Multi-Goal Path and Motion Planning

Jan Faigl

Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague

Lecture 06

B4M36UIR – Artificial Intelligence in Robotics

Overview of the Lecture

- Part 1 – Improved Sampling-based Motion Planning
  - Selected Sampling-based Motion Planners
- Part 2 – Multi-Goal Planning and Robotic Information Gathering
  - Multi-Goal Planning
  - Robotic Information Gathering
  - Robotic Exploration
  - Inspection Planning
  - Unsupervised Learning for Multi-Goal Planning

Part I

Part 1 – Improved Sampling-based Motion Planning

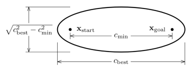
Improved Sampling-based Motion Planners

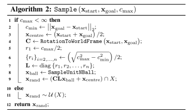
- Although asymptotically optimal sampling-based motion planners such as RRT* or RRG may provide high-quality or even optimal solutions of the complex problem, their performance in simple, e.g., 2D scenarios, is relatively poor
  In a comparison to the ordinary approaches (e.g., visibility graph)
- They are computationally demanding and performance can be improved similarly as for the RRT, e.g.,
  - Goal biasing, supporting sampling in narrow passages, multi-tree growing (Bidirectional RRT)
- The general idea of improvements is based on **informing** the sampling process
- Many modifications of the algorithms exists, selected representative modifications are
  - **Informed RRT***
  - Batch Informed Trees (**BIT***)
  - Regionally Accelerated BIT* (**RABIT***)

Informed RRT*

- Focused RRT* search to increase the convergence rate
- Use Euclidean distance as an admissible heuristic
- Ellipsoidal informed subset – the current best solution c_{best}

$$X_{\hat{f}} = \{\mathbf{x} \in X \,|\, ||\mathbf{x}_{start} - \mathbf{x}||_2 + ||\mathbf{x} - \mathbf{x}_{goal}||_2 \leq c_{best}\}$$





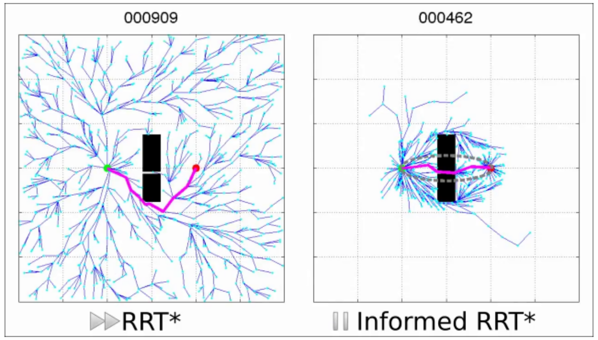
- Directly Based on the RRT*
- Having a feasible solution
- Sampling inside the ellipse

Algorithm 1: Informed RRT*($\mathbf{x}_{start}, \mathbf{x}_{goal}$)

```
V ← {x_start};
E ← ∅;
X_soln ← ∅;
T = (V, E);
for iteration = 1 ... N do
    c_best ← min_{x_soln ∈ X_soln} {Cost (x_soln)};
    x_rand ← Sample (x_start, x_goal, c_best);
    x_nearest ← Nearest (T, x_rand);
    x_new ← Steer (x_nearest, x_rand);
    if CollisionFree (x_nearest, x_new) then
        V ← ∪ {x_new};
        X_near ← Near (T, x_new, r_RRT*);
        x_min ← x_nearest;
        c_min ← Cost (x_min) + c · Line (x_nearest, x_new);
        for ∀x_near ∈ X_near do
            c_new ← Cost (x_near) + c · Line (x_near, x_new);
            if c_new < c_min then
                if CollisionFree (x_near, x_new) then
                    x_min ← x_near;
                    c_min ← c_new;
        E ← E ∪ {(x_min, x_new)};
        for ∀x_near ∈ X_near do
            c_near ← Cost (x_near);
            c_new ← Cost (x_new) + c · Line (x_new, x_near);
            if c_new < c_near then
                if CollisionFree (x_new, x_near) then
                    x_parent ← Parent (x_near);
                    E ← E \ {(x_parent, x_near)};
                    E ← E ∪ {(x_new, x_near)};
        if InGoalRegion (x_new) then
            X_soln ← X_soln ∪ {x_new};
return T;
```

Gammell, J. B., Srinivasa, S. S., Barfoot, T. D. (2014): Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. IROS.

Informed RRT* – Demo



https://www.youtube.com/watch?v=d7dX5MvDYTc

Gammell, J. B., Srinivasa, S. S., Barfoot, T. D. (2014): Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. IROS.

Batch Informed Trees (BIT*)

- Combining RGG (Random Geometric Graph) with the heuristic in incremental graph search technique, e.g., Lifelong Planning A* (LPA*)
  - The properties of the RGG are used in the RRG and RRT*
- Batches of samples – a new batch starts with denser implicit RGG
- The search tree is updated using LPA* like incremental search to reuse existing information

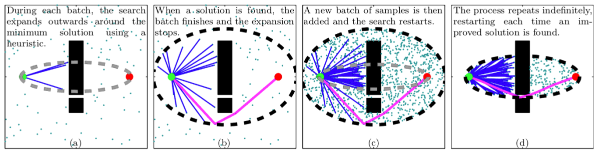
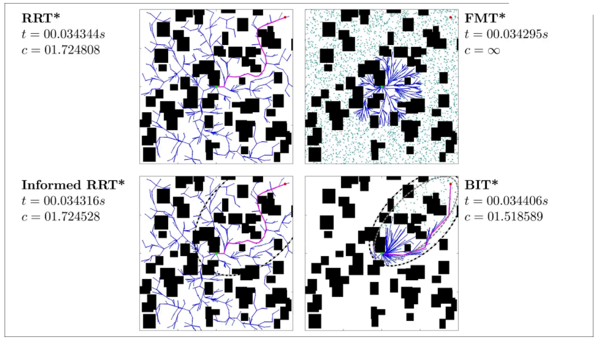


Fig. 3. An illustration of the informed search procedure used by BIT*. The start and goal states are shown as green and red, respectively. The current solution is highlighted in magenta. The subproblem that contains any better solutions is shown as a black dashed line, while the progress of the current batch is shown as a grey dashed line. Fig. (a) shows the growing search of the first batch of samples, and (b) shows the first search ending when a solution is found. After pruning and adding a second batch of samples, Fig. (c) shows the search restarting on a denser graph while (d) shows the second search ending when an improved solution is found. An animated illustration is available in the attached video.

Gammell, J. B., Srinivasa, S. S., Barfoot, T. D. (2015): Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. ICRA.

Batch Informed Trees (BIT*) – Demo



https://www.youtube.com/watch?v=TQIoCC48gp4

Gammell, J. B., Srinivasa, S. S., Barfoot, T. D. (2015): Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. ICRA.

