Robótica Móvel e Inteligente

Path Planning and Obstacle Avoidance

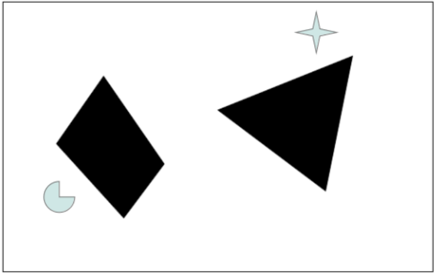
Artur Pereira <artur@ua.pt>

DETI / Universidade de Aveiro

Sumário

Navigation
Definition

- "Given partial knowledge about its environment and a goal position, navigation encompasses the ability of the robot to act, based on its knowledge and sensor values so as to reach its goal positions as efficiently and as reliable as possible" [Siegwart]
- In mobile robotics the knowledge about the environment and situation is usually only partially and uncertain



Navigation
Questions and topics

- Where am I?
  - localization
- Where have I been?
  - mapping
- Where should I going?
  - decision
- **What's the best way to get there?**
  - **Path planning**
- How do I get there?
  - Path following and obstacle avoidance (Motion)

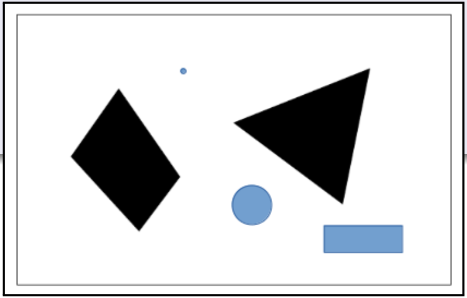
Path planning
Definition

- Path planning – the task of computing a path for the robot such that it can reach the desired goal without colliding with known obstacles
- Optimal paths can be hard to compute, specially for robots that can not move in arbitrary directions (i.e. non-holonomic robots)

- Path and trajectory are often used with the same meaning
  - But one can distinguish them
- Path – only geometric considerations
  - a way from a start position/pose/configuration to a goal one
- Trajectory – includes geometric and time considerations
  - the dynamics is also considered
- Another term is motion, applied to mobile robots or manipulators
  - considers other constraints, like collision avoidance (of dynamics obstacles)

Path planning
Some concepts

- Configuration space (C-Space) – set of possible valid configurations (poses) of the robot
  - Defines the search space and the set of allowable paths
- Free space (F-Space) – set of valid configurations that do not intercept obstacles in the environment
  - Depends on the robot shape
- Cases to be considered:
  - Point robot
  - Symmetric robot 
  - Non-symmetric robot 



Path planning
Some assumptions

- Assumption 1:
  - Often, path planning methods assume that the robot is symmetric and holonomic, and treat it simply as a point
  - If the robot is treated as a point, the obstacles must be “inflated” in order to compensate the robot radius
  - This approach greatly simplifies path planning
- Assumption 2:
  - there exists a good enough representation (map) of the environment that can be used to compute a path
- Assumption 3:
  - there exists a good enough estimatation of the robot’s pose

Path planning
Environment representation

- There are different ways to represent the environment
- Two common ones are topological maps and metric maps
  - metric maps may represent a continuous environment

- Path-planning algorithms often are only applicable to discrete maps
  - Transform the possibly continuous environmental map model into a discrete map suitable for the chosen path-planning algorithm
  - Use the generated discrete map to perform the path-to-goal search

