PanoContext: Supplementary Material

Yinda Zhang Shuran Song Ping Tan† Jianxiong Xiao

Princeton University †Simon Fraser University
http://panocontext.cs.princeton.edu

In this supplementary material, we first explain some details of the algorithm in Section 1, and then talk about how we obtain the 3D ground truth from 2D human annotation in Section 2. We present more evaluation results in Section 3, and show more visualization of the final results in Section 4.

1 Algorithm Details

1.1 Room layout sampling

We assume the room layout could be represented by a vanishing direction aligned cuboid. We parameterize the 3D position of an axis-aligned cuboid by 6 parameters: $(x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max})$, which are the minimal and maximal coordinates of all 8 cuboid vertices. For room layout estimation, we sample some line segments detected on panorama and take them as constraints to solve for the 6 parameters.

The 3D size of the room is proportional to the camera height, and therefore our 3D reconstruction is up to a scale. Each edge of the cuboid gives one equation with the 6 knowns, so 5 among 12 edges of the cuboid have to be known to uniquely determine the cuboid. Therefore, we sample 5 line segments as a minimal solution to generate a room layout hypothesis.

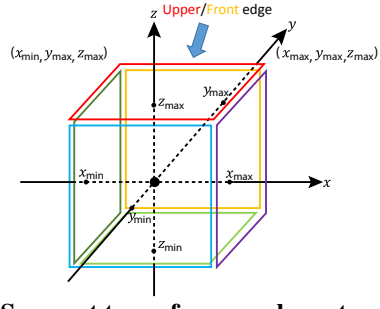


Fig. 1: **Segment types for room layout sampling.**

Suppose we stand inside a cuboid, which is the room, and facing towards one vanishing direction, each face of the cuboid can be assigned with one of the types: *upper, down, left, right, front, back*, depends on its relative location to us. Fig. 1 illustrates the assignment of face type. The three axes corresponds to three vanishing directions. Each color corresponds to a type as indicated by the text. Correspondingly, every edge can get two types from its two adjacent faces. Geometrically, there are totally 12 possible different combinations of types, and the 12 edges of a valid cuboid must contain one of

each combination. On the other hand, each line segments detected from panorama can also be assigned two types according two their relative location with the three vanishing point. So sampling 5 line segments, one from each type combination, is necessary to generate a cuboid room layout hypothesis.

Suppose we sample a line segment $\mathbf{l} = \langle \mathbf{n}, \theta, \phi \rangle$ as the edge with type *(upper,front)*, the plane spanned by $(x_{\min}, y_{\max}, z_{\max})$ and $(x_{\max}, y_{\max}, z_{\max})$ must take the same normal direction as the line segment. This provides a linear constraint as follow:

$$n_y \cdot y_{max} + n_z \cdot z_{max} = 0 \tag{1}$$

in which n_y, n_z are y, z component of $\mathbf{n}$. Constraints from edges of different type combination would be in the same mathematical form but with two different cuboid parameters as we defined above, which are listed in Table 1.

Table 1: Edge type and contrained parameters

Edge type	Constrain on	Edge type	Constrain on	Edge type	Constrain on
upper/front	$y_{\max}, z_{\max}$	down/front	$y_{\max}, z_{\min}$	left/front	$x_{\min}, y_{\max}$
upper/left	$x_{\min}, z_{\max}$	down/left	$x_{\min}, z_{\min}$	right/front	$x_{\max}, y_{\max}$
upper/back	$y_{\min}, z_{\max}$	down/back	$y_{\min}, z_{\min}$	right/back	$x_{\max}, y_{\min}$
upper/right	$x_{\max}, z_{\max}$	down/right	$x_{\max}, z_{\min}$	left/back	$x_{\min}, y_{\min}$

Constraints from 5 line segments lead to a linear system with 5 equations. To solve it for unique solution up to a scale, the 5 equations have to be independent. We check all the $C_{12}^5 = 792$ cases of 5-edge combination, and find 408 degenerative cases, i.e. rank of linear system is smaller than 5. So in practice, we randomly choose a non-degenerative 5-edge case, and randomly select one line segment for each type combination in the case. This will guarantee that the chosen 5 line segments will form a unique room layout up to a scale.

