Natalie Lindsay
05/2020

Masters Report

Roominoes: Learning to Assemble 3D Rooms into Floor Plans

Project Overview

Our objective is to assemble novel and realistic floor plans through the use of existing 3D room scan data and a graph neural network, which will ultimately be rendered in 3D. This could be useful particularly for training autonomous agents in varied home environments.
The overall pipeline starts with the room data from multiple datasets: Matterport3D, LiFull, and RPLAN. They are all converted into a normalized 2D representation, which is then converted to a graph representation and passed to the Graph Neural Network. In these graphs, each node represents a region, and the edges indicate which regions share portals with one another. The next step is to do room retrieval, i.e. find rooms that fit together reasonably well and assemble a floor plan. Directly finding rooms that perfectly match would leave each insertion with a very limited number of options, and would be quite difficult to implement. Instead, we adopt a metric learning perspective: instead of predicting rooms that will fit perfectly, we learn an embedding of rooms, such that similar rooms are grouped closer to each other. Given a partial floor plan, the goal is then to predict a distribution over the embedding space of rooms that will yield rooms that can fit well into the current floorplan. These won't fit perfectly together, so the next step once the floor plan is complete is to run an optimization program on it to snap the rooms together and resize if necessary. The final step is to take these completed floor plans and render them in 3D. This involves using the original region mesh, plus the new positions and sizes output by the optimizer, to deform the mesh using ARAP.

My Contributions

My first contribution to this project was the parsing of the Matterport3D data. This consisted of taking the exported files from Matterport and translating them into a representation that was usable by our network. The Matterport data consisted of mesh files and various segmentation files for each house and region in the dataset. I implemented a script in Python to translate this data to the representation we wanted, which was a JSON file with information for each region about the walls, objects, and portals. Most of this was straightforward, with the exception of the portals. The data didn't have a consistent way to annotate where the actual holes in the walls were for each door. I ended up having to implement a script that would cast rays at a wall and detect where the hole was. This partially worked, but couldn't cover all cases because some portals had no annotations at all. As a result, some regions that were parsed ended up with no recorded portals, and these had to be filtered out of the dataset.

My other main contribution to this project was work on Stage 3 of the pipeline in transitioning the 2D output from the network to a 3D mesh representation. This stage needed to take the output from the network that had been sent through the optimizer, and use it to deform the original mesh of that region to fit the new positions of the walls and objects. We used the libigl As Rigid As Possible 3D shape deformation library to accomplish this. As Rigid As Possible (ARAP) uses a number of preset “control points” as input - these are calculated beforehand to be the exact positions we want them to be in the final mesh, and all other points in the mesh will then deform, while staying as rigid as possible, to fit those new positions. We wanted our control points to form a “control cage” around the region we were trying to deform, meaning that the points would be placed all along the edges of the mesh to form a cage-like structure around it. At first, we selected points only that lay along the edges of each wall of the room, but this allowed too much deformation of the walls. Instead, we decided to select all points that lay on the plane of any of the walls of the room, and that resulted in a much more desirable structure.

There was also the question of what to do with the objects inside the room during ARAP. The first thing we tried was to do nothing with them, but this approach led to noticeably unrealistic deformation of the objects. Next, we tried pinning them to their original positions, but this didn’t look great either, since the walls and floor of the room had stretched. We finally decided to have the optimizer output new positions for the centroids of all objects, and use those new positions to pin them.