Path planning
Example of metric and topological maps

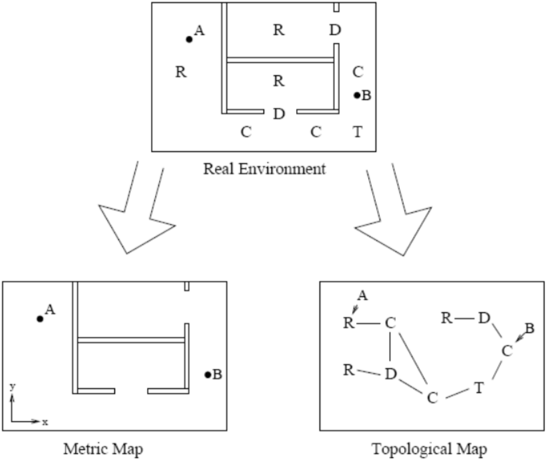
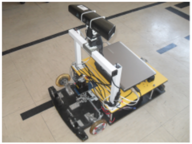
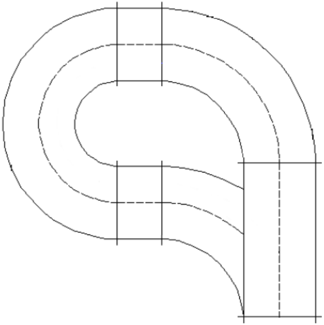
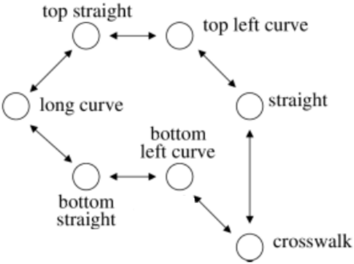


Figure from Meyer, “Map-based navigation in mobile robotics”. 2003

Path planning
Another example of metric and topological maps



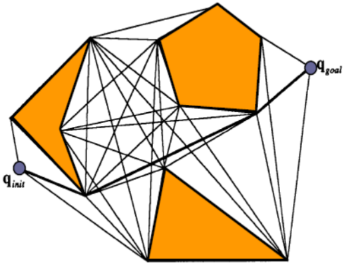




Path planning approaches
Classical approaches

- Potential field methods – Define a potential function over the free space that has a global minimum at the goal and follow the steepest descent of the potential function
- Roadmap techniques – Represent the connectivity of the free space by a network of 1-D curves
- Cell decomposition algorithms – Decompose the free space into cells and represent the connectivity of the free space by the connectivity (adjacency) graph of these cells
- Rapidly Exploring Random Tree – randomly builds a space-filling tree, constructed incrementally from samples drawn from the free space

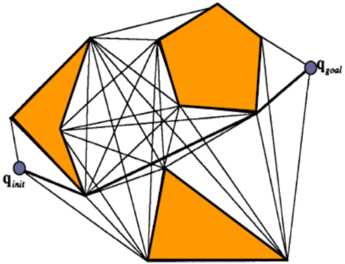
Path planning approaches
Visibility graph



- Obstacles are treated as polygons
  - for every obstacle, a polygon including it can be defined
- A graph is defined where:
  - q_{init}, q_{goal}, and all obstacle vertices are the nodes
  - for every pair of nodes which can be connected by a line segment, not passing through an obstacle, there is an edge
    - such pair of nodes can "see" each other)

- With the detected nodes and edges, a connectivity graph (visibility graph) is then generated
  - every edge can be labelled with some cost function value (for example, the Euclidian distance)

Path planning approaches
Visibility graph (2)

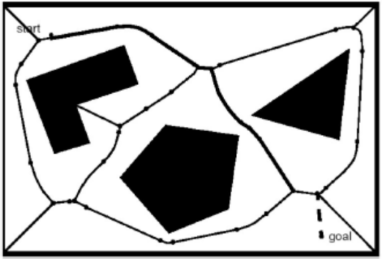


- A graph search algorithm can be used to find the shortest path along the "roads" defined by the visibility graph
- Efficient method for sparse environments

- Problem – may cause the robot to move too close to obstacles
  - little margin for errors in motion
- Solution – grow obstacles even more, to give more clear space between robot and obstacle

Path planning approaches

Voronoi diagrams



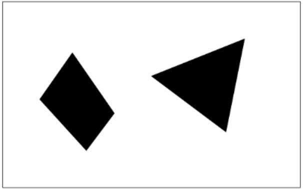
- For each point in free space its distance to the nearest obstacle is computed
- Then, the set of points equidistant from the nearest two or more obstacle boundaries are extracted