Regionally Accelerated BIT* (RABIT*)

- Use local optimizer with the BIT* to improve the convergence speed
- Local search Covariant Hamiltonian Optimization for Motion Planning (CHOMP) is utilized to connect edges in the search graphs using local information about the obstacles

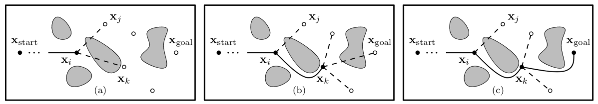
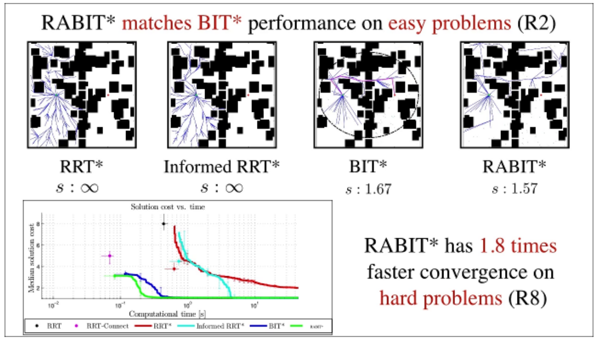


Fig. 2. An illustration of how the RABIT* algorithm uses a local optimizer to exploit obstacle information and improve a global search. The global search is performed, as in BIT*, by incrementally processing an edge queue (dashed lines) into a tree (a). Using heuristics, the potential edge from $\mathbf{x}_i$ to $\mathbf{x}_k$ is processed first as it could provide a better solution than an edge from $\mathbf{x}_i$ to $\mathbf{x}_j$. The initial straight-line edge is given to a local optimizer which uses information about obstacles to find a local optima between the specified states (b). If this edge is collision free, it is added to the tree and its potential outgoing edges are added to the queue. The next-best edge in the queue is then processed in the same fashion, using the local optimizer to once again propose a better edge than a straight-line (c).

Choudhury, S., Gammell, J. D., Barfoot, T. D., Srinivasa, S. S., Scherer, S. (2016): Regionally Accelerated Batch Informed Trees (RABIT*): A Framework to Integrate Local Information into Optimal Path Planning. ICRA.

Regionally Accelerated BIT* (RABIT*) – Demo



https://www.youtube.com/watch?v=mgq-DW36jSo

Choudhury, S., Gammell, J. D., Barfoot, T. D., Srinivasa, S. S., Scherer, S. (2016): Regionally Accelerated Batch Informed Trees (RABIT*): A Framework to Integrate Local Information into Optimal Path Planning. ICRA.

Overview of Improved Algorithm

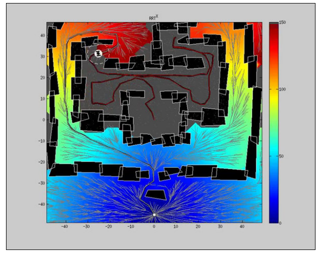
- Optimal path/motion planning is an active research field

Approaches	Constraints	Planning Mode	Kinematic Model	Sampling Strategy	Metric
1. RRT* [7]	Holonomic	Offline	Point	Uniform	Euclidean
2. Anytime RRT* [4]	Non-holonomic	Online	Dubin Car	Uniform	Euclidean + Velocity
3. B-RRT* [58]	Holonomic	Offline	Rigid Body	Local bias	Goal biased
4. RRT*FN [33]	Holonomic	Offline	Robotic Arm	Uniform	Cumulative Euclidean
5. RRT*-Smart [35]	Holonomic	Offline	Point	Intelligent	Euclidean
6. Optimal B-RRT* [36]	Holonomic	Offline	Point	Uniform	Euclidean
7. RRT# [50]	Holonomic	Offline	Point	Uniform	Euclidean
8. Adapted RRT* [64], [49]	Non-holonomic	Offline	Car-like and UAV	Uniform	A* Heuristic
9. SRRT* [44]	Non-holonomic	Offline	UAV	Uniform	Geometric + dynamic constraint
10. Informed RRT* [34]	Holonomic	Offline	Point	Direct Sampling	Euclidean
11. IB-RRT* [37]	Holonomic	Offline	Point	Intelligent	Greedy + Euclidean
12. DT-RRT [39]	Non-holonomic	Offline	Car-like	Hybrid	Angular + Euclidean
13. RRT*i [3]	Non-holonomic	Online	UAV	Local Sampling	A* Heuristic
14. RTR+CS* [43]	Non-holonomic	Offline	Car-like	Uniform + Local Planning	Angular + Euclidean
15. Mitsubishi RRT* [2]	Non-holonomic	Online	Autonomous Car	Two-stage sampling	Weighted Euclidean
16. CARRT* [65]	Non-holonomic	Online	Humanoid	Uniform	MW Energy Cost
17. PRRT* [48]	Non-holonomic	Offline	P3-DX	Uniform	Euclidean

Noreen, I., Khan, A., Habib, Z. (2016): Optimal path planning using RRT* based approaches: a survey and future directions. IJACSA.

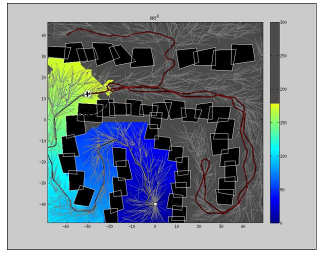
Motion Planning for Dynamic Environments – RRT^X

- Refinement and repair of the search graph during the navigation (quick rewiring of the shortest path)



RRT^X – Robot in 2D

https://www.youtube.com/watch?v=S9pguCPUo3M



RRT^X – Robot in 2D

https://www.youtube.com/watch?v=KxFivNgTV4o

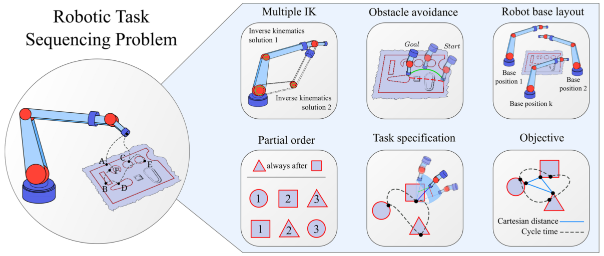
Otte, M., & Frazzoli, E. (2016). RRT^X: Asymptotically optimal single-query sampling-based motion planning with quick replanning. The International Journal of Robotics Research, 35(7), 797–822.

