Discriminative Learning with Latent Variables for Cluttered Indoor Scene Understanding

Huayan Wang[1], Stephen Gould[2], and Daphne Koller[1]

[1] Computer Science Department, Stanford University, CA, USA
[2] Electrical Engineering Department, Stanford Univeristy, CA, USA

Abstract. We address the problem of understanding an indoor scene from a single image in terms of recovering the layouts of the faces (floor, ceiling, walls) and furniture. A major challenge of this task arises from the fact that most indoor scenes are cluttered by furniture and decorations, whose appearances vary drastically across scenes, and can hardly be modeled (or even hand-labeled) consistently. In this paper we tackle this problem by introducing latent variables to account for clutters, so that the observed image is jointly explained by the face and clutter layouts. Model parameters are learned in the maximum margin formulation, which is constrained by extra prior energy terms that define the role of the latent variables. Our approach enables taking into account and inferring indoor clutter layouts *without* hand-labeling of the clutters in the training set. Yet it outperforms the state-of-the-art method of Hedau et al. [4] that requires clutter labels.

1 Introduction

In this paper, we focus on holistic understanding of indoor scenes in terms of recovering the layouts of the major faces (floor, ceiling, walls) and furniture (Fig. 1). The resulting representation could be useful as a strong geometric constraint in a variety of tasks such as object detection and motion planning. Our work is in spirit of recent work on holistic scene understanding, but focuses on indoor scenes.

For parameterizing the global geometry of an indoor scene, we adopt the approach of Hedau et al. [4], which models a room as a *box*. Specifically, given the inferred three vanishing points, we can generate a parametric family of boxes characterizing the layouts of the floor, ceiling and walls. The problem can be formulated as picking the box that best fits the image.

However, a major challenge arises from the fact that most indoor scenes are cluttered by a lot of furniture and decorations. They often obscure the geometric structure of the scene, and also occlude boundaries between walls and the floor. Appearances and layouts of clutters can vary drastically across different indoor scenes, so it is extremely difficult (if not impossible) to model them consistently. Moreover, hand-labeling of the furniture and decorations for training can be an extremely time-consuming (*e.g.*, delineating a chair by hand) and ambiguous task. For example, should windows and the rug be labeled as clutter?

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part IV, LNCS 6314, pp. 497–510, 2010.

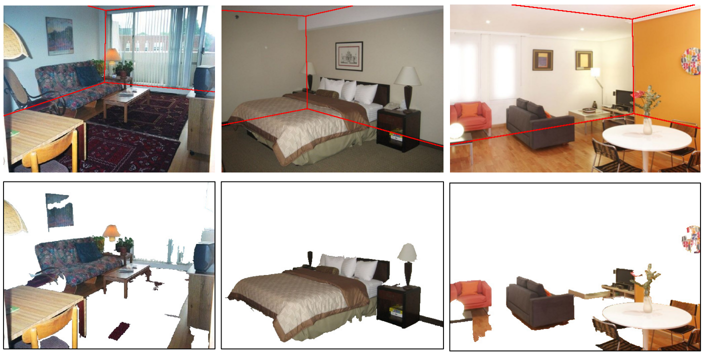

Fig. 1. Example results of recovering the "box" (1st row) and clutter layouts (2nd row) for indoor scenes. In the training images we only need to label the "box" but not clutters.

To tackle this problem, we introduce latent variables to represent the layouts of clutters. They are treated as *latent* in that the clutter is not hand-labeled in the training set. Instead, they participate in the model via a rich set of joint features, which tries to explain the observed image by the synergy of the box and the clutter layouts. As we introduce the latent variables we bear in mind that they should account for the *clutter* such as chairs, desks, sofa *etc.* However, the algorithm has no access to any supervision information on the latent variables. Given limited training data, it is hopeless to expect the learning process to figure out the concept of *clutter* by itself. We tackle this problem by introducing *prior* energy terms that capture our knowledge on *what the clutter should be*, and the learning algorithm tries to explain the image by the box and clutter layouts constrained by these prior beliefs. Our approach is attractive that it effectively incorporates complex and structured prior knowledge into a discriminative learning process with little human effort.

We evaluated our approach on the same dataset as used in [4]. Without hand-labeled clutters we achieve the average pixel error rate of 20.1%, in comparison to 26.5% in [4] without hand-labeled clutters, and 21.2% *with* hand-labeled clutters. This improvement can be attributed to three main contributions of our work (1) we introduce latent variables to account for the clutter layouts in a principled manner without hand-labeling them in the training set; (2) we design a rich set of joint features to capture the compatibility between image and the box-clutter layouts; (3) we perform more efficient and accurate inference by making use of the parameterization of the "box" space. The contribution of all of these aspects are validated in our experiments.

1.1 Related Work

Our method is closely related to a recent work of Hedau *et al* [4]. We adopted their idea of modeling the indoor scene geometry by generating "boxes" from the vanishing points, and using struct-SVM to pick the best box. However, they used supervised classification of surface labels [6] to identify clutters (furniture), and used the trained surface label classifier to iteratively refine the box layout estimation. Specifically, they use the estimated box layout to add features to supervised surface label classification, and use the classification result to lower the weights of "clutter" image regions in estimating the box layout. Thus their method requires the user to carefully delineate the clutters in the training set. In contrast, our latent variable formulation does not require any label of clutters, yet still accounts for them in a principled manner during learning and inference. We also design a richer set of joint feature as well as a more efficient inference method, both of which help boost our performance.

