Path Planning

Team Parking

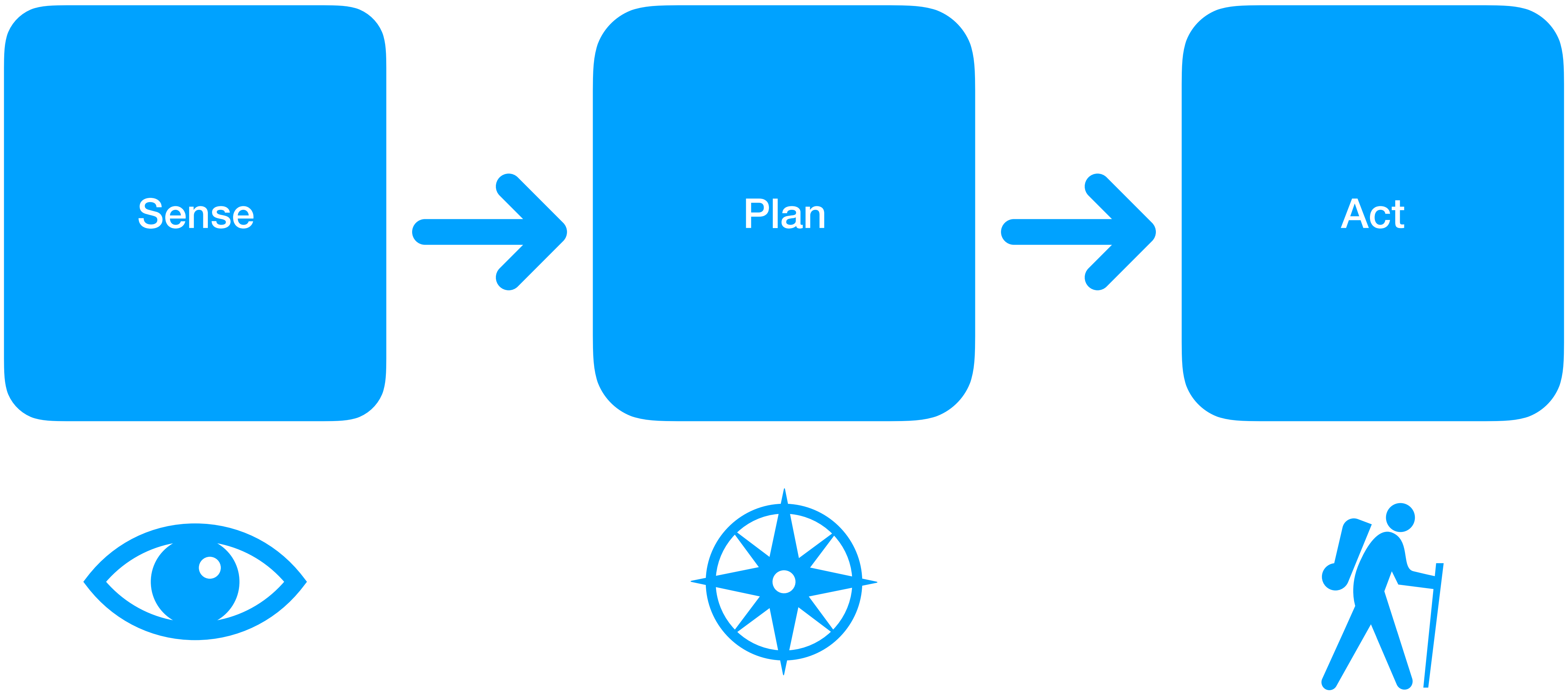
Daniel Svendsen

09.05.2019

fortiss

Table of Contents

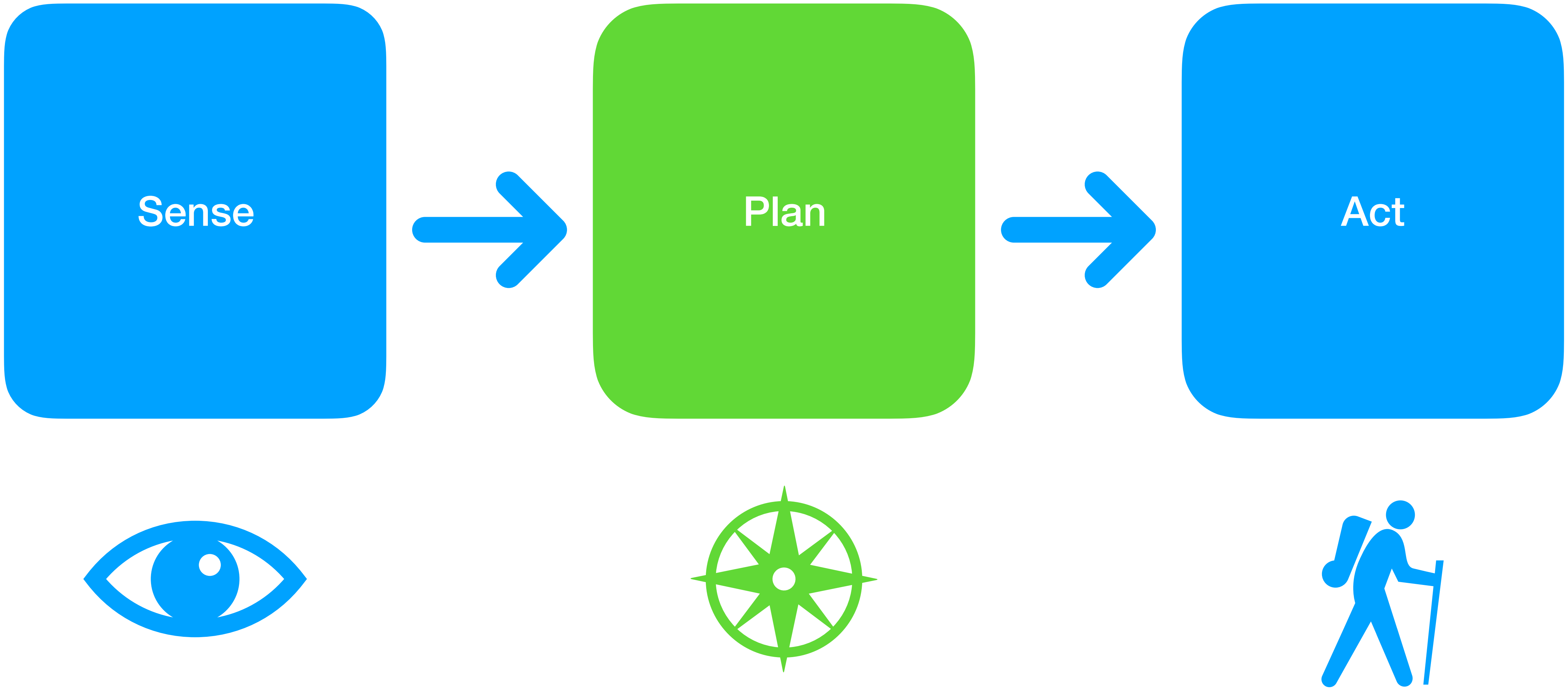
- Path planning
- BUG - Algorithm



Sense

Plan

Act



Sense

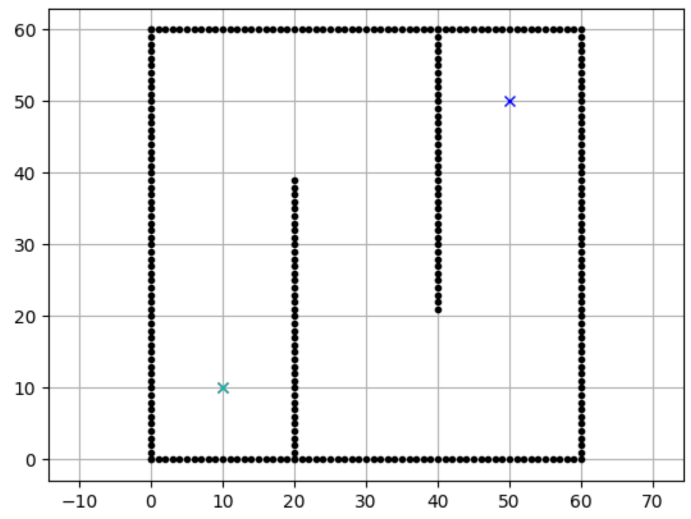
