NA 568 Mobile Robotics: Methods & Algorithms Winter 2023 – Homework 4 – Mapping

Maani Ghaffari
University of Michigan

February 28, 2023

This is a reminder that no late HW is accepted. We are using Gradescope for turning in HW; see relevant information on the course Canvas site. For Homeworks 1-5, the lowest grade will be automatically dropped for everyone. Homework 6 is a bonus and can be submitted to replace your lowest grade of Homeworks 1-5, effectively dropping the two lowest grades. If you do not submit Homework 6, your grade will be 0 and automatically dropped as your lowest grade (This is perfectly fine).

This problem set counts for about 11% of your course grade. You are encouraged to talk at the conceptual level with other students, but you must complete all work individually and may not share any non-trivial code or solution steps. See the syllabus for the full collaboration policy.

Submission Instructions

Your assignment must be received by 11:55 pm on Friday, March 17th (Anywhere on Earth Time). This corresponds to 6:55 AM on March 18th in Eastern Time. This is selected out of fairness to all our students, including those who take the course remotely. You are to upload your assignment directly to the Gradescope website as two attachments:

1. A `.tar.gz` or `.zip` file *containing a directory* named after your uniqname with the structure shown below (matlab or python files, a python notebook will be acceptable as well):

   `uniqname_hw4.tgz`:
   ```
   uniqname_hw4/
   uniqname_hw4/ogm_CSM.m
   uniqname_hw4/ogm_continuous_CSM.m
   uniqname_hw4/ogm_S_CSM.m
   uniqname_hw4/ogm_continuous_S_CSM.m
   ```

2. A PDF with the written portion of your write-up. Scanned versions of hand-written documents, converted to PDFs, are perfectly acceptable. No other formats (e.g., .doc) are acceptable. Your PDF file should adhere to the following naming convention: `uniqname_hw6.pdf`.

1 Discrete Counting Sensor Model

Assuming we are updating the i-th map cell m_i. We define the measurements $Y = \{y_1, ..., y_N | y_j \in \{0, 1\}\}$ indicating whether a beam was reflected by ($y_j = 1$) or passed through ($y_j = 0$) the map cell m_i.

Adopting a conjugate model (more detail in Robotic Mapping slide 10-15), we have the prior over θ_i given by $Beta(\alpha_0, \beta_0)$, where $\alpha_0, \beta_0 \in \mathbb{R} > 0$ are prior hyperparameters, usually set as $\alpha_0 = \beta_0 \approx 0$ to place a small, uninformative prior on occupancy probability. Applying Bayes' rule, we find that the posterior is given by $Beta(\alpha_i, \beta_i)$, where α_i and β_i are defined as follows:

$$\alpha_i := \alpha_0 + \sum_{j=1}^{N} y_j \tag{1}$$

$$\beta_i := \beta_0 + \sum_{j=1}^{N} (1 - y_j). \tag{2}$$

That is, α_i maintains a count of instances where a beam is reflected in the i-th grid cell, while β_i is a count of instances where a beam passed through the cell, giving the model its name. We can update the mean and variance of our counting sensor model by the following equations:

$$\mathbb{E}[\theta_i] = \frac{\alpha_i}{\alpha_i + \beta_i} \quad \text{and} \quad \mathbb{V}[\theta_i] = \frac{\alpha_i \beta_i}{(\alpha_i + \beta_i)^2 (\alpha_i + \beta_i + 1)} \tag{3}$$

A detailed algorithm is listed in Algorithm 1, where beam parameters $w_{obstacle}$ and w_{beam} are defined to determine occupied space and free space. $w_{obstacle} = 2 \cdot grid\_size$ is the width of obstacle, and $w_{beam} = \frac{2\pi}{number\_of\_beams}$ is the width of beam.

Algorithm 1 `discrete_CSM`(m_i, z)

Require: map cell $m_i = (r, \phi)$ range and bearing difference to the robot pose; neighbor robot pose's scan $z = (z_l, \theta)$ range and bearing of the scan;

$k = \arg\min_j |\phi - \theta_j|$ ▷ Find the beam that goes through cell m_i.
if $r > \min(z_{max}, z_l^k + \frac{w_{obstacle}}{2})$ or $|\phi - \theta_k| > \frac{w_{beam}}{2}$ **then**
    pass ▷ The map cell is outside of the perception field.
else if $z_l^k < z_{max}$ and $|r - z_l^k| < \frac{w_{obstacle}}{2}$ **then**
    $\alpha_i = \alpha_i + 1$ ▷ Update occupied space.
else if $r < z_l^k$ and $z_l^k < z_{max}$ **then**
    $\beta_i = \beta_i + 1$ ▷ Update free space.
end if

