Introduction to Mobile Robotics 2023-2024

Assignment

The assignment is due by 2023-11-24 12:00

This assignment will extend a lot of what was previously learned in the lecture, tutorial, and labs. You will be working by yourself. The assignment is broken down into 4 sections that cover perception, localisation and mapping. The code template and detailed instructions for this assignment are in a repository (repo) on Github which you can find through the following link: `https://github.com/kevinDuan1/MOB-23-UoE-CW.git`. Please use Noteable and clone the Github repo to start your work.

1 Semantic Segmentation (15 marks)

As we have learned in the lecture and Lab, evaluation metrics impact the quality of semantic segmentation.

In this section, you are tasked with implementing the Dice Similarity Coefficient (DSC), expressed as:

$$(\text{DICE}(A, B) = \frac{2 \times |A \cap B|}{|A| + |B|}) \tag{1}$$

Much like the Intersection over Union (IoU) metric, DICE assesses the model's performance by measuring the overlap between the predicted and the actual ground-truth pixels. (9 marks)

Then, compare the effects of using different evaluation metrics and illustrate how might model architecture choice, such as U-net, influence the performance of the model. (6 marks)

2 Object Detection (15 marks)

As we discussed in the lecture and lab, non-maximum suppression is the algorithm that commonly uses in object detection for removing significant overlapping bounding boxes. However, as we see in the object detection lab, non-maximum suppression was recursively applied to the remaining boxes and it will cause a detection miss if an object lies within the predefined overlap threshold.

To address this issue, Bodla, N. etc. proposed an improved version of non-maximum suppression called Soft-NMS. Soft-NMS only decays the detection score of all other object-bounding boxes as a continuous function of their overlap with the current object-bounding box. (Hence no object bounding box is eliminated in this process)

In this section, you will be implementing the Soft-NMS algorithms. Hint: Soft-NMS only has minor differences compared to regular non-maximum suppression. You could start with implementing regular non-maximum suppression first, then change the part where you remove the score and bounding box from the list.

- **Soft-NMS**

Implement soft_nms() in the notebook. Here is the pseudo code for Soft-NMS.

Note that this is a *simplified* version of Soft-NMS. The actual implementation of Soft-NMS is slightly different than what you see in the pseudo-code we provided you. But the main idea is the same. You use a penalty function to update the detection score rather than removing bounding boxes with a fixed threshold.

Algorithm 1 Soft-NMS

Require: $B = \{b_1, ..., b_n\}$, $S = \{s_1, ..., s_n\}$, σ

B is the list of initial detection boxes.
S contains corresponding detection scores.
σ is the variances used in the Gaussian penalty function

$D \leftarrow \{\}$ ▷ For store detection bounding boxes result from NMS
$Z \leftarrow \{\}$ ▷ For store detection scores result from NMS
while $B \neq empty$ **do**
  $m \leftarrow argmax S$
  $D \leftarrow D \cup b_m$; $Z \leftarrow Z \cup s_m$
  $B \leftarrow B - b_m$; $S \leftarrow S - s_m$
  for b_i in B **do**
    $M_{iou} \leftarrow iou(b_m, b_i)$
    $s_i \leftarrow s_i e^{\frac{-M_{iou}^2}{\sigma}}$ ▷ Gaussian penalty function
return D, Z

3 (Stereo) Visual Odometry and Extended Kalman filter (45 marks)

SVO is a procedure that determines ego-motion by identifying and tracking visual landmarks in the environment, using cameras mounted onboard the vehicle. In this section, you are going to use opencv to apply the feature matching you learnt in the lecture and lab and track the position of the truck through a sequence of Kitti dataset images.

3.1 Feature matching (10 marks)

Implement extract_features(), match_feature(), filter_matches_distance() and estimate_motion() functions in the notebook.

3.2 Motion estimation (15 marks)

Implement estimate_motion() and visual_odometry() functions using Essential Matrix decomposition or PnP in the notebook.

