General 3D Room Layout from a Single View by Render-and-Compare – Supplementary Material

Sinisa Stekovic[1], Shreyas Hampali[1], Mahdi Rad[1], Sayan Deb Sarkar[1], Friedrich Fraundorfer[1], and Vincent Lepetit[1,2]

[1] Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria
[2] Université Paris-Est, École des Ponts ParisTech, Paris, France
{sinisa.stekovic, hampali, rad, sayan.sarkar, fraundorfer, lepetit}@icg.tugraz.at
Project page: https://www.tugraz.at/index.php?id=40222

1 Results on ScanNet-Layout

Figures 1 to 10 show additional qualitative results on our ScanNet-Layout benchmark. We also submitted a video that provides a better visualization of our 3D reconstructions.

2 Computation Times

Table 1 details the computation times for the different steps of our method. Our current implementation is not optimized and runs on a single CPU, except for the network predictions which run on a NVIDIA GeForce 1080 Ti. As the computation times for one image depend on the number of estimated layout components, we provide the average times over the images of our ScanNet-Layout benchmark.

Depending on the networks that are utilized during inference for different scenarios, the run-time slightly varies, but the time for network predictions never exceeds $0.3\ s$. Most notably, the bottleneck of our computations is during the plane fitting step, which can be significantly optimized. When using PlaneRCNN to estimate the plane parameters, this step is skipped entirely.

	Time (s)
Networks Predictions	< 0.3
Fitting Layout Planes with RANSAC	8.2 (*)
Finding Candidate Corners and Edges	0.08
Optimal Polygon Search	1.26
Iterative Refinement	3.12

Table 1. Run-time performance. (*) This step can be significantly improved in our implementation. When using PlaneRCNN to estimate the plane parameters, it is skipped entirely.

3 Implementation details

3.1 Handling Noisy Observations

To handle noisy observations in plane detection, semantic segmentation, and depth predictions, we rely on the simple heuristics given below:

- We consider a planar region predicted by PlaneRCNN to be part of the layout only if more than 0.4 pixels locations of the plane belong to one of the layout categories—wall, floor, ceiling.
- After calculating the planes parameters using RANSAC, we discard planes that are inconsistent with the depth map for the input image across the corresponding planar region. If the number of outliers during the RANSAC procedure is larger than 30% of the planar region, we discard the corresponding plane. As outliers, we consider the points with distance larger than $0.01m$ to the fitted plane. When plane parameters are predicted using PlaneRCNN, we skip this step.
- PlaneRCNN might falsely predict multiple planar regions in place of one. In order to handle such mistakes, we merge layout planes that are parallel, by measuring the angles between the normals vectors. At the same time, the planes are only merged if the planes have similar camera offsets, measured by the absolute difference. In case of RGBD image as input, we set these hyper-parameters to 15° and 0.3 respectively. When using RGB image as input, to compensate for the noise in plane estimation, we set these parameters to 30° and 0.5. However, it is also possible that a given scene consists of multiple layout components with same plane parameters. Hence, we merge the planes only if they are neighbouring layout planes. In other words, there are no other planar regions from M_L in-between them. For the merged planes, the new parameters are simply calculated as a mean of the two planes.
- Layout refinement is performed only if the discrepancy between the rendered layout depth and the depth map for the input image is larger than a threshold value for at least 1000 image locations. We set this threshold value to $0.1m$ when utilizing measured depth, and $0.5m$ for predicted depths.

3.2 Networks for Plane Detection, Semantic Segmentation, and Depth Prediction

For all networks, we consider 640×480 input images. PlaneRCNN [5] is the original network that was pre-trained on the ScanNet dataset [2] by the authors. We trained DeepLabV3+ [1] on the SUNRGB-D dataset [8], a collection of datasets, made of an original dataset and additional datasets previously published [7,4,9]. We took care not to include any of the test images from the NYUv2 303 dataset during training. We retrained SharpNet [6] on the NYUv2 dataset [10,7] without the test images from the NYUv2 303 dataset.

