Large-scale Scene Understanding Challenge: Room Layout Estimation

Yinda Zhang, Fisher Yu, Shuran Song, Pingmei Xu, Ari Seff, Jianxiong Xiao

Princeton University

1 Task description

The objective of room layout challenge is to estimate the boxy representation of an indoor scene. Given an input indoor scene image (Fig 1(a)), the algorithm should predict the position of room corners, and intersection between ceiling, floor, and walls. The estimated room layout can be represented as a set of corners (Fig 1(b)) or pixel-wise segmentation (Fig 1(c)).

2 Data

2.1 Room layout representation

Room layout is represented by polygons, each corresponds to one region from the ceiling, floor, right/middle/left wall. However, this representation is ambiguous across different walls. In this challenge, we define 11 types of room layouts from the perspective of global structure. As shown in Fig. 2, these 11 room layouts cover most of the possible situations under typical camera poses. Each room layout is determined by a set of corner points. In the ground truth data, the position of the corner points are saved following the index on the point in Fig. 2.

2.2 Training, Validation, and Testing

The images are all from SUN database[2]. There are 4000 images for training, 394 for validation, and 1000 for testing. Please download the zip files for image and room layout ground truth, and unzip them into a same folder, e.g. Root. The training set and validation set, provided with ground truth, contains the following data field:

– *image*: The name of the image.
  The image can be found at "Root/images/*image*.jpg". The ground truth segmentation can be found at "Root/layout_seg/*image*.mat".
– *resolution*: The image resolution [height, width].
– *scene*: The scene type of the image. This is an additional information to encourage scene-related algorithms. Whether to use the scene type or not is a free option, and we will compare algorithms with and without using scene type separately.

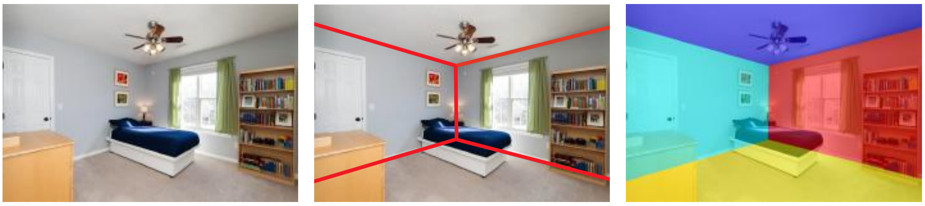
(a) Input image (b) Room layout: corner representation (c) Room layout: segmentation representation

Fig. 1: **Task.** (a) An input indoor scene image. (b) The output room layout represented by room corners. (c) The output room layout represented by a segmentation dividing image pixels into ceiling, wall, and floor regions.

– *type*: The ground truth room layout type. This is defined in Fig. 2.
– *point*: The image location of each corner of the room layout. The sequential order of the points are indexed in Fig. 2.

The testing data contain only *image*, *scene*, and *resolution* fields. People may choose whether to use *scene* for prediction freely but are required to report this in the submission.

3 Evaluation metrics

We adopt standard evaluation metrics: corner error and pixelwise error[1]. In corner error, we count the average distance between pairs of corresponding room corners, normalized by the diagnal of the image border. In pixelwise error, we count the percentage of pixels that are with the same label between prediction and ground truth. However, the above definition is confusing among different wall regions, especially when the type of prediction and ground truth are different. To handle this problem, for the corner error metric, we first calculate the distance between any pair of corners, one from prediction and one from the ground truth, and search for a bipartite matching with the minimal cost. We handle pixelwise error in a similar way by searching a maximal cost bipartite matching with consistency as cost. Any unmatched corner is penalized with 1/3, and unmatched region are automatically penalized in the bipartite matching.

4 Toolkit

– **demo.m**. General pipeline about how to use the toolkit.
– **GlobalParameters.m**. Define global parameters. You should set up "ROOT_DIR" to the root folder of the data.
– **predictFunc**. An example showing what to output for a prediction function.
– **evaluationFunc**. The evaluation function we will call on the server. It will take prediction and ground truth, and output both kind of errors.

