Assignment 1 Report

Transformations, Camera Modelling and DLT

Puru Gupta(20171187) , Meher Shashwat Nigam(20171062)

Question 1 (python3 q1.py)

Given information :

- Points in the LIDAR frame (*lidar-points.bin*)
- K matrix of the camera (*K.txt*)
- Relative position of the camera wrt to LIDAR frame
- Original image:

To find :

- R,t transformation required to go from LIDAR frame to camera frame both in homogenous matrix form and XYZ Euler angles (RPY)-translation form.
- Project the LIDAR points onto the image plane and color code them according to depth.

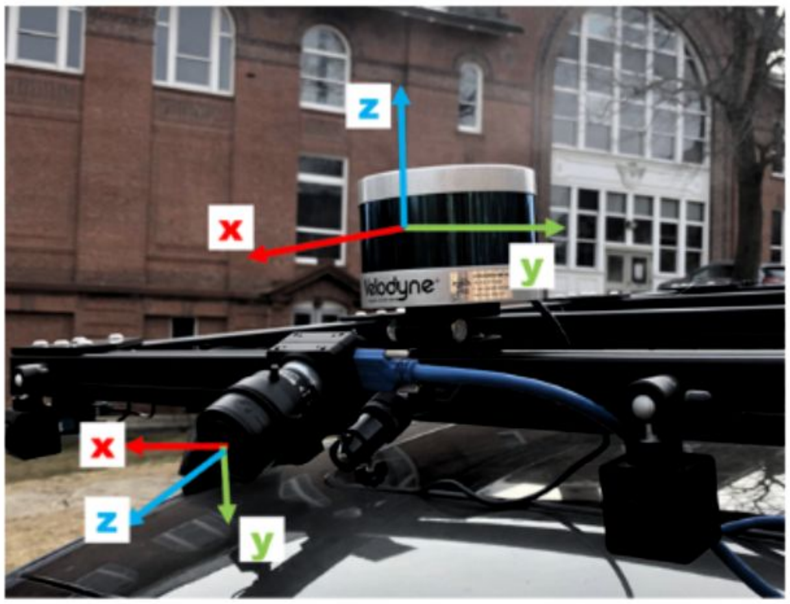


Figure 1: Frame descriptions.

The Lidar's Frame is defined as follows,

- X axis points forward
- Y axis points left
- Z axis points upward

The Camera's Frame is defined as follows,

- X axis points right
- Y axis points downward
- Z axis points forward

We first calculate the transformation required to go from camera frame to Lidar frame.
The transformation matrix is given as : $\begin{vmatrix} R_{LC} & t_{LC} \\ 0 & 1 \end{vmatrix}$

Where R_{LC} denotes the Rotation matrix to align Camera frame to LIDAR frame and t_{LC} denotes the translation required to bring them together.

Writing the vectors of camera frame in the LIDAR frame :

$$\mathbf{R} = \begin{vmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{vmatrix}$$

$$\mathbf{t} = \text{transpose}[\, 0.27 ,0.06 ,-0.08 \,]$$

The homogenous transformation matrix hence is T = [R| t] =

$$\begin{vmatrix} 0 & 0 & 1 & 0.27 \\ -1 & 0 & 0 & 0.06 \\ 0 & -1 & 0 & -0.08 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

