Bottom-Up/Top-Down Image Parsing with Attribute Grammar

Feng Han and Song-Chun Zhu

Departments of Computer Science and Statistics
University of California, Los Angeles, Los Angeles, CA 90095

{hanf, sczhu}@stat.ucla.edu.

Abstract

This paper presents a simple attribute graph grammar as a generative representation for man-made scenes, such as buildings, hallways, kitchens, and living rooms, and studies an effective top-down/bottom-up inference algorithm for parsing images in the process of maximizing a Bayesian posterior probability or equivalently minimizing a description length (MDL). This simple grammar has one class of primitives as its terminal nodes – the projection of planar rectangles in 3-space into the image plane, and six production rules for the spatial layout of the rectangular surfaces. All the terminal and non-terminal nodes in the grammar are described by attributes for their geometric properties and image appearance. Each production rule is associated with some equations that constrain the attributes of a parent node and those of its children. Given an input image, the inference algorithm computes (or constructs) a parse graph, which includes a parse tree for the hierarchical decomposition and a number of spatial constraints. In the inference algorithm, the bottom-up step detects an excessive number of rectangles as weighted candidates, which are sorted in certain order and activate top-down predictions of occluded or missing components through the grammar rules. The whole procedure is, in spirit, similar to the data-driven Markov chain Monte Carlo paradigm [40], [34], except that a greedy algorithm is adopted for simplicity. In the experiment, we show that the grammar and top-down inference can largely improve the performance of bottom-up detection.

This manuscript is submitted to IEEE Trans. on PAMI. A short version was published in ICCV05.

I. Introduction

In real world images, especially man-made scenes, such as buildings, offices, and living spaces, a large number of visual patterns and objects can be decomposed hierarchically into a small number of primitives arranged by a small set of spatial relations. This is similar to language, where a huge set of sentences can be generated from a relatively small vocabulary through some grammar rules that group words to phrases, clauses, and sentences. In this paper, we present a simple attribute graph grammar as a generative image representation and study an effective top-down/bottom-up inference algorithm for parsing images in the process of maximizing a Bayesian posterior probability or equivalently minimizing a description length (MDL).

In the following, we shall briefly introduce the representation and algorithm, and then discuss the literature and our contributions.

A. Overview of the generative representation

Our simple grammar has one root node for the scene, one recursive non-terminal node for objects or surfaces, one class of primitives as its terminal nodes – planar rectangular surfaces projected on images. All the terminal and non-terminal nodes are described by attributes for their geometric properties and image appearance.

The grammar has six production rules for the spatial layout of the rectangular surfaces. Of the six rules, one expands the root node (scene) into m independent objects, and one instantiates a non-terminal node to a primitive. The other four rules arrange the objects or surfaces recursively in four possible ways: (i) aligning m objects in a line, for example, a row of windows in a wall, (ii) nesting one object inside the other, such as a window frame, (iii) align three rectangular surfaces in a cube, and (iv) arranging up to $m \times n$ object in a mesh/tile structure, such as a tile floor. Each production rule is associated with some equations that constrain the attributes of a parent node and those of its children.

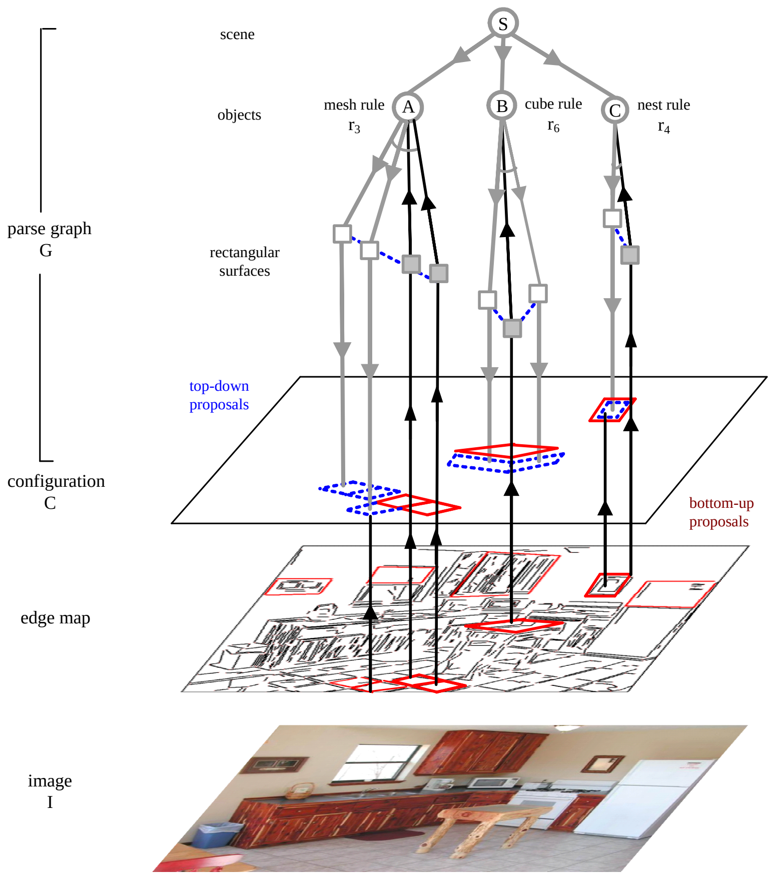


Fig. 1. The hierarchic parse graph is a generative representation and it produces a configuration in the image plane. This configuration generates the image in a primal sketch model. The parse graph is constructed by an iterative top-down/bottom-up algorithm. The rectangular primitives detected in bottom-up step activate the grammar rules to predict missing or occluded components in top-down process.

Given an input image, our objective is to compute a hierarchical parse graph where each non-terminal node corresponds to a production rule. In this parse graph, the vertical links show the decomposition of the scene and objects into their components, and the horizontal (dashed) links specify the spatial relations between components through constraints on their attributes. Fig. 1 illustrates the hierarchical representation for a kitchen scene and the computational algorithm for constructing the parse graph in an iterative bottom-up/top-down procedure.

Note that the parse graph is not pre-determined but constructed "on-the-fly" from the input image. It is also a generative representation instead of a discriminative model. The parse graph produces a planar configuration in the image plane. The configuration consists of rectangular line segments. These line segments are further broken into smaller image primitives for edge elements, bars, and corners in an image primitive dictionary, which, in turn, generate the image by the primal sketch model [10], [11]. Therefore our model (combined with primal sketch model) is fully generative from the scene node to the pixels. This property enables a Bayesian formulation with a prior probability on the parse graph and a likelihood model for the primal sketch (from image primitives to pixels).

B. Overview of the top-down/bottom-up inference algorithm

This paper is focused on designing an effective inference algorithm that integrates top-down and bottom-up inference for attribute grammars. We adopt a greedy algorithm for maximizing the Bayesian posterior probability that proceeds in three phases.

Phase I is bottom-up detection. We compute edge segments from the input image and estimate a number of vanishing points (usually three) in the image using the method studied in [39]. Then the line segments converging to the same vanishing point are put in a line set. The rectangle hypotheses are generated in a method similar to RANSAC [8]. We draw two pairs of line segments from two out of the three line sets, and then evaluate them by the goodness of fit (compatibility) to a rectangle. The two pairs of line segments that pass a minimum compatibility test becomes a weighted hypothesis. We thus generate an excessive number of rectangles as *bottom-up proposals* which may overlap or conflict with each other and are sorted in decreasing order by their weights.

Phase II initializes the terminal nodes of the parse graph in a greedy way. In each step, the algorithm picks the most promising bottom-up rectangle hypothesis with the heaviest weight among all the candidates and accepts it if it increases the Bayesian probability or reduces the description length. Then the weights of all the candidates that overlap or conflict with this accepted rectangle are reduced as in the matching pursuit algorithm [21].

Phase III integrates top-down/bottom-up inference. Each rectangle in the current parse graph matches (often partially) to a production rule with attributes passed to the non-terminal node. These non-terminal nodes are in turn matched to other production rules which then generate *top-down proposals* for predictions, see the downward arrows in Figure 1. The weights of the top-down proposals are calculated based on the posterior probabilities. For example, two adjacent rectangles may activate the line rule (or a mesh rule or a cube rule), which then generates a number of rectangles along the aligned axis. Some of these top-down proposals may have existed in the candidate sets of bottom-up proposal. Such proposals bear both the upward and downward arrows and their weights increase.

In phase III, each of the five grammar rules (omitting the scene rule) maintains a data structure which stores all its weighted candidates. Each step of the algorithm picks the most promising proposal (with the heaviest weight) among all the five candidate sets. This proposal is accepted if it increases the Bayesian probability or reduces the description length. Thus a new non-terminal node is added to the parse graph. This corresponds to recognizing a new sub-configuration and activates the following actions: (i) Create potentially new "top-down" proposals and inserting them into the lists; (ii) Re-weight some proposals in the candidate sets; (iii) Pass attributes between a node and its parent through the constraint equations associated with this production rule.

The top-down and bottom-up computing is illustrated in Fig. 1 for the kitchen scene. For most images, the parse graph has about 3 layers with about 20 nodes, so the computation can be done by AI search algorithms, such as best first search. In our experiments, we observed that the top-down and prior information helps detecting some weak rectangles which are missing in the bottom-up detection. Some "illusory rectangles" could also be hallucinated, especially due to the line and mesh grammar rules. Comparison experiments show that the top-down process improves the performance by a large margin.

C. Related work on attribute grammar, rectangle detection, and image parsing

In the literature, the study of syntactic pattern recognition was pioneered by Fu et al [9], [37], [38], Riseman[12], and Ohta et al [26], [25], and other image understanding systems with top-down/bottom-up inference [2], [14], [15], [7], [22], [23] in the 1970-80s. Its applicability has been limited by two difficulties. The first is known as the "semantic gap". The primitive patterns (terminators) used in their grammar could not be computed reliably from real images. The second is the lack of expressive power of the early work which was mostly focused on string grammars and stochastic context free grammars (SCFG). In recent years, attribute grammars [1] and context sensitive graph grammars [30] have been developed in visual diagrams parsing. In the vision literature, grammars are mostly studied in binary shape recognition, such as the grammars for medial axis [41] and shock graphs [31]. Most recently, there is a resurgence of compositional computing for segmentation [32], [35] and object recognition[17], [4]. However, a more general representation and computational framework has yet been developed. A comprehensive survey of these work is referred to [43].

Detecting rectangular structures in images has been well studied in the vision literature, especially for detecting building roofs in aerial images. One class of methods [18], [20], [33] detects edge segments and line primitives and then groups them into rectangles. The other types of methods [44], [39] use Hough Transforms on edge maps to detect rectangles globally. A Markov chain Monte Carlo method was developed in rectangular scene construction in [5], which also uses compositional structures. Putting detecting rectangle structures in the broader topic of modelling structural variability, our work is also closely related to a variety of representations including shape grammars with algebraic constraints[27], [28], [19].

