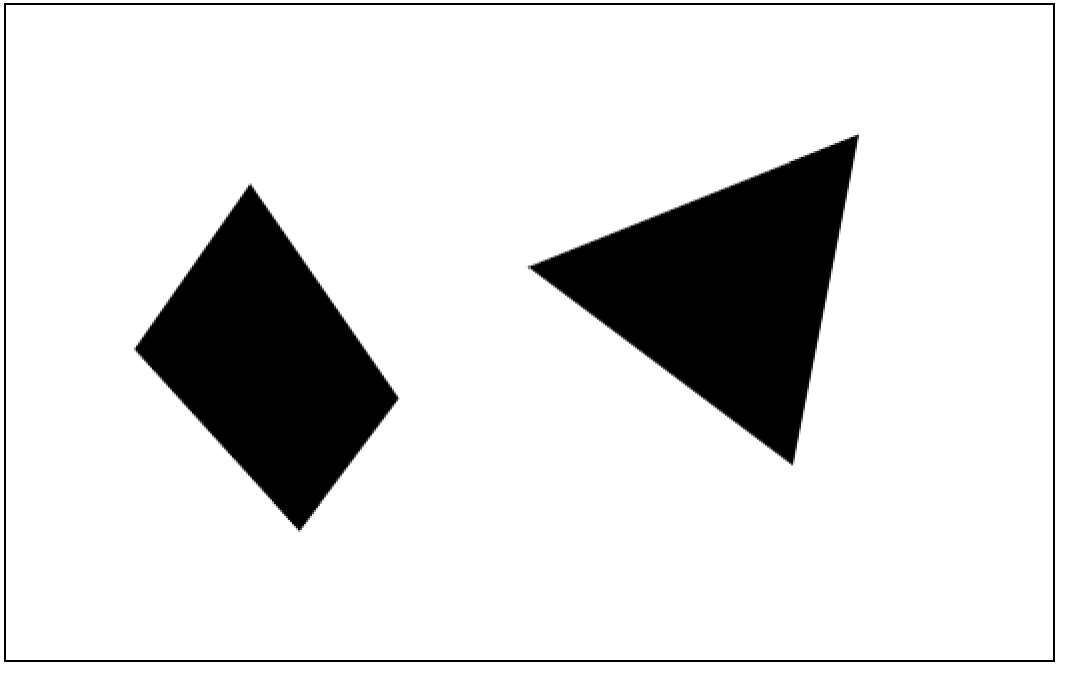
# Visibility graph

- **Graph based representation**
  - Nodes are obstacles angles
  - Edges connect nodes that are visible from each other



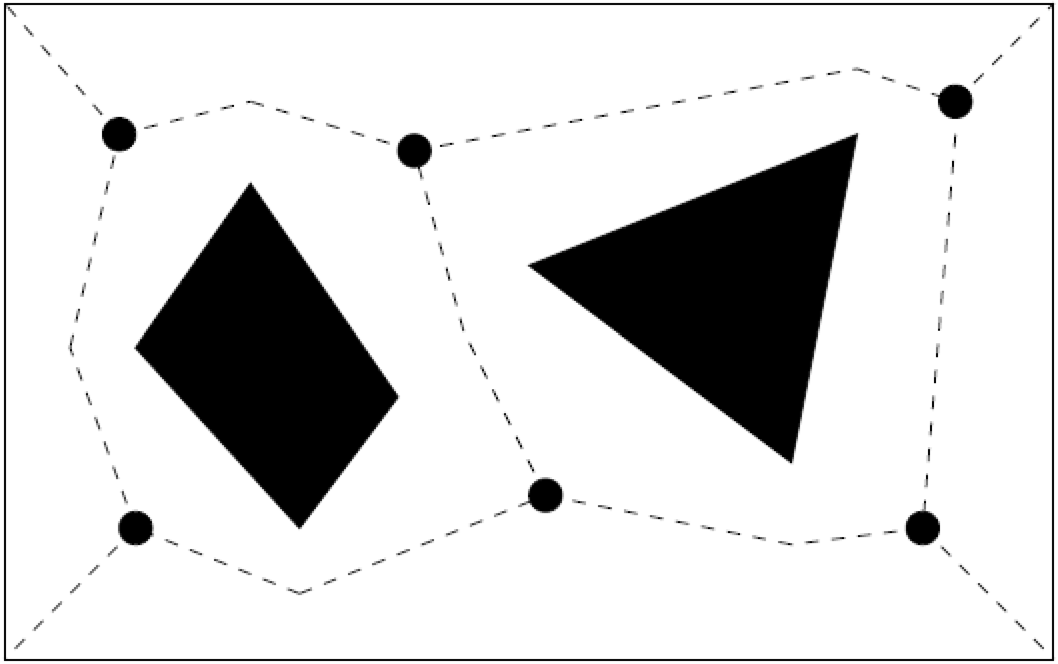
# Voronoi diagram

- **Graph based representation**
  - Voronoi edges are equidistant to closest obtacles
  - Nodes are situated at the points where edges meet



# Voronoi diagram

- **Graph based representation**
  - Voronoi edges are equidistant to closest obtacles
  - Nodes are situated at the points where edges meet



# Graph based planning

- **Search the graph to find optimal path**
- **Which path is optimal?**
  - Minimal distance
  - Safest
  - Best view!
- **Searching algorithms**
  - Dikjstra
  - A*
  - (…) D*

# Dijkstra algorithm

1. **Init**
   - Assign starting node with a 0 distance, all other nodes with infinite distance, current = start, visited = {}
2. **Update minimum distances of neighbors to current node**
   - While updating minimum distance keep track of previous node in minimum path
3. **Add current to visited set**
4. **Current = minimum distance node AND not in visited**
5. **Repeat from step 2 until current = target**

