Room Layout Estimation Using a Machine Learning Technique

Conference Paper · December 2021

DOI: 10.1109/ICECET52533.2021.9698767

2 authors:

Nattaon Techasarntikul
Osaka University
6 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Tomohiro Mashita
Osaka University
88 PUBLICATIONS **579** CITATIONS
SEE PROFILE

Proc. of the International Conference on Electrical, Computer and Energy Technologies (ICECET)
9-10 December 2021, Cape Town-South Africa

Room Layout Estimation Using a Machine Learning Technique

Nattaon Techasarntikul
Osaka University
nattaon@lab.ime.cmc.osaka-u.ac.jp

Kazumi Tsuchida
Osaka University
tsuchida.kazumi@lab.ime.cmc.osaka-u.ac.jp

Tomohiro Mashita
Osaka University
mashita@ime.cmc.osaka-u.ac.jp

***Abstract*—The purpose of this paper is to propose a room layout estimation pipeline from indoor point cloud data using a machine learning technique. During data pre-processing, each point cloud was aligned along the X, Y, and Z axes, and then projected along the XZ dimensions to a 2D image called a point density image; the number of points along the Y dimension (room height) that fell into the same pixel coordinate was also counted. The pixels with a high accumulation of points were typically wall areas and were colored in a higher color range index. We trained pairs of point density images and ground truth images using the U-Net model. The trained model could predict rough wall areas of unseen images. Post-processing was applied to the predicted result in order to estimate connected walls, generate a vector image, and draw the length of the walls. Because the results from the automated process always contained some errors, we created a user interface to complete the final layout with just a few modifications.**

***Index Terms*—Room layout prediction, Floor plan prediction, U-Net.**

I. Introduction

A floor plan image is necessary for understanding the layout of a house or a building. However, it sometimes gets lost when ownership is transferred or becomes outdated due to renovation. Recreating the floor plan is laborious due to the need to measure and draw all dimensions and positions of the walls, pillars, doors, and windows. For example, a floor plan for a typical 3-bedroom house takes approximately 10 hours to complete and costs around 500 to 2,000 USD [1]. However, today's commercial depth cameras have become affordable (starting at 149 USD) [2]. Moreover, recent 3D reconstruction techniques [3], [4] enable anyone to have a 3D scan of their house. A depth camera measures the grid distance to the object and produces data in the form of dense points around the object surface, which are then transformed into a 3D mesh for 3D visualization. Nevertheless, the 3D scan data (3D mesh) contains all objects in a house that the camera sees, which is redundant for the purpose of visualizing a floor plan.

In this work, we present a pipeline for recreating a room layout image from scanned indoor point cloud data. The 3D point cloud data is transformed into a 2D point density image (top-view) to be used as training data. We annotated wall positions as a ground truth image of each point density image and then trained them on a U-Net model. The model could predict a rough layout image that eliminated furniture and other non-essential items. We then applied an automated wall joining function to the predicted result. A vector image of the room layout with a wall-size depiction was calculated. A layout editing interface was developed and included in the system for completing a final layout.

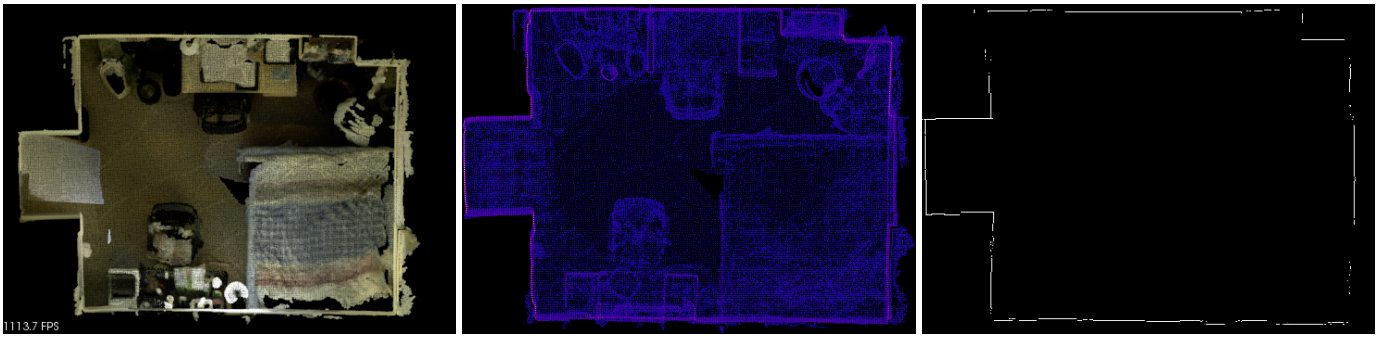
The significance of this paper is the use of a U-Net model for training on cropped 2D-projected point cloud image data to predict a layout that excluded extraneous information (e.g., furniture and clutter), which existed in the input image, as well as a post-processing technique that automatically joins wall segments to create the final layout image with the dimensions indicated at each wall or showing the layout in 3D.