Our work is also related to some previous work on object recognition[6], [16], and image parsing by data-driven Markov chain Monte Carlo (DDMCMC) [40], [34]. The common goal is to design effective algorithms by integrating bottom-up and top-down steps for inferring single objects or hierarchical image structures. In DDMCMC, each step is made reversible for backtracking and observes the detailed balance equations. Each step chooses a proposal with certain probability and accepts the proposal with a probability. This is often computationally expensive. When the proposal is strong, especially at the early stage of computation as the proposals are sorted in decreasing order, it is often accepted with probability 1. The reversible moves are mostly needed at places where the image is ambiguous. Thus we adopt a greedy algorithm in this paper and accept the proposal deterministically when it increases the posterior probability.

In comparison to the previous work, this paper has the following novel aspects.

1. It extends the representation in image parsing [40], [34] with an attribute grammar, which sets the ground for recognizing generic objects with structural variabilities.

2. It derives a generative model, which is tightly integrated with the primal sketch models [11] to yield a full generative representation from scene to pixels.

3. It develops the bottom-up and top-down mechanism for grammar based image parsing. This strategy has been used in some recent object recognition work [4], [36].

The remainder of the paper is organized as follows. We first present the attribute grammar representation in Section II. Then we derive the probability models and pose the problem as Bayesian inference in Section III. The top-down/bottom-up inference algorithm is presented in Section IV. Some experimental results are shown in Section V. We then conclude the paper with a discussion of future work in Section VI.

II. Attribute graph grammar for scene representation

In this section, we introduce the attribute graph grammar representation to set the background for the probabilistic models in the next section.

A. Attribute graph grammar

An attribute graph grammar is augmented from the stochastic context free grammar by including attributes and constraints on the nodes.

Definition 1: An *attribute graph grammar* is specified by a 5-tuple

$$\mathcal{G} = (V_N, V_T, S, \mathcal{R}, P). \tag{1}$$

V_N and V_T are the sets of non-terminal and terminal nodes respectively, S is the initial node for the scene. $\mathcal{R}$ is a set of production rules for spatial relationships. P is the probability for the grammar.

A non-terminal node is denoted by capital letters $A, A_1, A_2 \in V_N$, and a terminal node is denoted by lower case letters $a, b, c, a_1, a_2 \in V_T$. Both non-terminal and terminal nodes have a vector of attributes denoted by $X(A)$ and $x(a)$ respectively. $\mathcal{R} = \{r_1, r_2, ..., r_m\}$ is a set of production rules expanding a non-terminal node into a number of nodes in $V_N \cup V_T$. Each rule is associated with a number of constraint equations. For example, the following is a rule that expands one node A into two nodes $A_1, A_2 \in V_N$.

$$r: \ A \to (A_1, A_2). \tag{2}$$

The associated equations are constraints on the attributes.

$$g_i(X(A)) = f_i(X(A_1), X(A_2)), \ \ i = 1, 2, ..., n(r). \tag{3}$$

$g_i()$ and $f_i()$ are usually projection functions that take some elements from the attribute vectors. For instance, let $X(A) = (X_1, X_2, X_3)$ and $X(A_1) = (X_{11}, X_{12})$, then an equation could simply be an equivalence constraint (or assignment) for passing the information between nodes A and A_1 in either directions, $X_1 = X_{11}$. In the parsing process, sometimes we know the attributes of a child node X_{11} and then pass it to X_1 in rule r. This is called "bottom-up message passing". Then X_1 may be passed to another child node's attribute X_2 with $X_{21} = X_1$. This is called "top-down message passing".

A production rule may instantiate a non-terminal node to a terminal node

$$r: \ A \to a, \tag{4}$$

with constraints

$$g_i(X(A)) = f_i(x(a)), \quad i = 1, 2, ..., n(r). \tag{5}$$

Definition 2: A *parse graph* $\mathbf{G}$ is a tree-structured representation expanded from a root node S by a sequence of production rules $(\gamma_1, \gamma_2, ..., \gamma_k)$ and augmented with spatial relations and constraints.

Definition 3: A *configuration* C is a set of terminal nodes (rectangles in this paper),

$$C = \{(a_i, x(a_i)) : a_i \in V_T, i = 1, 2, ..., K\}. \tag{6}$$

It is deterministically generated by a parse graph, i.e. $C = C(Graph)$ and its attributes are denoted by $X(C)$.

If a configuration C has multiple parse graph, then the grammar is said to be ambiguous.

Definition 4: A *language* of grammar $\mathcal{G}$ is the set of all valid configurations that can be derived by the production rules starting from a root node S. It is denoted by

$$\Sigma(\mathcal{G}) = \{(C, X(C)) : S \overset{\gamma_1,...,\gamma_k}{\longrightarrow} C, \ \gamma_i \in \mathcal{R}, i = 1, 2, ..., k.\}, \tag{7}$$

B. A class of primitives — rectangles

Our simple grammar uses only one class of primitives – the projection of planar rectangles in 3-space into the image plane. Illustrated in Fig. 2, it has two pairs of parallel line segments in 3D which intersect at two vanishing points v_1, v_2 in the image plane. Therefore, the set of terminal nodes is denoted by

$$V_T = \{(a, x(a)) : x(a) \in \Omega_a\}. \tag{8}$$

There are many equivalent ways to define the attributes $x(a)$ for a rectangle. We choose the variables to simplify the constraint equations, and thus denote a by 8 variables: two vanishing points $v_1 = (x_1, y_1)$ and $v_2 = (x_2, y_2)$, two orientations θ_1 and θ_2 for the two boundaries converging at v_1, and two orientations θ_3 and θ_4 for the two boundaries converging at v_2.

$$x(a) = (x_1, y_1, x_2, y_2, \theta_1, \theta_2, \theta_3, \theta_4). \tag{9}$$

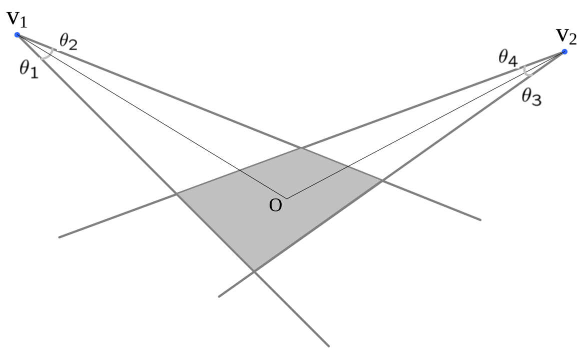


Fig. 2. A planar rectangle (shaded) is described by 8 variables. The two vanishing points $v_1 = (x_1, y_1)$ and $v_2 = (x_2, y_2)$ and four directions θ_1, θ_2, θ_3 and θ_4 at the two vanishing points.

C. Six production rules

As a generic grammar for image interpretation, our representation has the root node S for the scene and one non-terminal nodes A for objects and surfaces.

$$V_N = \{(S, X(S)), (A, X(A)) : X(S) = n, X(A) \in \Omega_A\}. \tag{10}$$

The scene node S generates n independent objects. The object node A can be instantiated (assigned) to a rectangle (rule r_5), or be used recursively by the other four production rules: r_2 – the line production rule, r_3– the mesh production rule, r_4– the nesting production rule, and r_6 –the cube production rule. The six production rules are summarized in Fig. 3.

This simple grammar can generate a language with a huge number of configurations for generic objects and scenes. Figure 4 shows two typical configurations – a floor pattern and a toolbox pattern, and their corresponding parse graphs.

The attribute $X(A) = (\ell(A), n(A), X_o(A))$ includes a label ℓ for the type of object (structure) represented by A, $\ell(A) \in \Omega_\ell = \{\text{line, mesh, nest, rect, cube}\}$, $n(A)$ for the number of children nodes in A, and $X_o(A)$ for its geometric properties and appearance. The variables in $X_o(A)$ depend on object type of A, therefore we denote the attribute space of A as the union of the five different subspaces.

$$\Omega_A = \Omega_A^{\text{line}} \cup \Omega_A^{\text{mesh}} \cup \Omega_A^{\text{nest}} \cup \Omega_A^{\text{rect}} \cup \Omega_A^{\text{cube}} \tag{11}$$

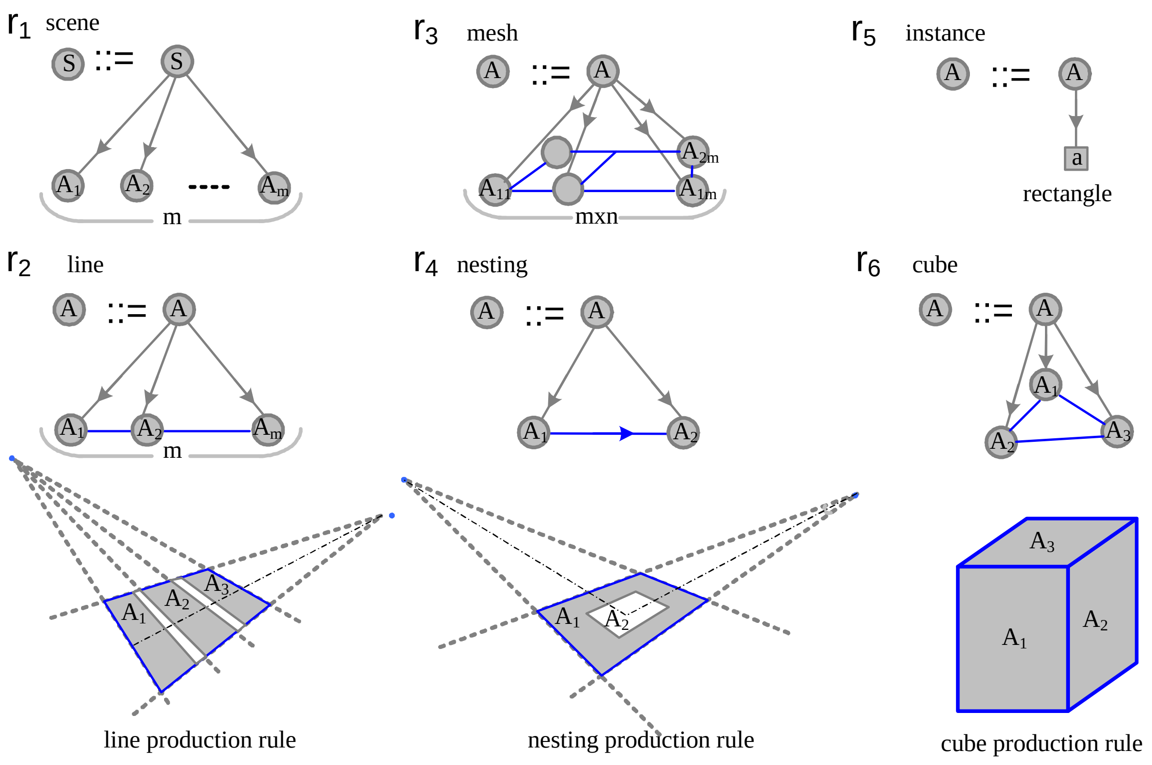


Fig. 3. Six attribute grammar rules. Attributes will be passed between a node to its children and the horizontal lines show constraints on attributes. See text for explanation.

The geometric attributes for all the four different objects (except rectangle) are described as follows. For clarity, we introduce the appearance attributes (intensity) in the next section together with the primal sketch model.

1. For a line object of $n = n(A)$ rectangles,

$$X_o(A) = (v_1, v_2, \theta_1, \theta_2, \theta_3, \theta_4, \tau_1, ..., \tau_{2(n-1)}). \tag{12}$$

The first eight parameters define the bounding box for the n rectangles, and the other $2n-2$ orientations are for the remaining directions of the n rectangles in the object.

2. For a mesh object of up to $n(A) = m \times n$ rectangles,

$$X_o(A) = (v_1, v_2, \theta_1, ..., \theta_4, \tau_{1\cdot 1} ..., \tau_{2(m-1)\cdot(n-1)}) \tag{13}$$

Again, the first eight parameters define the bounding box for the mesh, and the rest includes $2(m-1)(n-1)$ orientations for the remaining directions specifying the individual rectangles

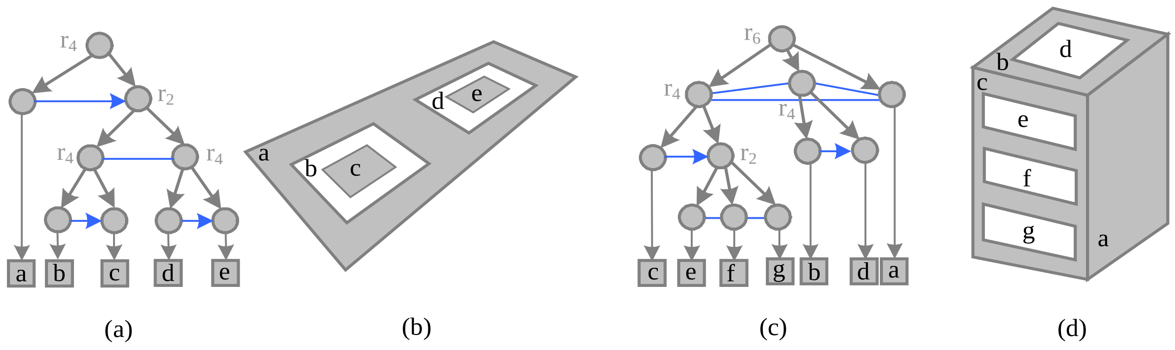


Fig. 4. Two examples of rectangle object configurations (b) and (d) and their corresponding parse graphs (a) and (c). The production rules are shown as non-terminal nodes.

in the object, some of which could be empty.

3. For a nest object with $n(A) = 2$ rectangles, $X_o(A) = (v_1, v_2, \theta_1, ..., \theta_4, \tau_1, ..., \tau_4)$.

4. For a cube object $n(A) = 3$, and $X_o(A) = (v_1, v_2, v_3, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$. It has three vanishing points and 3 pairs of orientation angles.

Remarks. If the rectangles are arranged regularly in the line or mesh objects, for example, equally spaced, then we can omit all the orientations τ_i for defining the individual rectangles. The sharing of bounding boxes and orientations are intrinsic reasons for grouping and composition, as it reduces the description length. The rectangle elements in the above could be the bounding box (hidden) for other objects to allow recursive applications of the rules. In addition to these hard constraints for passing attributes among nodes, we shall introduce probabilities to impose soft constraints on the free variables (mostly the τ's) so that the elements are nearly equally spaced.

In the following we briefly explain the constraint equations associated with the rules. In most cases, the constraint equations are straightforward but tedious to enumerate. Therefore we choose to introduce the typical examples.

The simplest rule is r_5 for instantiation. It assigns a rectangle and the associated attributes to a non-terminal node A. Therefore the constraint equation is simply an assignment for the 8 variables.

$$r_5: \ A \to a; \ X_o(A) = x(a).$$

This assignment may go in either direction in the computation.

For the line production rule r_2, we choose $m = 3$ for simplicity.

$$\begin{aligned} r_2: \quad & A \to (A_1, A_2, A_3); \\ & g_i(X_o(A)) = f_i(X_o(A_1), X_o(A_2), X_o(A_3)), i = 1, 2, ..., k. \end{aligned}$$

A is the bounding rectangle for A_1, A_2, A_3 and shares with them the two vanishing points and 4 orientations. Given $X_o(A)$, the three rectangles A_1, A_2, A_3 have only 4 degrees of freedom for the two intervals, all the other $3 \times 8 - 4 = 20$ attributes are decided by the above attribute equations. One can derive the constraint equations for the other rules in a similar way.

III. Probability models and Bayesian formulation

In the generative model, an input image $\mathbf{I}$ is generated by a sketch C_{sk} which includes the planar configuration $C = C(\mathbf{G})$ produced by parse graph $\mathbf{G}$ and some free sketches (line segments), denoted by C_{free} for other non-rectangular structures.

$$C_{\text{sk}} = (C(\mathbf{G}), C_{\text{free}}). \tag{14}$$

In a Bayesian framework, our objective is to maximize a posterior probability,

$$\mathbf{G}^* = \arg\max p(\mathbf{I}|C_{\text{sk}})p(\mathbf{G})p(C_{\text{free}}). \tag{15}$$

The prior model $p(\mathbf{G})$ is the fifth component in the definition of the grammar $\mathcal{G}$ in eqn. 1. $p(C_{\text{free}})$ follows the primal sketch model. We discuss $p(\mathbf{G})$ and likelihood $p(\mathbf{I}|C_{\text{sk}})$ in the following two subsections.

A. Prior model $p(\mathbf{G})$ for the parse graph

Let $\Delta_N(\mathbf{G})$ and $\Delta_T(\mathbf{G})$ be the sets of non-terminal nodes (including the root node) and terminal nodes respectively in the parse graph $\mathbf{G}$. Then a parse graph includes the following three sets of variables,

$$\mathbf{G} = (\{(\ell(A), n(A), X_o(A)) : \forall A \in \Delta_N(\mathbf{G})\},\ \{X_o(a) : \forall a \in \Delta_T(\mathbf{G}))\tag{16}$$

Due to the hierarchical tree structure, we factorize the probability $p(\mathbf{G})$ as

$$p(\mathbf{G}) = \prod_{A\in\Delta_N(\mathbf{G})} [p(\ell(A))p(n(A)|\ell(A))p(X_o(A)|\ell(A), n(A)) \prod_{B\in\text{child}(A)} p(X_o(B)|X_o(A))]\tag{17}$$

Firstly, $\ell(A)$ is a "switch" variable for selecting one of the rules. We denote the probabilities for the five rules as $q(\ell)$, which sum to one: $\sum_{\ell\in\Omega_\ell} q(\ell) = 1$.

Secondly, at the root S and non-terminal node A, we have a variable $n(A)$ for the number of their children and $p(n(A)|\ell(A))$ penalizes this number (or complexity), such as $p(n(A)|\ell(A)) = \frac{\beta_{\ell(A)}^{n(A)} e^{-\beta_{\ell(A)}}}{n(A)!}$. $p(n(A)|\ell(A))$ is deterministic when A is the cube or the nesting rule:

$p(n(A) = 3|\ell(A) = \text{“}cube\text{”}) = 1$ and $p(n(A) \neq 3|\ell(A) = \text{“}cube\text{”}) = 0$;

$p(n(A) = 2|\ell(A) = \text{“}nest\text{”}) = 1$ and $p(n(A) \neq 2|\ell(A) = \text{“}nest\text{”}) = 0$.

Thirdly, $p(X_o(A)|\ell(A), n(A)) = \frac{1}{Z}e^{-\phi_{\ell(A)}(X_o(A))}$ (the normalization constant Z can be computed empirically either by sampling or integration) is a singleton probability on the geometric and appearance of A. Similarly, we have $p(X_o(B)|X_o(A) = \frac{1}{\pi}e^{-\psi_{\ell}(A)(X_o(A),X_o(B))}$ for those of $A's$ children. The potential functions $\phi()$ and $\psi()$ take quadratic forms to enforce some regularities, such as ensuring that aligned rectangles in a group are having almost the same shape and evenly spaced. For example, the potential functions for a line rule $A \to (A_1, A_2, A_3)$ (with $n(A) = 3$ and the aligning axis being denoted as $\updownarrow$) are:

$$\phi_{line}(X_o(A)) = \sum_{i=1}^{2}(d(X_o(A_i), X_o(A_{i+1})) - \overline{d})^2 + \frac{1}{2}\sum_{i=1}^{3}(w(X_o(A_i)) - \overline{w})^2,$$

$$\overline{d} = \frac{1}{2}\sum_{i=1}^{2} d(X_o(A_i), X_o(A_{i+1})),\ \overline{w} = \frac{1}{3}\sum_{i=1}^{3} w(X_o(A_i)).\tag{18}$$

$$\psi_{line}(X_o(A), X_o(A_i)) = (|\theta_{3i} - \theta_3|^2 + |\theta_{4i} - \theta_4|^2).\tag{19}$$

In eqn. 18, $d(X_o(A_i), X_o(A_j))$ is a function to compute the distance between the neighboring A_i and A_j, and $w(A_i)$ is a function to compute the spanning width of A_i along $\updownarrow$. In eqn. 19, θ_3, θ_4 are the two orientations of the two boundaries of A parallel to $\updownarrow$, while θ_{3i}, θ_{4i} are the two orientations of the two boundaries of A_i parallel to $\updownarrow$. $\phi_{line}(X_o(A))$ globally constrains A_1, A_2, A_3 to have similar shape and be evenly spreading in the line group, while $\psi_{line}(X_o(A), X_o(A_i))$ enforces the individual properties of A_i to fit with respect to the whole group.

In recent work [43], $p(\mathbf{G})$ is generalized to a non-factorized form for context sensitive graph grammar.

B. Likelihood model $p(\mathbf{I}|C_{\text{sk}})$

For the likelihood model, we adopt the primal sketch model for $p(\mathbf{I}|C_{\text{sk}})$, and refer to two previous papers [10], [11] for this model and algorithm. The reconstruction (synthesis) of images using a configuration is shown in the experiment section (See Figs.10 and 11). In the following we briefly introduce the model for the paper to be self-contained.

