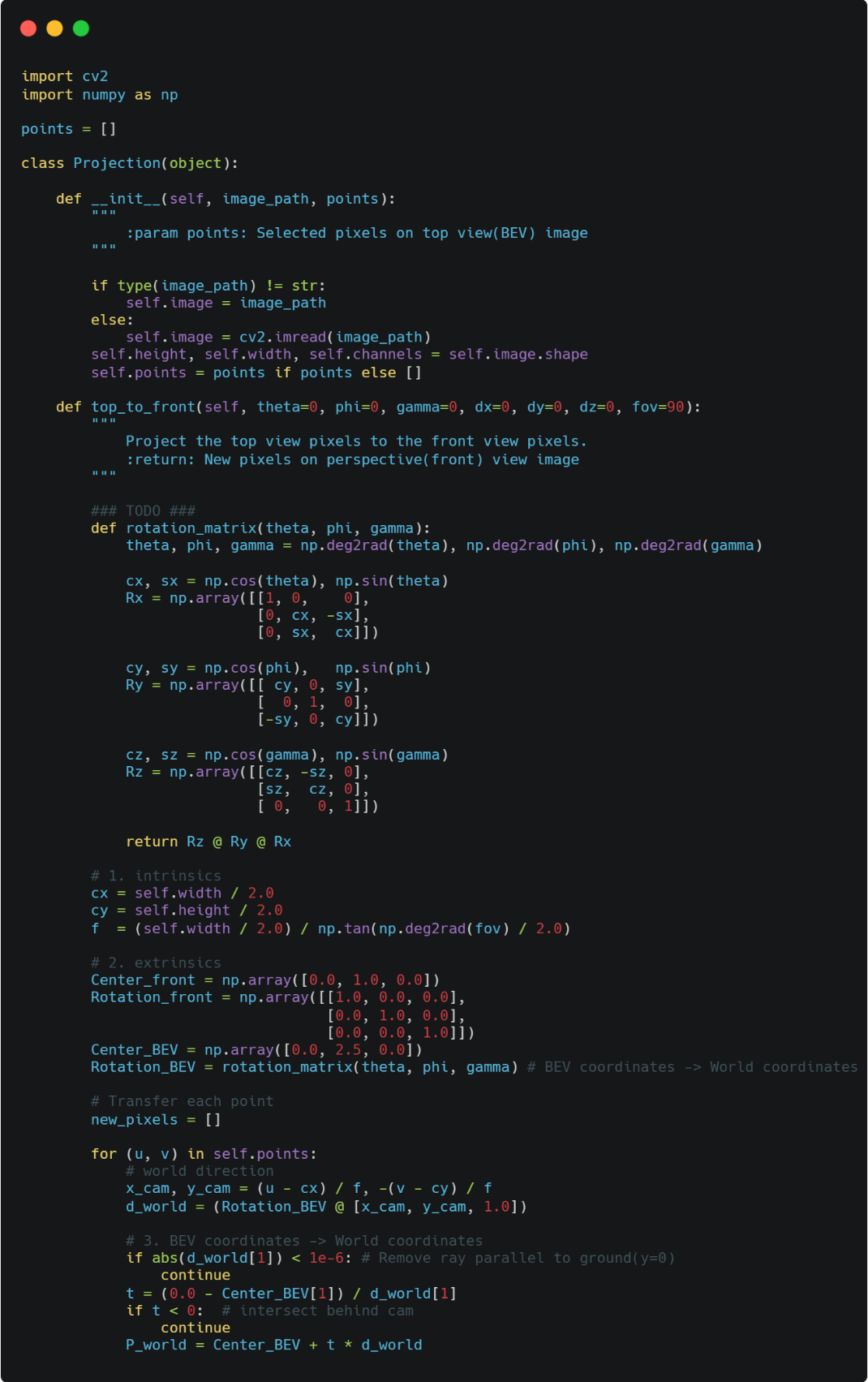
Physical AI Homework 1:

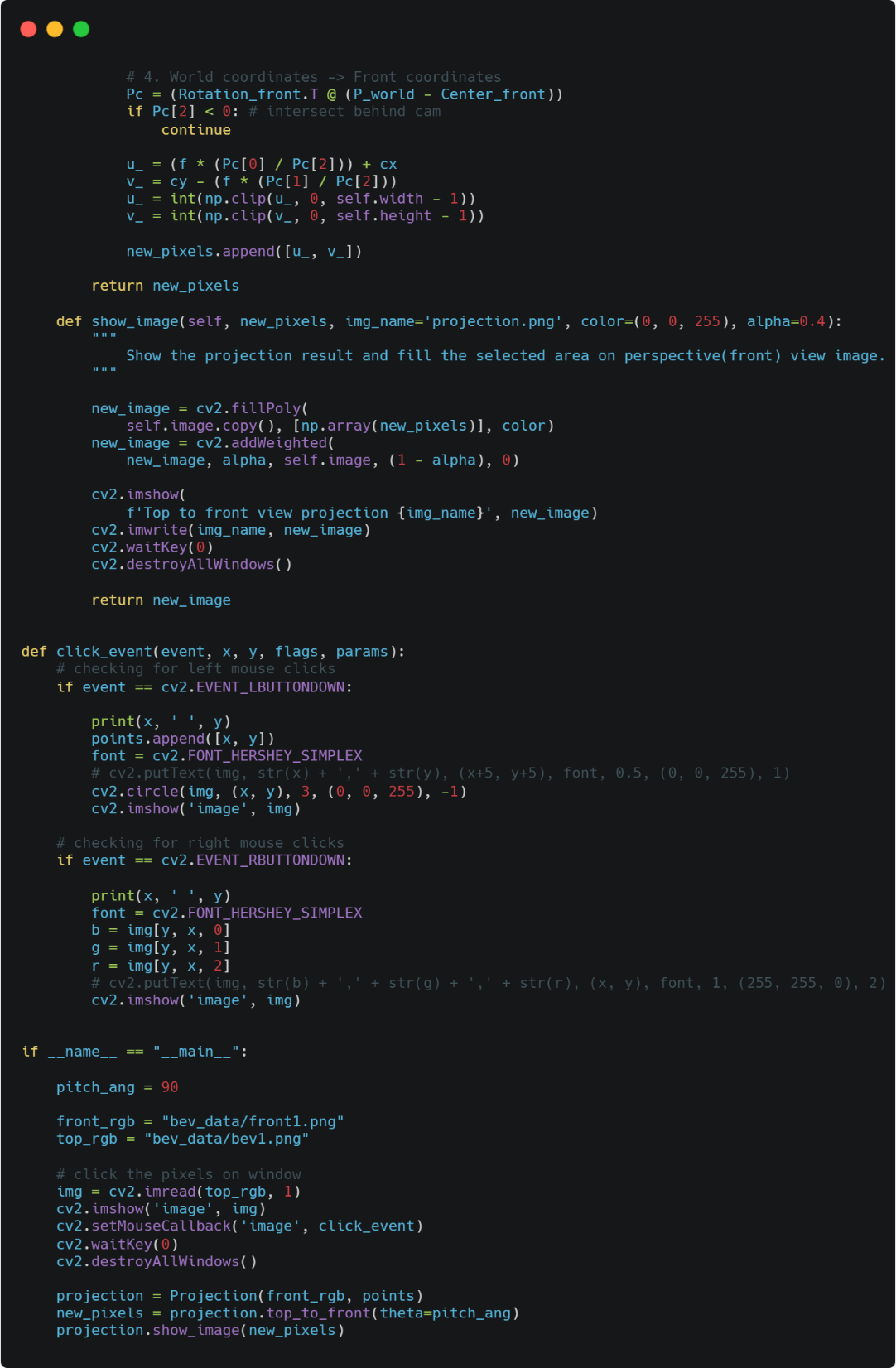
BEV projection and 3D Scene Reconstruction

