iPLAN: Interactive and Procedural Layout Planning-Supplementary Material

Feixiang He
University of Leeds, UK
scfh@leeds.ac.uk

Yanlong Huang
University of Leeds, UK
y.l.huang@leeds.ac.uk

He Wang [*]
University of Leeds, UK
h.e.wang@leeds.ac.uk

1. Architecture of BCVAE

The detailed architecture of BCVAE is illustrated in Tab. 1. For a given layout, $\boldsymbol{Q} = \{q_k\}_{k=1}^{K}$ represents room types, where K denotes the number of room types in $\boldsymbol{D}$ and $q_k \in \mathbb{Z}$ corresponds to the number of rooms under the k-th type.

Before feeding $\boldsymbol{Q}$ into BCVAE, a reformulation is implemented. We first determine the largest number of rooms for each type k across the whole dataset and denote it as $q_k^* \in \mathbb{Z}$. Then, for each $q_k \in \boldsymbol{Q}$ ($q_k \leq q_k^*$), we transform it into a q_k^*-D vector, i.e., v_k, whose first q_k elements are set as 1 while the remaining elements as 0. By concatenating all transformed vectors, we can obtain an alternative representation of $\boldsymbol{Q}$, i.e., $\boldsymbol{V} = [\boldsymbol{v}_1^T \boldsymbol{v}_2^T \dots \boldsymbol{v}_K^T]^T$.

We denote the output of BCVAE as $\hat{\boldsymbol{V}}$ and use binary cross entropy as the reconstruction loss:

$$\mathcal{L}_{rec} = \sum_{j=1}^{n\_c} l_j, l_j = -[v_j log \hat{v}_j + (1-v_j) log(1-\hat{v}_j)], \quad (1)$$

where $n\_c = \sum_{i=1}^{K} q_k^*$ represents the length of $\boldsymbol{V}$.

The total loss of BCVAE is:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda D_{KL}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) || \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})), \quad (2)$$

where D_{KL} denotes the Kullback–Leibler (KL) divergence, $\lambda = 0.5$.

2. Post-Processing for Room Partitioning Prediction

After the room partitioning prediction, sometimes gaps may exist between the predicted rooms $\hat{\boldsymbol{R}} = \{\hat{\boldsymbol{r}}_1, \hat{\boldsymbol{r}}_2, \dots, \hat{\boldsymbol{r}}_N\}$. We employ a simple post-processing step to ensure that the interior area of $\boldsymbol{B}$ is fully covered and the room bounding boxes are located within $\boldsymbol{B}$. We formulate it as a generic optimization problem:

$$\arg\min_{\hat{\boldsymbol{R}}} \mathcal{L} = \arg\min_{\hat{\boldsymbol{R}}} \mathcal{L}_{coverage}(\hat{\boldsymbol{R}}, \boldsymbol{B}) + \mathcal{L}_{interior}(\hat{\boldsymbol{R}}, \boldsymbol{B}) \quad (3)$$

where $\mathcal{L}_{coverage}$ and $\mathcal{L}_{interior}$ constrain the spatial consistency between $\boldsymbol{B}$ and the room bounding box set $\hat{\boldsymbol{R}}$.

To explain $\mathcal{L}_{coverage}$ and $\mathcal{L}_{interior}$ clearly, we introduce a distance function $d(p, \boldsymbol{r})$ to measure the coverage of a point p by a box $\boldsymbol{r}$:

$$d(p, \boldsymbol{r}) = \begin{cases} 0, & \text{if } p \in \Omega_{in}(\boldsymbol{r}) \\ \min_{q \in \Omega_{bd}(\boldsymbol{r})} ||p - q||, & otherwise \end{cases} \quad (4)$$

where $\Omega_{in}(\boldsymbol{r})$ denotes the interior area of the box $\boldsymbol{r}$ and $\Omega_{bd}(\boldsymbol{r})$ represents the boundary of $\boldsymbol{r}$.

The coverage loss can be defined as:

$$\mathcal{L}_{coverage}(\hat{\boldsymbol{R}}, \boldsymbol{B}) = \frac{\sum_{p \in \Omega_{in}(\boldsymbol{B})} \min_i d^2(p, \hat{\boldsymbol{r}}_i)}{|\Omega_{in}(\boldsymbol{B})|}, \quad (5)$$

where $|\Omega_{in}|$ is the number of pixels in the set $\Omega_{in}(\boldsymbol{B})$.

