Extracting 3D Layout From a Single Image Using Global Image Structures

Zhongyu Lou, Theo Gevers, *Member, IEEE*, and Ninghang Hu

***Abstract*— Extracting the pixel-level 3D layout from a single image is important for different applications, such as object localization, image, and video categorization. Traditionally, the 3D layout is derived by solving a pixel-level classification problem. However, the image-level 3D structure can be very beneficial for extracting pixel-level 3D layout since it implies the way how pixels in the image are organized. In this paper, we propose an approach that first predicts the global image structure, and then we use the global structure for fine-grained pixel-level 3D layout extraction. In particular, image features are extracted based on multiple layout templates. We then learn a discriminative model for classifying the global layout at the image-level. Using latent variables, we implicitly model the sublevel semantics of the image, which enrich the expressiveness of our model. After the image-level structure is obtained, it is used as the prior knowledge to infer pixel-wise 3D layout. Experiments show that the results of our model outperform the state-of-the-art methods by 11.7% for 3D structure classification. Moreover, we show that employing the 3D structure prior information yields accurate 3D scene layout segmentation.**

***Index Terms*— Stage classification, 3D layout, structural SVM.**

I. Introduction

DERIVING the pixel-level 3D layout from 2D still images is important for many computer vision tasks such as object localization, image understanding and video segmentation [1]–[4].

The problem of extracting the 3D layout from 2D images has been widely studied [5]–[8]. A number of approaches infer 3D information from single images [9]–[11]. However, these algorithms require expensive pixel-wise labeling to learn their classification scheme [11]. Other methods such as [10] require accurate ground-truth of depth information.

Nedović et al. [1] takes a generic approach in which scenes are categorized into a limited set of image-level 3D geometry classes called 'stages'. Stages represent the general 3D geometry structure of scenes such as *sky-background-ground*, *sky-ground*, and *box*, see Fig. 1. We focus on stages which limit the number of scenes and hence make the problem of inferring pixel-level 3D layout more tractable.

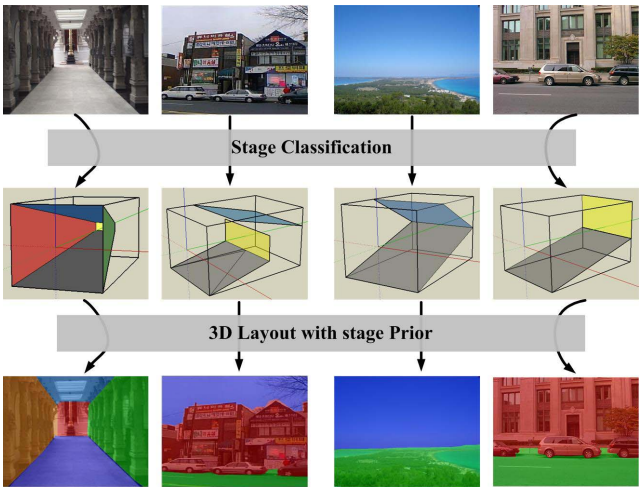


Fig. 1. The pipeline: to obtain the pixel-level 3D layout of a single image, we first estimate the image-level 3D structure by exploiting stage classification. Next, we generate the pixel-level 3D layout using the stage type of the image. Note that stage classification generates an image-level label for each image. The figures on the second row are displayed only to show the 3D geometry-structure of each stage.

In this paper, we propose a framework to infer the pixel-level 3D layout using global image structure. There are two steps in the proposed framework. In the first step, as shown in Fig. 1, the stage (i.e. an image-level label which represents the 3D structure of an image) is predicted. We formulate the classification of stages from 2D still images as a structural learning problem. Some previous methods (see [2]) use a holistic image representation to classify scenes. Due to the large intra-class variation in stage classification, a holistic representation may not have enough capacity to model general images. A probabilistic system is proposed to model the interdependency among the subtopics of the subregions (e.g. sky, building, ground, etc.). In this probabilistic system, latent variables are used to model the subtopics and a graphical model is introduced to learn the structure of scenes. In the second step, we exploit the stage prior to infer the pixel-level 3D layout, as shown in Fig. 1.

This work has three main contributions: a) Segmentation is exploited to obtain robust stage classification. b) An algorithm is proposed using stage classification to generate 3D layouts. c) No need for pixel-wise labeled training data and applicable to generic images. The source code and the new dataset will be made publicly available.

Manuscript received October 16, 2014; revised March 12, 2015; accepted April 29, 2015. Date of publication May 8, 2015; date of current version June 9, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Chun-Shien Lu.

Z. Lou and N. Hu are with the Intelligent Systems Laboratory Amsterdam, Informatics Institute, University of Amsterdam, Amsterdam 1098XH, The Netherlands (e-mail: z.lou@uva.nl; n.hu@uva.nl).

T. Gevers is with the Intelligent Systems Laboratory Amsterdam, Informatics Institute, University of Amsterdam, Amsterdam 1098XH, The Netherlands, and also with the Computer Vision Center, Universitat Autònoma de Barcelona, Barcelona 08193, Spain (e-mail: th.gevers@uva.nl).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The supplementary video shows the 3D layout generated by fitting the segmentation results to the rough geometry model. The total size of the video is 17.1 MB. Contact zyulou@gmail.com for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2015.2431443

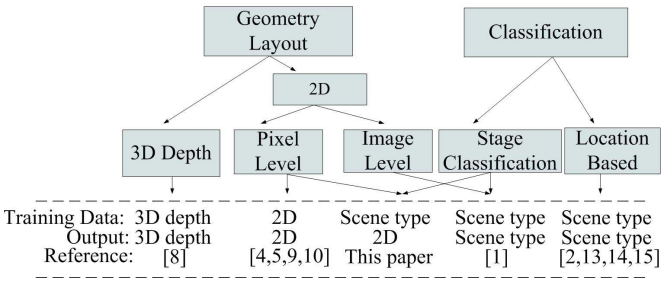


Fig. 2. Relation between this paper and the related work.

The paper is organized as follows. In Section II, related work is discussed. We describe the stage classification algorithm in Section III. Then, the 3D layout algorithm is presented in Section IV. Experiments and results of the stage classification and the 3D layout method are discussed in Sections V and VI respectively.

II. Related Work

The proposed method is closely related to two research topics: 3D layout construction from a single image and scene understanding by prototype categorization. The related work is considered below. In Fig. 2, the relation of this paper with other work is presented. This paper is closely related to two topics. 1) Work on 3D layout from a single image (Section II-A). 2) Work on scene understanding by prototype categorization (Section II-B).

A. 3D Layout From a Single Image

To learn the relationship between scene depth and image features, Saxena et al. [10] use a Markov Random Field to infer the 3D position and orientation for each image patch. Based on a few assumptions, such as connectivity and coplanarity, this method performs well for images taken from outdoor scenes. A drawback of the method is that it requires 3D depth ground-truth labels for training which limit the applicability of the method.

Other methods focus on parallel lines in indoor [5]–[7], [9], [12], [13] or outdoor images [14]. Lee et al. [12], Hedau et al. [5] and Wang et al. [6] derive parallel lines in indoor scenes to infer a set of predefined models to generate the 3D layout. Barinova et al. [14] focus on urban scenes. By detecting parallel lines, vanishing points are estimated and used to infer the 3D layout in a Conditional Random Field framework. However, these algorithms are highly dependent on parallel lines and can only handle indoor images [5]–[7], [9], [12], [13] or outdoor images [14] containing parallel lines.

Hoiem et al. [11] model a scene by three geometrical components. They train a set of classifiers to parse the scenes into three parts: sky, vertical and ground. This means the model can only handle outdoor images. Moreover, the method by Hoiem et al. [11] requires pixel-wise ground-truth training data which need tedious labeling work. In contrast, the proposed method only need image-level ground truth and it can handle general images.

(a) Traditional representation. (b) Proposed representation.

Fig. 3. Traditionally, for stage classification, features are extracted on a regular grid and pooled into a vector. Therefore, the patches (highlighted with bounding boxes) that correspond in (a) may result in a large difference. However, the proposed representation (b) measures the distance between segments (best viewed in color).

Fig. 4. For stage classification, the same class may contain several sub-topics. Directly learning a mapping from input features to output classes may not be able to model the relation between those sub-topics and the scene geometry type. In this paper, we model sub-topics using latent variables.

B. Scene Understanding by Prototype Categorization

Oliva and Torralba [2] use general image properties (e.g. naturalness and openness) to represent the spatial structure of a scene. Employing the Gist descriptor, they infer the relation between spatial properties and scene categories. Further, approaches based on the Bag of Words model [15] provide reasonable accuracy for scene categorization. Fei-Fei and Perona [16] propose the use of *theme* entities to represent image regions and learn the distribution of scene categories for these themes. These approaches classify images in different scene categories such as coast, beach, mountain, kitchen, and bedroom. However, the number of scene categories is very large. For example, Quattoni and Torralba [17] define 67 categories to describe indoor scenes. The SUN Dataset [18] has hundreds of scene classes.

Instead of considering scene categories, Nedović et al. [1] take a more abstract approach to classify 3D geometries. They show that scenes can be categorized into a limited number of 3D geometries called *stages*. A stage indicates the image-level 3D structure of a scene. To model scene geometries, 15 stages are defined such as *sky-background-ground*, *sky-ground*, *corner*, and *box*(some examples of stages are shown in Fig. 1). In this paper, we follow the notion of stages as the number of image-level 3D scene structures (i.e. stages) is limited and, hence, the problem is more tractable.

However, the stage classification algorithm [1] has two drawbacks. Firstly, features are extracted using a uniform grid which may result in large difference for the same class. For example, the two images in Fig. 3 belong to the same class which is *sky-background-ground* but they may yield a large difference. To this end, our aim is to generate a set of segments based on predefined templates and measure the difference between the segments. Further, instead of directly learning a mapping from the input features to output classes, we use latent variables to model possible subtopics, see Fig. 4.

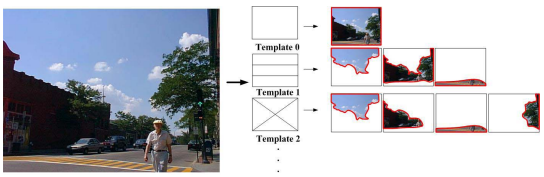


Fig. 5. Image representation based on templates. We define a set of templates by using segmentation to generate one subregion for each component of a template and subsequently extract appearance features (i.e. HOG descriptors) from each subregion.

III. Stage Classification

In this section, the first step (i.e. stage classification) of the framework is introduced. For stage classification, a graphical model is proposed to learn a mapping from image features to a stage. Each input image is parsed by a fixed number of predefined templates to obtain the input features. This allows the algorithm to learn the geometry of the image using structural learning.

A. Template Based Feature Extraction

We parse the scenes by a fixed set of predefined templates. These templates capture the basic geometry of stages, such as *sky-background-ground* (the Template 1 displayed in Fig. 5) and *box* (the second template in Fig. 5). Here, 'component' is used to denote a sub-region of a single template. For example, the Template 1 in Fig. 5 contains three components: the upper, middle and bottom part. Instead of extracting features from each component of these templates directly, we apply segmentation based on the image templates. Then the image features are extracted from the segment of each component of the templates.

1) Template Based Soft Segmentation: Soft segmentation is based on the figure-ground segmentation method proposed by Carreira and Sminchisescu [19]. Foreground seeds are uniformly placed on a grid and the background seeds are placed on the borders of the image. Foreground segments are generated from each seed at different scales by solving the max-flow energy optimization. Next, we keep the obtained segments which are highly intra-consistent (i.e. segments generated several times by different seeds). In this way, hundreds of segments are generated for each image. For each component of the templates, the segment which has the largest overlap-to-union score is selected.

2) Template Based Hard Segmentation: Since the soft segmentation is not always reliable, we use hard segmentation as compensation. Hard segmentation is predefined by the template. For example, for Template 1 in Fig. 5, hard segmentation parses the image uniformly into three sub-regions: top, middle and bottom.

3) Feature Representation: We use the two segmentation methods to generate two segments at each component of the templates. We extract features (e.g. HOG) from both of the segments, and these features are concatenated into a single feature vector. Such a feature vector is computed for each component of the templates. By collecting these feature vectors for all the templates, we obtain the inputs $\mathbf{x} = \{x_0, x_1, x_2, \ldots, x_M\}$ for each image. Here, each x_i is a feature vector of one component of the templates. M is the total number of template components, as shown in Fig. 6.

B. Model Formulation

Fig. 6 illustrates the graphical model of the proposed stage classification method. Let us denote each sample (i.e. image) by $(\mathbf{x}, \mathbf{z}, y) \in \mathcal{X} \times \mathcal{Z} \times \mathcal{Y}$. $\mathbf{x} = \{x_0, x_1, \ldots, x_M\}$ are the observations. Each x_i is a feature extracted from each component of the template, as described in Section III-A. y represents the stage label. $\mathbf{z} = \{z_1, z_2, \ldots, z_M\}$ are the latent variables. Each x_j has a corresponding z_j. Latent variables are implicitly defined and are learned from the training data. Fig. 4 intuitively shows the meaning of latent variables. For example, the latent variable of *background* may indicate *building*, *mountain* or *bus*. The latent variable of *ground* may indicate *grass ground*, *cultivated field* or *highway ground* and so on. $\mathbf{x}$ represent the input features, and it is always observed. y is the stage label, and it is known only during the training. In contrast, $\mathbf{z}$ is always hidden in both training and testing.

The stage label can be predicted by

$$\hat{y} = \text{argmax}_{y \in \mathcal{Y}} F(\mathbf{x}, \mathbf{z}, y; w), \tag{1}$$

where w is a set of parameters that are learned from training data, and $F(\mathbf{x}, \mathbf{z}, y; w)$ is the objective function describing the relation between input features and random variables in the graph.

C. Objective Function

The potential functions measure the compatibility between different variables. Three types of potentials are defined in the graphical model. The objective function is obtained by summing over all potentials in the graph structure:

$$F(\mathbf{x}, \mathbf{z}, y; w) = w^1 \cdot \Phi_1(x_0, y) + \sum_{i=1}^{M} w_i^2 \cdot \Phi_2(x_i, z_i) + \sum_{t=1}^{T} w_t^3 \cdot \Phi_3(y, Z_t), \tag{2}$$

where M is the total number of components of all templates, T is the total number of templates (except for Template 0). The latent variables $\mathbf{z}$ are implicitly defined and learned from the training data.