Part II

Part 2 – Multi-Goal Planning and Robotic Information Gathering

Multi-Goal Planning

- Having a set of locations to be visited, determine the cost-efficient path to visit them and return to a starting location.
  - Locations where a robotic arm or mobile robot performs some task
- The problem is called **robotic task sequencing problem** within the context of robotic manipulators

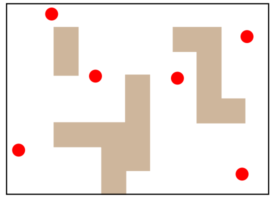


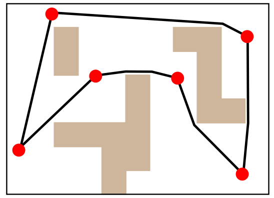
Alatartsev, S., Stellmacher, S., Ortmeier, F. (2015): Robotic Task Sequencing Problem: A Survey. Journal of Intelligent & Robotic Systems.

- It is also called **Multi-Goal Path Planning** (MTP) problem

Multi-Goal Path Planning (MTP)

- Multi-goal path planning problem is a problem to determine how to visit the given set of locations
- It consists of point-to-point planning problems how to reach one location from another
- The main "added" challenge to the path planning is a determination of the optimal sequence of the visits to the locations (with respect to the cost-efficient solution to visit all the given locations)





- Determining the sequence of visits is a **combinatorial optimization problem** that can be formulated as the **Traveling Salesman Problem**

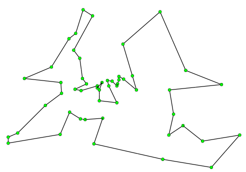
Traveling Salesman Problem (TSP)

> Given a set of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city.

- The TSP can be formulated for a graph $\boldsymbol{G}(V, E)$, where V denotes a set of locations (cities) and E represents edges connecting two cities with the associated travel cost c (distance), i.e., for each $v_i, v_j \in V$ there is an edge $e_{ij} \in E$, $e_{ij} = (v_i, v_j)$ with the cost c_{ij}.
- If the associated cost of the edge (v_i, v_j) is the Euclidean distance $c_{ij} = |(v_i, v_j)|$, the problem is called the **Euclidean TSP** (ETSP).
  In our case, $v \in V$ represents a point in $\mathbb{R}^2$ and solution of the ETSP is a path in the plane.
- It is known, the TSP is NP-hard (its decision variant) and several algorithms can be found in literature.
  William J. Cook (2012) – In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation

Existing Approaches to the TSP

- Efficient heuristics from the Operational Research have been proposed
- LKH – K. Helsgaun efficient implementation of the Lin-Kernighan heuristic (1998)
  http://www.akira.ruc.dk/~keld/research/LKH/
- Concorde – Solver with several heuristics and also optimal solver
  http://www.math.uwaterloo.ca/tsp/concorde.html



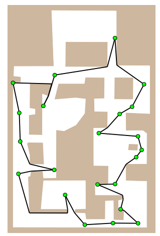
Problem Berlin52 from the TSPLIB

Beside the heuristic and approximations algorithms (such as Christofides 3/2-approximation algorithm), other („soft-computing") approaches have been proposed, e.g., based on genetic algorithms, and memetic approaches, ant colony optimization (ACO), and neural networks.

Multi-Goal Path Planning (MTP) Problem

> Given a map of the environment $\mathcal{W}$, mobile robot $\mathcal{R}$, and a set of locations, what is the shortest possible collision free path that visits each location exactly once and returns to the origin location.

- **MTP problem** is a **robotic variant of the TSP** with the edge costs as the length of the *shortest* path connecting the locations
- For n locations, we need to compute up to n^2 shortest paths (solve n^2 motion planning problems)
- The paths can be found as the shortest path in a graph (roadmap), from which the $G(V, E)$ for the TSP can be constructed



Visibility graph as the roadmap for a point robot provides a straight forward solution, but such a shortest path may not be necessarily feasible for more complex robots

Multi-Goal Motion Planning

- In the previous cases, we consider existing roadmap or relatively "simple" collision free (shortest) paths in the polygonal domain
- However, determination of the collision-free path in high dimensional configuration space ($\mathcal{C}$-space) can be a challenging problem itself
- Therefore, we can generalize the MTP to multi-goal **motion** planning (MGMP) considering motion planners using the notion of $\mathcal{C}$-space for avoiding collisions.

- An example of MGMP can be

Plan a cost efficient trajectory for hexapod walking robot to visit a set of target locations.

Problem Statement – MGMP Problem

- The working environment $\mathcal{W} \subset \mathbb{R}^3$ is represented as a set of obstacles $\mathcal{O} \subset \mathcal{W}$ and the robot configuration space $\mathcal{C}$ describes all possible configurations of the robot in $\mathcal{W}$
- For $q \in \mathcal{C}$, the robot body $\mathcal{A}(q)$ at q is collision free if $\mathcal{A}(q) \cap \mathcal{O} = \emptyset$ and all collision free configurations are denoted as $\mathcal{C}_{free}$
- Set of n goal locations is $\mathcal{G} = (g_1, \ldots, g_n)$, $g_i \in \mathcal{C}_{free}$
- Collision free path from q_{start} to q_{goal} is $\kappa : [0, 1] \to \mathcal{C}_{free}$ with $\kappa(0) = q_{start}$ and $d(\kappa(1), q_{end}) < \epsilon$, for an admissible distance ϵ
- **Multi–goal path** τ is **admissible** if $\tau : [0, 1] \to \mathcal{C}_{free}$, $\tau(0) = \tau(1)$ and there are n points such that $0 \leq t_1 \leq t_2 \leq \ldots \leq t_n$, $d(\tau(t_i), v_i) < \epsilon$, and $\bigcup_{1<i\leq n} v_i = \mathcal{G}$

- **The problem is to find the path τ^* for a cost function c such that $c(\tau^*) = \min\{c(\tau) \mid \tau$ *is admissible multi–goal path*$\}$**

MGMP – Existing Approches

- Determination of all paths connecting any two locations $g_i, g_j \in \mathcal{G}$ is usually very computationally demanding
- Several approaches can be found in literature, e.g.,
  - Considering Euclidean distance as an approximation in the solution of the TSP as the Minimum Spanning Tree (MST) – Edges in the MST are iteratively refined using optimal motion planner until all edges represent a feasible solution
    Saha, M., Roughgarden, T., Latombe, J.-C., Sánchez-Ante, G. (2006): Planning Tours of Robotic Arms among Partitioned Goals. IJRR.
  - **Synergistic Combination of Layers of Planning (SyCLoP)** – A combination of route and trajectory planning
    Plaku, E., Kavraki, L.E., Vardi, M.Y. (2010): Motion Planning With Dynamics by a Synergistic Combination of Layers of Planning. T-RO.
  - Steering RRG roadmap expansion by unsupervised learning for the TSP

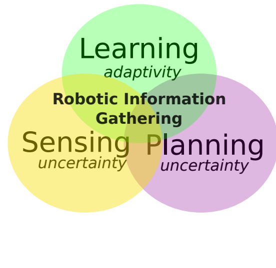
Faigl (2016), WSOM

Robotic Information Gathering

Create a model of phenomena by autonomous mobile robots performing measurements in a dynamic unknown environment.