111550132 張家睿

1. Implementation

Task1

* 1. Code

**\_\_init\_\_():** Save the points to self.points.

**top\_to\_front():** Map user-clicked BEV pixels to their corresponding pixels in the front-view image. First, derive intrinsics from image size and FOV. Second, define camera poses with rotation\_matrix(), a function that build rotation from Euler angles. Third, for each point, transfer the BEV pixels to BEV coordinates with (u - cx) / f and -(v - cy) / f. Transfer BEV coordinates to world coordinates with a ray Center\_BEV + t \* d\_world with ground (y = 0), Remove the point parallel to ground and behind camera. Forth, transfer world point to front coordinates with Rotation\_front.T @ (P\_world - Center\_front). Transfer front coordinates to front pixels f \* (Pc.x / Pc.z) + cx and cy - f \* (Pc.y / Pc.z), then clip the result and save the point to array that will return.

**\_\_name\_\_ == "\_\_main\_\_":** Change pitch\_ang to 90.

* 1. Result and Discussion
     1. Result of your projection (2 different pairs). Like the example result above.

bev1.png front1.png b ev2.png front2.png

* + 1. Anything you want to discuss

Will there be any deviation when approaching the edge? Why?

The projection of the table is offset.

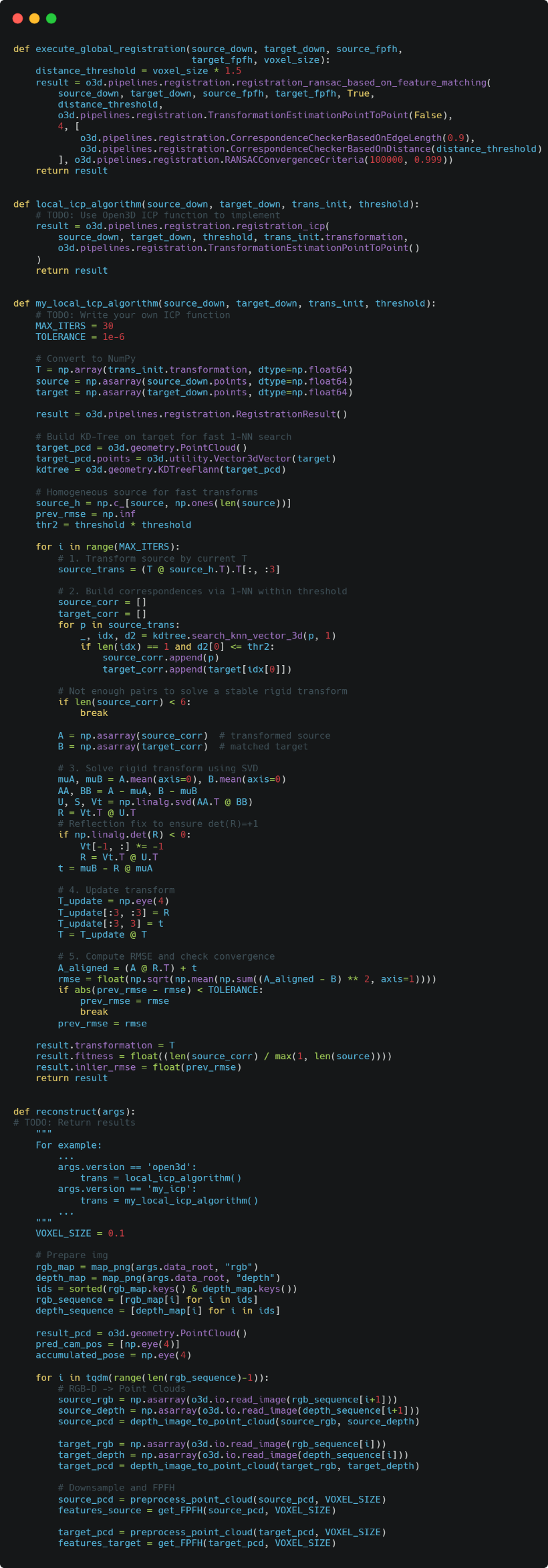
Yes, when the deviation is amplified near the image edges because the viewing angle becomes steep there; the denominator dy becomes small, leading to a magnification effect.

