Part 1

To compute the plane sweep homography for each depth, we make use of the following equation:

$$H_{\Pi_m, P_k} = K_k \left(R_k^{\mathsf{T}} - \frac{R_k^T C_k {n_m}^{\mathsf{T}}}{d_m} \right) K_{ref}^{-1}$$

To avoid over-using the for loop, we utilize numpy function np.einsum to multiply matrix and depth vector efficiently.

Part 2

To compute the plane sweep volume, we first warp all the image with all given depth and store the result into a huge 5D array with dimension {M, D, H, W, 3}, where M is the number of images, D is the number of depths, H is the high of each image, W is the width of image and 3 is the number of channels. The 5D array can be viewed as a 2D array of 3D images.

After getting all the warped images, we can efficiently compute the variance of the warped images. Here, the variance is the l1-loss of each channel. We also filter out the pixel [0, 0, 0], which mean no image is warped to that position.

Part 3

To compute the depth image, we just need to get the argument minimum as index to select give depth.

Part 4

For the post processing, we make use the method that mentioned by TA:

We filter out the position that too few pixels are considered when calculating the l1-loss. Here, we set the count threshold to be 5 because we find out that when we set the threshold to be 6, then too many pixels will be filtered out.

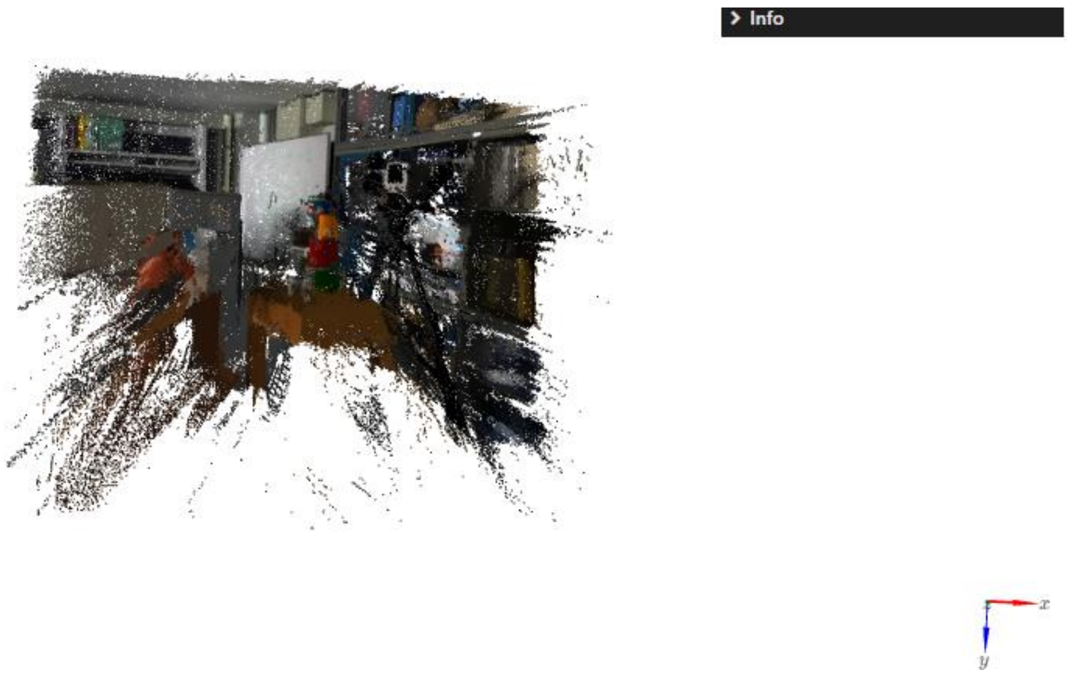
We also make use of the gaussian filter to smooth out the l1-loss.

Furthermore, we filter out the pixel that have too large l1-loss. We define a pixel has too large l1-loss as the following:

> For each pixel in 3D plane sweep volume, we first calculate the average l1-loss among all depth and the average result will form a 2D array of size {H, W}. Then we calculate the average l1-loss, denoted as avg, again but this time, we are finding the average l1-loss among all pixels. Then we will filter out the pixel with average l1-loss among depth larger than avg.

Result

Before Processing:



After Processing: