Spatial Geometric Reasoning for Room Layout Estimation via Deep Reinforcement Learning

Liangliang Ren[1,2,3,*], Yangyang Song[1,2,3,*], Jiwen Lu[1,2,3,†], Jie Zhou[1,2,3,4]

[1]Department of Automation, Tsinghua University, China
[2]State Key Lab of Intelligent Technologies and Systems, Tsinghua University, China
[3]Beijing National Research Center for Information Science and Technology, China
[4]Tsinghua Shenzhen International Graduate School, Tsinghua University, China
{renll16, syy18}@mails.tsinghua.edu.cn; {lujiwen, jzhou}@tsinghua.edu.cn

Abstract. Unlike most existing works that define room layout on a 2D image, we model the layout in 3D as a configuration of the camera and the room. Our spatial geometric representation with only seven variables is more concise but effective, and more importantly enables direct 3D reasoning, e.g. how the camera is positioned relative to the room. This is particularly valuable in applications such as indoor robot navigation. We formulate the problem as a Markov decision process, in which the layout is incrementally adjusted based on the difference between the current layout and the target image, and the policy is learned via deep reinforcement learning. Our framework is end-to-end trainable, requiring no extra optimization, and achieves competitive performance on two challenging room layout datasets.

Keywords: Room layout estimation, reinforcement learning

1 Introduction

Room layout estimation is to locate the wall-wall, wall-ceiling and wall-floor boundaries on an image of an indoor room [11], which is usually regarded as a 3D cuboid. Applications such as indoor robot navigation and augmented reality can benefit from the knowledge of where each face of the room is located on the image. What makes the task interesting and challenging is the fact that there are often some parts of face boundaries occluded by foreground objects, which have to be inferred based on prior knowledge about the relationship between objects and the room. It sets room layout estimation apart from semantic segmentation [3, 18, 38], in which predictions are for visible objects.

Most conventional methods adopt the framework proposed by Hedau *et al.* [11], where a room layout is determined by three vanishing points and four rays, and the layout that maximizes a score function is the solution. However, this approach is heavily affected by the accuracy of vanishing point estimation [11].

* Indicates equal contribution.
† Corresponding author.

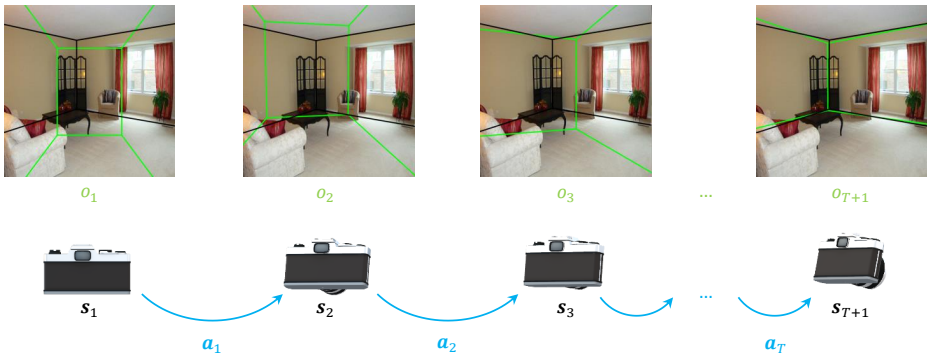

Fig. 1: We define room layout in 3D and formulate room layout estimation as a Markov decision process. The camera is moved inside the room step by step to recover the position and pose under which the input image was captured, based on how the captured layout (drawn in green) differs from the image (groundtruth layout drawn in black).

As for deep methods, at first convolutional neural networks are adopted to produce some intermediate results, *e.g.* pixelwise face segmentation [4] or edge probability map [36], and then an extra post-optimization is performed to obtain parameterized layout prediction. However, in the case of face segmentation (ceiling, floor, left/front/right wall), the three walls do not have any difference in appearance, but in their relative position in the image, making them difficult for the network to predict. And if less than three walls are captured, labels for the walls cannot be uniquely determined. For instance, in the two-wall case, one can label them as "left and front", or alternatively "front and right". In recent work, end-to-end prediction without optimization is achieved by estimating each keypoint of the layout separately [14]. However, no constraint exists about the keypoints, so the predicted layout is not guaranteed to be valid and impossible layouts might occur according to our observation (see Fig. 5).

To avoid the above problems introduced by defining layout on the 2D image, we model the layout before the imaging process, as a configuration of the camera and the room in 3D space, including the shape of the room, extrinsic and intrinsic parameters of the camera. Note that our layout representation is very concise with only seven variables, and naturally incorporates the cuboid room constraint. The position and pose of the camera with respect to the room is more useful in applications that require 3D reasoning such as indoor robot navigation, than simply locating faces on the 2D image, which is another advantage of our representation.