- The Voronoi diagram is obtained by linking these points, from where the shortest path can be computed
- The result is a path of maximum distance from obstacles
  - usually far from optimal, in the sense of total path length

Path planning approaches
Cell decomposition

- Divide space into simple, connected regions called cells
  - cells can be either free, occupied or partially occupied
- Discretize the space by constructing an adjacency graph of the free cells
- Adjacency graph
  - Nodes – free cells
  - Edges – there is an edge between every pair of nodes whose corresponding cells are adjacent
- Locate “goal” and “start” cells and search for the shortest path in the adjacency graph that join them
  - Typically paths are assumed to pass through the mid-points of the cells

Path planning approaches
Cell decomposition (2)

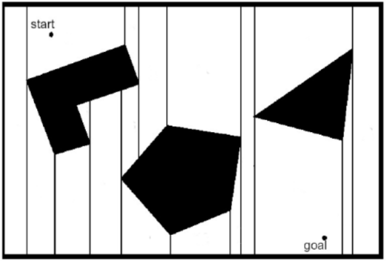


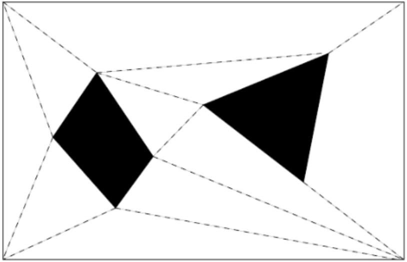
- Given an environment, how to decompose it?

- Two possible cell decomposition methods
  - Exact cell decomposition and approximate cell decomposition
- Exact cell decomposition
  - Free cells correspond exactly to free space
    - There is no partially occupied cells
- Approximate cell decomposition
  - Some free space is included in occupied cells
    - the partially occupied cells are considered as occupied
  - Cells can have fixed-size or variable-size

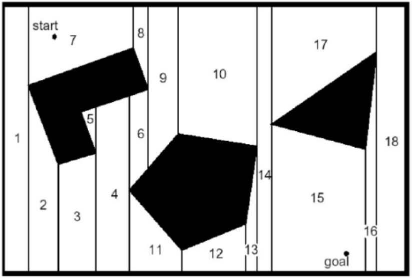
Path planning approaches
Exact cell decomposition

- The free space F is represented by a set of non-overlapping convex cells whose union is exactly F
- Examples of convex shapes: trapezoids, triangles
- The basic abstraction behind this method is that the position of the robot within each cell does not matter, only the ability to travel to another free cell

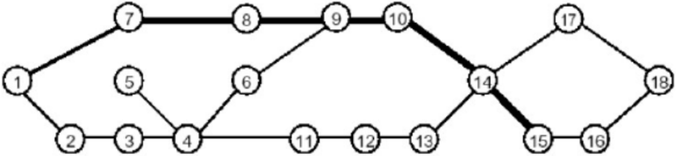




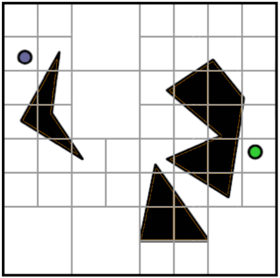
Path planning approaches
Exact cell decomposition (2)



- A connectivity graph can be constructed, where:
  - nodes represent the free cells
  - every adjacent pair of cells is connected by an edge
- Result can be complex if the world is complex
- Good for sparse environments



Path planning approaches
Approximate cell decomposition

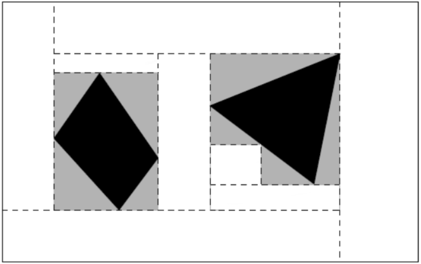


- Free space F is represented by a set of non-overlapping cells whose union is contained in F
- Cells usually have simple, regular shapes, like rectangles, squares, hexagons