Hence, by merging the outputs of the networks we obtain the planes parameters for the layout components of the scene. PlaneRCNN outputs instance plane segmentation for all of the visible planes in the scene. Next, we infer the planar regions in the image belonging to different layout components as intersections between the predicted planar regions and the predicted semantic regions corresponding to the layout categories—wall, floor, ceiling. Using depth information, from sensor or predicted, we then perform plane fitting for each of the planar layout regions to obtain the final planes parameters. Alternatively, when depth information is not available, we rely on PlaneRCNN that directly outputs the planes parameters.

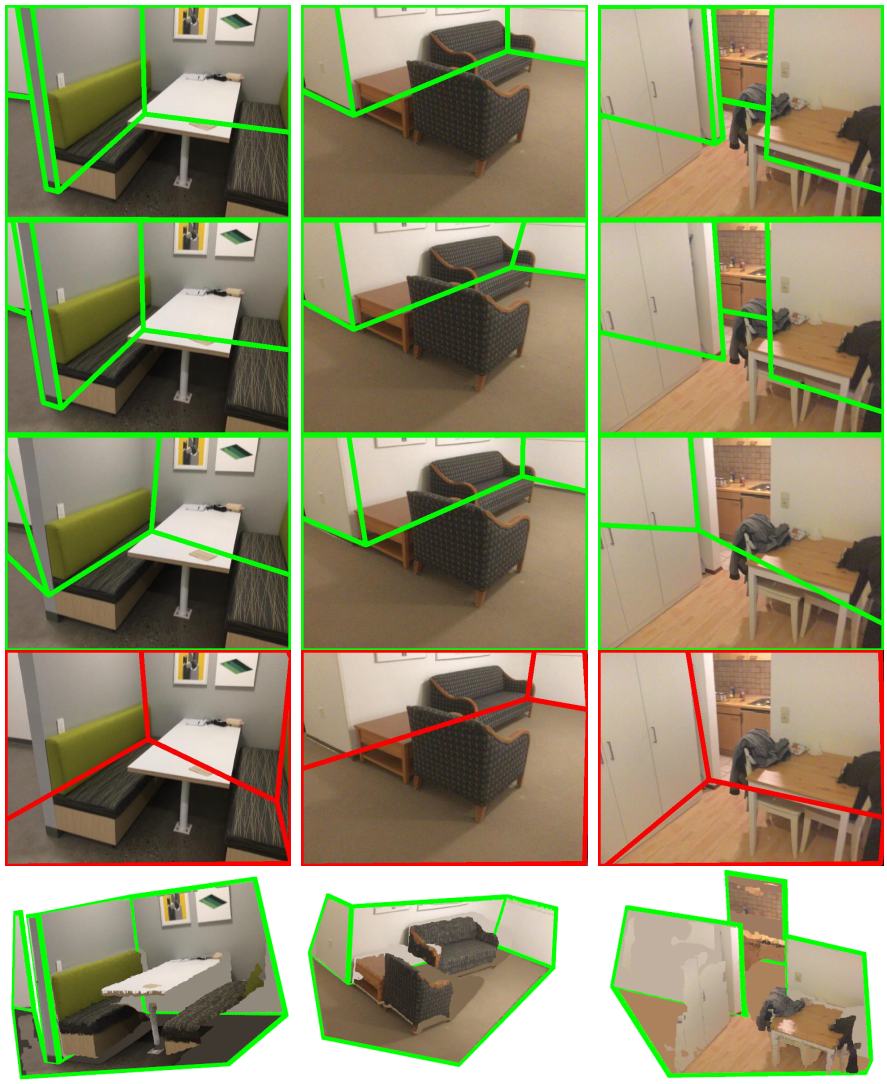

Fig. 1. Examples from our ScanNet-Layout benchmark (Part 1). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts.