Challenges in Robotic Information Gathering

- **Where to take new measurements?**
  To improve the phenomena model
- **What locations visit first?**
  On–line decision–making
- **How to efficiently utilize more robots?**
  To divide the task between the robots
- **How to navigate robots to the selected locations?**
  Improve Localization vs Model



Robotic Information Gathering and Multi-Goal Planning

- **Robotic information gathering** aims to determine an optimal solution to collect **the most relevant data** (measurements) in a **cost-efficient way**.
  - It builds on a simple path and trajectory planning – *point-to-point planning*
  - It may consist of determining locations to be visited and a combinatorial optimization problem to determine the sequence to visit the locations
- It can be considered as a general problem for various tasks and missions which may include **online decision-making**
  - Informative path/motion planning and persistent monitoring
  - *Robotic exploration* – create a map of the environment as quickly as possible

  and **determining a plan** according to the particular **assumptions and constraints**; a plan that is then executed by the robots
  - **Inspection planning** - Find a shortest tour to inspect the given environment
  - **Surveillance planning** - Find the shortest (a cost efficient) tour to periodically monitor/capture the given objects/regions of interest
  - **Data collection planning** – Determine a cost efficient path to collect data from the sensor stations (locations)
- In both cases, **multi-goal path planning** allows solving (or improving the performance) of the particular missions

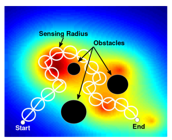
Informative Motion Planning

- Robotic information gathering can be considered as the **informative motion planning** problem to a determine trajectory $\mathcal{P}^*$ such that

$$\mathcal{P}^* = \text{argmax}_{\mathcal{P}\in\Psi} I(\mathcal{P}), \text{ such that } c(\mathcal{P}) \leq B, \text{ where}$$

  - Ψ is the space of all possible robot trajectories,
  - $I(\mathcal{P})$ is the information gathered along the trajectory $\mathcal{P}$
  - $c(\mathcal{P})$ is the cost of $\mathcal{P}$ and B is the allowed budget

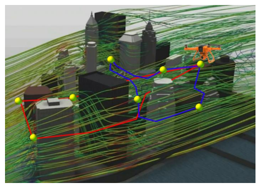
- Searching the space of all possible trajectories is complex and demanding problem
- A discretized problem can be solved by combinatorial optimization techniques
  Usually scale poorly with the size of the problem
- A trajectory is from a continuous domain
- **Sampling-based motion planning techniques** can be employed for finding maximally informative trajectories



Hollinger, G., Sukhatme, G. (2014): Sampling-based robotic information gathering algorithms. IJRR.

Persistent Monitoring of Spatiotemporal Phenomena

- Persistent environment monitoring is an example of the robotic information gathering mission
- It stands to determine suitable locations to collect data about the studied phenomenon
- Determine cost efficent path to visit the locations, e.g., considering limited travel budget
  Orienteering Problem
- Collect data and update the phenomenon model
- Search for the next locations and path to further improve model

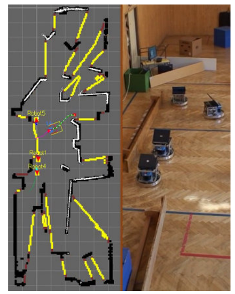


- **Robotic information gathering** combines several challenges
  - Determining locations to be visited regarding the particular mission objective
    Optimal sampling design
  - Finding optimal paths/trajectories
    Trajectory planning – Path/motion planning
  - Determining the optimal sequence of visits to the locations
    Multi-goal path/motion planning
- Moreover, solutions have to respect particular constraints
  - Kinematic and kinodynamic constraints of the vehicle, collision-free paths, limited travel budget

In general, the problem is very challenging, and therefore, we consider the most imporant and relevant constraints, i.e., we address the problem under particular assumptions.

Robotic Exploration of Unknown Environment

- Robotic exploration is a fundamental problem of robotic information gathering
- The problem is:
  How to efficiently utilize a group of mobile robots to autonomously create a map of an unknown environment
  - Performance indicators vs constraints
    Time, energy, map quality vs robots, communication
  - Performance in a real mission depends on the on-line **decision-making**
  - It includes challlenges such a
    - Map building and localization
    - Determination of the navigational waypoints
      Where to go next?
    - Path planning and navigation to the waypoints
    - Coordination of the actions (multi-robot team)



Courtesy of M. Kulich

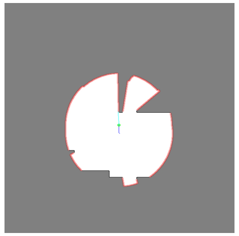
Mobile Robot Exploration

- Create a map of the environment
- **Frontier**-based approach

  Yamauchi (1997)
- Occupancy grid map

  Moravec and Elfes (1985)
- Laser scanner sensor
- Next-best-view approach

  Select the next robot goal



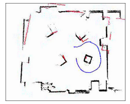
Performance metric:

Time to create a map of the whole environment

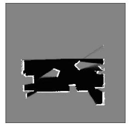
search and rescue mission

Environment Representation – Mapping and Occupancy Grid

- The robot uses its sensors to build a map of the environment
- The robot should be localized to integrate new sensor measurements into a globally consistent map
- **SLAM** – Simultaneous Localization and Mapping
  - The robot uses the map being built to localize itself
  - The map is primarily to help to localize the robot
  - The map is a "side product" of SLAM

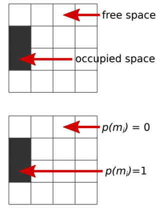


- **Grid map** – discretized world representation
  - A cell is **occupied** (an obstacle) or **free**
- **Occupancy grid map**
  - Each cell is a binary random variable modeling the occupancy of the cell



Occupancy Grid

- **Assumptions**
  - The area of a cell is either completely free or occupied
  - Cells (random variables) are independent of each other
  - The state is **static**
- A cell is a binary random variable modeling the occupancy of the cell
  - Cell m_i is occupied $p(m_i) = 1$
  - Cell m_i is not occupied $p(m_i) = 0$
  - **Unknown** $p(m_i) = 0.5$



- Probability distribution of the map m

$$p(m) = \Pi_i p(m_i)$$

- Estimation of map from sensor data $z_{1:t}$ and robot poses $x_{1:t}$

$$p(m|z_{1:t}, x_{1:t}) = \Pi_i p(m_i|z_{1:t}, x_{1:t})$$

Binary Bayes filter – Bayes rule and Markov process assumption

Binary Bayes Filter

- Sensor data $z_{1:t}$ and robot poses $x_{1:t}$
- Binary random variables are indepedent and states are static

$$p(m_i|z_{1:t}, x_{1:t}) \overset{\text{Bayes rule}}{=} \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \overset{\text{Markov}}{=} \frac{p(z_t|m_i, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})}$$

$$p(z_t|m_i, x_t) = \frac{p(m_i, z_t, x_t)p(z_t, x_t)}{p(m_i|x_t)}$$

$$p(m_i, z_{1:t}, x_{1:t}) \overset{\text{Bayes rule}}{=} \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i|x_t)p(z_t|z_{1:t-1}, x_{1:t})} \overset{\text{Markov}}{=} \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1}, x_{1:t})}$$