Incorporating image context to aid certain vision tasks and to achieve holistic scene understanding have been receiving increasing concern and efforts recently [3,5,6]. Our paper is another work in this direction that focuses on indoor scenes, which demonstrate some unique aspects of due to the geometric and appearance constraints of the room.

Latent variables has been exploited in the computer vision literature in various tasks such as object detection, recognition and segmentation. They can be used to represent visual concepts such as occlusion [11], object parts [2], and image-specific color models [9]. Introducing latent variables into struct-SVM was shown to be effective in several applications [12]. It is also an interesting aspect in our work that latent variables are used in direct correspondence with a concrete visual concept (clutters in the room), and we can visualize the inference result on latent variables via recovered furniture and decorations in the room.

2 Model

We begin by introducing notations to formalize our problem. We use $\boldsymbol{x}$ to denote the input variable, which is an image of an indoor scene; $\boldsymbol{y}$ to denote the output variable, which is the "box" characterizing the major faces (floor, walls, ceiling) of the room; and $\boldsymbol{h}$ to denote the latent variables, which specify the clutter layouts of the scene.

For representing the face layouts variable $\boldsymbol{y}$ we adopt the idea of [4]. Most indoor scenes are characterized by three dominant vanishing points. Given the position of these points, we can generate a parametric family of "boxes". Specifically, taking a similar approach as in [4] we first detect long lines in the image, then find three dominant groups of lines corresponding to three vanishing points. In this paper we omit the details of these preprocessing steps, which can be found in [4] and [8]. As shown in Fig. 2, we compute the average orientation of the lines corresponding to each vanishing point, and name the vanishing point corresponding to mostly horizontal lines as $\boldsymbol{vp}_0$; the one corresponding to mostly vertical lines as $\boldsymbol{vp}_1$; and the other one as $\boldsymbol{vp}_2$.

A candidate "box" specifying the face layouts of the scene can be generated by sending two rays from $\boldsymbol{vp}_0$, two rays from $\boldsymbol{vp}_1$, and connecting the four

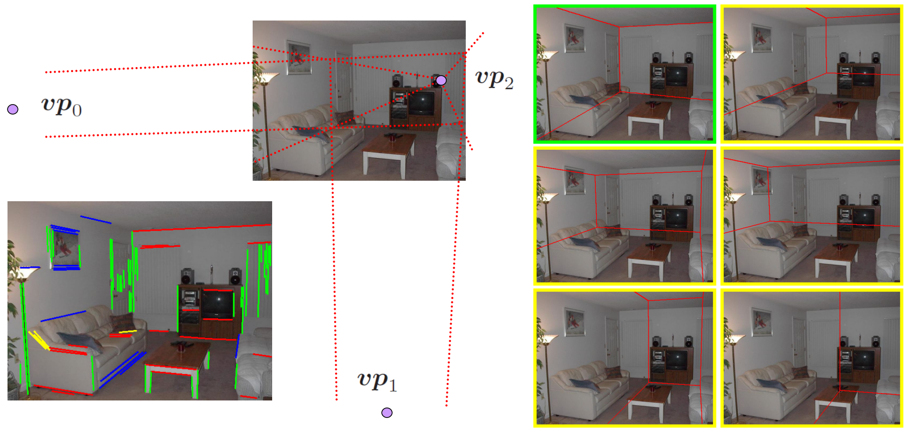

Fig. 2. Lower-Left: We have 3 groups of lines (shown in R, G, B) corresponding to the 3 vanishing points respectively. There are also "outlier" lines (shown in yellow) which do not belong to any group. **Upper-Left**: A candidate "box" specifying the boundaries between the ceiling, walls and floor is generated. **Right**: Candidate boxes (in yellow frames) generated in this way and the hand-labeled ground truth box layout (in green frame).

intersections with $\boldsymbol{vp}_2$. We use real parameters $\{y_i\}_{i=1}^4$ to specify the position[1] of the four rays sent from $\boldsymbol{vp}_0$ and $\boldsymbol{vp}_1$. Thus the position of the vanishing points and the value of $\{y_i\}_{i=1}^4$ completely determine a box hypothesis assigning each pixel a face label, which has five possible values $\{$*ceiling*, *left-wall*, *right-wall*, *front-wall*, *floor*$\}$. Note that some of the face labels could be absent; for example one might only observe *right-wall*, *front-wall* and *floor* in an image. In that case, some value of y_i would give rise to a ray that does not intersect with the extent of the image. Therefore we can represent the output variable $\boldsymbol{y}$ by only 4 dimensions $\{y_i\}_{i=1}^4$ thanks to the strong geometric constraint of the vanishing points[2]. One can also think of $\boldsymbol{y}$ as the face labels for all pixels. We also define a base distribution $p_0(\boldsymbol{y})$ over the output space estimated by fitting a multivariate Gaussian with diagonal covariance via maximum likelihood to the label boxes in the training set. The base distribution is used in our inference method.