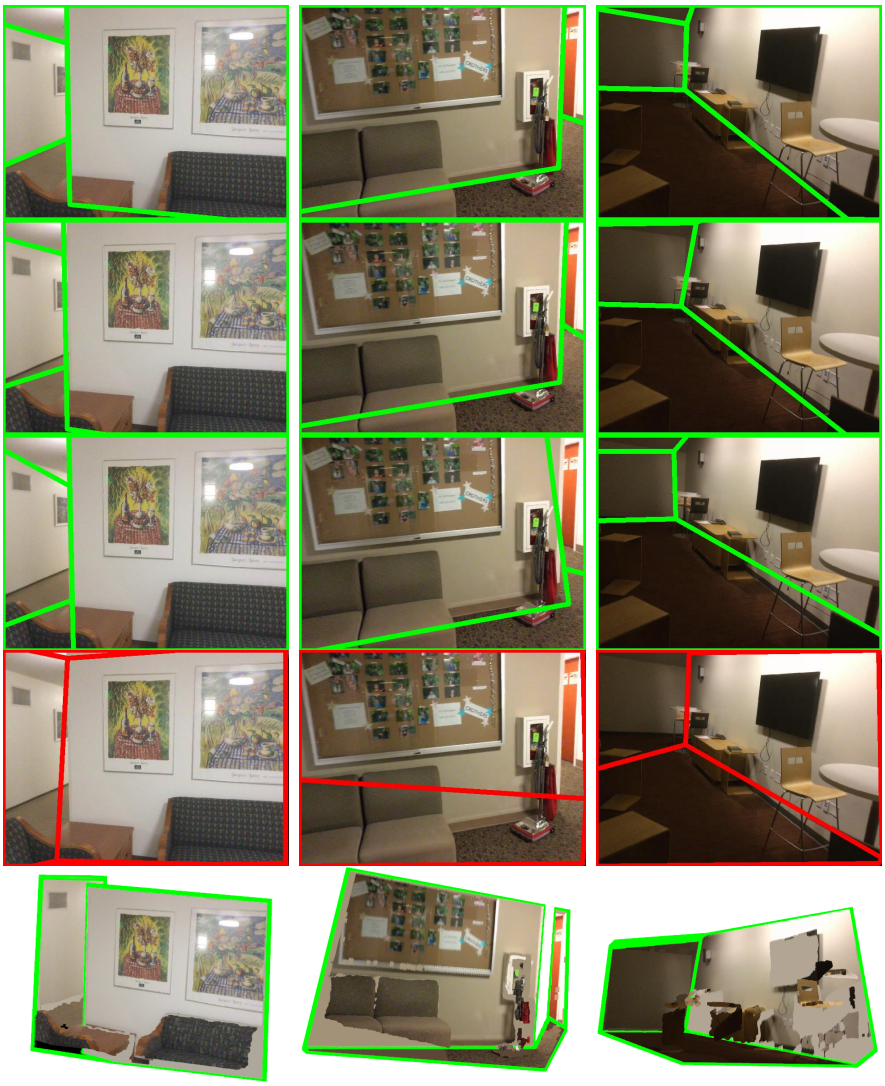

Fig. 2. Examples from our ScanNet-Layout benchmark (Part 2). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts.