2 Continuous Counting Sensor Model

From the lecture (more detail in Robotic Mapping slide 16-18), we know that if we have the concentration parameter (α_i, β_i), where α_i represents the free-space class and β_i represents the occupied class, we can have our continuous counting sensor model updated by the following equations:

Algorithm 2 `continuous_CSM`(m_i, x_t, z)

Require: map cell $m_i = (x_i, r, \phi)$ map cell coordinates in global frame, range and bearing difference to the robot pose; neighbor robot pose x_t; neighbor robot pose's scan $z = (z_l, \theta)$ range and bearing of the scan;

$k = \arg\min_j |\phi - \theta_j|$ ▷ k is the beam index, not to be confused by the kernel function.
$x_k^* = global(x_k)$ ▷ $x_k^* \in \mathbb{R}^2$ is the global coordinates of the k-th scan's end point x_k.
$d_1 = \|x_i - x_k^*\|$ ▷ $x_i \in \mathbb{R}^2$ is the global coordinates of the map cell m_i.
if $d_1 < l$ **then**
    $\alpha_i = \alpha_i + \text{kernel}(d_1)$ ▷ Update occupied space.
end if
for every sample point x_l along beam z_k **do** ▷ $x_l \in \mathbb{R}^2$ is a sample point in the robot frame.
    $x^* = global(x_l)$ ▷ $x^* \in \mathbb{R}^2$ is the global coordinates of sample point x_l.
    $d_2 = \|x_i - x*\|$
    if $d_2 < l$ **then**
        $\beta_i = \beta_i + \text{kernel}(d_2)$ ▷ Update free space.
    end if
end for

$$\alpha_i \coloneqq \alpha_0 + \sum_{j=1}^{N} k(x_i, x_j^*) y_j \tag{4}$$

$$\beta_i \coloneqq \beta_0 + \sum_{j=1}^{N} k(x_i, x_j^*)(1 - y_j) \tag{5}$$

$$\mathbb{E}[\theta_i] = \frac{\alpha_i}{\alpha_i + \beta_i} \quad \text{and} \quad \mathbb{V}[\theta_i] = \frac{\alpha_i \beta_i}{(\alpha_i + \beta_i)^2(\alpha_i + \beta_i + 1)} \tag{6}$$

In this assignment, we choose the sparse kernel for $k(x_i, x_j^*) = \text{kernel}(d)$. That is:

$$\text{kernel}(d) = \begin{cases} \sigma_0[\frac{1}{3}(2 + \cos(2\pi\frac{d}{l}))(1 - \frac{d}{l}) + \frac{1}{2\pi}\sin(2\pi\frac{d}{l})], & \text{if } d < l \\ 0, & \text{if } d \geq l \end{cases} \tag{7}$$

where hyperparameters are $\sigma_0 = 0.1$, $l = 0.2$, and the distance metric is the usual Euclidean norm, i.e., $d = \|x_i - x_j^*\|$. x_i is the global coordinates of the map cell m_i, and x_j^* is the end point of the beam measurement in the global coordinate.

Notice: There are several free space representation in continuous CSM: Doherty *et al.* [2] use the line projection method to sample free points. Gan *et al.* [3] use linear interpolation to model free points. In the listed algorithm, we are using the sampling method, but both methods are acceptable.

3 Semantic Counting Sensor Model

Now we have some noisy semantically labeled measurements of the Intel dataset, as shown in Figure 1. The measurements $Y = \{y_1, ..., y_N | y_j \in \{1, 2, 3, 4, 5, 6\}\}$ are classified into six different categories: *north rooms, west rooms, east rooms, south rooms, middle rooms, and hallways.* In the result map, we will add one more category, that is the *free space.*

