# Residential floor plan recognition and reconstruction supplementary material

Xiaolei Lv, Shengchu Zhao, Xinyang Yu, Binqiang Zhao
Alibaba Group
{yanjun.lxl, shengchu.sc, xinyang.yu, binqiang.zhao}@alibaba-inc.com

## 1. RFP dataset

### 1.1. RFP dataset annotation.

Details of annotations of dataset are described below, including structural elements annotation, text and symbols annotation and scale annotation.

**Structural elements annotation.** From left to right, an input floor plan image, manual annotation image and rasterized pixel-by-pixel annotation are shown in Figure 1. The black frame represents the cropping area. Blue walls represent load-bearing walls, usually 0.24m thick, and black walls represent non-load-bearing walls with a thickness of 0.12m. The end of the wall is indicated by a cross. The end points of the door (green) and window (red) are represented by dots. The megenta color represents the wall connected by the doorway (deep blue).

**Text and symbols annotation.** Text annotation are described in Figure 2. Multiple Chinese characters are combined into a complete word or phrase to form a category. Symbols annotation are described in Figure 2. Multiple symbols are combined into a combo, then marked. A sofa and a tea table are marked together as "sofa combo". Dinner table combo, cooker combo, toiletries combo are also shown in that figure. The cooker combo is composed of oven and wash basin in kitchen. The bed combo consists of beds and bedside table. The dining table combo is composed of dinner table and chairs. The toiletries combo of closestool, bathtub, and wash basin in bathroom.

**Scale annotation.** Scale annotation are described in Figure 3. Endpoints are divided into four classes, according to the location related to the ROI region of floor plan images. Also, every number area is marked.

### 1.2. Annotation consistency

In order to ensure the consistency of annotations, we have established a quality inspection process, which is divided into two stages. The first round of the process is completed by the annotator, who labels the floor plan image and checks all annotations, and finally corrects all possible errors. The second round is completed by different quality inspectors, who randomly check a fixed proportion of annotations (for example, 30%). If the error ratio exceeds a certain percentage (for example, 5%), the current result will be returned to the first round of annotators for annotation. If the error ratio is less than a certain percentage, these annotations will be the final version.

| | CVC-FP[9] | R2V[13] | Cubi-Casa5K[10] | RFP |
|---|---|---|---|---|
| Images | 122 | 815 | 5000 | 7000 |
| Res | 905-7383 | 96-1920 | 50-8000 | 180-3615 |
| Room | 1320 | 7466 | 68877 | 50303 |
| Wall | 6089 | 16139 | 147024 | 232415 |

Table 1. Metrics between available datasets.

### 1.3. RFP dataset statistics

In Figure 4, we show the statistical results of the number of room types. Since our floor plan is mainly residential, the housing type is relatively single. Figure 5,6,8 provide statistical information about the RFP data set, including image resolution distribution, and effective area ratio distribution. In Figure 8, we can see that at least half of the data have an effective area ratio of less than 50%. Therefore, ROI detection is very critical. Finally, in Table 1, we further compare some key statistics with all existing floor plan data sets, including image number, resolutions, room number and wall number.

## 2. Method

### 2.1. Multi task loss

We add automatic weight adjustment for each loss, and rewrite as follows:

$$\mathcal{L}_{\text{ce}} = \sum_{c}^{C} \left[ \frac{1}{\sigma_c^2} \sum_{i} -y_i \log p_i(c) + \log \sigma_c \right] \quad (1)$$

$$\mathcal{L}_{\text{affinity}} = \sum_{c}^{C} \left[ \frac{1}{\sigma_c^2} \sum_{i} (\mathcal{L}_{\text{group}}^{ic} + \mathcal{L}_{\text{separate}}^{ic}) + \log \sigma_c \right] \quad (2)$$


Figure 1. Structural elements annotation. From left to right, an input floor plan image, manual annotation image, rasterized pixel-by-pixel annotation.


Figure 2. Text and symbols annotation example.

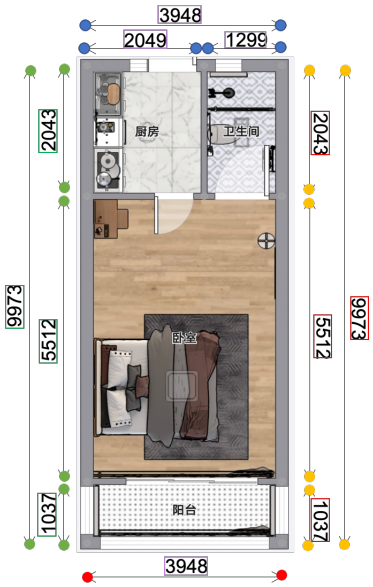


Figure 3. Scale annotation example.

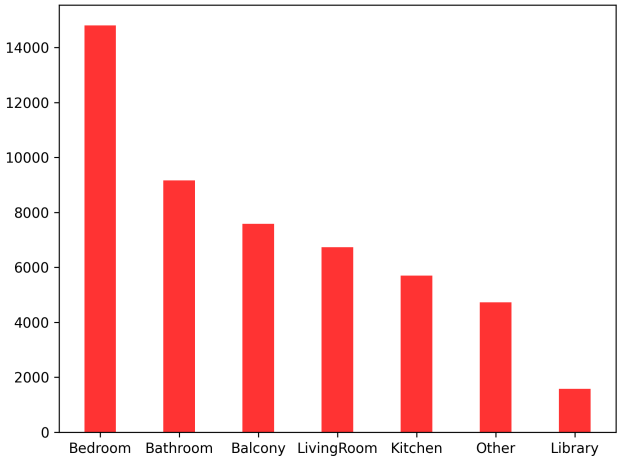


Figure 4. Number of rooms in the RFP dataset.

$$\mathcal{L}_{\text{regression}} = \sum_{k}^{K} \left[ \frac{1}{\sigma_k^2} \sum_{q} ||S_k(q) - S_k^*(q)||^2 + \log \sigma_k \right] \tag{3}$$

$$\mathcal{L} = \mathcal{L}_{\text{ce}} + \mathcal{L}_{\text{affinity}} + \mathcal{L}_{\text{regression}} \tag{4}$$

where $\sigma$ in each loss is the different uncertainty parameter that is learnt during training. $\log \sigma$ is used as a regularization term to avoid trivial solutions. For the sake of simplicity, we have omitted the superscript of $\sigma$.

## 2.2. Pipeline of scale calculation

The pipeline of scale calculation is shown in Figure 7. Firstly, numbers and lines are processed respectively. The

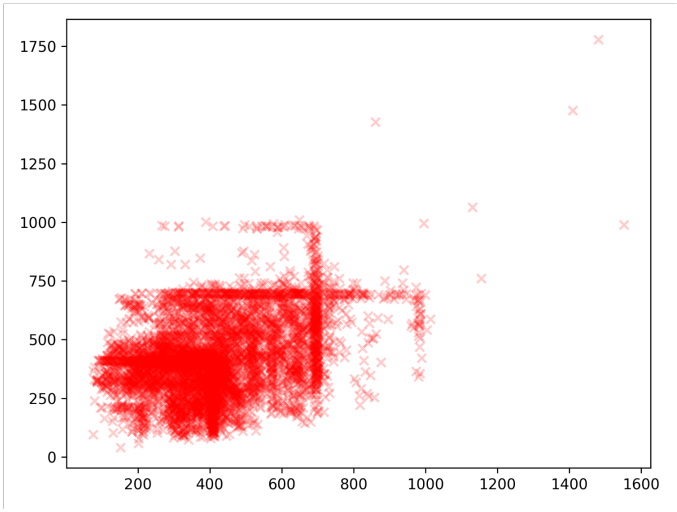


Figure 5. Cropped image resolutions of RFP dataset.

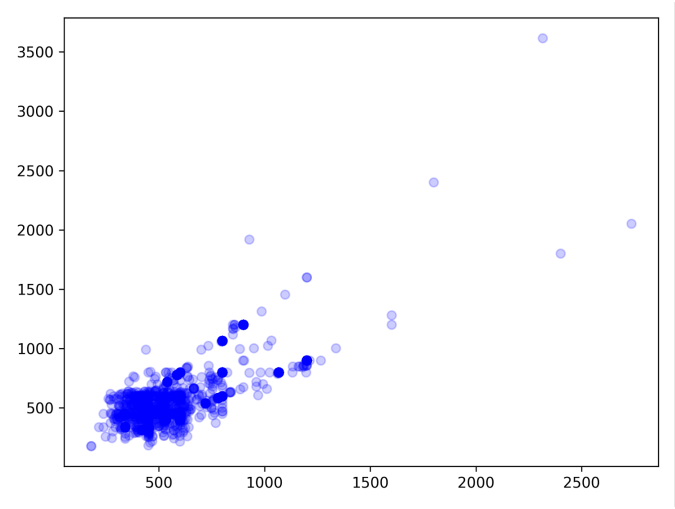


Figure 6. Source image resolutions of RFP dataset.

numbers recognition module consists of number area detection, digits recognition and digits quantity regression part. Number area detection part can detect all number areas on the floor plan images. Digits recognition module could recognize each digit in number areas. Two results are obtained in this part. One output is digits, and the other is number of digits. In order to ensure the quality of number area detection results, a double inspection mechanism has been established. We use the digits quantity regression module to calculate the other result of the number of digits in the number area. If the results of the above two quantities are the same, the number area will be selected as the input of the next module. While processing the digital part, the line segment is processed in parallel. Endpoints of line segments are detected by a deep neural network. Lines are generated using

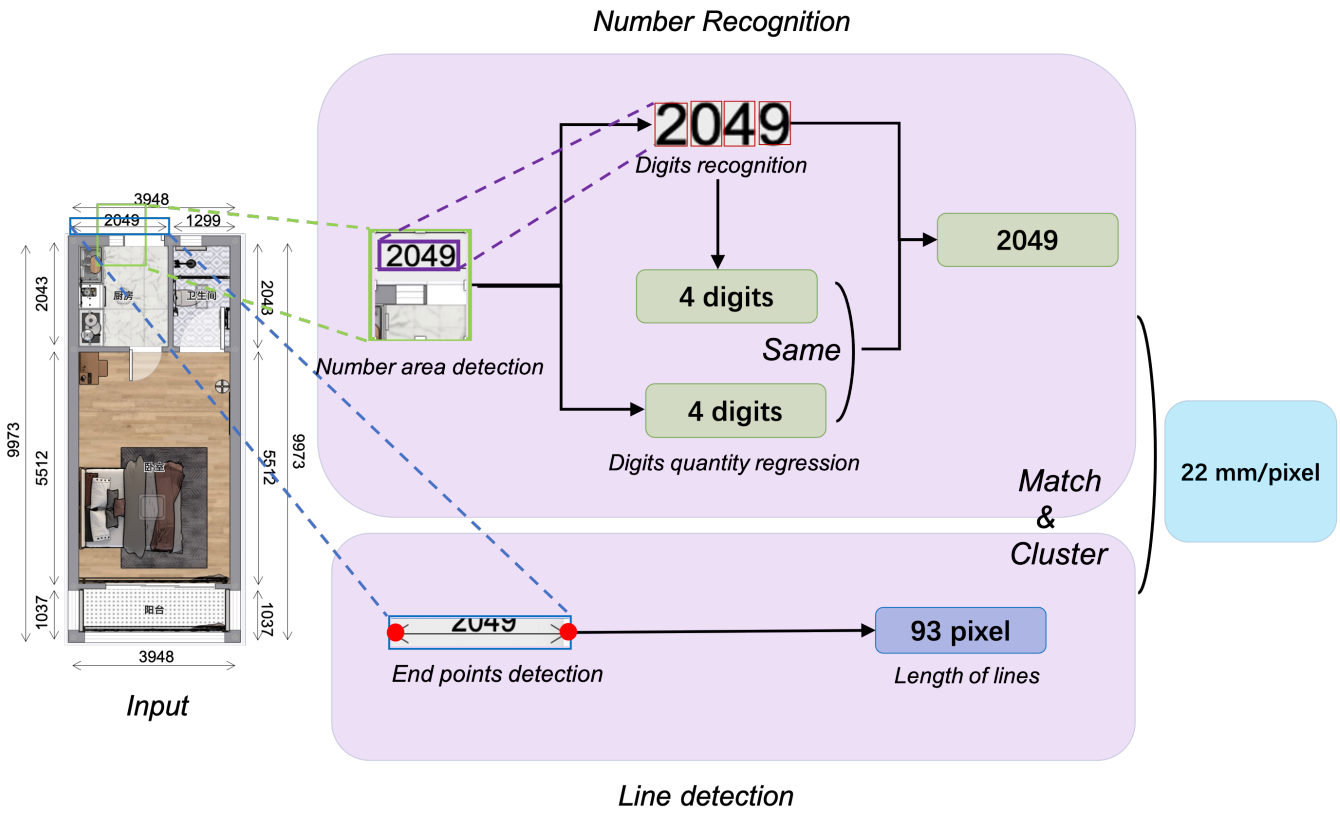


Figure 7. Pipeline of scale recognition.

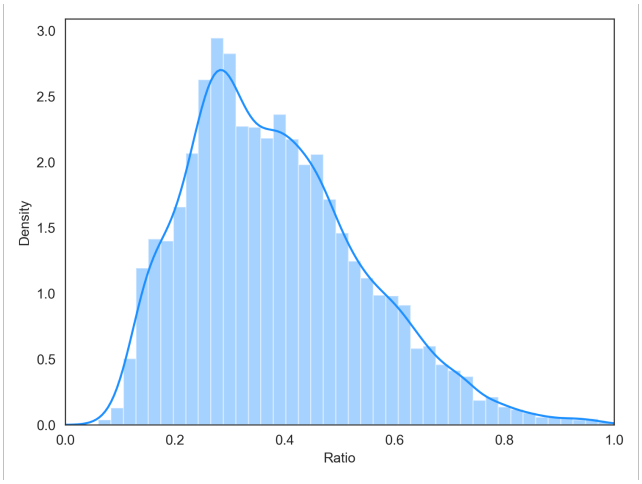


Figure 8. Effective area ratio distribution. The area under the curve is 1.0.

those endpoints. Line segments and numbers are matched to calculate scale. Each pair of lines and numbers can get one scale. All those results are clustered by k-means method. After cluster process, the class which owns the largest quantity are reserved and others are removed as noise. Results in the reserved class are averaged to obtain scale. The line segment detection part is modeled as an FCN [14] network with deconvolution layers. The task focuses on detecting endpoints of line segments. The architecture is shown in Figure 9.

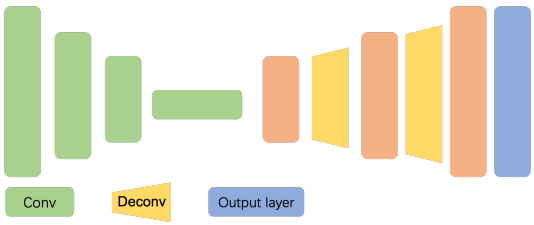


Figure 9. Architecture of line segment detection.

### 2.3. Pixel wall width calculation

Based on the result of pixel segmentation, a general method is used to determine the pixel width of the wall to assist the vectorization process. The pixel width of the wall is determined by two factors, one is the image resolution, and the other is the physical size of the house. The relation-

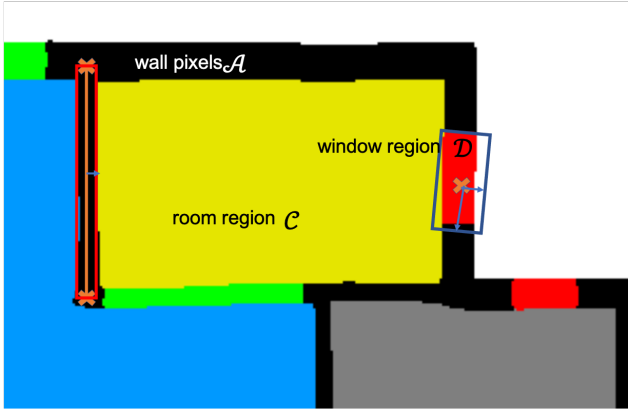


Figure 10. Pixel wall width calculation (right) and vectorized wall width calculation (left).-

ship of the three can be expressed as:

$$\text{Scale} = \frac{\text{physical size}}{\text{image resolution}} \tag{5}$$

Normally, the physical thickness of the wall in the residential floor plan is fixed, such as 0.24m or 0.12m in China. But due to the different scales, the width of the pixels on the floor plan varies greatly. Previous work [13] used picture resolution to determine the threshold, which assumes that all pictures have the same scale. We propose an adaptive method to obtain the pixel width of the wall. Since the walls and doors are on fixed walls, and our segmentation method divides each area (walls, doors and windows pixels) into several parts. So, we extract the bounding box of the doors and windows to get the pixel width of the wall. For the area $\mathcal{D}$ of each door or window, we calculate its median value as the center point $(x, y)$ of the door and window. As shown in Figure 10, we use a rectangle $\mathcal{R}$ to approximate the area of each door and window. The rectangle is parameterized by the rotation angle $\theta$, the width $w$ and the height $h$. The parameters of the rectangle are obtained by optimizing the following formula:

$$\max_{\theta,w,h} \text{IOU}(\text{Rasterized}(\mathcal{R}), \mathcal{D}) \tag{6}$$

After obtaining the length and width of the doors and windows represented by rectangles, we use the median of the width of the doors and windows as the pixel wall width.

### 2.4. Vectorized wall width calculation

After the vectorization is completed, because the thickness of the walls is different, we need to calculate the thickness of each part of the wall. At this time, the coordinates of the starting/ending points of each part of the wall are known, and we use rectangles to approximate the walls. We need to optimize the wall thickness $w$, as shown in Figure 10. The goal is to make the area covered by the rectangle $\mathcal{R}$ overlap the wall pixels $\mathcal{A}$ as much as possible.

$$\max_{w} \text{IOU}(\text{Rasterized}(\mathcal{R}), \mathcal{A}) \tag{7}$$

## 3. Experiments Results

### 3.1. $\mathcal{L}_{\text{alignment}}$ experiments

If the angle between the wall line segment and the x-axis or y-axis is less than 5 degrees, we consider that the line segment is still horizontal or vertical, and if it is greater than 5 degrees, we consider it to be an inclined wall. We counted the proportion of inclined walls in ground truth, our method ("Ours" item of Table 2 in the original paper) and our method without $\mathcal{L}_{\text{alignment}}$ ("Ours-$\mathcal{L}_{\text{alignment}}$" item of Table 2). The ratio of the inclined wall in the ground truth is 2%, the ratio using our method is 4%, and the ratio without $\mathcal{L}_{\text{alignment}}$ is 21%. Therefore, adding $\mathcal{L}_{\text{alignment}}$ to our optimization goal (Formula 15 in the original paper) can effectively avoid the generation of inclined walls.

### 3.2. Scale calculation

Performance is evaluated from two dimensions: positive rate and average error. True samples represents results which error rates are under specific threshold (2%, 5%, and 10% in our experiments), compared with ground truth. Positive rate demonstrates rate of true samples. Average error of all samples is 4.6mm. An ablation analysis of clustering module and digits quantity regression module are investigated, details are shown in Table 2. Those two modules bring more than 25% rate increase than systems without them.

### 3.3. Structural elements extraction

An ablation analysis of the loss in structural elements extraction is presented here.

- w/o D: the ROI detection module is removed. The input is the original input image instead of the cropped one.
- w/ D + ce: The input image is cropped by ROI detection module. We only use cross entropy loss and opening regression loss.
- w/ D + ce + aaf: The input image is cropped by ROI detection module. We use cross-entropy loss, affinity field loss and opening regression loss.
- Our version: we use the full version of our algorithm.

Table 3 shows the comparison between the above schemes and the complete method. Models are trained and tested on the RFP dataset. From Table 3, we can see that the

| Kmeans | Digits quantities regression | Average error (mm) | Positive rate | | |
|---|---|---|---|---|---|
| | | | 2% | 5% | 10% |
| ✓ | ✓ | **4.6** | **0.72** | **0.83** | **0.86** |
| ✓ | ✗ | 5.9 | 0.69 | 0.80 | 0.83 |
| ✗ | ✓ | 7.1 | 0.57 | 0.74 | 0.80 |
| ✗ | ✗ | 24.4 | 0.45 | 0.58 | 0.66 |

Table 2. Scale results. ✓stands for with, ✗stands for without

| | w/o D | w/ D + ce | w/ D + ce + aaf | Our full version |
|---|---|---|---|---|
| *Acc.* | 0.81 | 0.92 | 0.95 | 0.96 |
| *mIoU* | 0.77 | 0.80 | 0.83 | 0.84 |
| *fwIoU* | 0.83 | 0.92 | 0.93 | 0.95 |

Table 3. Ablation study of structural elements extraction.

structural elements extraction module performs best when equipped with affinity field loss and multi task loss. Since DeepLabv3+ can meet the requirements as a basic network, and the network structure is not the innovation of this article, we do not study the influence of the different segmentation network on the results.

### 3.4. Text and symbols detection

$AP_{0.5}$ and $AP_{0.75}$ are calculated, according to different text classes. $AP_{0.5}$ and $AP_{0.75}$ of class bathroom is 0.98 and 0.85 respectively. Compared with other classes, size of class bathroom is larger, and it contains more contextual information. That is the reason that why it performs well. Class other shows a low-level performance, because it covers all situations which does not belong to room types we defined clearly. $AP_{0.75}$ is significantly lower than $AP_{0.5}$. The possible reason is that text are small objects. Therefore, tiny location error could lead huge IOU error. Symbols detection performance are also described by $AP_{0.5}$ and $AP_{0.75}$. Quantitative results of text detection are shown in Table 4.

Every symbol class performs well, because those objects are large and not hard to recognize. $AP_{0.5}$ of all of classes are more than 0.9. The decline of $AP_{0.75}$ is less than text detection module. Results are shown in Table 5.

| Type | $AP_{0.5}$ | $AP_{0.75}$ |
|---|---|---|
| Living room | 0.94 | 0.79 |
| Bedroom | 0.91 | 0.78 |
| Kitchen | 0.97 | 0.84 |
| Bathroom | 0.98 | 0.85 |
| Library | 0.89 | 0.67 |
| Balcony | 0.95 | 0.81 |
| Other | 0.82 | 0.62 |

Table 4. Average precision of text detection.

| Type | $AP_{0.5}$ | $AP_{0.75}$ |
|---|---|---|
| Sofa Combo (Living room) | 0.94 | 0.88 |
| Dining Table Combo (Living room) | 0.94 | 0.89 |
| Bed Combo (Bedroom) | 0.95 | 0.91 |
| Cooker Combo (Kitchen) | 0.93 | 0.83 |
| Toiletries Combo (Bathroom) | 0.91 | 0.80 |

Table 5. Average precision of symbols detection.

### 3.5. Generalization experiments on more data

We use dataset rent3d [12] and Cubi-Casa5K [10] to evaluate generalization capability of our system. Figure 11 shows results. Although the images styles of in those datasets are significantly different from ours, structure and room types could still be identified accurately.

## 4. Experimental Implementation Details

**ROI.** We use the entire RFP data set. During training and testing, input images are resized to 300×300. We use mean average precision as metric in measuring the accuracy of object detectors.

**Structure info extraction.** We use the entire RFP data set and follow the division of test set and training set. We resized the detected image to 512×512, and used a batch size of 8 with a synchronized batch normalization implementation [5] cross multiple GPUs. We use the Adam optimizer to update the parameters and train the network with a fixed learning rate of 1e-3. The number of epochs is 300. For the other methods we compare, we use the original hyper-parameters that have been reported in their original papers to train their networks. In order to obtain the best recognition results, we further evaluated the results in each training period and only reported the best one.

**Text and Symbol detection.** Text detection model and symbols detection model are similar. They are both based on YOLOv4 model. The degree of alignment of features and anchors influence network performance significantly [16, 8]. However, YOLOv4 has no ROI pooling or ROI align modules [15] to align feature and anchors. Increasing number of different sizes of anchors, by adjusting the parameter of yolo layer of YOLOv4 [7], is a method to alleviate the problem of misalignment between anchors and features. 5 anchors ,instead of 3, are set in each grid of yolo layer [7]. Furthermore, minimizing difference between size


Figure 11. Floor plan recognition and reconstruction results on rent3d(first three rows) and cubicasa5k(last two rows). From left to right, an input floor plan image, semantic segmentation result with post-processing, reconstructed vector-graphics representation with room type annotation, the corresponding 3D reconstruction model.


Figure 12. More floor plan recognition and reconstruction results. From left to right, an input floor plan image, semantic segmentation result with post-processing, reconstructed vector-graphics representation with room type annotation, the corresponding 3D reconstruction model. The original images come from Lianjia [3] (row 1, 5), Kujiale [2] (row 2) and Fangtianjia [1] (row 3, 4).


Figure 13. More floor plan recognition and reconstruction results. From left to right, an input floor plan image, semantic segmentation result with post-processing, reconstructed vector-graphics representation with room type annotation, the corresponding 3D reconstruction model. The original images come from Lianjia [3] (row 1, 2, 5), Sanweijia [4] (row 3, 4).

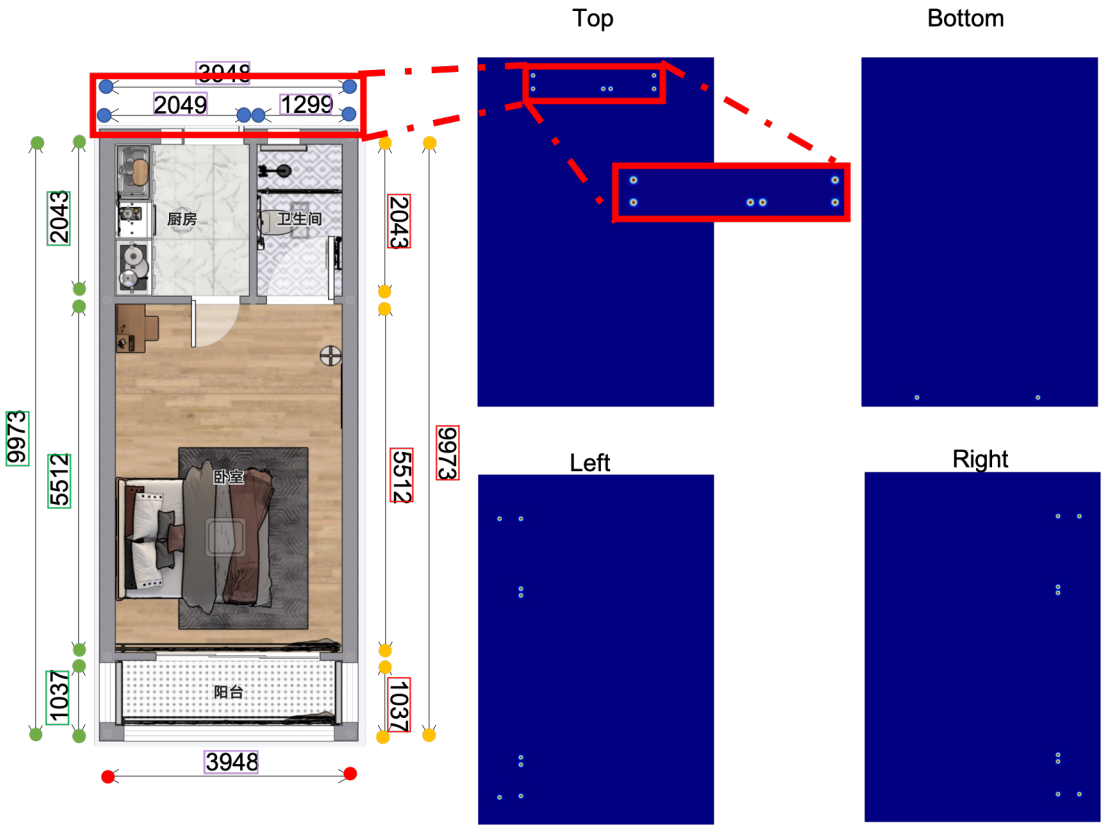


Figure 14. Endpoints of line segments heatmaps.

of anchors and objects could lead to less misalignment issues. Thereby, height and width of anchors are obtained by cluster analysis of bounding boxes of training set using k-means++ [6]. Data augmentation techniques are utilized to enhance performance of detector. Images are rotated to different directions randomly when training, and Gaussian noise is added to image. Height of inputs is resized randomly, and width is same as height. To train text detection model, batch size is 128 and 1000 epochs are processed. To train symbol detection model, batch size is 128 and 1200 epochs are processed. When testing, inputs are resized to 608×608 and sent to models. Outputs are bounding boxes of objects, with location of center points of objects and the class of each object.

**Scale calculation.** To train number area detection model, a multi-scale input strategy is utilized. Height of inputs is adjusted from 480 to 1024 randomly. The interval is 32. The width is the same as the height. Batch size is 128, and 1500 epochs are processed. The backbone, Darknet, of yolov4, is pretrained on MSCOCO dataset [11]. To train endpoints detection model, size of input is 512×512. Labels are heatmaps, are shown in Figure 14. The training batch size is 8, and number of epochs is 500. To train digits recognition model, image scale is fixed to 320×320. The number of training epochs is 1000. To train digits quantity recognition model, the input is output of number area detection part. It is resized to 128×128. The batch size is 128 and 500 epochs are processed.

In testing process, the pipeline is divided to two parts. First part focuses on numbers. The size of the floor plan image need to be adjusted to 608×608, as input of number area detection model. After that, outputs of number area model are resized to 320×320 and passed to digits recognition network. Also, those outputs are adjusted to 128×128 to pass to digit quantity regression network. Quantity of digits is calculated by digit quantity regression network and results of digits recognition network respectively, as shown in Figure 7. Number areas, which acquire the same quantity in processes mentioned above, is chosen to generate number. The second part of pipeline is to achieve line segments detection task. Image is resized to 512×512. After processing by endpoints detection network, heatmaps are obtained. Max pooling is applied in those heatmaps to acquire endpoints. Endpoints which belong to the same line and same class are utilized to generate a line segment. Lines and number areas are matched to acquire scale results. Those results

are clustered to eliminate noise, then averaged to calculate final scale.

## References


[1] Fangtianxia. https://www.fang.com/. 8

[2] Kujiale. https://www.kujiale.com/. 8

[3] Lianjia. https://www.lianjia.com/. 8, 9

[4] Sanweijia. https://www.3vjia.com/. 9

[5] Synchronized-batchnorm. https://github.com/vacancy/Synchronized-BatchNorm-PyTorch. 6

[6] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006. 10

[7] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 6

[8] Yuntao Chen, Chenxia Han, Naiyan Wang, and Zhaoxiang Zhang. Revisiting feature alignment for one-stage object detection. *arXiv preprint arXiv:1908.01570*, 2019. 6

[9] Lluís-Pere de las Heras, Oriol Ramos Terrades, Sergi Robles, and Gemma Sánchez. Cvc-fp and sgt: a new database for structural floor plan analysis and its groundtruthing tool. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(1):15–30, 2015. 1

[10] Ahti Kalervo, Juha Ylioinas, Markus Häikiö, Antti Karhu, and Juho Kannala. Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis. In *Scandinavian Conference on Image Analysis*, pages 28–40. Springer, 2019. 1, 6

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 10

[12] Chenxi Liu, Alexander G Schwing, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Rent3d: Floor-plan priors for monocular layout estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3413–3421, 2015. 6

[13] Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. Raster-to-vector: Revisiting floorplan transformation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2195–2203, 2017. 1, 5

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 4

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 6

[16] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2965–2974, 2019. 6