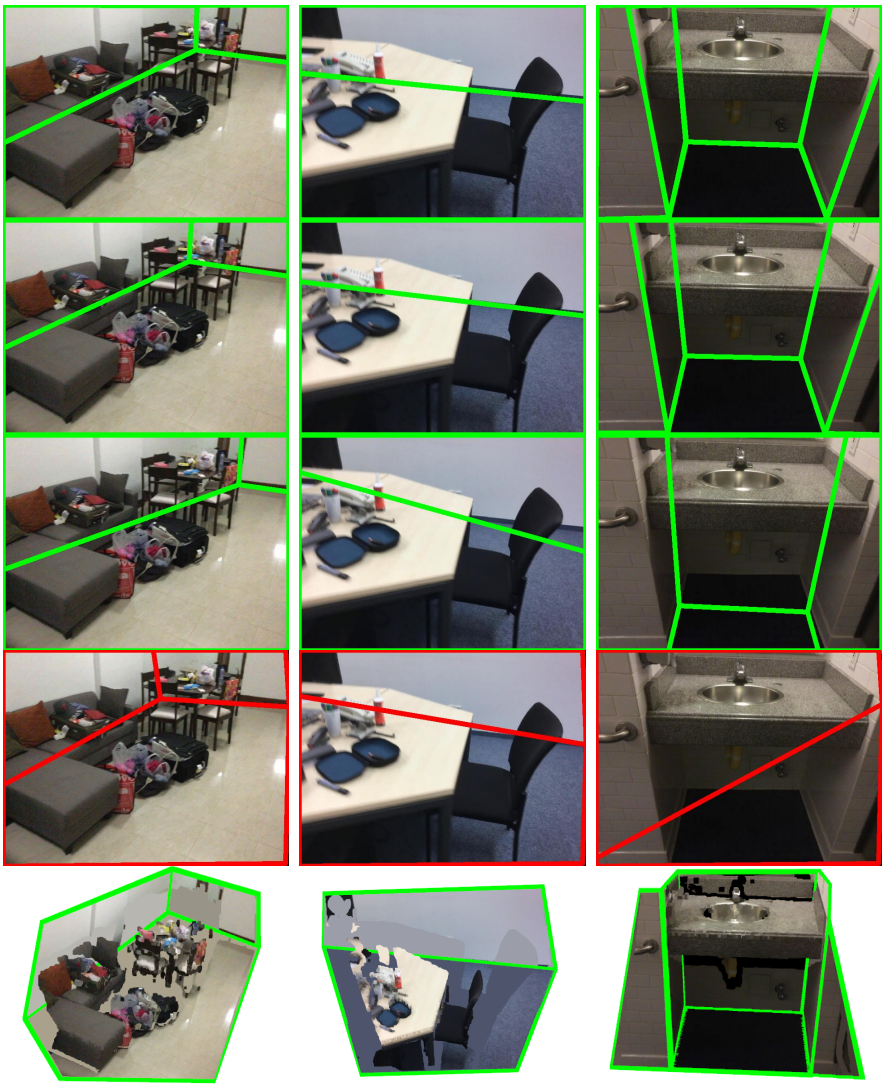

Fig. 3. Examples from our ScanNet-Layout benchmark (Part 3). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts.

Fig. 4. Examples from our ScanNet-Layout benchmark (Part 4). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts.