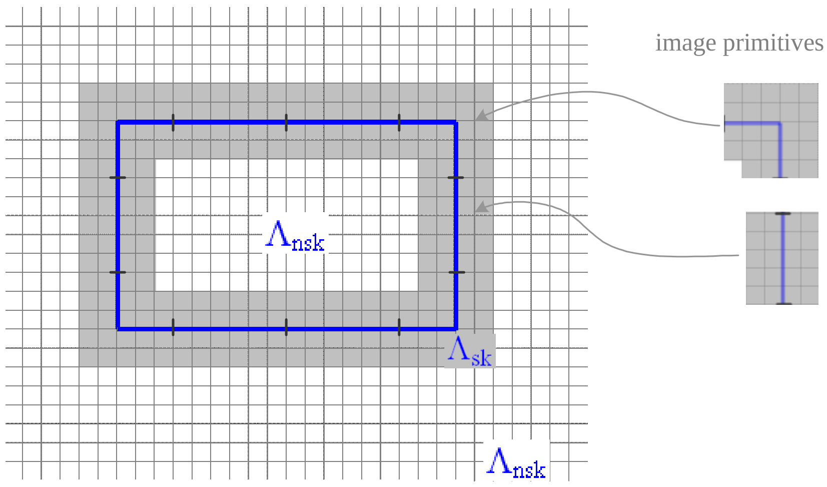


Fig. 5. Partition of image lattice Λ into two parts. The shaded pixels Λ_{sk} around the rectangles and the remaining part Λ_{nsk}. The rectangles are divided into small edge and corner segments. Therefore Λ_{sk} is divided into many image primitives.

$C(\mathbf{G})$ is a set of rectangles in the image plane. Fig.5 shows a rectangle in a lattice. A lattice is denoted by Λ and is divided into two disjoint parts: the sketchable part for the shaded pixels around the rectangles and the non-sketchable part for the remaining part.

$$\Lambda = \Lambda_{\text{sk}} \cup \Lambda_{\text{nsk}}, \quad \Lambda_{\text{sk}} \cap \Lambda_{\text{nsk}} = \emptyset.$$

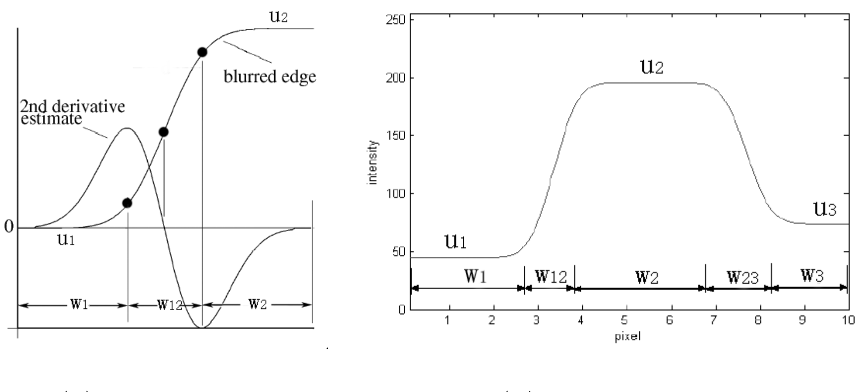
Λ includes pixels which are $2 \sim 5$ pixels away from the rectangle boundaries. The rectangles are divided into short segments of 5-11 pixels long for lines and corners. Therefore Λ_{sk} is divided into N image primitives (patches) of 5×7 pixels along these segments.

$$\Lambda_{\text{sk}} = \cup_{k=1}^{N} \Lambda_{\text{sk,k}}. \tag{20}$$

For example, Fig.5 shows two image primitives: one for line segment and one for a corner. The primal sketch model collects all these primitives in a primitive dictionary represented (clustered) in parametric form,

$$\Delta_{\text{sk}} = \{B_t(u, v; x, y, \tau, \sigma, \Theta) : \forall x, y, \tau, \sigma, \theta, t\}.$$

t indexes the type of primitives, such as edges, bars, corners, crosses etc. (u, v) are the coordinates of the patch centered at (x, y) with scale σ and orientation τ. Θ denotes the parameters for the intensity profiles perpendicular to each line segment. A corner will have two profiles. The intensity profiles along the line segment in a primitive are assumed to be the same.



(a) Edge profile (b) Ridge profile

Fig. 6. The parametric representation of an edge profile (a) and a ridge profile. From Guo, Zhu and Wu, 2005.

Therefore there are two types of profiles as Fig. 6 shows: one is a step edge at various scales (due to blurring effects) and the other is a ridge (bar). The step edge profile is specified with five parameters: $\Theta = (u_1, u_2, w_1, w_{12}, w_2)$, which denotes the left intensity, the right intensity, the width of the left intensity (from the leftmost to the left second derivative extrema), the blurring scale, and the width of the right intensity respectively as shown in Figure 5. The ridge profile is represented by seven parameters: $\Theta = (u_1, u_2, u_3, w_1, w_{12}, w_2, w_{23}, w_3)$. A more detailed description is given in [11]. Using the above edge/ridge model, the 1D intensity function of the profile for the rectangle boundaries can be fully obtained.

Therefore we obtain a generative model for the sketchable part of the image.

$$\mathbf{I}(x,y) = B_{t_k}(x - x_k, y - y_k; \tau_k, \sigma_k, \Theta_k) + n(x,y),\ (x,y) \in \Lambda_{\text{sk,k}},\ \forall k = 1,2,...,N. \tag{21}$$

The residue is assumed to be iid Gaussian noise $n(x,y) \sim G(0, \sigma_o^2)$. This model is sparser than the traditional wavelet representation as each pixel is represented by only one primitive.

As mentioned previously, rectangles are only part of the sketchable structures in the images, though they are the most common structures in man-made scenes. The remaining structures are represented as free sketches which are object boundaries that cannot be grouped into rectangles. These free sketches are also divided into short line segments and therefore represented by image primitives in the same way as the rectangles.

The non-sketchable part is modeled as textures without prominent structures, which are used to fill in the gaps in a way similar to image inpainting. Λ_{nsk} is divided into a number of $M = 3 \sim 5$ disjoint homogeneous texture regions by clustering the filter responses,

$$\Lambda_{\text{nsk}} = \cup_{m=1}^{M} \Lambda_{\text{nsk,m}}.$$

Each texture region is characterized by the histograms of some Gabor filter responses

$$h(\mathbf{I}_{\Lambda_{\text{nsk,m}}}) = \mathbf{h}_m, \quad m = 1,2,...,M.$$

The probability model for the textures are the FRAME model [42] with the Lagrange parameters (vector) β_m as the learned potentials. These textures use the sketchable part Λ_{sk}

as the boundary condition in calculating the filter responses.

In summary, we have the following primal sketch model for the likelihood.

$$p(\mathbf{I}|C_{\text{sk}}) = \frac{1}{Z}\exp\{-\sum_{k=1}^{N}\sum_{(x,y)\in\Lambda_{\text{sk,k}}}\frac{(\mathbf{I}(x,y)-B_k(x,y))^2}{2\sigma_o^2} - \sum_{m=1}^{M}\langle\beta_m, h(\mathbf{I}_{\Lambda_{\text{nsk,m}}})\rangle\} \qquad (22)$$

The above likelihood is based on the concept of primitives, not rectangles. Therefore the recognition of rectangles or larger structures (cube, mesh, etc.) only affects the likelihood locally. In other words, our parse graph is built on the primal sketch representation. This is important in designing an effective inference algorithm in the next section.

One may argue for a region based representation by assuming homogeneous intensities within each rectangles. We find that the primal sketch has the following advantages over a region based representation: (1) The intensity inside a rectangle can be rather complex to model, as it may include shading effects, textures and surface markings. (2) The rectangles are occluding each other. One has to infer a partial order relation between the rectangles (i.e. a layer representation) so that the region based model can be applied properly. This needs extra computation. (3) Besides all the regions covered by the rectangles, one still needs to model the background. Thus the detection of a rectangle must be associated with fitting the likelihood for the rectangle region. In comparison, the primal sketch model largely reduces the computation.

IV. Inference algorithm

Our objective is to compute a parse graph $\mathbf{G}$ by maximizing the posterior probability formulated in the previous section. The algorithm should achieve two difficult goals: (i) Constructing the parse graph, whose structure is not pre-determined but constructed "on-the-fly" from the input image and primal sketch representation. (ii) Estimating and passing the attributes in the parse graph.

There are several ways to infer the optimal parse graph, and Data-Drive Markov Chain Monte Carlo (DDMCMC) has been used in [40], [34]. In this paper, our domain is limited to rectangle scenes, and the parse graph is not too big (usually ~ 20 nodes). Thus, the best first search algorithm in artificial intelligence can be directly applied to compute the parse graph by maximizing the posterior probability in a steepest ascent way. This algorithm is, in spirit, very similar to DDMCMC.

Our algorithm consists of three phases. In phase I, we compute a primal sketch representation, and initialize the configuration to the free sketches. Then a number of rectangle proposals are generated from the sketch by a bottom-up detection algorithm. In phase II, we adopt a simplified generative model by assuming independent rectangles (only r_5 and r_1 are considered). Thus we recognize a number of rectangles proposed in phase I to initialize rule r_5 in the parse graph. The algorithm in phase II is very much like matching pursuit [21]. Finally phase III constructs the parse graph with bottom-up/top-down mechanisms.

A. Phase I: primal sketch and bottom-up rectangle detection

We start with edge detection and edge tracing to get a number of long contours. Then we compute a primal sketch representation $C_{\rm sk}$ using the likelihood model in eqn. 22. We segment each long contour into a number of n straight line segments by polygon-approximation. In man-made scenes, the majority of line segments are aligned with one of three principal directions and each group of parallel lines intersect at a vanishing point due to perspective projection. We define all lines ending at a vanishing point to be a parallel line group. A rectangle has two pairs of parallel lines which belong to two separate parallel line groups. We run the vanishing point estimation algorithm [39] to group all the line segments into three groups corresponding to the principal directions. With these three line groups, we generate the rectangle hypotheses as in RANSAC [8]. We exhaustively choose two lines candidates from each set as shown in Fig.7.(a), and run some simple compatibility tests on their positions to see whether two pairs of lines delineate a valid rectangle. For example, the two pairs of line segments should not intersect each other as shown in Fig.7.(b). This will eliminate some obviously inadequate hypotheses.