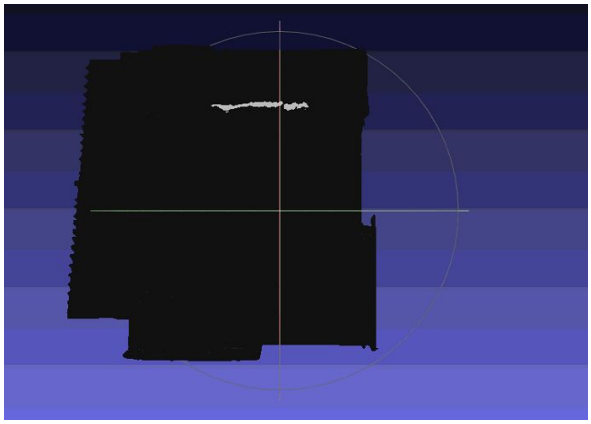

Figure 1: The original top down view of the mesh

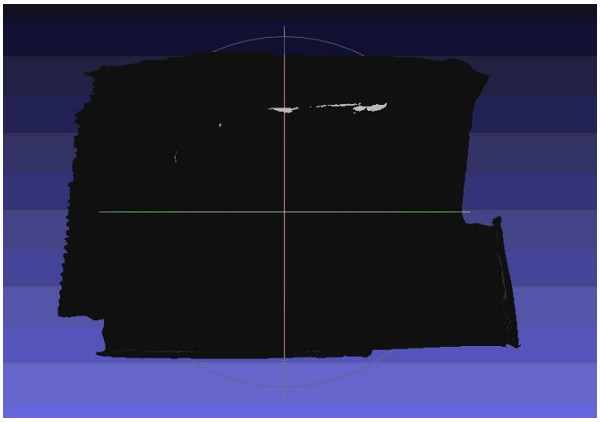

Figure 2: The top view after ARAP

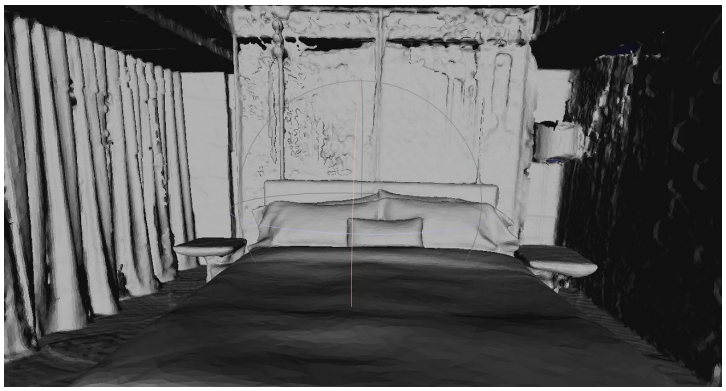

Figure 3: the interior of the region prior to performing ARAP.

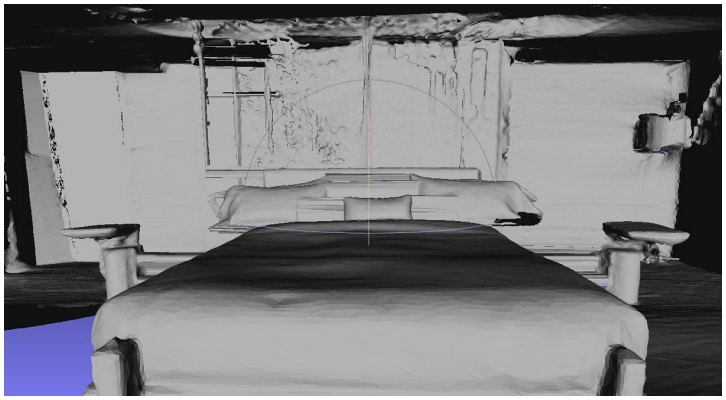

Figure 4: the interior after ARAP. There are a couple noticeable issues. 1. There is stretching between the pillows on the bed as well as between the bedside tables and the bed because they are each considered a separate object. 2. The hole in the floor gets larger because we haven't forced the vertices along the edges of the mesh to stay put.

Figure 5: a view of the wall-mounted tv in the original mesh, prior to arap.

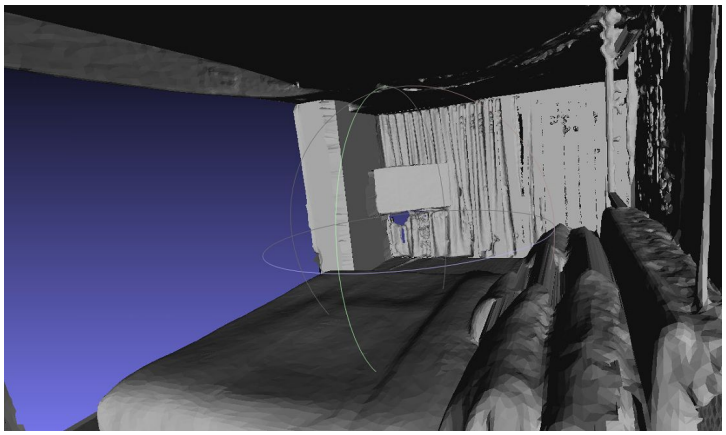

Figure 6: a view of the wall mounted tv in the deformed mesh, after ARAP is performed. It doesn't look too bad, but there is a bit of weird deformation going on where it touches the wall and the curtain.

Conclusion

One significant limitation of this pipeline is that we can only handle rectilinear regions - that is, we can't handle regions with diagonal walls. This has to do with the way the optimizer works, and so in order to fix this problem we'd have to find another way to optimize the floor plans. Another limitation, specifically with ARAP, is that there is some inaccuracy in the deformation of the objects due to incomplete labeling of object vertices. Some vertices get left behind as the object moves, creating artifacts in the mesh. Another limitation to do with the data is the labeling of portals. We had to throw away a significant portion of the Matterport data because those regions had no labeled portals. If we had more time, we could have written something to detect those portals, or even manually labeled where they were, in order for that data to be usable.

One more issue that could be improved with ARAP is that when objects are attached to a wall in some way, this can cause artifacts in the mesh (this type of situation can be seen in Figures 5 and 6, with a wall-mounted tv). This is because the object will be pinned to a specific position as the wall stretches, and therefore we can get issues like triangles getting flipped and possibly bunching of the mesh. To fix this, we could make objects on a wall move somewhat with the wall as it stretches or shrinks, but still maintain their original shape.

A couple other issues with ARAP are illustrated in Figure 4. One issue is that objects that should stay together as a unit may be stretched apart because they are considered separate pieces of the mesh. To fix this, we plan to group certain types of objects, like beds and bedside tables, together. The other visible issue is that the hole in the floor gets larger as the room stretches. To fix this, we plan to pin find the vertices along any edges of the mesh, and pin them to their original positions.