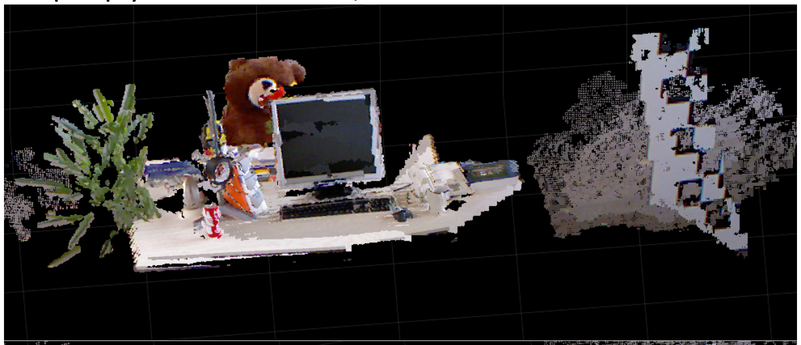
Readme

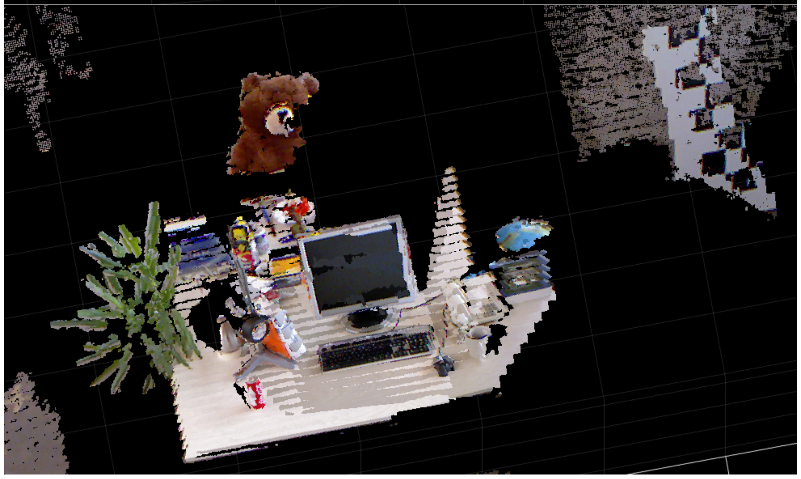
Libraries used : numpy,itertools for math and matrix operations. Cv2 and open3d for image reading and writing pointcloud