1.2 Design of rectangle detector

Our detector consists of several deformable parts, which is similar as DPM but without the spring like constraints. We design 8 parts on corner, middle of edge, and center of the rectangles. We use the HOG feature with 8×8 pixel per block, and the size of each part is 5×5 blocks. Each part detector is trained separately. The 8 parts are combined into an integrate rectangle detector, but arranged with multiple placement to cope with different aspect ratio, as shown in Fig. 2. We uniformly sample 25 aspect ratios from 0.2 to 5, hence the size of the rectangle detector ranges from 15×15 to 15×55 blocks.

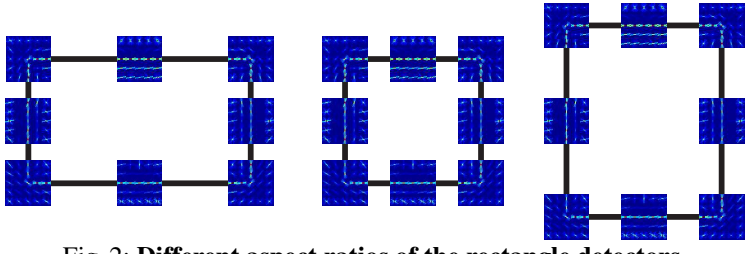
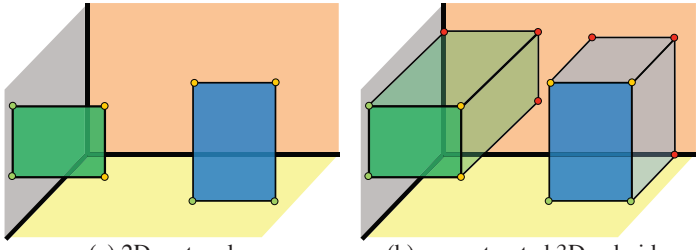


Fig. 2: **Different aspect ratios of the rectangle detectors.**

Although there are 25 rectangle detectors in our model, the detection is very efficient because they share the same part templates. During the testing, we first run each part detectors with sliding window and get their responses on the testing image. Then the score of different parts will be added up according to a part placement to get the score for the corresponding rectangle detector. Therefore, we only need to run sliding window convolution once, and scores of all rectangle detectors can be obtained in constant time.

1.3 Bottom up hypothesis generation: from 2D rectangle to 3D cuboid

The 2D rectangles and cuboid projections detected on panorama are already aligned with vanishing directions. We then convert them to axis-aligned 3D cuboids as object hypotheses. In order to retrieve 3D information from a single 2D image, we first take a room layout hypothesis, and assume that all cuboids are attached to a wall or stand on the floor. Then the 3D shape and location of a 2D rectangle can be computed by studying its correlation with the given room layout, such as in Fig. 3 (a), the green rectangle can only be attached to the left wall, and the blue rectangle can only stand on the floor. In the following, we will talk about how to generate 3D cuboid from a single 2D rectangle, continued with a more complicated case with two rectangles. For the cuboid projections, because they are two or three orthogonal rectangles that are perfectly connected, they can be processed in the same way as two rectangle case.



(a) 2D rectangles (b) reconstructed 3D cuboids

Fig. 3: **Example of one rectangle.**

Given a room layout and a rectangle, we can reconstruct one corresponding cuboid attached to two walls, including ceiling and floor. We take the green rectangle in Fig. 3 (a) as an example. We know the two green points are the on the left wall. Because the room layout is given, the 3D plane equation of the left wall is known. Each point on the panorama corresponds to a ray in 3D space. So that we can get the 3D coordinates of two green points by computing the intersection of their corresponding rays with the left wall plane. Because the rectangle is assumed to be axis aligned, its normal direction is also known. With the 3D coordinates of the green points, we can compute the plane function for the green rectangle. With the same method, we can sequentially compute the 3D coordinates of yellow points and then red points by computing intersection of a plane with a ray. Finally, we can know the 3D coordinates of all the cuboid vertices as shown in Fig. 3 (b).