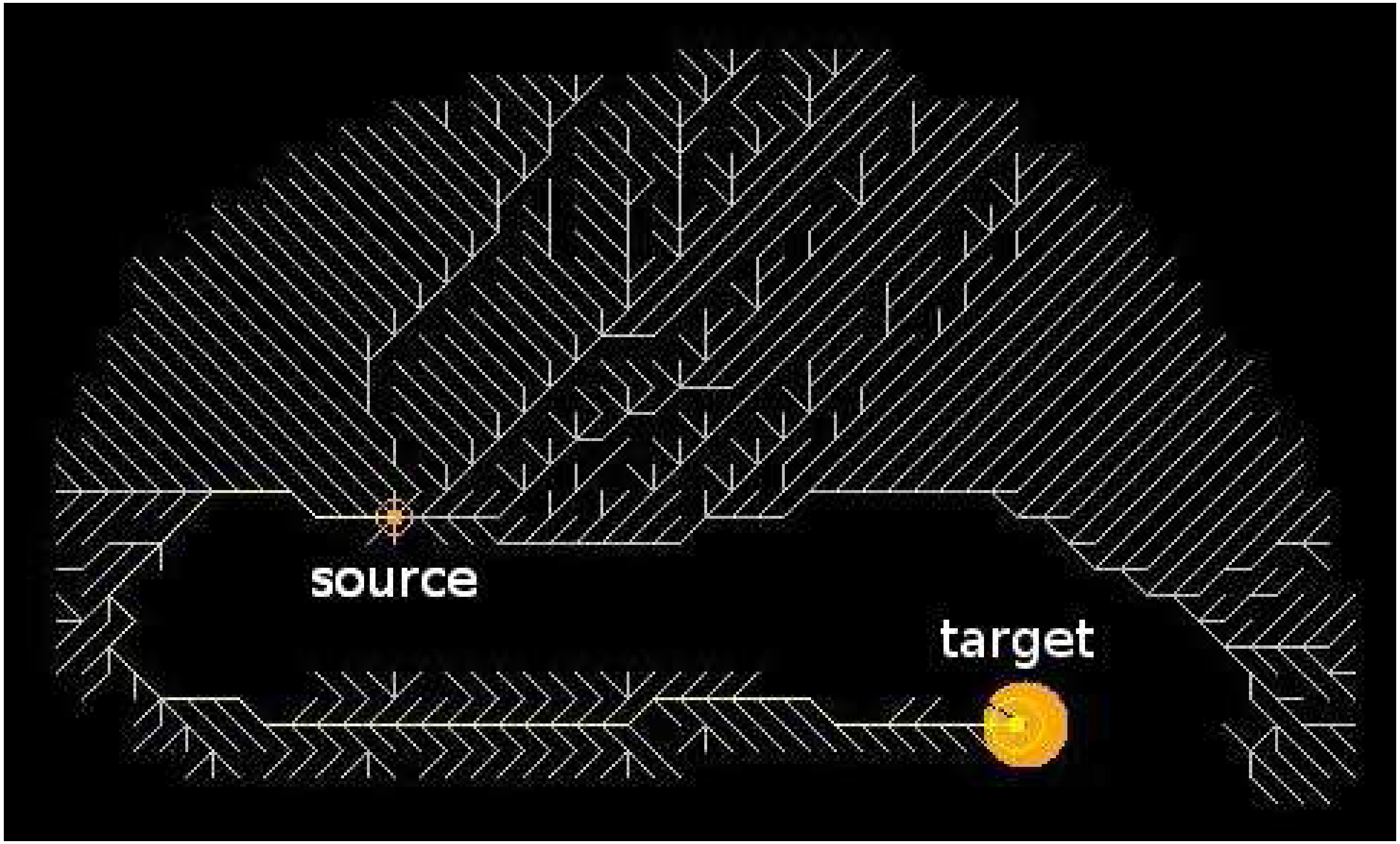
# A* algorithm

**Similar to Dijkstra but selection takes into distance to target into account:**

4. **Current = minimum distance to start + euclidian distance to target node AND not in visited**

- **Returns optimal path**
- **Tends to search in the direction of the target**

# A* algorithm

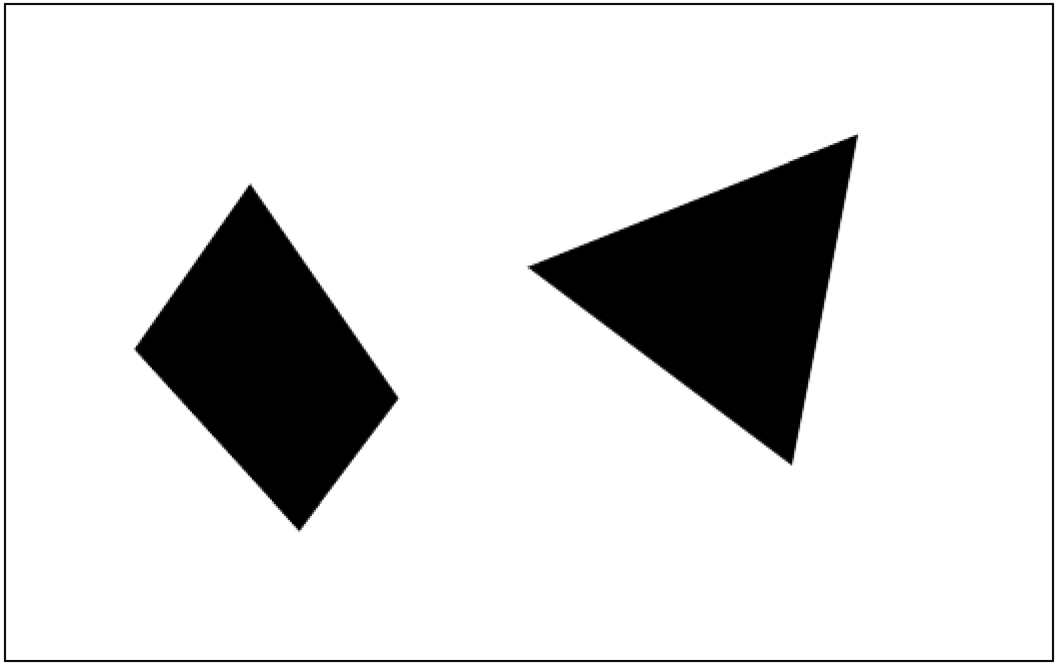


# Cell decomposition

- **Exact cell decomposition**
- **Rectangular cell decomposition**
- **Regular cell decomposition**
- **Quadtree cell decomposition**