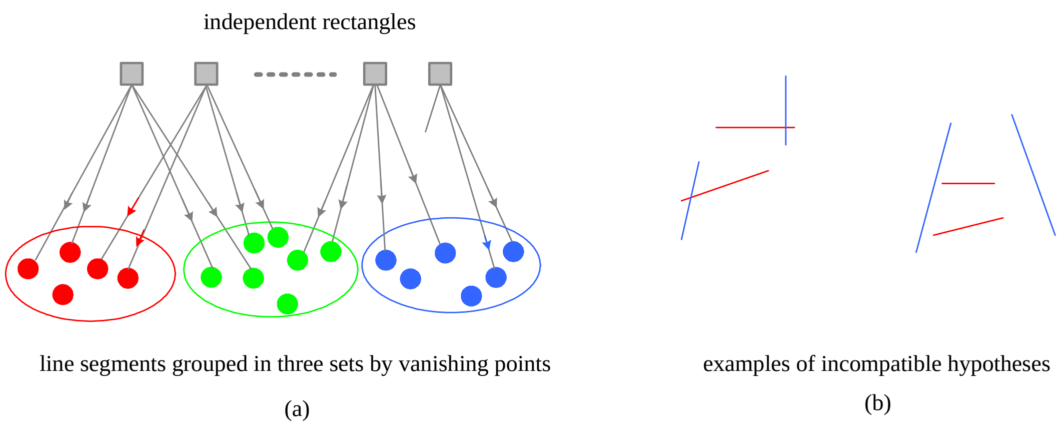

independent rectangles
line segments grouped in three sets by vanishing points
(a)
examples of incompatible hypotheses
(b)

Fig. 7. Bottom-up rectangle detection. The n line segments are grouped into three sets according to their vanishing points. Each rectangle consists of 2 pairs of nearly parallel line segments (represented by a small circle).

This yields an excessive number of **bottom-up** rectangle candidates denoted by

$$\Phi = \{\pi_1, ..., \pi_L\}.$$

These candidates may conflict with each other. For example, two candidate rectangles may share two or more edge segments and only one of them should appear. We mark this conflicting relation among all the candidates. Thus if one candidate is accepted in the later stage, those conflicting candidates will be downgraded or eliminated.

B. Phase II: pursuing independent rectangles to initialize the parse graph

The computation in phase I results in a free sketch configuration $C_{\text{sk}} = C_{\text{free}}$, $C(\mathbf{G}) = \emptyset$, and a set of rectangle candidates Φ. In phase II, we shall initialize the terminal nodes of the parse graph.

We adopt a simplified model which uses only two rules r_1 and r_5. This model assumes the scene consists of a number of independent rectangles selected from Φ which explain away some line segments and the remaining lines are free sketches. A similar model has been used on signal decomposition with wavelets and sparse coding, thus our method for selecting the rectangles is similar to the matching pursuit algorithm [21].

```
*Rectangle Pursuit: initialize the terminal nodes of* $\mathbf{G}$
Input candidate set $\Phi = \{\pi_1, \pi_2, ..., \pi_M\}$ from phase I.
1. Initialize parse graph $\mathbf{G} \leftarrow \emptyset$, $m = 0$.
2. Compute weight $\omega_i$ for $\pi_i \in \Phi$, $i = 1, ..., |\Phi|$, thus obtain
   $\{(\pi_i, \omega_i) : i = 1, 2, ..., |\Phi|\}$.
3. Select a rectangle $\pi_+$ with the highest weight in $\Phi$,
   $\omega(\pi_+) = \max\{\omega(\pi) : \pi \in \Phi\}$.
4. Create a non-terminal node $A_+$ in graph $\mathbf{G}$,
   $\mathbf{G} \leftarrow \mathbf{G} \cup \{A_+\}$,
   $\Phi \leftarrow \Phi \setminus \{\pi_+\}, m \leftarrow m + 1$.
   $C(\mathbf{G}) \leftarrow C(\mathbf{G}) \cup \{\pi_+\}$.
5. Update the weights $\omega(\pi)$ for $\pi \in \Phi$ if $\pi$ overlaps with $\pi_+$
6. Repeat 3-5 until $\omega(\pi_+) \leq \delta_0$.
Output a set of independent rectangles $\mathbf{G} = \{A_1, A_2, ..., A_m\}$.
```

In the following, we calculate the weight $\omega(\pi)$ for each rectangle $\pi \in \Phi$ and the weight change.

A rectangle $\pi \in \Phi$ is represented by a number of short line segments and corners (primitives) denoted by $L(\pi)$, some of which are detected in C_{free} and some of which are missing. The missing components are the missing edges or gaps between primitives in C_{free}. Thus we define two sets

$$L(\pi) = L_{\text{on}}(\pi) \cup L_{\text{off}}(\pi), \text{ with } L_{\text{on}}(\pi) = L(\pi) \cap C_{\text{free}}.$$

Suppose at step m, the current representation includes a number of rectangles in $C(\mathbf{G})$ and a free sketch C_{free}.

$$\mathbf{G},\ C_{\text{sk}} = (C(\mathbf{G}), C_{\text{free}}).$$

Steps 3-4 in the above pursuit algorithm select π_+, then the new representation will be

$$\mathbf{G}' = \mathbf{G} \cup \{A_+\}, \;\; C(\mathbf{G}') = C(\mathbf{G}) \cup L(\pi_+), \; C'_{\text{free}} = C_{\text{free}} \setminus L_{\text{on}}(\pi_+), \; C'_{\text{sk}} = (C(\mathbf{G}'), C'_{\text{free}})$$

The weight of π_+ will be the change (or equally the log-ratio) of log-posterior probabilities in eqn. (15),

$$\omega(b_+) = \log[\frac{p(\mathbf{I}|C'_{\text{sk}})}{p(\mathbf{I}|C_{\text{sk}})} \cdot \frac{p(G')}{p(G)} \cdot \frac{p(C'_{\text{free}})}{p(C_{\text{free}})}] \tag{23}$$

Choosing a rectangle π_+ with the largest weight $\omega(\pi_+) > 0$ increases the posterior probability in a greedy fashion. The weight can be interpreted in three terms and are computed easily. The first term $\log \frac{p(\mathbf{I}|C'_{\text{sk}})}{p(\mathbf{I}|C_{\text{sk}})}$ measures the changes of the log-likelihood in a small domain covered by the primitives in $L_{\text{off}}(\pi_+)$. Pixels in this domain belonged to Λ_{nsk} before and are in Λ_{rmsk} after adding π_+. The likelihood does not change for any other pixels. The second term $\log \frac{p(G')}{p(G)}$ penalizes the model complexity of rectangles (see eqn 17). The third term $\log \frac{p(C'_{\text{free}})}{p(C_{\text{free}})}$ awards the reduction of complexity in the free sketch.

The above weights are computed independently for each $\pi \in \Phi$. After adding π_+ in step 5 we should update the weight $\omega(\pi) \in \Omega$ if π overlaps with π_+, i.e.

$$L(\pi) \cap L(\pi_+) \neq \emptyset.$$

because the update of C_{free} and $C(\mathbf{G})$ in step 4 changes the first and third terms in calculating $\omega(\pi)$ in eqn 23. This update of weight involves only a local computation on $L(\pi) \cap L(\pi_+)$. When we detect the rectangles in phase I, we have computed the overlapping information. This weight update was used in wavelet pursuit where it is interpreted as "lateral inhibition" in neuroscience.

C. Phase III: Bottom-up and top-down construction of parse graph

The algorithm for constructing the parse graph adopts a similar greedy method as in phase II. In phase III, we include the four other production rules r_2, r_3, r_4, r_6 and use the top-down mechanism for computing rectangles which may have been missed in bottom-up detection. We start with an illustration of the algorithm for the kitchen scene.

In Fig. 1, the four rectangles (in red) are detected and accepted in the bottom-up phases I-II. They generate a number of candidates for larger groups using the production rules, and three of these candidates are shown as non-terminal nodes A, B, and C respectively. We denote each candidate by

$$\Pi = (r_\Pi, A_{(1)}, ..., A_{(n_\Pi)}, B_{(1)}, ..., B_{(k_\Pi)}).$$

In the above notation, r_Π is the production rule for the group. It represents a type of spatial layout or relationship of its components. For example, A, B, C in Fig. 1 use the mesh r_3, cube r_6, and nesting r_4 rules respectively. In Π, $A_{(i)}, i = 1, 2, ..., n_\Pi$ are the existing non-terminal nodes in $\mathbf{G}$ which satisfy the constraint equations of rule r. $A_{(i)}$ can be either a non-terminal rectangle accepted by rule r_5 in phase II or the bounding box of a non-terminal node with three rules r_2, r_3, r_4. The cube object does not have a natural bounding box. We call $A_{(i)}, i = 1, 2, ..., n_\Pi$ the bottom-up nodes for Π and they are illustrated by the upward arrows in Fig. 1. In contrast, $B_{(j)}, j = 1, 2, ..., k_\Pi$ are the top-down non-terminal nodes predicted by rule r_Π, and they are shown by the blue rectangles in Fig. 1 with downward arrows. Some of the top-down rectangles may have already existed in the candidate set Φ but have not been accepted in Phase II or simply do not participate in the bottom-up proposal of Π. Such nodes bear both upward and downward arrows.

Fig. 8 shows the five candidate sets for the five rules. Ψ_i is the candidate set of rule r_i for $i = 2, 3, 4, 6$ respectively. Each candidate $\Pi \in \Psi_i$ is shown by an ellipse containing a number of circles $A_{(1)}, i = 1, ..., n_\Pi$ (with red upward arrows) and $B_{(j)}, j = 1, ..., k_\Pi$ (with blue downward arrows). These candidates are weighted in a similar way as the rectangles in Φ by the log-posterior probability ratio.

$$\Psi_i = \{(\Pi_j, \omega_j) : i = 1, 2, ...N_i\}, i = 2, 3, 4, 6.$$

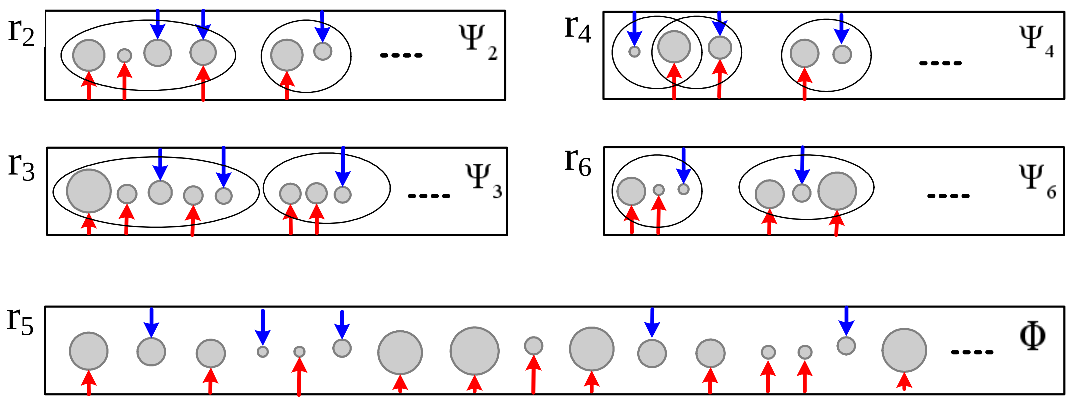


Fig. 8. Four sets of proposed candidates $\Psi_2, \Psi_3, \Psi_4, \Psi_6$ for production rules r_2, r_3, r_4, r_6 respectively and the candidate set Φ for the instantiation rule r_5. Each circle represents a rectangle π or a bounding box of a non-terminal node. The size of the circle represents its weight $\omega(\pi)$. Each ellipse in $\Psi_2, \Psi_3, \Psi_4, \Psi_6$ stands for a candidate Π which consists of a few circles. A circle may participate in more than one candidate.