- Probability a cell is occupied

$$p(m_i|z_{\mathbf{1}:t}, x_{\mathbf{1}:t}) = \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{\mathbf{1}:t-\mathbf{1}}, x_{\mathbf{1}:t-\mathbf{1}})}{p(m_i)p(z_t|z_{\mathbf{1}:t-\mathbf{1}}, x_{\mathbf{1}:t})}$$

- Probability a cell is not occupied

$$p(\neg m_i|z_{\mathbf{1}:t}, x_{\mathbf{1}:t}) = \frac{p(\neg m_i|z_t, x_t)p(z_t|x_t)p(\neg m_i|z_{\mathbf{1}:t-\mathbf{1}}, x_{\mathbf{1}:t-\mathbf{1}})}{p(\neg m_i)p(z_t|z_{\mathbf{1}:t-\mathbf{1}}, x_{\mathbf{1}:t})}$$

- Ratio of the probabilities

$$\frac{p(m_i|z_{\mathbf{1}:t}, x_{\mathbf{1}:t})}{p(\neg m_i|z_{\mathbf{1}:t}, x_{\mathbf{1}:t})} = \frac{p(m_i|z_t, x_t)p(m_i|z_{\mathbf{1}:t-\mathbf{1}}, x_{\mathbf{1}:t-\mathbf{1}})p(\neg m_i)}{p(\neg m_i|z_t, x_t)p(\neg m_i|z_{\mathbf{1}:t-\mathbf{1}}, x_{\mathbf{1}:t-\mathbf{1}})p(m_i)}$$

$$= \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} \frac{p(m_i, z_{\mathbf{1}:t-\mathbf{1}}, x_{\mathbf{1}:t-\mathbf{1}})}{1 - p(m_i|z_{\mathbf{1}:t-\mathbf{1}}, x_{\mathbf{1}:t-\mathbf{1}})} \frac{1 - p(m_i)}{p(m_i)}$$

sensor model z_t, recursive term, prior

- Log odds ratio is defined as $l(x) = \log \frac{p(x)}{1-p(x)}$
- and the probability $p(x)$ is $p(x) = 1 - \frac{1}{1-e^{l(x)}}$
- The product modeling the cell m_i based on $z_{1:t}$ and $x_{1:t}$

$$l(m_i|z_{1:t}, x_{1:t}) = \underbrace{l(m_i|z_t, x_t)}_{\text{inverse sensor model}} + \underbrace{l(m_i, |z_{1:t-1}, x_{1:t-1})}_{\text{recursive term}} - \underbrace{l(m_i)}_{\text{prior}}$$

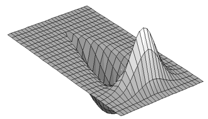
Occupancy Mapping Algorithm

Algorithm 1: OccupancyGridMapping($\{l_{t-1,i}\}, x_t, z_t$)

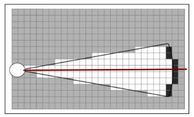
foreach *m_i of the map m* **do**
- **if** *m_i in the perceptual field of z_t* **then**
  - $l_{t,i} := l_{t-1,i} + \text{inv\_sensor\_model}(m_i, x_t, z_t) - l_0$;
- **else**
  - $l_{t,i} := l_{t-1,i}$;

return $\{l_{t,i}\}$

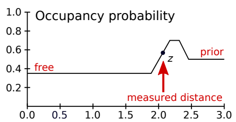
- Occupancy grid mapping developed by Moravec and Elfes in mid 80'ies for noisy sonars



Inverse sensor model for sonars range sensors



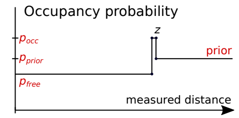
Field of view of the sonar range sensor



Occupancy value depending on the measured distance

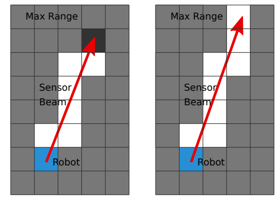
Laser Sensor Model

- The model is "sharp" with a precise detection of the obstacle
- For the range measurement d_i, update the grid cells along a sensor beam



Algorithm 2: Update map for $\mathcal{L} = (d_1, \ldots, d_n)$

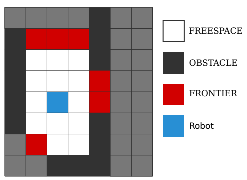
foreach $d_i \in \mathcal{L}$ **do**
- **foreach** *cell m_i raycasted towards* $\min(d_i, range)$ **do**
  - $p := grid(m_i)p_{free}$;
  - $grid(m_i) := p/(2p - p_{free} - grid(m_i) + 1)$;
- $m_d :=$ cell at d_i;
- **if** *obstacle detected at m_d* **then**
  - $p := grid(m_d)p_{occ}$;
  - $grid(m_i) := p/(2p - p_{occ} - grid(m_i) + 1)$
- **else**
  - $p := grid(m_d)p_{free}$;
  - $grid(m_i) := p/(2p - p_{free} - grid(m_i) + 1)$



Frontier-based Exploration

- The basic idea of the **frontier** based exploration is navigation of the mobile robot towards unknown regions

  Yamauchi (1997)
- **Frontier** – a border of the known and unknown regions of the environment
- Based on the probability of individual cells in the occupancy grid, cells are classified into:
  - `FREESPACE` – $p(m_i) < 0.5$
  - `OBSTACLE` – $p(m_i) > 0.5$
  - `UNKNOWN` – $p(m_i) = 0.5$
- **Frontier cell** is a `FREESPACE` cell that is incident with an `UNKNOWN` cell
- Frontier cells as the navigation waypoints have to be reachable, e.g., after obstacle growing



Use grid-based path planning

Frontier-based Exploration Strategy

Algorithm 3: Frontier-based Exploration

map := init(*robot*, *scan*);

while *there are some reachable frontiers* **do**
- Update occupancy *map* using new sensor data and Bayes rule;
- $\mathcal{M}$:= Created grid map from *map* using thresholding;
- $\mathcal{M}$:= Grow obstacle according to the dimension of the robot;
- $\mathcal{F}$:= Determnine frontier cells from $\mathcal{M}$;
- $\mathcal{F}$:= Filter out unreachable frontiers from $\mathcal{F}$;
- f := Select the closest frontier from $\mathcal{F}$, e.g. using shortest path;
- *path* := Plan a path from the current robot position to f;
- Navigate robot towards f along *path* (for a while);