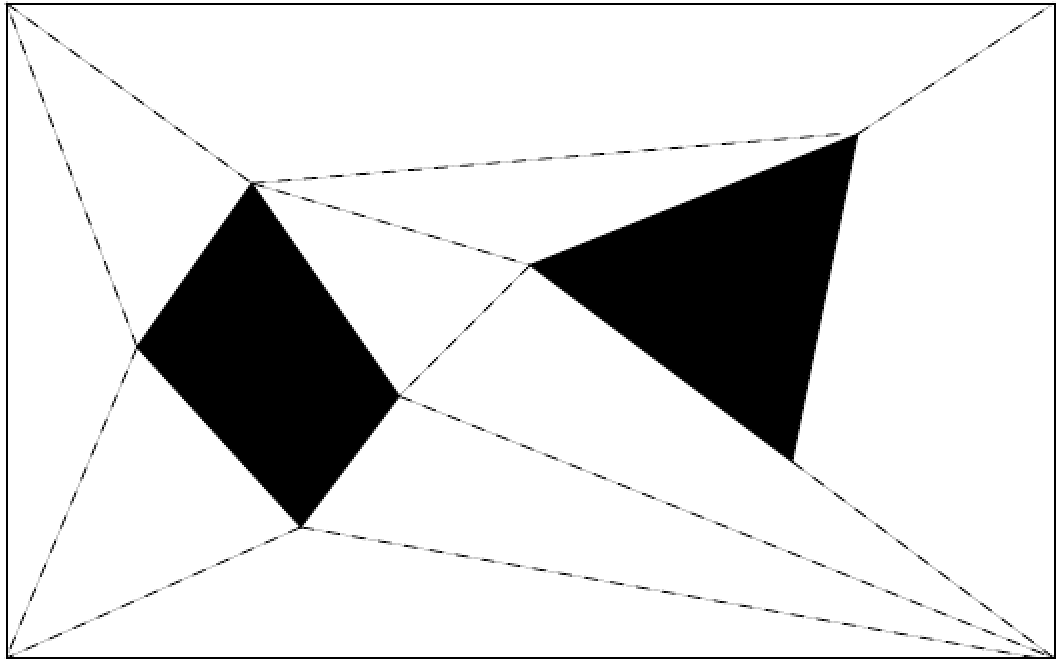
# Cell decomposition

- **Exact cell decomposition**
  - Partition the free space into convex polygons



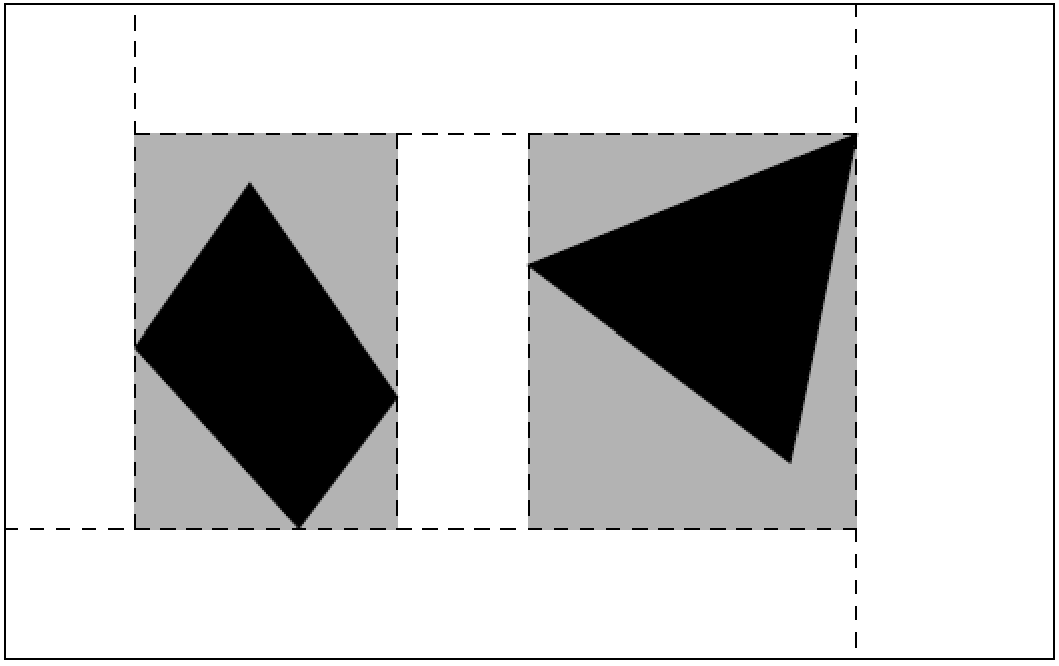
# Cell decomposition

- **Exact cell decomposition**
  - Partition the free space into convex polygons



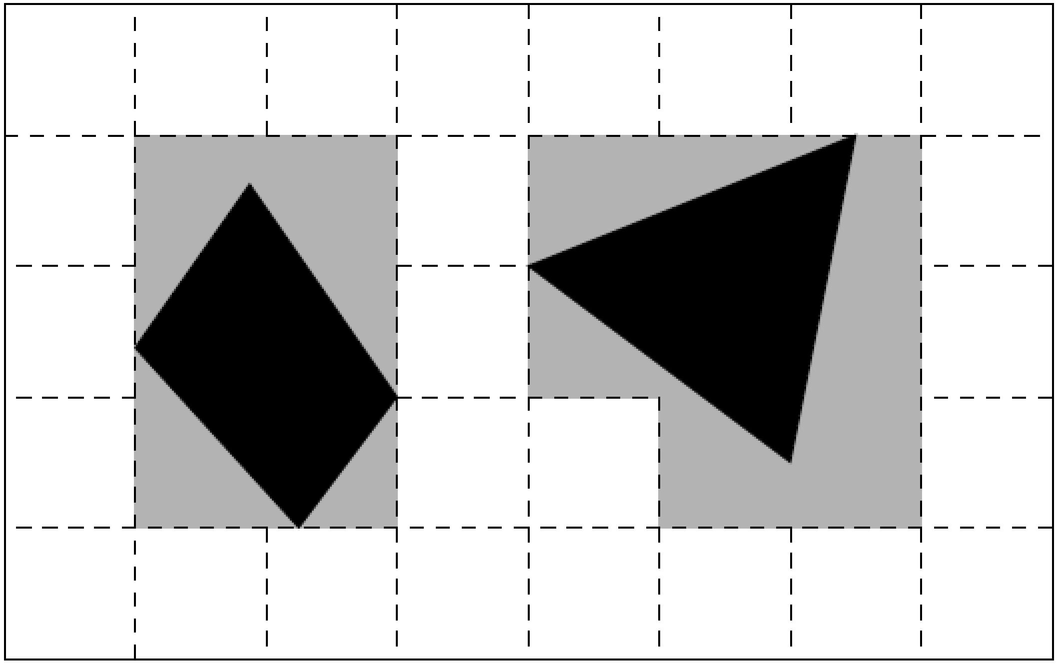
# Cell decomposition

- **Rectangular cell decomposition**



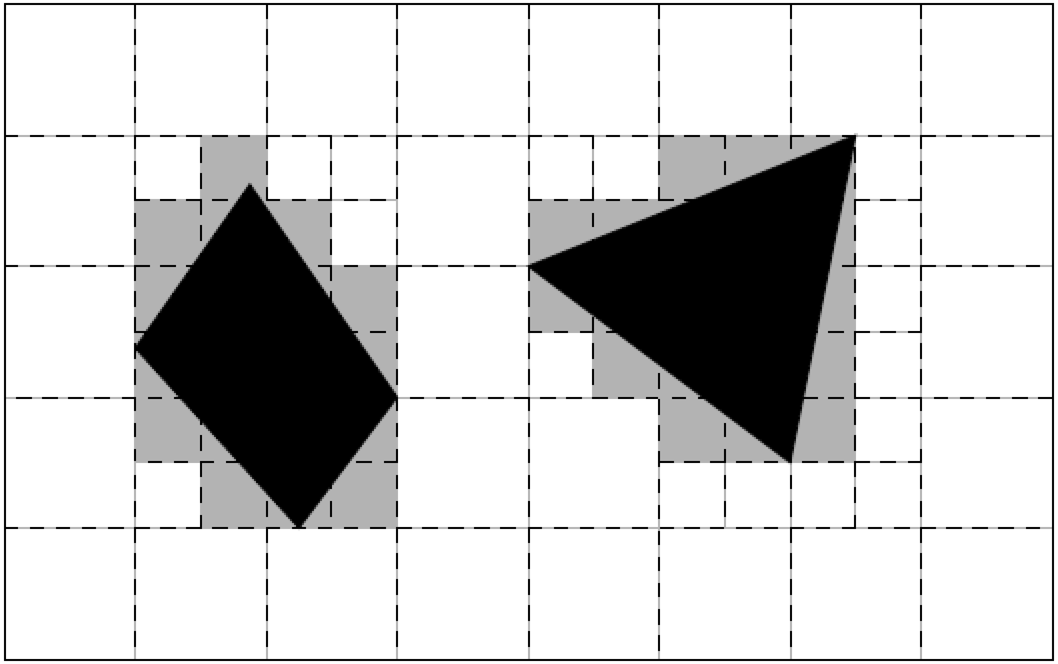