Unfortunately, there is no groundtruth annotation for our 3D layout representation in currently existing datasets [11, 35], so we cannot regress the layout from image input directly through supervised learning. Imagine we are standing inside the room with a camera in our hand, and trying to recover the camera pose under which the input image was captured. One intuitive solution is taking

a photo under the current pose, moving the camera based on the difference between the captured layout and the image, and repeating the process for certain steps, as illustrated in Fig. 1. For example, looking at o_1 in Fig. 1, we humans naturally know the camera should be turned towards right and also depressed a little bit.Motivated by this insight, we formulate layout estimation as a Markov decision process and adopt reinforcement learning to learn the policy of layout adjustment. The proposed framework "LIM" (Layout via Incremental Movements), is end-to-end trainable and needs no post-optimization during inference.

2 Related Work

Room Layout Estimation: For simplicity, most work consider the room as a 3D cuboid as in [11], instead of general Manhattan world models [7, 15, 39]. Hedau *et al.* [11] first estimate vanishing points corresponding to the three orthogonal directions of the room, and then a room layout can be determined by four rays traveling through two of the vanishing points. A number of layout hypotheses are sampled from layout space, and the one which maximizes a previously learned score function is selected as the solution. This framework is further extended by many subsequent works. At first, line membership [11] and geometric context [13] are utilized as cues for room layout in the score function, and later more feature designs are proposed, such as orientation map [15], junction feature [24] and informative edge feature [19]. To address the error caused by discretization in the optimization of the score function, Schwing and Urtasun [28] develop an efficient branch-and-bound algorithm to search for the exact optimal solution based on integral geometry [27]. Chao *et al.* [2] improve the estimation of vanishing points by reasoning the geometric relationship between humans and the scene in 3D. Also incorporating 3D reasoning, Gupta *et al.* [8] sample object cuboid hypotheses along with layout hypotheses, and the combinations which violate the physical rules in 3D space, such as "objects should be contained in the room" and "object should never intersect with each other", are filtered out from the search space. Wang *et al.* [32] propose to explicitly model the clutter labels as a latent variable, which allows joint optimization of the layout and the clutter labels. Apart from the above framework, some works model the problem from the perspective of probability [5–7].

The convolutional neural network is first used in the work by Mallya and Lazebnik [19], to predict pixelwise edge probability map directly from image input. Although the obtained edge map reveals the location of face boundaries, it is still used as a piece of image feature to score sampled layouts, which limits performance of the method. However, Zhao *et al.* [36] treat their obtained edge map as an intermediate result, and the parameterized layout is produced by their proposed physics-inspired optimization. Similarly, Dasgupta *et al.* [4] perform optimization on pixelwise face probability map predicted by an FCNN. Lee *et al.* achieve end-to-end prediction by training the network to predict heatmap of each keypoint separately, and thus the keypoints can be directly predicted as the max activation locations in the heatmaps. There are also some works utilizing

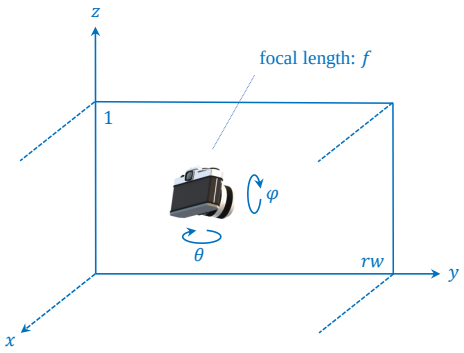

Fig. 2: We define a room layout in 3D space as a configuration of the room and the camera with seven variables.

extra information than a single RGB image, such as depth map [34] and floor plan [17], or focusing on recovering the 3D room model from a panorama [39].

Deep Reinforcement Learning: Different from supervised learning which requires labeled input-output pairs, reinforcement learning (RL) deals with learning from some kind of indirect supervision, reward given by the environment. RL aims to learn a policy of making decisions that maximize the cumulative reward, and thus RL is usually employed in the context of Markov decision process [30]. Deep reinforcement learning (DRL) [12,16,20,21,26,31] utilizes deep neural networks as an approximation of value function in value based algorithms such as DQN [21], or approximation of policy function in policy based algorithms such as DDPG [16]. In recent years, DRL has achieved great success in Atari games and some continuous control problems and also been successfully applied to computer vision tasks, *e.g.* object detection [1], visual tracking [33] and face recognition [25], which are all formulated as Markov decision processes. Caicedo *et al.* [1] localize an object by sequentially moving or resizing its bounding box, which covers a very large area of the image at the beginning. Rao *et al.* [25] propose to find attention in the videos to facilitate video face recognition, which is done by starting from videos of full length and removing valueless frames step by step. To the best of our knowledge, the proposed approach is the first to address room layout estimation using reinforcement learning.

3 Approach

In this section, we first represent room layout in 3D space, and then elaborate the design of *state*, *observation*, *action*, *reward* and *policy* in the context of reinforcement learning, and finally describe the procedures of training and inference. Our network architecture consists of a feature network, an actor network and also a critic network, whose detailed description is given in Fig. 3.