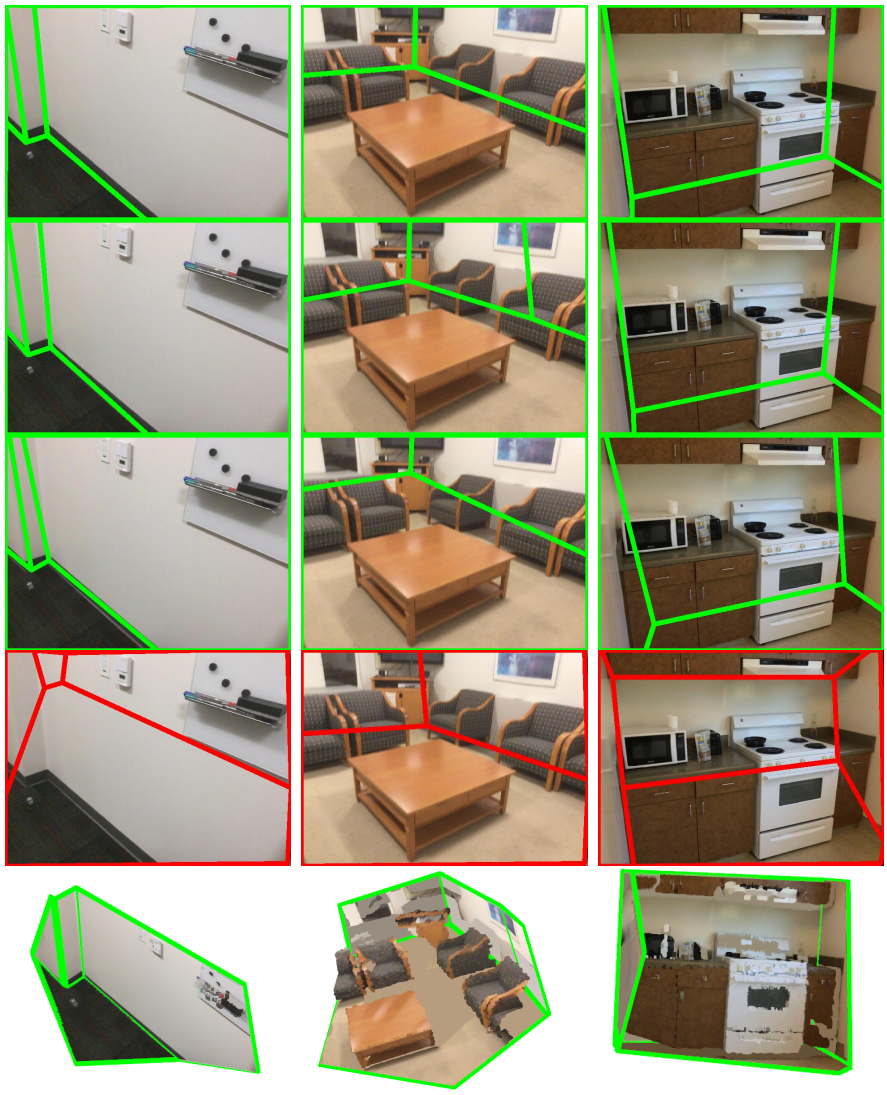

Fig. 5. Examples from our ScanNet-Layout benchmark (Part 5). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts.

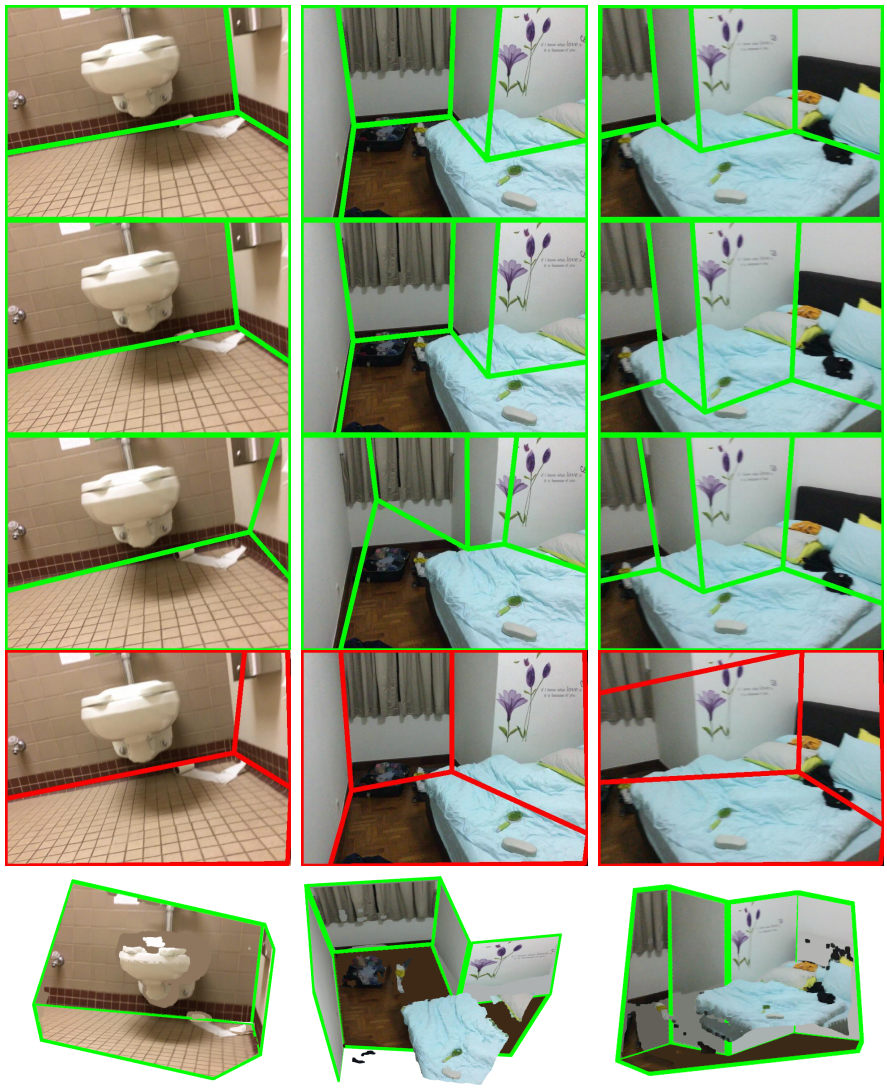

Fig. 6. Examples from our ScanNet-Layout benchmark (Part 6). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts.