Improvements of the basic Frontier-based Exploration

Several improvements have been proposed in the literature

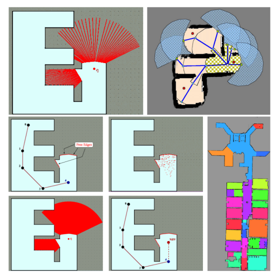
- Introducing utility as computation of expected covered area from a frontier

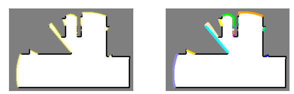
  González-Baños, Latombe (2002)
- Map segmentation for identification of rooms and exploration of the whole room by a single robot

  Holz, Basilico, Amigoni, Behnke (2010)
- Consider longer planning horizon (as a solution of the Traveling Salesman Problem (TSP))

  Zlot, Stentz (2006), Kulich, Faigl (2011, 2012)
- Representatives of free edges

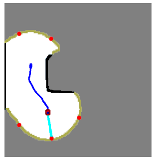
  Faigl, Kulich (2015)



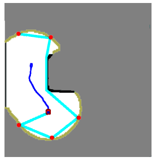


Variants of the Distance Cost

- **Simple robot-goal distance** – *next-best view*



  - Evaluate all goals using the robot–goal distance
    A length of the path from the robot position to the goal candidate.
  - Greedy goal selection – the closest one
  - Using frontier representatives improves the performance a bit
- **TSP distance cost** – *Non-myopic next-best view*



  - Consider visitations of all goals
    Solve the associated traveling salesman problem (TSP)
  - A length of the tour visiting all goals
  - Use frontier representatives
  - the TSP distance cost improves performance about 10-30% without any further heuristics, e.g., expected coverage (utility)

Kulich, M., Faigl, J, Přeučil, L. (2011): On Distance Utility in the Exploration Task. ICRA.

Multi-Robot Exploration – Overview

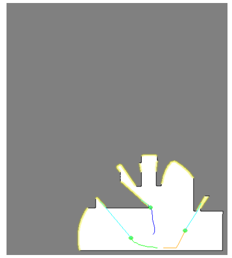
- We need to assign navigation waypoint to each robot, which can be formulated as the **task-allocation problem**
- Exploration can be considered as an **iterative procedure**

1. Initialize the occupancy grid *Occ*
2. $\mathcal{M} \leftarrow$ create_navigation_grid(*Occ*)
   cells of $\mathcal{M}$ have values {freespace, obstacle, unknown}
3. $\boldsymbol{F} \leftarrow$ detect_frontiers($\mathcal{M}$)
4. Goal candidates $\boldsymbol{G} \leftarrow$ generate($\boldsymbol{F}$)
5. **Assign next goals to each robot** $r \in \boldsymbol{R}$,
   $(\langle r_1, g_{r_1}\rangle, \ldots, \langle r_m, g_{r_m}\rangle) = \text{assign}(\boldsymbol{R}, \boldsymbol{G}, \mathcal{M})$
6. Create a plan $\boldsymbol{P}_i$ for each pair $\langle r_i, g_{r_i}\rangle$
   consisting of simple operations
7. Perform each plan up to s_{max} operations
   At each step, update Occ using new sensor measurements
8. If $|G| == 0$ exploration finished, otherwise go to Step 2

- There are several parts of the exploration procedure where important decisions are made regarding the exploration performance, e.g.
  - How to determined goal candidates from the the frontiers?
  - How to plan a paths and assign the goals to the robots?
  - How to navigate the robots towards the goal?
  - When to replan?
  - etc.

Exploration Procedure – Decision-Making Parts

1. Initialize – set plans for m robots, $\mathcal{P} = (P_1, \ldots, P_m)$, $P_i = \emptyset$.
2. `Repeat`
   2.1 **Navigate robots** using the plans $\mathcal{P}$;
   2.2 Collect new measurements;
   2.3 Update the navigation map $\mathcal{M}$;
   `Until` **replanning condition is met**.



3. **Determine goal candidates $\boldsymbol{G}$** from $\mathcal{M}$.
4. `If` $|\boldsymbol{G}| > 0$ **assign goals to the robots**
   - $(\langle r_1, g_{r_1}\rangle, \ldots, \langle r_m, g_{r_m}\rangle)$=assign($\boldsymbol{R}, \boldsymbol{G}, \mathcal{M}$), $r_i \in \boldsymbol{R}, g_{r_i} \in \boldsymbol{G}$;
   - **Plan paths** to the assigned goals
     $\mathcal{P} = \text{plan}(\langle r_1, g_{r_1}\rangle, \ldots, \langle r_m, g_{r_m}\rangle, \mathcal{M})$;
   - Go to Step 2.
5. Stop all robots or navigate them to the depot

All reachable parts of the environment are explored.

Goal Assignment Strategies – Task Allocation Algorithms

Exploration strategy can be formulated as the **task-allocation problem**

$$(\langle r_1, g_{r_1}\rangle, \ldots, \langle r_m, g_{r_m}\rangle) = \text{assign}(\boldsymbol{R}, \boldsymbol{G}(t), \mathcal{M}),$$

where $\mathcal{M}$ is the current map

1. **Greedy Assignment**
   - Randomized greedy selection of the closest goal candidate
     Yamauchi B, Robotics and Autonomous Systems 29, 1999
2. **Iterative Assignment**
   - Centralized variant of the broadcast of local eligibility algorithm (BLE)
     Werger B, Mataric M, Distributed Autonomous Robotic Systems 4, 2001
3. **Hungarian Assignment**
   - Optimal solution of the task-allocation problem for assignment of n goals and m robots in $O(n^3)$
     Stachniss C, C implementation of the Hungarian method, 2004
4. **Multiple Traveling Salesman Problem – MTSP Assignment**
   - $\langle$cluster–first, route–second$\rangle$, the TSP distance cost
     Faigl et al. 2012

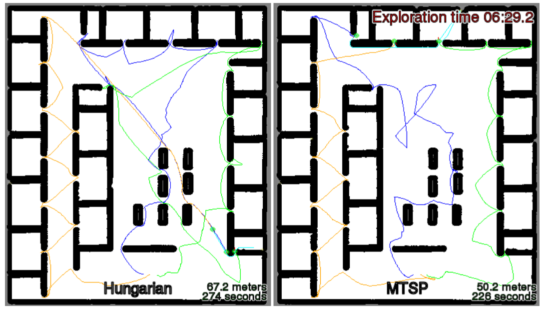
MTSP-based Task-Allocation Approach

- Consider the task-allocation problem as the **Multiple Traveling Salesman Problem (MTSP)**
- MTSP heuristic *⟨cluster–first, route–second⟩*
  1. Cluster the goal candidates $\boldsymbol{G}$ to m clusters
     $$\boldsymbol{C} = \{C_1, \ldots, C_m\}, C_i \subseteq \boldsymbol{G}$$
     using K-means
  2. For each robot $r_i \in \boldsymbol{R}, i \in \{1, \ldots m\}$ select the next goal g_i from C_i using **the TSP distance cost**
     Kulich et at., ICRA (2011)
     - Solve the TSP on the set $C_i \cup \{r_i\}$
       the tour starts at r_i
     - The next robot goal g_i is the first goal of the found TSP tour

Faigl, J., Kulich, M., Přeučil, L. (2012): Goal Assignment using Distance Cost in Multi-Robot Exploration. IROS.