3.1 3D Layout Representation

A room layout is usually defined on the 2D image, as either pixelwise face segmentation [4], or a combination of layout type and a list of keypoints [14]. However, it is essentially the projection of a room onto an image plane. In other words, a room layout can naturally be defined in 3D space, as a configuration of the room and the camera, as illustrated in Fig. 2.

We regard the room as a cuboid, whose shape is parameterized by its width, height, and depth. However, the height can be set to unit length 1, since layout representation under a relative scale is all we need. And the depth is set to infinity because a camera that has an ordinary field of view and is located in normal positions cannot capture all six faces of a room at the same time. As a result, the shape of the room is completely determined by its width of rw.

Following common practice, the camera is treated as a pinhole model. Its extrinsic parameters include position and orientation relative to the world coordinate system, which is established along the edges of the room. Camera position is characterized by the coordinate of its center $[x, y, z]^T(\boldsymbol{C})$ with all three degrees of freedom (DOF). For the sake of clarity, we define axis X, Y in the camera coordinate system as the axis parallel to the horizontal and vertical direction of the image respectively, and axis Z is perpendicular to the image plane. Although camera orientation also has three DOF, we approximate the rotation about axis Z by zero for simplicity, since it is insignificant in most cases. Then any camera orientation can be reached by first rotating about axis Y for angle θ, then about axis X for angle φ, from certain reference orientation. Therefore the rotation matrix R of the camera is derived below. Here R_0 stands for rotation matrix corresponding to the reference orientation - the most typical camera poses with image plane parallel to the front wall and axis X located on the horizontal plane.

$$R = R_X(\varphi)R_Y(\theta)R_0 \tag{1}$$

Generally speaking, there are four intrinsic parameters for a pinhole camera: focal length f, principle point coordinate, and a skew parameter [9]. Here, the skew is set to zero, which is true for normal cameras, and the principle point is fixed to the image center by assuming the image has not been cropped. Thus the intrinsic parameter matrix K has the following form, where h and w represent height and width of the image respectively.

$$K = \begin{bmatrix} f & 0 & w/2 \\ 0 & f & h/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

In conclusion, the seven variables $\{rw, x, y, z, \theta, \varphi, f\}$ form a complete 3D representation of a room layout under some assumptions and approximations, from which any 2D representation can be computed through perspective projection with the camera matrix P:

$$P = K \begin{bmatrix} R & -R\boldsymbol{C} \end{bmatrix} \tag{3}$$

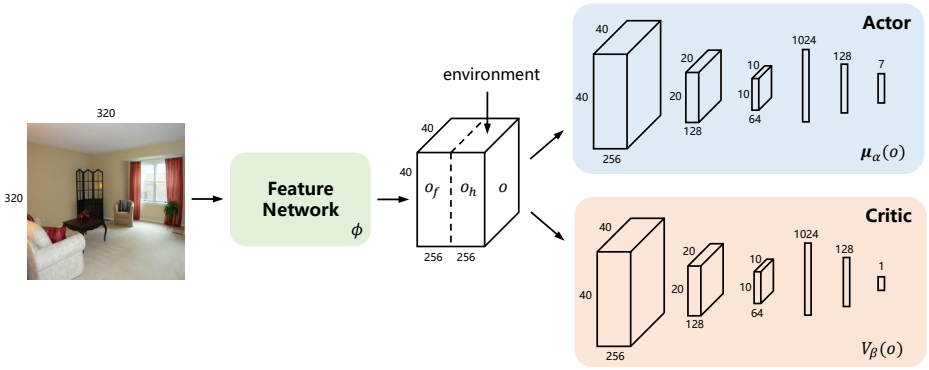

Fig. 3: An overview of our network architecture. Feature network adopts part of RoomNet basic structure [14], where feature observation o_f comes after the third last convolutional layer. Together with heatmap observation o_h given by the environment, a full observation o can be obtained by concatenating o_f and o_h along axis of channel. The actor and critic network share the same structure except the different number of output nodes, where actor outputs a 7-d vector representing the mean of the action, and critic outputs value for this observation. They both consist of three convolutional layers and two fully connected layers, where ReLU activation [22] is applied after every layer except the last ones. All convolutions are performed with kernel size 3×3 and stride 1, on properly zero-padded inputs, and maxpooling with stride 2 is done after the second and third convolutional layers.

3.2 Incremental Layout Estimation

As stated before, we formulate layout estimation as a Markov decision process, in which layout is adjusted step by step to arrive at the final prediction. In the framework of reinforcement learning, an agent is responsible for making such decisions, and an environment is something that the agent interacts with to learn the policy. In our formulation, the agent changes the layout (internal state $\boldsymbol{s}$ of the environment) by taking an action $\boldsymbol{a}$ based some observation o about how current layout differs from the target layout encoded in the image. And then the environment gives observation of the changed layout as well as a reward r, which evaluates how much the predicted layout is improved by that action. Since the action space is continuous, a policy based RL algorithm - Asynchronous Advantage Actor-Critic (A3C) [20] is employed, which involves an actor network modeling the policy $\pi_\alpha(o, \boldsymbol{a})$ and also a critic network approximating the state value function $V_\beta(o)$, where α, β denote parameters in the two networks respectively.