To compactly represent the clutter layout variable $\boldsymbol{h}$, we first compute an over-segmentation of the image using mean-shift [1]. Each image is segmented into a number (typically less than a hundred) of regions, and for each region we assign it to either *clutter* or *non-clutter*. Thus the latent variable $\boldsymbol{h}$ is a binary

[1] There could be different design choices for parameterizing the "position" of a ray sent from a vanishing point. We use the position of its intersection with the image central line (use vertical and horizontal central line for $\boldsymbol{vp}_0$ and $\boldsymbol{vp}_1$ respectively).

[2] Note that $\boldsymbol{y}$ resides in a confined domain. For example, given the prior knowledge that the camera cannot be above the ceiling or beneath the floor, the two rays sent by $\boldsymbol{vp}_0$ must be on different sides of $\boldsymbol{vp}_2$. Similar constraints also apply to $\boldsymbol{vp}_1$.

vector with the same dimensionality as the number of regions in the image that resulted from the over-segmentation.

We now define the energy function $\mathbf{E}_{\boldsymbol{w}}$ that relates the image, the box and the clutter layouts:

$$\mathbf{E}_{\boldsymbol{w}}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{h}) = \langle \boldsymbol{w}, \boldsymbol{\Psi}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{h})\rangle - \mathbf{E}^0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{h}) \; . \tag{1}$$

$\boldsymbol{\Psi}$ is a joint feature mapping that contains a rich set of features measuring the compatibility between the observed image and the box-clutter layouts, taking into account image cues from various aspects including color, texture, perspective consistency, and overall layout. $\boldsymbol{w}$ contains the weights for the features that needs to be learned. $\mathbf{E}^0$ is an energy term that captures our prior knowledge on the role of the latent variables. Specifically, it measures the appearance consistency of the major faces (floor and walls) when the clutters are taken out, and also takes into account the overall clutterness of each face. Intuitively, it defines the latent variables (clutter) to be *things that appears inconsistently in each of the major faces*. Details about $\boldsymbol{\Psi}$ and $\mathbf{E}^0$ are introduced in Section 3.3.

The problem of recovering the face and clutter layouts can be formulated as:

$$(\bar{\boldsymbol{y}}, \bar{\boldsymbol{h}}) = \arg\max_{(\boldsymbol{y},\boldsymbol{h})} \mathbf{E}_{\boldsymbol{w}}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{h}) \; . \tag{2}$$

3 Learning and Inference

3.1 Learning

Given the training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$ with hand-labeled box layouts, we learn the parameters $\boldsymbol{w}$ discriminatively by adapting the large margin formulation of struct-SVM [10,12],

$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m} \xi_i \; , \quad \text{s.t. } \forall i, \;\; \xi_i \geq 0 \quad \text{and} \tag{3}$$

$$\forall i, \; \boldsymbol{y} \neq \boldsymbol{y}_i, \quad \max_{\boldsymbol{h}_i} \mathbf{E}_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{h}_i) - \max_{\boldsymbol{h}} \mathbf{E}_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{h}) \geq 1 - \frac{\xi_i}{\Delta(\boldsymbol{y}, \boldsymbol{y}_i)} \; , \tag{4}$$

where $\Delta(\boldsymbol{y}, \boldsymbol{y}_i)$ is the loss function that measures the difference between the candidate output $\boldsymbol{y}$ and the ground truth $\boldsymbol{y}_i$. We use pixel error rate (the percentage of pixels that are labeled differently by the two box layouts) as the loss function.

As $\mathbf{E}^0$ encodes the prior knowledge, it is fixed to constrain the learning process of model parameters $\boldsymbol{w}$. Without the slack variables ξ_i the constraints (4) essentially state that, for each training image i, any candidate box layout $\widehat{\boldsymbol{y}}$ cannot better explain the image than the ground truth layout $\boldsymbol{y}_i$. Maximizing the compatibility function over the latent variables gives the clutter layouts that best explain the image and box layouts under the current model parameters. Since the model can never fully explain the intrinsic complexity of real-world images, we have to slacken the constraints by the slack variables, which are scaled by the loss function $\Delta(\widehat{\boldsymbol{y}}, \boldsymbol{y}_i)$ indicating that hypothesis deviates more from the ground truth violating the constraint would incur a larger penalty.

The learning problem is difficult because the number of constraints in (4) is infinite. Even if we discretize the parameter space of $\boldsymbol{y}$ in some way, the total number of constraints is still huge. And each constraint involves an embedded inference problem for the latent variables. Generally this is tackled by gradually adding most violated constraints to the optimization problem [7,10], which involves an essential step of *loss augmented inference* that tries to find the output variable $\hat{\boldsymbol{y}}$ for which the constraint is most violated given the current parameters $\boldsymbol{w}$. In our problem, it corresponds to following inference problem:

$$(\widehat{\boldsymbol{y}}, \widehat{\boldsymbol{h}}) = \arg\max_{\boldsymbol{y},\boldsymbol{h}} (1 + \mathbf{E}_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{h}) - \mathbf{E}_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{h}_i)) \cdot \Delta(\boldsymbol{y}, \boldsymbol{y}_i)\,, \tag{5}$$

where the latent variables $\boldsymbol{h}_i$ should take the value that best explains the ground truth box layout under current model parameters:

$$\boldsymbol{h}_i = \arg\max_{\boldsymbol{h}} \mathbf{E}_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{h})\,. \tag{6}$$

The overall learning algorithm (follows from [10]) is shown in Algorithm 1. In the rest of this section, we will elaborate on the inference problems of (5) and (6), as well as the details of $\boldsymbol{\Psi}$ and $\mathbf{E}^0$.

