Raster-to-Vector: Revisiting Floorplan Transformation Supplementary material

Chen Liu
Washington University in St. Louis
chenliu@wustl.edu

Jiajun Wu
Massachusetts Institute of Technology
jiajunwu@mit.edu

Pushmeet Kohli†
DeepMind
pushmeet@google.com

Yasutaka Furukawa*
Simon Fraser University
furukawa@sfu.ca

The supplementary document provides more experimental results and some algorithmic details on our pre-processing and post-processing steps.

1. More experimental results

Figures 1, 2, 3, and 4 show more experimental results on randomly sampled floorplan images in our dataset, demonstrating their challenges and varying styles. Figures 5 and 6 show our results on other image sources, in particular, Rent3D dataset [2] and Google Image Search, respectively. Fig. 7 shows some typical failure cases.

Lastly, the main paper has provided quantitative evaluations. We here provide qualitative evaluations comparing Ahmed *et al*. [1] and our approaches with or without Integer Programming, as well as ablation experiment examples in Fig. 8.

*At Washington University in St. Louis at the time of the project.
†At Microsoft Research Redmond at the time of the project.

Figure 1: Additional experimental results on test images in our database. The three numbers indicate the same statistics as in the main paper: 22/23/24 mean that there are 23 correct predictions to make, where our algorithm makes 24 predictions and collected 22 correct ones.

Figure 2: Continued.

Figure 3: Continued.

Figure 4: Continued.

Figure 5: Additional results on images in Rent3D dataset [2].