If two perpendicular rectangles matches perfectly on one edge, they can generate two different cuboid based on different understanding of their relative location to the

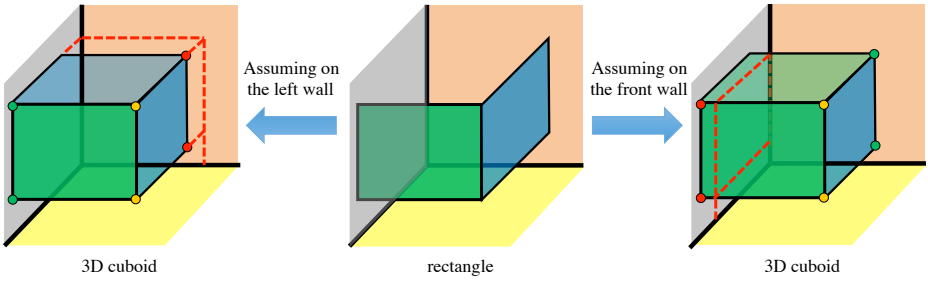


Fig. 4: **Example of two rectangle.**

room layout. In Fig. 4, the corresponding 3D cuboid could be attached to the left wall or the right wall, depends on whether the green rectangle or the blue rectangle is contacting with the wall. Under each assumption, the method above can be applied to solve the 3D position of the cuboid vertices. Two resulting cuboids are shown in Fig. 4. The left one assumes the blue rectangle is attached to the wall, so that it is partially out of the room layout. The right one assumes the green rectangle is attached to the wall, so it leaves some empty space with the front wall. In practice, we first keep cuboids generated under all possible assumptions, and then filter out those cuboids obviously outside the room.

1.4 Room alignment: ground truth room transformation

Room alignment is to test whether a sampled room is structurally reasonable. Intuitively, we can compare the sampled room with all ground truth and use the matching cost to measure how likely the sampled room would appear in ground truth. Because the number of training data is quite limited, to increase the space of possible room layout, we apply transformations on our ground truth to generate more rooms that are still reasonable in the current setting.

The transformation we apply on ground truth room are basically changing the size and aspect ratio of the floor map. To guarantee that the transformed rooms are still reasonable, we follow these rules during the room resizing:

1. The size of the object will not change.
2. Object will keep on attaching to the wall/corner as it is in ground truth room.
3. Contacting objects keep connected. If two objects are connected initially, they will be grouped together and considered as a single object during the room resizing.
4. The empty space on floor has to be connected.
5. Nothing would occupy the space right in front of a door.

We further provide two ways to decide object location during the room resizing. We normalize the coordinates of objects with the size of the room, and maintain these relative coordinates. Alternatively, we maintain the absolute distance of all objects to a particular wall, which will generate more structurally reasonable rooms. Because in our algorithm we assumed a camera height, which may not be exactly true, we further apply universal scaling transformation on the size of the room. Typically, one ground truth room would be transformed to $1000 \sim 3000$ different, yet still reasonable, rooms.

During the testing, a sampled room will compare with each transformed ground truth rooms. Intuitively, we should find the best transformation registering two rooms

and compute difference between them. Considering the efficiency issue, however, we register two rooms by moving the center of the room layout together. Then for each pair of the objects, one from sampled room (obj_s) and the other from a transformed ground truth (obj_t), we compute their distance as the sum of the center distance, volume intersection over union after aligning object centers, and consistency of the object semantic label:

$$D(\text{obj}_s, \text{obj}_t) = ||\mathbf{c}_s - \mathbf{c}_t|| + \min(w_s, w_t)\min(h_s, h_t)\min(l_s, l_t) + \lambda[T(obj_s) \neq T(obj_t)]$$

where $\mathbf{c} = (x, y, z) + (w, h, l)/2$ is the center of the object, $T(\text{obj})$ means the semantic label of obj, λ is a weight on semantic label consistency. Based on the object distance, we search for a bipartite matching to find the minimal matches between objects from two rooms, and use the cost of the matching as the room alignment score.