# Cell decomposition

- **Regular cell decomposition**

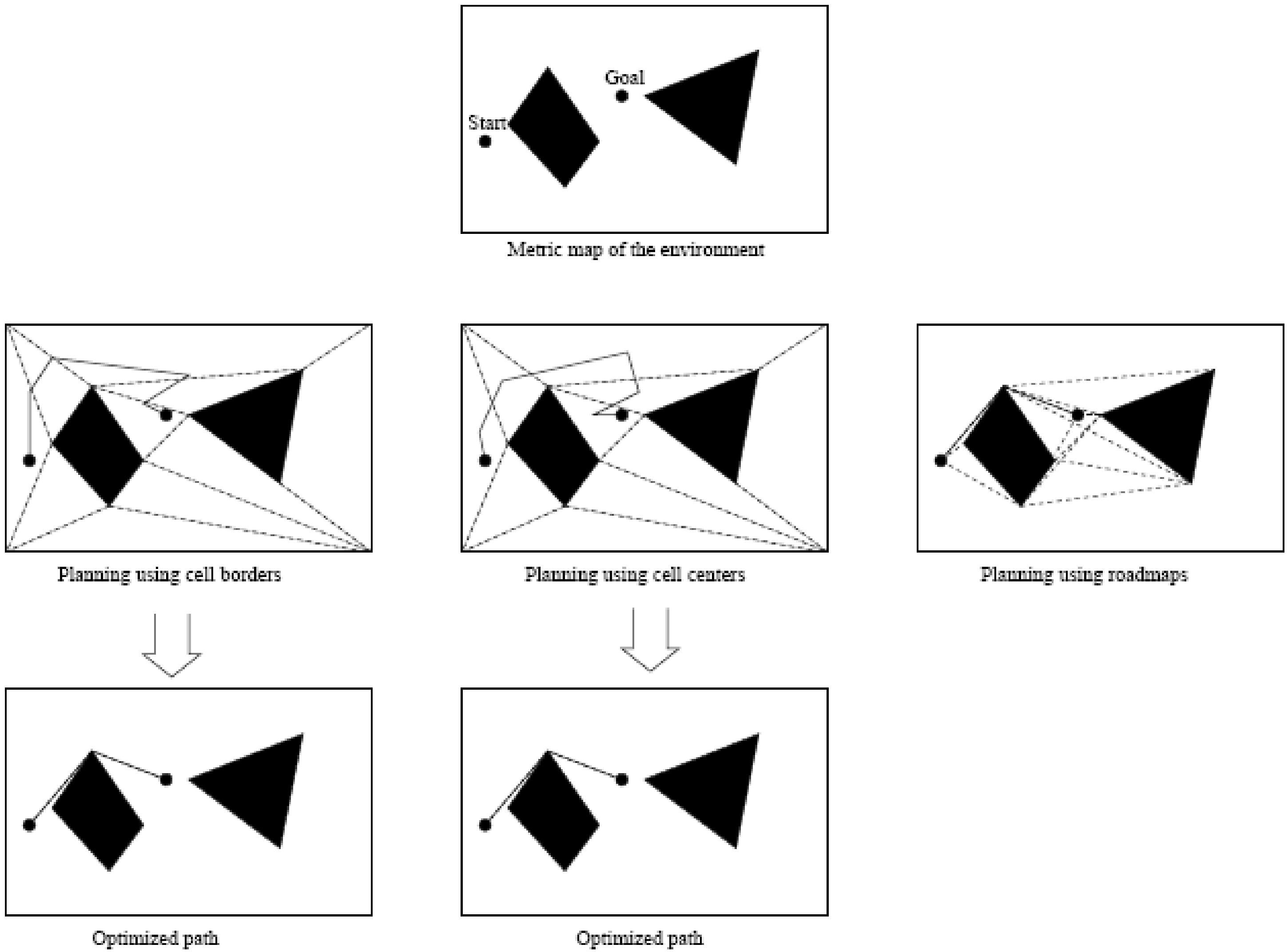


# Cell decomposition

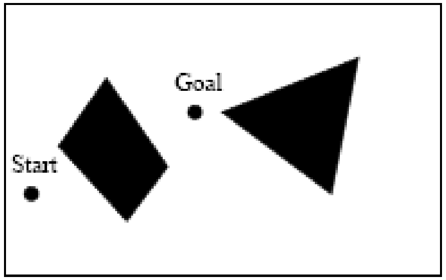
- **Quadtree cell decomposition**



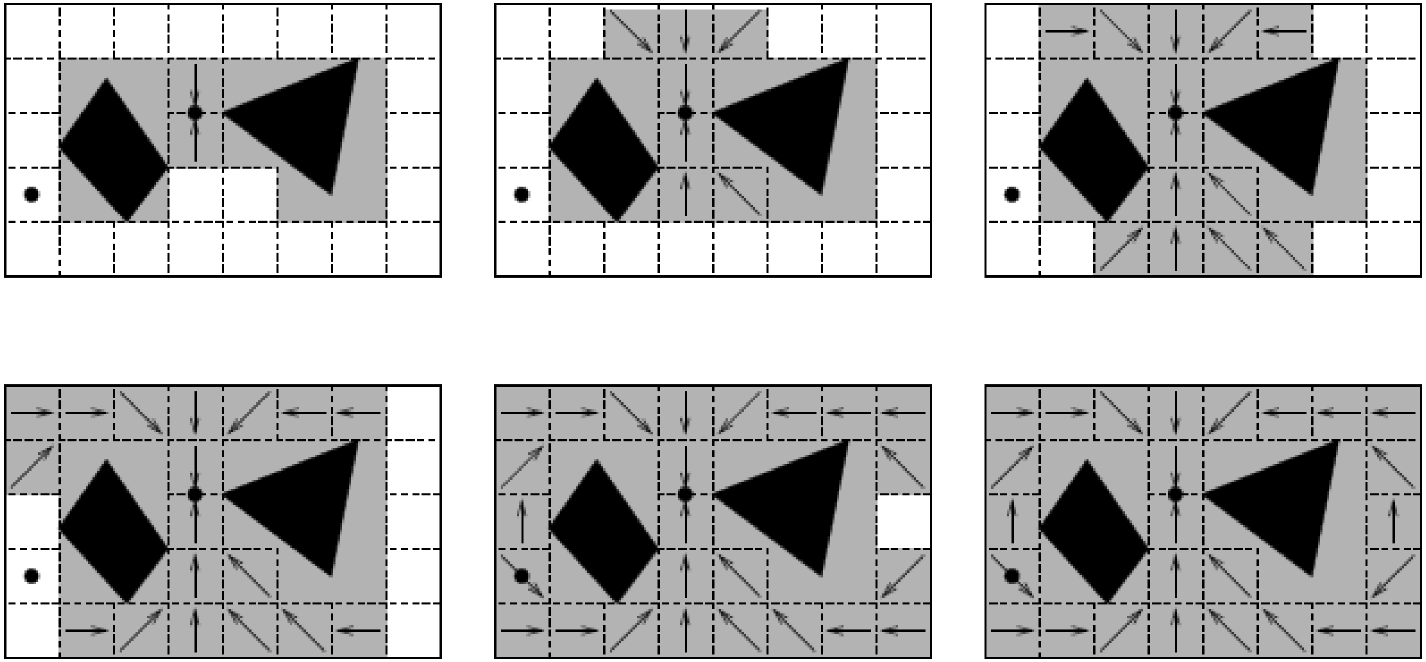
# Planning


Goal
Start
Metric map of the environment
Planning using cell borders
Planning using cell centers
Planning using roadmaps
Optimized path
Optimized path

