Point Cloud Room Estimation for 3D Mesh Placement

Anand Bhavsar
Stanford Center of Professional Development
Stanford University
anandb0@stanford.edu

Abstract

Point clouds can help us easily determine different components within a room. In this paper, we demonstrate a RANSAC heuristic model to detect floors and ceilings by using an RGB and depth image, applying plane segmentation to the generated point cloud. We apply these predictions to place a 3D mesh chair within the room, using our results to scale and align our chair object. We do so by identifying free space within the floor's point cloud, then translating the chair to that location. Results show that our model is capable of identifying empty floor space sufficient to place a chair mesh, in a variety of rooms, given adequate conditions. On the other hand, cluttered rooms and large horizontal objects, such as tables, pose a challenge for the heuristics used, and options to improve accuracy are discussed, as well as performance improvements to search point clouds efficiently.

1. Introduction

1.1. Estimation

Using an RGB-D image, the intrinsic parameters of a camera, and the extrinsic parameters of an image, room layouts and interiors can be estimated as a point cloud, which can be used to estimate the locations of floors and ceilings [17]. While the true layout cannot be completely reconstructed with only one image and a depth map, due to occlusions, the three dimensional locations of objects within the room can be calculated with the camera parameters, depth, and image points.

Since methods such as RANSAC plane segmentation rely on distance between points to cluster them together, the idea is that large, continuous objects such as walls, floors, and ceilings will be easy to detect. The challenge comes from clutter in the room, which can make clustering points using a methodology such as RANSAC difficult, and different camera orientations can complicate angle calculation to isolate the floors and ceilings. [15] Enough clutter can result in the RANSAC algorithm detecting vertical planes that segment the floors and ceilings, failing to detect the shapes desired. Large horizontal objects in a room may also be detected as false positives, often occluding the floors or ceilings desired.

1.2. Placement

If a floor has been identified, the model uses a straightforward algorithm to determine if there's space for the chair mesh. Using the bounding boxes of the floor and the chair [13], the model searches the points in the floor for a position to place the chair, if possible. If ceiling data is provided, the chair can be scaled as per assumptions of standard room heights and chair heights. Then, the chair is rotated and translated to its final location [11].

A challenge comes from the structure of the point clouds, which may not preserve the shape of the original image array, which requires a technique to locate the points involved in possible collisions. This may require lots of computational power to go through each of the point cloud's points, which will make the application highly inefficient. Different images may require that a chair not be placed, since there may be no floor, or no space available on the floor. However, other horizontal objects might still be present.

Another challenge comes from the tilt of the room, requiring the chair object to be tilted similarly, so that it is parallel to the floor. This requires the accelerometer data as well to calculate the angles by which the chair should be rotated.

2. Background/Related Work

This section provides related works to solving room estimation problems using similar techniques. Some of them go over more advanced methods that could be applied in the future for solving this problem. Other works explore useful techniques to understand and evaluate point cloud results, comparing the results with existing techniques.

Figure 1. Original RGB Images.

2.1. Related Work

Anagnostopoulos et al. provides an example of detecting walls, floors, and ceilings in point cloud data for rooms as well as large buildings, using RANSACs and planes parallel to the yz and xz planes, and then it applies octree division [17]. It relies on the Manhattan-World heuristic, assuming that buildings will have mutually orthogonal directions, horizontal floors and ceilings, and vertical walls. Segments are extracted in order based on number of points, and normal vectors are computed to determine orientation. They finally use thresholds of point cloud density to extract floors and ceilings.

Bozpolat et al. uses RANSAC to perform plane segmentation on Kinect point clouds. [15] These point clouds are calibrated using the Kinect's intrinsic parameters. They remove noise and and down-sample the data for faster processing times. These processing steps are essential to yield better results in less execution time.

Cao et al. estimates poses in point clouds uses trained RANSAC hypotheses based on evaluating metrics, such as point cloud distance. [12] They notice that correspondence based evaluation metrics have been overlooked, which they develop to increase performance. The results are compared with the standard high density evaluation methods, maintaining accuracy and reducing runtime.

Lin et al. develops a new method of point cloud evaluation, called Density Chamfer Distance. [16] They notice that Chamfer distance and Earth Mover's distance are common methods used, explore the problems with those methods, and develops their new method derived off of Chamfer distance. They then compare the quality of results with their new method versus the standard Chamfer and Earth Mover's Distance.

