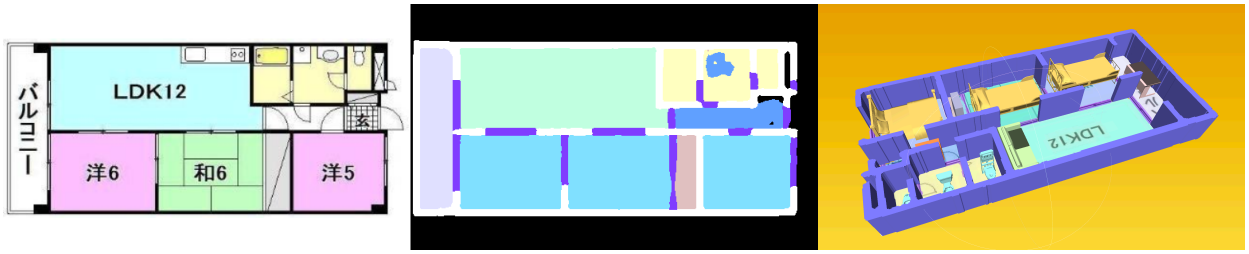
CSCI5210 Programming Assignment Report

ZHANG Yuechen, 1155091988



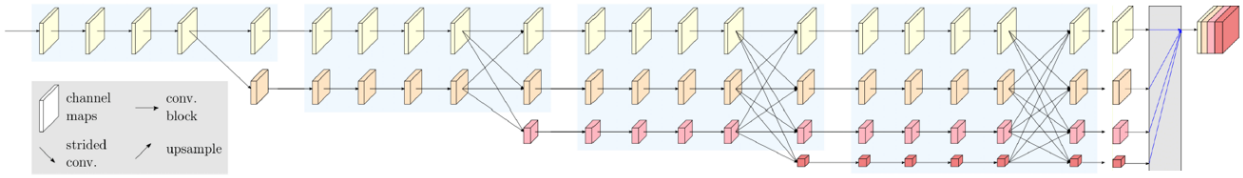
1. OVERVIEW

In this assignment, a simple semantic segmentation framework was implemented to parse real-world floor plan drafts into semantic regions. These regions include different types of room, walls, and openings (door and window). Then, a postprocessing step was applied to generate three-dimensional model based on the segmentation result, with a strategy of furniture placement. This report will show the technical details and innovations in this assignment.

2. METHODS

2.1 Semantic Segmentation for Floor Plan Parsing

As the first part of this assignment, a training framework was implemented. Since the restriction of not directly using codebases from the Internet, this training codebase is implemented referencing by Detectron2 and MMSegmentation. High-Resolution Representation Network (HRNet) was used as the baseline of this task, which is a light-weighted high-performance network in the semantic segmentation task. The overview of the segmentation network is shown in the following figure, a high-resolution representation is preserved as an end-to-end branch in this network, which could get a finer result in the segmentation task.



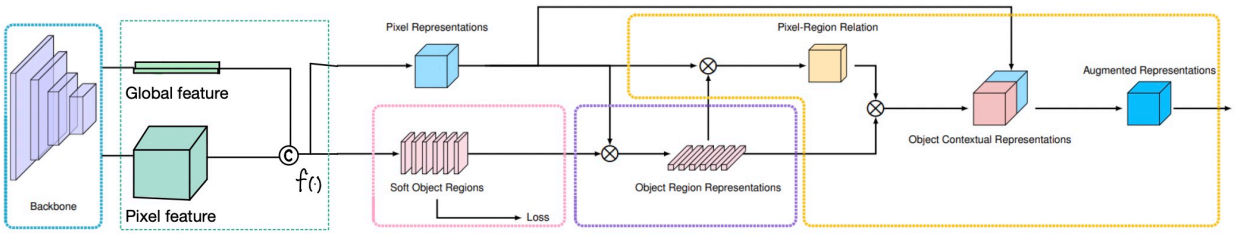
Data augmentation is also an important factor influencing the segmentation performance. In this task, after rescaling all images to size (512, 512), multiple data augmentation processes were applied, like horizontal/vertical flip, image resize, random crop, random rotation, brightness and contrast adjustments. An important thing is that all augmentations were assigned with a relatively small probability to operate in sampling, which means during the training, we want to reserve the main feature distribution of the data as same as the original training dataset. The main reason of this operation is that in this dataset, room information like text may be damaged by spatial transformations.

