AER 1515

Tushar Aggarwal
999356913

Assignment 1

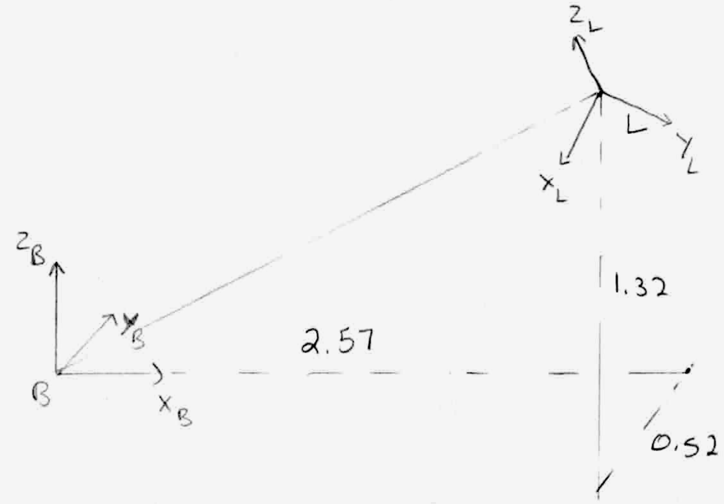
Question 1

$$T_{BL} = \begin{bmatrix} C_{BL} & r_B^{LB} \\ 0 & 1 \end{bmatrix}$$ where $C_{BL} \to$ rotation of L w.r.t. B

$r_B^{LB} \to$ vetor from B to L expressed in B.

Lidar Rotation follow Tait-Bryan extrinsic rotation: z, y, x

Axis	Angles (°)
z	-90
y	-23
x	-10



$$r_B^{LB} = \begin{bmatrix} 2.57 \\ -0.52 \\ 1.32 \end{bmatrix}$$

$$C_x(-10) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-10) & -\sin(10) \\ 0 & \sin(10) & \cos(10) \end{bmatrix}, \quad C_y(-23) = \begin{bmatrix} \cos(-23) & 0 & \sin(-23) \\ 0 & 1 & 0 \\ -\sin(-23) & 0 & \cos(-23) \end{bmatrix},$$

$$C_z = \begin{bmatrix} \cos(-90) & -\sin(-90) & 0 \\ \sin(-90) & \cos(-90) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C_{BL}(z,y,x) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.9205 & 0 & -0.3907 \\ 0 & 1 & 0 \\ 0.3907 & 0 & 0.9205 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.9848 & 0.1736 \\ 0 & -0.1736 & 0.9848 \end{bmatrix}$$

$$C_{BL}(z,y,z) = \begin{bmatrix} 0 & 0.985 & 0.174 \\ -0.920 & -0.068 & 0.384 \\ 0.391 & -0.160 & 0.906 \end{bmatrix}$$

$$T_{BL} = \begin{bmatrix} C_{BL} & r_B^{LB} \\ 0 & 1 \end{bmatrix}$$

$$T_{BL} = \begin{bmatrix} 0 & 0.985 & 0.174 & 2.57 \\ -0.920 & -0.068 & 0.384 & -0.52 \\ 0.391 & -0.160 & 0.906 & 1.32 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

<u>Question 1.2</u> lidar detected a feature at point f $r_L^{Lf} = [3.64, 8.30, 2.45]^T$
Express point in body frame.

So $r_B^{fB} = C_{BL} r_L^{fL} + r_B^{LB}$

$$= \begin{bmatrix} 0 & 0.985 & 0.174 \\ -0.920 & -0.068 & 0.384 \\ 0.391 & -0.160 & \cdot 0.906 \end{bmatrix} \begin{bmatrix} 3.64 \\ 8.30 \\ 2.45 \end{bmatrix} + \begin{bmatrix} 2.57 \\ -0.52 \\ 1.32 \end{bmatrix}$$

$$r_B^{fB} = \begin{pmatrix} 11.172 \\ -3.492 \\ 3.627 \end{pmatrix}$$

Ques1.3 $T_{LB} = \begin{bmatrix} 0 & -0.920 & 0.391 & -0.994 \\ 0.985 & -0.068 & -0.160 & -2.355 \\ 0.174 & 0.385 & 0.907 & -1.443 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$T_{BL} = \begin{bmatrix} 0 & 0.985 & 0.174 & 2.57 \\ -0.920 & -0.068 & 0.384 & -0.52 \\ 0.391 & -0.160 & 0.966 & 1.32 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