Figure 2. Original Depth Images.

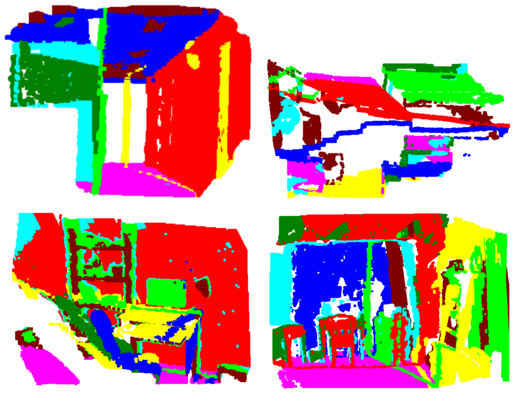
Figure 3. RANSAC Plane Segementation.

3. Approach

3.1. Procedure

First, the original RGB and depth images are combined to create an RGBD image, shown in Figure 1. This is used with the intrinsic and extrinsic camera parameters to generate a point cloud for each image, as in Figure 2. Estimates can be calculated once all the point clouds are created.

The model retrieves all the inliers from running RANSAC plane segmentation, using Open 3D, shown in Figure 3. The model imports the accelerometer data to get the tilt angle for each image [6]. With a threshold of 0.1 radians, or 84-95 degrees, it calculates the absolute values of each plane's angles. RANSAC delivers the plane equations in the form:

$$Ax + By + Cz + D = 0 \tag{1}$$

The normal vector is simply (A, B, C). The model calculates its projection in the xy and yz planes, since y represents the height axis in the point clouds, then subtracts the vector by its projection.

$$distances = a - \frac{a \cdot v}{v^2} v \tag{2}$$

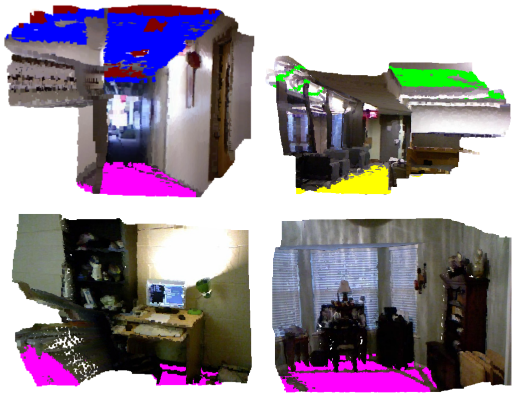
Figure 4. All Horizontal Planes

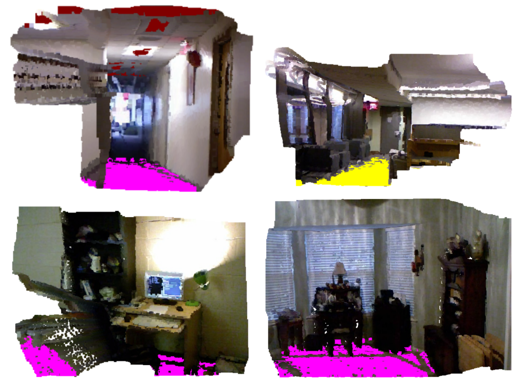
Figure 5. Detected Floors and Ceilings

Where v is either (0, 0, 1) (the xy plane) or (1, 0, 0) (the yz plane), and a is the normal vector of the plane. Arctangent of y/x and y/z is calculated to get the respective angles in radians. If both angles are within the threshold, the model appends it to a list of horizontal planes which it returns. In order to ensure that the yz angle fits within the threshold, it is calculated as:

$$angle_{yz} = arctan(y/z) - \frac{tiltAngle}{2} \tag{3}$$

Now, these horizontal planes in Figure 4 need to be examined to determine which one is the floor and which one is the ceiling. The model uses the median of the image, the middle of the image's height. The lowest plane below the median is considered the floor and the highest plane above the median is considered the ceiling. The model uses Open3D to remove any statistical outliers from the floor and/or ceiling point clouds, with 20 neighbors required and a standard deviation of 2.0. The model returns the floor and/or ceiling point clouds, and the corresponding angles of their normal vectors [7].