Plan

Act

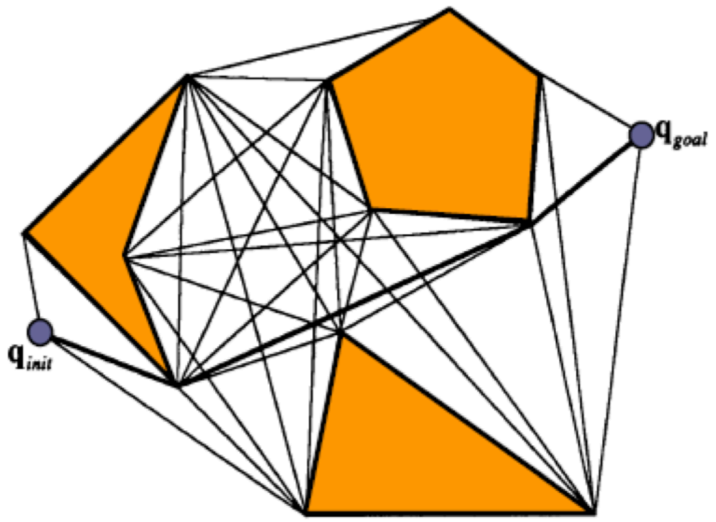
“A **nonholonomic system** is a system whose state depends on the path taken in order to achieve it. ”

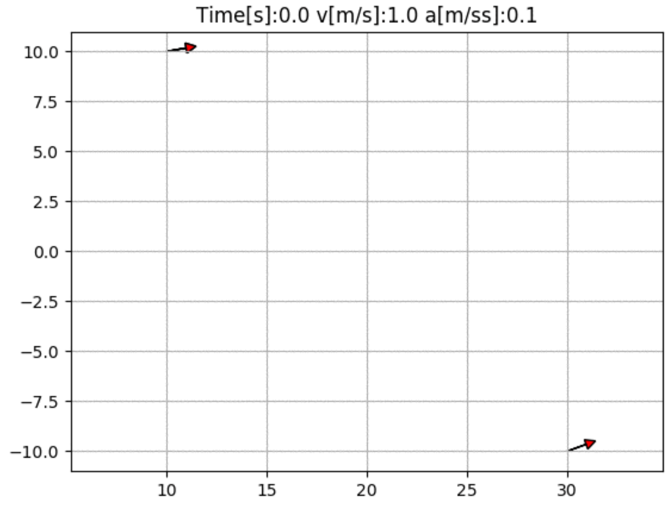
https://en.wikipedia.org/wiki/Nonholonomic_system