taking Inverse of T_{BL} i.e. T_{BL}^{-1} we get T_{LB}

$$\therefore \boxed{T_{LB} = T_{BL}^{-1}}$$

The inverse Translational matrix helps to restore a mapped point to its original position by reversing the translation & rotation.
Using car example
So if we know how a point is defined w.r.t. our body frame but we would like to find out where that point is w.r.t. lidar. we can use the inverse Transformation matrix T_{BL}^{-1} (or T_{LB}) to find that point w.r.t. lidar frame.

~~Anyo~~ Another example is maybe a robot arm ~~is~~ has grabbed the object w.r.t. to base & we switch the end gripper with a bigger one. knowing the new gripper's reverse Transformation matrix we can put a marker on an object from robot's body frame of reference to new gripper's frame of reference.

Ques 2. Camera Projections:

Parameters (Intrinsic)	
focal length (x)	959.79
" " (y)	956.93
Principal point (x)	696.02
Principal point	224.18

Distortion	Parameters
k_1	-0.369
k_2	0.197
k_3	1.35×10^{-3}
τ_1	5.68×10^{-4}
τ_2	-0.068

$$\begin{bmatrix} \gamma_C^{PC} \\ 1 \end{bmatrix} = T_{CB} T_{BL} \begin{bmatrix} \gamma_L^{PL} \\ 1 \end{bmatrix}$$

$$C_y(90^\circ) = \begin{bmatrix} \cos 90 & 0 & \sin 90 \\ 0 & 1 & 0 \\ -\sin 90 & 0 & \cos 90 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$\gamma_B^{CB} = [2.82, 0.11, 1.06]^T$$

2.1 $$T_{CB} = \begin{bmatrix} 0 & 0 & 1 & 2.82 \\ 0 & 1 & 0 & 0.11 \\ -1 & 0 & 0 & 1.06 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{BC} = T_{CB}^{-1} = \begin{bmatrix} 0 & 0 & -1 & 1.06 \\ 0 & 1 & 0 & -0.11 \\ 1 & 0 & 0 & -2.82 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ques 2.2 Point P in body frame = [4.47, -0.206, 0.731]

① Camera Frame Transformation

So point in camera frame is

$$\begin{bmatrix} r_c^{PC} \\ 1 \end{bmatrix} = T_{BC}^{-1} \begin{bmatrix} r_B^{PB} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & 1.06 \\ 0 & 1 & 0 & -0.11 \\ 1 & 0 & 0 & -2.82 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4.47 \\ -0.206 \\ 0.731 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} r_c^{PC} \\ 1 \end{bmatrix} = \begin{bmatrix} 0.329 \\ -0.316 \\ 1.65 \\ 1 \end{bmatrix}$$ where $r_c^{PC} = [0.329, -0.316, 1.65]$ (x, y, z)

Note: $\frac{x}{z} = \frac{0.329}{1.65} = 0.19939$

$\frac{y}{z} = \frac{-0.316}{1.65} = -0.191515$

Ideal pin hole camera

$$\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y/z \\ y/z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 959.79 & 0 & 696.02 \\ 0 & 956.93 & 224.18 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1993939 \\ -0.1915151 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \begin{bmatrix} 887.396 \\ 40.9134 \\ 1 \end{bmatrix}$$

③ Normalize Image Plane Projection

$$\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} 0.329/1.65 \\ -0.316/1.65 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1993939 \\ -0.1915151 \\ 1 \end{bmatrix}$$