2 Ground Truth Reconstruction from 2D to 3D

The raw ground truth data is annotated in 2D panorama images. Similar as hypothesis generation, we generate perfect 3D cuboid/rectangle representation from 2D annotation, because the 3D model is the most natural representation of real 3D world and it facilitates the learning of context information. However, it is more difficult than object hypothesis generation. First, the annotations are not geometrically perfect rectangles on panorama. Human shows poor performance in telling vanishing direction and maintaining orthogonal relations on panorama. So 2D image location of the vertices of the cuboids and rectangles are usually with big noise, and solving points one by one would be unstable. Second, some objects do not align with vanishing directions, so that we may not have sufficient constraints as before. Third, the room layout is unknown. Because the reconstruction of indoor objects is heavily depends the room layout, even small error in room layout would greatly affect the location and shape of the objects. Last but not least, the interaction between objects and room could be more complicated. In a typical room, nothing should be partially outside the room. Objects like bed and table would be more likely to be pushed against a room corner instead of leaving a small gap with a wall. Some objects may stand on the top of other objects, like tv on the desk. All these constraints make it impossible to solve each object separately. To tackle all problems mentioned above, we estimate the 3D cuboids/rectangles for all objects and the room jointly, and solve the problem by a non-linear optimization.

Optimization Due to the gravity, we only allow each object to rotate around a vertical axis. So a 3D cuboid, both for object and for room layout, can be represented by a 7 dimensional vector: $\text{obj} = (x, y, z, w, h, l, r)$, in which x, y, z are the 3D coordinates of a cuboid vertex, w, h, l is the size of the cuboid, and r is the rotation angle in horizontal planes. A ground truth room R^* consists of a set of objects $\text{obj}_1, \ldots, \text{obj}_n$ and a big room layout obj^r. The reconstruction of ground truth room becomes to minimize reprojection error between R and the raw 2D annotation on panorama:

$$R^* = \arg\min_R \sum_{i=1,\ldots,n} (P(\text{obj}_i) - A_i) + P(\text{obj}^r) - A^r \tag{2}$$

in which $P(\text{obj})$ means the 2D projection of obj to the panorama, A_i means the annotation of obj_i, and A^r means the annotation of room layout obj^r. Because of the horizontal rotation, the problem is non-linear and could only be solved for a local minimum. So a good initial, for each object and the room layout, is essential to get a final good result.

Initial estimation and constraints We estimate obj for each object individually with the similar way as mentioned in Section 1.3. According to the relation between the room layout and objects, a 3D room is built in the following steps:

1. Room layout. We build the room layout firstly as it is required for constructing other objects.
2. Objects that are confident about their relation with the room layout. If the lower four vertices of a cuboid are all involved in floor region of the room layout and far away from the room corner, the cuboid is sure to be on the ground. Similarly, we can identify objects that confidently attach to a wall. These objects can be reconstructed by the method mentioned in Section 1.3.
3. Other objects. For the other objects, each of them may has multiple different relations with the room layout or other objects, which could be standing on ground, attaching to s wall, or standing on another object. We choose the most likely rules for each object and solve there 3D coordinates.

After the whole room being constructed, we slightly adjust the room to obey our prior knowledge about a reasonable room. This adjustment also suggests some constraints for the overall optimization as in Equation 2. Specifically,

1. If the rotation angle of an object is very small, e.g. $r < 10^\circ$, the r will be fixed to 0°. This object will be considered to be axis-aligned and is not allowed to rotate anymore in the later optimization.
2. If an object penetrates into a wall, it will be pushed fully inside the room layout. Moreover, if this object is axis-aligned, it is considered to be attached on the penetrated walls.
3. If the distance of an axis aligned cuboid to a wall is smaller than 20cm, it is considered to be attached on the wall.

The constraints from all the objects are written with the parameters of R and added into optimization 2. We use the adjusted room as the initial solution and search for a local minimum. Empirically, this initial is good enough, and the optimization could converge to in a few second.

3 Evaluation

We evaluate the performance of global data-driven sampling and local refinement with three tasks: object detection, semantic labeling, and room layout estimation. Some numbers are reported in the paper. Here, we provide the table for the remaining numbers.

