Robotic Information Gathering - Exploration of Unknown Environment

Jan Faigl

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

Lecture 04

B4M36UIR – Artificial Intelligence in Robotics

Overview of the Lecture

- Part 1 – Robotic Information Gathering - Robotic Exploration
  - Robotic Information Gathering
  - Robotic Exploration
  - Information Theoretic Approaches
  - Inspection Planning – Multi-Goal Planning

Part I

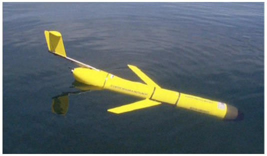
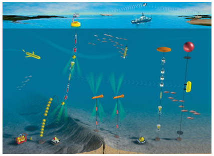
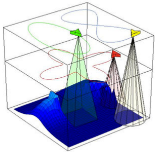
Part 1 – Robotic Exploration

Outline

- Robotic Information Gathering
- Robotic Exploration
- Information Theoretic Approaches
- Inspection Planning – Multi-Goal Planning

Robotic Information Gathering

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

Challenges in Robotic Information Gathering

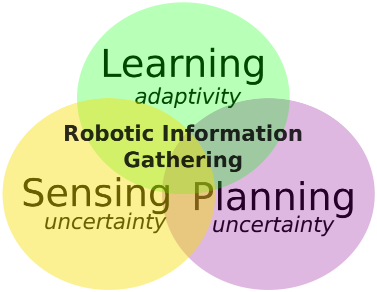
- **Where to take new measurements?**

  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



How to address all these aspects altogether to find a cost efficient solution using in–situ decisions?

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

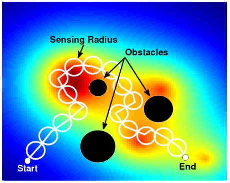
$$\mathcal{P}^* = \text{argmax}_{\mathcal{P}\in\Psi} I(\mathcal{P}), \text{ such that } c(\mathcal{P}) \leq B, \text{ where}$$

  - 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



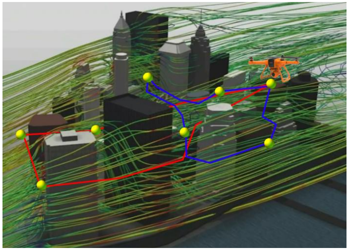
- **Sampling-based motion planning techniques** can be employed for finding maximally informative trajectories

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 efficient 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 important and relevant constraints, i.e., we address the problem under particular assumptions.

Outline

- Robotic Information Gathering
- Robotic Exploration
- Information Theoretic Approaches
- Inspection Planning – Multi-Goal Planning

Robotic Exploration of Unknown Environment

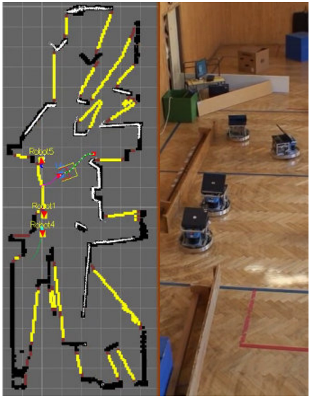
- Robotic exploration is a fundamental problem of robotic information gathering
- The problem is:

  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 challenges such as
    - 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

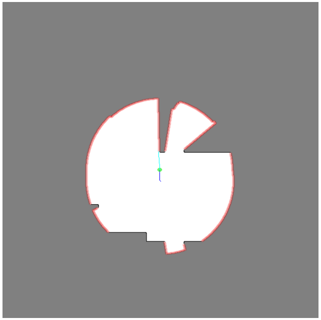
Mobile Robot Exploration

- 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:

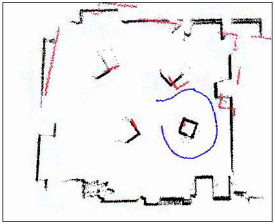
Time to create a map of the whole environment

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

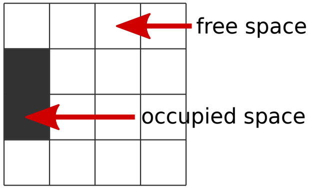
- **Simultaneous Localization and Mapping (SLAM)**
  - The robot uses the map being built to localize itself
  - The map is primarily to help to localize the robot
  - The map is a "side product" of SLAM
- **Grid map** – discretized world representation
  - A cell is **occupied** (an obstacle) or **free**
- **Occupancy grid map**
  - Each cell is a binary random variable modeling the occupancy of the cell