Algorithms



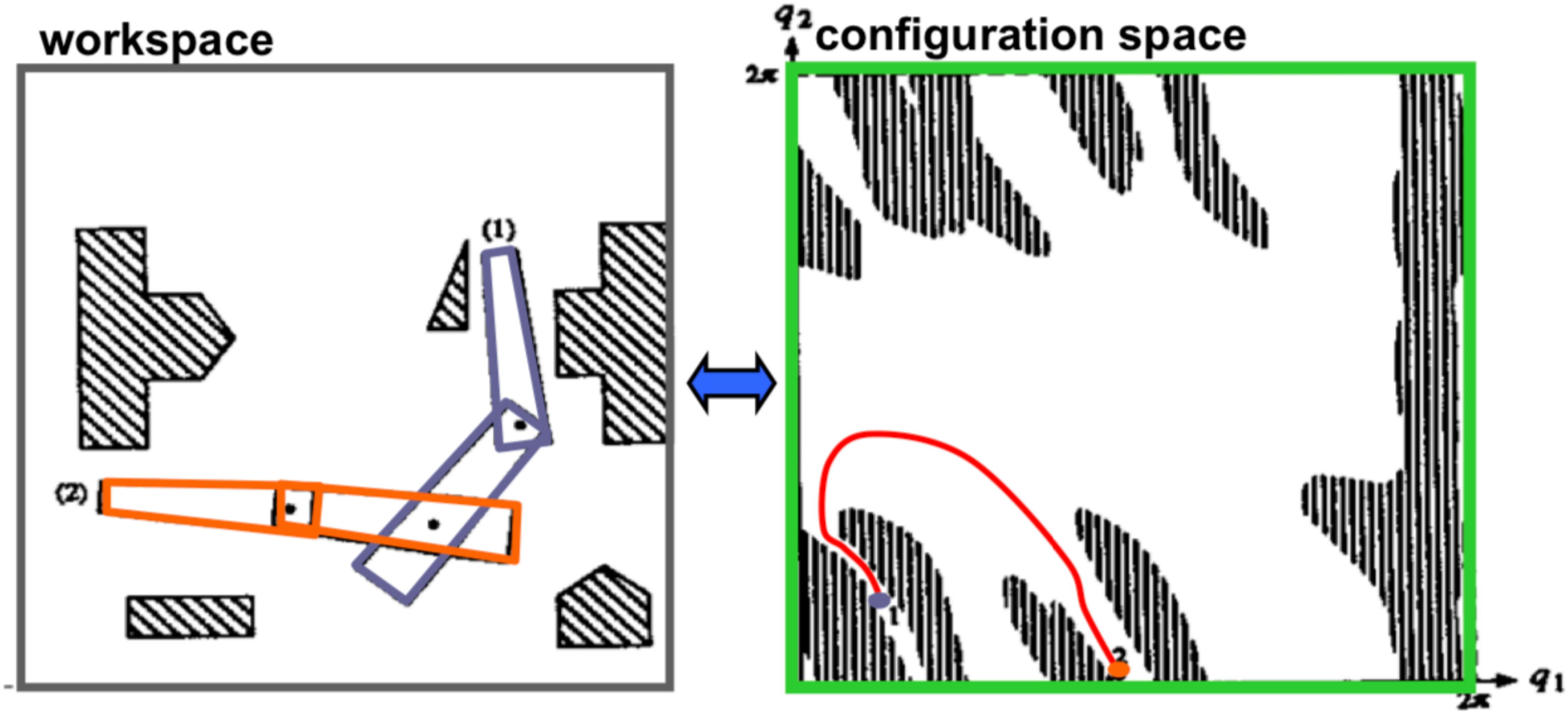
- Grid based search algorithm (A*)
- Geometric algorithm (Visibility graph)
- High order polynomial algorithm
- Bug algorithm
- And more …





Slides adopted from Robot Motion Planning lecture: http://robvis01.informatik.tu-muenchen.de/courses/motion/slides/Set2-PathPlanning.pdf

Diagrams: https://github.com/AtsushiSakai/PythonRobotics#quintic-polynomials-planning