# Wavefront planning

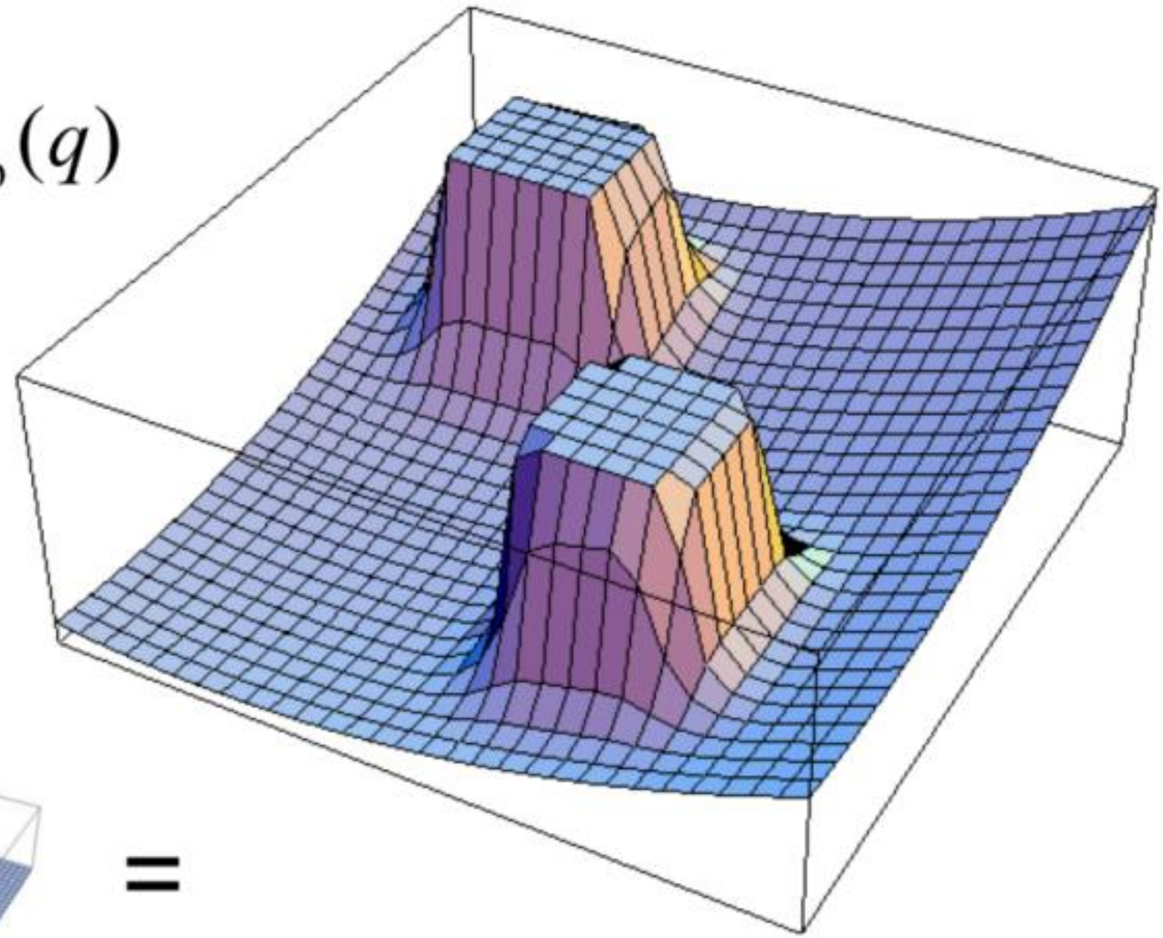


Metric map of the environment

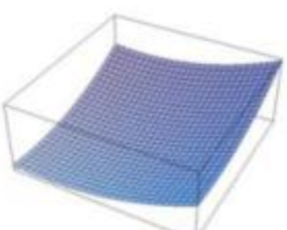


# Potential Field Local Planning

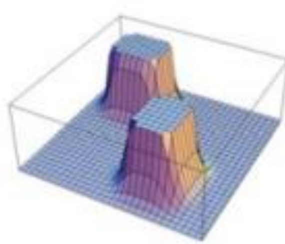
$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$