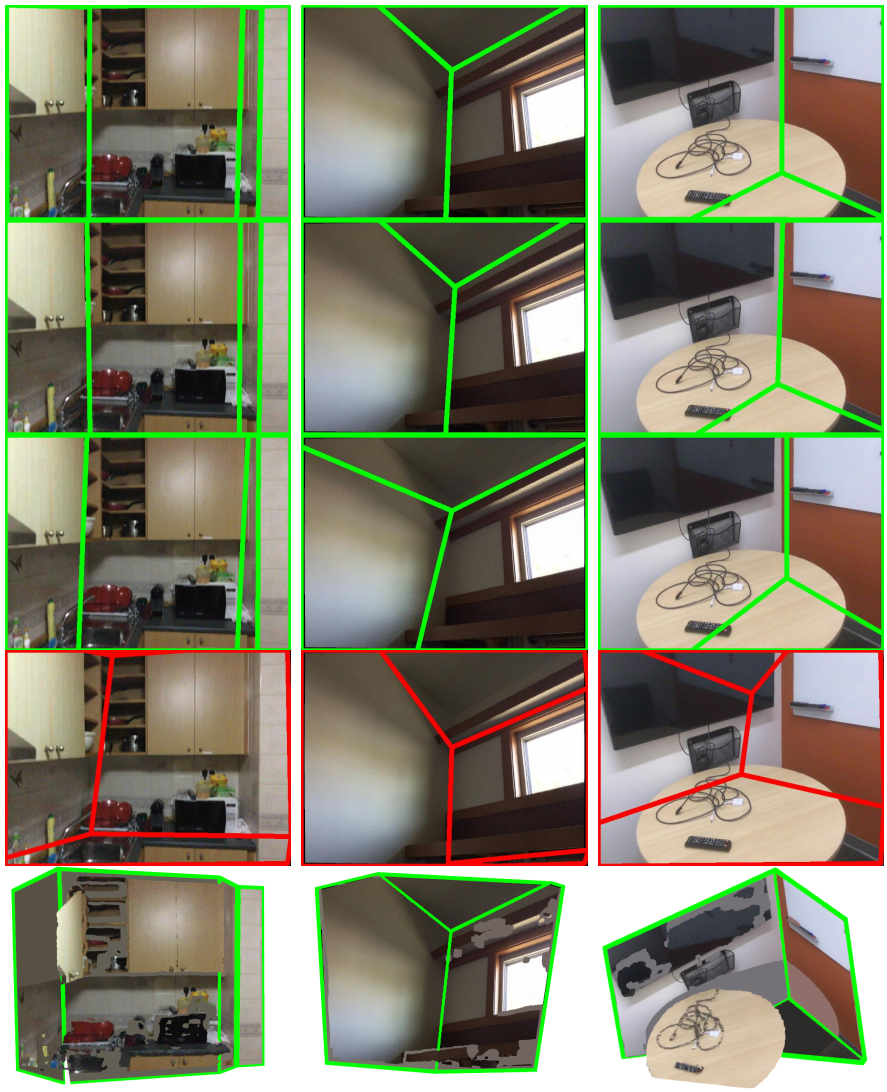

Fig. 7. Examples from our ScanNet-Layout benchmark (Part 7). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts.