Path: Continuous curve connecting two configurations

Trajectory: Path parameterised by time

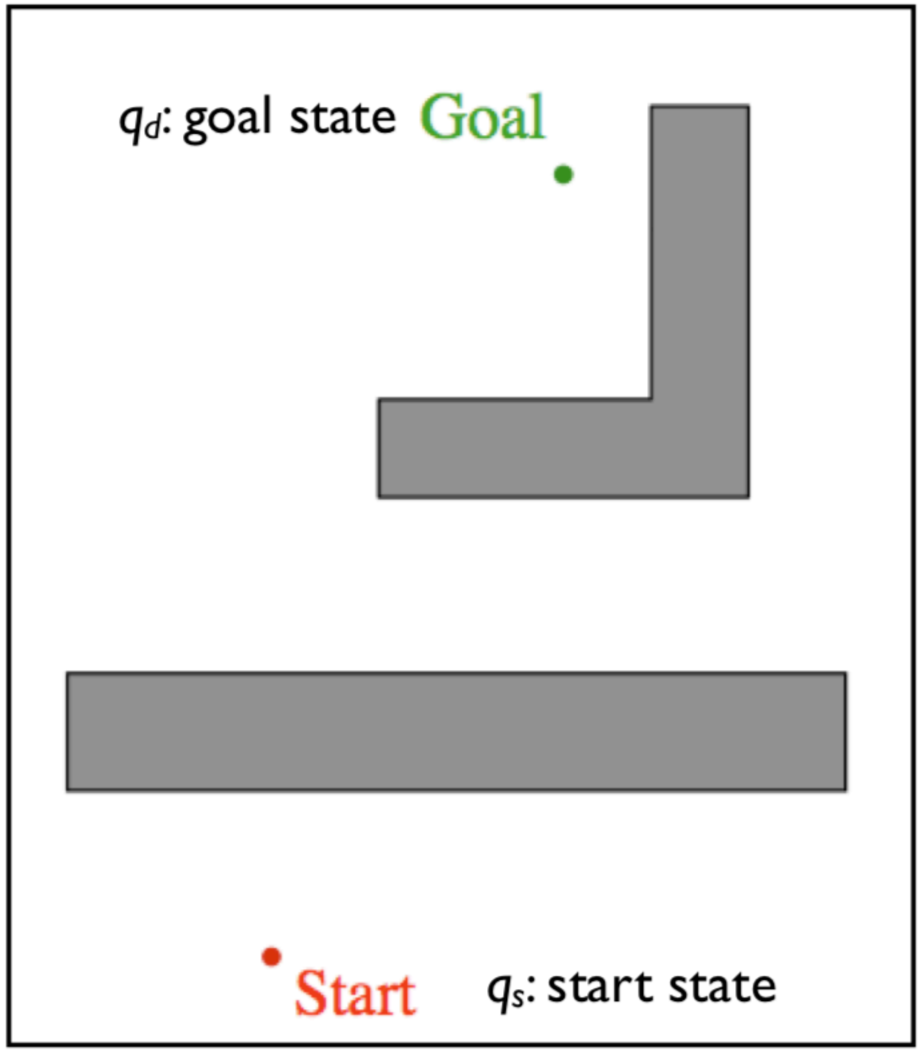
Obstacle: Moving object collides with workspace obstacle

Slides adopted from Robot Motion Planning lecture: http://robvis01.informatik.tu-muenchen.de/courses/motion/slides/Set2-PathPlanning.pdf

Bug Algorithm

- Invented in the late 80`s
- No global model of the world is assumed
- Evolve from the Maze-solving algorithm

Bug 0

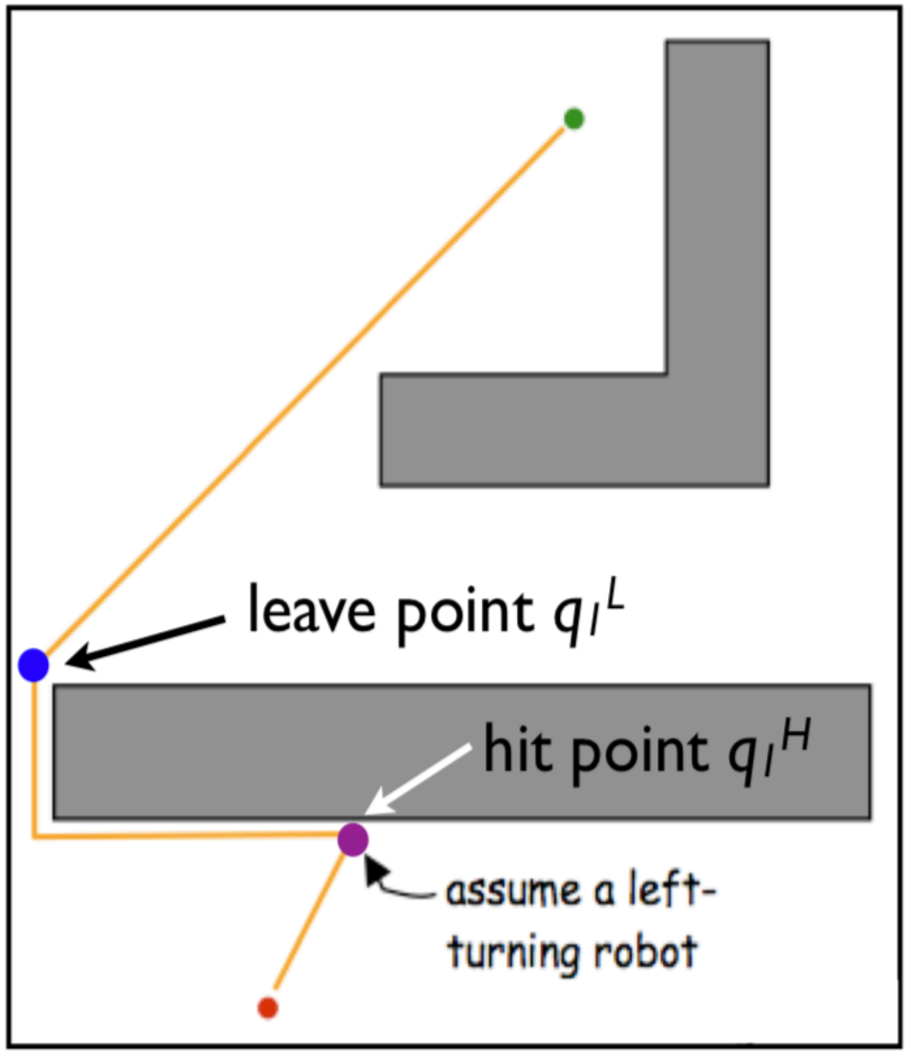


q_s : *Start*

q_d : *Goal*

Slides adopted from Robot Motion Planning lecture: http://robvis01.informatik.tu-muenchen.de/courses/motion/slides/Set2-PathPlanning.pdf