II. Related Work

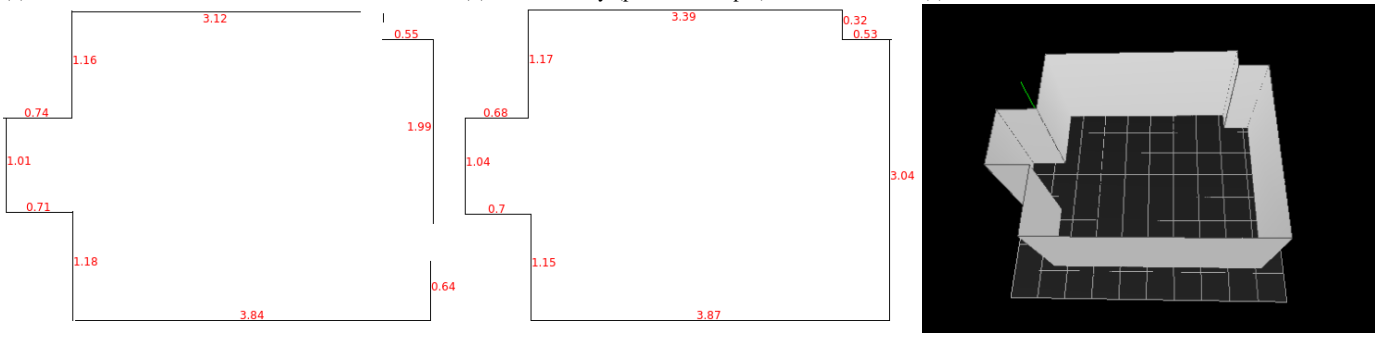
Estimating a room or multi-room layout (floorplan) from point cloud data has been done in the past using 2D and 3D heuristic approaches that defined rules for detecting the room elements such as walls. Since a wall is usually constructed perpendicular to the ground, the 2D method is computationally efficient, because it simplifies the 3D point cloud data by projecting it to a 2D plane. Most research detects lines in 2D data and planar surfaces in 3D data using the RANSAC algorithm [5], [6] or the Hough transform algorithm [7], [8]. Due to the variety of input data, these techniques need to configure parameters on each set of data.

Recently, data-driven approaches (i.e., training a large quantity of data using machine learning techniques) are increasing. One example is Floornet [9], which uses three neural network branches including PointNet and two CNNs to extract 3D point cloud features to predict a floor plan image. It was trained with 152 sparse indoor scanned point cloud data captured by the Google Tango smartphone. Floor-SP [10] trains the Mask-RCNN semantic segmentation model to predict room segments from a 2D-projected point cloud image and applies post-processing to generate a vector-graphics floor plan. It was trained with 527 point cloud data captured by a production quality RGBD camera. DeepPerimeter [11] classifies ceilings, floors, and walls through the pyramid scene parsing (PSP) network from point cloud data. It introduces ClusterNet to cluster unordered point cloud data and uses the linear least-squares technique to estimate 2D line parameters. This research uses publicly available datasets from ScanNet,

978-1-6654-4231-2/21/$31.00



(a) Point cloud (b) Point density (prediction input) (c) Predicted result



(d) Automated post-processing (e) Revised by a user (f) 3D visualization

Fig. 1. Images of each step of room layout estimation from input point cloud data through final layout.