Courtesy of M. Kulich

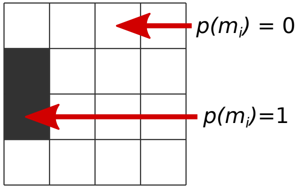
Occupancy Grid

- **Assumptions**
  - The area of a cell is either completely free or occupied
  - Cells (random variables) are independent of each other
  - The state is static



- Probability distribution of the map m
- A cell is a binary random variable modeling the occupancy of the cell, e.g.,
  - 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 the 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 independent 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

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

- Probability a cell is not occupied

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

- Ratio of the probabilities

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

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 $x_{1:t}$

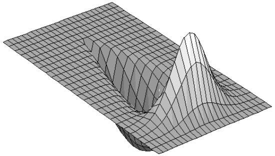
$$l(m_i|z_{1:t}, x_{1:t}) = \underbrace{l(m_i|z_t, x_t)}_{\text{inverse sensor model}} + \underbrace{l(m_i, |z_{1:t-1}, x_{1:t-1})}_{\text{recursive term}} - \underbrace{l(m_i)}_{\text{prior}}$$

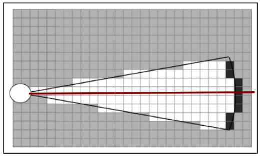
Occupancy Mapping Algorithm

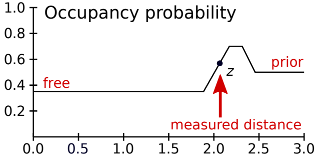
Algorithm 1: OccupancyGridMapping($\{l_{t-1,i}\}$,x_t, z_t)

```
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



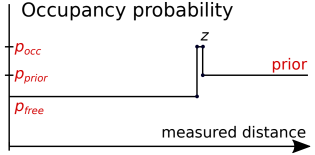
Occupancy value depending on the measured distance

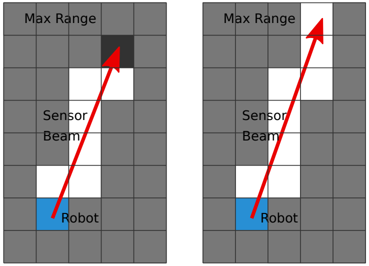
Laser Sensor Model

- The model is “sharp” with the precise obstacle detection
- For the range measurement d_i, update the grid cells along a sensor beam, e.g., using Bresenham's alg.

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
        $p := grid(m_d) p_{free}$;
        $grid(m_i) := p/(2p - p_{free} - grid(m_i) + 1)$
```



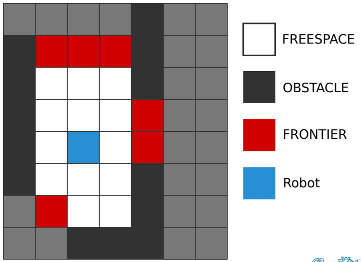


- Multiple cells can be updated by beam raycasting

J. Amanatides and A. Woo (1987), A Fast Voxel Traversal Algorithm for Ray Tracing, Eurographics.
X. Wu (1991), An Efficient Antialiasing Technique, SIGGRAPH Computer Graphics.
C. Schulz and A. Zell (2019), Sub-Pixel Resolution Techniques for Ray Casting in Low-Resolution Occupancy Grid Maps, ECMR.

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 three classes, e.g.,
  - `FREESPACE`: $p(m_i) < 0.4$
  - `UNKNOWN`: $0.4 \leq p(m_i) \leq 0.6$
  - `OBSTACLE`: $p(m_i) > 0.6$
- **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