The first potential measures the compatibility between the global feature x_0 and stage label y:

$$\Psi_1(x_0, y) = w^1 \cdot \Phi_1(x_0, y). \tag{3}$$

We define $\Phi_1(x_0, y)$ to be the feature mapping function that transforms the input and the assignment into the feature space and w^1 to be the parameters. Suppose y has C states (i.e. $y \in \{1, 2 \ldots C\}$) and x_0 is a L dimension vector. Then the parameter w^1 is a $C \times L$ dimensional vector and the mapped feature is given by:

$$\Phi_1(x_0, y) = [\underbrace{0 \ldots 0}_{L \times (y-1)\text{dimension}} \quad x_0^T \ldots 0]. \tag{4}$$

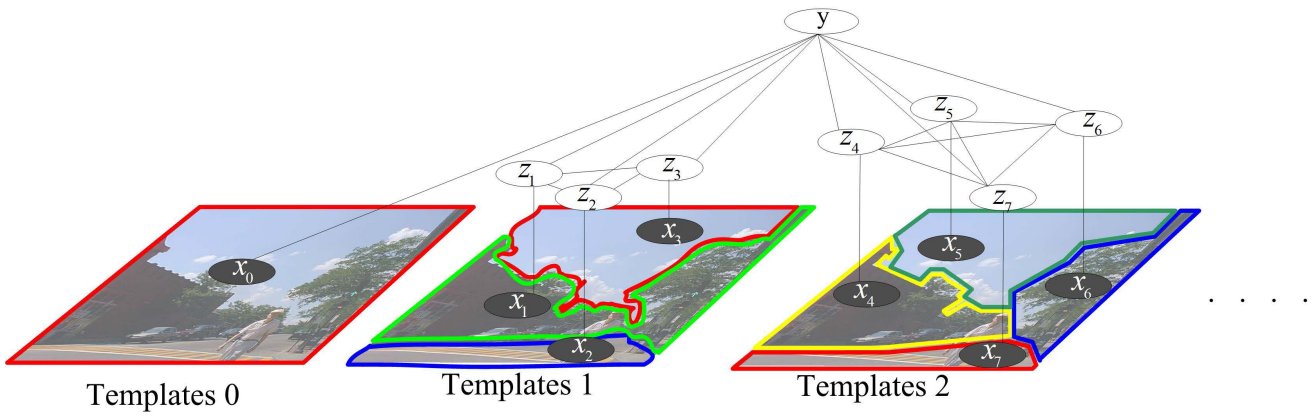


Fig. 6. The proposed graphical model for stage classification. The shaded nodes indicate the observed variables $\{x_0, x_1, x_2, \ldots, x_M\}$. The node y refers to the stage label of the image. The variables $\{z_1, z_2, \ldots, z_M\}$ are latent. As an example, only three templates are shown here, but note that our model uses up to five templates in total. All the observed nodes are connected with y but the links are not visualized in the graph for the purpose of clarity.

Note that if only this potential is used with the maximum margin approach, the model corresponds to a multi-class SVM. Hence, by using this potential, we can model the global mapping between the input features and the output prediction.

The second potential measures the score of having an observation x_i $(i > 0)$ and a latent variable z_i:

$$\Psi_2(x_i, z_i) = w_i^2 \cdot \Phi_2(x_i, z_i), \quad i \in \{1, 2, ...M\}. \tag{5}$$

In this potential, the observation x_i $(i > 0)$ is a subregion defined by the templates. Φ_2 is defined in the same way of Eq. (4). Each component of a template generates a subregion which is encoded into a feature vector x_i. The variable z_i is a latent variable modeling the subtopics. For example, $(z_i = 1)$ may represent the fact that a subregion is likely to be mountain and $(z_i = 2)$ could represent that a subregion is a building (see Fig. 4). In our model, parameters are not shared by all the potentials. Hence, for each pair of x_i and z_i, there is a corresponding parameter vector w_i^2 to be learned.

The third potential corresponds to the compatibility of a joint assignment of y and latent variable z. Suppose there are T templates, Z_t is used to represent the latent variables for subregions of the $t-th$ template e.g. $Z_1 = \{z_1, z_2, z_3\}$:

$$\Psi_3(y, Z_t) = w_t^3 \cdot \Phi_3(y, Z_t), \tag{6}$$

where $\Phi_3(y, Z_t)$ is an indicator function which has the format of $[0, 0, \ldots, 1, \ldots, 0]$, where the index of 1 is the index of the combinatorial state of y and Z_t in the whole value space. For instance, if there are 3 latent variables, the cardinality of each latent variable is 2 and the cardinality of y is 12, then the dimension of $\Phi_3(y, Z_t)$ is $2 \times 2 \times 2 \times 12 = 96$. This potential can be interpreted as the likelihood of the co-occurrence of y and Z_t. For example, the potential of jointly assigning the class label as *background-ground* and the subtopics as *buildings* and *ground* should be larger than that of assigning the class label as *sky-ground* and the subtopics as *buildings* and *ground*.

D. Inference

For stage classification, the aim is to predict stage label y. Given the parameters $w = [w^1 \ w^2 \ w^3]$ and input $\mathbf{x}$, performing inference on the model corresponds to maximizing the matching score of the joint assignment of class label y and latent variables $\mathbf{z}$ as follows:

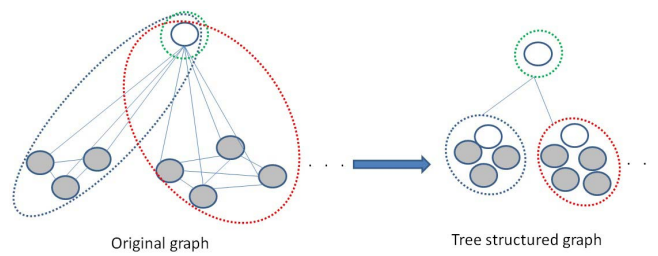


Fig. 7. Converting the original graph into a tree structured graph.

$$(\hat{\mathbf{z}}, \hat{y}) = \text{argmax}_{\mathbf{z} \in Z, y \in Y} F(\mathbf{x}, \mathbf{z}, y; w). \tag{7}$$

Here $F(y)$ in sec III-D is the objective function described in section III-C. Solving Eq. (7) is a very hard problem since it involves a combinatorial search over the target variable y and the latent variables $\mathbf{z}$. However, the graph can be converted into a tree-structure. As shown in Fig. 7, for each template t there is a set of latent variables Z_t. The variables in each template are collapsed with the output y which results in larger factors. By collecting all the factors, a tree structured graph is obtained. Then, by applying two rounds of message passing, exact inference is obtained [20].

E. Learning

Suppose N input training samples (images) are given $\{(\mathbf{x}_1, \mathbf{z}_1, y_1), (\mathbf{x}_2, \mathbf{z}_2, y_2), .., (\mathbf{x}_N, \mathbf{z}_N, y_N)\} \in \mathcal{X} \times \mathcal{Z} \times \mathcal{Y}$, the aim is to learn the parameters of the model. The learning consists of solving the following optimization problem:

$$\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{j=1}^{N} \Delta(y_j, \hat{y}_j) \right\}, \tag{8}$$

where C is the penalty parameter and $\Delta(y_i, \hat{y}_i)$ is the loss function. The loss function penalizes wrong predictions in

comparison to the ground-truth label. We define the loss function as follows:

$$\Delta(y_j, \hat{y}_j) = \begin{cases} 0 & \text{if } y_j = \hat{y}_j \\ 1 & \text{if } y_j \neq \hat{y}_j. \end{cases} \tag{9}$$

Since the loss function is non-differentiable, optimizing the problem of Eq. (8) is difficult. Therefore, we use an upper bound on the loss function by employing the methods of [21] and [22]. Subsequently, by introducing slack variables, the optimization function becomes:

$$\min_{\mathbf{w},\xi} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{j=1}^{N} \xi_j \right\}$$
$$s.t.\ \forall j \in \{1, 2, \ldots, N\}, \forall \mathbf{y}, \forall \mathbf{z} \in \mathcal{Z}:$$
$$\xi_j \geq \Delta(\mathbf{y}_j, \mathbf{y}) + F(\mathbf{y}, \mathbf{z}, \mathbf{x}_j; \mathbf{w}) - F(\mathbf{y}_j, \mathbf{z}_j^*, \mathbf{x}_j; \mathbf{w}). \tag{10}$$

The objective function F is defined by Eq. (2) and involves the inference of the latent variables $\mathbf{z}^*$ for each sample based on the current parameters. The optimization problem in Eq. (10) is not a convex energy function since $\mathbf{z}^*$ depends on the parameters w. Therefore, a local minimum can be found by exploiting the CCCP framework for latent structured-SVM as described in [22].