Algorithm 1. Overall Learning Procedure

```
Input: $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$, $C$, $\epsilon_{final}$
Output: $\boldsymbol{w}$
$Cons \leftarrow \emptyset$
$\epsilon \leftarrow \epsilon_0$
repeat
  for $i = 1$ to $m$ do
    find $(\widehat{\boldsymbol{y}}, \widehat{\boldsymbol{h}})$ by solving (5) using Algorithm 2
    if the constraint in (4) corresponding to $(\widehat{\boldsymbol{y}}, \widehat{\boldsymbol{h}})$ is violated more than $\epsilon$ then
      add the constraint to $Cons$
    end if
  end for
  update $\boldsymbol{w}$ by solving the QP given $Cons$
  for $i = 1$ to $m$ do
    update $\boldsymbol{h}_i$ by solving (6)
  end for
  if # new constraints in last iteration is less than threshold then
    $\epsilon \leftarrow \epsilon/2$
  end if
until $\epsilon < \epsilon_{final}$ and # new constraints in last iteration is less than threshold
```

3.2 Approximate Inference

Because the joint feature mapping $\boldsymbol{\Psi}$ and prior energy $\mathbf{E}^0$ are defined in a rather complex way in order to take into account various kinds of image cues, the inference problems (2), (5) and (6) cannot be solved analytically. In [4] there was no latent variable $\boldsymbol{h}$, and the space of $\boldsymbol{y}$ is still tractable for simple discretization, so the constraints for struct-SVM can be pre-computed for each training image before the main learning procedure. However in our problem we are confronting the combinatorial complexity of $\boldsymbol{y}$ and $\boldsymbol{h}$, which makes it impossible to pre-compute all constraints.

For inferring $\boldsymbol{h}$ given $\boldsymbol{y}$, we use iterated conditional modes (ICM) [13]. Namely, we iteratively visit all segments, and flip a segment (between *clutter* and *non-clutter*) if it increase the objective value, and we stop the process if no segment is flipped in last iteration. To avoid local optima we start from multiple random initializations. For inferring both $\boldsymbol{y}$ and $\boldsymbol{h}$, we use stochastic hill climbing for $\boldsymbol{y}$, and the algorithm is shown in Algorithm 2.

The test-time inference procedure (2) is handle similarly as the loss augmented inference (5) but with a different objective. We can use a looser convergence criterion for (5) to speed up the process as it has to be performed multiple times in learning. The overall inference process is shown in Algorithm 2.

Algorithm 2. Stochastic Hill-Climbing for Inference

```
Input: $\boldsymbol{w}$, $\boldsymbol{x}$
Output: $\bar{\boldsymbol{y}}$, $\bar{\boldsymbol{h}}$
for a number of random seeds do
    sample $\bar{\boldsymbol{y}}$ from $p_0(\boldsymbol{y})$
    $\bar{\boldsymbol{h}} \leftarrow \arg\max_{\boldsymbol{h}} \mathbf{E}_{\boldsymbol{w}}(\boldsymbol{x}, \bar{\boldsymbol{y}}, \boldsymbol{h})$ by ICM
    repeat
        repeat
            perturb a parameter of $\boldsymbol{y}$ as long as it increases the objective
        until convergence
        $\bar{\boldsymbol{h}} \leftarrow \arg\max_{\boldsymbol{h}} \mathbf{E}_{\boldsymbol{w}}(\boldsymbol{x}, \bar{\boldsymbol{y}}, \boldsymbol{h})$ by ICM
    until convergence
end for
```

In experiments we also compare to another inference method that does not make use of the continuous parameterization of $\boldsymbol{y}$. Specifically we independently generate a large number of candidate boxes from $p_0(\boldsymbol{y})$, infer the latent variable for each of them, and pick the one with the largest objective value. This is similar to the inference method used in [4], in which they independently evaluate all hypothesis boxes generated from a uniform discretization of the output space.

3.3 Priors and Features

For making use of color and texture information, we assign a 21 dimensional appearance vector to each pixel, including HSV values (3), RGB values (3), Gaussian filter in 3 scales on all 3 Lab color channels (9), Sobel filter in 2 directions and 2 scales (4), and Laplacian filter in 2 scales (2). Each dimension is normalized for each image to have zero mean and unit variance.