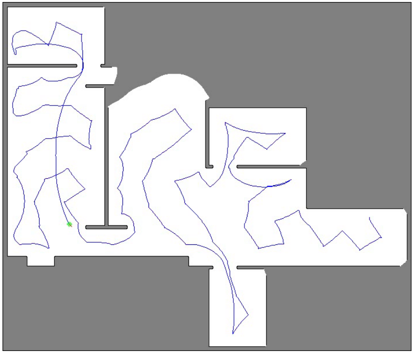


Use grid-based path planning

Frontier-based Exploration Strategy

Algorithm 3: Frontier-based Exploration

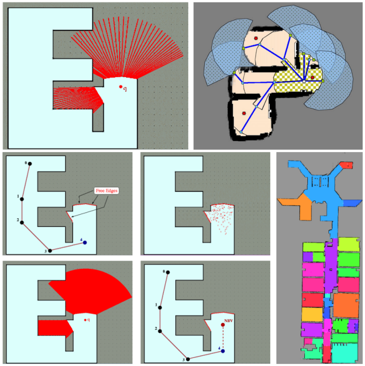
```
$map$ := init($robot$, $scan$);
while there are some reachable frontiers do
    Update occupancy $map$ using new sensor data and Bayes rule;
    $\mathcal{M}$ := Created grid map from $map$ using thresholding;
    $\mathcal{M}$ := Grow obstacle according to the dimension of the robot;
    $\mathcal{F}$ := Determine 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);
```

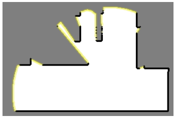


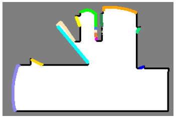
Improvements of the basic Frontier-based Exploration

Several improvements have been proposed in the literature

- Introducing utility based on the expected covered area from a particular location (frontier cell)
  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
  Frontier cells are formed into connected components representing the free edges
  Faigl, Kulich (2013)

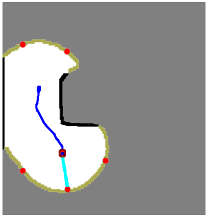




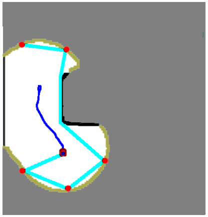


Variants of the Distance Cost

- **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)

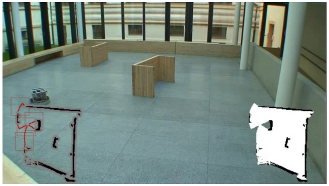


Kulich, M., Faigl, J, Přeučil, L. (2011): On Distance Utility in the Exploration Task. ICRA.

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** $r \in \boldsymbol{R}$,
   $(\langle r_1, g_{r_1}\rangle, \ldots, \langle r_m, g_{r_m}\rangle) =$ 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

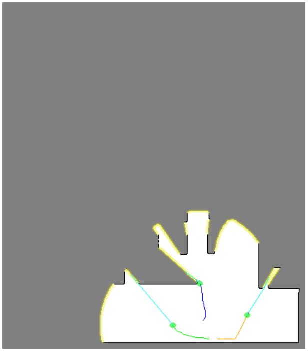


- There are several parts of the exploration procedure where important decisions are made regarding the exploration performance, e.g.
  - How to determine goal candidates from the the frontiers?
  - How to plan a paths and assign the goals to the robots?
  - How to navigate the robots towards the goal?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

Exploration Procedure – Decision-Making Parts

1. Initialize – set of 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


Determination of goal locations and path (cost) to them.

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

   For $n < m$: use Iterative assignment or dummy tasks

   For $n > m$: add dummy robots with costly assignments

4. **Multiple Traveling Salesman Problem – MTSP Assignment**
   - ⟨cluster–first, route–second⟩, 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 $\langle$*cluster–first, route–second*$\rangle$
  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)

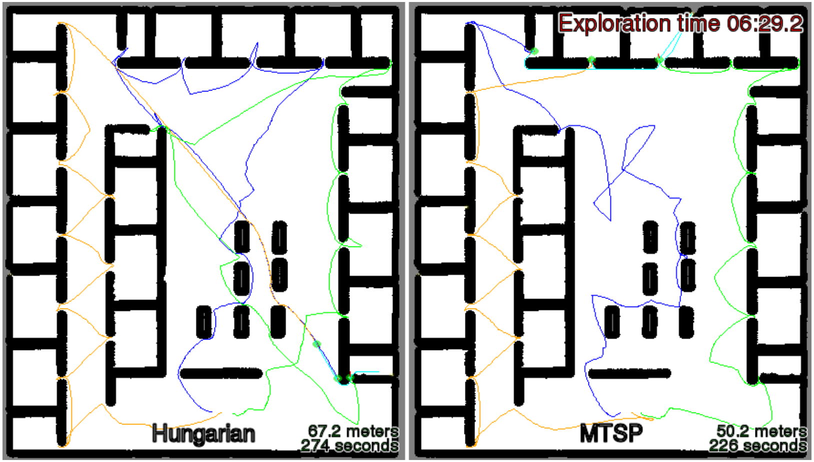
     - Solve the TSP on the set $C_i \cup \{r_i\}$