State: We define the internal state $\boldsymbol{s}$ of the environment as $[rw, x, y, z, \theta, \varphi, f]^T$, which is a m-dimension vector (here $m = 7$). Every variable in $\boldsymbol{s}$ is expected to be within a predefined range and states with variables out of range are "terminate states" in the decision process. For example, the camera center should never move

outside the room, and the elevation angle of the camera should never exceed 90 degrees. And terminate states also include the situation where the camera captures no edges but a blank face.

Observation: The observation o consists of two parts: o_h telling the agent what current layout is like, and o_f, image feature which gives cues about the target layout encoded in the input image. We adopt RoomNet-basic [14] which has been demonstrated successful in layout prediction as our image feature network, and o_f is the output of its third last convolutional layer with shape $40 \times 40 \times 256$. This feature network is not fixed but trainable in our approach with parameters denoted by ϕ. To be consistent with the RoomNet feature, we first project the room onto the image plane to obtain a list of keypoints (defined in [14]), and then construct o_h by summing Gaussian heatmaps centered at each keypoint with a standard deviation of two pixels on a 40×40 grid. o_h is repeated for 256 times along the channel axis and then concatenated with o_f to obtain the complete observation o as illustrated in Fig. 3. The purpose of the repetition is to prevent o_h being overwhelmed by o_f due to the huge imbalance on channel number.

Action: The action $\boldsymbol{a}$ is a vector of the same dimension as $\boldsymbol{s}$, and each variable v in $\boldsymbol{s}$ is changed with the corresponding increment Δv in $\boldsymbol{a}$. Variables like x, y, z, θ, φ which describe quantities about position are changed with additive increments, while rw and f which describe quantities about length are changed with multiplicative increments. And for the latter case, the logarithm of the increment instead of the increment itself is used to ensure the symmetry about zero. The equation below shows how the state is updated by the action:

$$v = \begin{cases} v + \Delta v, & \text{for } v \text{ in } \{x, y, z, \theta, \varphi\} \\ v \times e^{\Delta v}, & \text{for } v \text{ in } \{rw, f\} \end{cases} \tag{4}$$

Reward: Here, the choice of reward is quite straightforward: the improvement of layout error made by action $\boldsymbol{a}$. There are two commonly used error metrics, corner error e_c and pixelwise error e_p [35], leading to "corner reward" r_c and "pixel reward" r_p:

$$r_c = 100\Delta e_c \tag{5}$$

$$r_p = 100\Delta[e_p/2 + |n_1 - n_2|/3max(n_1, n_2)] \tag{6}$$

In the above equations, Δ represents the decrease of a certain quantity, and n_1, n_2 denote the number of keypoints in the current layout and groundtruth layout respectively. The second addend in r_p serves as a penalty to the unmatched number of keypoints. The final reward r is designed to be a weighted sum of r_c and r_p followed by a normalization as shown below, and the weight λ is dynamically changed during training as described in Section 4.2.

$$r = [(1 - \lambda)r_c + \lambda r_p]/8 + 1 \tag{7}$$

Policy: The policy $\pi_\alpha(o, \boldsymbol{a})$ is modeled as a multivariate normal distribution with independent components, whose mean $\boldsymbol{\mu}_\alpha(o)$ is given by the output of the actor network, whose variances $\sigma_1^2, \cdots, \sigma_m^2$ are manually set to control agent's

exploration while training, since in this problem random policy is not needed. Every element in action $\boldsymbol{a}$ is restricted within a predefined range to avoid huge change in single step, which ensures the stability of the layout adjustment process. To achieve this, the action sampled from the policy above is clipped by that action bound. And the action mean $\boldsymbol{\mu}_\alpha(o)$ is also fitted within the bound by applying *tanh* activation and a subsequent rescaling.

$$\pi_\alpha : \boldsymbol{a} \sim \mathcal{N}(\boldsymbol{\mu}_\alpha(o), diag(\sigma_1^2, \cdots, \sigma_m^2)) \tag{8}$$

To train our network, consider an episode in which the agent interacts with the environment for T steps following policy π_α and in the end collects a series of data $(o_1, \boldsymbol{a}_1, r_1), \cdots, (o_T, \boldsymbol{a}_T, r_T), o_{T+1}$.