The prior energy-term $\mathbf{E}^0$ consists of 2 parts,

$$\mathbf{E}^0(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}) = \alpha^a \mathbf{E}^a(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}) + \alpha^c \mathbf{E}^c(\boldsymbol{y}, \boldsymbol{h}) . \tag{7}$$

The first term $\mathbf{E}^a$ summarizes the **a**ppearance variance of each major face excluding all clutter segments, which essentially encodes the prior belief that the major faces should have a relatively consistent appearance after the clutters are taken out. Specifically $\mathbf{E}^a$ is computed as the variance of the appearance value within a major face excluding clutter, summed over all the 21 dimensions of appearance values and 5 major faces. The second term $\mathbf{E}^c$ penalizes **c**lutterness of the scene to avoid taking out almost everything and leaving a tiny uniform piece that is very consistency in appearance. Specifically, for each face we compute $\exp(\beta s)$, where s is the area percentage of clutter in that face and β is a constant factor. This value is then averaged over the 5 faces weighted by their areas. The reason for adopting the exponential form is that it demonstrates superlinear penalty as the percentage of clutter increases. The relative weights between these 2 terms as well as the constant factor β were determined by cross-validation on the training set and then fixed in the learning process.

The features in $\boldsymbol{\Psi}$ come from various aspects of image cues as summarized below (228 features in total).

1. **Face Boundaries:** Ideally the boundaries between the 5 major faces should either be explained by a long line or occluded by some furniture. Therefore we introduce 2 features for each of the 8 boundaries[3], computed by the percentage of its length that is (1) in a clutter segment and (2) approximately overlapping with a line. So there are 16 features in this category.
2. **Perspective consistency:** The idea behind perspective consistency features is adopted from [4]. The lines in the image can be assigned into 3 groups corresponding to the 3 vanishing points (Fig. 2). For each major face, we are more likely to observe lines from 2 of the 3 groups. For example, on the front wall we are more likely to observe lines belonging to $\boldsymbol{vp}_0$ and $\boldsymbol{vp}_1$, but not $\boldsymbol{vp}_2$. In [4] they defined 5 features by computing the length percentage of lines from the "correct" groups for each face. In our work we enlarge the number of features to leave the learning algorithm with more flexibility. Specifically we count the total length of lines from all 3 groups in all 5 faces, and treating clutter and non-clutter segments separately, which results in $3 \times 5 \times 2 = 30$ features in this category.
3. **Cross-face difference:** For the 21 appearance values, we compute the difference between the 8 pairs of adjacent faces (excluding clutters), which results in 168 features.
4. **Overall layouts:** For each of 5 major faces, we use a binary feature indicating whether it is observable or not, and we also use a real feature for its area percentage in the image. Finally, we compute the likelihood of each of the 4 parameters $\{y_i\}_{i=1}^4$ under $p_0(\boldsymbol{y})$. So there are 14 features in this category.

[3] If all 5 faces are present, there are 8 boundaries between them.

Table 1. Quantitative results. **Row 1:** pixel error rate. **Row 2 & 3**: the number of test images (out of 105) with pixel error rate under 20% & 10%. **Column 1 ([6])**: Hoiem et al.'s region labeling algorithm. **Column 2 ([4] w/o)**: Hedau et al.'s method without clutter label. **Column 3 ([4] w/)** : Hedau et al.'s method with clutter label (iteratively refined by supervised surface label classification [6]). The first 3 columns are directly copied from [4]. **Column 4 (Ours w/o)**: Our method (without clutter label). **Column 5 (w/o prior)**: Our method without the prior knowledge constraint. **Column 6 ($\boldsymbol{h} = \mathbf{0}$)**: Our method with latent variables fixed to be zeros (assuming "no clutter"). **Column 7 ($\boldsymbol{h}$ = GT)**: Our method with latent variables fixed to be hand-labeled clutters in learning. **Column 8 (UB)**: Our method with latent variables fixed to be hand-labeled clutters in both learning and inference. In this case the testing phase is actually "cheating" by making use of the hand-labeled clutters, so the results can only be regarded as some upperbound. The deviations in the results are due to the randomization in both learning and inference. They are estimated over multiple runs of the entire procedure.

	[6]	[4] w/o	[4] w/	Ours w/o	w/o prior	$\boldsymbol{h} = \mathbf{0}$	$\boldsymbol{h}$ = GT	UB
Pixel	28.9%	26.5%	21.2%	**20.1±0.5%**	21.5±0.7%	22.2±0.4%	24.9±0.5%	19.2±0.6%
≤20%	–	–	–	**62±3**	58±4	57±3	46±3	67±3
≤10%	–	–	–	**30±3**	24±2	25±3	20±2	37±4

4 Experimental Results

For experiments we use the same datast[4] as used in [4]. The dataset consists of 314 images, and each image has hand-labeled box and clutter layouts. They also provided the training-test split (209 for training, 105 for test) on which they reported results in [4]. For comparison we use the same training-test split and achieve a pixel-error-rate of 20.1% *without* clutter labels, comparing to 26.5% in [4] without clutter labels and 21.2% with clutter labels. Detailed comparisons are shown in Table 1 (the last four columns are explained in the following subsections).