IV. 3D Layout From a Single Image

The next step is to obtain the pixel-wise 3D layout. To this end, we use the stage as prior information to segment the image into a 3D layout. Our segmentation is based on the random walks method [23]. Before explaining our algorithm, we give a short review of the random walker based segmentation algorithm. More detailed derivation can be found in [23].

A. Random Walks Based Segmentation

Random walks is based on graph theory. For random walker segmentation, a graph $\mathcal{G} = (\mathcal{E}, \mathcal{V})$ is defined for an image. Each pixel in the image is a node $v \in \mathcal{V}$ in the graph. Each pair of two neighbouring nodes v_i and v_j has an edge $e_{ij} \in \mathcal{E}$ connecting the two nodes (i.e. 4-neighbour connection). Note that since $\mathcal{G}$ is an undirect graph, e_{ij} is the same as e_{ji}. There is a weight, denoted by w_{ij}, for each edge e_{ij}. The sum of the weights of all the edges connected to node v_i, denoted by

$$d_i = \sum_j w_{ij}, \tag{11}$$

is defined to be the degree of a node. The weight w_{ij} represents the change of the intensity between two pixels in a random walker segmentation. Empirically, the weight w_{ij} is defined using a Gaussian weighting function by

$$w_{ij} = exp(-\beta(g_i - g_j)^2), \tag{12}$$

where g_i is the intensity of pixel i and β is a free parameter ($\beta = 90$ used in our experiments).

Previous work [23], [24] shows that the random walker problem has the same solution as the combinatorial Dirichlet problem. For solving the combinatorial Dirichlet problem, a combinatorial Laplacian matrix is defined as

$$L_{ij} = \begin{cases} d_i, & \text{if i = j,} \\ -w_{ij}, & \text{if } v_i \text{ and } v_j \text{ are adjacent nodes,} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

With the Laplacian matrix, a discrete version of the weighted Dirichlet integral can be formulated as

$$\mathcal{I}_x = \frac{1}{2}x^T L x, \tag{14}$$

where x is a vector containing all the nodes (pixels) of graph $\mathcal{G}$, and $\mathcal{I}$ is the pixel set.

For random walker segmentation, seed points are given at the start. The nodes of the whole graph are divided into two parts, the marked seeds V_M and the unseeded nodes V_U. $V_M \cup V_U = V$ and $V_M \cap V_U = \varnothing$, where V is the whole set of the nodes. Note that the marked seeds V_M may contain multiple labels. Eq. 14 can be decomposed by

$$\begin{aligned} \mathcal{I}_x &= \frac{1}{2}[x_M^T x_U^T] \begin{bmatrix} L_M & B \\ B^T & L_U \end{bmatrix} [x_M x_U] \\ &= \frac{1}{2}(x_M^T L_M x_M + 2x_U^T B^T x_M + x_U^T L_U x_U). \end{aligned} \tag{15}$$

Here, x_U is the potential of the unseeded nodes and x_M is the potential of the seeded nodes. The matrix L_M denote the relationship of the seeded nodes and L_U denote the relationship of the unseeded nodes. The matrix B and B^T describe the relationship between the seeded and unseeded nodes. By differentiating $\mathcal{I}_x$ with respect to x_U, the critical point can be found by solving the equation:

$$L_U x_U = -B^T x_M. \tag{16}$$

This is a system of linear equations with unknown x_U and it can be solved by using an iterative solver such as conjugate gradients [25]. Note that if there are K possible labels x^s, $s \in \{1, \ldots, K\}$ for x, then x_U and x_M have K columns. Each column denotes the probability of assigning x_U to be the corresponding label. Since the summation of the probabilities for each node should be equal to 1, $K - 1$ linear systems should be solved. For more details, please refer to [23].

B. Random Walk Segmentation With Global Structure Prior

Random walk segmentation method provides a natural way of using the stage prior which can be exploited in two ways: seeds and edges. Seeds are placed in the image regions corresponding to the geometric planes, i.e. the seeds of the sky in the class *sky-background-ground* are placed at the top of the image while the seeds of ground are placed at the bottom, as shown in Fig. 8. That means $x_M = 1$ for points that are selected as seeds.

We also enhance the edges between two planes and suppress the edges within each plane by generating a pre-defined weighting map $W(ij)$ for each stage, as shown in the Fig. 8(b). Therefore, the weights of edges defined in equation 12 becomes

$$w_{ij} = W(ij)exp(-\beta(g_i - g_j)^2). \tag{17}$$

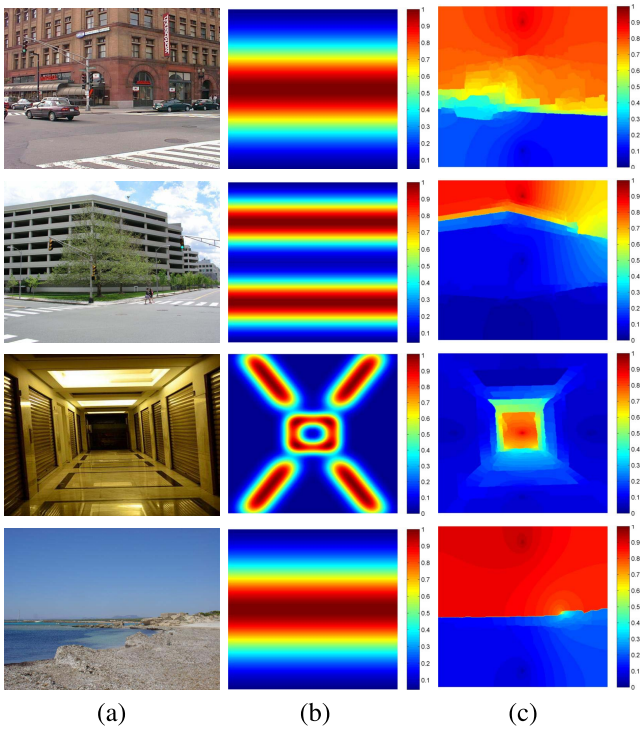


Fig. 8. (a) The original image. (b) The edge prior $W(ij)$ generated for each stage. (c) The probability of assigning each pixel to the corresponding label s (s represents background, sky, the front face and sky for the corresponding stage from first column to the fourth column). Note that the seeds for each label are also shown in (c) (the ones with probability equal to 1).

In this way, we use the global structure obtained by the stage classification as a prior. Segmentation is generated based on the new weight map and seeds.

Note that there are segments generated in Section III-A which are available for 3D layout. However, there are two reasons not to use the segmentation results directly. Firstly, for some stages there are no corresponding templates. Hence, some stages do not have an associated segmentation to be used directly. The second reason is that the segments generated may be characterized by unexpected overlaps or holes.