The training setting is following general semantic segmentation tasks. SGD optimizer is applied with learning rate 0.01, and a doubled learning rate on the decoder part. Polynomial learning rate scheduler is applied with a power number 0.9. After a simple calculation on the ground truth of the validation set, a weighted cross-entropy loss was applied on the prediction p.

$$L = \sum_i^C \frac{w_i}{\sum w_j} CE(p, q_i), \qquad w_i = 1 + \log\left(\frac{\sum t_j}{t_i}\right)$$

In which w_i is the weight of class i, which is normalized by its summation. t_i is the number of ground truth pixel in class i in the validation set. This loss item could solve the class unbalanced problem in some degree by using the data distribution of one dataset, and the logarithmic operation on the class weight could let the regression process more smoothly.

Besides, an augmented network architecture with a hybrid decoder head was applied to further improve the overall accuracy. Inspired by the pyramid pooling module in PSPNet, we replaced the original head with an attention module using Global Average Pooling (GAP). Also, Object Contextual Representation module (OCR) was applied on the refined feature, which is an attention module based on the object class region. The structure of our hybrid decoder head is shown in following:



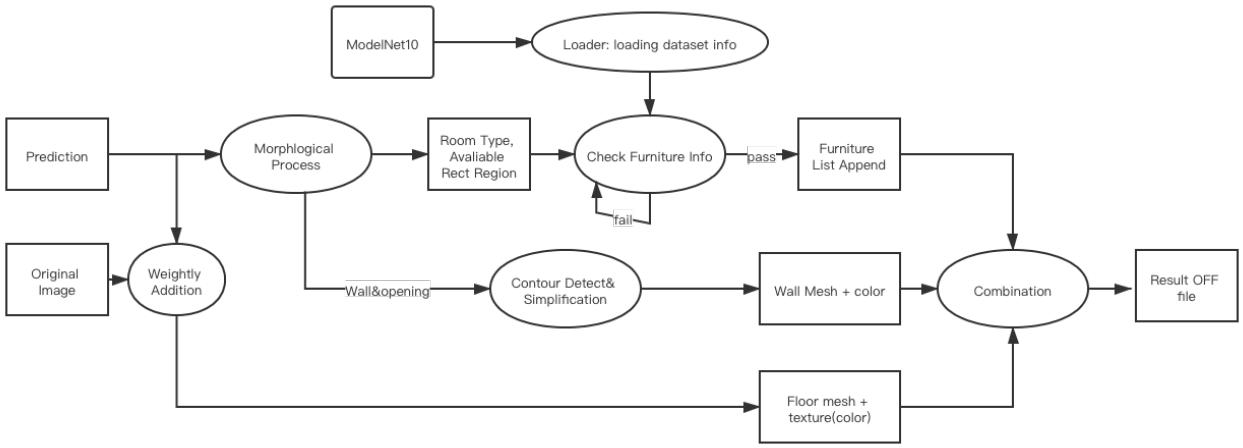
GAP is shown in the green block, in which a global feature will be extracted and expanded after transformation functions. It concatenated with the pixel feature though a channel compression transformation. All transformation functions were implemented by combinations of 1x1 convolution, batch normalization and ReLU activation layers. Then, a full OCR module will extract a soft object prediction from the output of GAP, use the prediction region as the attention key-value pair for further pixel representation attention processes.

What is more, in this augmented setting, two training tricks were applied. Firstly, in some ground truth label images, several gaps exist between room and walls, which may mislead the model training. So instead of filling background, we fill the gaps between the room and room boundaries (walls/openings) with the ignore label. This morphological process is down only in training, and we test on the origin label image during the test phase. Besides, the auxiliary output determined the attention context, which will have a large influence on the final output. To solve this problem, linear annealing function was designed to balance the auxiliary loss and final loss during training. At first, the auxiliary loss dominant the loss term with a larger weight factor. Finally, the final output is more important for the model optimization.