I used this in calculations

③ Lens Distortion

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \overbrace{\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right)\begin{bmatrix} x_n \\ y_n \end{bmatrix}}^{\text{radial Distortion}} + \overbrace{\begin{bmatrix} 2\tau_1 x_n y_n + \tau_2(r^2 + 2x_n^2) \\ 2\tau_2 x_n y_n + \tau_1(r^2 + 2y_n^2) \end{bmatrix}}^{\text{Tangential Distortion}}$$

$$= \left(1 + r^2\left[k_1 + r^2\left[k_2 + k_3 r^2\right]\right]\right)\begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

where $r = \sqrt{x_n^2 + y_n^2}$

$$= \frac{\sqrt{0.329^2 + 0.316^2}}{1.65}$$

$$= \frac{\sqrt{0.208097}}{1.65}$$

$$r = 0.27647060606623$$

Radial Distortion

$$= \left(1 + r^2\left[k_1 + r^2\left[k_2 + k_3 r^2\right]\right]\right)\begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

$$\left(1 + (0.276470606)^2\left[-0.369 + (0.276470606)^2\left[0.197 + (0.276470606)^2(1.35 \times 10^{-3})\right]\right]\right)$$

$$= 0.97294668\,5 \begin{bmatrix} \frac{0.329}{1.65} \\ -\frac{0.316}{1.65} \end{bmatrix} = \begin{array}{c} \begin{bmatrix} 0.193999672 \\ -0.186334031 \end{bmatrix} \end{array} \begin{array}{l} B \\ C \end{array}$$

$\tau_1 - 0$

$\tau_2 = \cancel{0}e$

Tangential Distortion

$$\begin{bmatrix} 2 \times (5.68 \times 10^{-4})(0.193999672)(-0.186334031) + (-0.068)\left[(0.27640606)^2 + 2 \times (0.193999672)^2\right] \\ 2(-0.068)(0.193999672)(-0.186334031) + (5.68 \times 10^{-4})\left[(0.27640606)^2 + 2(-0.1863340\ldots\right. \end{bmatrix}$$

$$\begin{bmatrix} -4.106496 \times 10^{-5} - 0.010316126 \\ 4.916228792 \times 10^{-3} + 8.28579878 \times 10^{-5} \end{bmatrix} = \begin{bmatrix} -0.01035719 \\ 4.99908678 \times 10^{-3} \end{bmatrix}$$

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} 0.193999672 \\ -0.186334031 \end{bmatrix} +$$

$$\begin{bmatrix} 2 \times 5.68 \times 10^{-3} \dfrac{(0.329)}{1.65} \dfrac{(-0.316)}{1.65} + (-0.068)\left[(0.276470606)^2 + 2 \times \left(\dfrac{0.329}{1.65}\right)^2\right] \\ 2 \times (-0.068) \dfrac{(0.329)(-0.316)}{1.65 \times 1.65} + (5.68 \times 10^{-3})\left[(0.276470606)^2 + 2\left(\dfrac{-0.316}{1.65}\right)^2\right] \end{bmatrix}$$

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} 0.193999672 \\ -0.186334031 \end{bmatrix} + \begin{bmatrix} -0.110385318 \\ 0.006044246 \end{bmatrix} = \begin{bmatrix} 0.1829611401 \\ -0.180289785 \end{bmatrix}$$

(4) Pixel coordinate

$$\begin{bmatrix} X_s \\ Y_s \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 959.79 & 0 & 696.02 \\ 0 & 956.93 & 224.18 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1829611401 \\ -0.180289785 \\ 1 \end{bmatrix}$$

$$\boxed{\begin{bmatrix} X_s \\ Y_s \\ 1 \end{bmatrix} = \begin{bmatrix} 871.624 \\ 51.655 \\ 1 \end{bmatrix}} \longrightarrow \begin{array}{l} X_s = 871.624 \\ Y_s = 51.655 \end{array}$$

Q2.3

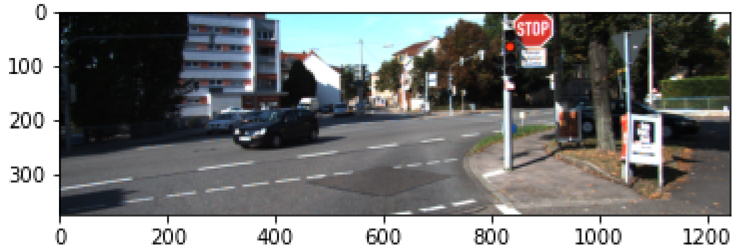
The object at (871.624, 51.655) is a stop sign.