The value targets are computed using the equation below, where γ denotes the discount factor. If o_{T+1} corresponds to a terminate state, then $V_\beta(o_{T+1})$ is substituted by zero.

$$R_i = \sum_{j=i}^{T} \gamma^{j-i} r_j + \gamma^{T-i+1} V_\beta(o_{T+1}) \tag{9}$$

Then value error can be obtained by subtracting the value target with value predicted by the critic network,

$$\delta_i = R_i - V_\beta(o_i) \tag{10}$$

The critic loss is the mean squared value error,

$$L_c = \frac{1}{T} \sum_{i=1}^{T} \delta_i^2 \tag{11}$$

The value error δ_i defined above is used as an estimate of the advantage function, and the policy gradient can be converted back to an optimization of the following actor loss,

$$L_a = -\frac{1}{mT} \sum_{i=1}^{T} ln\pi_\alpha(o_i, \boldsymbol{a}_i)\delta_i \tag{12}$$

The total loss is the sum of actor loss and critic loss, from which gradients propagate to parameter ϕ of the feature network through o_f. However, note that $V_\beta(o_{T+1})$ in (9) and δ_i in (12) are treated as constants, which gradients cannot propagate through.

$$\min_{\alpha,\beta,\phi} L = L_a + L_c \tag{13}$$

As mentioned before, we adopt A3C [20] to train our network, where multiple agents interact with multiple instances of the environment separately at the same time. It requires a main process which maintains a global network, and also multiple worker processes, each with a local network. Each worker collects

Table 1: Performance on LSUN validation-set.

Method	e_c (%)	e_p (%)
Hedau *et al.* (2009) [11]	15.48	24.23
Mallya and Lazebnik (2015) [19]	11.02	16.71
Delay (2016) [4]	8.20	10.63
RoomNet basic (2017) [14]	6.95	10.46
RoomNet recurrent 3-iter (2017) [14]	6.30	**9.86**
LIM	**6.23**	10.00

data and computes gradient of the loss parameters of the local network, which is instead applied to the global network, and then local parameters are synchronized with the global ones. As a result, agents in different worker processes can benefit from each other's experience. Multiple worker processes running in parallel also greatly increase the training speed.

As for inference, the layout prediction is also obtained by adjusting the layout step by step from some initial state. However, we remove the randomness in policy, $\boldsymbol{a} = \boldsymbol{\mu}_\alpha(o)$. Since agent might take bad actions which pull the layout away from the true layout, the last non-terminate state may not be the best one. Note that the value of a state is the expected return starting from it under the certain policy, and here the expected return means the expected error improvement, so a state of a lower value has lower error. Thus we select the state of the lowest estimated value (by critic network) in one episode as our prediction.

4 Experiments

4.1 Datasets

We conduct experiments on two standard datasets for room layout estimation LSUN [35] and Hedau dataset [11]. LSUN (Large-scale Scene Understanding Challenge dataset) consists of 4000 training, 394 validation and 1000 testing images, which are annotated with layout type and an ordered list of keypoints, but the annotations for testing images are not made public. A smaller one, Hedau dataset contains 209 training and 105 testing images, annotated with pixelwise face segmentation. In all experiments, only LSUN train-set is used for training, and evaluations are performed on LSUN validation-set as well as Hedau test-set. We use the two evaluation metrics mentioned before: corner error e_c which measures the Euclidean distance between predicted and groundtruth keypoints, and pixelwise error e_p which measures the percentage of pixels that are assigned to the wrong face [35]. We use the room layout challenge toolkit [35] provided by LSUN to compute these errors. The only data augmentation employed in our approach is horizontal image flipping while training.

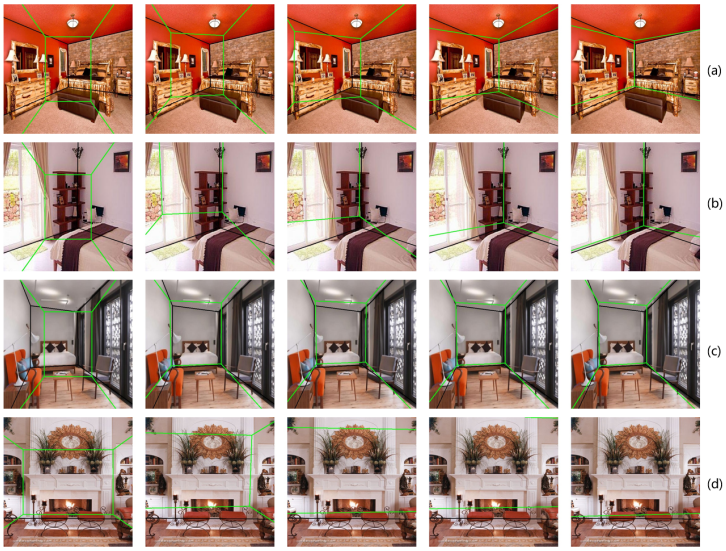

Fig. 4: Visualization of the layout adjustment process for images in LSUN validation-set. The first and the last column shows the initial state and the final prediction respectively.