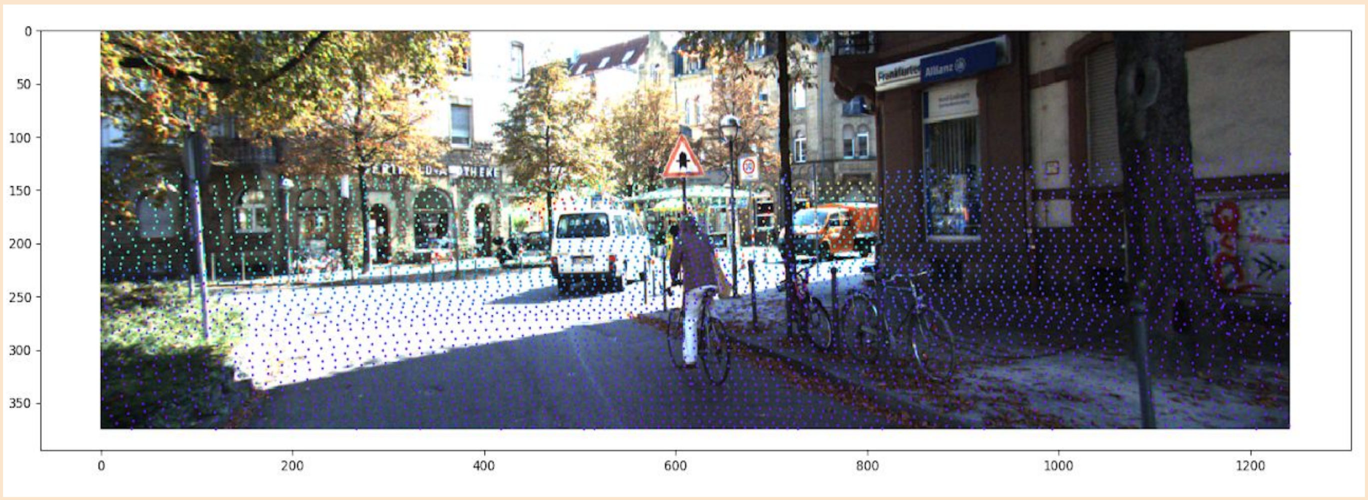
The transformation matrix to go from lidar frame to camera frame would be the inverse of the above matrix, given by inverse(T) :

$$\begin{vmatrix} 0 & -1 & 0 & 0.06 \\ 0 & 0 & -1 & -0.08 \\ 1 & 0 & 0 & -0.27 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Euler angles are given as (90, 0, 90).

Now, we project the Lidar points onto the image plane using the given camera calibration matrix K. Also we put limits on the points in the X and Y directions after projecting onto the image plane(according to the resolution of the given image) because the LIDAR has a larger field of view(360 degree) than the camera.

We colour code the points according to the depth(Z) values from the Lidar frame (Its X axis value). Output obtained is as shown :



The darker colours are points closer to the camera and lighter ones are farther away.

Question 2 (python3 q2.py < input_camera.txt)

The car's left bottom wheel is assumed to be the world origin. The camera has been placed on top of the car, 1.65 m from the ground, and the image plane is perfectly perpendicular to the ground. We have been given the K matrix of the camera.

Now in the camera frame(assuming Y axis points towards the ground and X towards the right), the left back wheel of the car has Y coordinate 1.65m.

Using camera matrix and the following calculations for calculating the camera coordinates :

The pixel values of the left rear wheel was found to be (827,310).

```
Y_cam = 1.65

Z_cam = (f_y * Y_cam)/ (310-o_y)

X_cam = ((827-o_x) * Z_cam)/ f_x
```

Where f_y, o_y, f_x, o_x come from the K matrix.

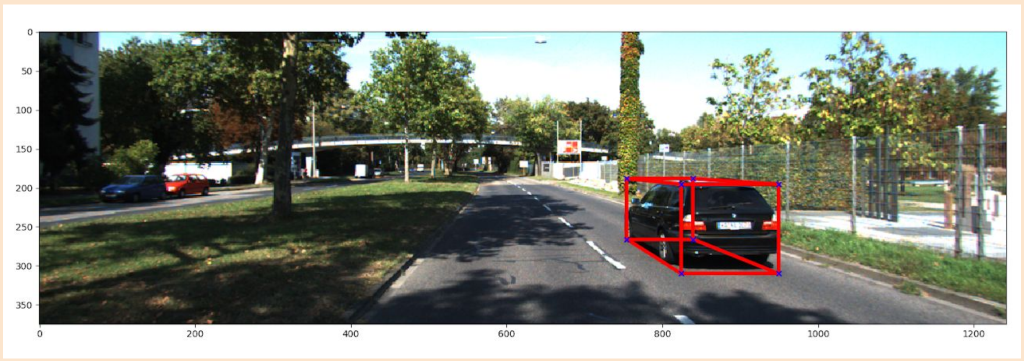
After getting this we accordingly calculate the the 8 points of the bounding box of the car from the given dimensions of the car(all points the camera frame).