3.3 Extended Kalman filter (20 marks)

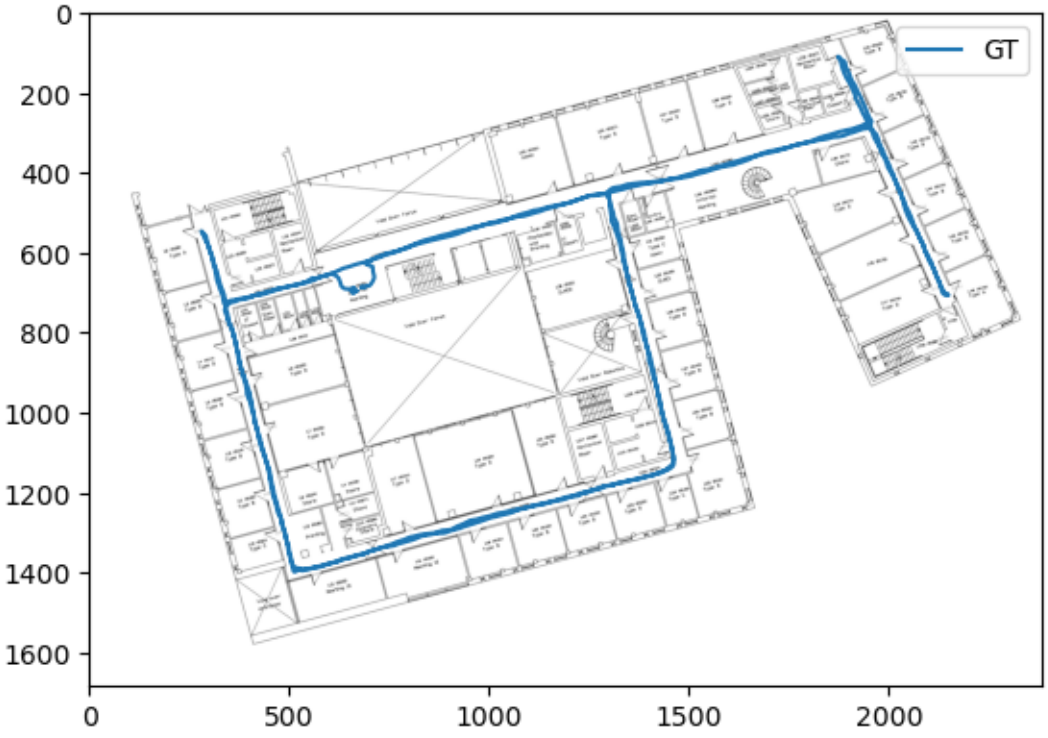

Figure 1: Ground truth trajectory on the 1st floor of the Informatics Forum

In this section, you are going to use the extended Kalman filter to fuse the readings from an Inertia measurement unit and the WiFi positioning for the 1st floor at Informatics Forum. You are given the motion model (obtained from the inertial odometry) and the measurement model (obtained from a GPS-like WiFi positioning system) formulas, please complete the code and visualize the trajectory.

4 Point cloud transformation and filtering (25 marks)

LiDAR point cloud is very useful when we do mapping (e.g. Occupancy Grid) as it could provide accurate geometry information of surroundings. But when we do mapping with LiDAR, any other moving objects in the scene will affect our mapping result as they are not stationary and cause some errors to our map. One way to address it is running object detection first to recognize all the objects in the scenes and remove them from the LiDAR point cloud.

In this section you will be implementing a similar filtering algorithm but with a simpler approach. You will first need to project 3D LiDAR point clouds to 2D image plane of the camera. Then you filter out all the points that are within bounding boxes.

4.1 Project LiDAR point cloud to image (15 marks)

You first need to project LiDAR point cloud to the image plane. First, you should transform 3D LiDAR points from LiDAR coordinates to camera coordinates. Then transform points in camera coordinates to the image plane.

Implement lidar_pc_to_cam() function in the notebook.