4.2 Implementation Details

Our approach is implemented in PyTorch [23] - a popular deep learning platform, and all experiments are performed on NVIDIA TITAN Xp. The image preprocessing is performed by first resizing it to 320×320 resolution using bicubic interpolation and then rescaling the pixel intensities from $[0, 255]$ to $[0, 1]$. The feature network is pretrained under the framework of RoomNet [14] except Adam optimizer used in place of SGD. Both actor and critic network are initialized using the technique presented in [10]. In the training of our network, Adam optimizer is adopted as well, with parameters set to default except learning rate. Initial learning rates for ϕ, α and β are 10^{-5}, 5×10^{-6} and 5×10^{-5} respectively, and they will drop by a factor of 5 at global episode 500,000 and 700,000. Training stops at global episode 800,000 (equivalent to 100 epochs) and the maximum length of an episode is 10 steps. The discount factor γ is set to 0.9.

In our experiments, the main process along with 7 worker processes are created, and all networks are allocated on 2 GPUs in total. Parameters in global networks are shared across processes so that all workers can access and modify them. In our implementation, locks are applied to code blocks involving writing and reading global parameters to prevent concurrent operations from different processes, which will make training unstable according to our observation. Statistics in the Adam optimizer are also shared in GPU memory among processes.

Table 3: Comparison of different training and inference settings (evaluated on LSUN validation-set).

Setting	e_c (%)	e_p (%)
training		
fixed feature network	6.53	11.04
inference		
last state (10 steps)	6.73	10.65
last state (20 steps)	7.65	11.98
state of min value (10 steps)	6.42	10.29
state of min value (20 steps) - default	**6.23**	**10.00**

Table 2: Performance on Hedau test-set.

Method	e_p (%)
Hedau *et al.* (2009) [11]	21.20
Del Pero *et al.* (2012) [5]	16.30
Gupta *et al.* (2010) [8]	16.20
Zhao *et al.* (2013) [37]	14.50
Ramalingam *et al.* (2013) [24]	13.34
Mallya and Lazebnik (2015) [19]	12.83
Schwing *et al.* (2012) [27]	12.8
Del Pero *et al.* (2013) [6]	12.7
Delay (2016) [4]	9.73
LayoutNet (2018) [39]	9.69
RoomNet recurrent 3-iter (2017) [14]	**8.36**
LIM	9.12

Variance in our policy is manually set to control the exploration of the agent. At the beginning of training, exploration should be encouraged because the agent knows very little about how to get high rewards. However, as the training progresses, the agent gains more and more experience, and thus exploitation of current knowledge should be emphasized more. So the variance should be high enough at first but decrease as training proceeds. If the action bound of one element in the action is denoted by range $[-a_h, a_h]$ (always symmetric about zero), then its standard deviation is initially set to $a_h/2$ and decreased linearly to 0 at the end of training. As mentioned before, the reward r is computed by a weighted sum of corner reward and pixel reward. Before global episode 500,000, the weight λ is set to 0, which means in this period the reward is completely based on corner error, but after that, λ increases from 0 to 0.5. The purpose of this design is to encourage the agent to further focus on pixel error when it already performs well in corner error.

The initial state of the environment is designed to be "typical", so that all possible layouts are not too far to reach. The room width is equal to its height, and the focal length of the camera is properly set to produce an ordinary field of view. The camera is in the reference orientation described in Section 3.1 and is positioned so that all five faces of the room are captured, among which the front wall is located at the center of the image, as shown in the first column in Fig. 4. We have also tried initialization from layout predicted by RoomNet, which is already a good estimate, but observed no further improvements to the accuracy.

This in turn shows the robustness of our approach to the choice of initial state. One possible explanation is that our agent is not likely to get stuck into some local optima, and thus it tends to reach the same layout regardless of where it starts from.

4.3 Results and Analysis

Accuracy: The proposed approach LIM achieves competitive performance on both LSUN [35] and Hedau dataset [11], as reported in Table 1 and Table 2. On LSUN dataset LIM shows noticeable improvement on both corner error and pixel error compared to RoomNet basic. And it achieves performance comparable to a strong baseline - RoomNet recurrent 3-iter, which extends RoomNet basic with the recurrent structure for iterative refinement of keypoint heatmaps [14]. Although LIM and RoomNet recurrent 3-iter both complete the task through multiple steps, it is actually not fair to compare them because the multi-step procedures are done in different senses. Each step in LIM corresponds to a camera movement in the physical world, and only when all steps are performed is the final estimate obtained. However, in RoomNet recurrent, the output after the first step is already a valid estimate, the two additional steps only help to refine the estimate. Although a recent work [36] demonstrates amazingly high performance on the two datasets, another indoor scene dataset similar to LSUN, SUN RGB-D [29], is used to pretrain the network on semantic segmentation task, which provides very rich information about the scene configuration.

Efficiency: Although in our approach, the layout prediction is obtained by multiple steps of layout adjustment, LIM is still very efficient when it comes to running time. As reported in Table 4, in spite of running for at most 20 steps, LIM achieves 6.71 fps on average on LSUN validation-set, which is orders of magnitude faster than methods requiring post-optimization [4, 36].