       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

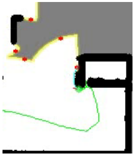
- Replanning as quickly as possible; $m = 3, \rho = 3$ m

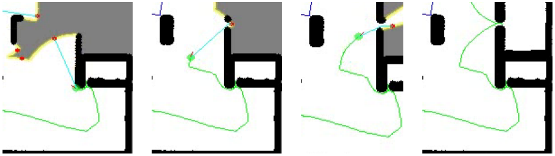


The MTSP assignment provides better performance

Influence of Decision-Making (Exploration Strategy) and Implementation of the Navigation Stack

- Even though we can consider the "best" possible solutions of each invidual parts, the exploration performance depends on the whole solution.
- Thus, locally optimal Hungarian algorithm for task-allocation might not necessarily provide better solutions than, e.g., MTSP-based approach.
- Similarly, a solution of the particular sub-task (**goal candidate selection**) might have side-effects that are exhibited during the missions, e.g., with a combination with the particular navigation technique. For example
  - Vector Field Histrogram (VFH) slows down the robot close to the obstacles.
    Borenstein, J. and Koren, Y. (1991): The vector field histogram-fast obstacle avoidance for mobile robots, T-RO
  - A side effect of the representatives of free edges is that goal candidates are "in the middle of free-edges" and the robot is navigated towards them, which results in faster motion because it is relatively far from the obstacles.





Outline

- Robotic Information Gathering
- Robotic Exploration
- Information Theoretic Approaches
- Inspection Planning – Multi-Goal Planning

Information Theory in Robotic Information Gathering

- Frontier-based exploration assumes perfect knowledge about the robot states and the utility function depends only on the map
- We can introduce uncertainty in the state estimation by entropy-based utility function in which we consider a candidate action within some time horizon with the collection of the data during the action execution
- Control policy is a rule how to select the robot action that reduces the uncertainty of estimate by learning measurements:

$$\mathrm{argmax}_{a \in A}\, I_{MI}[x; z|a],$$

  where A is a set of possible actions, x is a future estimate, and z is future measurement
  - **Mutual information** – how much uncertainty of x will be reduced by learning z

$$I_{MI}[x; z] = H[x] - H[x|z]$$

    $H[x]$ – the current **entropy**
    $H[x|z]$ – future/predicted entropy
  - **Conditional Entropy** $H[x|z]$ is the expected uncertainty of x after learning unknown z (collecting new measurements)
  - **Entropy** – uncertainty of x: $H[x] = -\int p(x) \log p(x) dx$

Decrease Computational Requirements of the Information-based Approaches

- Sensor placement approach with raycasting of the sensor beam and determination of the distribution over the range returns
- We can decrease the computational requirements by using simplified approach where the action is selected to maximize the entropy over the sensed regions in the current map

$$\operatorname{argmax}_{a \in A} \sum_{x \in \boldsymbol{R}(a)} H[P(x)],$$

  where $\boldsymbol{R}(a)$ represents the region sensed by the action a

  A. A. Makarenko et al., (2002): An experiment in integrated exploration. IROS.

  - We can further include uncertainty in the pose estimation (localization)

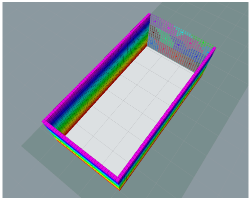
    H. Carrillo et al., (2018): Autonomous robotic exploration using a utility function based on Rényi's general theory of entropy. Autonomous Robots, 2018.