$$F(q) = -\nabla U(q)$$



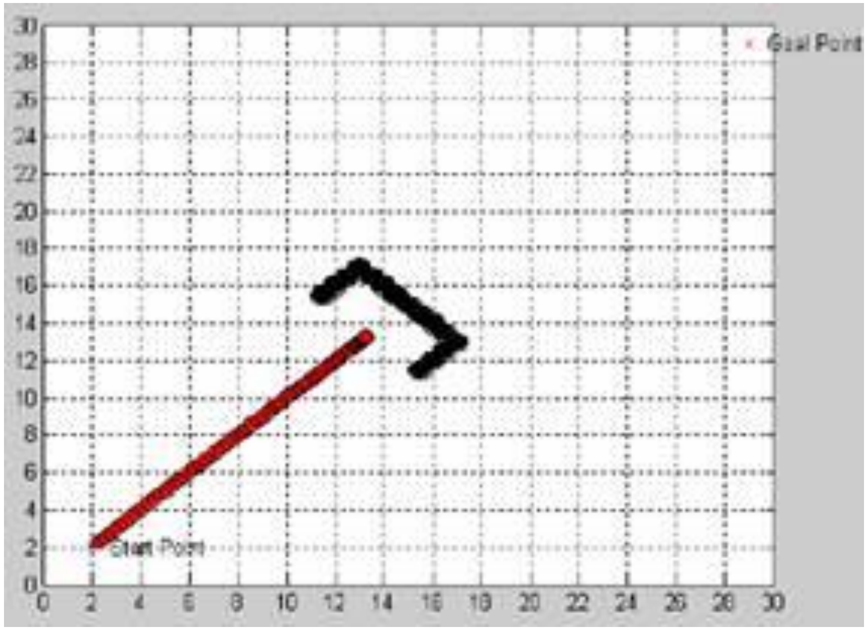
+

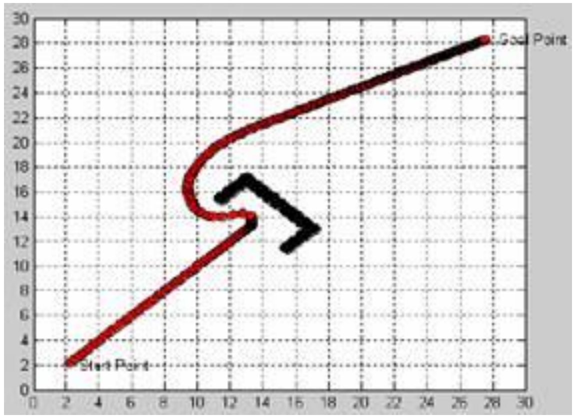


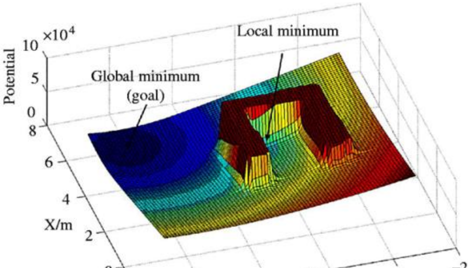
=

# Potential Field Planning

- http://www.cs.mcgill.ca/~hsafad/robotics/index.html



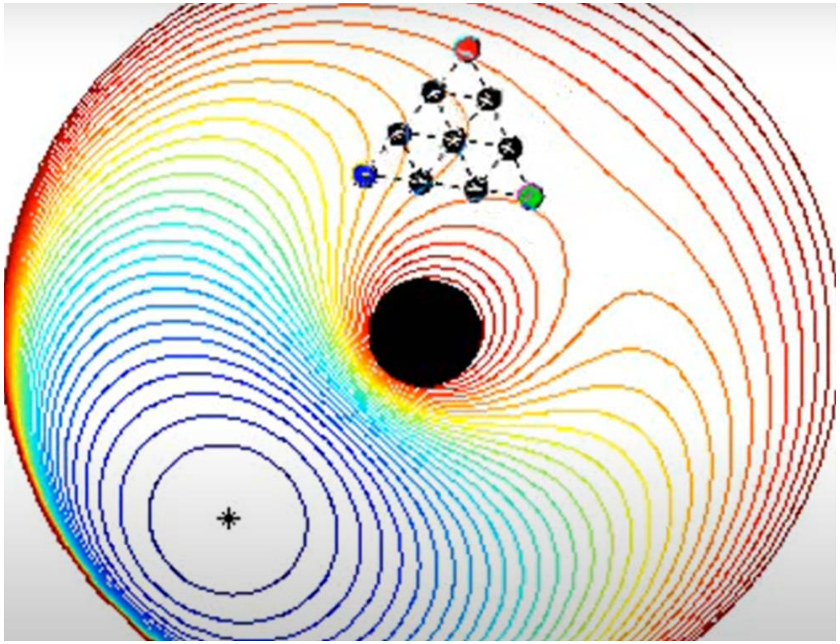




http://www.emeraldinsight.com/journals.htm?issn=0143-991X&volume=37&issue=4&articleid=1846407&show=pdf

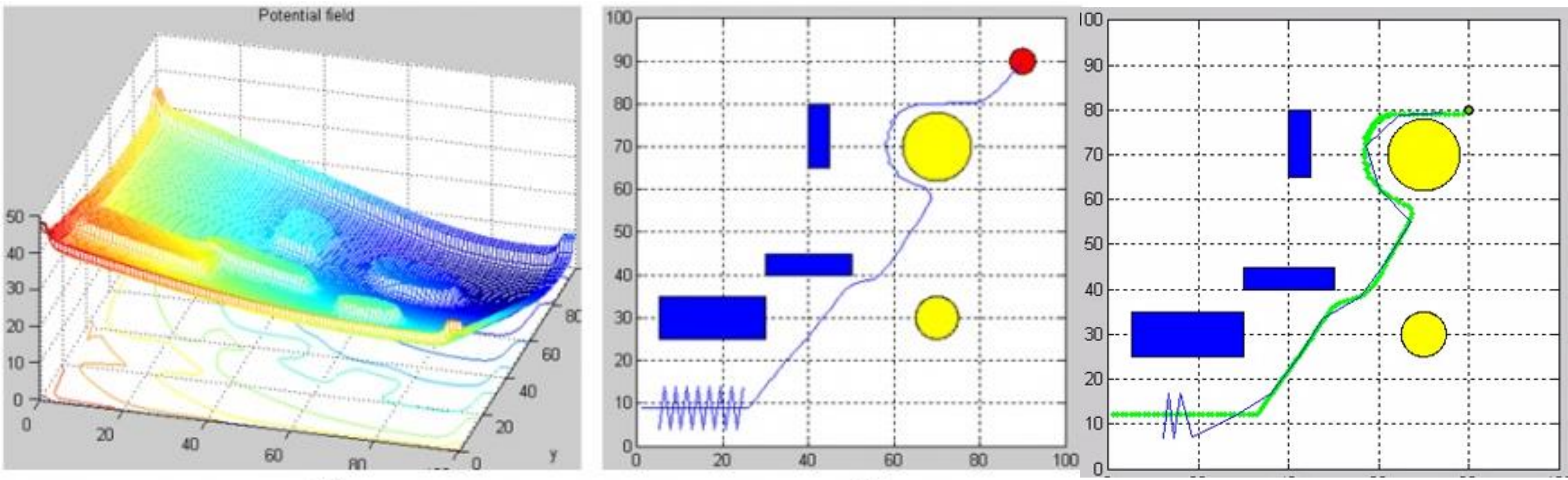
# Also in Swarm

- http://youtu.be/r9FD7P76zJs



# Also Manipulators…

- http://taylorwang.files.wordpress.com/2012/04/potential-field1_robot.jpg
- http://taylorwang.wordpress.com/2012/04/06/collision-free-path-planning-using-potential-field-method-for-highly-redundant-manipulators/
- http://youtu.be/QTp1HRjXSSc




(a) (b)

# Mapping

- **Mapping is the process of building an internal estimate of the metric map of the environment**
- **What should be represented?**
  - Each cell is occupied or unoccupied
  - Each cell has a continuous value (high-occupied, low unoccupied)
  - Probability that each cell is occupied P(H) and probability that each cell is unoccupied P(~H)

    $0 <= P(H) <= 1$

    $1-P(H) = P(\sim H)$

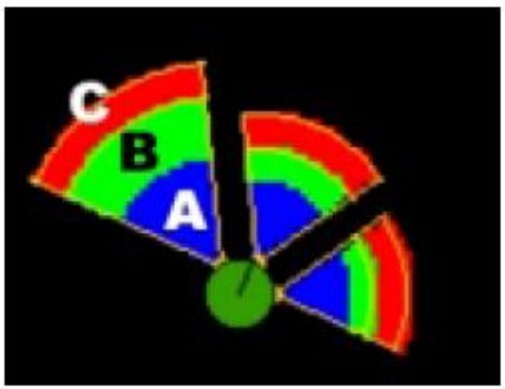
# Mapping

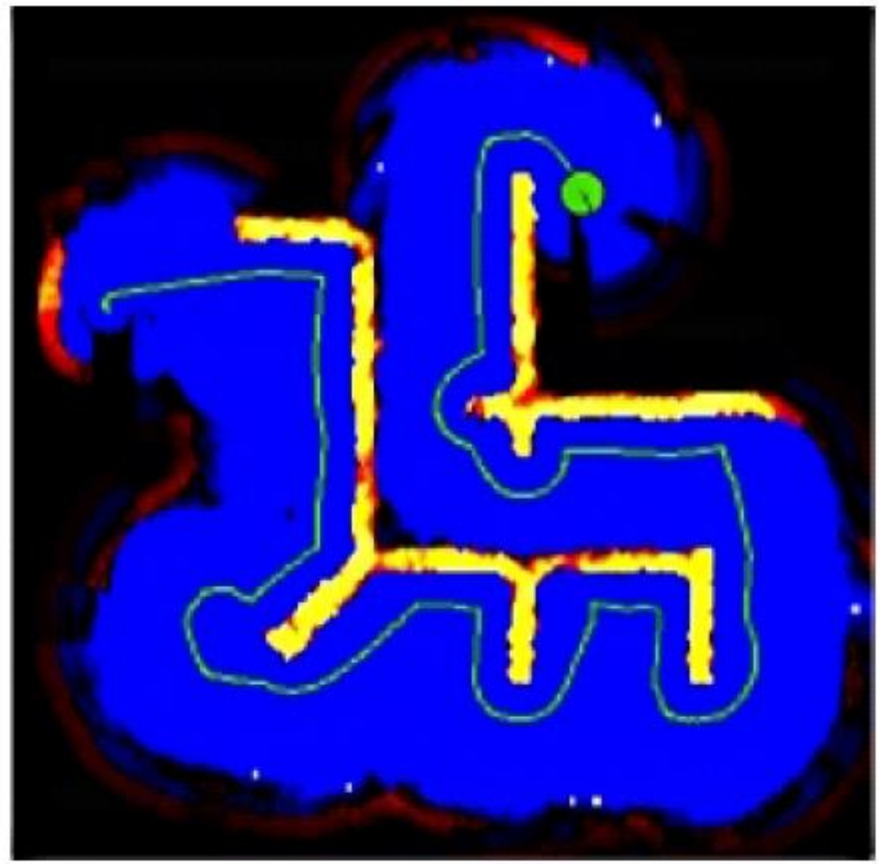
- **Mapping is the process of building an internal estimate of the metric map of the environment**
- **What should be represented?**
  - Each cell is occupied or unoccupied
  - Each cell has a continuous value (high-occupied, low unoccupied)
  - Probability that each cell is occupied P(H) and probability that each cell is unoccupied P(~H)