Comparison of different training and inference settings: To better understand our approach, we change some training and inference settings to see their influence, which is summarized in Table 3. Compared to the fixed feature network while training (pretrained under the framework of RoomNet), trainable feature network leads to a performance boost of 0.3 % on corner error, and 1.04 % on pixel error. It suggests that our method can be further improved with a more effective feature. As for inference, it is not necessarily the best choice to terminate an episode after 10 steps as in the training stage, since given more steps, the agent can go further and may arrive at a better layout. Nevertheless, risks increase as well, because the agent sometimes takes bad actions. As shown in Table 3, if the last non-terminate state in an episode is selected as the final estimate, accuracy on both metrics drops significantly when an episode is extended from 10 steps to 20 steps. But if the state of the lowest estimated value (by critic network) is selected instead, the estimate will benefit from the extension of an episode. And for either episode length, the state of the lowest estimated value is more accurate than the last one on average.

Qualitative analysis: To visually demonstrate the effectiveness of our approach, we present a series of snapshots of the layout adjustment process in Fig. 4.

Fig. 5: RoomNet sometimes produces "impossible" layout prediction due to the lack of constraints on keypoints (first row), while our approach always gives valid estimate (second row).

Table 4: Comparison of deep methods on time complexity.

Method	**FPS**
Delay (2016) [4]	0.03
ST-PIO (2017) [36]	0.56
RoomNet recurrent 3-iter (2017) [14]	5.96
LIM (state of min value, 20 steps)	**6.71**

All layouts reach a satisfying final estimate from the same initial state, whether the starting layout is similar to the target layout, *e.g.* Fig. 4 (c), or far from it *e.g.* Fig. 4 (a)(b)(d). Since we build the feature network upon RoomNet basic, we further compare our approach with RoomNet. RoomNet defines the layout as a layout type along with a list of keypoints, which are predicted independently with no constraints imposed [14]. Therefore RoomNet might give invalid layout predictions, as shown in the first row of Fig. 5. However, our layout representation naturally incorporates the cuboid room constraint, and thus LIM is guaranteed to produce valid estimates even if it is not very good.

Fig. 6 provides some cases where LIM fails to give predictions consistent with the groundtruth. In (a) and (b), most area of the image is occupied with clutter and the wall-floor the boundary is almost completely occluded, making it very challenging to figure out the room layout. In addition, the wall-wall boundary in (b) is nearly invisible, causing the agent to believe there is only one wall. In fact, humans recognize two walls from the observation that the picture on the right wall has a noticeably different orientation than the left one. This indirect inference originates from our knowledge and experience about the world we live in, and thus is very difficult for a machine. (c) does not satisfy the cuboid room assumption upon which our approach is built, so LIM cannot find a well-fitting layout. (d) (e) (f) illustrate the problem about layout annotation. The groundtruth layout in (d)

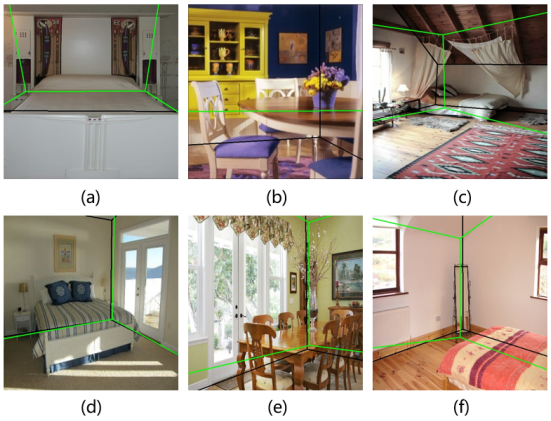

Fig. 6: Some cases where our approach produces predictions inconsistent with the groundtruth.

includes a very tiny ceiling that LIM fails to recognize due to its small size and uniform color distribution. However, another tiny ceiling in (e) is not annotated, which on the contrary is successfully estimated by our approach. This shows the inconsistency in the annotation in LSUN dataset. (f) is an ambiguous case where LIM treats the curved wall as a ceiling while the groundtruth does not.

5 Conclusion

In this paper, we have proposed a framework LIM for room layout estimation. We derive a concise but effective 3D layout representation, which enables direct 3D reasoning. Then we formulate the problem as a Markov decision process and employ reinforcement learning to learn the policy of layout adjustment. Our unified and efficient framework demonstrates very competitive performance on LSUN and Hedau datasets. And we believe our approach can be further improved with feature that is more robust to clutter and has a more holistic understanding about the scene.

Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700802, in part by the National Natural Science Foundation of China under Grant 61822603, Grant U1813218, Grant U1713214, and Grant 61672306, in part by Beijing Natural Science Foundation under Grant No. L172051, in part by Beijing Academy of Artificial Intelligence (BAAI), in part by a grant from the Institute for Guo Qiang, Tsinghua University, in part by the Shenzhen Fundamental Research Fund (Subject Arrangement) under Grant JCYJ20170412170602564, and in part by Tsinghua University Initiative Scientific Research Program.