* + 1. Any reference you take

None

Task2

* 1. Code



**map\_png():** Scans a folder contains \*.png files. Returns a dict mapping the integer frame id to its filepath. This lets you align RGB and depth by frame number.

**depth\_image\_to\_point\_cloud():** Turns one RGB–Depth pair into a 3D point cloud using Open3D’s RGBD pipeline. Builds pinhole intrinsic from image size and a 90° FOV; converts depth (0–255) linearly to meters (0–10 m), then creates an RGBD Image and a PointCloud. Assumes the /255\*10 scaling puts depth in meters (depth\_scale=1.0 m).

**preprocess\_point\_cloud():** Voxel-downsamples the input point cloud with the given voxel\_size to reduce point count and speed up registration steps.

**get\_FPFH():** Estimates normals (radius = 2×voxel\_size, max\_nn=30) and computes FPFH descriptors (radius = 5×voxel\_size, max\_nn=100) on the point cloud. Returns an Open3D Feature used for global matching, so that we can use RANSAC in execute\_global\_registration().

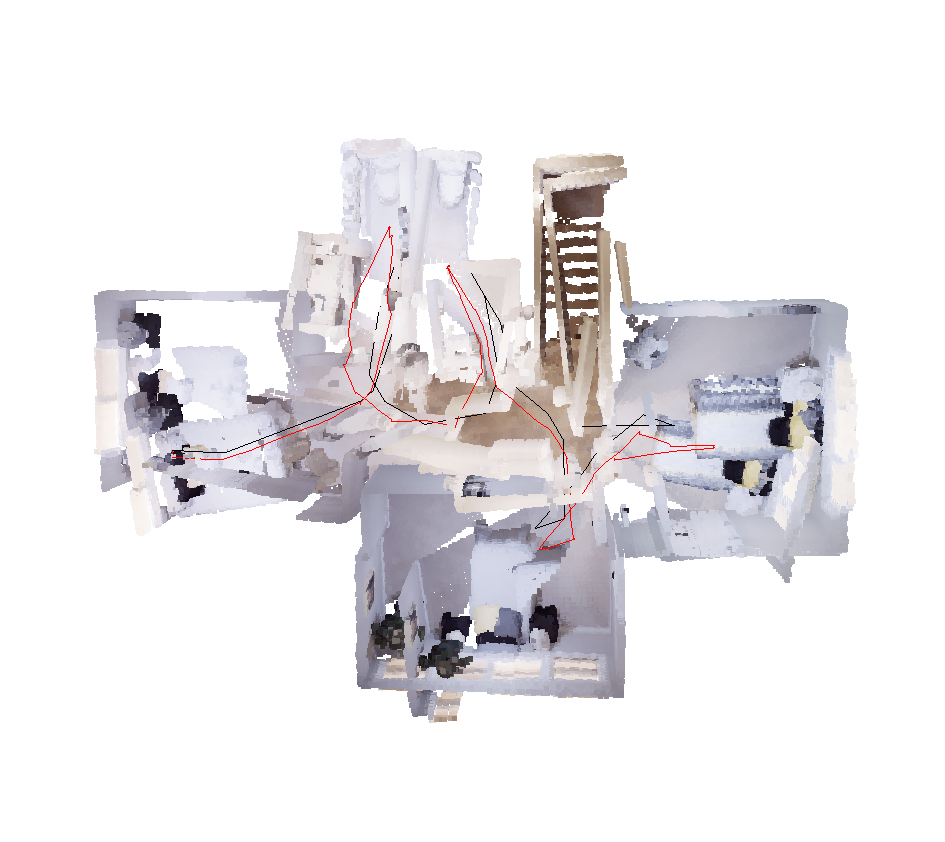
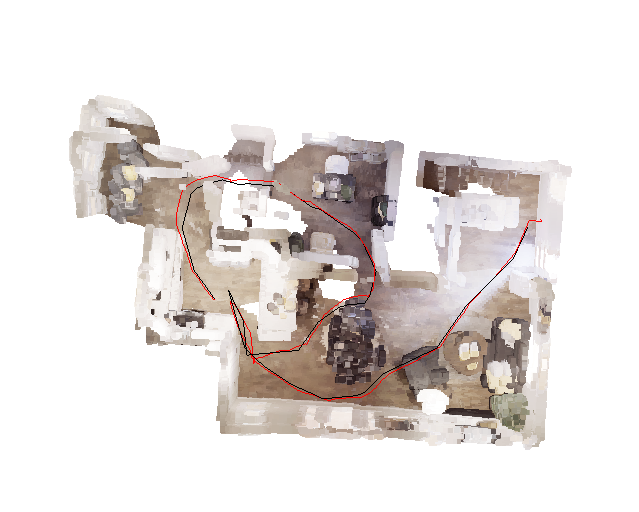
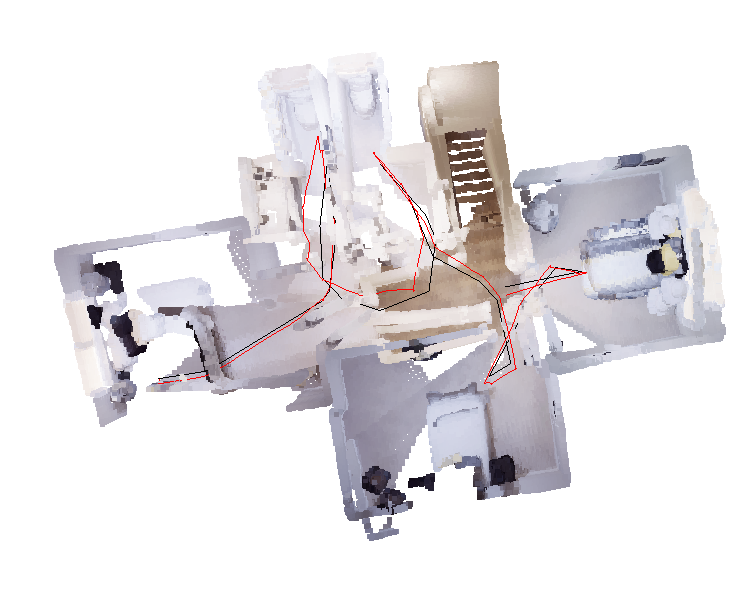
**execute\_global\_registration():** Performs RANSAC global registration using the source & target FPFH features. Uses edge-length and distance checkers, ransac\_n=4, and a correspondence threshold=1.5×voxel\_size; returns a coarse RegistrationResult transform.