Hint: crop_pc_and_reformat() function takes raw points cloud input from the file and filters in range points based on the camera field of view. But we keep the intensity for you to remove it. Think about what you could do with it to make your transformation easier.

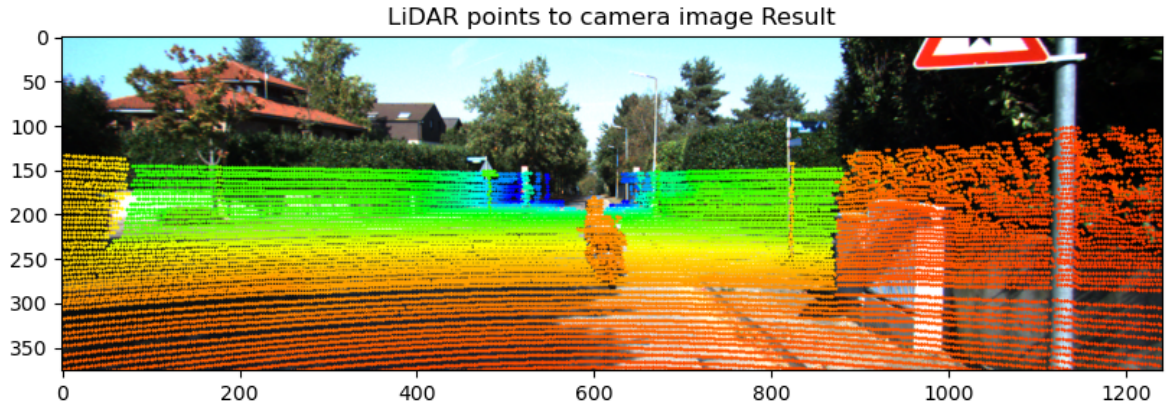

Figure 2: Example output of projecting point cloud to image

4.2 Filter point cloud (10 marks)

Implement filter_in_bbox_points() function in the notebook. This function should check if the points (which are already projected to the image plane) are in a bounding box and decide if should remove them from the point cloud

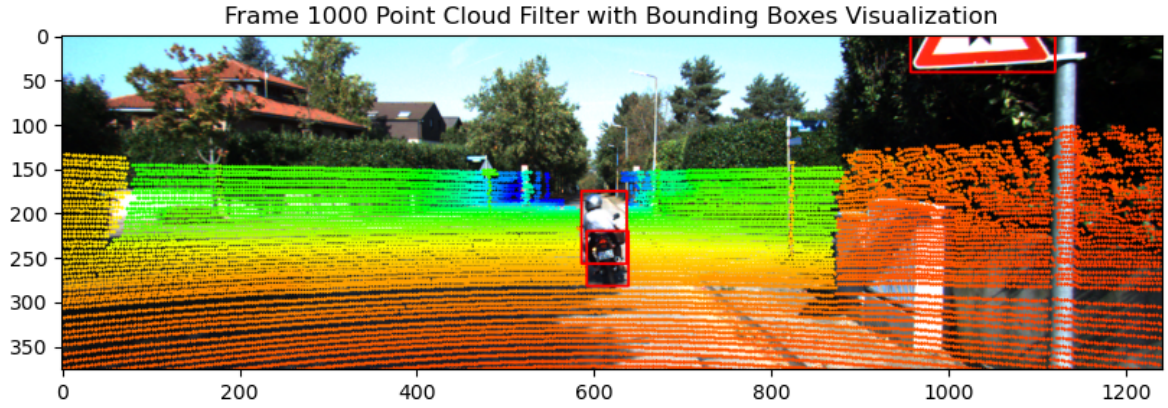

Figure 3: Example output of filtered point cloud

5 Submission

We will use the Learn system for uploading your code and the uploading systems will be available from Week 8 onwards. Your submission on Learn page should be a compressed package containing the required 4 codes. Please follow the instructions in these 4 notebooks and show your result as required. The submission file name of the package should be 'studentNo.zip'.