Bug 0



q_s : *Start*

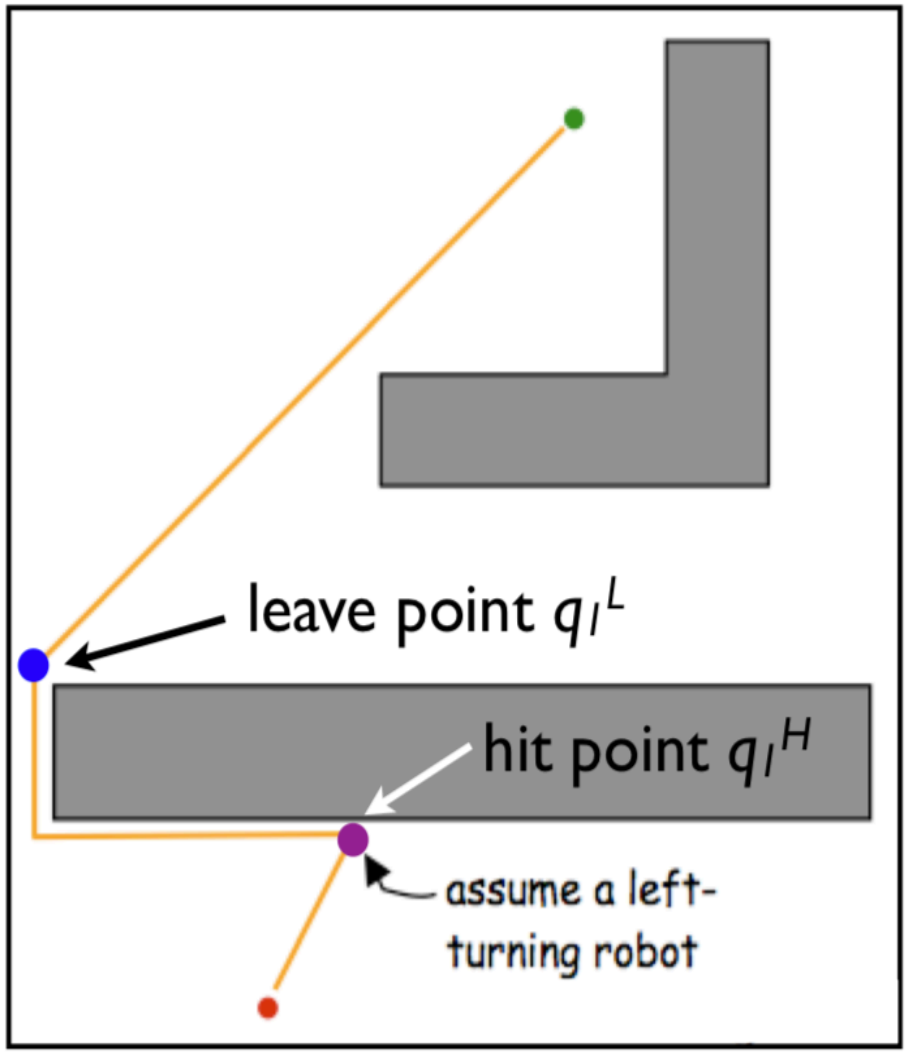
q_d : *Goal*

q_i^H : *Hit*

q_i^L : *Leave*

Slides adopted from Robot Motion Planning lecture: http://robvis01.informatik.tu-muenchen.de/courses/motion/slides/Set2-PathPlanning.pdf