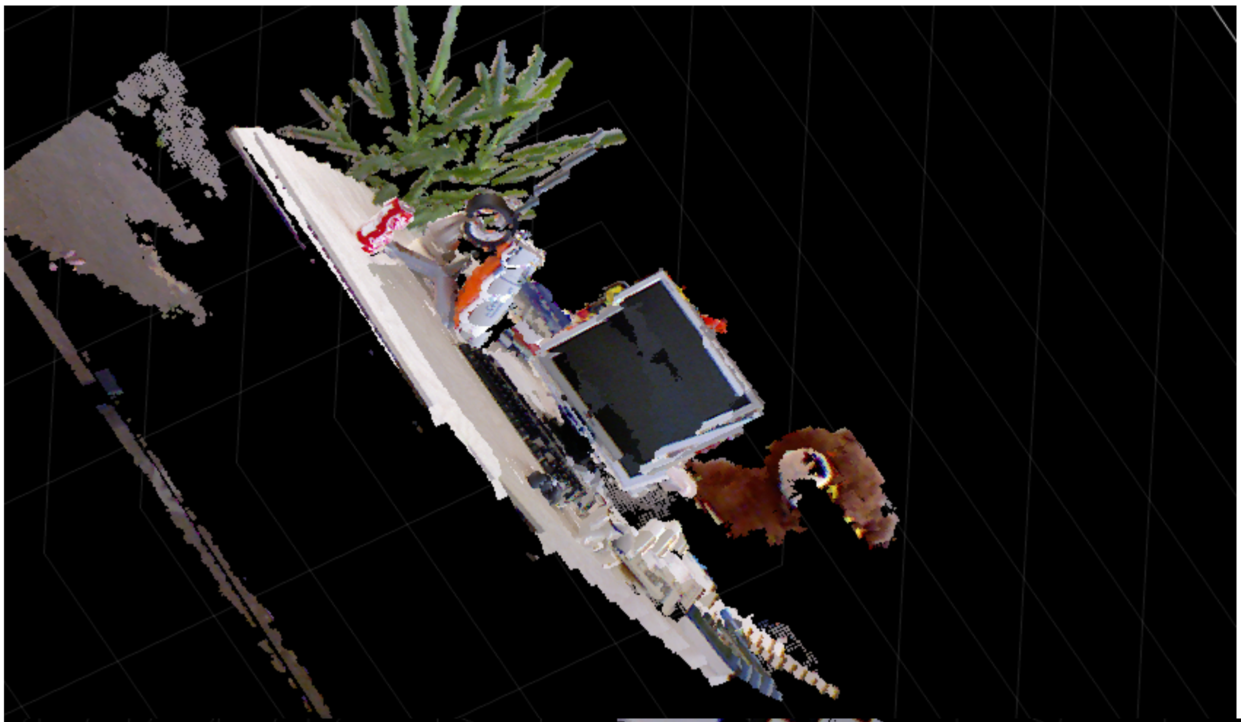
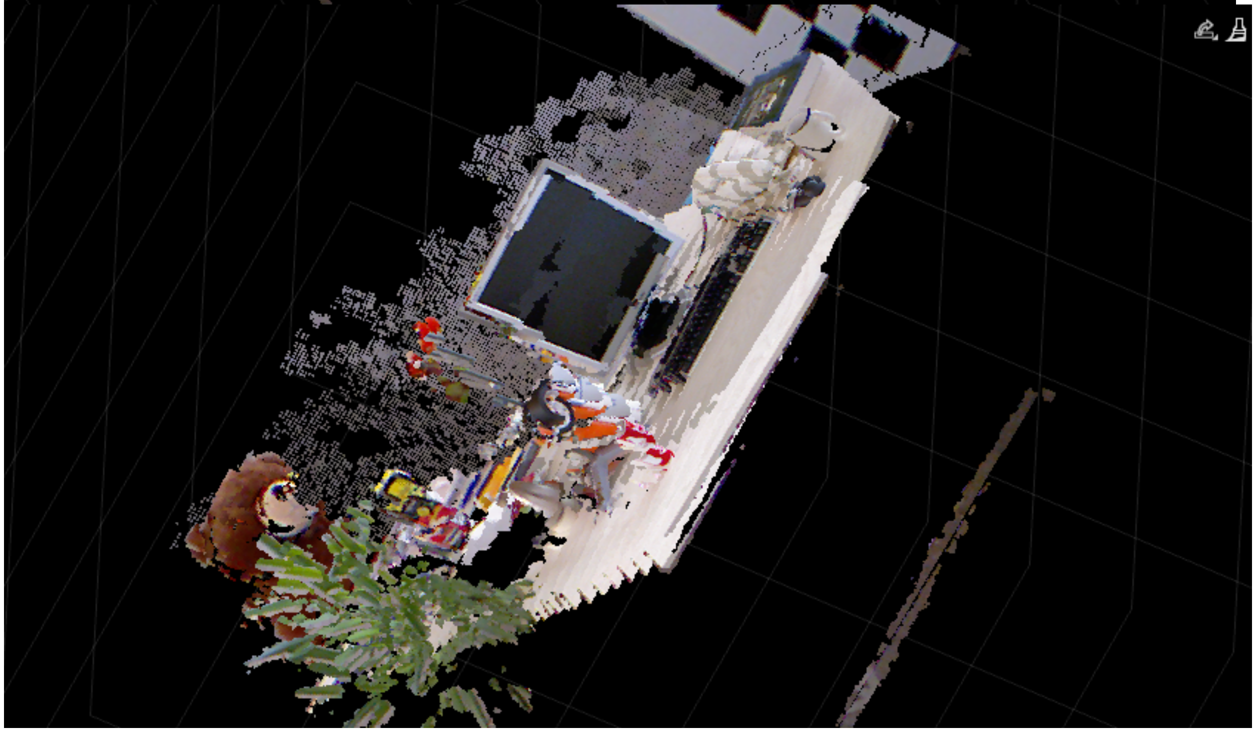
Problem 1 :
Code is in problem1.py

- Initialized constants, convolution kernels and intrinsic matrix.
- Part 1 is implemented in function detect_corners() which takes image as input and returns top 100 pixel indices with strongest response.
- Part 2 is implemented in function corners_to_3Dpoints() which takes as input indices of candidate corner pixels and image, and outputs a dictionary with indices as key and 3D coordinates as values.
- Part 3 is implemented in function match_corners() which takes as input 2 images to compare and their top 100 indices, and outputs top 10 matches with least SAD.
- Part 4 is implemented in estimate_pose() which takes as input the top 10 matches and 3d coordinates dictionary of the images of the matches, and outputs the Rotation and translation matrices between the 2 frames.R,t are calculated using RANSAC and are selected on the bases of SSD with 3-4 pixel error for at least 8 points out of 10.
- Part 5 is implemented in function to_3D() which takes as input rgb_image, depth_map and the rotation and translation matrices and gives the transformed 3d coordinates and their pixel color values.

Output ply is in submission, below are the screenshots :





Problem 2:
Code is in problem2.py

- Compute_3d takes depth map as input and outputs a dictionary with the 3D world coordinates as values and indices as keys.
- get_normal takes a list of 3d points as input and outputs the best normal vector between the points, this is done using covariance matrix eigen decomposition.
- surf_norm() takes depthmap, point dictionary and kernel_size as input and outputs the normal_map of the image.

Output image is in submission, below is the screenshot :