2.2 3D Model Generation

As the second part of this assignment, this section is to use the segmentation result and the furniture model database to generate a 3D floor plan model. OFF format was chosen in this assignment, because of the simple usage.

The pipeline of the reconstruction process is shown in the next figure. Before reading visualization results, a loader was implemented to load all data in the ModelNet10 dataset, recorded the width, length, and x/y/z offset information. Then we can extract rooms from network predictions. Rooms were separated after a erode and a connect component operation, filtered by an area threshold. Similar to the postprocess step of the reference baseline, the major indexes were calculated in each room to determine the type of this room. After that, a simple max inner-rectangle algorithm was performed to get the max available area of one room to place the furniture. Finally, a checking was performed randomly several times to get access the geometric information saved in the loader we mentioned. Until we can get a suitable furniture in that room.



In the 3D reconstruction step, the floor is composed by pixels from the original texture image. For the wall regions, instead of using cubes for each pixel, a simplified contour-based method was applied to save the model file size and remove redundant vertices/surfaces. A vertex offset value in the program could maintain the growth of the file, enabled the combination of multiple components (floor, walls, furniture in each room) into one .OFF file.

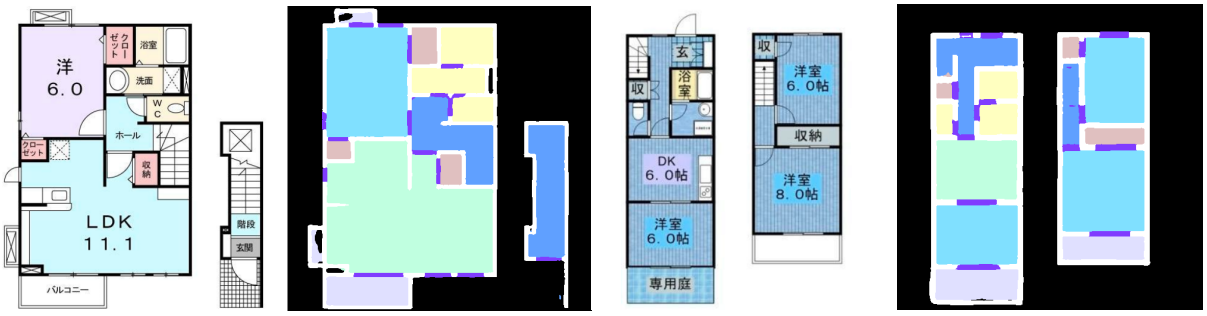
3. RESULTS

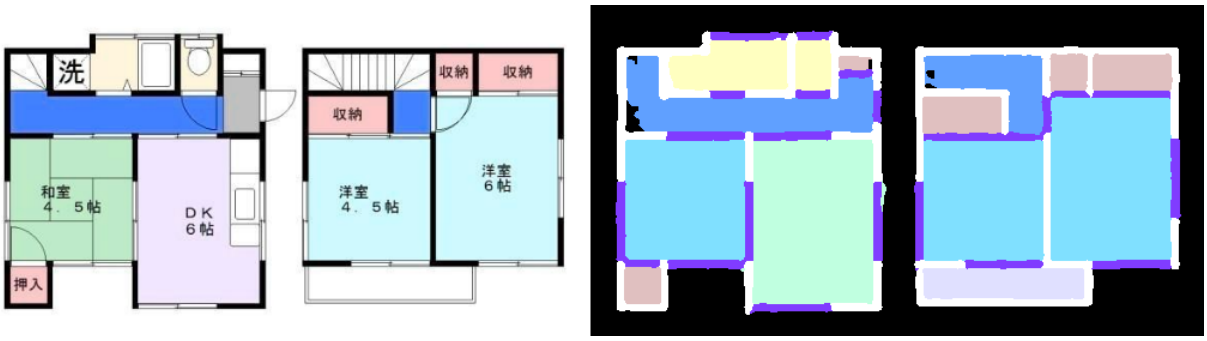
3.1 Accuracy for Segmentation Model

The segmentation model was trained on a NVIDIA Titan RTX, with a batch size 24 and 20000 iterations. For more details, please refer to the configuration file in the experiment folder. The following table shows the segmentation result (pixel accuracy) on the R2V dataset in different config settings mentioned in the previous section. Best results are shown in **bold**.