Bug 0



q_s : *Start*

q_d : *Goal*

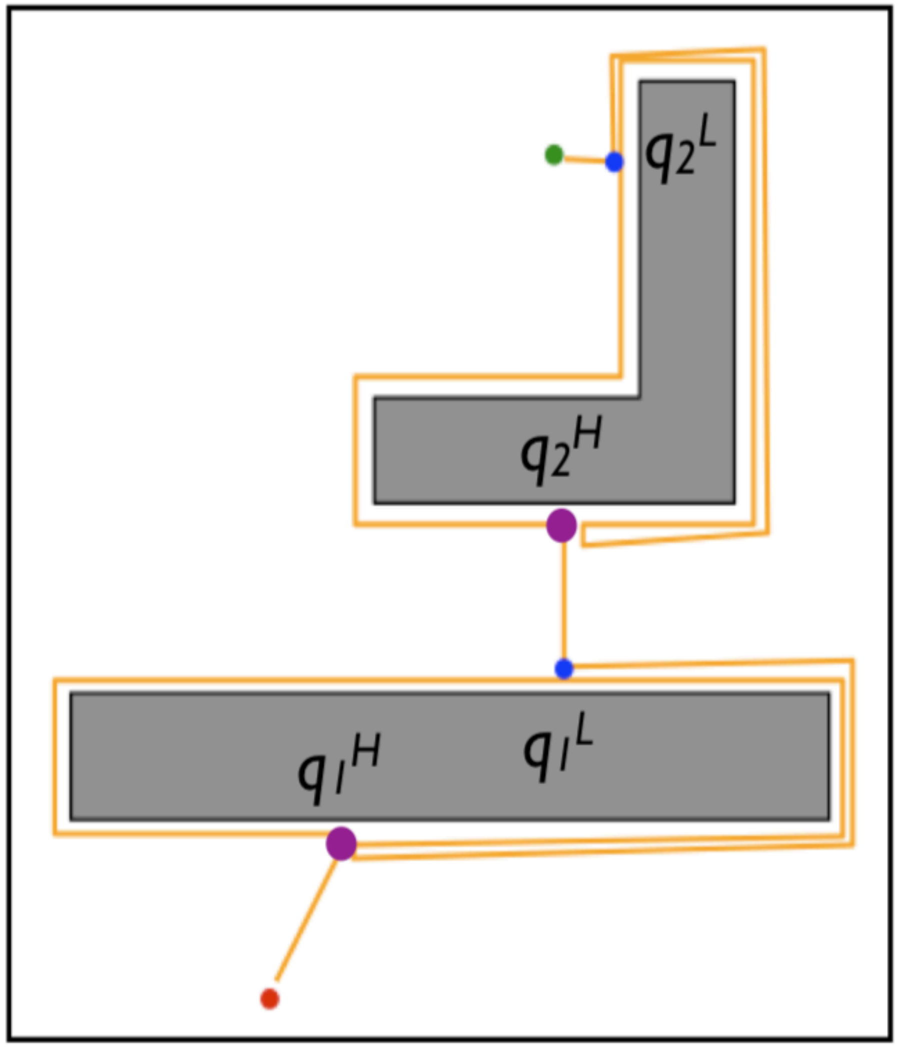
q_i^H : *Hit*

q_i^L : *Leave*

1. **Head towards goal**
2. **When hit, follow wall until leave point**
3. **Repeat step 1**

Slides adopted from Robot Motion Planning lecture: http://robvis01.informatik.tu-muenchen.de/courses/motion/slides/Set2-PathPlanning.pdf

Bug 1



q_s : *Start*

q_d : *Goal*

q_i^H : *Hit*

q_i^L : *Leave*

$$L_{Bug1} \leq d(q_{start}, q) + 1.5 \cdot \sum_{i=1}^{n} p_i$$

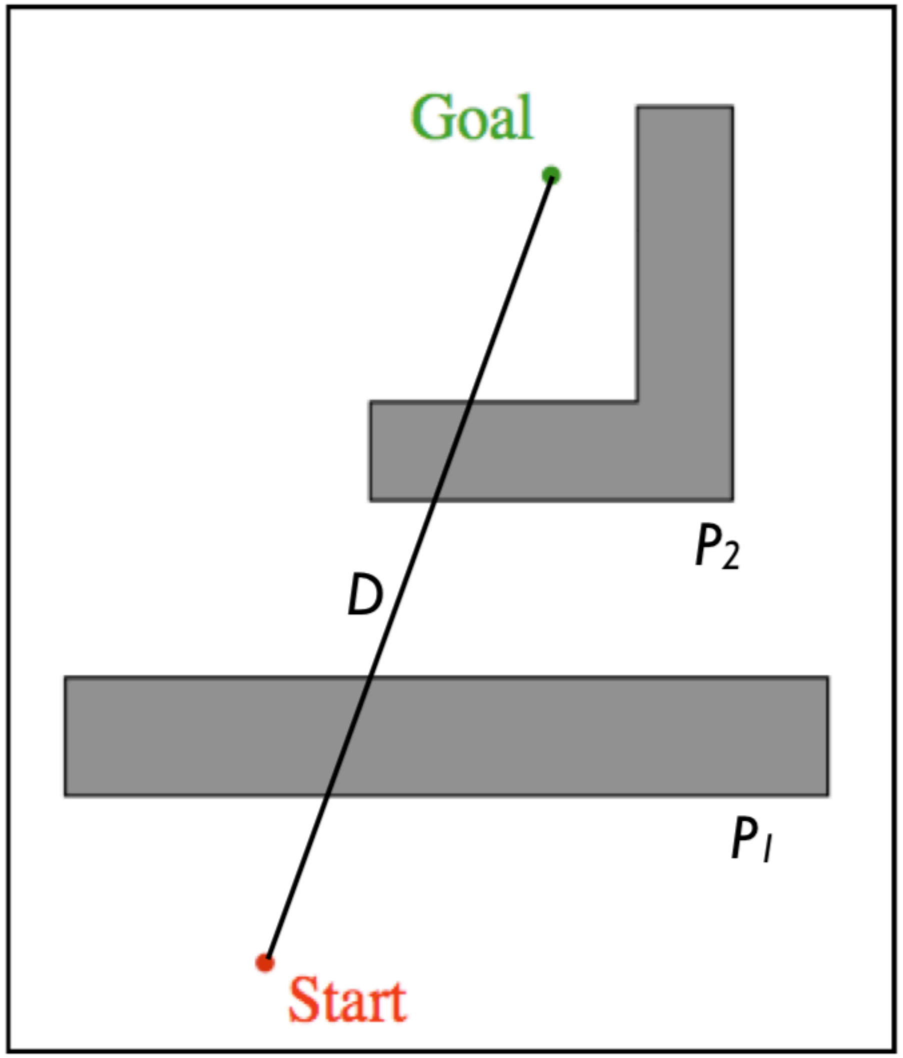
d : *linear distance*

p_i : *obstacle perimeter*

1. **Head towards goal**
2. **When hit, circumnavigate obstacle**
   - **Choose leave point close to goal**
3. **Repeat step 1**