In the object detection, we take the best scene hypothesis ranked by scene SVM as the detection, and compute the precision and recall for each object category in the

same way as PASCAL VOC. The only difference is that the intersection over union is defined on densely uniformly sampled vectors on the direction sphere. Table 2 shows the performance of both bedroom and living room scene. Though the local refinement is expected to slightly adjust the scene, the detection performance for some most important object categories, e.g. bed, nightstand, and mirror in bedroom, door, coffee table, and sofa in living room, are all significantly improved.

Table 2: Object detection performance

(a) bedroom

object type	bed	desk	window	mirror	door	nightstand	wardrobe	cabinet	painting	tv	chair	sofa
global precision (%)	62.16	40.28	24.00	28.89	30.65	27.50	13.89	0.00	54.79	25.00	6.15	0
global recall (%)	69.70	36.25	22.64	31.71	25.68	33.33	17.86	0.00	34.48	27.59	5.80	0
local precision (%)	63.15	47.89	22.45	34.78	29.23	36.36	16.22	12.50	57.14	27.03	11.59	20.00
local recall (%)	71.21	42.50	20.75	39.02	25.68	48.48	21.43	5.88	37.93	34.48	11.59	3.23

(b) living room

object type	painting	door	cabinet	dining table	window	heater	chair	sofa	coffee table	end table	tv stand
global precision (%)	43.75	30.25	15.00	39.29	16.00	0.00	22.39	44.09	37.84	0.00	6.25
global recall (%)	44.21	27.69	9.38	30.56	8.00	0.00	11.90	39.05	33.33	0.00	4.35
local precision (%)	59.49	45.36	22.73	38.71	30.77	20.00	21.05	59.49	39.39	20.00	22.22
local recall (%)	49.47	33.85	15.63	33.33	16.00	16.67	9.52	44.76	30.95	5.88	8.70

The semantic labeling provides a different evaluation as in PASCAL VOC segmentation challenge. We assign each uniformly sampled vector on a sphere with a semantic label considering the occlusion, and test the labeling accuracy for each category. Table 3 shows the full result.

Table 3: Semantic labeling accuracy

(a) bedroom

object type	background	bed	desk	window	mirror	door	nightstand	wardrobe	cabinet	painting	tv	chair	sofa
global (%)	86.90	78.58	29.55	35.58	38.15	19.40	39.66	27.44	0.00	38.70	34.81	9.61	11.10
local (%)	87.13	80.76	33.10	22.78	42.90	25.47	55.67	25.31	5.46	41.58	32.88	17.20	7.74

(b) living room

object type	background	painting	door	cabinet	dining table	window	heater	chair	sofa	coffee table	end table	tv stand
global (%)	91.98	44.66	41.07	7.87	24.24	12.59	0.00	15.46	47.05	42.33	3.87	1.21
local (%)	93.50	47.50	36.75	16.27	21.80	12.37	11.19	14.95	49.47	42.78	3.99	7.66

To compare with room layout estimation algorithms for regular FOV images, Table 4 shows the pixel-wise accuracy of room layout estimation on a set of randomly sampled perspective views from panorama. The perspective views are either looking horizontally or slightly downward (15°). The FOV is fixed to 54.4°, and the size is 640×480 pixels. These perspective images look very similarly as those tested by OM and GC in their paper. Table 4 shows our performance before and after local refinement, as well as the performance of OM and GC. Our room layout estimation that uses the whole panorama image is significantly better than OM and GC that use only a regular FOV image.

Table 4: Room layout pixel-wise accuracy

bedroom	accuracy (%)
global	91.83
local	92.47
OM	75.10
GC	69.67

living room	accuracy (%)
global	90.64
local	90.09
OM	74.34
GC	69.62

4 More results

In the following 12 pages, we show some randomly choosen results. The first column is the input panorama and the output object detection results. The second column contains intermediate steps for generating cuboid hypotheses from bottom-up sampling as well as the combination of OM and GC. The third column is the output visualized in 3D.