Some sample results are shown in Figure 5. With the floor and ceiling point clouds obtained, the model can now attempt to place the mesh, using Mesh Lab [11] and Pymeshlab[13]. The model checks if the floor is available, and if so, it calculates the bounding boxes of the floor and the chair. It acquires the height of the chair from this bounding box. If the ceiling is also available, the model calculates the room height in pixels using the ceiling and floor's bounding boxes. The chair is then scaled based on the room height, with the assumption that the room is 2.5 meters tall and the chair is 1 meter tall. The chair is also rotated based on the angles of the floor's plane [3]:

$$x_{angle} = -\frac{\pi}{2} + x_{floorAngle} \tag{4}$$

$$y_{angle} = -\frac{2\pi}{3} \tag{5}$$

$$z_{angle} = \frac{3\pi}{2} - z_{floorAngle} \tag{6}$$

Since the y angle is not associated with the angles of the floors, it can theoretically be any value. $-60°$ was chosen for the chair object file used.

The chair's bounding box is recalculated to match the scaling and rotation. The model loads the floor's point cloud and checks every point (x,y,z) on it for another candidate point (a,b,c) that can accommodate the area of the chair:

$$a - x > length_{chair} \tag{7}$$

$$c - z > width_{chair} \tag{8}$$

To make sure there are no obstacles in this area found, the model checks the full image point cloud for any points (i,j,k) in that fall between (x,z) and (a,c). For each of these points, the model makes sure they don't fall within the chair's space:

$$max(y, b) <= j <= max(y, b) + height_{chair} \tag{9}$$

The model only checks the first point that is far enough away to fit the chair, starting over after (x,y,z). This remains accurate because the most important area covered remains the same, the area of the chair's bounding box. While the time complexity remains the same, this modification significantly reduces the runtime for average images where the floor is larger than the bottom of the chair's bounding box, since it's likely that there will be space for the chair in the front of the point cloud.

Once the candidate points have been validated, the chair is translated from its original position (c_x, c_y, c_z) to its new position (t_x, t_y, t_z). This is simply calculated as:

$$(c_x - t_x, c_y - t_y, c_z - t_z) \tag{10}$$

With the scaling, rotation, and translation done, the chair can now be placed within the 3D point cloud, and is viewable via Mesh Lab [11] or any other 3D rendering software.

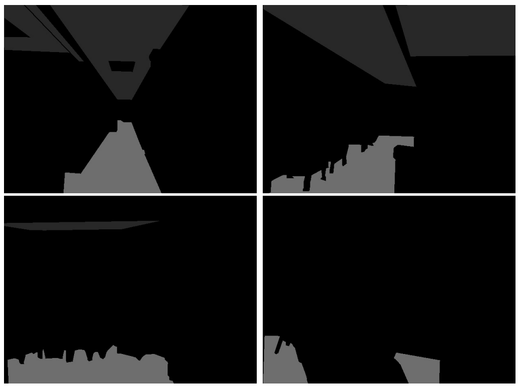
Figure 6. Examples of Labels Used

A new chair object file is created and saved, to avoid mixing it between different mlp files. If a candidate point could not be found, then the model reports that there was no space for the chair in the room and does not generate an mlp or obj file.

4. Experiment

For the experiment, I collected estimates for all the floors and ceilings, then compared the estimates with the original labels to measure the accuracy of the results. Figure 6 shows a sample set of labels. I looked for any false positives or negatives of the floors and ceilings, as well as the number that were correct. In order to determine the accuracy of the floors and ceilings detected, I also calculated the Chamfer distance for each set of images and labels, when possible.

For the second part of the experiment, I used Meshlab to attempt to place a chair in the images. I evaluated as many of them as I could manually, by examining them to determine if the chair placement was valid. I also checked samples of photos that did not result in a chair placement to determine if there really was no space for the chair.

4.1. Dataset

For testing purposes, I have found and downloaded the chair linked here. [2]. This paper will be using the labeled data from the NYU Depth Data-set V2.[14] It contains 1449 images and depth maps, with labels for objects, accelerometer data, and the intrinsic and extrinsic camera parameters. The original data arrived in a hdf5 file, which was a compressed format that stored the images and numerical information. This had to be extracted correctly to acquire the RGB images and depth maps.[10, 9] In order to access the depth data, I pre-processed the depth values, scaling them from 0 to 255 to save as an image for each corresponding RGB image. In the Open3D library, the RGB image is combined with the depth image to make an RGB-D image[17]. In order to view the label data, I removed all the points corresponding to labels other than the floors and ceilings [4]

$$\begin{bmatrix} 5.19e+02 & 0 & 3.26e+02 \\ 0 & 5.19e+02 & 2.54e+02 \\ 0 & 0 & 1 \end{bmatrix}$$
Figure 7. RGB Intrinsic Parameters

$$\begin{bmatrix} 1 & 5.05e-3 & 4.30e-3 & 2.50e-2 \\ -5.04e-3 & 1 & -3.69e-3 & -2.93e-4 \\ -4.32e-3 & 3.67e-3 & 1 & 6.62e-4 \end{bmatrix}$$
Figure 8. Extrinsic Parameters