V. Stage Classification: Experiments

In this section, we evaluate the first step of the proposed framework: stage classification. Section V-B evaluates the performance of our stage classification algorithm for a varying number of templates. Section V-C evaluates the algorithm with varying cardinality of the latent variables. Finally, Section V-D compares the proposed stage classification method to other methods including the algorithm of Nedović [1], Fisher & pyramid [26], SVM with a set of popular features for scene classification.

A. Implementation Details

1) Dataset: We test our stage classification method on the dataset proposed by Nedović et al. [1]. This dataset contains 15 categories of geometric scenes. The dataset consists of more than two thousand images in total. We follow [1] in which some categories are combined, resulting in 12 stages in total: sky-background-ground (*skyBkgGnd*), background-ground (*bkgGnd*), sky-ground (*skyGnd*), table-person-background (*tabPersBkg*), person-background (*persBkg*), box, ground-diagonal background (*gndDiagBkgRL*), gnd, diagonal background (*diagBkgRL*), 1 side wall (*1sidewallRL*), corner and no depth (*noDepth*). For a fair comparison, we follow this configuration and test our method on these 12 stages.

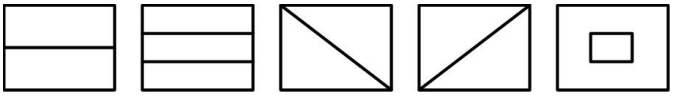


Fig. 9. The templates used in the experiments.

2) Features: In our model, we use HOG descriptors, color and parameters of the Weilbull distribution as features. HOG [27] is widely used in object detection for describing the shape of objects. For different stages, the shape inter-class variance is relatively high. For example, for the stages *box* (e.g. corridor) and *sky-background-ground*, the shape is very different. *box* normally has five consistent regions and four groups of parallel line projecting while *sky-background-ground* has a sky region in the upper side and a ground region in the lower side. Since HOG feature are indicative of this type of shape information, we use HOG in our algorithm. Nedović et al. [1] show that parameters of the Weibull distribution are informative to capture local depth ordering. Therefore, we exploit the parameters of the Weibull distribution as another feature. Finally, color cues are used to differentiate stages. For feature extraction, each region generated by soft and hard segmentation is divided into 4×4 grid patches. Then, the HOG descriptor [27] (9 dimensions), mean color values (RGB and HSV) (6 dimensions) and the parameters of Weilbull distributions [28] (4 dimensions) are computed for each patch. The final feature vector for each component of the templates is constructed by concatenating the features of each segment generated by both hard and soft segmentation.

3) Latent Variables Initialization: The latent variables are initialized in two ways. 1) Random initialization. 2) Data-driven initialization. Suppose there are K possible states per variable. Then, for random initialization, states are randomly chosen from 1 to K for each variable. For data-driven initialization, K-means clustering is used to group the inputs of a variable into K clusters. Subsequently, the label of each group is assigned as the initial latent state.

4) Templates: The five templates are shown in Fig. 9. To design these templates, we aim to generate stages using a the combination of those templates.

5) Metrics: In our experiments, four types of metrics are used: accuracy, precision, recall and F-score. The accuracy is defined as the total number of correctly predicted samples divided by the total number of test samples. The precision $Pr(i)$ of the ith category is defined as the number of correct predictions of the ith category divided by the total number of samples which are predicted to be in the ith category. The recall $Re(i)$ of the ith category is defined as the number of correct predictions for the ith category divided by the total number of the ith category in the test set. Finally, the F-score

TABLE I

THE CLASSIFICATION PERFORMANCE OF STAGES. THE ACCURACY, PRECISION, RECALL AND F1-SCORE ARE PROVIDED. HERE, $T = 0$ REPRESENTS NO TEMPLATE AND IN THAT CASE OUR MODEL CORRESPONDS TO MULTI-CLASS SVM. INIT-CLUSTER CORRESPONDS TO THE LATENT VARIABLES THAT ARE INITIALIZED BY CLUSTERING WHILE INIT-RANDOM MEANS INITIALIZING BY RANDOM LATENT VARIABLES

METHOD	ACCURACY %	PRECISION %	RECALL %	F1-SCORE %
T=0(MULTI-CLASS SVM)	40.9	38.4	39.0	38.2
T=1 (INIT-CLUSTER)	44.1	42.2	42.2	41.5
T=1 (INIT-RANDOM)	43.5	41.5	41.6	40.8
T=2 (INIT-CLUSTER)	46.0	43.6	43.9	42.9
T=2 (INIT-RANDOM)	44.9	43.1	43.1	42.1
FULL MODEL (T=5,INIT-CLUSTER)	**47.3**	**45.6**	44.9	**44.2**
FULL MODEL (T=5,INIT-RANDOM)	46.7	45.1	**45.2**	44.1

TABLE II

THE PERFORMANCE USING EACH SINGLE TEMPLATE SHOW IN FIG. 9. THE 1st TO 5th HERE IS THE FIVE TEMPLATES FROM LEFT TO RIGHT

METHOD	ACCURACY %	PRECISION %	RECALL %	F1-SCORE %
1st	44.1	**42.2**	42.2	41.5
2nd	43.1	42.0	42.6	41.6
3rd	42.6	40.8	42.0	41.0
4th	42.6	40.1	41.6	40.3
5th	**44.4**	41.5	**43.6**	**42.1**

TABLE III

THE INFLUENCE OF THE CARDINALITY OF LATENT VARIABLES TO THE TASK OF STAGE CLASSIFICATION

METHOD	ACCURACY %	PRECISION %	RECALL %	F1-SCORE %
CARDINALITY(Z)=1	46.5	44.9	44.2	43.2
CARDINALITY(Z)=10	46.9	45.3	**45.1**	44.1
CARDINALITY(Z)=20	47.1	45.2	45.0	44.1
CARDINALITY(Z)=30	**47.3**	**45.6**	44.9	**44.2**
CARDINALITY(Z)=40	46.7	44.8	44.7	43.8

$F(i)$ is defined as follows:

$$F(i) = \frac{2 * Pr(i) * Re(i)}{Pr(i) + Re(i)}. \tag{18}$$

Beside these metrics, the confusion matrix is also used to get more insight of the results (Fig. 11).

B. The Influence of the Number of Templates

T is the number of spatial templates. As shown in Fig. 6, when $T = 0$, the model only contains the Template 0 in which case the model corresponding to the multi-class SVM. When $T = 1$, the model contains Template 0 and Template 1 (all other nodes will be deleted except y). By changing T, the influence of the number of templates is evaluated. The parameters are obtained by cross-validation. We set $C = 4$ and $\epsilon = 0.1$ for all the experiments. We randomly choose half of the images to be the training set and the rest to be the testing set. By repeating the experiment five times, the average accuracy, precision, recall and F1-scores are obtained.

As shown in table I, the accuracy of the multi-class SVM is 40.9%. By adding one template, the accuracy increases for 3.2% from 40.9% up to 44.1%. With more templates added, the improvement ratio decrease slightly. This is due to the potential lack of interdependency between templates. By adding all templates, the final result reaches 47.3%. This experiment shows that the templates in our model help in the task of stage prediction.

To evaluate the performance of different templates, we conduct another experiment using a single template each time (i.e. $T = 1$). The results are shown in Table II. The 1st to 5th template is the five templates from left to right. The results show that higher accuracy is obtained when using the 1st and 5th template as shown in Fig. 9. The 1st template separates an image into two parts, upper and bottom parts. Since a large number of outdoor images in the dataset contain horizontal layers, it is logical that this template works well. The 5th template separates an image into middle and bottom parts. Since this dataset is mainly from TV scenarios, there is a large number of images with a person in the middle. The 5th template separates the person from the background and results in a better representation of the image.