The interior loss can be denoted as follows:

$$\mathcal{L}_{interior}(\hat{\boldsymbol{R}}, \boldsymbol{B}) = \frac{\sum_i \sum_{p \in \Omega_{in}(\hat{\boldsymbol{r}}_i)} d^2(p, \hat{\boldsymbol{B}})}{\sum_i |\Omega_{in}(\hat{\boldsymbol{r}}_i)|}, \quad (6)$$

where $\hat{\boldsymbol{B}}$ is the bounding box of the boundary. Note that $\boldsymbol{B} \subseteq \hat{\boldsymbol{B}}$.

Therefore, in the inference stage, we directly adjust the predicted rooms $\hat{\boldsymbol{R}} = \{\hat{\boldsymbol{r}}_1, \hat{\boldsymbol{r}}_2, \dots, \hat{\boldsymbol{r}}_N\}$ by minimizing the loss $\mathcal{L}$ in Eq. (3).

3. Additional Qualitative Comparisons

Fig. 1 and Fig. 2 show qualitative results on RPLAN and LIFULL, respectively. In both datasets, Graph2Plan and Our$_I$ are provided with the full human input (including the boundary, room types and room locations), their generated layouts are expected to be similar to the GT. While it is the case for Our$_I$, it doesn't seem to be so for Graph2Plan. The shaded areas on the layouts produced by Graph2Plan show clear differences from the GT layouts. In contrast, the layouts from Our$_I$ are nearly the same as the GT.

In addition, we also compare iPLAN with other methods on a more challenging dataset, LIFULL. When only the house boundary is provided, Our$_{III}$ outperforms Rplan. Our$_{II}$ corresponds to the case when the house boundary and

[*]Corresponding author

Architecture	Layer	Specification	Output Size
embedding network	$conv\_bn\_relu_1$	$1 \times 16 \times 4 \times 4(s = 2, p = 1)$	$64 \times 64 \times 16$
	$conv\_bn\_relu_2$	$16 \times 16 \times 4 \times 4(s = 2, p = 1)$	$32 \times 32 \times 16$
	$conv\_bn\_relu_3$	$16 \times 32 \times 4 \times 4(s = 2, p = 1)$	$16 \times 16 \times 32$
	$conv\_bn\_relu_4$	$32 \times 32 \times 4 \times 4(s = 2, p = 1)$	$8 \times 8 \times 32$
	$conv\_bn\_relu_5$	$32 \times 16 \times 4 \times 4(s = 2, p = 1)$	$4 \times 4 \times 16$
	$conv\_bn\_relu_6$	$16 \times 16 \times 4 \times 4(s = 2, p = 1)$	$2 \times 2 \times 16$
	$flatten$	N/A	1×64
encoder	$concat$	N/A	$1 \times (n\_c + 64)$
	$linear\_relu1$	$(n\_c + 64) \times 128$	1×128
	$linear\_relu2$	128×64	1×64
	$linear_{31}$	64×32	1×32
	$linear_{32}$	64×32	1×32
decoder	$concat$	N/A	1×96
	$linear\_relu1$	96×96	1×96
	$linear\_relu2$	96×64	1×64
	$linear_3$	$64 \times n\_c$	$1 \times n\_c$
	$sigmoid$	N/A	$1 \times n\_c$

Table 1. The BCVAE architectural specification. s and p respectively denote stride and padding. n_c is the dimension of house type. Convolution kernels and layer output are separately specified by $(N_{in} \times N_{out} \times W \times H)$ and $(W \times H \times C)$.