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
Figure 9. Point Cloud Rotation

and scaled their color values to make them more discrete. All this image data was cropped by 8 pixels on each side to remove the borders. These borders had been interfering with the RANSAC estimates, so removing them helped improve accuracy.

I have used the RGB intrinsic parameters in Figure 7. My extrinsic parameters are in Figure 8. The point cloud was derived using the Open3D library [17]. Then I transformed the resulting point cloud by the matrix in Figure 9 to orient it correctly.

4.2. Evaluation

The accuracy of detection can be determined by whether a floor was detected when a floor label was present, as well as whether a ceiling was detected when a ceiling label is present. I collected false negatives and positives for each image to determine the overall accuracy, saving this information for statistical use [8].

The quality of detected results is evaluated through Chamfer distance between the estimated versus the labeled point clouds. I have calculated the English names of each of the labeled codes [1]. Through the labeled dataset, I have identified the label '4' as referring to the ceiling and '11' as referring to the floor objects. For each image, a floor and ceiling point cloud can be constructed by only using the points corresponding to these labels [17]. Figure 6 shows examples of what these labels look like once they've been properly prepared. The light gray segments correspond to the floors and ceilings, while the black points in the rest of the image are discarded during testing.

Since there can be a wide variety of valid placements for a 3D mesh chair in any given image, the results from chair placement will need to be evaluated manually. The main false cases are images where there was enough space for the chair to show up but it didn't, or images that didn't have a floor but still placed a chair.

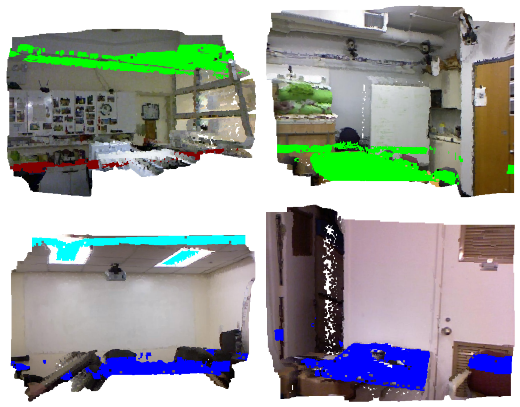

Figure 10. Examples of floor false positives, mostly flat singular or sets of objects.

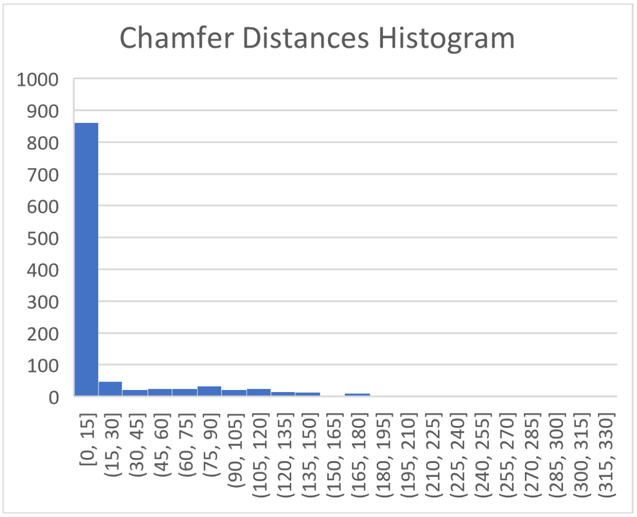

Figure 11. Histogram Plot of all the Chamfer Distances [5]