- Two approaches:
  - Fixed-size cell decomposition
  - Variable-size cell decomposition

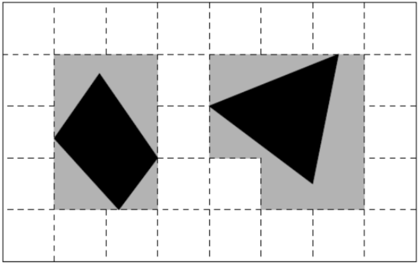
Path planning approaches
Variable-size cell decomposition

- Decomposing using variable-size cells
  - with a rectangular shape
- Grey areas are free space considered as occupied



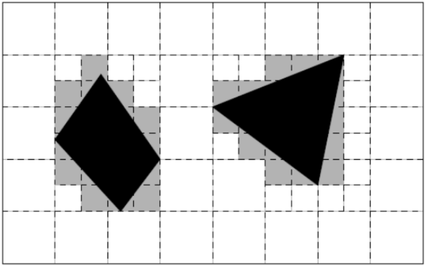
Path planning approaches
Fixed-size cell decomposition

- Decomposing using fixed-size cells
  - with a square shape
- Grey areas are free space considered as occupied
- Cell size is not dependent on the obstacle size in the environment
  - narrow passage ways can be lost
- Low computational complexity of path planning



Path planning approaches
Quad-tree cell decomposition

- Decomposing using variable-size cells
  - with a square shape
- Partially occupied cells are subdivided until a given granularity
- At each level of resolution only the cells whose interiors lie entirely in the free space are used to construct the connectivity graph
- Efficient representation, adapted to the complexity of the environment
  - sparse environments contain fewer cells thus consuming less memory

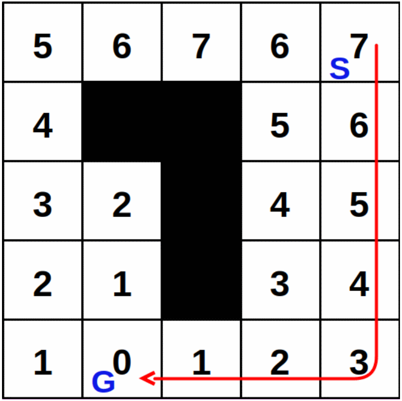


Search algorithms
Methods

- Once a graph is obtained, finding the shortest path between start node and goal node can be done using graph search algorithms
- Many graph search algorithms require visiting each node in the graph to determine the shortest path
  - Computationally tractable for sparsely connected graphs
  - Computationally expensive for highly connected graphs (e.g., regular grid)
- Covered methods:
  - Wavefront expansion
  - Dijkstra's algorithm
  - A* algorithm

Search algorithms
Wavefront expansion

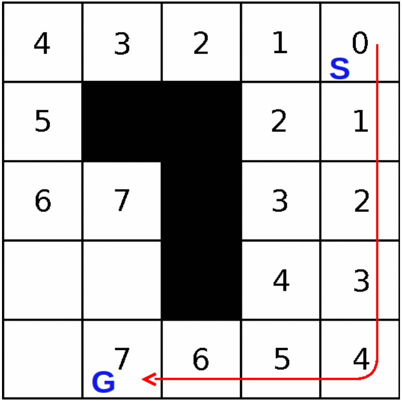
- Wavefront expansion (aka NF1 or grassfire), useful to find paths in fixed-size cell arrays
- Starting at the goal, mark in each adjacent cell its distance to the goal (using Manhattan distance)
- This process continues until the cell corresponding to the start position is reached
- The planner calculate a path to reach the goal by linking together cells that are adjacent and closer to the goal