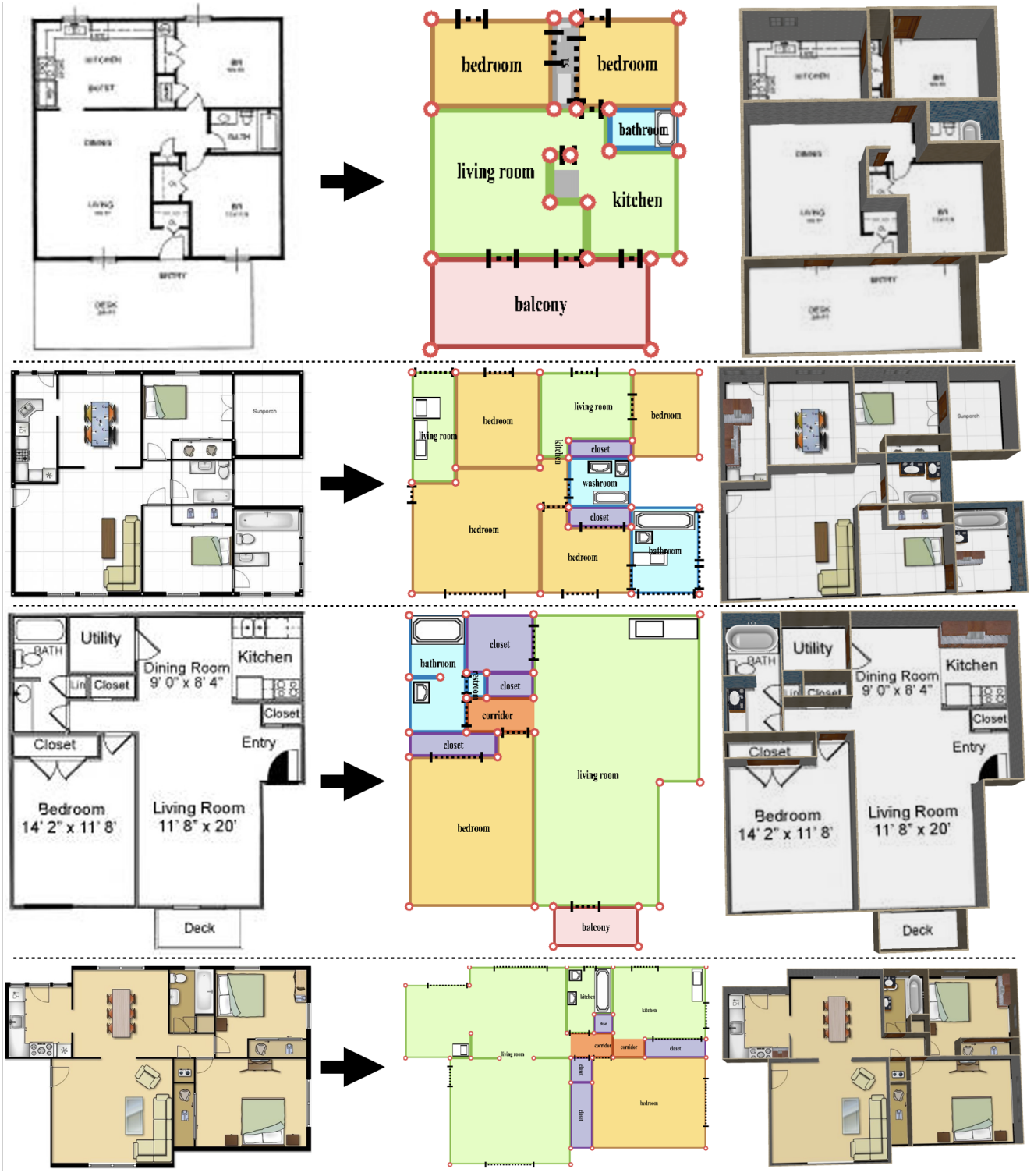

Figure 6: Additional results on images from Google image search via a keyword ”floorplan”.

Figure 7: Typical failure cases. Relying on Manhattan assumption, our method is unable to detect walls which are neither horizontal nor vertical as shown in the first example. This is the major limitation of our method, which could be addressed by adding a special wall junction type. In the second example, our method puts a wall which does not exist. In some cases, our method misses I-junctions as shown in the third example and detailed wall structure as shown in the fourth example.