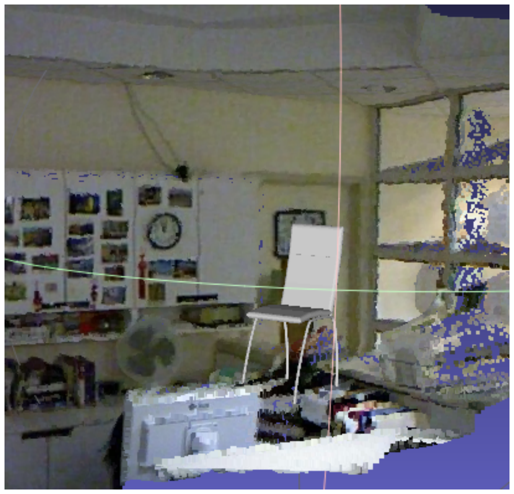

Figure 12. Point Cloud 12, where there table below the chair was detected as a floor

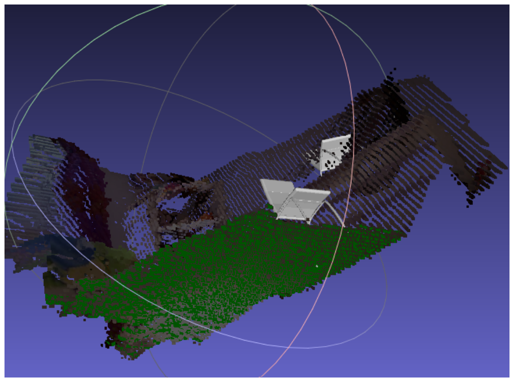

Figure 13. A tiny piece of the chair's bottom right leg is sticking through the floor of this point cloud. This is a result of the down-sampling averaging out the points in the floor.

4.3. Results

Out of all the images in the dataset, the following lists the results obtained for the floor and ceiling:

Floor False -	Floor False +
202	75

Ceil False -	Ceil False +
344	16

Floor Correct	Ceil Correct
1172	1089
.81	.75

Additionally, for images that did estimate a floor and/or ceiling correctly, the average Chamfer distance is 22.24 pixels. The standard deviation is 44.879, indicating a high variance between images with very low distances and images with very high distances, most likely due to either a missing floor or a ceiling that should have been detected. The smallest distance value was 0.015547628 pixels, and the largest was 326.2775313 pixels.

For chair placement, the model was run on the full set of images, and manually checked if the chair was placed in an accurate manner. Out of the 1449 images, the model was able to place a chair in 608 of them. Increasing the tolerable floor height by 1 pixel i.e. carpets, raised the number of chair placements. Chairs could not be placed at all in images without a floor detected, and many images with floors were too cluttered for the chair to find a position. This is considered valid, since in real life, a person would not be able to place a chair where there's no space.

Rotation introduced problems with the chair's position, as did the tilt angle. After taking a random sample of 60 chairs, the model had an accuracy of .63, or $\frac{63}{100}$. Some invalid placements include images with incorrect floors, as in Figure 12, in which case the model found the wrong object as the floor, usually a table or a group of chairs. How-

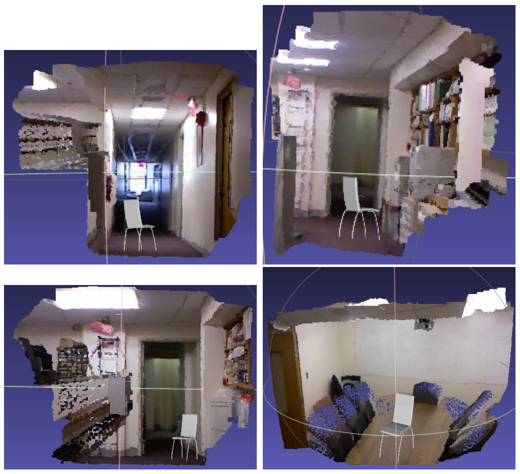
Figure 14. Examples of Properly Scaled Chair Images

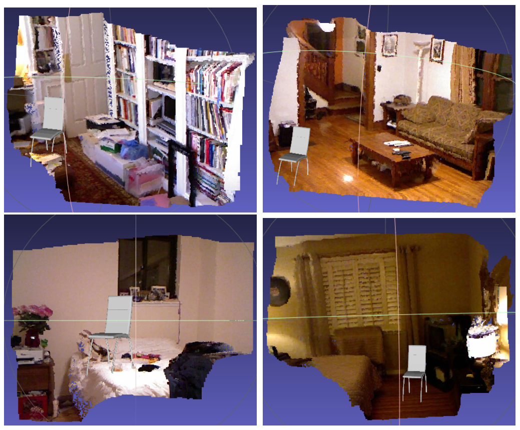
Figure 15. Examples of Unscaled Images with no Ceiling Detected.

ever, other invalid placements are images where the chair is slightly below the floor, with two or more chair legs sticking out. I confirmed that the chair and the floor should be parallel, after the rotation step. Through testing, I confirmed that these minor inaccuracies are caused by the voxel down-sampling. Since the down-sampling returns the average point in the floor, the chair is translated a little more further than it should be. The effect is worse the more down-sampling occurs. However, the down-sampling is necessary to reduce the run-time of the algorithm, due to the very large number of points that need to be examined. An example can be seen in 13.