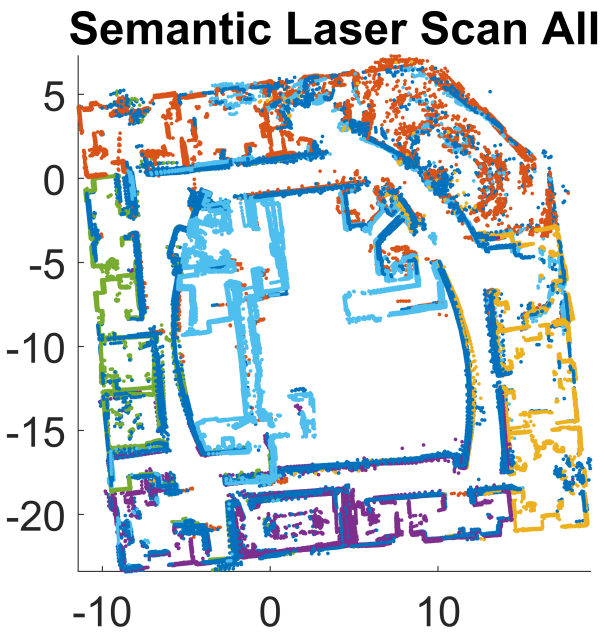

Figure 1: Noisy semantic point cloud measurements; the Intel dataset.

We are using parameters $\alpha_i^k = \{\alpha_i^1, ..., \alpha_i^K\}$ where $K = 7$ is the total number of semantic classes, each α_i^k represents a segmentation class, and α_i^7 represents the free space.

The discrete parameters can be updated by the following equation, where $\delta_{y_j k}$ is the Kronecker delta:

$$\alpha_i^k := \alpha_0^k + \sum_{j=1}^{N} \delta_{y_j k}, \quad \delta_{y_j k} = \begin{cases} 0 & \text{if } y_j \neq k, \\ 1 & \text{if } y_j = k. \end{cases} \tag{8}$$

And the continuous parameters can be updated by the following equation using the defined kernel:

$$\alpha_i^k := \alpha_0^k + \sum_{j=1}^{N} k(x_i, x_j^*)\delta_{y_j k}, \quad \delta_{y_j k} = \begin{cases} 0 & \text{if } y_j \neq k, \\ 1 & \text{if } y_j = k. \end{cases} \tag{9}$$

For both discrete and continuous cases, we can update the mean and variance of our semantic counting sensor model by the following equations, where α_i^{k*} is the highest α value in cell i among all classes:

$$\mathbb{E}(\theta_i^k) = \frac{\alpha_i^k}{\sum_{k=1}^K \alpha_i^k} \quad \text{and} \quad \mathbb{V}(\theta_i^k) = \frac{(\frac{\alpha_i^{k*}}{\sum_{k=1}^K \alpha_i^k})(1 - \frac{\alpha_i^{k*}}{\sum_{k=1}^K \alpha_i^k})}{\sum_{k=1}^K \alpha_i^k + 1} \tag{10}$$

4 Coding Setup

In this assignment, you will be implementing both discrete and continuous counting sensor models for occupancy grid maps and semantic grid maps.

We will evaluate the 2D mapping algorithm using the `Intel dataset` [1] in `data/`. We've provided visualization tools for you in `utils/` and `utils.py`. After completing each task, you can test your code by running the command `run(task_num, true)` or `python run.py --task_num TASK_NUM` to visualize your map.

Depends on your computer, it takes 1-5 minutes to finish building non-semantic maps and 8-15 minutes to finish building semantic maps. To debug while coding try to only build a small part of the map to make sure it works or use a larger grid size to speed up the computation.

Task 1: Discrete Counting Sensor Model (20 points)

A. (20 pts) Implement a 2D counting sensor model (CSM) for occupancy grid mapping in `ogm_CSM.m` or `ogm_CSM.py`. Visualize your map with `grid_size = 0.135 m` and its associated variance map. Include those two figures in your pdf.

Task 2: Continuous Counting Sensor Model (30 points)

A. (15 pts) Implement a 2D continuous counting sensor model (CSM) in `ogm_continuous_CSM.m` or `ogm_continuous_CSM.py`. Visualize your map with `grid_size = 0.135 m` and plot its associated variance map. Include those two figures in your pdf.

B. (10 pts) Generate the map and the variance map using `grid_size = [0.135, 0.270, 0.5]`. Study the effects of varying the map resolution on the map inference. Provide a conclusion after discussing the results and clearly explain your reasons behind your conclusion. Include all figures (6 plots in total) in your pdf with appropriate captions.

C. (5 pts) Compare the continuous CSM to discrete CSM. What's the difference? What is the advantage of continuous CSM?

Task 3: Discrete Semantic Counting Sensor Model (20 points)

A. (20 pts) Implement a semantic counting sensor model (S-CSM) in `ogm_S_CSM.m` or `ogm_S_CSM.py`. Visualize your map with `grid_size = 0.135 m` and visualize the variance of the class with highest probabilities at each grid. Include those two plots in your pdf.