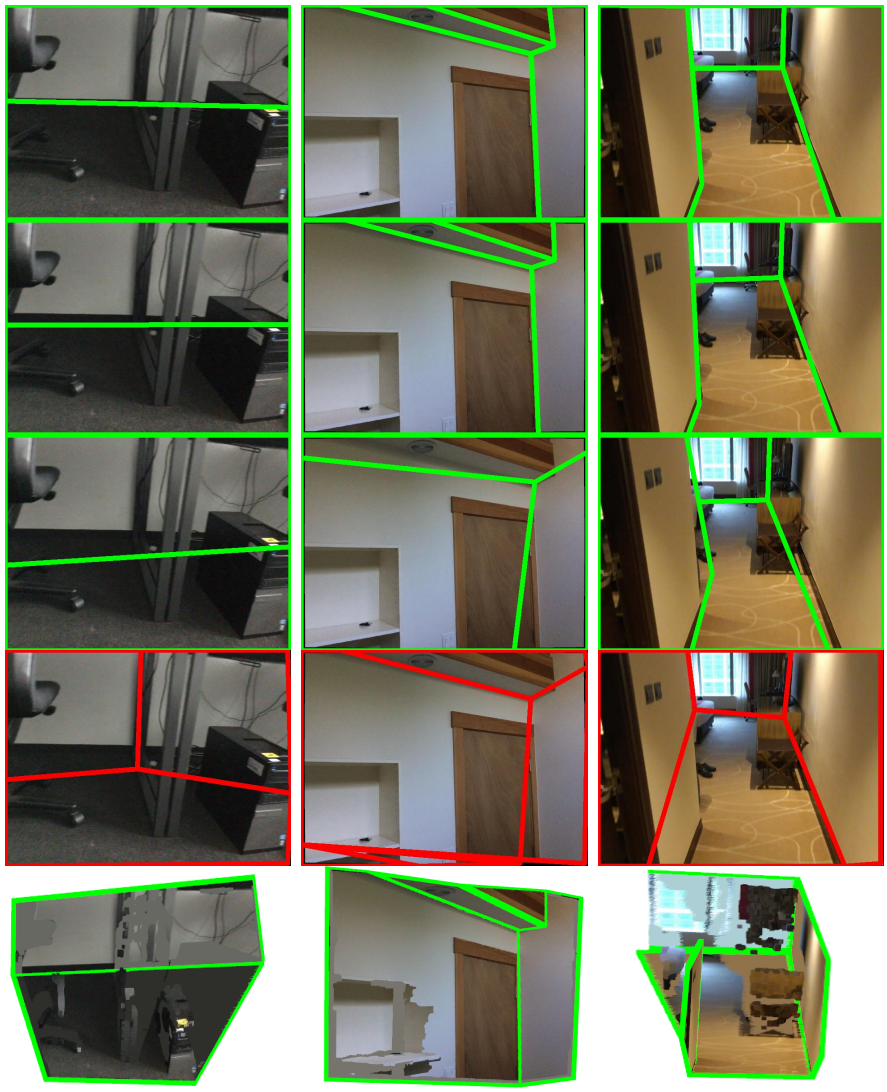

Fig. 8. Examples from our ScanNet-Layout benchmark (Part 8). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts.