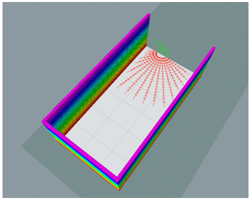
- Computational cost can be decreased using Cauchy-Schwarz Quadratic Mutual Information (CSQMI) defined similarly to mutual information

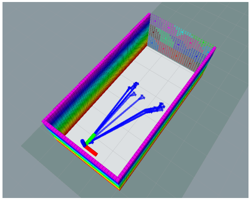
  Charrow, B. et al., (2015): Information-theoretic mapping using Cauchy-Schwarz Quadratic Mutual Information. ICRA.

Actions

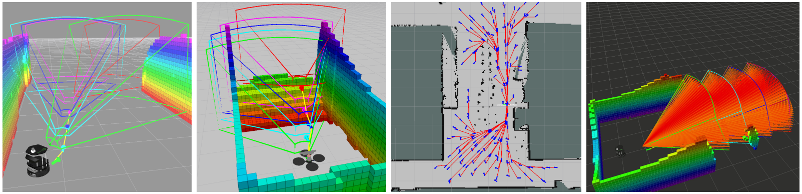
- Actions are shortest paths to cover the frontiers


Detect and cluster frontiers

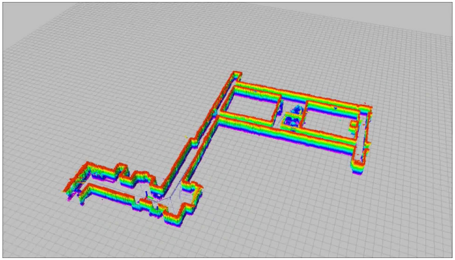

Sampled poses to cover a cluster


Paths to the sampled poses

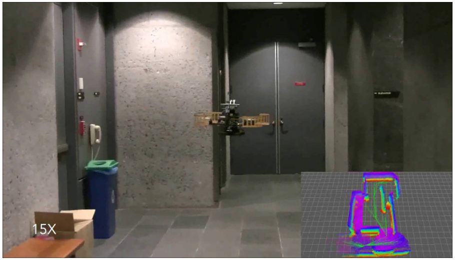
- Select an action (a path) that maximizes the rate of Cauchy-Schwarz Quadratic Mutual Information



Example of Autonomous Exploration using CSQMI



Ground vehicle



Aerial vehicle

- Planning with trajectory optimization – determine trajectory maximizing I_{CS}

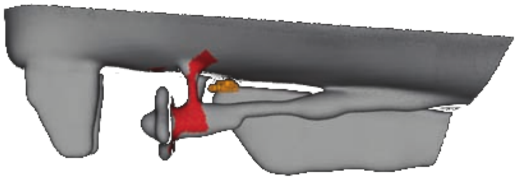
  Charrow, B. et al., (2015): Information-Theoretic Planning with Trajectory Optimization for Dense 3D Mapping. RSS.

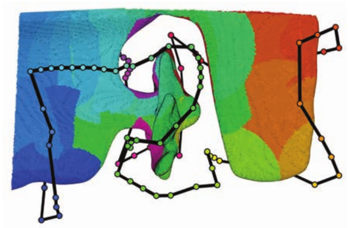
Outline

- Robotic Information Gathering
- Robotic Exploration
- Information Theoretic Approaches
- Inspection Planning – Multi-Goal Planning

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.





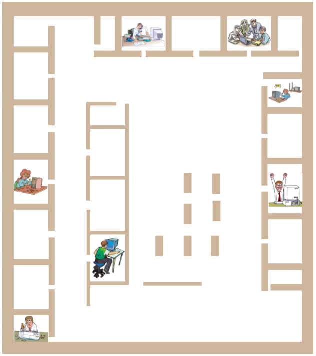
https://www.youtube.com/watch?v=8azP_9VnMtM

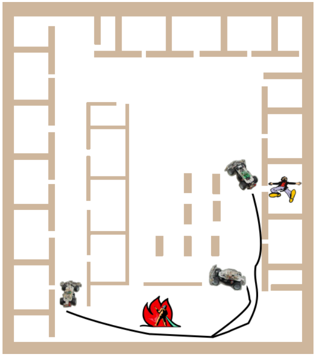
Englot, B., Hover, F.S. (2013): **Three-dimensional coverage planning for an underwater inspection robot**. Robotics and Autonomous Systems.

