Task 1:BEV projection

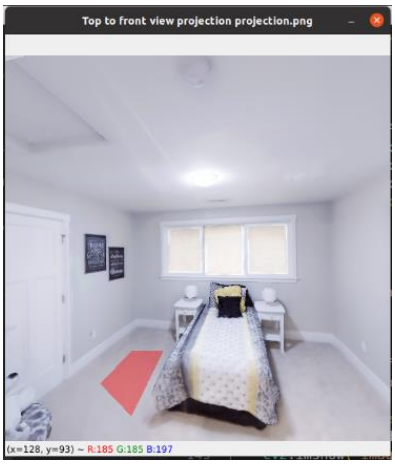
1. Because I have the fov angle in constant, so I can use the function below to obtain the focal length

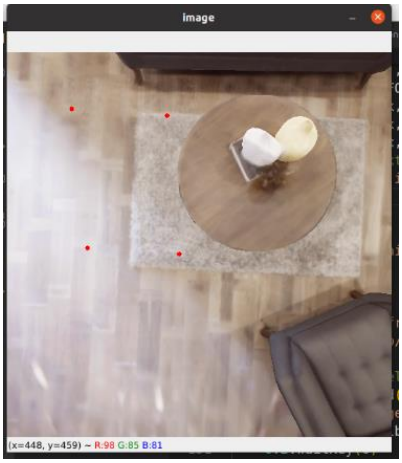
```
##先求出focal length
fov_theta = 90
f = (512/2)*(1/tan(np.radians(fov_theta/2)))
```

2. I use the transformation matric of rotation 90 degree by x to rotate from BEV to front image, and have a translation of z = -1.5 of BEV related to front image.
3. Use the relationship below to obtain the real BEV XYZ (the center of the image must notice), and use the transformation matrix to obtain the real XYZ after translation and rotation, and then use the relationship below to obtaion the front point on the image.

$$x = \begin{bmatrix} u \\ v \end{bmatrix} \quad \frac{u}{f} = \frac{X}{Z}$$

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \frac{v}{f} = \frac{Y}{Z}$$

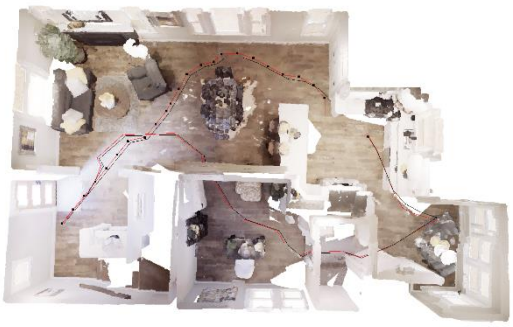




Task2:ICP alignment

1. Collect the data from load.py and store the number of the screenshot in number.txt
2. Use reconstruction.py and use the function(depth_image_to_point_cloud)to obtain the .pcd in every single screen shot by front image and bev image
3. Run the code below to implement the ICP, and store the point cloud data in a list
4. Because the ICP need the feature compare in the point cloud, so the scale of the point cloud cannot be to big, first, I update my point cloud io every loop, but when the point cloud become bigger, the ICP cannot work well because it cannot find the feature in the point cloud(scale is too big), so I change every single point cloud coordinate related to first image(use result_icp.transformation to get the point related to first image), and compare them in the current perspective then append them into a same list A, I work well and have a great reconstruction.
5. Trajectory of the estimated line, I use the function(estimated_camera_trajectory) to store the vector of the camera in a list. The ICP algorithm will get the result_icp.transformation, and the column 4 of this transformation matrix is the translation between the two point cloud. The coordinate of the point clouds are all in the first picture's coordinate, so I append the translation in a list and it represent the estimate trajectory about camera.
6. Trajectory of the ground truth line, I use the function (true_camera_trajectory)to get the load.py xyz(store in the Current_position.txt), and get the ground truth line.

Dataset:142



Dataset:70