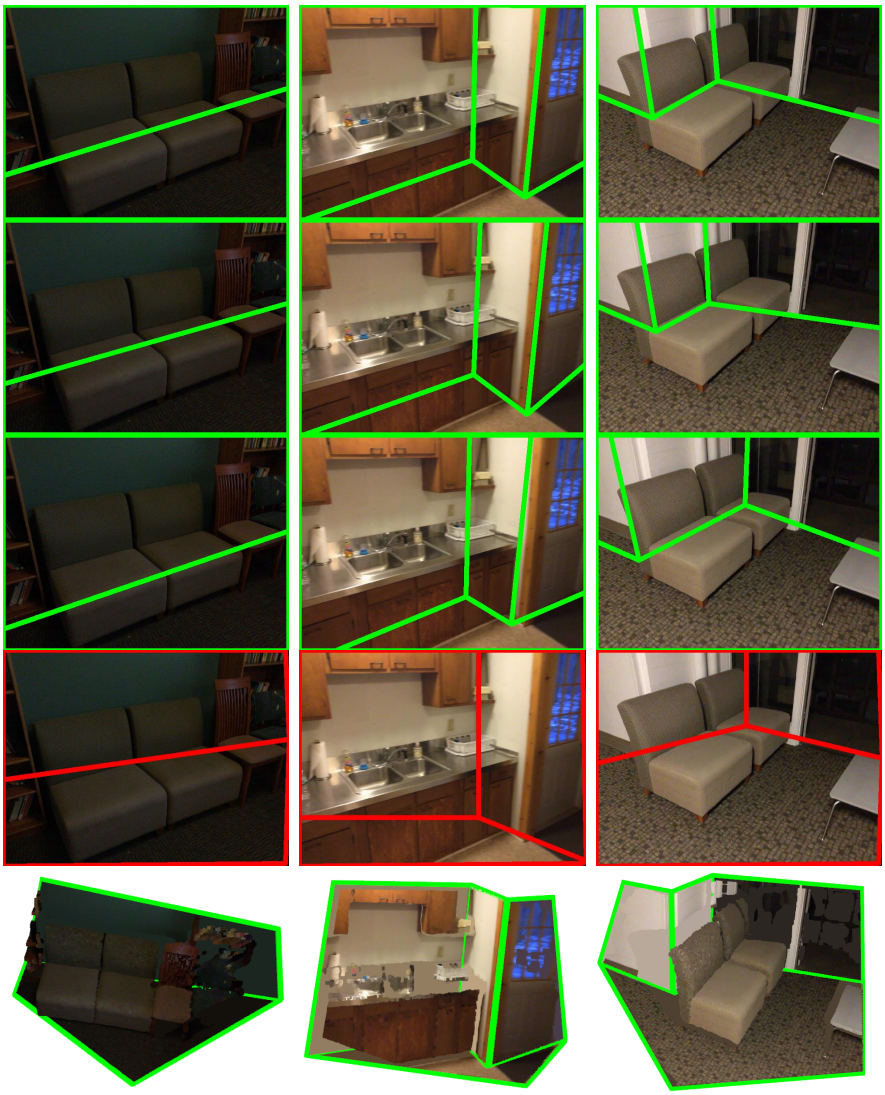

Fig. 9. Examples from our ScanNet-Layout benchmark (Part 9). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts.

Fig. 10. Examples from our ScanNet-Layout benchmark (Part 10). For each example, first row: 2D annotations; second row: our results with RGBD input; third row: our results with RGB input; fourth row: results from Hirzer's cuboid method [3]; fifth row: 3D visualization of our results with RGBD input. For the 3D visualization, we inpaint the occluded parts the 3D polygons with the mean color of their visible parts. For these cases, the floor was entirely occluded and we assume the camera was $1.5m$ above the floor for the visualization.

References

1. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In: European Conference on Computer Vision (2018)
2. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Niessner, M.: Scannet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In: Conference on Computer Vision and Pattern Recognition (2017)
3. Hirzer, M., Roth, P.M., Lepetit, V.: Smart Hypothesis Generation for Efficient and Robust Room Layout Estimation. IEEE Winter Conference on Applications of Computer Vision (2020)
4. Janoch, A., Karayev, S., Jia, Y., Barron, J.T., Fritz, M., Saenko, K., Darrell, T.: A Category-Level 3D Object Dataset: Putting the Kinect to Work. In: Consumer Depth Cameras for Computer Vision (2013)
5. Liu, C., Kim, K., Gu, J., Furukawa, Y., Kautz, J.: Planercnn: 3D Plane Detection and Reconstruction from a Single Image. In: Conference on Computer Vision and Pattern Recognition (2019)
6. Ramamonjisoa, M., Lepetit, V.: SharpNet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation. In: International Conference on Computer Vision Workshops (2019)
7. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor Segmentation and Support Inference from RGBD Images. In: European Conference on Computer Vision (2012)
8. Song, S., Lichtenberg, S.P., Xiao, J.: SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite. In: Conference on Computer Vision and Pattern Recognition (2015)
9. Xiao, J., Owens, A., Torralba, A.: SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels. In: ICCV (2013)
10. Zhang, J., Kan, C., Schwing, A.G., Urtasun, R.: Estimating the 3D Layout of Indoor Scenes and Its Clutter from Depth Sensors. In: International Conference on Computer Vision (2013)