In order to validate the effects of prior knowledge in constraining the learning process, we take out the prior knowledge by adding the two terms $\mathbf{E}^a$ and $\mathbf{E}^c$ as ordinary features and try to learn their weights. The performance of recovering box layouts in this case is shown in Table 1, column 5. Although the difference between column 4 and 5 (Table 1) is small, there are many cases where recovering more reasonable clutters does help in recovering the correct box-layout. Some examples are shown in Figure 3, where the 1st and 2nd column (from left) are the box and clutter layouts recovered by the learned model with prior constraints, and the 3rd and 4th column are the result of learning without prior constraints. For example, in the case of the 3rd row (Fig. 3), the boundary between the *floor* and the *front-wall* (the wall on the right) is correctly recovered even though it is largely occluded by the bed, which is correctly inferred as "clutter", and the

[4] The dataset is available at
`https://netfiles.uiuc.edu/vhedau2/www/groundtruth.zip`

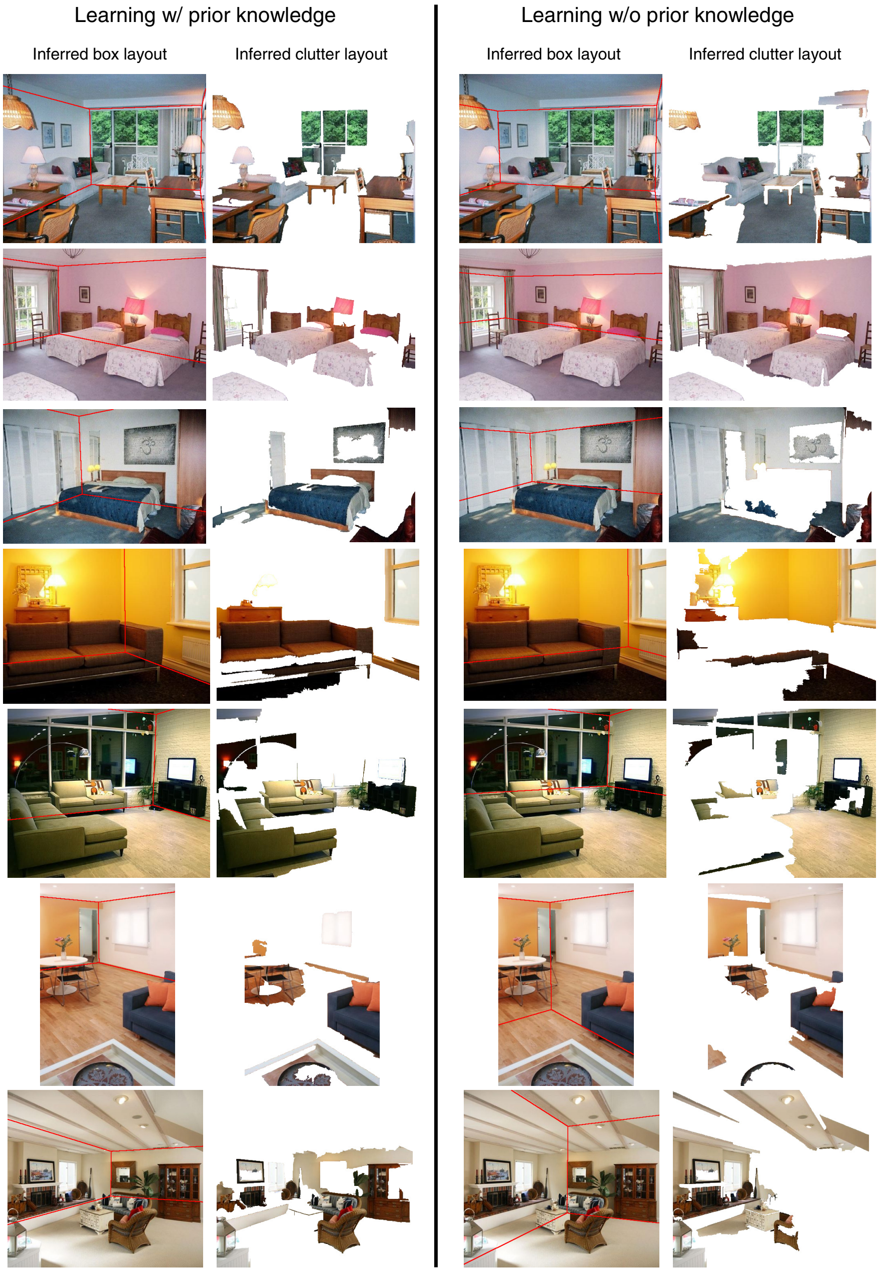

Fig. 3. Sample results for comparing learning with and without prior constraints. The 1st and 2nd column are the result of learning with prior constraints. The 3rd and 4th column are the result of learning without prior constraints. The clutter layouts are shown by removing all non-clutter segments. In many cases recovering more reasonable clutters does help in recovering the correct box layout.

boundary is probably found by the appearance difference between the floor and the wall. However, with the model learned without prior constraints, the bed is regarded as non-clutter whereas the major parts of the floor and walls are inferred as clutter (this is probably because the term $\mathbf{E}^c$ is not acting effectively with the learned weights), so it appears that the boundary between the *floor* and the *front-wall* is decided incorrectly by the difference between the white pillow and blue sheet.

We tried to fix the latent variables $\boldsymbol{h}$ to be all zeros. The results are shown in column 6 of Table 1. Note that in obtaining the result of 26.5% without clutter labels in [4], they only used "perspective consistency" features, although other kinds of features are incorporated as they resort to the clutter labels and the supervised surface label classification method in [6]. By fixing $\boldsymbol{h}$ to be all zeros (assuming no clutter) we actually decomposed our performance improvement upon [4] into two parts: (1) using the richer set of features, and (2) accounting for clutters with latent variables. Although the improvement brought by the richer set of features is larger, the effect of accounting for clutters is also significant.