    $0 \le P(H) \le 1$

    $1 - P(H) = P(\sim H)$

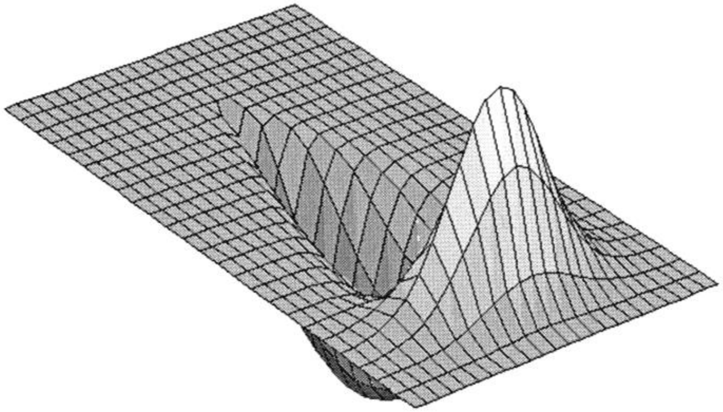
# Mapping



# Mapping

- **Conditional probabilities**
  - We want to determine P(H|s)
    - Probability cell is occupied given a certain measure s
  - Let's start by determining P(s|H)
    - Probability of getting measure s if there is H is occupied
      - In general P(H|s) is not equal to P(s|H)
    - This is the sensor model



# Mapping

- **Conditional probabilities**
  - We want to determine P(H|s)
  - From Bayes' Rule:

$$P(H \mid s) = \frac{P(s \mid H).P(H)}{P(s \mid H)P(H) + P(s \mid \sim H)P(\sim H)}$$

  - P(s|H) and P(s|~H) are known from the sensor model
  - P(H) and P(~H) are unconditional probabilities or prior probabilities
    - If no information is available P(H)=P(~H)=0.5 can be assumed

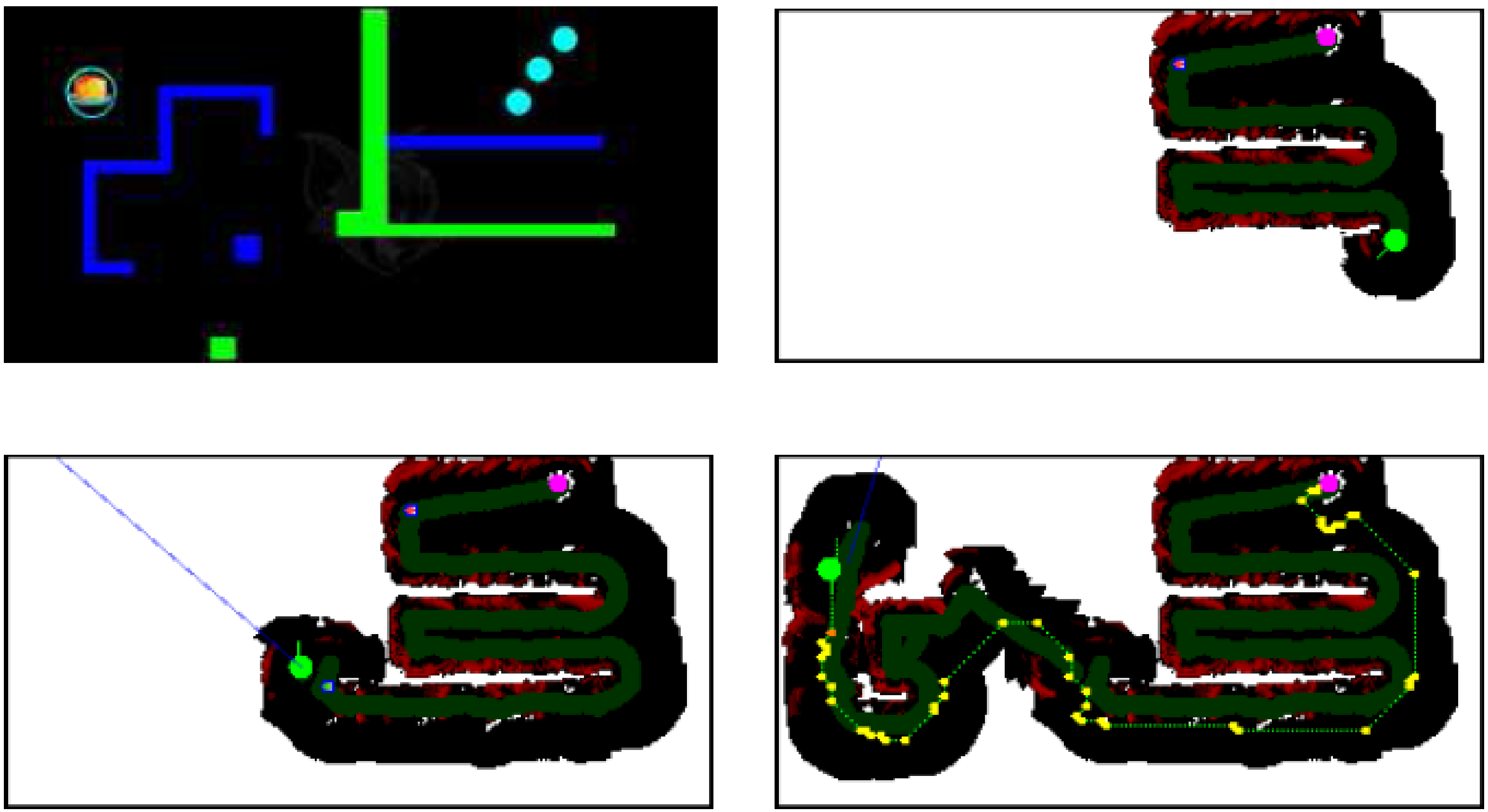
# Mapping

- **Updating with the Bayes' rule**
  – How to fuse the computed probabilities with new sensor readings?

$$P(H \mid s_n) = \frac{P(s_n \mid H).P(H \mid s_{n-1})}{P(s_n \mid H)P(H \mid s_{n-1}) + P(s_n \mid \sim H)P(\sim H \mid s_{n-1})}$$