Most valid placements were in relatively empty rooms with a one-point perspective shape, like an alley or a tunnel. Most valid images where there was no placement were crowded two-point perspective images.

Some examples of scaled images, where both a floor and ceiling were detected, are shown in Figure 14. Placements worked as expected for both scaled and unscaled images, as shown in Figure 15. In unscaled images, sometimes the chair would not be in proportion to the rest of the objects in the room.

5. Conclusion

The results from only using heuristics look promising, suggesting better possibilities from using more advanced methods, such as a convolutional neural network. I would like to setup a better evaluation metric for the RANSAC plane segmentation, given my heuristic results. There were cases where the RANSAC algorithm detected a set of chairs or a table instead of the floor, and cases where all the planes were vertical even when there was supposed to be a floor or ceiling, so these improved models would likely be able to solve these corner cases.

Chair placement has more room for improvement, namely in terms of accuracy and speed. Greater accuracy with floor estimates will greatly improve chair placement, as they are directly correlated. Many invalid chair positions would be removed through accurate floor detection. Chair placement can be greatly optimized through better means to search through the point clouds, perhaps through setting up a better structure for their data. It would be preferable to access the point cloud values as a grid, so that only the relevant section needs to be searched.

If all objects in the room could be detected, this problem could extend to moving any of them to different places in the same room. This would be useful as a virtual environment where users can test different layouts for their rooms, loading up an empty version of their room, then placing and moving the other objects around. Users might also want a live version of this application, where they can add an object to their room to decide if they want to purchase it.

References

[1] Ascii chart. https://en.cppreference.com/w/cpp/language/ascii. Accessed: 2022-02-21. 4

[2] Chair simple 3d model. https://free3d.com/3d-model/chair-simple-2585.html. Accessed: 2022-03-02. 4

[3] Convert string to float in numpy. https://www.delftstack.com/howto/numpy/numpy-convert-string-array-to-float-array/. Accessed: 2022-03-09. 3

[4] Image segmentation using color spaces in opencv + python. https://realpython.com/python-opencv-color-spaces/. Accessed: 2022-03-15. 4

[5] Pulling nonzero values from a column. https://www.mrexcel.com/board/threads/pulling-nonzero-values-from-a-column.284741/. Accessed: 2022-03-18. 5

[6] Python - files i/o. https://www.tutorialspoint.com/python/python_files_io.htm. Accessed: 2022-02-19. 2

[7] Python file write. https://www.w3schools.com/python/python_file_write.asp. Accessed: 2022-03-09. 3

[8] Python write csv file. https://www.pythontutorial.net/python-basics/python-write-csv-file/. Accessed: 2022-02-19. 4

[9] Quick start guide. https://docs.h5py.org/en/stable/quick.html#quick. Accessed: 2022-02-16. 4

[10] Read .mat files in python. https://stackoverflow.com/questions/874461/read-mat-files-in-python. Accessed: 2022-02-16. 4

[11] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. D. Chiara, and U. Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 1, 3

[12] S. Q. Q. Z. Y. Z. Z. C. Jiaqi Yang, Zhiqiang Huang. Toward efficient and robust metrics for ransac hypotheses and 3d rigid registration. pages 893–906, 2022. 2

[13] A. Muntoni and P. Cignoni. PyMeshLab, Jan. 2021. 1, 3

[14] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 4

[15] H. B. Rifat Kurban, Florenc Skuka. Plane segmentation of kinect point clouds using ransac. pages 545–551. 1, 2

[16] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, and D. Lin. Density-aware chamfer distance as a comprehensive metric for point cloud completion. *CoRR*, abs/2111.12702, 2021. 2

[17] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 1, 2, 4

6. Supplementary

The link to my repository is here: Github Repository Link