ADE20K, SunRGBD, FloorNet, and its own synthetic dataset. Scan2Plan [12] uses the PointNet++ backbone as a feature extractor of point cloud data, then estimates a closed perimeter by use of a voting technique, and joins them to form a floor plan. It is trained on a BKE dataset published by Floor-SP and a synthetic dataset from Structured3D.

Among previous data-driven approaches, the results from predictions still contain some errors, which cannot be used as a final layout image. In this research, we exploited the data-driven approach to predict a room layout. We have improved the predicted result by proposing the automated post-processing and user-driven interface to help finalize the completed layout image.

III. Layout Estimation

Our goal was to auto-generate a layout that conveyed both the dimensions and shape of a room. We adopted a data-driven approach that could learn many layout patterns by providing them as training data to estimate the layout. A U-Net model was trained with pairs of room point density top-view images and annotated room layout images. The prediction resulted in a sparse layout image that did not contain dimensional information. We therefore conducted a further automated post-processing technique to join the wall sections, and show the dimensions of each wall. A manual editing interface was also developed to enable the user to correct the final layout.

A. Data Collection

Indoor point cloud data in PLY format was collected from the SceneNN dataset [13] containing 104 indoor scenes captured by the Asus Xtion Pro camera, and the Stanford 2D-3D-Semantics dataset [14], containing 168 indoor scenes captured by the Matterport camera. Points in both datasets present a floor, a ceiling, and walls, as well as clutter (e.g., furniture). While data from the Stanford dataset completely scanned the ceiling, the SceneNN did not. Data which had no points representing walls were discarded. This left us with 222 sets of point cloud data to be used in our experiment.

B. Pre-processing

In this step, we created pairs of point density images and ground truth layout images to use as training data in the model training process. First, points that represented a floor were rotated manually to align parallel to the XZ plane and points that represented walls were rotated to align parallel to the XY or YZ planes. The final alignment in top-view is shown in Fig. 1 (a). Then, the room points were sliced into grids measuring 0.01 m (1 cm) along the X and Z dimensions. Points located in each grid were counted. The number of grids was used as a point density image size. For example, a point cloud with XZ dimensions of 4.50×6.77 m would be projected to a 450×677 pixels image.

On an average wall height of 2.5–3 m, there were around 0 to 150 points accumulated per grid. Note that our approach projected all points to the 2D point density image without removing floor and ceiling points from the data. The number of points accumulated in each grid was scaled into the 0 to 255 range. Then these numbers were mapped to a unique color that was assigned to each pixel in the point density image. We used the Plasma color map for visualization which ranges

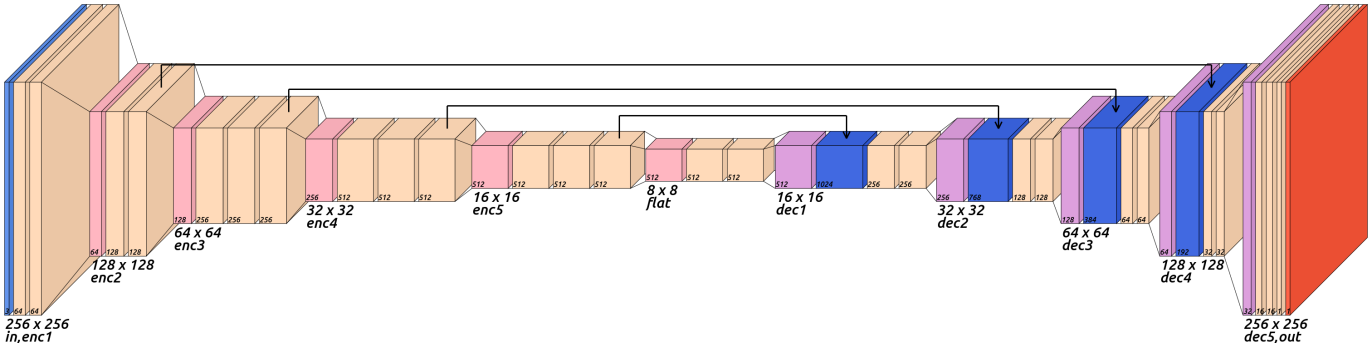


Fig. 2. U-Net model architecture.