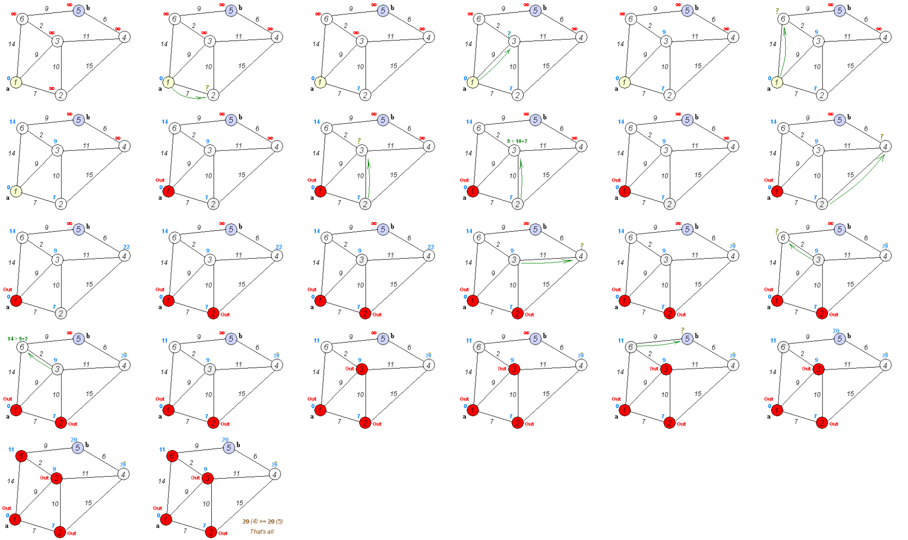
Search algorithms

Dijkstra's algorithm – cell environment

- Beginning at the start node, the algorithm marks all adjacent neighbors with the cost to get there
- It then proceeds to the node with the lowest cost marking all of its adjacent nodes with the lowest cost to reach them
- Once all adjacent neighbors of a node have been marked, the algorithm proceeds to the node with the next lowest cost not visited yet
- Once the algorithm visits the goal node, it terminates
- The path to goal may be obtained starting from the goal node and following the edges pointing towards the lowest node cost



Search algorithms
Dijkstra's algorithm – graph example

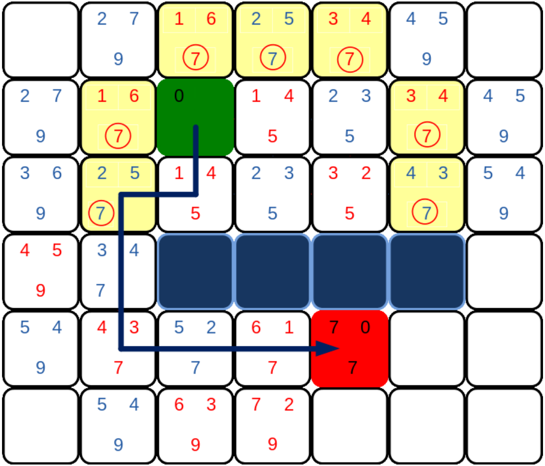


Taken from `https://en.wikipedia.org/wiki/Dijkstra's_algorithm`

Search algorithms
A* search algorithm

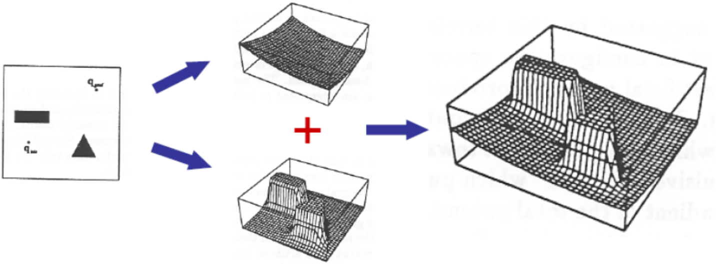
- The idea is to use a cost function to rank the choices, choose the best one first, and try it
- Cost function:
  - f*(n) = g(n) + h*(n) – ('*' means they are estimates)
  - where:
    - g(n) is the cost of going from the start to node n
    - h*(n) is an estimated cost of going from node n to the goal
- h* is a "heuristic function" – (a way of guessing the cost of going from node n to goal
  - the robot can't "see" the path between node n and the goal
  - h*(n) should never be greater than h(n): h*(n) $\leq$ h(n)
  - Must always underestimate remaining cost to reach goal
  - The Euclidian (straight line) distance is often a good choice