Slides adopted from Robot Motion Planning lecture: http://robvis01.informatik.tu-muenchen.de/courses/motion/slides/Set2-PathPlanning.pdf

Bug 1



q_s : *Start*

q_d : *Goal*

q_i^H : *Hit*

q_i^L : *Leave*

$$L_{Bug1} \le d(q_{start}, q) + 1.5 \cdot \sum_{i=1}^{n} p_i$$

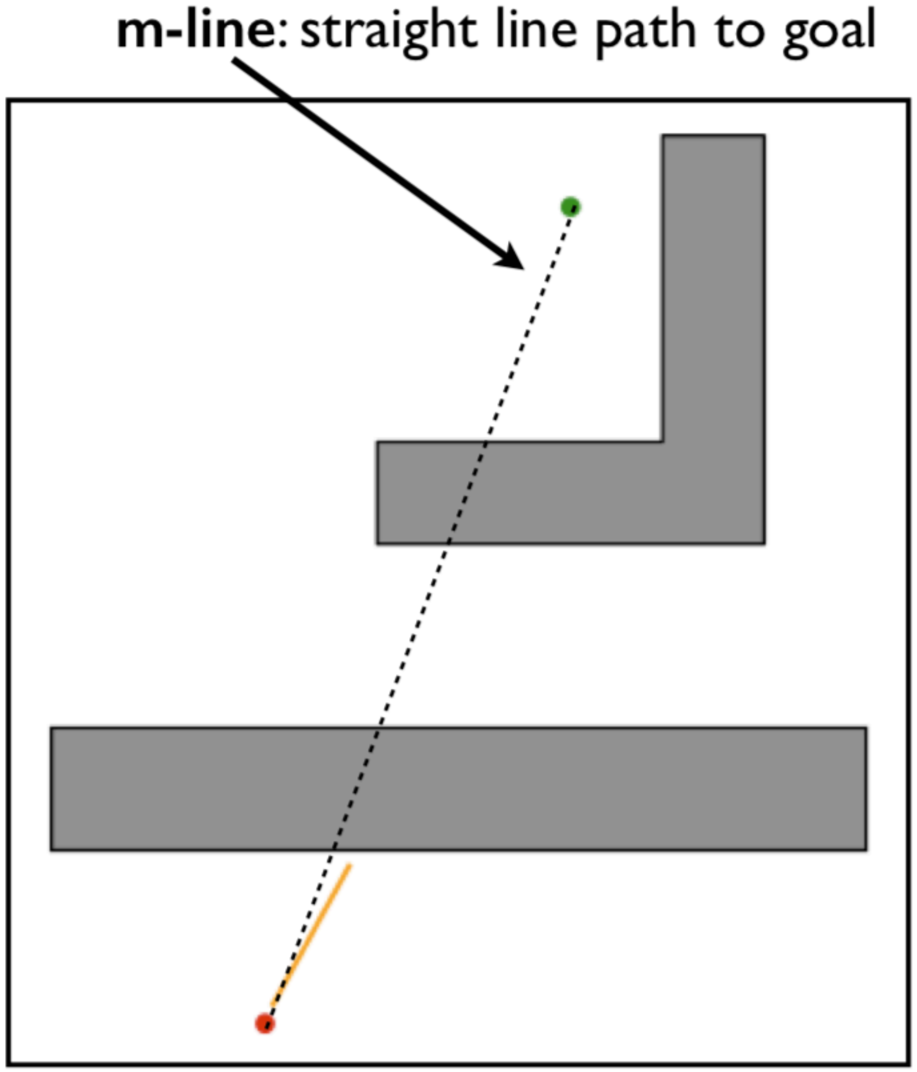
d : *linear distance*

p_i : *obstacle perimeter*

1. **Head towards goal**
2. **When hit, circumnavigate obstacle**
   - **Choose leave point close to goal**
3. **Repeat step 1**

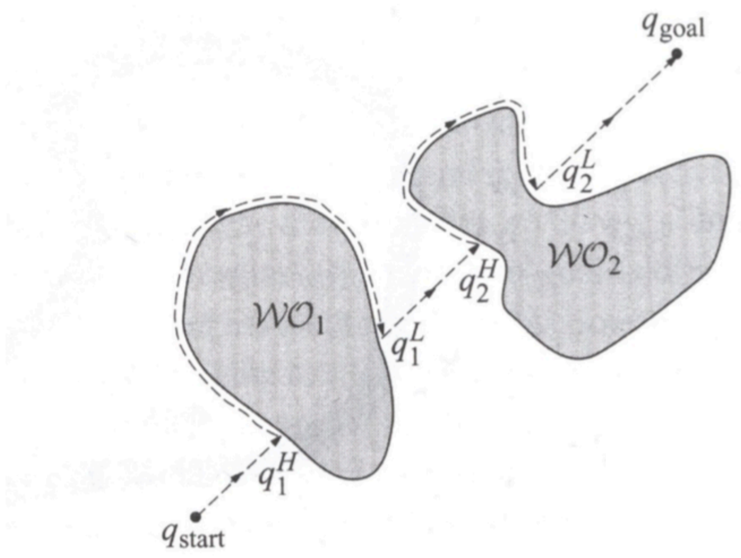
Slides adopted from Robot Motion Planning lecture: http://robvis01.informatik.tu-muenchen.de/courses/motion/slides/Set2-PathPlanning.pdf