$\Phi = \{(\pi_i, \omega_i) : i = 1, 2, ..., M\}$ for rule r_5 has been discussed in Phase II. Now Φ also contains top-down candidates shown by the circles with downward arrows. They are generated by other rules. A non-terminal node A in graph $\mathbf{G}$ may participate in more than one group of candidate Π's, just as a line segment may be part of multiple rectangle candidate π's. This creates overlaps between the candidates and needs to be resolved in a generative model.

At each step the parsing algorithm will choose the candidate with the largest weight from the five candidate sets and add a new non-terminal node to the parse graph. If the candidate is $\pi \in \phi$, it means accepting a new rectangle. Otherwise the candidate is a larger structure Π, and the algorithm creates a non-terminal node of type r by grouping the existing nodes $A_{(i)}, i = 1, 2, ..., n_\Pi$ and inserts the top-down rectangles $B_{(j)}, j = 1, ..., k_\Pi$ into the candidate set Φ.

The key part of the algorithm is to generate proposals for π's and Π's and maintain the five weighted candidate sets $\Phi, \Psi_i, i = 2, 3, 4, 6$ at each step. We summarize the algorithm as follows:

```
*The algorithm for constructing the parse graph* $\mathbf{G}$
Input $\mathbf{G} = \{A_1, ..., A_m\}$ from phase II and $\Phi = \{(\pi_i, \omega_i) : i = 1, ..., M - m\}$ from phase I.
1. For rule $r_i, i = 2, 3, 4, 6$.
    Create candidate set $\Psi_i = \text{Proposal}(\mathbf{G}, r_i)$.
    Compute the weight $\omega(\Pi)$ for $\Pi \in \Psi_i$.
2. Select a candidate with the heaviest weight, create a new node $A_+$ with bounding box.
    $\omega_+(A_+) = \max\{\omega(A) : A \in \Phi \cup \Psi_2 \cup \Psi_3 \cup \Psi_4 \cup \Psi_6\}$.
3. Insert $A_+$ to the parse graph $\mathbf{G}$ $\mathbf{G} \leftarrow \mathbf{G} \cup \{A_+\}$.
4. Set the parent node of $A_+$ to the non-terminal node which proposed $A_+$ in the
    top-down phase or to the root $S$ if $A_+$ was not proposed in top-down.
5. If $A_+ = \pi \in \Phi$ is a single rectangle, then
    Add the rectangle to the configuration: $C(\mathbf{G}) \leftarrow C(\mathbf{G}) \cup \{\pi_+\}$.
6. else $A_+ = \Pi = (r_\Pi, A_{(1)}, ..., A_{(n_\Pi)}, B_{(1)}, ..., B_{(k_\Pi)})$, then
    Set $A_+$ as the parent node of $A_{1(\Pi)}, ..., A_{n(\Pi)}$
    Insert top-down candidates $B_{(1)}, ..., B_{(k_\Pi)}$ in $\Phi$ with parent nodes $A_+$.
7. Augment the candidate sets $\Psi_i, i = 2, 3, 4, 6$ with the new node $A_+$.
8. Compute weights for the new candidates and update $\omega(\Pi)$ if $\Pi$ overlaps with $A_+$.
9. Repeat 2-8 until $\omega_+$ is smaller than a threshold $\delta_1$.
Output a parse graph $\mathbf{G}$.
```

Fig. 9 shows a snapshot of one iteration of the algorithm on the kitchen scene. Fig. 9.(b) is a subset of rectangle candidates Φ detected in phase I. We show a subset for clarity. At the end of phase II, we obtain a parse graph $\mathbf{G} = \{A_1, A_2, ..., A_{21}\}$ whose configuration $C(\mathbf{G})$ is shown in (c). By calling the function Proposal$(\mathbf{G}, r_i)$, we obtain the candidate sets $\Psi_i, i = 2, 3, 4, 6$. The candidate sets are shown in (d-f). For each candidate $\Pi = (r_\Pi, A_{(1)}, ..., A_{(n_\Pi)}, B_{(1)}, ..., B_{(k_\Pi)}))$, $A_{(i)}, i = 1, 2, ..., n_\Pi$ are shown in red and

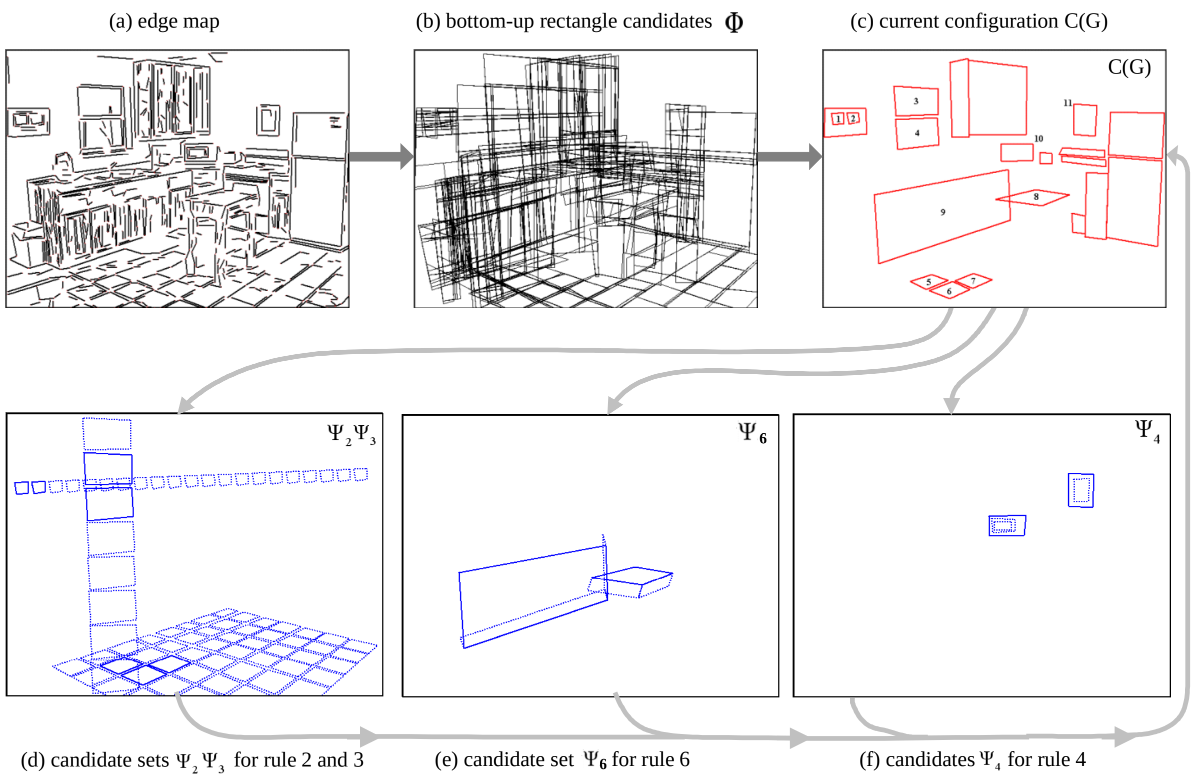

(a) edge map
(b) bottom-up rectangle candidates Φ
(c) current configuration C(G)
C(G)
Ψ2Ψ3
Ψ6
Ψ4
(d) candidate sets Ψ2,Ψ3 for rule 2 and 3
(e) candidate set Ψ6 for rule 6
(f) candidates Ψ4 for rule 4

Fig. 9. A kitchen scene as running example. (a) is the edge map, (b) is a subset of Φ for rectangle candidates detected in phase I. We show a subset for clarity. (c) is the configuration $C(\mathbf{G})$ with a number of accepted rectangles in phase II. (d-f) are candidates in $\Psi_2, \Psi_3, \Psi_4, \Psi_6$ respectively. They are proposed based on the current node in $\mathbf{G}$ (i.e. shown in (b)).

$B_{(j)}, j = 1, 2, ..., k_\pi$ are shown in blue.

The function Proposal$(\mathbf{G}, r_i)$ for generating candidates from the current nodes $\mathbf{G} = \{A_i : i = 1, 2, ..., m\}$ using r_i is not so hard, because the set $|\mathbf{G}|$ is relatively small $(m < 50)$ in almost all examples. Each A_i has a bounding box (except the cubes) with 8 parameters for the two vanishing points and 4 orientations. We can simply test any two nodes A_i, A_j by the constraint equations of r_i. It is worth mentioning that each $A \in \mathbf{G}$ alone creates a candidate Π for each rule r_2, r_3, r_4, r_6 with $n(\Pi) = 1$. In such cases, the top-down proposals $B_{(j)}, j = 1, ..., k_\Pi$ are created using both the constraint equations of r_i and the edge maps. For example, based on one rectangle A_8, the top of the kitchen table in Fig.9.(c), it proposes two rectangles by the cube rule r_6 in Fig.9.(f). The parameters of those two rectangles are decided by the constraint equations of r_6 and the edges in the images.

The algorithm for constructing the hierarchical parse graph is similar to the DDMCMC algorithm[40], [34], except that we adopt a deterministic strategy in this paper in generating the candidates and accepting the proposal. As the acceptance is not reversible, it is likely to get locally optimal solutions.

V. Experiments

We test our algorithm on a number of scenes with rectangle structures and show both qualitative results through image reconstruction (or synthesis) using the generative model and quantitative results through an ROC curve comparing the performance of two approaches: (i) pure bottom-up rectangle detection, and (ii) our methods.

1. Qualitative results. We show six results of the computed configurations and synthesized images in Figures 10 and 11. In these two figures, the first row shows the input images, the second row shows the edge detection results, the third row shows the detected and grouped rectangles in the final configurations and missing rectangles compared with the ground truth (with true positive, false positives and missing rectangles being shown in different line styles), and the fourth row are the reconstructed images based on the rectangle results in the third row. We can see that the reconstructed images miss some structures. Then we add the generic sketches (curves) in the edges, and final reconstructions are shown in the last row.