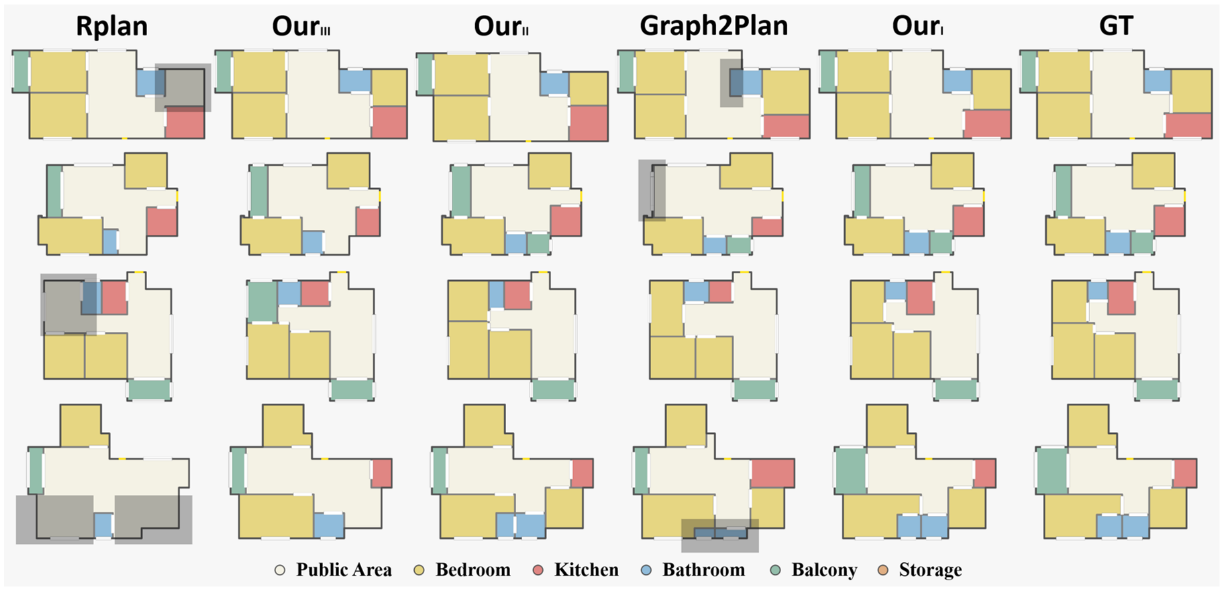

Figure 1. Qualitative comparisons on RPLAN. Shaded areas indicate design choices that are questionable.

the room types are known, which achieves slightly better predictions than Our$_{III}$ as more information is fed. Furthermore, if the full human input is provided, Our$_I$ performs better than Our$_{II}$. Note that Our$_I$ is superior to Graph2Plan which is also fed with the full human input.

4. Additional Generalization Evaluations

More generalization results on RPLAN are presented in Fig. 3. The first row and second row correspond to Rplan and Our$_{III}$, respectively. Our$_{III}$ achieves better results. Consistent with our analysis in the main paper, Rplan is prone to splitting the public area into two main areas, causing potential inconvenience for family activities (the first