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)

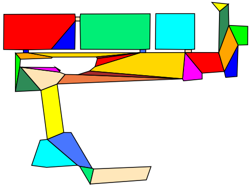


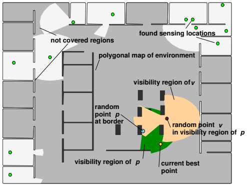


Inspection Planning – Decoupled Approach

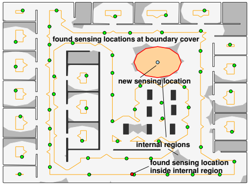
1. Determine sensing locations such that the whole environment would be inspected (seen) by visiting them (**Sampling design problem**)

In the geometrical-based approach, a solution of the Art Gallery Problem


Convex Partitioning (Kazazakis and Argyros, 2002)



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


found sensing locations at boundary cover
new sensing location
internal regions
found sensing location inside internal region
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

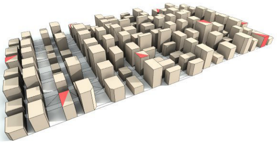
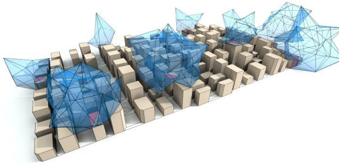
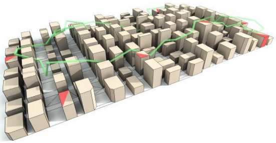
- Determine a cost-efficient path from which a given set of target regions is covered
- For each target region a subspace $S \subset \mathbb{R}^3$ from which the target can be covered is determined

  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)

  PRM – Probabilistic Roadmap Method – sampling-based motion planner, see lecture 7
- The problem can be 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

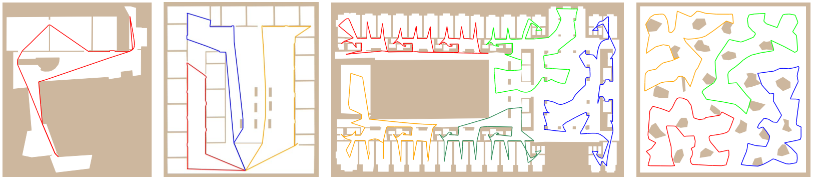
  Janoušek and Faigl, (2013) ICRA

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**)

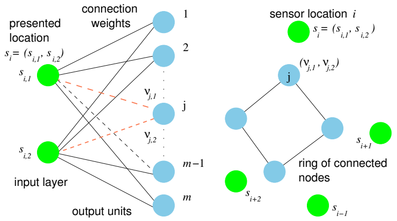
- Neurons' weights represent nodes $\mathcal{N} = \{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_m\})$ in a plane
- Nodes are organized into a ring
- Sensing locations $\boldsymbol{S} = \{\mathbf{s}_1, \ldots \mathbf{s}_n\}$ are presented to the network in a random order
- Nodes compete to be winner according to their distance to the presented goal $\mathbf{s}$

$$\boldsymbol{\nu}^* = \mathrm{argmin}_{\boldsymbol{\nu} \in \mathcal{N}} |\mathcal{D}(\{\boldsymbol{\nu}, \mathbf{s})|$$

- 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 $|\mathcal{D}(\boldsymbol{\nu}, s)|$
- 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

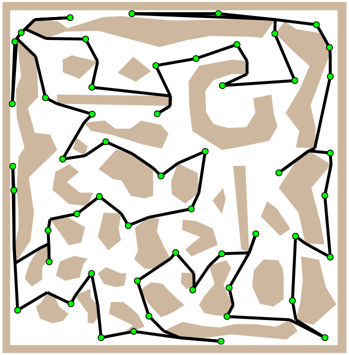
Fort, J.C. (1988), Angéniol, B. et al. (1988), Somhom, S. et al. (1997), etc.