The image reconstruction proceeds in the following way. First, for the sketchable parts, we reconstruct the image from the image primitives after fitting some parameters for the intensity profiles. For the remaining area Λ_{nsk}, we follow [10] and divide Λ_{nsk} into homogeneous texture regions by k-means clustering and then synthesize each texture region by sampling the Julesz ensemble so that the synthesized image has histograms matching the observed histograms of filter responses. More specifically, we compute the histograms of the derivative filters within a local window (e.g. 7×7 pixels). For example, we use 7 filters and

7 bins are used for each histogram, then in total we have a 49-dimensional feature vector at each pixel. We then cluster these feature vectors into different regions.

In the computed configurations, some rectangles are missing due to the strong occlusion. For instance, some rectangles on the floor in the kitchen scene are missing due to the occlusion caused by the table on the floor. In addition, the results clearly show that high level knowledge introduced by the graph grammar greatly improves the results. For example, in the building scene at the third column in Fig. 10, the windows become very weak on the left side of the image. By grouping them into a line rectangle group, the algorithm can recover these weak windows, which will not appear using the likelihood model alone.

During our experiments, Phase I is the most time-consuming stage and takes about 2 minutes on a 640x480 image since we have to test many combinations to generate all the rectangle proposals and build up their occlusion relations. Phase II and III are very fast and take about 1 minute altogether.

2. Quantitative evaluation To evaluate our algorithm in a quantitative way, we collect a dataset with 40 images. Six have been shown in Figures 10 and 11. We then manually annotate these images to get the ground truth for all the rectangles in each image.

Then we randomly select 15 images from this dataset as training data to tune all the parameters and thresholds in our algorithm. After that, we run the Phase II and then Phase III of our algorithm on the rest images to generate detection results (Please note: the detection results shown in Figures 10 and 11 are obtained when these six images are in testing data). Due to inherent randomness in splitting the dataset into training data and testing data, we repeat the experiment 6 times. Figure 12 shows the ROC curves with confidence intervals [24] for Phase II (using bottom-up only) and Phase III (using both bottom-up and top-down), which are obtained by changing the threshold in Phase II. From these ROC curves, we can clearly see the dramatic improvement by using top-down mechanism over the traditionally bottom-up mechanism only. Intuitively, some rectangles

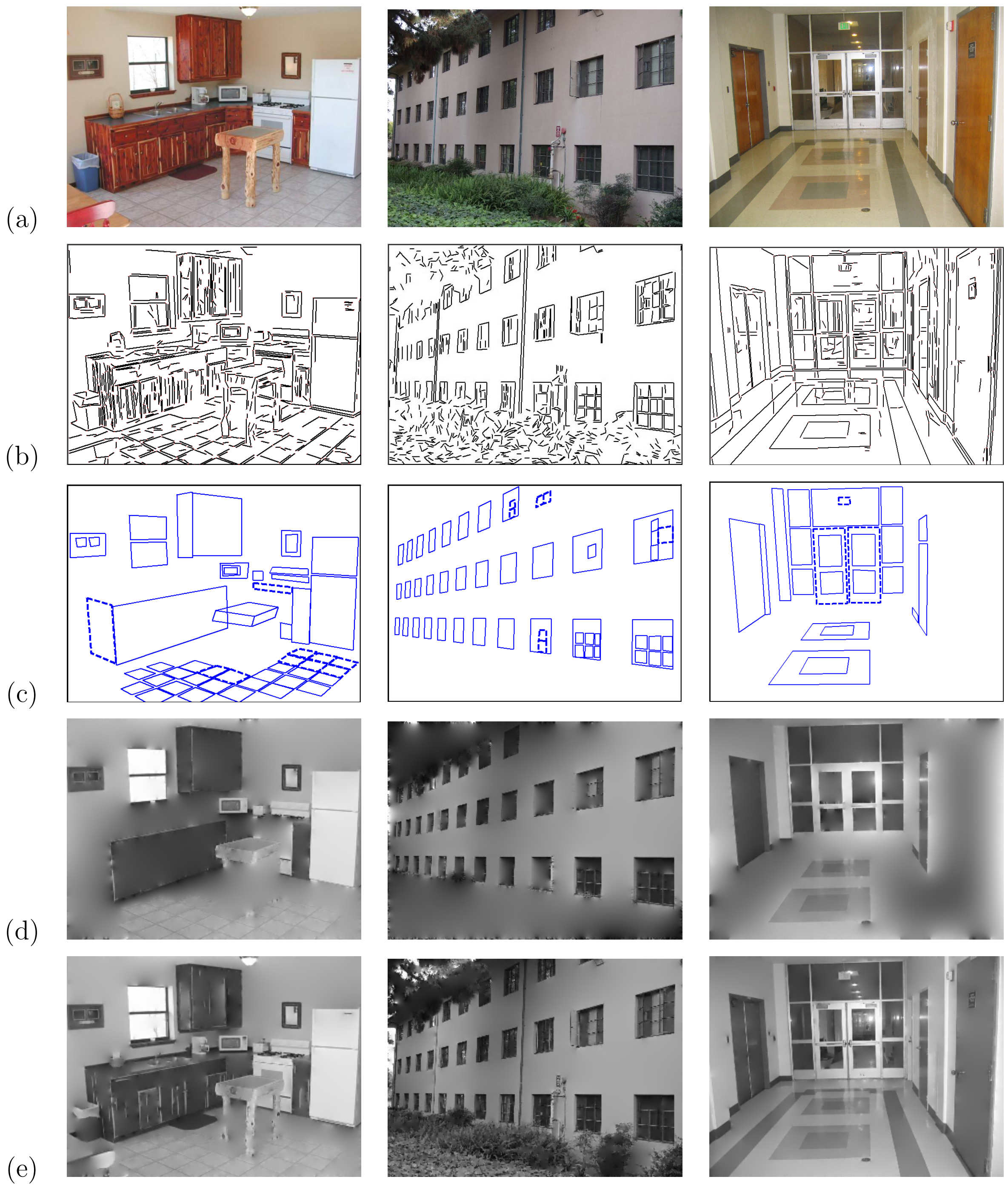


Fig. 10. Some experimental results. (a) Input image. (b) Edge map. (c) Computed rectangle configurations and missing rectangles compared with the ground truth: true positive rectangles are shown with solid lines, false positive rectangles are shown in dotted lines and missing rectangles are shown in dashed lines. (d) Reconstructed image from the primal sketch model using the rectangle configurations only. (e) Reconstructed images after adding some background sketches to the configurations.

(a)
(b)
(c)
(d)
(e)

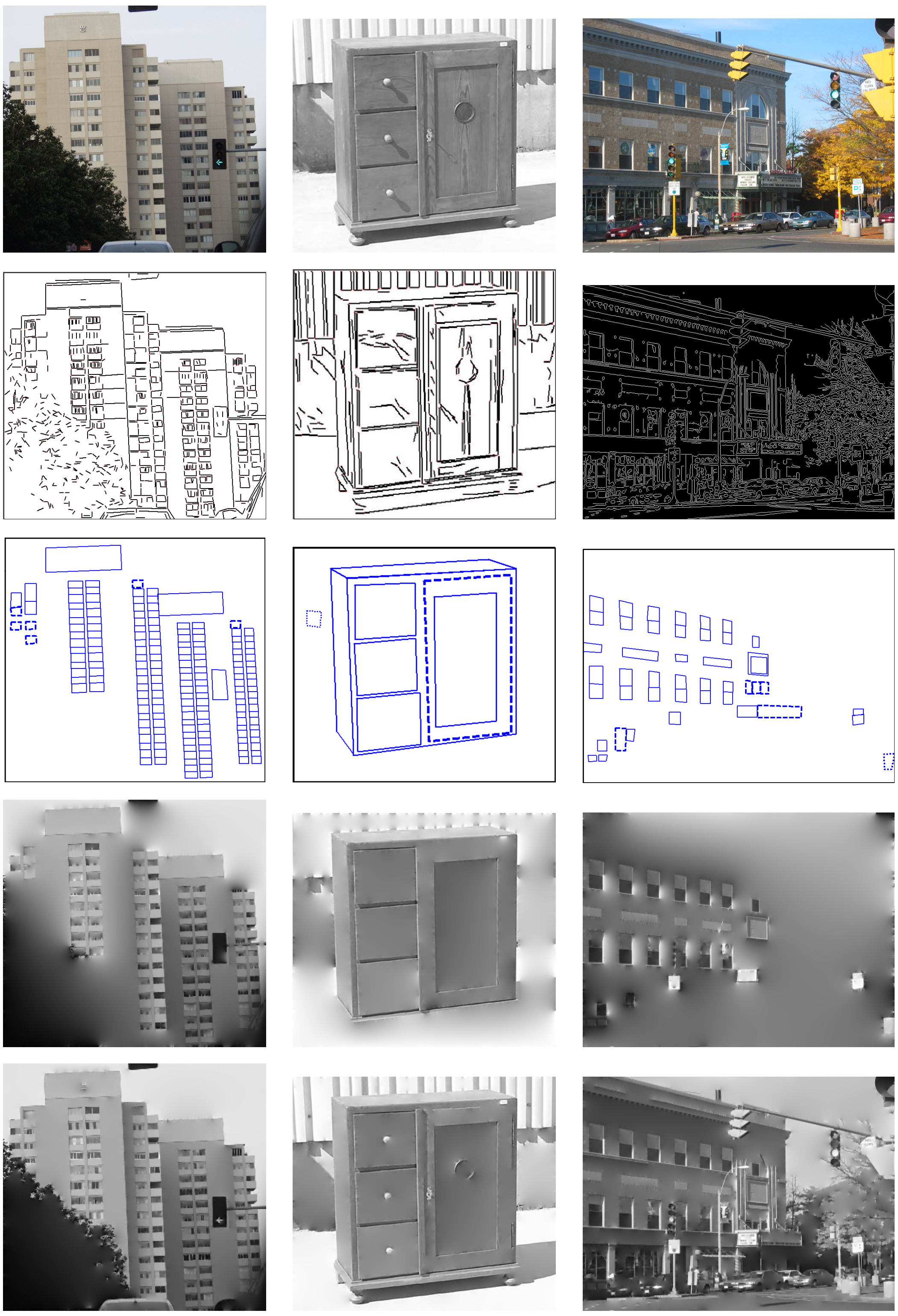


Fig. 11. More experimental results. (a) Input image. (b) Edge map. (c) Computed rectangle configurations and missing rectangles compared with the ground truth: true positive rectangles are shown with solid lines, false positive rectangles are shown in dotted lines and missing rectangles are shown in dashed lines. (d) Reconstructed image from the primal sketch model using the rectangle configurations only. (e) Reconstructed images after adding some background sketches to the configurations.

are nearly impossible to detect using the bottom-up methods and can only be recovered through the context information using the grammar rules.