Rough dimensions of the car - h: 1:38 m, w: 1:51, l: 4:10.

Now we rotate, by 5 degrees about the vertical axis.

Now we again use K matrix to get the pixel coordinates from the rotated box coordinates in camera frame, by projecting onto the image plane.

We plotted the box around the car after this, by joining the points.



Question 3 (python3 q3.py < input.txt)

First, we find the homography matrix corresponding to each point using DLT .While doing this we ignore the Z value as we do not have any depth information (all the points lie on the same plane).

After finding the homography matrix(3x3) we try to estimate its pose. To do this we first multiply H with :

inv(K) [inv(K)*H]

which gives us a 3x3 matrix.

Now we decompose this 3x3 matrix to find rotation 1 ,rotation 2 and translation of the camera(extrinsic parameters).

H1 = inv(K)*H

We divide H1 by the first column vector of H1. Then we take the first two columns of H1 and their cross product as r1, r2 and r3 respectively.

R = [h1, h2, h1 x h2]

We use SVD to decompose the R matrix and force eigenvalue diagonal to be

[1,1,det(U transpose(V))]

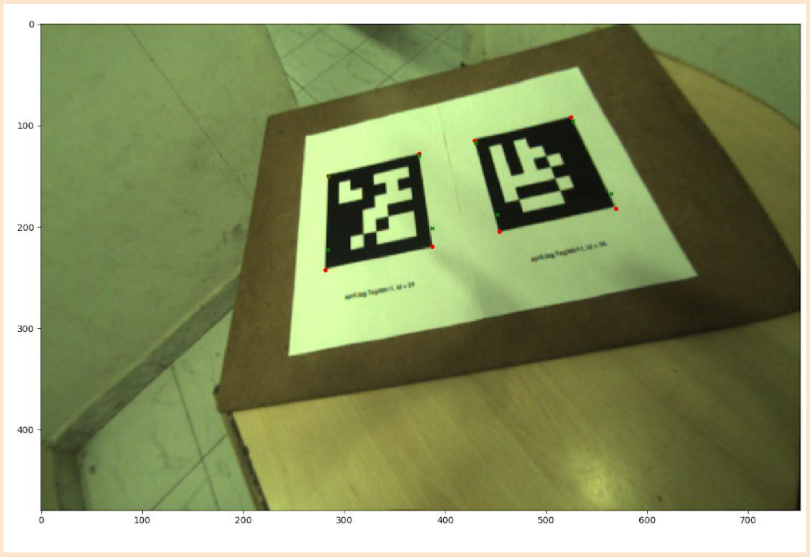
Here, U and V are the left and right eigenvectors of R obtained using SVD.

After multiplying the decomposed R matrix we choose the first 2 columns of R as r1 and

r2 of the camera and last column of H1 as T.

Decomposed H = K* [r1,r2,T]

In the following figure, the red circles come from the H matrix without decomposition and the green crosses come after application of R and T which we got after decomposition of the H matrix:



The H matrix:

```
[[ 6.87587769e-01 -3.85289232e-01  3.11674573e-01]
 [-1.99508989e-01  4.63629985e-01  1.63935259e-01]
 [-1.54782009e-04 -1.27844313e-03  1.09566694e-03]]
```

Decomposed [r1, r2, T] :

```
[[ 0.97668082  0.10754929  -0.11715101]
 [-0.19202733  0.82467682  -0.13673411]
 [-0.09602128 -0.55528487   0.58337134]]
```