**local\_icp\_algorithm():** Refines alignment with Open3D ICP (point-to-point) starting from the RANSAC initial transform and a given distance threshold. Returns an Open3D RegistrationResult with the refined transform.

**my\_local\_icp\_algorithm():** Custom point-to-point ICP. Takes downsampled source\_down & target\_down, an initial transform trans\_init, and a max correspondence distance threshold. Internally it: (1) Resolves the initial 4×4 T. (2) Builds a KD-Tree on the target to speed up.(Complexity per iteration is O(NlogM)) (3) Iterates up to 30 times: transform source by T, for each transformed source point, find its 1-NN in target and keep pairs within threshold. If fewer than 6 pairs, stop. (4) Solve the rigid motion with SVD (Umeyama, no scale): R = VtT @ UT and t = - R @ . (5) Form an incremental transform 𝑇update=[[R, t], [0, 1]] and left-multiply T = T\_update @ T. (6) Compute RMSE = and stop when improvement < 1e-6. Returns an Open3D-style RegistrationResult with transformation = T, fitness = (# inlier matches / # source points), and inlier\_rmse = final RMSE. Complexity per iteration is O(NlogM) due to KD-Tree.

**reconstruct():** Full incremental reconstruction over the RGB-D sequence. Set VOXEL\_SIZE = 0.1. For each consecutive pair: makes point clouds, downsamples, computes FPFH, runs RANSAC for a coarse pose, refines with ICP (Open3D or custom), chains the relative pose into a global accumulated\_pose, transforms the source cloud into a global point cloud, and records the camera poses. Returns result\_pcd & pred\_cam\_pos.

**remove\_ceiling():** Filters the point cloud along the Y axis to remove ceiling points for clearer room visualization. Uses a threshold set as starting\_height - offset and keeps points with y > threshold, returning the filtered cloud.

* 1. Result and Discussion
     1.  Result of your reconstruction (Floor 1 and Floor2, Both open3d implementation and your own implementation) Make sure your execution time for each reconstruction less than 5 mins, or you will get 0 point in this part.

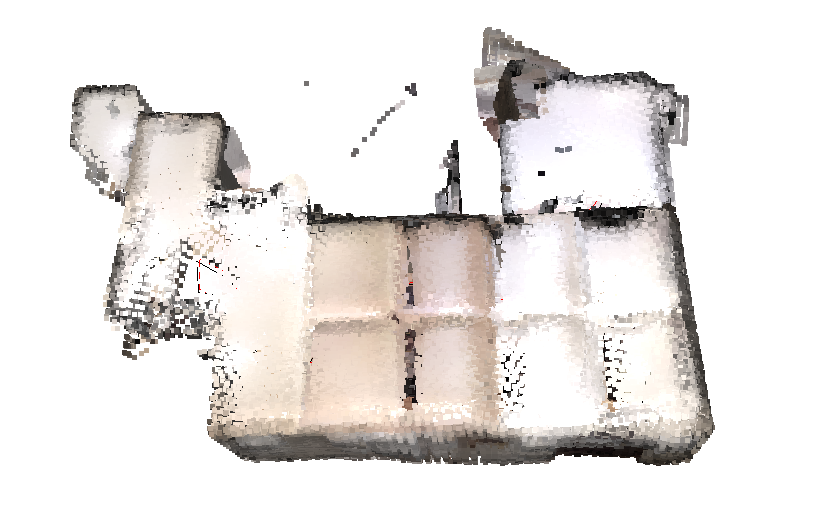
1F/open3d 1F/my\_own 2F/open3d 2F/my\_own

* + 1. Mean L2 distance between ground truth and estimated trajectory.

|  |  |  |
| --- | --- | --- |
|  | open3d | my\_local\_icp\_algorithm |
| 1F Mean L2 distance | 0.3563992774349988 | 0.3428335922227827 |
| 2F Mean L2 distance | 0.24607247247257658 | 0.37385491764591783 |

* + 1. Anything you want to discuss, such as comparing the performance of two implementations.

(Discovered in the process) Why does the result look weird in visualization, but it has normal L2 distance?

Because the point cloud includes the ceiling, I need to remove the points that are too high.

* + 1. Any reference you take

<https://www.open3d.org/docs/release/tutorial/pipelines/global_registration.html>

1. Questions
   1. What’s the meaning of extrinsic matrix and intrinsic matrix?

**Extrinsic matrix:** camera pose, rotation R and translation t, that maps world coordinates to the camera frame.

= R \* + t

Intrinsic matrix: camera’s internal parameters that map camera-frame 3D points to pixels—mainly focal length fx, fy. and principal point (cx, cy).

* 1. Have you ever tried to do ICP alignment without global registration, i.e. RANSAC? How’s the performance? Explain the reason. (Hint: The limitation of ICP alignment)

Yes, the result is horrible. If you remove RANSAC and only use ICP, it is easy to get stuck in local extreme values ​​or diverge directly, causing the entire map to be messed up.

* 1. Describe the tricks you apply to improve your ICP alignment.

(1) Strong initialization from FPFH+RANSAC, which puts ICP inside its small basin of convergence. (2) Distance-gated 1-NN correspondences via KD-Tree, which both speed up matching and reject outliers beyond a threshold.

(3) Stable SVD rigid solve plus RMSE early stopping, ensuring numerically sound updates and halting once improvements are negligible to avoid overfitting.