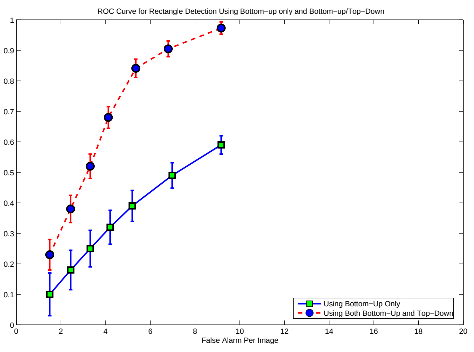


Fig. 12. ROC curves for the rectangle detection results by using bottom-up only and, using both bottom-up and top-down.

To plot the ROC curves, we need to classify each detected rectangle as either a true positive or false alarm in comparison to a rectangle in the ground truth. To be considered as a correct detection, the area of overlap between the detected rectangle a_{det} and the ground truth rectangle a_{gt} is required to exceed 95%,

$$\gamma = \frac{Area(a_{\text{det}} \cap a_{\text{gt}})}{Area(a_{\text{det}} \cup a_{\text{gt}})} > 0.95$$

VI. Discussion

In this paper, we study an attribute grammar for image parsing in man-made scenes. The paper makes two main contributions to the vision literature. First, it uses an attributed grammar to incorporate prior knowledge. Such grammar representations have long been desired for high level vision, especially scene understanding and parsing. Secondly, it integrates a top-down/bottom-up procedure for computing the parse graph with grammars. It extends the previous DDMCMC image parsing work [34] by including more flexible and hierarchical representations. The computing algorithm is compatible with the DDMCMC scheme but we use deterministic ordering for efficiency considerations.

For future work, we shall study the following three aspects: (i) The image parsing is only for generic image interpretation in the current work. In ongoing projects, we are extending this framework to recognizing object categories [43], especially functional objects where objects within each category exhibit a wide range of structural variabilities [4]. The extended grammar will have many more production rules. (ii) In the current work, we manually tune some probabilities and parameters in the energy function. These parameters should be learned automatically when we have a large number of manually parsed training examples, e.g. through supervised learning. We are currently collecting a large manually parsed image dataset for learning grammars. An automatic learning algorithm is presented in a recent work [29], [43]. In experiments, we observe that the stopping threshold Δ_0 and Δ_1 in phase II and phase III have to be decided by minimizing the detection errors (missing rate and false alarm) and cannot be decided by the posterior probability alone.

Acknowledgements

This work was supported in part by NSF grant IIS-0413214 and an ONR grant N00014-05-01-0543. We'd like to thank the Lotus Hill Institute for assistant in preparing dataset and the Lotus Hill data annotation project is supported partially by a subaward of the W.M. Keck foundation and a Chinese 863 grant 2006AA01Z121. We also would like to thank the anonymous reviewers, whose constructive comments help improve the paper in many aspects.

References

[1] S. Baumann, "A simplified attribute graph grammar for high level music recognition", *Third Int'l Conf. on Document Analysis and Recognition*, 1995.

[2] R. Brooks, "Symbolic Reasoning Among 3D Models and 2D Images", *Stanford AIM-343*, STAN-CS-81-861, 1981.

[3] O. Carmichael and M. Hebert, "Shape-based recognition of wiry objects", *IEEE Trans. on PAMI*, 26(12): 1537-1552, December 2004.

[4] H. Chen, Z.J. Xu, and S.C. Zhu, "Composite templates for cloth modeling and sketching," *IEEE Conf. on Computer Vision and Pattern Recognition*, June, 2006.

[5] A. R. Dick, P.H.S. Torr and R. Cipolla. "Modeling and Interpretation of Architecture from Several Images", *Int'l Journal of Computer Vision*, 60(2), pages 111-134, 2004.

[6] S. Dickinson, A. Pentland, and A. Rosenfeld, "3-D Shape Recovery using Distributed Aspect Matching", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), pp. 174–198, 1992.

[7] T. Fan, G. Medioni and R. Nevatia, "Recognizing 3-D Objects Using Surface Descriptions", *IEEE Trans. Pattern Anal. Mach. Intell.* 11(11), pp. 1140-1157, 1989.

[8] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", *Comm. of the ACM*, Vol 24, pp 381-395, 1981.

[9] K.S. Fu, *Syntactic Pattern Recognition and Applications*, Prentice Hall, 1981.

[10] C.E Guo, S.C. Zhu and Y.N. Wu, "A mathematical theory of primal sketch and sketchability", *Proc. Int'l Conf. Computer Vision*, 2003.

[11] C.E. Guo, S.C. Zhu and Y.N. Wu, "Primal sketch: integrating texture and structure", *Computer Vision and Image Understanding*, vol. 106, issue 1, 5-19, April, 2007.

[12] A. Hanson and E. Riseman, "Visions: a computer system for interpreting scenes", in *Computer Vision Systems*, 1978.

[13] K. Huang, W. Hong and Y. Ma, "Symmetry-based photo editing", *IEEE Workshop on Higher-Level Knowledge in 3D Modeling & Motion Analysis*, Nice, 2003.

[14] V. Hwang and T. Matsuyama,"SIGMA: A Framework for Image Understanding: Integration of Bottom-Up and Top-Down Analyses", *IJCAI85* 36(2/3), pp. 908-915, 1985.

[15] V. Hwang, L.S. Davis and T. Matsuyama,"Hypothesis Integration in Image Understanding Systems", *CVGIP* 36(2/3), pp. 321-371, 1986.

[16] S. Ioffe and D. Forsyth, "Probabilistic Methods for Finding People", *Int'l J. Computer Vision*, 43(1), pp. 45-68, 2001.

[17] Y. Jin and S. Geman, "Context and hierarchy in a probabilistic image model, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, New York, June, 2006.

[18] D. Lagunovsky and S. Ablameyko, "Straight-line-based primitive extraction in grey-scale object recognition", *Pattern Recognition Letters*, 20(10):10051014, October 1999.

[19] A. Levinshtein, C. Sminchisescu, S.J. Dickinson, "Learning Hierarchical Shape Models from Examples", *EMMCVPR*, 2005.

[20] C. Lin and R. Nevatia, "Building detection and description from a single intensity image", *Computer Vision and Image Understanding*, 72(2):101121, 1998.

[21] S. Mallat, and Z. Zhang, "Matching pursuit with time-frequency dictionaries", *IEEE Trans. on Signal*

Processing, vol. 41, no. 12, 3397-3415, 1993.

[22] W. Mann and T. Binford, “Successor: Interpretation Overview And Constraint System”, *IUW 96* pp. 1505-1518, 1996.

[23] D. McKeown, W. Harvey, L. Wixson, “Automating knowledge acquisition for aerial image interpretation”, *Computer Vision, Graphics, and Image Processing* 46(1), pp. 37-81, 1989.

[24] S. Munder and D. Gavrila, “An Experimental Study on Pedestrian Classification”, *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11), 2006.

[25] Y. Ohta, T. Kanade, and T. Sakai, ”An analysis system for scenes containing objects with substructures”, *Proc. 4th Int'l Joint Conf. on Pattern Recognition*, pp.752-754, Kyoto, 1978.

[26] Y. Ohta, *Knowledge-based interpretation of outdoor natural color scenes*, Pitman, 1985.

[27] I. Pollak, J.M. Siskind, M.P. Harper, and C.A. Bouman, “Parameter Estimation for Spatial Random Trees Using the EM Algorithm”, *Proc. Int'l Conf.e on Image Processing*, 2003.

[28] I. Pollak, J.M. Siskind, M.P. Harper, and C.A. Bouman, “Modeling and Estimation of Spatial Random Trees with Application to Image Classification”, *Proc. Int'l Conf. on Acoustics, Speech, and Signal Processing*, 2003.

[29] J. Porway, Z.Y. Yao, and S.C. Zhu, ”Modeling and learning object categories from small sample sets”, *UCLA Statistics Dept Technical Report*, 2007.

[30] J. Rekers and A. Schürr, “Defining and parsing visual languages with layered graph grammars”, *J. Visual Language and Computing*, Sept. 1996.

[31] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker, ”Shock graphs and shape matching,” *IJCV*, 35(1), 13-32, 1999.

[32] J.M. Siskind, J. Sherman, I. Pollak, M.P. Harper, C.A. Bouman, “Spatial random tree grammars for modeling hierarchal structure in images”, *IEEE Trans. on PAMI*, (to appear).

[33] W.-B. Tao, J.-W. Tian, and J. Liu, “A new approach to extract rectangle building from aerial urban images”, *Int'l Conference on Signal Processing*, 143146, 2002.

[34] Z.W. Tu, X.R. Chen, A.L. Yuille, and S.C. Zhu, ”Image parsing: unifying segmentation, detection and recognition,” *Int'l J. of Computer Vision*, 63(2), 113-140, 2005.

[35] W. Wang, I. Pollak, T.-S. Wong, C.A. Bouman, M.P. Harper, and J.M. Siskind, ”Hierarchical stochastic image grammars for classification and segmentation”, *IEEE Trans. on Image Processing*, 15(10):3033-3052, October 2006.

[36] T.F. Wu, G.S. Xia, and S.C. Zhu, ”Compositional boosting for computing hierarchical image structures”, *Proc. IEEE. Conf. on Computer Vision and Pattern Recognition* , June, 2007.

[37] F.C You and K.S. Fu, “A syntactic approach to shape recognition using attributed grammars”, *IEEE Trans. on SMC*, vol. 9, pp. 334-345, 1979.

[38] F.C You and K.S. Fu, “Attributed grammar: A tool for combining syntatic and statistical approaches to pattern recognition”, *IEEE Trans. on SMC*, vol. 10, pp. 873-885, 1980.

[39] W. Zhang and J. Kosecka, “Extraction, matching and pose recovery based on dominant rectangular structures”, *IEEE Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, Nice, 2003.

[40] S.C. Zhu, R. Zhang, and Z. W. Tu. “Integrating top-down/bottom-up for object recognition by DDM-CMC”, *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000.

[41] S.C. Zhu and A.L. Yuille, ”FORMS: A Flexible Object Recognition and Modeling System,” *Int’l J. Computer Vision* 20(3), pp.187-212, 1996.

[42] S. C. Zhu, Y. N. Wu, and D. Mumford,“Minimax entropy principle and its application to texture modeling”,*Neural Computation*, 9:16271660, 1997.

[43] S.C. Zhu and D. Mumford, ”A Stochastic Grammar of Images”, *Foundations and Trends in Computer Graphics and Vision*, Vol.2, No.4, pp 259-362, 2006.

[44] Y. Zhu, B. Carragher, F. Mouche, and C. Potter, “Automatic particle detection through efficient hough transforms”, *IEEE Transactions on Medical Imaging*, 22(9): 10531062, September 2003.