Search algorithms
A* search algorithm (2)



Potential field path planning
Main idea

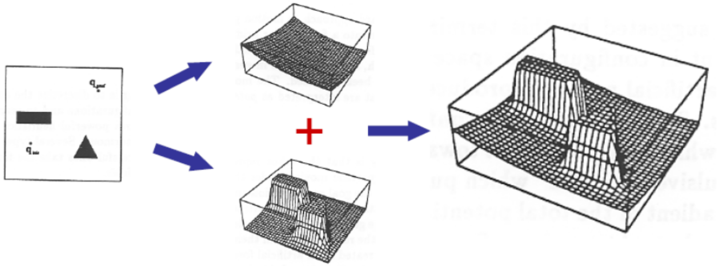
- Think the robot as a particle in a potential field
- Define a potential function over the free space that has a global minimum at the goal
- Define high potentials for the obstacles
- The robot follow the steepest descent of the potential function



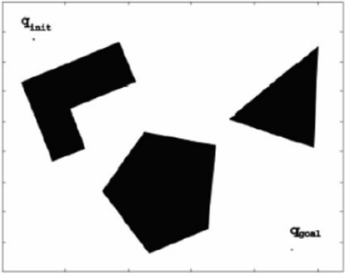
Potential field path planning

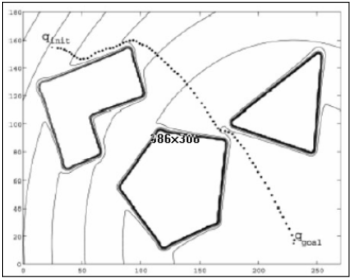
Rolling down the hill

- The goal location generates an attractive potential, pulling the robot towards the goal
- The obstacles generate a repulsive potential, pushing the robot far away from the obstacles
- The negative gradient of the total potential is treated as an artificial force applied to the robot
- Generated robot movement is similar to a ball rolling down the hill

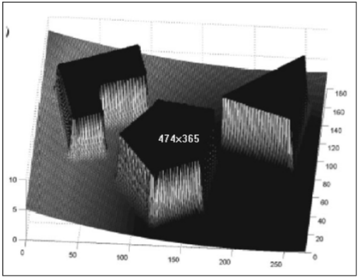


Potential field path planning
Problems





- Often leads to oscillating motion
- Trapped in local minima in the potential field
- Parameter tuning problems



Navigation
Questions and topics

- Where am I?
  - localization
- Where have I been?
  - mapping
- Where should I going?
  - decision
- What's the best way to get there?
  - Path planning
- **How do I get there?**
  - Path following and **obstacle avoidance** (Motion)

Obstacle avoidance
Purpose

- In general, the environment is not fully modeled
  - Some obstacles (chairs, for example) may not be represented
- The environment might also change dynamically
  - for example, people moving around
- In such cases, to navigate, the robot may need to modify the planned path
  - reacting, re-planning
- Obstacle avoidance relies on
  - Information about the goal (position, plan)
  - current localization (on the map)
  - Recent sensory information (a local map)
- The purpose of the obstacle avoidance algorithms is to avoid collisions with obstacles, while pursuing the goal/plan

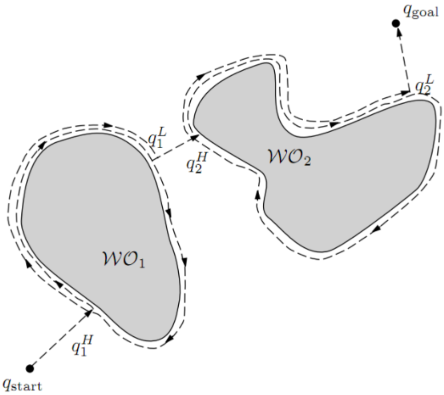
Obstacle avoidance
Methods

- Efficient obstacle avoidance should be optimal with respect to
  - The overall goal
  - The actual speed and kinematics of the robot
  - The on-board sensors
  - The current and future risk of collision

