CSE-483: Mobile Robotics **Monsoon 2019**

Assignment-1

Release: 9th August *Deadline: 23rd August*

Instructions

- This assignment is designed to make you familiar with transformations, camera modelling, and the Direct Linear Transform (DLT) algorithm. Work in teams of two.
- Code can be written in Python or C++ only. OpenCV or any equivalent library can be used for image input & output. Make sure your code is modular since you might be reusing them for future assignments.
- Submit your code files and a report (or as a single Jupyter notebook if you wish) as a zip file, named as ⟨team_id⟩_assignment1.zip.
- The report should include all outputs and results and a description of your approach. It should also briefly describe what each member worked on. The scoring will be primarily based on the report.
- Refer to the late days policy and plagiarism policy in the course information document.
- Start early! This assignment may take fairly long.

Q1) LiDAR-Camera Fusion

You are working on a self-driving car and the team has decided to fuse image data from a camera with distance measurements from a LiDAR (a laser scanner with a 360° field-of-view that records distance measurements) in order to associate every point in the image with accurate distance measurements. A LiDAR frame and its corresponding camera image have been provided to you as `lidar-points.bin` and `image.png` respectively. The camera calibration matrix, K, is provided inside `K.txt`.

The LiDAR's frame is defined such that its X-axis points forward, Y-axis points to the left, and its Z-axis points upwards. And the camera's frame is defined such that its Z-axis points forward, X-axis points to the right, and Y-axis points downwards. The camera's center is found to be (via extrinsic calibration) 8 cm below, 6 cm to the left, and 27 cm in front of the LiDAR's center (as measured from the LiDAR). This is illustrated in the following image.

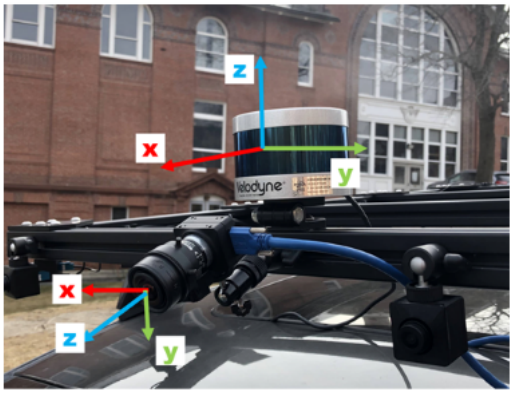

Figure 1: Frame descriptions.

Both the sensors are positioned such that the camera's Z-axis and the LiDAR's X-axis are perfectly parallel. Compute the transformation (R, t) required to transform points in the LiDAR's frame to the camera's frame. Give the transformation in both (a) homogeneous matrix form, and (b) XYZ Euler angles-translation form.

Then, using this computed transformation and the provided camera calibration matrix, project the LiDAR's points onto the image plane and visualize it as shown below. The color code (colormap) corresponds to the depth of the points in the image (color is optional, but it helps in debugging). Use `matplotlib` or any equivalent library for plotting the points on the image. Code for loading the LiDAR points in Python is provided.

Figure 2: LiDAR points projected onto the image and colored according to depth.

Q2) Drawing 3D bounding boxes

You've been provided with an image, also taken from a self-driving car, that shows another car in front. The camera has been placed on top of the car, 1.65 m from the ground, and assume the image plane is perfectly perpendicular to the ground. K is provided to you. Your task is to draw a 3D-bounding box around the car in front as shown. Your approach should be to place eight points in the 3D world such that they surround all the corners of the car, then project them onto the image, and connect the projected image points using lines. You might have to apply a small 5° rotation about the vertical axis to align the box perfectly. Rough dimensions of the car - h: 1.38 m, w: 1.51, l: 4.10. (*Hint:* Fix a point on the ground as your world origin.)

Figure 3: 3D bounding box projected onto the image.

Q3) How do AprilTags work

AprilTags are artificial landmarks that play an important role in robotics and augmented-reality. These are 2D QR-code like tags that are designed to be easily recognized. They are used in robotics for object recognition and accurate pose estimation where perception (from natural features) is not possible or is not the central focus. They are greatly robust to occlusion, warping, distortions, and lighting variations.

An image of a planar surface with two AprilTags stuck on it is provided. Your task is to estimate the pose of the camera using these tags.

You can do this by applying the Direct Linear Transform (DLT), like we saw in Zhang's method for camera calibration. The dimensions of each tag and their corner pixel locations are provided in

Figure 4: AprilTags attached to robots. The corners of the tags are used for pose estimation while the pattern carries within it a (unique) ID information that's used for recognition.

`april_tags_info.txt`. Using these locations and their corresponding world locations, estimate the 3×3 homography matrix, H, that maps these world points to their image locations. Verify the accuracy of your estimated H matrix by projecting the physical corner points (in world frame) onto the image plane, and visualize them along with the provided corner pixel locations.

[*Bonus*] Decompose your estimated homography matrix to compute the tag's position and orientation. **This** is a helpful resource for the same.

The End