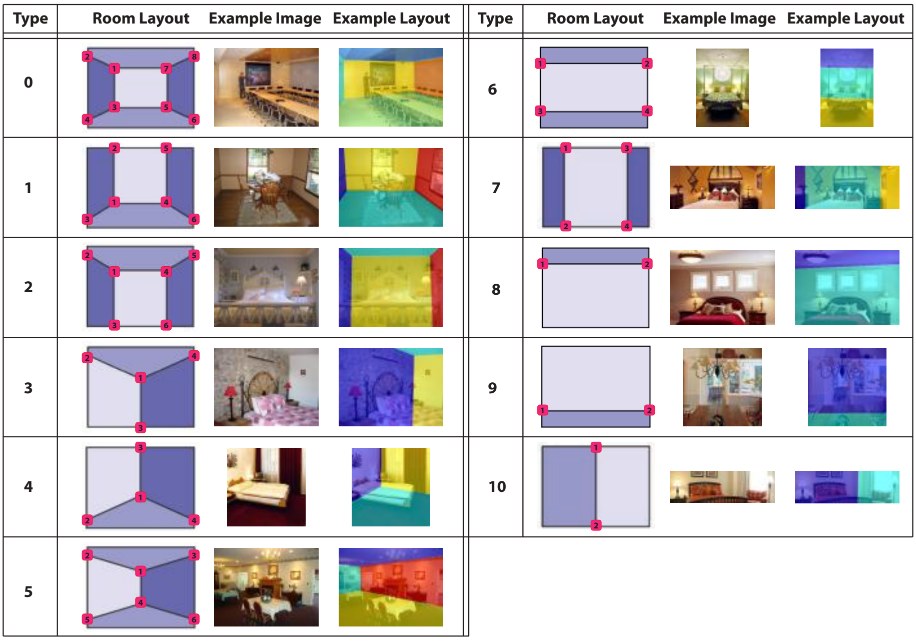

Fig. 2: **Definition of room layout types.** The type is indexed from 0 to 10. The number of on each corner defines the order of points saved in ground truth.

- **cornerError**. Evaluate corner error. It takes two sets of points, and search a bipartite matching with the minimal cost. The cost of the matching will be further normalized by the image resolution.
- **pixelwiseAccuracy**. Evaluate pixelwise error. It takes two segmentation masks, and search a bipartite matching with the maximal consistency.

5 What to submit

Participants are supposed to run their algorithm on testing set and organize the result in the format exactly the same as the output of the **predictFunc**. A single matlab structure should contain the following fields:

- **type** The room layout type as defined in Fig. 2
- **point** The location of the points saved in the order as shown in Fig. 2
- **layout** The room layout segmentation mask.

layout is optional. If not provided, we will fill in the field automatically by calling function **getSegmentation**. It generally perform well to transfer all ground truth point representation to segmentation, however has not been fully tested to handle corner cases.

More specifically, the result should be a single file in HDF5 format (the default format from "save" function in Matlab). This HDF5 file contains a single variable, named "result", holding the result in the format as mentioned above. We also requires you to fill up a **submission information form**, which can be downloaded from the LSUN website. Take the name for your submission as: "**[TeamName]**_**[version]**", where [version] is a number you choose for your submissions if there are multiple ones. Name your HDF5 file as "[submission_name].mat" and your submission form as "[submission_name].txt", and zip them to "[submission_name].zip". Please then follow the instruction on LSUN webpage to submit the zip file. We will send you a confirmation in email once we successfully evaluate your submission.

References

1. Hedau, V., Hoiem, D., Forsyth, D.: Recovering the spatial layout of cluttered rooms. In: Computer vision, 2009 IEEE 12th international conference on. pp. 1849–1856. IEEE (2009)
2. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: Computer vision and pattern recognition (CVPR), 2010 IEEE conference on. pp. 3485–3492. IEEE (2010)