C. The Influence of the Cardinality of Latent Variables

In this experiment, we evaluate the influence of the cardinality of latent variables. The full model with HOG, color, parameters of Weibull distribution is exploited. By changing the cardinality of latent variables, we report the performance of our model in table III. The accuracy of our algorithm improves with the cardinality increasing from 1 to 30. When the cardinality reaches 30, we obtain the best performance. By increasing the cardinality more, the performance decreases. This experiment shows that the proper cardinality of the latent

TABLE IV
COMPARISON TO THE STATE-OF-THE-ART METHODS FOR STAGE CLASSIFICATION

METHOD	ACCURACY %	PRECISION %	RECALL %	F1-SCORE %
NEDOVIC ET AL. [1]	35.6	39.2	37.9	36.4
FISHER+PYRAMID [26]	38.7	35.7	34.6	34.8
SVM (HOG+COLOR+WEIBULL)	40.3	37.1	38.6	37.2
SVM (HOG+COLOR+WEIBULL+GIST)	42.8	38.3	41.3	39.1
OUR FULL MODEL	**47.3**	**45.6**	**44.9**	**44.2**

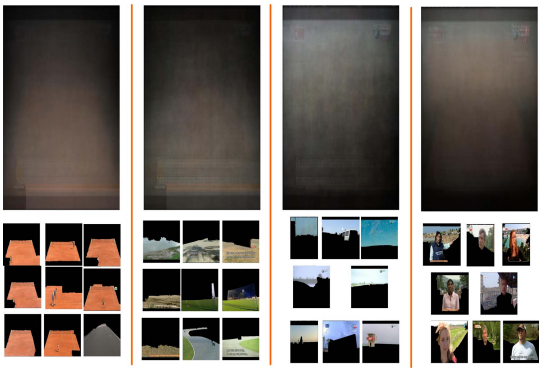


Fig. 10. Visualization of latent states. The first row shows the average image value of four latent states. Below the average image, segments corresponding to that state are shown.

variables is approximately 30. In all the other experiments, we set the cardinality to be 30.

The latent variables incorporate certain semantic interpretations in our model. It acts as a middle-level representation of the image. To understand the representation of the latent variables, we visualize a number of latent states by averaging all the segments for the corresponding latent state. Then, we show the segments belonging to that state. As shown in Fig. 10, from left to right, four states are visualized. The first two are latent states of the bottom part of the first template in Fig. 9. The other two states are the latent states of the upper part of the first template in Fig. 9. Those latent states capture different meanings. For example, the first column is mainly about sports-ground, high-way ground and the second column contains more complicated ground such as water-surface, grass and sand ground. The third column is mainly about blue sky while the fourth column is persons standing in outdoor scenes.

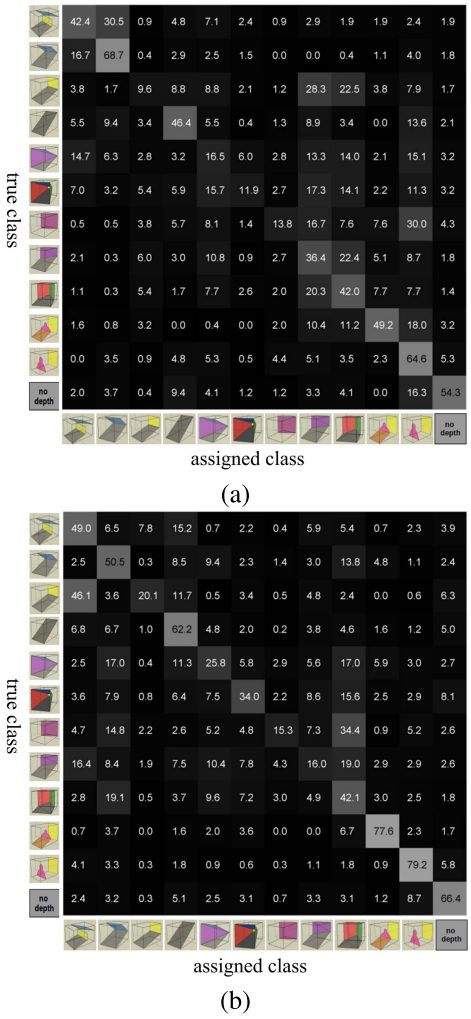


Fig. 11. The confusion matrix of (a) [1] and (b) ours.

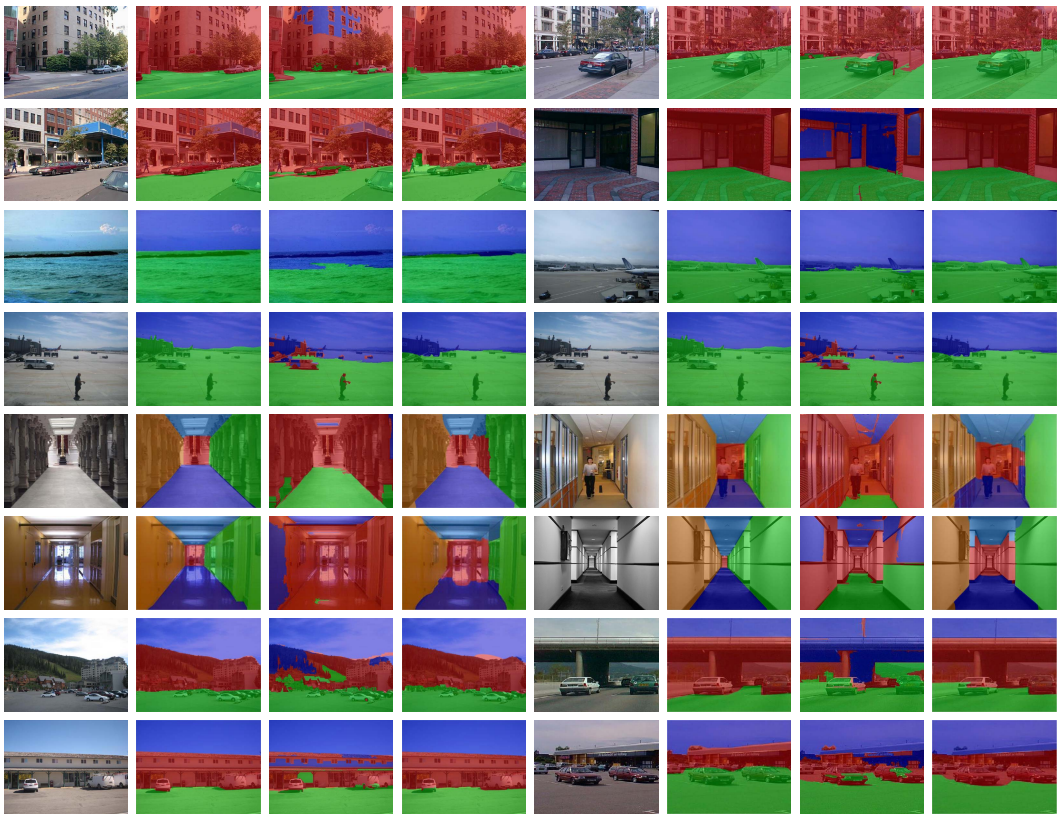
D. Comparison to the State-of-the-Art

In this experiment, we compare the performance of the proposed method with other state-of-the-art methods. In [1], a SVM classifier is used with different features (e.g. texture gradient features, atmospheric scattering features and anisotropic Gaussian features). The best results for the different settings are presented here. As shown in table IV, the average accuracy of [1] is 35.6% while ours is 47.3%. The proposed algorithm significantly outperforms the method of [1] by 11.7%. The confusion matrixes of [1] and ours are shown in Fig. 11.