Performance of the MTSP vs Hungarian Algorithm

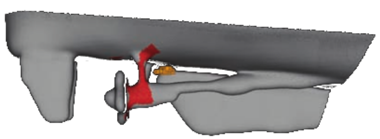
- Replanning as quickly as possible; $m = 3, \rho = 3$ m

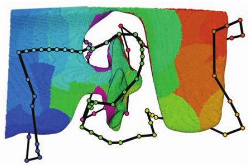


The MTSP assignment provides better performance

Gathering Information in Inspection of Vessel's Propeller

- The planning problem is to determine a shortest inspection path for Autonomous Underwater Vehicle (AUV) to inspect a propeller of the vessel.





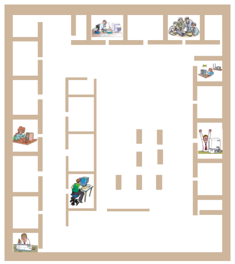
https://www.youtube.com/watch?v=8azP_9VnMtM

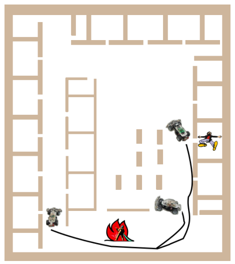
Englot, B., Hover, F.S. (2013): Three-dimensional coverage planning for an underwater inspection robot. Robotics and Autonomous Systems.

Inspection Planning

Motivations (examples)

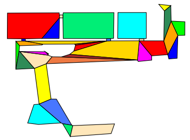
- Periodically visit particular locations of the environment to check, e.g., for intruders, and return to the starting locations
- Based on available plans, provide a guideline how to search a building to find possible victims as quickly as possible (search and rescue scenario)



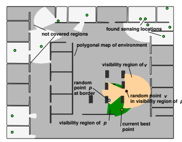


Inspection Planning – Decoupled Approach

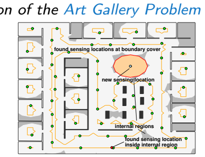
1. Determine sensing locations such that the whole environment would be inspected (seen) by visiting them
   A solution of the Art Gallery Problem



Convex Partitioning (Kazazakis and Argyros, 2002)



Randomized Dual Sampling (González-Baños et al., 1998)



Boundary Placement (Faigl et al., 2006)

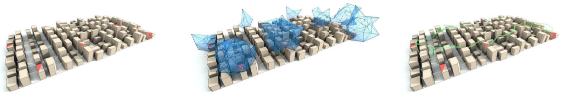
*The problem is related to the **sensor placement** or **sampling design***

2. Create a roadmap connecting the sensing location
   E.g., using visibility graph or randomized sampling based approaches
3. Find the inspection path visiting all the sensing locations as a solution of the multi-goal path planning (a solution of the robotic TSP)
   *Inspection planning can also be called as **coverage path planning** in the literature*

Galceran, E., Carreras, M. (2013): A survey on coverage path planning for robotics. Robotics and Autonomous Systems.

Planning to Capture Areas of Interest using UAV

- Determine a cost-efficient path from which a given set of target regions is covered
- For each target region a subspace $S \subset \mathbb{R}^3$ from which the target can be covered is determined
  S represents the neighbourhood
- We search for the best sequence of visits to the regions
  Combinatorial optimization
- The PRM is utilized to construct the planning roadmap (a graph)
- The problem is formulated as **the Traveling Salesman Problem with Neighborhoods**, as it is not necessary to visit exactly a single location to capture the area of interest

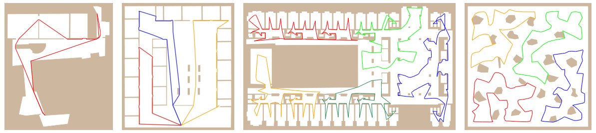


Janoušek and Faigl, (2013) ICRA

Inspection Planning – "*Continuous Sensing*"

- If we do not prescribe a discrete set of sensing locations, we can formulate the problem as the **Watchman route problem**

Given a map of the environment $\mathcal{W}$ determine the shortest, closed, and collision-free path, from which the whole environment is covered by an omnidirectional sensor with the radius ρ



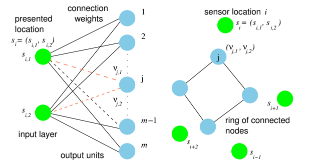
Faigl, J. (2010): Approximate Solution of the Multiple Watchman Routes Problem with Restricted Visibility Range. IEEE Transactions on Neural Networks.

Unsupervised Learning based Solution of the TSP

Kohonen's type of **unsupervised** two-layered neural network (**Self-Organizing Map**)

- Neurons' weights represent nodes $\mathcal{N} = \{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_m\}$ in a plane
- Nodes are organized into a ring
- Sensing locations $\boldsymbol{S} = \{\mathbf{s}_1, \ldots \mathbf{s}_n\}$ are presented to the network in a random order
- Nodes compete to be winner according to their distance to the presented goal $\mathbf{s}$
  $$\boldsymbol{\nu}^* = \operatorname{argmin}_{\boldsymbol{\nu} \in \mathcal{N}} |\mathcal{D}(\{\boldsymbol{\nu}, \mathbf{s}\})|$$
- The winner and its neighbouring nodes are adapted (moved) towards the city according to the neighbouring function
  $$\boldsymbol{\nu}' \leftarrow \mu f(\sigma, d)(\boldsymbol{\nu} - \mathbf{s})$$

$$f(\sigma, d) = \begin{cases} e^{-\frac{d^2}{\sigma^2}} & \text{for } d < m/n_f, \\ 0 & \text{otherwise,} \end{cases}$$



- Best matching unit $\boldsymbol{\nu}$ to the presented prototype $\mathbf{s}$ is determined according to the distance function $|\mathcal{D}(\boldsymbol{\nu}, \mathbf{s})|$
- For the Euclidean TSP, $\mathcal{D}$ is the Euclidean distance
- However, for problems with obstacles, the multi-goal path planning, $\mathcal{D}$ should correspond to the length of the shortest, collision free path

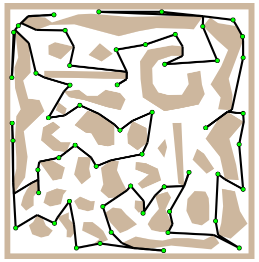
Fort, J.C. (1988), Angéniol, B. et al. (1988), Somhom, S. et al. (1997), etc.