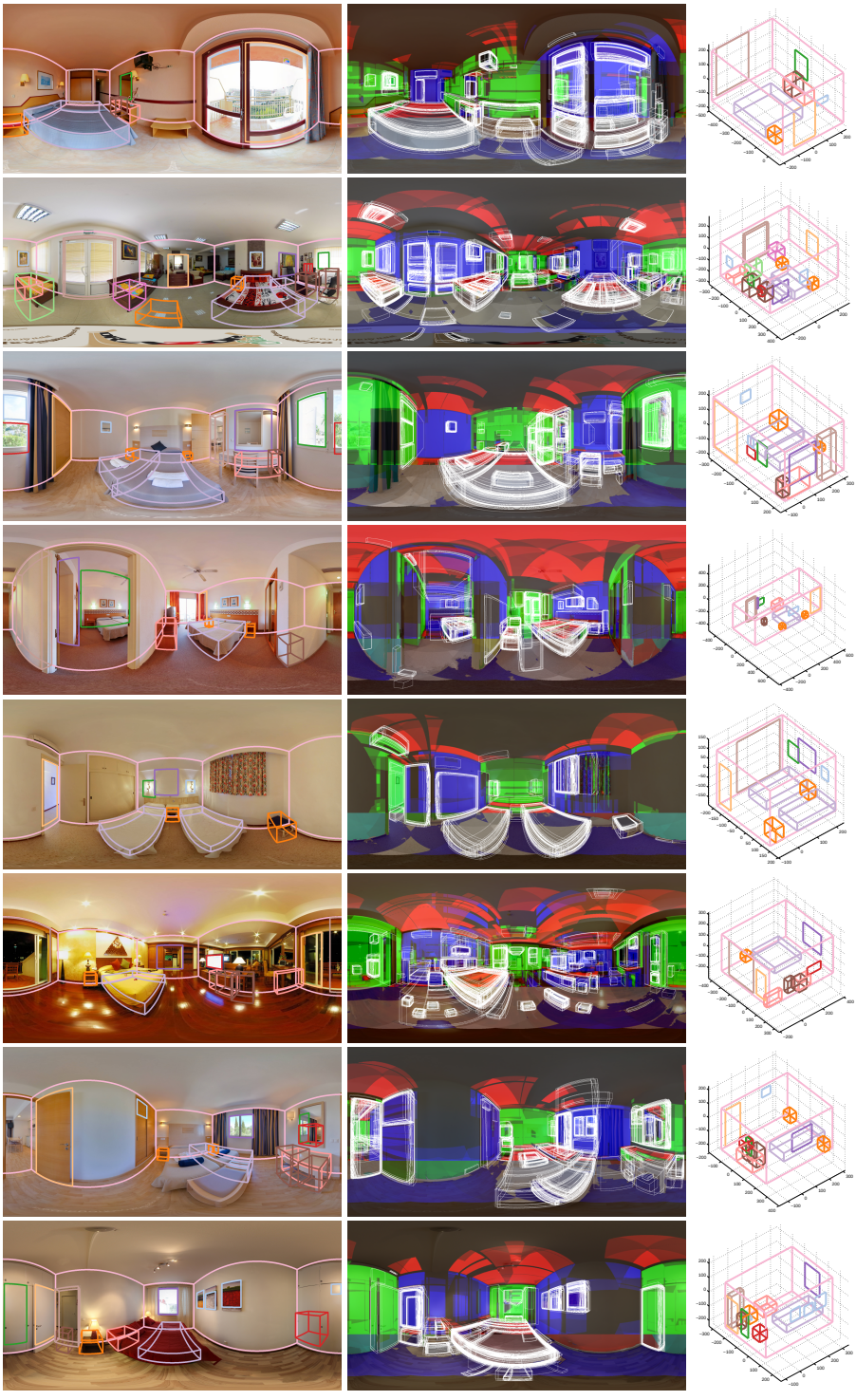
4.1 Bedroom















4.2 Living room

room bed window door nightstand desk sofa chair coffee table
painting mirror cabinet wardrobe dining table tv stand end table tv