Another baseline is 'fisher vector & spatial pyramid' for scene classification [26]. For the fisher vector, dense SIFT features are extracted every 2 pixels and the number of components for GMM is set to 256. For the spatial pyramid, three levels of pyramids are exploited - 1×1, 2×2 and 1×3. As shown in table IV, this algorithm has an average accuracy of 38.7%.

We also compare with the standard SVM (we use libSVM [29]). As shown in Table IV, using the same features (HOG, color and parameters of Weibull distribution), our algorithm outperforms SVM. Using SVM with more features, such as Gist [2], our algorithm still obtains highest accuracy.

In [1], a SVM classifier is used in combination with their features. However, their performance is lower than the SVM classifier using the proposed features (HOG, color and parameters of Weibull distribution).



(a) Input (b) GT (c) [11] (d) Ours (e) Input (f) GT (g) [11] (h) Ours

Fig. 12. Visual results of 3D geometry layout. (a, e) The input image. (b, f) The ground-truth layout of the input image. (c, g) The 3D layout of Hoiem et al. [11]. (d, h) The 3D layout of the proposed algorithm. Given that Hoiem et al. [11] label each pixel, it may generate erroneous labels inside the geometry plane. However, since our method consider the task globally, our results are more consistent within a geometry plane. Each color in this figure represent one geometric plane. For example, blue is sky, red is background and green is ground for outdoor images.

VI. 3D Layout From a Single Image: Experiments

In this section, we evaluate the second step of the proposed framework: 3D Layout from a Single Image. In section VI-A, the experimental setup is introduced. The results of different algorithms are shown in section VI-B.

A. Experimental Setup

Most of the images obtained by [1] are TV scenarios. Images from TV scenarios mostly contain crowds i.e. a number of persons in indoor scenes. In this case, it is ambiguous to assign each person to the appropriate geometric plane. So a new dataset is collected containing 456 images in total 212 images for training and 244 images for testing. These images are taken from other datasets, such as SUN [18]. The new dataset contains both indoor and outdoor images (111 indoor images and 345 outdoor images). The images in this dataset are mainly scenes with relatively clear geometric structure. Some examples are shown in Fig. 12. The proposed stage classifier is trained on the combined dataset of [1] and the training set of the new dataset. Then the image-level 3D layout algorithm is tested using the trained classifier. The pixel level ground-truth annotations from the test set are used for evaluation. Note that the ground-truth is only used for evaluating the 3D layout algorithm. Our approach only needs image-level label ground-truth for training.

We compare our algorithm with normalized cut (Ncut) [30], Vezhnevets et al. [31], Hoiem et al. [11] and Hedau et al. [5]. For Ncut, we set the segment number as a prior. Since Ncut does not generate a semantic label for each segment, we label each segment with the label of the ground-truth region which has the largest overlap with the segment. For Hoiem et al. [11], we consider the *vertical* in their definition as the *background* in our algorithm. For the *box* class, the *vertical* region of the Hoiem is assigned to one of the three classes: *middle*, *left* and *right*. We choose the one which have the maximum overlap with the *vertical*. Since Hedau et al. [5] can only generate layout for *box* class, we only report the result for this class.

B. Comparison to Other Algorithms

In Table V, the result of Ncut [30], Vezhnevets et al. [31], Hoiem et al. [11], Hedau et al. [5] and ours are reported. Vezhnevets et al. [31] aim to segment objects (e.g. dogs and cars) and stuff (e.g. sky and grass). To obtain the geometric layout of an image, other methods outperform Vezhnevets et al. [31]. The average accuracies of Hoiem et al. [11] and Hedau et al. [5] are 72.1% and 74.3% respectively. Using the classifier trained on the combined training set, our average accuracy is 81.2% which outperforms all the other methods. To obtain a fair comparison, we train our model on the dataset of Hoiem et al. [11] and test it

TABLE V

THE ACCURACY OF THE IMAGE-LEVEL 3D LAYOUT. FOR OURS-1, THE STAGE CLASSIFIER IS TRAINED ON [1]. FOR OURS-2, THE STAGE CLASSIFIER IS TRAINED ON EXTENDED DATASET. FOR OURS-GEOCONTEXT, THE STAGE CLASSIFIER IS TRAINED ON THE DATASET OF [11] AND TEST ON OUR DATASET. FOR OURS-GT, WE USE THE GROUND-TRUTH LABEL TO GENERATE THE LAYOUT. NOTE THAT THE CATEGORY *box* ONLY CONTAINS INDOOR IMAGES AND THE OTHER CATEGORIES ONLY CONTAIN OUTDOOR IMAGES

Methods	[30]	[31]	[11]	[5]	Ours-1	Ours-GeoContext	Ours-2	Ours-GT %
Accuracy of stage	-	-	-	-	56.4	57.9	86.9	100
skgGnd	76.2	40.1	90.6	-	76.9	84.4	81.9	89.8
box	59.3	-	22.5	74.3	72.6	-	72.9	84.9
skyBkgGnd	48.9	41.3	88.0	-	59.8	69.5	78.4	82.8
skyGnd	48.0	40.5	87.2	-	81.7	90.7	91.7	93.4
Average	58.1	40.6	72.1	74.3	72.7	81.4	81.2	87.7

on our dataset. The results are shown in table V denoted by $Ours - GeoContext$. Because no Box images exist in the dataset of [11], there are only three categories, namely *skgGnd*, *skyBkgGnd* and *BkgGnd*. It can be derived from Table V (trained on dataset of [11] and tested on our own dataset), that our method obtains good and stable performance showing the generalizability of the proposed method. Note that only the image-level labels are used for training our model (i.e. we manually labeled each image indicating which stage it belongs to) while [11] uses pixel-level ground truth for training. When using the ground-truth stage type, our average accuracy is 87.7%. Recall that our training data only needs image level labeling and for some classes our method can generate more detailed layout (e.g. for the class *box*, our method can infer each plane positioning of the box). Hoiem et al. [11] and Hedau et al. [5] need pixel level ground truth training data to train the classifiers. Since our dataset does not require pixel-level ground-truth which is needed for the methods of [5] and [11]. We use the models provided by [5] and [11] in this experiment.

C. Discussion

Ncut [30] is used to segment images without considering any prior information. Here, the aim is to provide more insight in the performance when considering a geometric prior obtained by stage classification. Hence, the results of Ncut is just to give insight in how much the prior information contributes. Hoiem et al. [11] train a set of classifiers for different classes of patches in terms of the image-level 3D layout of each patch. They compute a mapping of the appearance to the geometric layout for a set of hierarchical segments. Therefore in their framework, the ground truth of the pixel-level layout is needed. In addition, since the mapping function of the image to 3D layout is learned from local segments, high level context of the geometry is hard to exploit. For example, for the class *box* (e.g. corridor), it is very hard to extract the geometry by observing a single segment. However, in our framework, the scene class is first predicated by considering the whole image and the interdependency of the subregions. Then, the image-level 3D layout is inferred by taking into account this prior information. In this way, we combine bottom-up segmentation with the global context of the image. Noteworthy is that Hoiem et al. [11] can only handle outdoor scenes. If indoor images are considered, another dataset of indoor images with ground-truth pixel layout needs to be built to train the classifiers. However, for our algorithm, we only need to train the scene classifier on one dataset with image level labels.