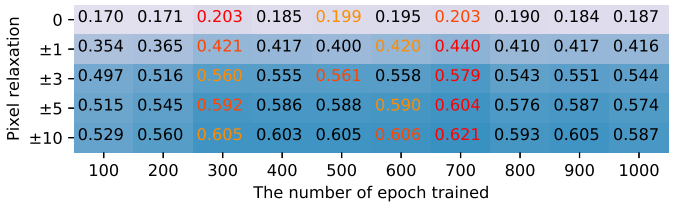
TABLE I
DETAILS OF THE IMPLEMENTED U-NET MODEL

Name	Layer type	Output size
in	input	$256 \times 256 \times 3$
enc1	[conv (f=64), ReLU] $\times 2$	$256 \times 256 \times 64$
	max pooling	$128 \times 128 \times 64$
enc2	[conv (f=128), ReLU] $\times 2$	$128 \times 128 \times 128$
	max pooling	$64 \times 64 \times 128$
enc3	[conv (f=256), ReLU] $\times 3$	$64 \times 64 \times 256$
	max pooling	$32 \times 32 \times 256$
enc4	[conv (f=512), ReLU] $\times 3$	$32 \times 32 \times 512$
	max pooling	$16 \times 16 \times 512$
enc5	[conv (f=512), ReLU] $\times 3$	$16 \times 16 \times 512$
	max pooling	$8 \times 8 \times 512$
flat	[conv (f=512), BN, ReLU] $\times 2$	$8 \times 8 \times 512$
dec1	upsampling	$16 \times 16 \times 512$
	concatenate [dec1+enc5]	$16 \times 16 \times 1024$
	[conv (f=256), BN, ReLU] $\times 2$	$16 \times 16 \times 256$
dec2	upsampling	$32 \times 32 \times 256$
	concatenate [dec2+enc4]	$32 \times 32 \times 768$
	[conv (f=128), BN, ReLU] $\times 2$	$35 \times 32 \times 128$
dec3	upsampling	$64 \times 64 \times 128$
	concatenate [dec3+enc3]	$64 \times 64 \times 384$
	[conv (f=64), BN, ReLU] $\times 2$	$64 \times 64 \times 64$
dec4	upsampling	$128 \times 128 \times 64$
	concatenate [dec4+enc2]	$128 \times 128 \times 192$
	[conv (f=32), BN, ReLU] $\times 2$	$128 \times 128 \times 32$
dec5	upsampling	$256 \times 256 \times 32$
	[conv (f=16), BN, ReLU] $\times 2$	$256 \times 256 \times 16$
out	conv (f=1)	$256 \times 256 \times 1$
	Sigmoid	$256 \times 256 \times 1$

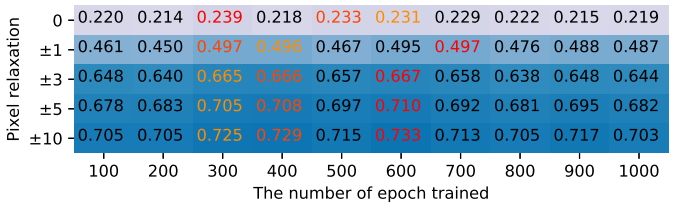
from blue to yellow. We modified the color map on the 0 index to represent the color black. The point density image which is shown in Fig. 1 (b) can be implied as follows: the black pixel is an area that has no data points. The blue pixel is an area that has few points accumulated along the Y axis, which is typically a floor or furniture area. Areas where the maximum number of density points has accumulated indicate walls, and appear as a yellow pixel in the image. However, due to imperfect scanning, only 70%-90% of the maximum point accumulated number is detected in positions with a high point accumulation, such as a wall or side of a tall piece of furniture. For this reason, these pixels are in the magenta to orange color range.

The ground truth layout was created by drawing a 1 pixel-width white color line along with the wall position over the point density image, which was then saved on a black background. These point density images and ground truth layout images of 222 room point cloud data had an average size of 506×466 pixels, with the largest size of 768×1504 pixels, and the smallest size of 128×160 pixels.

(a) Average F1 score



(b) Average precision



(c) Average recall

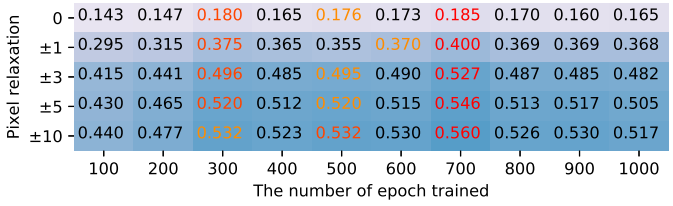


Fig. 3. Model performance on the test set.

C. Model Training

We trained the U-Net model (Fig. 2) implemented by Segmentation Models library [15] to map the point density images to the ground truth layout images. The model architecture is depicted in Table I. It is composed of five down-convolution layers, a flat-convolution layer, and five up-convolution layers. All convolution layers use 3×3 kernel size. The filter channel number in the down-convolution is double increased from 64 to 512 to learn features in the image, and double decreased back to 16 channels in the up-convolution to gain the spatial

position. The final layer uses a Sigmoid activation to convert the convolution layer output to a binary image.

To train the model, the same size of input images is required. Resizing the images into one exact size is a common approach. However, the predicted image of the resizing approach generates thick and blurred wall segments. In order to retrieve 1 pixel-width wall segments, we decided to crop the training images instead. Using the cropping approach is applicable to any size of room point density images, which allows for prediction of large scale room data by cropping and stitching.

Ten percent (22 images) of the 222 total images were separated out to use as a test set. The remaining 200 images were cropped to a size of 256×256 pixels with a stride of 128 pixels of both width and height. Any cropped images that contained wall label pixels less than 10% of the cropped size were ignored. Finally, 3,018 cropped images were ready for training. The Dice coefficient was used as a loss function. It performed very well on our dataset, which was extremely unbalanced on black and white pixels. We trained the model for 1,000 epoch with a batch size of eight images. The training data was separate 90:10 for training and evaluation.

D. Post-processing

While the U-Net model can predict rough layout images as shown in Fig. 1 (c), the line segments are needed to merge into a single wall segment. Initially, we applied the Hough transform algorithm to find line segments in the predicted image, but it mostly estimated short line segments, similar to the predicted result. Therefore we created a rule-based line joining algorithm as follows:

1) Scan the vertical lines along width dimension and group the lines that are in close proximity.
2) Combine each group into a single vertical line segment.
3) Scan the horizontal lines along height dimension and group the lines that are in close proximity.
4) Combine each group into a single horizontal line segment.
5) Merge the vertical and horizontal lines.
6) Find an intersect point of two nearby perpendicular line segments and extend them to meet at a corner.

Fig. 4 (1–6) shows each step performed through the automated post-processing. The pink segments indicate the segments to be manipulated, and the red segments show a result of the manipulation. However, the result from the automated line joining (at step 6) is still imperfect. We then created a user interface to add, remove, and join line segments. The user can modify the layout by selecting the tool mode, dragging the mouse over the line segments, and releasing the mouse to apply the operation as illustrated in Fig. 4 (a and b). The blue dash box shows the area selected by the mouse. The red line simulates an output, which will be applied when the mouse button is released.

The complete room layout revised by a user with the length of each wall segment is shown in Fig. 1 (e). The layout can be shown in 3D as in Fig. 1 (f) by applying the wall height value received from the point cloud dimension during pre-processing.

IV. Evaluation

A. Metrics

We evaluated the overlap between the areas of the ground truth and detected walls of the detected layout using a F1 score (Dice coefficient), precision, and recall. These metrics were computed as follows:

$$F1\ score = \frac{2 \times TP}{(FP + TP) + (FN + TP)}$$

$$Precision = \frac{TP}{(FP + TP)}, \quad Recall = \frac{TP}{(FN + TP)}$$

The metrics assume that true positive (TP) is the overlap between the area of ground truth and what is predicted, a false positive (FP) is what is predicted excluding ground truth, and a false negative (FN) is the area of ground truth excluding what is predicted.

Predicting the position of a line at 1 pixel accuracy is difficult for an input size of 256×256 pixels. Therefore, we loosened the TP metric similar to other works [10], [16] by considering 1, 3, 5, and 10 pixels around the predicted pixel as a neighboring pixel relaxation predicted image.

B. Result

Model performance of 22 test images are shown in Fig. 3. The top three scores are highlighted in red, orange, and dark yellow, respectively. The model that trains for 700 epochs predicted the best score for all metrics. When no neighboring pixel relaxation was applied, the average F1 score, precision, and recall were 0.225, 0.252, 0.205, respectively. When pixel relaxations were applied at 1, 3, 5, and 10 pixels (in cm unit for the actual size), the F1 score increased from 0.225 to 0.425, 0.578, 0.599, and 0.617, respectively. The precision increased from 0.252 to 0.490, 0.666, 0.695, and 0.720, respectively, and the recall increased from 0.205 to 0.382, 0.520, 0.537, and 0.552, respectively.

By observing the predicted images at the early epochs, it was noted that there were a number of sparse line segments. After training for 300 epochs, the line segments became thinner and sharper. At the later epochs, the predicted images covered more actual wall area.

Several predicted images on the test set from the model training for 700 epochs are shown in Fig. 5. The red segments indicated the predicted wall overlay on the gray segments of ground truth. The green segments are the overlap between the predicted floor plan and ground truth. By applying post-processing, the layout image becomes complete and thus can depict each wall dimension and visualize it in 3D.

V. Conclusion and Future work

In this paper, we presented the use of a machine learning technique to estimate a layout from a 2D projected point cloud image and post-processing procedures, including automated line joining and user interface to correct the result. We can

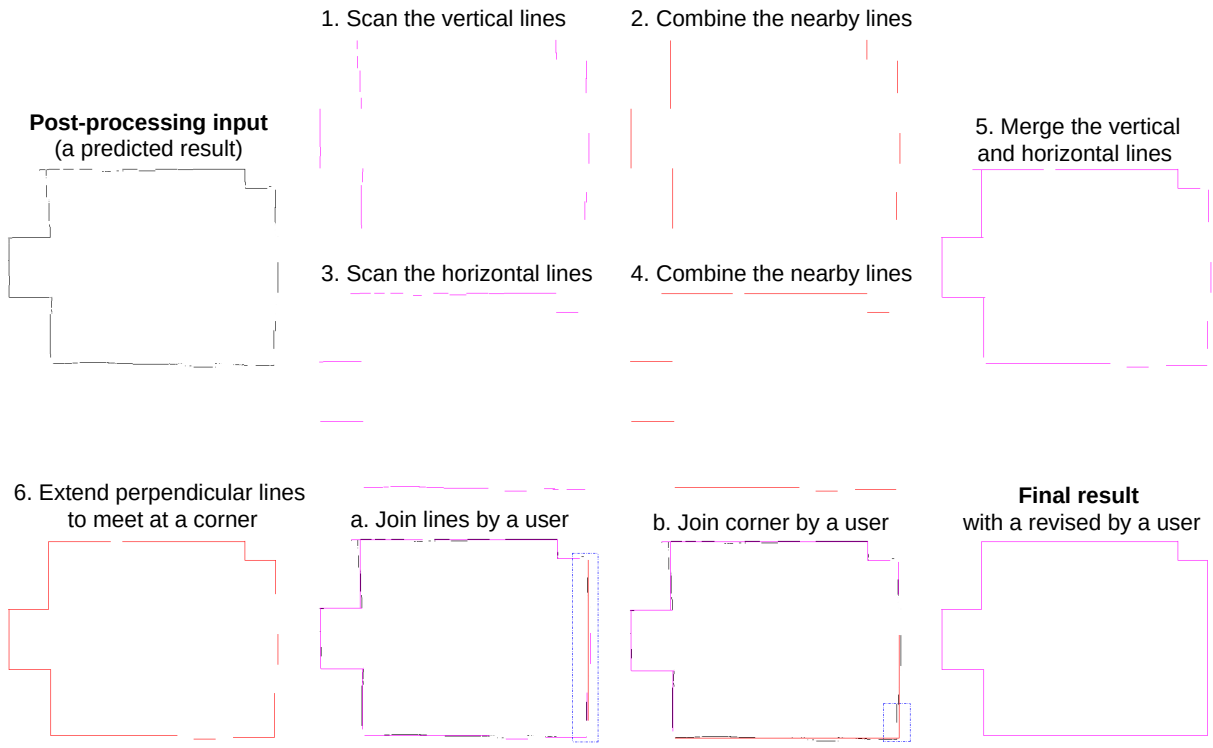


Fig. 4. Each step in the post-processing including automated processes (1–6) and manual processes (a and b).

predict any layout size by cropping an image into multiple smaller fixed-size images and feed them to the network.

This pipeline can be applied to any other indoor point cloud dataset to increase prediction performance and learn different patterns of layouts. However, this technique relies on the 2D approach, and the detection of opened doors and window position is a limitation of this work. Also, the proposed line joining algorithm cannot detect a diagonal line and produces error when applied to a layout that contains multiple rooms.

In the next step of our research, we will consider training a layout from 3D point cloud data on semantic relationships and expect the model to understand other objects such as windows, doors, and furniture.

VI. Acknowledgment

We would like to thank Daikin Industrial Japan for funding this project.

References

[1] HomeAdvisor, "Draftsmen costs - drafting blueprints or house plans," Accessed: March 15, 2021. [Online]. Available: https://www.homeadvisor.com/cost/architects-and-engineers/hire-a-draftsperson

[2] IntelCorporation, "Intel® realsense™ store – buy depth and tracking cameras," Accessed: March 15, 2021. [Online]. Available: https://store.intelrealsense.com/

[3] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5556–5565.

[4] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3d reconstruction with loop closure," in *European Conference on Computer Vision*. Springer, 2016, pp. 500–516.

[5] V. Angladon, S. Gasparini, and V. Charvillat, "Room floor plan generation on a project tango device," in *International Conference on Multimedia Modeling*. Springer, 2018, pp. 226–238.

[6] S. Ochmann, R. Vock, and R. Klein, "Automatic reconstruction of fully volumetric 3d building models from oriented point clouds," *ISPRS journal of photogrammetry and remote sensing*, vol. 151, pp. 251–262, 2019.

[7] S. Murali, P. Speciale, M. R. Oswald, and M. Pollefeys, "Indoor scan2bim: Building information models of house interiors," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6126–6133.

[8] B. Okorn, X. Xiong, B. Akinci, and D. Huber, "Toward automated modeling of floor plans," in *Proceedings of the symposium on 3D data processing, visualization and transmission*, vol. 2, 2010.

[9] C. Liu, J. Wu, and Y. Furukawa, "Floornet: A unified framework for floorplan reconstruction from 3d scans," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 201–217.

[10] J. Chen, C. Liu, J. Wu, and Y. Furukawa, "Floor-sp: Inverse cad for floorplans by sequential room-wise shortest path," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2661–2670.

[11] A. Phalak, Z. Chen, D. Yi, K. Gupta, V. Badrinarayanan, and A. Rabinovich, "Deepperimeter: Indoor boundary estimation from posed monocular sequences," *arXiv preprint arXiv:1904.11595*, 2019.

[12] A. Phalak, V. Badrinarayanan, and A. Rabinovich, "Scan2plan: Efficient floorplan generation from 3d scans of indoor scenes," *arXiv preprint arXiv:2003.07356*, 2020.

[13] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "Scenenn: A scene meshes dataset with annotations," in *International Conference on 3D Vision (3DV)*, 2016.

[14] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," *arXiv preprint arXiv:1702.01105*, 2017.

[15] P. Yakubovskiy, "Segmentation models," Accessed: March 15, 2021. [Online]. Available: https://github.com/qubvel/segmentation_models

[16] U. Gankhuyag and J.-H. Han, "Automatic 2d floorplan cad generation from 3d point clouds," *Applied Sciences*, vol. 10, no. 8, p. 2817, 2020.

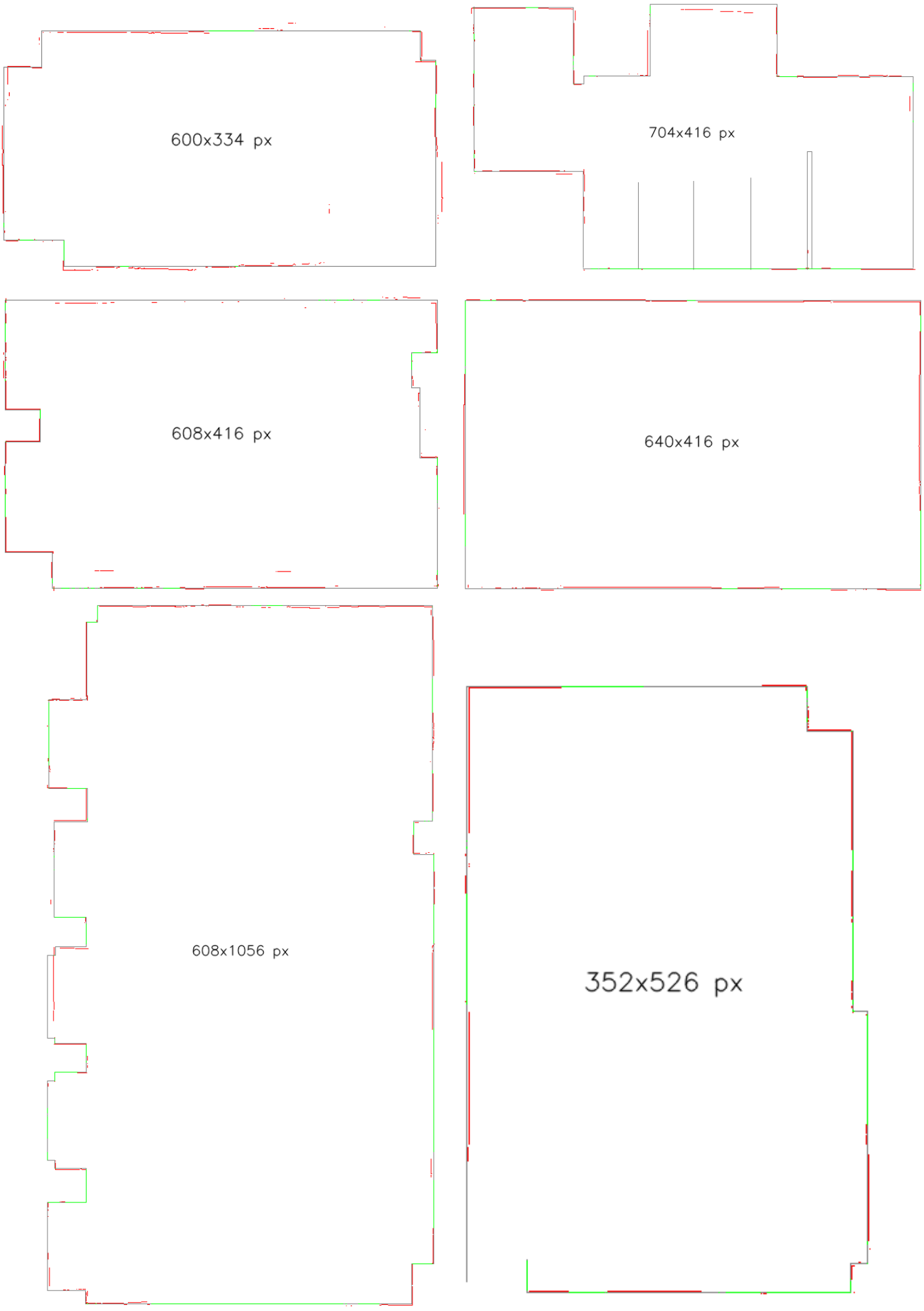


Fig. 5. Images show overlapping (green) and non-overlapping (red) pixels of predicted result overlay on ground truth (gray).