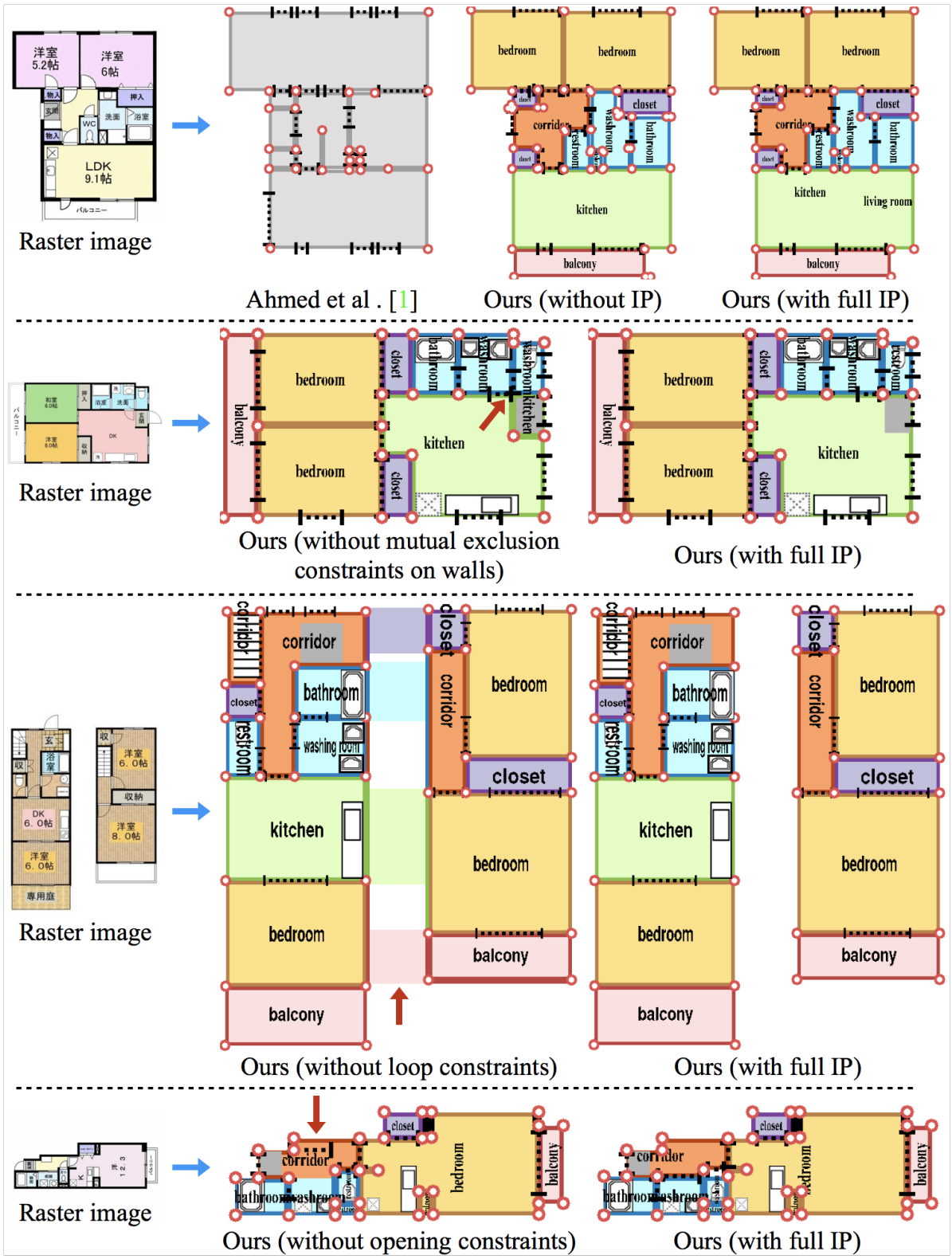

Figure 8: The first row shows qualitative comparison between Ahmed *et al*. [1], ours (without IP), and ours (with full IP). The last rows show ablation experiment examples for specific constraints in IP. The example in the second row indicates that, without mutual exclusion constraints on walls, two walls could intersect (as marked by red arrow). Without loop constraints, the exterior might not form a loop as shown in the third row. The fourth row shows that, without opening constraints, an opening could appear in the middle of a room, which causes severe visual artifacts.

2. Pre-processing details

This section provides algorithmic details on how we obtain human annotations then converts them into our own representation for training.

Our primitive representation is expressive and enables our learning-based inversion. However, it is trivial for a human subject to explicitly denote the location and type of all the junctions, and associate each wall with room labels. To ease the human annotation process, we ask human subjects to draw a line representing either a wall or an opening, draw a rectangle, and pick an object type for each object, or attach a label at a specific location. We need to convert such annotations to our junction layer representation for training the network, and to final representation for evaluation. We convert the annotation information to junction layer representation as follows.

- Openings and objects primitives are directly annotated, and thus we simply read junctions from the annotation (either endpoints of an opening or corners of an icon).
- An annotated wall often consists of multiple wall primitives. Thus we determine a wall junction by looking at its local connections of walls (i.e., checking if a wall exists on its up, bottom, left, or right side).

The junction layer representation is sufficient for network training. For evaluation, we first convert the junction layer representation to get information for walls, openings and icons in the final representation. The only missing piece in the final representation is rooms. We split closed polygons (formed by walls) in a similar manner explained in Section 3. Note that now we have room labels specified at certain locations instead of associated with each wall primitive. In this setting, we find a pair of facing walls which has one and only one room label attached in between, and assign this room label to the region between this pair of walls. The rest is the same with before.

3. Post-processing details

This section provides algorithmic details on how we split a closed polygon whose walls are associated with more than one room-type. This information is not essential but used to change the color/texture of walls and the floor geometry in our 2D and 3D visualizations.

We allow certain rooms not to be a closed polygon (e.g., a kitchen and a living room appear in the same closed polygon) (see Fig. 9 for an example). In such cases, with room labels associated with each wall (i.e., each edge of the polygon), it is unclear how to divide the closed polygon based on functionality. We address this issue based on a practical heuristic. For each pair of facing walls which have the same room label, we find the rectangle between these two walls, and assign the room label to the intersection region between the rectangle and the closed polygon as shown in Fig. 9. If a pixel is assigned with multiple room labels, then we pick the label with the highest priority, in the order of living room, kitchen, washing room, and corridor (other room types enforce loop constraints). For a pixel with no label assigned, we assign it to the label which appear in this polygon and has the highest priority. After assigning all the pixels inside the polygon with one room label, we find the connected components based on the assignment to be the final room regions.

Figure 9: A floorplan example where kitchen and living room appear in the same closed polygon. A pair of facing walls (red line and purple line) with the same room label (e.g., living room) determines that pixels colored with green belong to living room.

References

[1] S. Ahmed, M. Liwicki, M. Weber, and A. Dengel. Improved automatic analysis of architectural floor plans. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 864–869. IEEE, 2011. 1, 9

[2] C. Liu, A. G. Schwing, K. Kundu, R. Urtasun, and S. 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. 1, 6