When the stage classification provides poor results, the algorithm segments the image using probably false prior information. In this case, the layout may not be as expected. Actually, the error depends on the extent of confusion between the two classes. For example, if sky-ground is predicted into background-ground, then the algorithm can still correctly segment the image into two parts. But if sky-ground is predicted as a box, then the algorithm segments the image into five parts and will result in a large error. From the confusion matrix, it is shown that the most confused stages are background-ground and sky-background-ground. Around 46.1% of background-ground images are predicted as sky-background-ground. When this happens, the algorithm segments the background-ground images into three parts. The top part of the buildings is assigned to be sky. This will result in error, but this kind of error (predict the top part to be sky) is also present in other algorithms such as Hoiem et.al [11] (see the first row in Fig. 12).

VII. CONCLUSION

In this paper, we have presented a probabilistic model for inferring the rough geometric structure of a single image. A novel graphical model has been proposed in which subtopics are modeled as latent variables. Furthermore, the method exploits an image-level 3D structure prior to infer a pixel-wise 3D layout.

Experimental results show that our approach outperforms state-of-the-art methods in the context of stage classification. By using the stage as prior information, it is shown that pixel-wise 3D layout can be generated in a proper way. Finally, we are able to obtain the pixel-level 3D layout from a single 2D image without pixel-wise ground-truth label for training.

REFERENCES

[1] V. Nedović, A. W. M. Smeulders, A. Redert, and J.-M. Geusebroek, "Stages as models of scene geometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1673–1687, Sep. 2010.

[2] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[3] R. Lu, A. Gijsenij, T. Gevers, D. Xu, V. Nedović, and J.-M. Geusebroek, "Color constancy using 3D scene geometry," in *Proc. IEEE 12th ICCV*, Sep./Oct. 2009, pp. 1749–1756.

[4] N. Elfiky, T. Gevers, A. Gijsenij, and J. Gonzàlez, "Color constancy using 3D scene geometry derived from a single image," *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3855–3868, Sep. 2014.

[5] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *Proc. 12th IEEE ICCV*, Sep. 2009, pp. 1849–1856.

[6] H. Wang, S. Gould, and D. Koller, "Discriminative learning with latent variables for cluttered indoor scene understanding," in *Proc. 11th ECCV*, 2010, pp. 435–449.

[7] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade, "Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces," in *Proc. NIPS*, 2010, pp. 1288–1296.

[8] J. Lin, X. Ji, W. Xu, and Q. Dai, "Absolute depth estimation from a single defocused image," *IEEE Trans. Image Process.*, vol. 22, no. 11, pp. 4545–4550, Nov. 2013.

[9] V. Hedau, D. Hoiem, and D. Forsyth, "Thinking inside the box: Using appearance models and context based on room geometry," in *Proc. 11th ECCV*, 2010, pp. 224–237.

[10] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, May 2008.

[11] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *Int. J. Comput. Vis.*, vol. 75, no. 1, pp. 151–172, 2007.

[12] D. C. Lee, M. Hebert, and T. Kanade, "Geometric reasoning for single image structure recovery," in *Proc. IEEE Conf. CVPR*, Jun. 2009, pp. 2136–2143.

[13] E. Delage, H. Lee, and A. Y. Ng, "A dynamic Bayesian network model for autonomous 3D reconstruction from a single indoor image," in *Proc. IEEE Comput. Soc. Conf. CVPR*, Jun. 2006, pp. 2418–2428.

[14] O. Barinova, V. Konushin, A. Yakubenko, K. Lee, H. Lim, and A. Konushin, "Fast automatic single-view 3D reconstruction of urban scenes," in *Proc. 10th Eur. Conf. Comput. Vis. (ECCV)*, Marseille, France, Oct. 2008, pp. 100–113.

[15] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. CVPR*, Jun. 2006, pp. 2169–2178.

[16] L. Fei-Fei and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *Proc. IEEE Comput. Soc. Conf. CVPR*, Jun. 2005, pp. 524–531.

[17] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Proc. IEEE Conf. CVPR*, Jun. 2009, pp. 413–420.

[18] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Proc. IEEE Conf. CVPR*, Jun. 2010, pp. 3485–3492.

[19] J. Carreira and C. Sminchisescu, "CPMC: Automatic object segmentation using constrained parametric min-cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1312–1328, Jul. 2012.

[20] J. M. Mooij, "libDAI: A free and open source C++ library for discrete approximate inference in graphical models," *J. Mach. Learn. Res.*, vol. 11, pp. 2169–2173, Aug. 2010.

[21] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, Sep. 2005.

[22] C.-N. Yu and T. Joachims, "Learning structural SVMs with latent variables," in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, 2009, pp. 1169–1176.

[23] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1768–1783, Nov. 2006.

[24] P. G. Doyle and J. L. Snell, *Random Walks and Electric Networks*. Washington, DC, USA: MAA, 1984, pp. 101–109.

[25] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bureau Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[26] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *Int. J. Comput. Vis.*, vol. 105, no. 3, pp. 222–245, 2013.

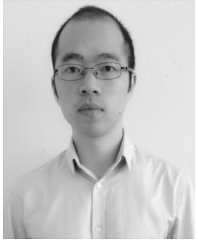
[27] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. CVPR*, Jun. 2005, pp. 886–893.

[28] J. M. Geusebroek and A. W. M. Smeulders, "A six-stimulus theory for stochastic texture," *Int. J. Comput. Vis.*, vol. 62, nos. 1–2, pp. 7–16, 2005.

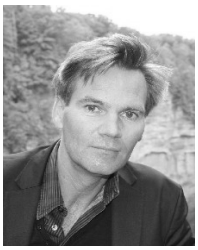
[29] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, 2011, Art. ID 27.

[30] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

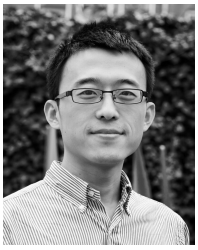
[31] A. Vezhnevets, V. Ferrari, and J. M. Buhmann, "Weakly supervised semantic segmentation with a multi-image model," in *Proc. IEEE ICCV*, Nov. 2011, pp. 643–650.



Zhongyu Lou received the B.Sc. and M.Sc. degrees from Xidian University, Xi'an, China, in 2008 and 2011, respectively. He is currently pursuing the Ph.D. degree with the Intelligent Systems Laboratory Amsterdam, University of Amsterdam, The Netherlands. His main research interests include computer vision, image processing, and machine learning.



Theo Gevers (M'01) is currently a Full Professor of Computer Vision with the University of Amsterdam (UvA), Amsterdam, The Netherlands, and the Computer Vision Center, Universitat Autònoma de Barcelona, Barcelona, Spain. He is a Founder and the Chief Science Officer with Sightcorp, a spinoff of the Intelligent Systems Laboratory, UvA. His main research interests are in the fundamentals of image understanding, object recognition, and color in computer vision. He is interested in different aspects of human behavior, in particular, emotion recognition. He is a Program Committee Member of a number of conferences and an Invited Speaker at major conferences. He is also a Lecturer delivering post-doctoral courses given at various major conferences, including the IEEE Conference on Computer Vision and Pattern Recognition, the International Conference on Pattern Recognition, the International Society for Optics and Photonics, and the Computer Graphics, Imaging, and Vision.



Ninghang Hu received the B.Sc. degree in software engineering and the M.Sc. degree in artificial intelligence from Xidian University, China, in 2008 and 2011, respectively. He is currently pursuing the Ph.D. degree with the Informatics Institute, University of Amsterdam, The Netherlands. His research interests are in machine learning and robot vision, with a focus on human activity recognition and data fusion.