In [1]:
```
#!/usr/env/bin python3

import matplotlib.pyplot as plt
import matplotlib.image as img
from matplotlib import cm

import numpy as np
import os
from math import sqrt

from utils import *
```

In [2]:
```
'''
Starter code for loading files, calibration data, and transformations
'''

# File paths
calib_dir = os.path.abspath('./data/calib')
image_dir = os.path.abspath('./data/image')
lidar_dir = os.path.abspath('./data/velodyne')
sample = '000000'

# Load the image
image_path = os.path.join(image_dir, sample + '.png')
image = img.imread(image_path)

# Load the LiDAR points
lidar_path = os.path.join(lidar_dir, sample + '.bin')
lidar_points = load_velo_points(lidar_path)

# Load the body to camera and body to LiDAR transforms
body_to_lidar_calib_path = os.path.join(calib_dir, 'calib_imu_to_velo.txt')
T_lidar_body = load_calib_rigid(body_to_lidar_calib_path)

# Load the camera calibration data
# Remember that when using the calibration data, there are 4 cameras with IDs
# 0 to 3. We will only consider images from camera 2.
lidar_to_cam_calib_path = os.path.join(calib_dir, 'calib_velo_to_cam.txt')
cam_to_cam_calib_path = os.path.join(calib_dir, 'calib_cam_to_cam.txt')
cam_calib = load_calib_cam_to_cam(lidar_to_cam_calib_path, cam_to_cam_calib_pa
th)
intrinsics = cam_calib['K_cam2']
T_cam2_lidar = cam_calib['T_cam2_velo']
```

In [3]:
```
#ntrinsics
```

In [4]:
```
plt.figure()
plt.imshow(image)
plt.show()
```



In [5]:
```
'''
For you to complete:
'''
# Part 1: Convert LiDAR points from LiDAR to body frame (for depths)
# Note that the LiDAR data is in the format (x, y, z, r) where x, y, and z are
# distances in metres and r is a reflectance value for the point which can be
# ignored. x is forward, y is left, and z is up. Depth can be calculated using
# d^2 = x^2 + y^2 + z^2
depth= []
point_xyz_list = []
lidar_body_points= []

for point in lidar_points:
    #depth Calculations
    point_xyz = point[:3]
    depth.append(sqrt(sum([x*x for x in point_xyz])))

    point_xyz = np.insert(point_xyz,3,1)
    point_xyz_list.append(point_xyz)

    # converting Lidar points from Lidar frame to Body Frame
    lidar_body_points.append(np.dot(T_lidar_body,point_xyz))
```

In [6]:
```
# Part 2: Convert LiDAR points from LiDAR to camera 2 frame
lidar_camera_points = []

for point in point_xyz_list:

    lidar_camera_point = np.dot(T_cam2_lidar,point)
    #print(lidar_camera_point)
    lidar_camera_points.append(lidar_camera_point)

# for more efficient code use list comprehension above

#print('Lidar to camera total points: ',len(lidar_camera_points))
#print('Total depth points: ',len(depth))
```

In [7]:
```
# Part 3: Project the points from the camera 2 frame to the image plane. You #
may assume no lens distortion in the image.
#Remember to filter out points where the projection does not lie within the im
age field, which is 1242x375.

points_in_image = []

for point in lidar_camera_points:

    #Normalize points and convert to camera frame
    point_in_image = np.dot(intrinsics,np.divide(point[:3],point[2]))
    points_in_image.append(point_in_image)

#print('Points in Image frame',len(points_in_image))
```

In [8]:
```
points_in_image_xy=[]
points_in_image_depth=[]

for i, point in enumerate(points_in_image):
    if (point[0]<=1242 and point[0]>=0):
        if (point[1]<=375 and point[1]>=0):
            #print("i value:",i," points: ",point)
            points_in_image_xy.append(point)
            points_in_image_depth.append(depth[i])

#print('Points in given image frame 1242X375: ',len(points_in_image_xy))
```

In [9]:
```
# Part 4: Overlay the points on the image with the appropriate depth values.
# Use a colormap to show the difference between points' depths and remember to
# include a colorbar.

x = []
y = []
for point in points_in_image_xy:

    x.append(float(point[0]))
    y.append(float(point[1]))

img_x_vector = np.array(x)
img_y_vector = np.array(y)
img_d_vector = np.array(points_in_image_depth)

plt.figure()

plt.scatter(img_x_vector,img_y_vector, s=1, c=img_d_vector, cmap = "viridis")
plt.colorbar()

plt.imshow(image)
plt.show()
```