- Covered methods:
  - Bug1
  - Bug2
  - Vector field histogram

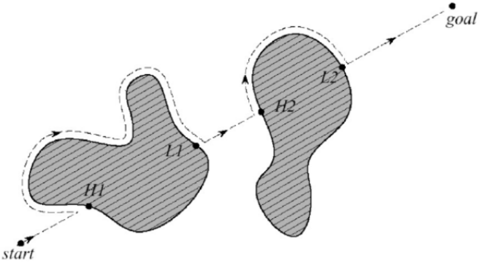
Obstacle avoidance
Bug1 algorithm

- Algorithm:
  - Go in direction to the goal until reach it or hit an obstacle
  - If goal reached, finish
  - Do a full tour around the obstacle, storing the closest point to the goal
  - Go to that point
  - Repeat
- Comment:
  - Inefficient but does the job



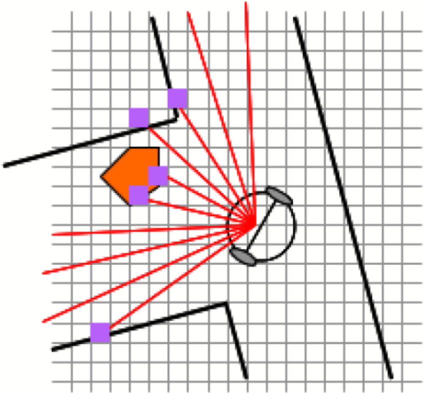
Obstacle avoidance
Bug2 algorithm

- Algorithm:
  - Go in direction to the goal until reach it or hit an obstacle
  - If goal reached, finish
  - Contour the obstacle, from left or right, until reach the line between start and goal
  - Repeat, keeping the same side (left or right) as before
- Comment:
  - More efficient but can fail



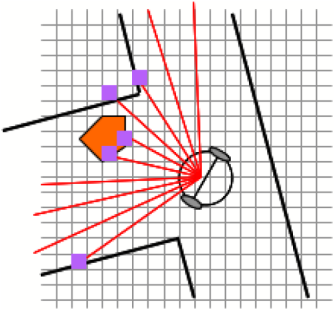
Obstacle avoidance
Vector field histogram (VFH)

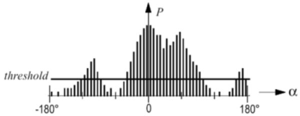
- Creates a local map of the environment around the robot
- Environment is represented as an occupancy grid (2D Cartesian grid called the histogram grid)
- Each grid cell (i, j) value holds the confidence that there is an obstacle at that location
- The grid is updated by relatively recent sensor data



Obstacle avoidance
Vector field histogram (VFH) – 2

- Information in the histogram grid is converted to a simpler representation, called the polar histogram
- The polar histogram retains the statistical information but reduces the amount of data that needs to be handled in real-time
- A threshold transforms the polar histogram in a get binary diagram with passable regions
- A sector with low obstacle density (below threshold) is a candidate to travel away from obstacle
- The one chosen depends on the target point





Bibliography

- “Introduction to Autonomous Mobile Robots”, Second Edition, Roland Siegwart et al., MIT Press, 2011
- “Principles of Robot Motion: Theory, Algorithms, and Implementations”, Howie Choset et al., MIT Press, Boston, 2005
- “Artificial Intelligence: A Modern Approach”, 3rd edition, Russel and Norvig, Pearson, 2009
- “Introduction to Autonomous Mobile Robots”, R. Siegwart, I. Nourbakhsh, D. Scaramuzza
- “The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots”, J. Borenstein and Y. Koren
- “VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots”, I. Ulrich and J. Borenstein