References

1. Caicedo, J.C., Lazebnik, S.: Active object localization with deep reinforcement learning. In: ICCV (December 2015)
2. Chao, Y.W., Choi, W., Pantofaru, C., Savarese, S.: Layout estimation of highly cluttered indoor scenes using geometric and semantic cues. In: ICIAP. pp. 489–499 (2013)
3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062 (2014)
4. Dasgupta, S., Fang, K., Chen, K., Savarese, S.: Delay: Robust spatial layout estimation for cluttered indoor scenes. In: CVPR
5. Del Pero, L., Bowdish, J., Fried, D., Kermgard, B., Hartley, E., Barnard, K.: Bayesian geometric modeling of indoor scenes. In: CVPR. pp. 2719–2726 (2012)
6. Del Pero, L., Bowdish, J., Kermgard, B., Hartley, E., Barnard, K.: Understanding bayesian rooms using composite 3d object models. In: CVPR. pp. 153–160 (2013)
7. Flint, A., Murray, D., Reid, I.: Manhattan scene understanding using monocular, stereo, and 3d features. In: ICCV. pp. 2228–2235 (2011)
8. Gupta, A., Hebert, M., Kanade, T., Blei, D.M.: Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In: NIPS. pp. 1288–1296 (2010)
9. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge University Press (2003)
10. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: ICCV. pp. 1026–1034 (2015)
11. Hedau, V., Hoiem, D., Forsyth, D.: Recovering the spatial layout of cluttered rooms. In: ICCV. pp. 1849–1856 (2009)
12. Heess, N., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S., Riedmiller, M., et al.: Emergence of locomotion behaviours in rich environments. arXiv preprint arXiv:1707.02286 (2017)
13. Hoiem, D., Efros, A.A., Hebert, M.: Recovering surface layout from an image. IJCV **75**(1), 151–172 (2007)
14. Lee, C.Y., Badrinarayanan, V., Malisiewicz, T., Rabinovich, A.: Roomnet: End-to-end room layout estimation. In: ICCV. pp. 4865–4874 (2017)
15. Lee, D.C., Hebert, M., Kanade, T.: Geometric reasoning for single image structure recovery. In: CVPR. pp. 2136–2143 (2009)
16. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
17. Liu, C., Schwing, A.G., Kundu, K., Urtasun, R., Fidler, S.: Rent3d: Floor-plan priors for monocular layout estimation. In: CVPR. pp. 3413–3421 (2015)
18. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. pp. 3431–3440 (2015)
19. Mallya, A., Lazebnik, S.: Learning informative edge maps for indoor scene layout prediction. In: ICCV. pp. 936–944 (2015)
20. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: ICML. pp. 1928–1937 (2016)
21. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)

22. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML. pp. 807–814 (2010)
23. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS Autodiff Workshop (2017)
24. Ramalingam, S., Pillai, J.K., Jain, A., Taguchi, Y.: Manhattan junction catalogue for spatial reasoning of indoor scenes. In: CVPR. pp. 3065–3072 (2013)
25. Rao, Y., Lu, J., Zhou, J.: Attention-aware deep reinforcement learning for video face recognition. In: ICCV. pp. 3931–3940 (2017)
26. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
27. Schwing, A.G., Hazan, T., Pollefeys, M., Urtasun, R.: Efficient structured prediction for 3d indoor scene understanding. In: CVPR. pp. 2815–2822 (2012)
28. Schwing, A.G., Urtasun, R.: Efficient exact inference for 3d indoor scene understanding. In: ECCV. pp. 299–313 (2012)
29. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: CVPR. pp. 567–576 (2015)
30. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
31. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: AAAI (2016)
32. Wang, H., Gould, S., Roller, D.: Discriminative learning with latent variables for cluttered indoor scene understanding. Communications of the ACM **56**(4), 92–99 (2013)
33. Yun, S., Choi, J., Yoo, Y., Yun, K., Young Choi, J.: Action-decision networks for visual tracking with deep reinforcement learning. In: CVPR. pp. 2711–2720 (2017)
34. Zhang, J., Kan, C., Schwing, A.G., Urtasun, R.: Estimating the 3d layout of indoor scenes and its clutter from depth sensors. In: ICCV. pp. 1273–1280 (2013)
35. Zhang, Y., Yu, F., Song, S., Xu, P., Seff, A., Xiao, J.: Large-scale scene understanding challenge: Room layout estimation (2016)
36. Zhao, H., Lu, M., Yao, A., Guo, Y., Chen, Y., Zhang, L.: Physics inspired optimization on semantic transfer features: An alternative method for room layout estimation. In: CVPR. pp. 10–18 (2017)
37. Zhao, Y., Zhu, S.C.: Scene parsing by integrating function, geometry and appearance models. In: CVPR. pp. 3119–3126 (2013)
38. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: ICCV. pp. 1529–1537 (2015)
39. Zou, C., Colburn, A., Shan, Q., Hoiem, D.: Layoutnet: Reconstructing the 3d room layout from a single rgb image. In: CVPR. pp. 2051–2059 (2018)