Bug 2



m-line: straight line path to goal

1. Head towards goal <span>on m-line</span>
2. When hit, traverse obstacle until m-line
3. Set leave point and exit obstacle
4. Repeat step 1



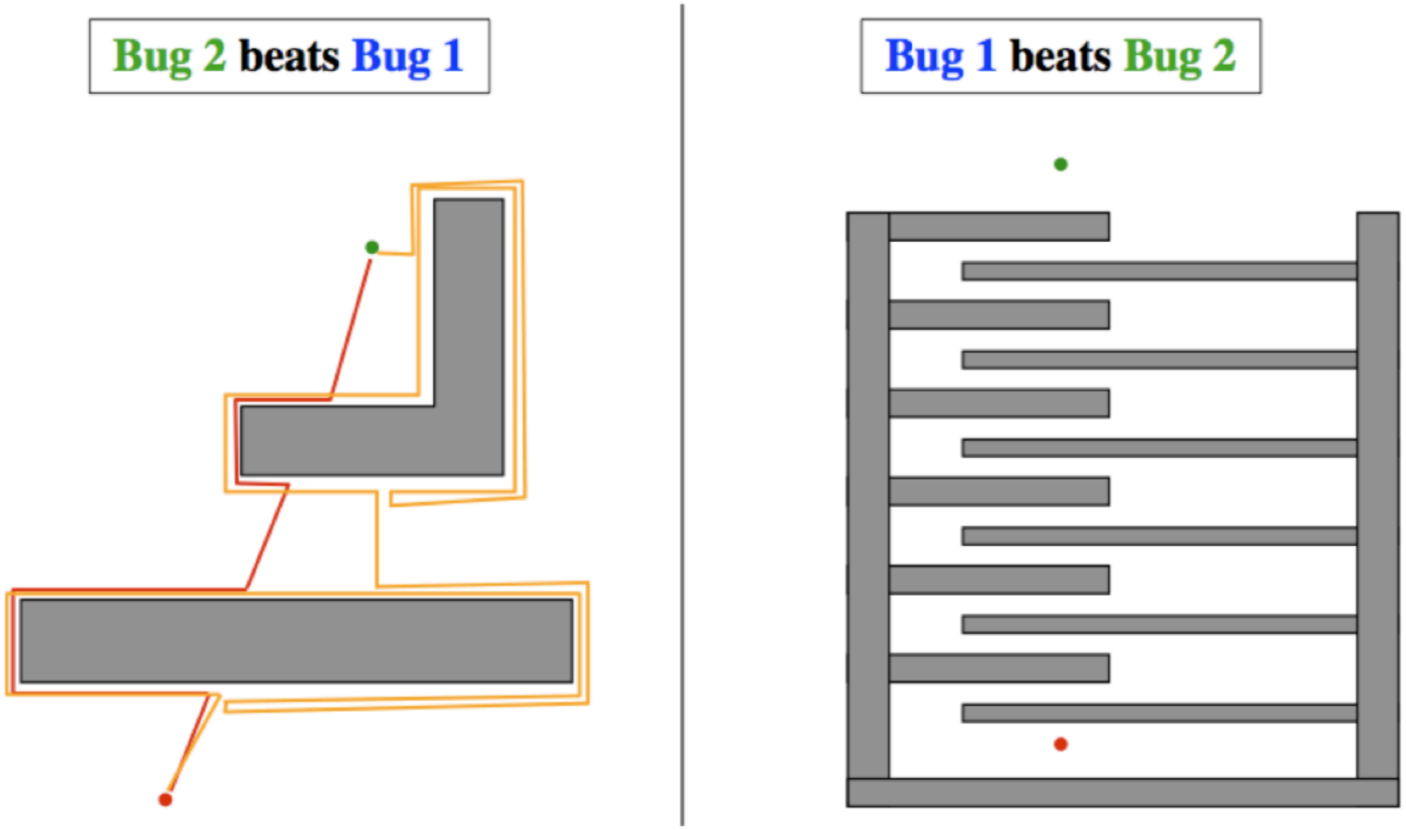
$$L_{Bug\,2} \leq d(q_{start}, q) + 0.5 \cdot \sum_{i=1}^{n} n_i p_i$$

d : *linear distance*

p_i : *obstacle perimeter*

Slides adopted from Robot Motion Planning lecture: http://robvis01.informatik.tu-muenchen.de/courses/motion/slides/Set2-PathPlanning.pdf

Bug 1 vs Bug 2



Slides adopted from Robot Motion Planning lecture: http://robvis01.informatik.tu-muenchen.de/courses/motion/slides/Set2-PathPlanning.pdf