Unsupervised Learning for the Multi-Goal Path Planning

Unsupervised learning procedure

Algorithm 4: SOM-based MTP solver

$\mathcal{N} \leftarrow$ initialization$(\nu_1, \ldots, \nu_m)$;
repeat
  error $\leftarrow 0$;
  foreach $g \in \Pi(\boldsymbol{S})$ **do**
    $\nu^* \leftarrow$ **selectWinner** $\operatorname{argmin}_{\nu \in \mathcal{N}} |S(g, \nu)|$;
    adapt$(S(g, \nu), \mu f(\sigma, l)|S(g, \nu)|)$;
    error $\leftarrow \max\{error, |S(g, \nu^*)|\}$;
  $\sigma \leftarrow (1 - \alpha)\sigma$;
until *error* $\leq \delta$;



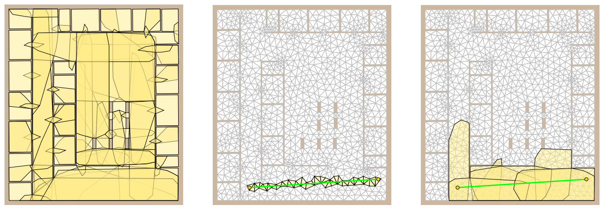
- For multi-goal path planning – the **selectWinner** and **adapt** procedures are based on the solution of the **path planning problem**

Faigl, J. et al. (2011): An Application of Self-Organizing Map in the non-Euclidean Traveling Salesman Problem. Neurocomputing.

SOM for the TSP in the Watchman Route Problem

During the unsupervised learning, we can compute coverage of $\mathcal{W}$ from the current ring (solution represented by the neurons) and adapt the network towards uncovered parts of $\mathcal{W}$

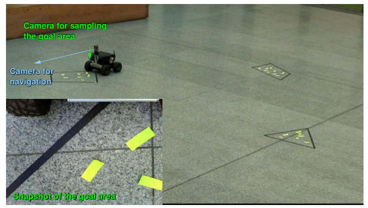
- Convex cover set of $\mathcal{W}$ created on top of a triangular mesh
- Incident convex polygons with a straight line segment are found by walking in a triangular mesh technique



Faigl, J. (2010), TNN

Multi-Goal Path Planning with Goal Regions

- It may be sufficient to visit a goal region instead of the particular point location
  E.g., to take a sample measurement at each goal



Not only a sequence of goals visit has to be determined, but also an appropriate sensing location for each goal need to be found

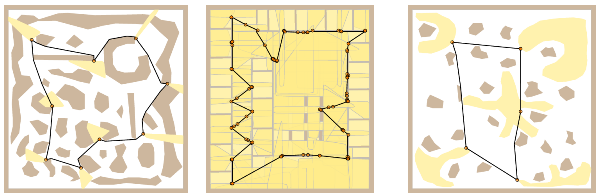
The problem with goal regions can be considered as a variant of the **Traveling Salesman Problem with Neighborhoods (TSPN)**

Traveling Salesman Problem with Neighborhoods

Given a set of n regions (neighbourhoods), what is the shortest closed path that visits each region.

- The problem is NP-hard and APX-hard, it cannot be approximated to within factor $2 - \epsilon$, where $\epsilon > 0$
  Safra and Schwartz (2006) – Computational Complexity
- Approximate algorithms exist for particular problem variants
  E.g., Disjoint unit disk neighborhoods
- Flexibility of the unsupervised learning for the TSP allows generalizing the unsupervised learning procedure to address the TSPN
- **TSPN provides a suitable problem formulation for planning various inspection and data collection missions**

SOM-based Solution of the Traveling Salesman Problem with Neighborhoods (TSPN)

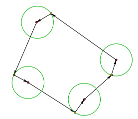
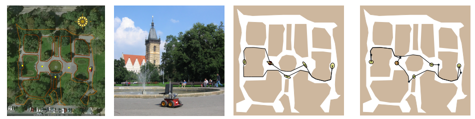


Polygonal Goals	Convex Cover Set	Non-Convex Goals
n=9, T= 0.32 s	n=106, T=5.1 s	n=5, T=0.1 s

Faigl, J. et al. (2013): Visiting Convex Regions in a Polygonal Map. Robotics and Autonomous Systems.

Example – TSPN for Planning with Localization Uncertainty

- Selection of waypoints from the neighborhood of each location
- P3AT ground mobile robot in an outdoor environment

TSP: L=184 m, E_{avg}=0.57 m   TSPN: L=202 m, E_{avg}=0.35 m

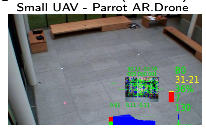
Real overall error at the goals decreased from 0.89 m → 0.58 m (about 35%)

- Decrease localization error at the target locations (indoor)

Small UGV - MMP5

Error decreased from 16.6 cm → 12.8 cm

Small UAV - Parrot AR.Drone



Improved success of the locations' visits 83%→95%

Faigl et al., (2012) ICRA

Summary of the Lecture

Summary

- Improved randomized sampling-based methods
  - Informed sampling – Informed RRT*
  - Improving by batches of samples and reusing previous searches using Lifelong Planning A* (LPA*)
  - Improving local search strategy to improve convergence speed
  - Planning in dynamic environments
- Multi-goal planning and robotic information gathering
  - Multi-goal path planning (MTP) and multi-goal motion planning (MGMP) problems are **robotic variants of the TSP**
  - Existing TSP solvers can be used, by further challenges of robotic systems have to be addressed
  - TSP-like solutions can improve performance in the online decision-making by considering longer planning horizon (non-myopic approaches), e.g., in robotic exploration
  - Inspection planning can be formulated as a robotic variant of the TSP
  - **TSP with Neighborhoods** (**TSPN**) is a benefitial problem formulation to save unnecessary travel cost
  - Unsupervised learning can be used as heuristic for various multi-goal path planning problems (TSP and TSPN like)

Topics Discussed

- Improved sampling-based motion planners
- Multi-goal planning and robotic information gathering missions
  - Multi-goal path planning (MTP) and multi-goal motion planning (MGMP)
  - Traveling Salesman Problem (TSP)
  - Robotic information gathering – informative path planning,
  - Robotic exploration and multi-goal path planning
  - Inspection planning
  - Unsupervised learning for multi-goal path planning
  - Traveling Salesman Problem with Neighborhoods (TSPN)
- Next: Data collection planning