Exp/Class index	Overall	0	1	2	3	4	5	6	7	8	Average Acc
ICCV-19	0.90	-	0.92	0.93	0.91	0.91	0.84	0.92	0.89	0.89	0.89
01 HRNet	0.947	0.944	**0.958**	0.972	0.944	0.964	0.931	0.972	**0.923**	**0.918**	**0.948**
02 HRNet-Aug	**0.959**	**0.964**	0.949	**0.973**	**0.947**	**0.974**	**0.940**	**0.974**	0.909	0.907	0.947

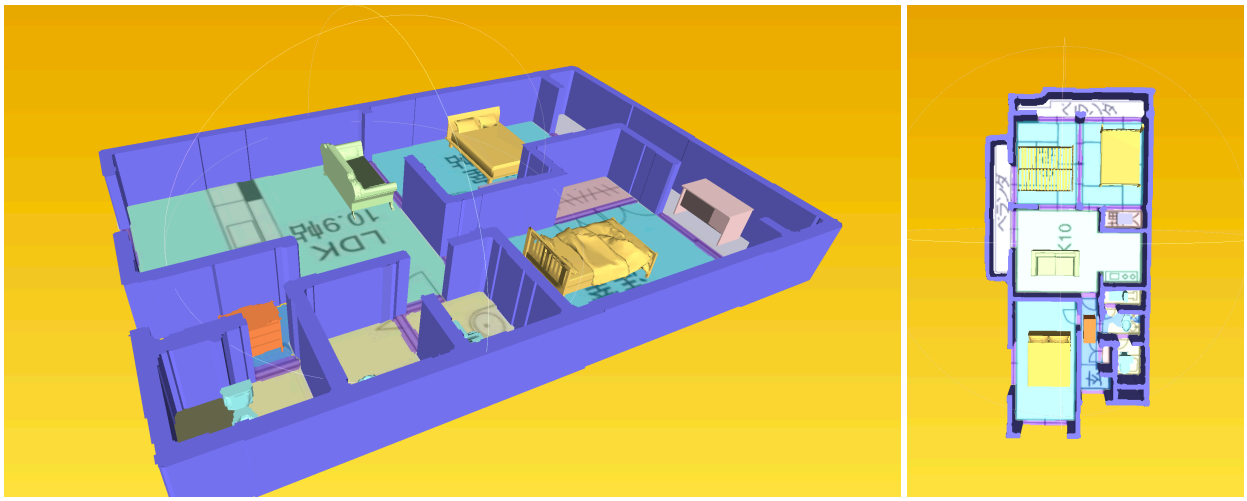
Both two experiments were significantly overpass the provided baseline result by at least 4.5% in overall accuracy. The augmented setting has a better overall accuracy, and a better class accuracy in most room categories. However, in non-room classes (7, 8 for wall & openings), it did not perform as well as the basel HRNet experiment. I think it is mainly because of the effect on the global context. The average class accuracy results of two experiments were almost same. Following shows more result visualizations in the augmented setting.





3.2 Model Generation Result

Following figures shows some 3D reconstruction results, better visualize in color.



4. CONCLUSION

To conclude, I would like to highlight what I think what are the advanced parts I had done in this assignment:

1. Implemented a more powerful semantic segmentation network architecture and a training procedure, improved the segmentation result by at least 4.5% in overall accuracy.
2. Use a simplified algorithm to reduce the 3D model file size by using contours in wall construction.
3. Applied a furniture placement algorithm to generate properly furniture positions automatically.

Note: I do not think copy codes from the Internet should be a restriction in this assignment. Actually, there are some opensource segmentation codebases, it is quite a heavy workload for me to *reimplement* them (actually, the network architectures, config parsing, and data augmentations in my assignment were almost as same as the official opensource version in some repositories). Maybe setting restrictions on opensource floor plan parsing / 3D reconstruction codes is more suitable.