Unsupervised Learning for the Multi-Goal Path Planning

- Unsupervised learning procedure for the Multi-goal Path Planning (**MTP**) problem a robotic variant of the Traveling Salesman Problem (TSP)

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 $\mathrm{argmin}_{\nu \in \mathcal{N}} |S(g, \nu)|$;
        adapt$(S(g, \nu), \mu f(\sigma, l) |S(g, \nu)|)$;
        $error \leftarrow \max\{error, |S(g, \nu^\star)|\}$;
    $\sigma \leftarrow (1 - \alpha)\sigma$;
until $error \leq \delta$;
```



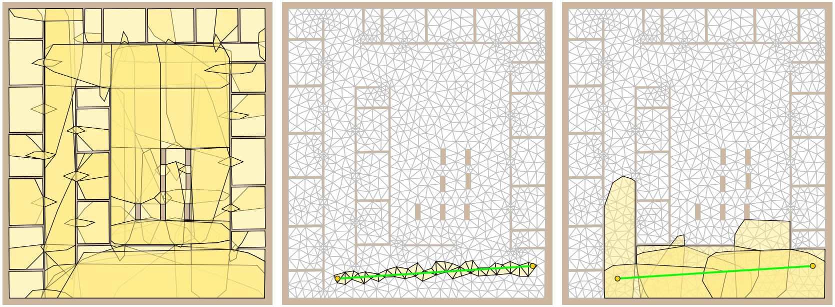
- For multi-goal path planning – the **selectWinner** and **adapt** procedures are based on the solution of the **path planning problem**

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 $\mathcal{W}$

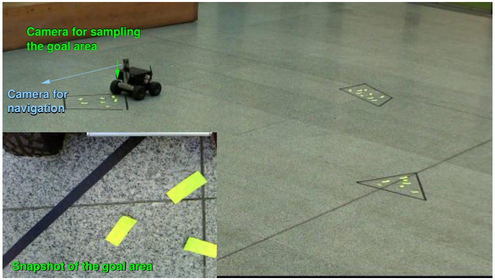
- Convex cover set of $\mathcal{W}$ created on top of a triangular mesh
- Incident convex polygons with a straight line segment are found by walking in a triangular mesh technique



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

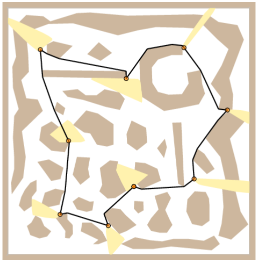
- Approximate algorithms exist for particular problem variants

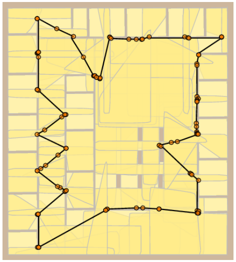
  E.g., Disjoint unit disk neighborhoods

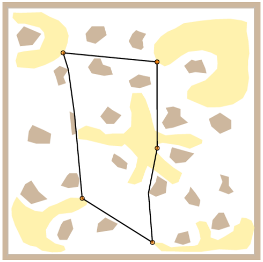
- Flexibility of the unsupervised learning for the TSP allows generalizing the unsupervised learning procedure to address the TSPN

- **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
n=9, T= 0.32 s

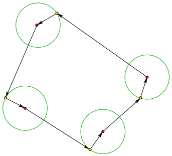
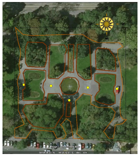

Convex Cover Set
n=106, T=5.1 s

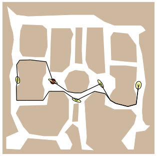

Non-Convex Goals
n=5, T=0.1 s

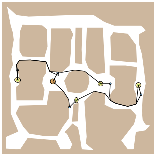
Faigl, J. et al. (2013): Visiting Convex Regions in a Polygonal Map. Robotics and Autonomous Systems.

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

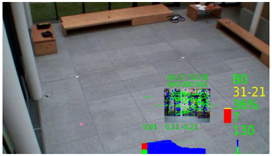
Real overall error at the goals decreased from 0.89 m → 0.58 m (about 35%)

- 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

Topics Discussed

- Robotic information gathering – informative path planning
- Robotic exploration of unknown environment
  - Occupancy grid map
  - Frontier based exploration
  - Exploration procedure and decision-making
  - **TSP-based distance cost** in frontier-based exploration
  - Multi-robot exploration and task-allocation
- Mutual information and informative path planning *informative and motivational*
- Inspection planning
  - Unsupervised learning for multi-goal path planning
  - Traveling Salesman Problem with Neighborhoods (TSPN)

- Next: Multi-goal (data collection) planning