We also tried fix the latent variables $\boldsymbol{h}$ to be the hand-labeled clutter layouts[5] The results are shown in column 7 of Table 1. We quantitatively compared our recovered clutter to the hand-labeled clutters, and the average pixel difference is around 30% on both the training and test set. However this value does not necessarily reflect the quality of our recovered clutters. In order to justify this, we show some comparisons between the hand-labeled clutters and the recovered clutters (from the test set) by our method in Fig. 4. Generally the hand labels include much less clutters than our algorithm recovers. Because delineating objects by hand is very time consuming, usually only one or two pieces of major furniture are labeled as clutter. Some salient clutters are missing in the hand-labels such as the cabinet and the TV in the image of the 1st row (Fig. 4), the smaller sofa in the image of the 5th row, and nothing is labeled in the image of the 3rd row. Therefore it is not surprising that learning with the hand-labeled clutter does not resulting in a better model (Table 1, column 7). Additionally, we also tried to fix the latent variable to be the hand-labeled clutters in *both* learning and inference. Note that the algorithm is actually "cheating" as it has access to the labeled clutters even in the testing phase. In this case it does give slightly better results (Table 1, column 8) than our method.

Although our method has improved the state-of-the-art performance on the dataset, there are still many cases where the performance is not satisfiable. For example in the 3rd image of Fig. 4, the ceiling is not recovered even though there are obvious image cues for it, and in the 4th-6th image of Fig. 4, the boundaries between the floor and the wall are not estimated accurately. There

[5] The hand-labeled clutters in the dataset are not completely compatible with our over-segmentation, *i.e.*, some segments may be partly labeled as clutter. In that case, we assign 1 to a binary latent variable if over 50% of the corresponding segment is labeled as clutter. The pixel difference brought by this "approximation" is 3.5% over the entire dataset, which should not significantly affect the learning results.

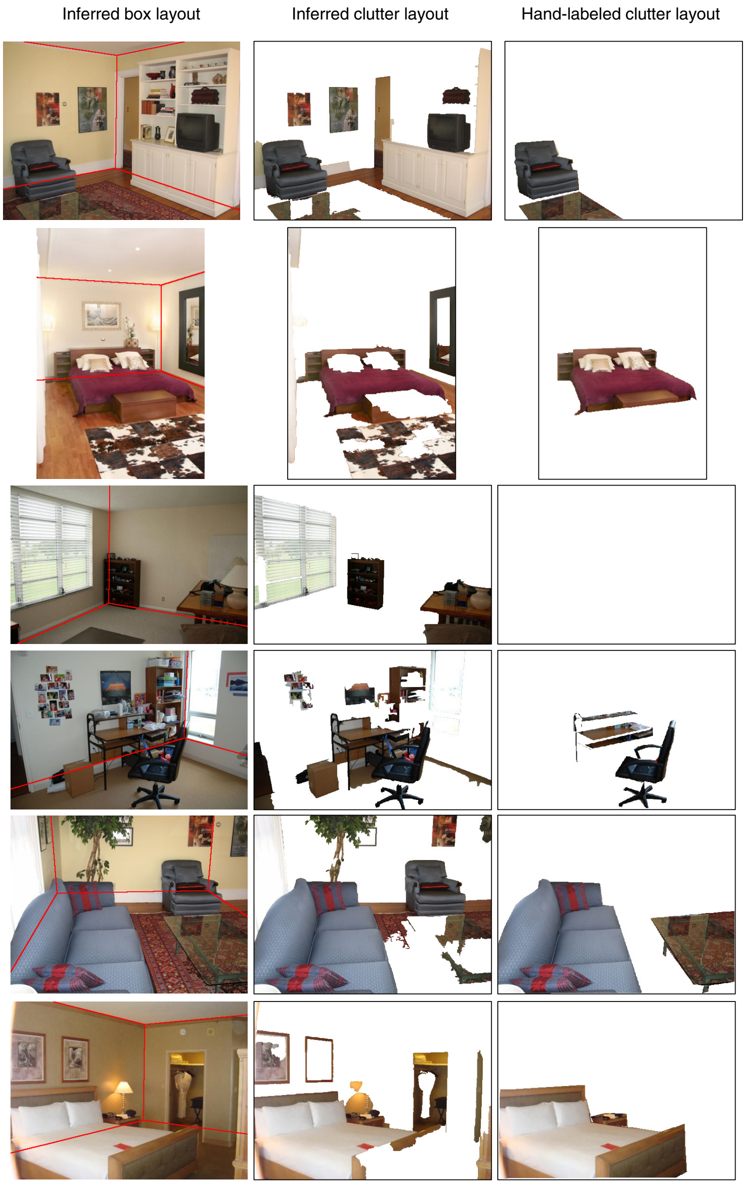

Fig. 4. Sample results for comparing the recovered clutters by our method and the hand-labeled clutters in the dataset. The 1st and 2nd column are recovered box and clutter layouts by our method. The 3rd column (right) is the hand-labeled clutter layouts. Our method usually recovers more objects as "clutter" than people would bother to delineate by hand. For example, the rug with a different appearance from the floor in the 2nd image, paintings on the wall in the 1st, 4th, 5th, 6th image, and the tree in the 5th image. There are also major pieces of furniture that are missing in the hand-labels but recovered by our method, such as the cabinet and TV in the 1st image, everything in the 3rd image, and the small sofa in the 5th image.