Task 4: Countinuous Semantic Counting Sensor Model (30 points)

A. (15 pts) Implement a continuous semantic counting sensor model (S-CSM) in `ogm_continuous_S_CSM.m` or `ogm_continuous_S_CSM.py` using the defined kernel. Visualize your map and variance of the class with highest probabilities at each grid. Include those two plots in your pdf.

B. (10 pts) Generate the map and the variance map using `grid_size = [0.135, 0.270, 0.5]`. Study the effects of varying the map resolution on the map inference. Provide a conclusion after discussing the results and clearly explain your reasons behind your conclusion. Include all figures (6 plots in total) in your pdf with appropriate captions.

C. (5 pts) Compare the continuous S-CSM to discrete S-CSM. What's the difference? What is the advantage of continuous S-CSM? Provide a discussion on which mapping algorithm(s) should be implemented on the robot and why.

Task 5: BKI Semantic Mapping (30 extra credits)

Build and run the BKI Semantic Mapping [3,4] on KITTI semantics dataset [5] sequence 04. Submit screenshots of the visualization in RViz in your pdf.

Notice the system dependencies:

- Ubuntu system. The mapping algorithm is build on Ubuntu system. It has been tested on Ubuntu 16.04 and Ubuntu 18.04. If you didn't have an Ubuntu system, you could create a virtual machine.
- ROS system. The mapping algorithm has been tested on ROS kinetic and ROS melodic. You can follow the installation guide for ROS in the documentation.
- A catkin workspace (catkin_ws). You can follow the steps in the tutorial.

Read the README in the repository and do the following steps:

1. Build the repository with catkin. Run the following commands in your catkin workspace (catkin_ws):
   - cd src/
   - git clone https://github.com/ganlumomo/BKISemanticMapping
   - cd ..
   - catkin_make
   - source catkin_ws/devel/setup.bash
2. Download semantic KITTI dataset sequence 04 data from `https://drive.google.com/file/d/19Dv1jQqf-VGKS2qvbygFlUzQoSvu17E5/view` and uncompress it into the data folder.
3. Run the demo with the following command:
   - roslaunch semantic_bki semantickitti_node.launch
4. Screenshot the visualization of the semantic map in RViz, include it in your PDF submission.
5. You should also create a video of the mapping sequence, include it in your code submission file. (You don't have to run the whole sequence if you only have limited computing resource)

   Remark 1. *It is expected that the program will stop at the 99th `.bin` file. To process more files you can change the corresponding parameter in `semantickitti.yaml` under the path `BKISemanticMapping/config/datasets`.*

5 Reference

[1] Intel Dataset: http://www2.informatik.uni-freiburg.de/~stachnis/datasets.html

[2] Task 1-2: Doherty, K., Shan, T., Wang, J., & Englot, B. (2019). Learning-aided 3-D occupancy mapping with Bayesian generalized kernel inference. IEEE Transactions on Robotics, 35(4), 953-966.

[3] Task 3-5: Gan, L., Zhang, R., Grizzle, J. W., Eustice, R. M., & Ghaffari, M. (2020). Bayesian spatial kernel smoothing for scalable dense semantic mapping. IEEE Robotics and Automation Letters, 5(2), 790-797.

[4] Task 5 Github repository: https://github.com/ganlumomo/BKISemanticMapping

[5] Semantics KITTI Dataset: http://www.cvlibs.net/datasets/kitti/eval_semantics.php

FAQ

1. Q: In Discrete Counting Sensor Model (Discrete CSM), the if-else statement in Algorithm 1 Line 4, why does the absolute error between a range of pose and range of scan have to be within the grid size, i.e., $|r - z_l^k| < \frac{w_{obstacle}}{2}$?
   A: It is similar to the occupancy map. The idea is that we want to find specific grids that are close to where the beam was stopped. This indicates an obstacle blocking the beam at that grid. If the distance from the robot to that grid is larger than the range we receive from the measurement, the grid is then either too close to the robot or too far away. If the grid is closer to the robot than the measurements, then it means the beam passes through it, and thus we should update it as free.

2. Q: When I discretize the Continuous Counting Sensor Model (Continuous CSM), I think the finer we make the discretization, the better. At the same time, if we discretize too finely, we may update data many times as the for loop would iterate over all the points, and we could have $d_2 < l$ multiple times. Should we do something like updating β only for the closest point? Or should we never discretize smaller than the grid size?
   A: As the algorithm states, β can be calculated multiple times instead of updating the closest point. You can avoid adding β several times within one grid by setting the step size close to the grid size.