  – This is the recursive version of the update formula
  – Each time a new observation is made it can be employed to update the occupancy grid

# Mapping



# Mapping

- **Dempster-Schafer Theory**
  - Belief functions instead of probabilities
    - Measure belief mass
    - Each sensor contributes with a belief mass of 1.0
    - Can distribute the mass to a set of propositions
  - Set of propositions is the Frame of Discernement (FOD)
    - In the case of occupancy grid FOD={Occupied, Empty}
    - It may include non exclusive propositions
    - A sensor reading may be considered as ambiguous

# Mapping

- **Dempster-Schafer Belief function properties**
  - Bel(X) measures the likelihood that previous evidence supports X
  - Bel({}) = 0
  - Bel ( set of all subsets of FOD ) = 1
  - In the case of the occupancy grid:

    Bel = m({Occupied}), m({Empty}), m(dontknow)

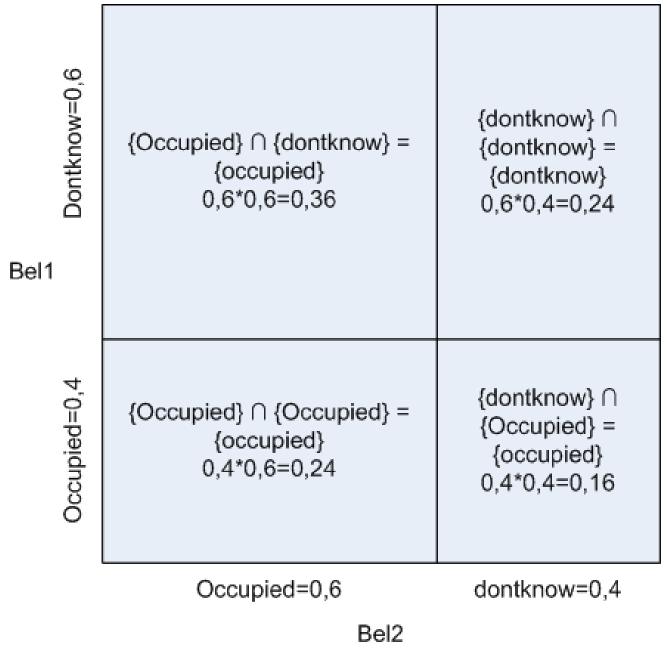
    dontknow={Occupied,Empty}

# Mapping

- **Belief function for sonar**
  - Region that supports evidence of having an obstacle
    - m(occupied)=evidence
    - m(empty)=0
    - m(dontknow)=1-evidence
  - Region that supports evidence of being empty
    - m(occupied)=0
    - m(empty)=evidence
    - m(dontknow)=1-evidence
  - The main difference from probabilities is that uncertainties in the reading count as belief mass for dontknow

# Mapping

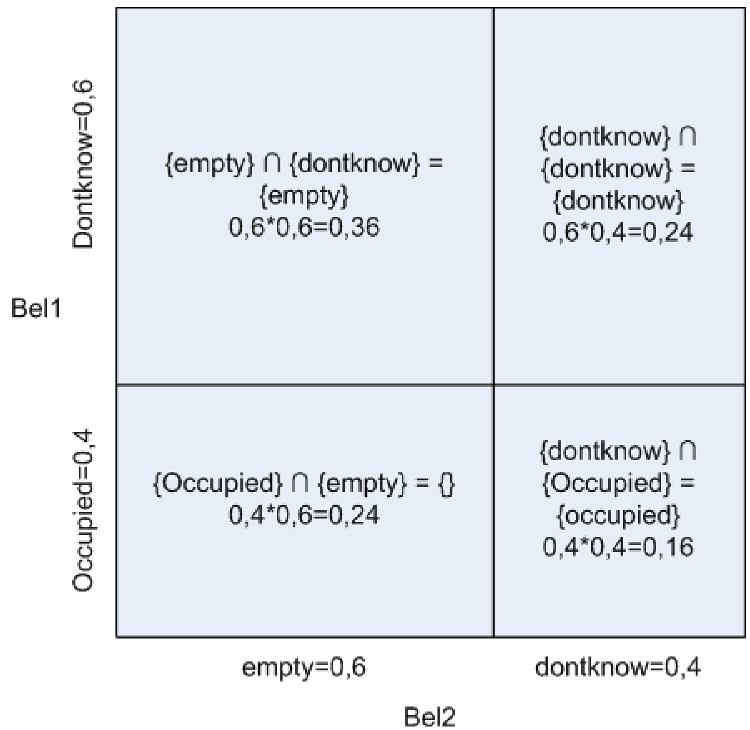
- **Rule of combination**
  - $Bel_1$ = m(occupied)=0.4, m(empty)=0, m(dontknow)=0.6
  - $Bel_2$ = m(occupied)=0.6, m(empty)=0, m(dontknow)=0.4




  - $Bel_3$ = m(occupied)=0,76, m(empty)=0,0, m(dontknow)=0,16

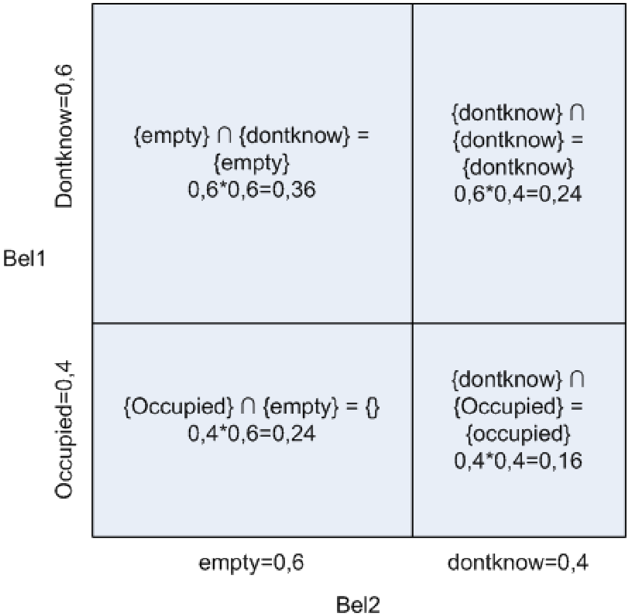
# Mapping

- **Rule of combination**
  - $Bel_1$ = m(occupied)=0.4, m(empty)=0, m(dontknow)=0.6
  - $Bel_2$ = m(occupied)=0.0, m(empty)=0,6, m(dontknow)=0.4



# Mapping

- **Rule of combination**




  - As m({}) must be 0, a normalization is needed
  - $Bel_3$ = m(occupied)=0,16/(1-0,24)=0,21
    m(empty)=0,36/(1-0,24)=0,47
    m(dontknow)=0,24/(1-0,24)=0,32

# Mapping

- **Rule of combination**

$$m(C_k) = \frac{\sum_{A_i \cap B_j = C_k ; C_k \neq \{\}} m(A_i).m(B_j)}{1 - \sum_{A_i \cap B_j = \{\}} m(A_i).m(B_j)}$$

  - It must be repeated for every subset of the orthogonal sum

# Additional Sources

- https://www.youtube.com/watch?v=Ls8EBoG_SEQ

Intelligent Robotics

# Mapping and Navigation

Luís Paulo Reis (University of Porto)

Nuno Lau (University of Aveiro)

Armando Sousa (University of Porto)