Hello, I’m Isaac and I will introduce Neural RGBD, which apply NeRF model to 3d-reconstruction​.

With NeRF, it’s able to generate very precise meshes but requires lots of memory resources.​

This method works by first embedding the information contained in the rays casted from each pixel in the picture such as (position, color and depth)

And then, gathering all the rays in the images to train model with loss functions on color, position and depth.​

And finally generating the meshes using the model with a series of queries about signed distance function and render the mesh using marching cube.​

Some of our methods produced outputs with redundant parts which are pure noises along with training process.

This was especially apparent in Neural RGBD​ which generates lots of unconnected chunks.

For example, in this picture, the lower part of the mesh is irrelevant to the room we are going to reconstructs.

So, we created a post-processing script that removes all Those components of the mesh. This boosted the overall quality significantly.

This is the result of Neural model and it apparently contains more details than Manhattan and TSDF, however, there’s still some wrong details in the meshes. This is because of lack of computing resources.

Also, it is worth noticing that the result doesn’t contain any visible gaps.

So, about the pro and cons,

Neural method apparently has less gaps, and it can generate a draft mesh very quickly and most importantly, it contains more details.

However, running the model requires lots of memory space and it takes long time to converge if you require a high precision.

Let’s move on to the Merics. Apparently, Intrinsic3D has the overall best quality with highest precision and F1, and lowest Dist1. At same time having descent dist2 and Recall rate. All the other methods performs quite average with several highlights and shortboards.

However, because we aim to apply 3d-reconstruction on robot navigation, in some scenarios, the quality is not the only thing that we need to put in mind. The efficiency, or in other word, running time is also critical to the practicality of them.

So, we ran the models on our own devices and renormalised the result using a uniform benchmark to make the result more comparable. And finally, we had this table.

This time, the previous top performer intrinsic 3d and related methods turns to be the slowest one. And Manhattan SDF, can secure a high fresh rate to match real-time usage. While the performance of other ones are quite acceptable.