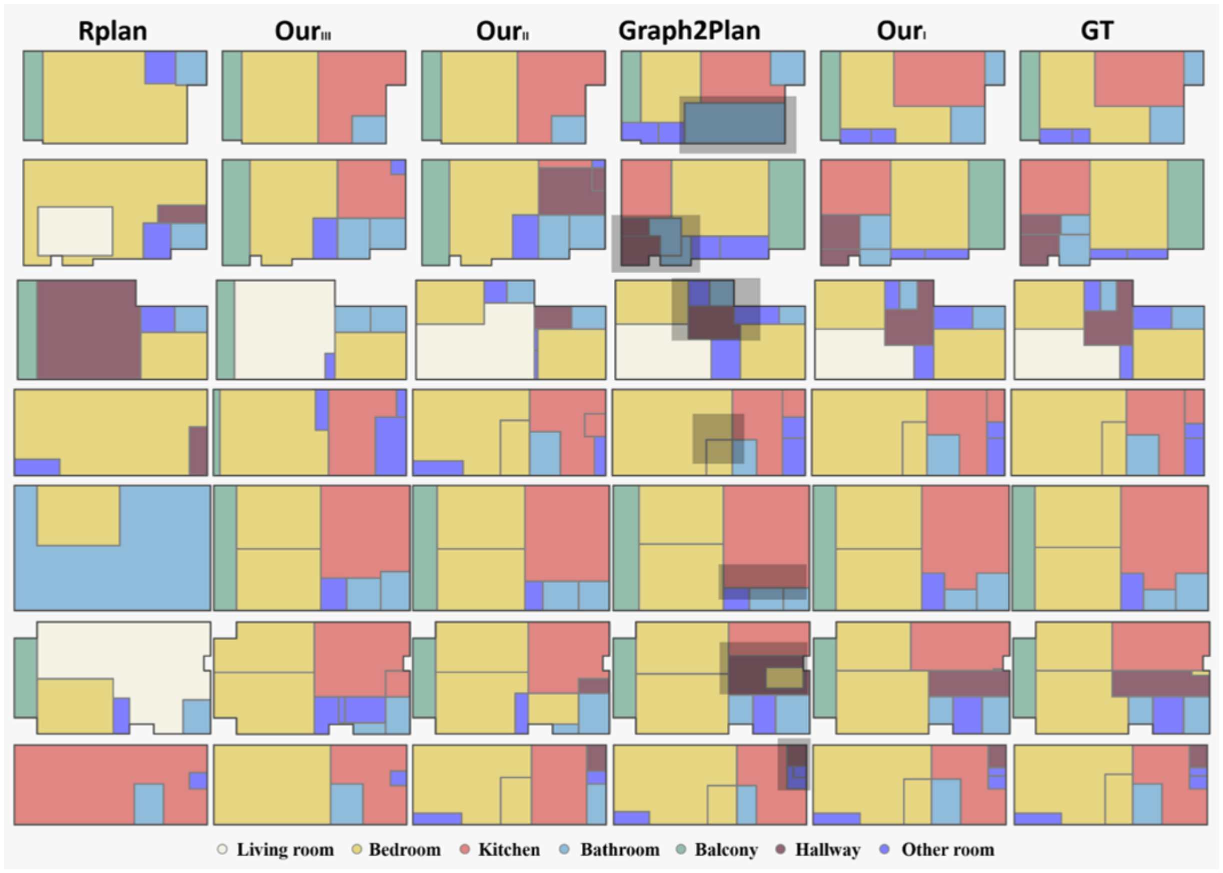

Figure 2. Qualitative comparisons on LIFULL. Shaded areas indicate design choices that are questionable.

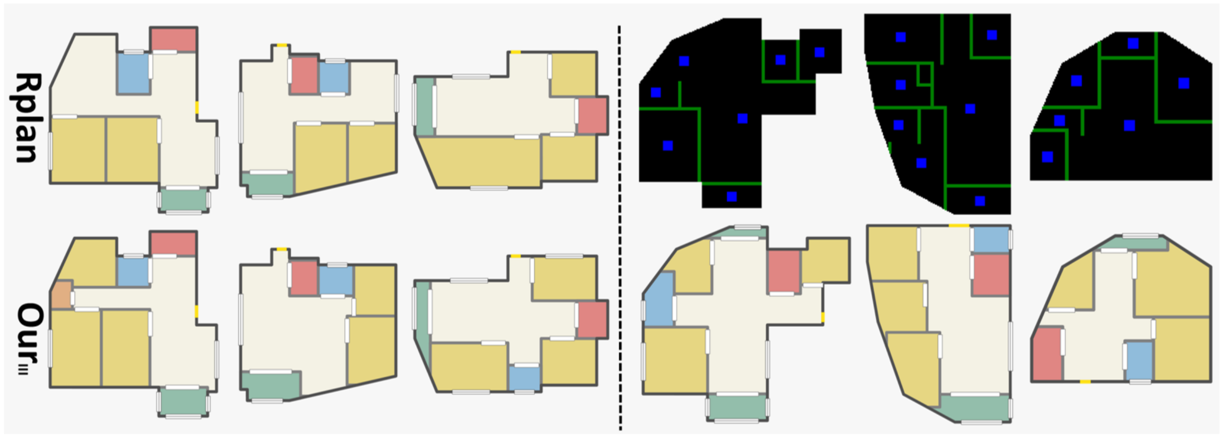

Figure 3. Layouts with non-axis aligned edges. Left: successful examples. Right: Rplan fails but our model succeeds.

two columns in Fig. 3). Sometimes, Rplan also fails to plan a bathroom in the layout (the third column in Fig. 3). Moreover, on some boundaries, Rplan fails to design the layouts (the last three columns in Fig. 3). In general, Our$_{III}$ outperforms Rplan when the boundary is non-axis aligned.

5. Implementation Details

We have implemented iPLAN in PyTorch. All models are trained and tested on a NVIDIA GeForce RTX 2080 Ti. It takes about two hours to train BCVAE, two days to optimize the room-locating network and one day to train the room area prediction model.