Introduction to Mobile Robotics 2022-2023

Assignment

The assignment is due by 2022-11-25 12:00

This assignment will be extending a lot of what was previously learned in the lecture, tutorial, and labs. You will be working by yourself. The assignment is going to be broken down into 4 sections that cover perception, localisation and mapping. The code template and detailed instruction for this assignment is in a repository (repo) on Github which you can find through the following link: https://github.com/MrTooOldDriver/MOB-22-UoE-CW. Please use Noteable and clone this Github repo to start your work.

1 Camera Calibration (15 marks)

As we have learned in the lecture and tutorial 1, a 3D checkerboard can be used for camera calibration.

In this section, we provide 40 images of 3D checkerboards from a video for camera calibration. These frames contain two checkerboards with 20 mm and 40 mm grid sizes. The frames are taken from different angles with different surroundings. Please choose **one image** for the calibration and explain why you choose it. (5 marks)

After the chosen, please use this image to calibrate the camera, which needs to calculate the intrinsic matrix $\mathbf{K}$, rotation matrix $\mathbf{R}$, and translation vector $\mathbf{t}$. (10 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., Singh, B., Chellappa, R., Davis, L. S. 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 using a penalty function to update the detection score rather than removing bounding boxes with a fixed threshold it.

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 on-board the vehicle. In this section you are going to use opencv

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 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

to apply feature matching you learnet in the letures 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)

In this section, you are going to use extended Kalman filter to fuse the odometry you obtained from cameras with the gps data we provided. You are given with motion model (vo) and measurement model (gps) 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 restrive accurate geometry information of surrounding. 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 more simple 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 image plane. First you should transform 3D LiDAR points from LiDAR coordinates to camera coordinates. Then transform points in camera coordinates to 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 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.

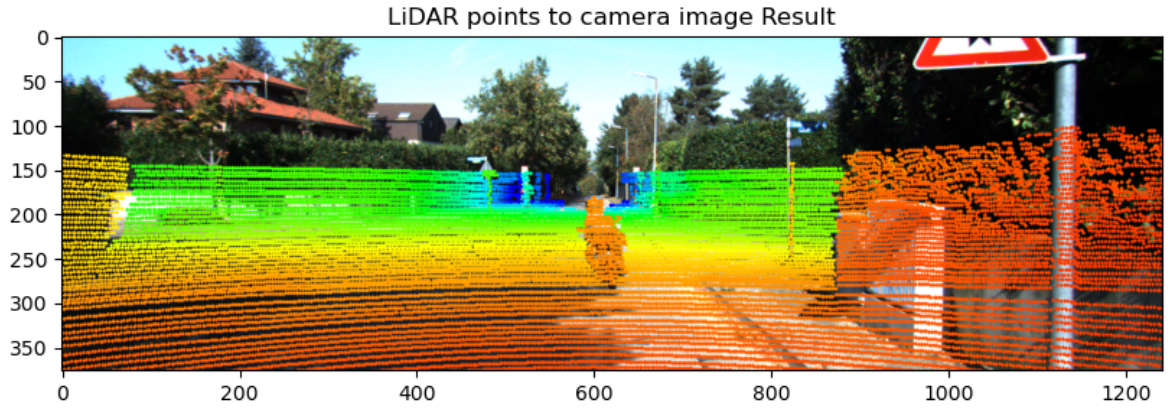

Figure 1: Example output of projecting point cloud to image

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 decided if should remove it from point cloud

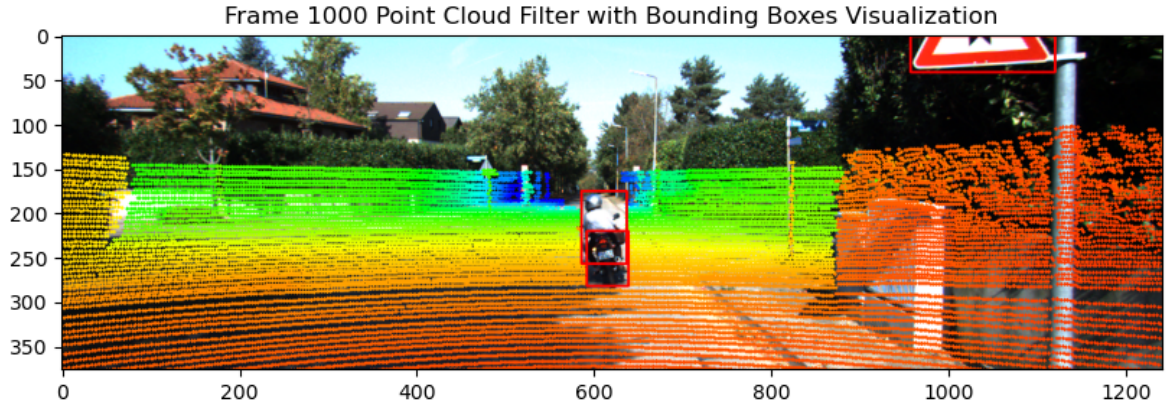

Figure 2: Example output of filtered point cloud

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 should be a compressed package containing required 4 codes. Please follow the instruction in these 4 notebooks and show your result as required. File name of the package should be studentNo.zip.