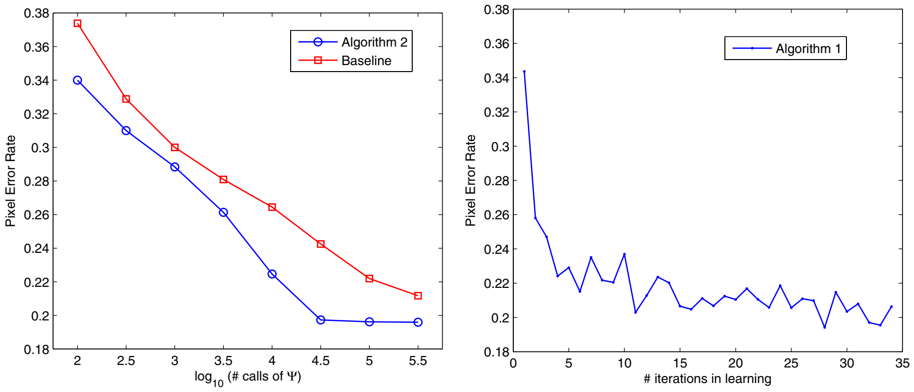

Fig. 5. Left: Comparison between the inference method described in Algorithm 2 and the baseline inference method that evaluates hypotheses independently. **Right:** Empirical convergence evaluation for the learning procedure.

is around 6-7% (out of the 20.1%) of the pixel error due to incorrect vanishing point detection results[6].

We compare our inference method (Algorithm 2) to the baseline method (evaluating hypotheses independently) described in Section 3.2. Fig. 5 (Left) shows the average pixel error rate over test set versus the number of calls to the joint feature mapping $\mathbf{\Psi}$ in log scale, which could be viewed as a measure of running time. The difference between the two curves is actually huge as we are plotting in log-scale. For example, for reaching the same error rate of 0.22 the baseline method would take roughly 10 times more calls to $\mathbf{\Psi}$.

As we have introduced many approximations into the learning procedure of latent struct-SVM, it is hard to theoretically guarantee the convergence of the learning algorithm. In Fig. 5 (Right) we show the performance of the learned model on test set versus the number of iterations in learning. Empirically the learning procedure approximately converges in a small number of iterations, although we do observe some fluctuation due to the randomized approximation used in the loss augmented inference step of learning.

5 Conclusion

In this paper we addressed the problem of recovering the geometric structure as well as clutter layouts from a single image. We used latent variables to account for indoor clutters, and introduced prior terms to define the role of latent variables and constrain the learning process. The box and clutter layouts recovered by our method can be used as a geometric constraint for subsequent tasks such

[6] The error rate of 6-7% is estimated by assuming a perfect model that always picks the best box generated from the vanishing point detection result, and performing stochastic hill-climbing to infer the box using the perfect model.

as object detection and motion planning. For example, the box layout suggests relative depth information, which constrains the scale of the objects we would expect to detect in the scene.

Our method (without clutter labels) outperforms the state-of-the-art method (with clutter labels) in recovering the box layout on the same dataset. And we are also able to recover the clutter layouts *without* hand-labeling of them in the training set.

Acknowledgements

This work was supported by the National Science Foundation under Grant No. RI-0917151, the Office of Naval Research under the MURI program (N000140710747) and the Boeing Corporation.

References

1. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Transactions on PAMI 24(5) (2002)
2. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. IEEE Transactions on PAMI (2010) (to appear)
3. Gould, S., Fulton, R., Koller, D.: Decomposing a scene into geometric and semantically consistent regions. In: ICCV (2009)
4. Hedau, V., Hoiem, D., Forsyth, D.: Recovering the spatial layout of cluttered room. In: ICCV (2009)
5. Heitz, G., Koller, D.: Learning spatial context: Using stuff to find things. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 30–43. Springer, Heidelberg (2008)
6. Hoiem, D., Efros, A., Hebert, M.: Recovering surface layout from an image. IJCV 75(1) (2007)
7. Joachims, T., Finley, T., Yu, C.-N.: Cutting-Plane Training of Structural SVMs. Machine Learning 77(1), 27–59 (2009)
8. Rother, C.: A new approach to vanishing point detection in architectural environments. In: IVC, vol. 20 (2002)
9. Shotton, J., Winn, J., Rother, C., Criminisi, A.: TextonBoost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout, and context. IJCV (2007)
10. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y., Singer, Y.: Large margin methods for structured and interdependent output variables. JMLR 6, 1453–1484 (2005)
11. Vedaldi, A., Zisserman, A.: Structured output regression for detection with partial occlusion. In: NIPS (2009)
12. Yu, C.-N., Joachims, T.: Learning structural SVMs with latent variable. In: ICML (2009)
13. Besag, J.: On the statistical analysis of dirty pictures (with discussions). Journal of the Royal Statistical Society, Series B 48, 259–302 (1986)